



		Project title: Development of sensor-based Citizens' Observatory Community for improving quality of life in cities
		Acronym: CITI-SENSE Grant Agreement No: 308524
		EU FP7- ENV-2012 Collaborative project

Deliverable D7.6 - Part 2: Platform Operation

D7.6: CITI-SENSE Platform and architecture Version 4 - Part 2: Operation

Work Package 7

Date: 30.11.2016

Version: 4.10

Leading Beneficiary:	SINTEF, Snowflake
Editor(s):	Arne J. Berre (SINTEF), Ballal Joglekar (Snowflake)
Author(s) (alphabetically):	Arne J. Berre (SINTEF), Ballal Joglekar (Snowflake), Bent Morten Dale (OBEO), Neil Devlin (Geotech/Envirologger), Dejan Drajić (DunavNet), Alberto Fernandez (Sensing&Control), Nicolas Ferry (SINTEF), Nicolas Ferry (SINTEF), Sungchul Hong (KICT), Mike Kellaway (Atmospheric), Seonho Kim (Saltlux), Daniele Miorandi (U-Hopper), Mirjam Fredriksen (NILU), Igone Garcia Perez (Tecnalia), Maja Pokric (Dunavnet), Dumitru Roman (SINTEF), Miha Smolnikar (JSI), Andrei Tamlin (U-Hopper), Matevž Vučnik (JSI)
Dissemination level:	Public



Versioning and contribution history

Version	Date issued	Description	Contributors
3.10	30.11.2015	Final version 3.10 – (D7.5 Part 2)	Arne J. Berre
4.01	14.03.2016	Initial version 4.01	Arne J. Berre
4.02	13.06.2016	Update on Sensor Services	Arne J. Berre
4.03	12.09.2016	Update on Sensor interfaces, App architecture, Portal architecture, Widgets and Survey/Questionnaire	Leo Santiago, Mirjam Fredriksen, Zvezdana Knežević, Andrei Taminin
4.04	27.09.2016	Update on Interoperability aspects and GEOSS	Arne J. Berre
4.05	29.09.2016	Update on Data and Product Services	Ballal Joglekar
4.10	30.11.2016	Update of reviewer's comments	Arne J. Berre

Peer review summary

Internal review 1			
Reviewer	Mirjam Fredriksen (NILU)		
Received for review	14.11.2016	Date of review	29.11.2016

Internal review 2			
Reviewer	Leonardo Santiago (Ateknea)		
Received for review	14.11.2016	Date of review	29.11.2016



Executive Summary

This D7.6 part 2 Operation deliverable provides the final version 4.0 of the operational CITI-SENSE Platform and Architecture. The emphasis of this document is on describing the data flows from a range of sensor data and app providers, through the centralised SEDS Data Platform and to the dissemination of the sensor and app data-to-data consumers, typically through visualization widgets.

The D7.6 part 2 starts with a detailed description of some of the core operational components of the CITI-SENSE platform:

It contains a description to “Access to CITI-SENSE Data Web Services” on the usage of the CITI-SENSE Data access server based on the WFS-T standard, showing the usage of the WFS-T server, in particular for sensors data access, observations & measurements and observations & questionnaires. “Access to SensApp server” contains a description of the access to the SensApp server for sensor data storage support in particular for mobile sensors, or for situations when no pre-existing sensor platform exists. “Visualisation Components” contains a description for the use of visualisation components in particular for HTML5-based measurement widgets.

This is followed by detailed operational platform architectures for the following data providers form sensor platforms and apps:

- Annex A: GeoTech - AQMesh
- Annex B: Atmospheric Sensors
- Annex C: OBEO MMA Radon and CO₂ sensor
- Annex D: PSP with LEO
- Annex E: LEO with ENCONTROL
- Annex F: SENSE-IT-NOW APP
- Annex G: SENSE-IT-NOW Acoustic Service
- Annex H: CityAir App
- Annex I: Vesna Personal Air Quality Pack
- Annex J: DunavNet EB700++ Mobile Sensor Device
- Annex K: South Korea Example System

The related D7.6 part 1 document describes the specification of the overall CITI-SENSE Architecture and platform, while this D7.6 part 2 document describes the more operational details of the services, as well as having annexes with practical examples of the usage of the CITI-SENSE Architecture and platform. The D7.6 part 3 document contains the description of the platform and architecture from a Citizens’ Observatory Toolbox developer’s perspective.



Table of contents

EXECUTIVE SUMMARY	3
TABLE OF CONTENTS	4
ABBREVIATIONS AND ACRONYMS	11
1 INTRODUCTION.....	12
2 ACCESS TO CITI-SENSE DATA WEB SERVICES.....	13
2.1 DATA INGESTION SERVICES.....	13
2.2 DATA PUBLICATION SERVICES	13
2.2.1 Publication Scenarios	14
2.3 OVERVIEW DATA PUBLICATION SERVICES.....	15
3 ACCESS TO SENSAPP SERVER.....	21
3.1 SETUP.....	21
3.2 INSTALLING SENSAPP DEPENDENCIES.....	21
3.3 DEPLOYING SENSAPP IN THE DEVELOPMENT ENVIRONMENT	21
3.4 USING SENSAPP	21
3.4.1 Registering sensors:.....	21
3.4.2 Retrieve Sensor Description.....	22
3.4.3 Add metadata to registered sensors.....	23
3.4.4 Create a composite sensor	23
3.4.5 Push data inside SensApp	23
3.4.6 Retrieving sensor's data	24
3.4.7 Retrieving data in CSV format.....	24
3.4.8 SensApp Admin web interface	25
3.4.8.1 Visualise the list of sensors and composite sensors	25
3.4.8.2 Create, update, delete sensors and composite sensors.....	27
3.4.8.3 Observe the description of a sensor or a composite sensor	28
3.4.8.4 Retrieve the JSON description of a sensors or a composite sensor	29
3.4.8.5 Visualise sensor data in the forms of a raw data table or charts	29
3.4.8.6 Export sensors data in SenML	31
3.4.8.7 Export composite sensor data in the form of CSV	31
3.4.9 SensAppHelper.....	32
4 VISUALISATION COMPONENTS	34
4.1 WIDGET FRAMEWORK FOR MEASUREMENTS BY DUNAVNET.....	34
4.1.1 Installation	34
4.1.2 Configuring data.....	34
4.1.3 Initialization.....	35
4.1.4 Get data.....	35
4.1.5 Draw basic widget.....	35
4.1.6 Positioning and Sizing.....	36
4.1.7 Process data	36
4.1.8 Example code	37
4.2 WIDGET FRAMEWORK FOR QUESTIONAIRES BY U-HOPPER	39
4.3 CITI-SENSE SENSORS VISUALIZATION WIDGETS	41
4.3.1 Tabular visualization of sensors metadata.....	42
4.3.2 Map visualization of sensors metadata	42
4.3.3 Tabular visualization of Air Quality Indexes of sensors.....	43
4.3.4 Map visualization of Air Quality Indexes of sensors	44
4.4 CITI-SENSE PERCEPTION ACQUISITION CAMPAIGNS.....	45
5 SUMMARY	47
6 ANNEX A: GEOTECH WITH AQMESH	48



6.1	INTRODUCTION TO USAGE CONTEXT.....	48
6.1.1	Introduction	48
6.1.2	Reference to CITI-SENSE Pilot work packages	48
6.1.3	Reference to CITI-SENSE Locations	48
6.2	ARCHITECTURE AND INTERFACES USED	48
	Data Collection (AQMesh).....	49
	Data Storage (AQMesh)	49
	Data Ingest	50
	Data Storage (SEDS)	50
	Data Publication.....	50
	Data Consumption	50
6.3	DATA (VOLUME, VELOCITY) BEING STORED AND RETRIEVED	50
6.3.1	Data exchange between AQMesh and the SEDS Platform.....	50
6.3.2	Data volumes.....	51
6.4	USER APPLICATIONS/APPS/PROCESSING/VISUALISATIONS.....	51
6.5	SUGGESTED IMPROVEMENTS AND PLANS FOR NEXT PHASE(S).....	53
7	ANNEX B: ATMOSPHERIC SENSORS	54
7.1	INTRODUCTION TO USAGE CONTEXT.....	54
7.1.1	Introduction	54
7.1.2	Reference to CITI-SENSE Pilot work packages	54
7.1.3	Reference to CITI-SENSE Locations	54
7.2	ARCHITECTURE AND INTERFACES USED	55
7.2.1	Data collection (ATMOS).....	55
7.2.2	Data storage (ATMOS).....	55
7.2.3	Data ingest (SEDS)	56
7.2.4	Data Storage (SEDS).....	57
7.2.5	Data publication (SEDS).....	58
7.2.6	Data consumption.....	58
7.3	DATA (VOLUME, VELOCITY) BEING STORED AND RETRIEVED	58
7.4	SUGGESTED IMPROVEMENTS AND PLANS FOR NEXT PHASE(S).....	58
8	ANNEX C: OBEO MMA RADON AND CO₂ SENSOR	59
8.1	INTRODUCTION TO USAGE CONTEXT.....	59
8.1.1	Introduction	59
8.1.2	Reference to CITI-SENSE Pilot workpackages	61
8.1.3	Reference to CITI-SENSE Locations	61
8.2	ARCHITECTURE AND INTERFACES USED	61
8.2.1	OBEO server side infrastructure	62
8.3	DATA (VOLUME, VELOCITY) BEING STORED AND RETRIEVED	62
8.4	USER APPLICATIONS/APPS/PROCESSING/VISUALIZATIONS	62
8.4.1	Suggested improvements and plans for the final phase	64
9	ANNEX D: PERSONAL SENSOR PLATFORM WITH LEO	65
9.1	INTRODUCTION	65
9.2	CITISENSE PSP.....	65
9.2.1	Reference to CITI-SENSE Pilot workpackages	65
9.2.2	Reference to CITI-SENSE Locations	65
9.3	ARCHITECTURE AND INTERFACES USED	65
9.4	DATA (VOLUME, VELOCITY) BEING STORED AND RETRIEVED	67
9.5	USER APPLICATIONS/APPS/PROCESSING/VISUALISATIONS.....	68
9.6	SUGGESTED IMPROVEMENTS AND PLANS FOR NEXT PHASE(S).....	69
9.7	GPS/ACC SOLUTION – SENSING & CONTROL (WITH ATEKNEA)	70
9.7.1	Introduction to usage context.....	70
9.7.2	Architecture and interfaces used.....	70
9.7.3	Data (volume, velocity) being stored and retrieved	71
9.7.4	User applications/apps/processing/visualisations.....	71
9.7.5	Suggested improvements and conclusions	72



9.8	LEO SENSOR DATA FLOW.....	72
9.8.1	ExpoApp2.....	74
9.8.2	How does ExpoApp work?	74
9.8.3	How to Start a New Collection	74
9.9	SUGGESTED IMPROVEMENTS AND CONCLUSIONS.....	75
10	ANNEX E: LEO AIR QUALITY MEASUREMENT INTEGRATION WITH ENCONTROL.....	76
10.1	INTRODUCTION TO USAGE CONTEXT.....	76
10.1.1	LEO device and Exposomics app	76
10.1.2	enControl™	76
10.2	ARCHITECTURE AND INTERFACES USED	78
10.3	USER APPLICATIONS/APPS/PROCESSING/VISUALISATIONS.....	80
10.4	SUGGESTED IMPROVEMENTS AND CONCLUSIONS	84
11	ANNEX F: SENSE-IT-NOW MULTI-PLATFORM SMARTPHONE APP	85
11.1	INTRODUCTION TO USAGE CONTEXT.....	85
11.1.1	Introduction	85
11.1.2	Reference to CITI-SENSE Pilot workpackages	85
11.1.3	Reference to CITI-SENSE Locations	85
11.2	ARCHITECTURE AND INTERFACES USED	85
11.2.1	Data flow	85
11.2.2	Architecture	87
11.2.3	Framework Architecture	87
11.3	DATA (VOLUME, VELOCITY) BEING STORED AND RETRIEVED	88
11.4	USER APPLICATIONS/APPS/PROCESSING/VISUALISATIONS.....	89
11.4.1	SENSE-IT-NOW	89
11.4.2	SENSE-IT-NOW – WP2 – Urban Quality based on LEO sensor.....	92
11.5	SUGGESTED IMPROVEMENTS AND CONCLUSIONS	95
11.6	SERVICE FOR CALCULATING PHYSICAL MOVEMENT	95
11.6.1	Introduction to usage context.....	95
11.6.2	Architecture and interfaces used.....	95
11.6.3	Data (volume, velocity) being stored and retrieved.....	95
11.6.4	User applications/apps/processing/visualizations.....	96
11.6.5	Suggested improvements and plans for next phase(s).....	96
11.7	PET CALCULATION	96
11.7.1	Introduction to usage context.....	96
11.7.2	Architecture and interfaces used.....	96
11.7.3	Data (volume, velocity) being stored and retrieved.....	96
11.7.4	User applications/apps/processing/visualizations.....	96
11.7.5	Suggested improvements and conclusions	96
12	ANNEX G: SENSE-IT-NOW: ACOUSTIC SERVICE	97
12.1	INTRODUCTION TO USAGE CONTEXT.....	97
12.1.1	Introduction	97
12.1.2	Reference to CITI-SENSE Pilot workpackages	97
12.1.3	Reference to CITI-SENSE Locations	97
12.2	ARCHITECTURE AND INTERFACES USED	97
12.3	DATA (VOLUME, VELOCITY) BEING STORED AND RETRIEVED	98
12.4	USER APPLICATIONS/APPS/PROCESSING/VISUALISATIONS.....	98
12.5	SUGGESTED IMPROVEMENTS AND CONCLUSIONS	100
13	ANNEX H: CITYAIR	102
13.1	INTRODUCTION TO USAGE CONTEXT.....	102
13.1.1	Introduction	102
13.1.2	Reference to CITI-SENSE Pilot workpackages	102
13.1.3	Reference to CITI-SENSE Locations	102
13.2	ARCHITECTURE AND INTERFACES USED	102
13.2.1	Data flow	102



13.2.2	Architecture	103
13.2.3	Framework Architecture	103
13.3	DATA (VOLUME, VELOCITY) BEING STORED AND RETRIEVED	104
13.4	USER APPLICATIONS/APPS/PROCESSING/VISUALISATIONS.....	109
13.4.1	CityAir – smart phone app	109
13.4.2	Visualization services.....	110
13.5	SUGGESTED IMPROVEMENTS AND CONCLUSIONS	112
13.6	USER STEPS FOR OPERATION – REFERENCE NECESSARY FOR USAGE	113
13.6.1	Installation	113
13.7	USAGE.....	115
14	ANNEX I: VESNA PERSONAL AIR QUALITY PACK.....	122
14.1	INTRODUCTION TO USAGE CONTEXT.....	122
14.1.1	Introduction	122
14.1.2	Reference to CITI-SENSE Pilot workpackages	123
14.1.3	Reference to CITI-SENSE Locations	123
14.2	ARCHITECTURE AND INTERFACES USED	123
14.2.1	Vesna WiFi system.....	123
14.2.2	VESNA Bluetooth Low Energy system	126
14.2.3	JSI server side infrastructure.....	127
14.3	DATA (VOLUME, VELOCITY) BEING STORED AND RETRIEVED	128
14.3.1	VESNA AQA v1.0	128
14.3.2	VESNA PAQ v2.0.....	129
14.4	USER APPLICATIONS/APPS/PROCESSING/VISUALISATIONS.....	129
14.4.1	Suggested improvements and conclusions	130
15	ANNEX J: DUNAVNET EB700++ MOBILE SENSOR DEVICE	131
15.1	INTRODUCTION TO USAGE CONTEXT.....	131
15.1.1	Introduction	131
15.1.2	Reference to CITI-SENSE Pilot workpackages	133
15.1.3	Reference to CITI-SENSE Locations	133
15.2	ARCHITECTURE AND INTERFACES USED	133
15.2.1	DNET server side infrastructure	136
15.3	DATA (VOLUME, VELOCITY) BEING STORED AND RETRIEVED	137
15.4	USER APPLICATIONS/APPS/PROCESSING/VISUALISATIONS.....	137
16	ANNEX K: SOUTH KOREA UPA EXAMPLE SYSTEM	140
16.1.1	System Architecture	140
16.1.2	Software Configuration	140
16.1.3	Sensor Data Collection.....	141
16.1.4	System Functions.....	150
16.2	EXAMPLES OF AIR POLLUTION WARNING SERVICE	153

Index of figures

Figure 3-1	Using the RESTConsole to POST requests.....	22
Figure 3-2	SensApp Admin GUI main page	26
Figure 3-3	Change the number of records per page	26
Figure 3-4	listing all composite sensors	27
Figure 3-5	Composite sensors administration.....	27
Figure 3-6	Sensor creation form	28
Figure 3-7	Managing sensors	28
Figure 3-8	Sensor description	28



Figure 3-9 retrieve the raw description of a sensor	29
Figure 3-10 List of sensors whose data can be visualized	29
Figure 3-11 Filtering sensor data	29
Figure 3-12 View data from a specific sensor	30
Figure 3-13 Displaying raw data	30
Figure 3-14 Displaying data as a chart.....	30
Figure 3-15 Export sensor data as SenML.....	31
Figure 3-16 Generate CSV file with all measurements from of composite sensor.....	31
Figure 3-17 Download the generated csv file	32
Figure 4-1 Example of map with sensor data	43
Figure 4-2 Example of map with air quality index.....	45
Figure 6-1 Data Flows for the CITI-SENSE Geotech AQMesh Sensor architecture.....	48
Figure 6-2 AQMesh station	49
Figure 6-3 AQMesh data flow	51
Figure 6-4 Screenshot of the AQMesh.net web portal.....	52
Figure 6-5 Displaying sensor data in a graph.....	53
Figure 6-6 Exporting data	53
Figure 7-1 Atmospheric Sensors node.....	54
Figure 7-2 Data Flows for the CITI-SENSE Atmospheric Sensor architecture	55
Figure 8-1 Picture of OBEO MMA.....	60
Figure 8-2 OBEO MMA Radon sensor installation process and connectors	61
Figure 8-3 Data Flows for the OBEO architecture	62
Figure 8-4 Visualisation of Radon and Temperature data	63
Figure 9-1 Data Flows for the CITI-SENSE PSP App and ExpoApp	66
Figure 9-2 User Interface - monitoring	67
Figure 9-3 User Interface - Connected.....	68
Figure 9-4: last captured sensor values	69
Figure 9-5: Configuration	71
Figure 9-6: LEO Device	72
Figure 9-7: Dataflow for LEO device	73
Figure 9-8 ExpoApp2 Main screen.....	74
Figure 9-10 ExpoApp2 configuration interface	75
Figure 10-1: Smart home architecture.....	77
Figure 10-2: enControl™ Dashboard	78
Figure 10-3 Data Flow Architecture for the LEO device, SEDS and enControl™	79
Figure 10-4 enControl™ user interface with LEO device measurement in area called “DefaultArea”..	81
Figure 10-5: Nitrogen Oxide (NO) raw data aggregated (avg) on 5 minute interval.....	82
Figure 10-6: Nitrogen Dioxide (NO ₂) raw data aggregated (avg) on 5 minute interval with AQI	82



Figure 10-7: Ozone (O ₃) raw data aggregated (avg) on 5 minute interval with AQI.....	83
Figure 10-8: Ozone (O ₃) raw data aggregated (avg) on 30 minute interval with AQI on background. .	83
Figure 11-1 Data Flows for the SENSE-IT-NOW architecture	86
Figure 11-2 SENSE-IT-NOW Interface architecture	87
Figure 11-3 PhoneGap/Cordova SmartPhone App architecture.....	88
Figure 11-4 SENSE-IT-NOW Settings page and visualisation	89
Figure 11-5 SENSE-IT-NOW Selection page.....	89
Figure 11-6 SENSE-IT-NOW Observation pages	90
Figure 11-7 SENSE-IT-NOW Noise page	90
Figure 11-8 SENSE-IT-NOW Questionnaire pages	91
Figure 11-9 SENSE-IT-NOW Thermal Index and Acoustic Comfort.....	92
Figure 11-10 SENSE-IT-NOW with ExpoApp.....	93
Figure 11-11 SENSE-IT-NOW Data selection pages	93
Figure 11-12 SENSE-IT-NOW Air Quality data selection pages	94
Figure 11-13 SENSE-IT-NOW Selecting graph of Air Quality Index.....	94
Figure 11-14 SENSE-IT-NOW map widget with position based data	95
Figure 12-1 <i>Acoustic Service – Data flow</i>	98
Figure 12-2 Citisense App User Interface	99
Figure 12-3 Event Notification Questionnaire.....	100
Figure 13-1 Data Flows for the CityAir architecture.....	103
Figure 13-2 CityAir Architecture	103
Figure 13-3 Part of CITI-SENSE Data model used by CityAir	108
Figure 13-4 CityAir App Interface	110
Figure 13-5 CityAir Map Visualisation	111
Figure 13-6 CityAir Map on local scale.....	112
Figure 13-7 CityAir description from Google Play store.....	114
Figure 13-8 CityAir description from Apple iTunes App Store	115
Figure 14-1 The VESNA v2.0 Personal Air Quality (PAQ) Sensor Pack	123
Figure 14-2 VESNA Air Quality Monitoring System Architecture.....	124
Figure 14-3 Data Flows for the VESNA PAQ architecture.....	124
Figure 14-4 Resource access protocol.....	125
Figure 14-5 Error handling	126
Figure 14-6 JSON Data Structure used by AQ app.....	126
Figure 14-7 BLE packet	127
Figure 14-8 VESNA BLE system	127
Figure 14-9 WFS-T support flowchart	128
Figure 14-10 JSI AQ App v2.0	129
Figure 15-1 The DNET EB700++	132



Figure 15-2 The DNET EB800	132
Figure 15-3 Data Flows for the CITI-SENSE DNET architecture.....	133
Figure 15-4 EB700++ System Architecture	134
Figure 15-5 Server side infrastructure	137
Figure 15-6 Web portal	139
Figure 16-1 Architecture of the Seoul Example System	140
Figure 16-2 The web-based pilot system under development.....	141
Figure 16-3 Data collection method for Seoul	146
Figure 16-4 Physical Sensor Interface	146
Figure 16-5 (Left) Sensor for temperature and humidity, and (right) gateway	148
Figure 16-6 Data expression of physical sensor	149
Figure 16-7 GUI of the pilot system under development	151
Figure 16-8 Start-up screen of UPA service.....	153
Figure 16-9 Display of real-time concentration of contaminated materials	154
Figure 16-10 Status of air pollution forecast/alert.....	155
Figure 16-11 Action tips for each air pollution material.....	155
Figure 16-12 Yearly Average Air Pollution Level per District.....	156
Figure 16-13 Variation trend of contaminated material concentration per district.....	156
Figure 16-14 Status of forecasts/alerts per year	157
Figure 16-15 Status of user reactions	158
Figure 16-16 Warning of increase in pollution material value.....	158
Figure 16-17 Hospital location and path guide	159
Figure 16-18 Recommendation of best district for home movement	159

Index of Tables

Table 0-1 Abbreviations and Acronyms	11
Table 3-1 Example of data aggregation	25
Table 8-1 Data characteristics of OBEO MMA.....	60
Table 16-1 Data representation of SenML data types.....	147
Table 16-2 Service names with cycle and parameters.....	149
Table 16-3 Services from the Pilot System	152
Table 16-4 Functions in Seoul Example Air quality Information System.....	152

Abbreviations and Acronyms

Table 0-1 Abbreviations and Acronyms

Abbreviation / Acronym	Description
CAQI	Common Air Quality Index
DAE	Digital Agenda for Europe
EIF	European Interoperability Framework
EIS	European Interoperability Strategy
FP7	Seventh Research Framework Programme of the European Union
GEOSS	Global Earth Observation System of Systems
GFM	General Feature Model
GML	Geography Markup Language
HTML	Hyper Text Markup Language
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
INSPIRE	Infrastructure for Spatial Information in the European Community
IoT	Internet of Things
ISO	International Organization for Standardization
LOD	Linked Open Data
O&M	Observations and Measurements
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
RDF	Resource Description Framework
REST	Representational State Transfer
SaaS	Software as a Service
SDI	Spatial Data Infrastructure
SOA	Service-oriented Architecture
SPARQL	SPARQL Protocol And RDF Query Language
SPARQL	SPARQL Protocol And RDF Query Language
SSNO	Semantic Sensor Networks Ontology
SSNO	Semantic Sensor Networks Ontology
URI	Uniform Resource Identifier
VGI	Volunteered Geographic Information



1 Introduction

This deliverable, D7.6 “Platform and Architecture – version 4.0” part 2 – contains the operational aspects of the version 4.0 of the specification and description for the CITI-SENSE architecture and platform.

The updates in D7.6 compared to D7.5 are in particular an update of details of the data access in the chapters on data products and services and the final version of the CITI-SENSE data model.

Further the D7.6 has been split into three parts (with part 3 as a new part for D7.6), with part 1 focusing on the overall architecture and specifications, and the new part 3 focusing on the components of the platform suitable for a new Citizens’ Observatory – as a Citizens’ Observatory Toolbox – from a developer’s view.

This part 2 is focusing on the operational aspects including more details of the various sensor platforms and app usages of the CITI-SENSE architecture and platform, as follows:

Chapter 2 “Access to CITI-SENSE Data Web Services” contains a description of the usage of the CITI-SENSE Data access server based on the WFS-T standard, showing the usage of the WFS-T server, in particular for sensors data access, observations & measurements and observations & questionnaires.

Chapter 3 “Access to SensApp server” contains a description of the access to the SensApp server for sensor data storage support in particular for mobile sensors, or for situations when no pre-existing sensor platform exists.

Chapter 4 “Visualisation Components” contains a description for the use of visualisation components in particular for HTML5-based measurement widgets.

Chapter 5 summarises the content of this Platform and Architecture part 2 deliverable.

The following annexes use a common structure for the description of the architecture of the various sensor platforms and apps: Introduction to usage context, Architecture and interfaces used, Data (Volume, Velocity) being stored and retrieved, user applications/visualisations and further plans.

This structure is being applied for the following descriptions:

Annex A "GeoTech - AQMesh"

Annex B "Atmospheric Sensors"

Annex C "OBEO MMA Radon and CO₂ sensor"

Annex D "PSP with LEO"

Annex E "LEO with ENCONTROL"

Annex F "SENSE-IT-NOW APP"

Annex G "SENSE-IT-NOW Acoustic Service"

Annex H "CityAir App"

Annex I "Vesna Personal Air Quality Pack"

Annex J "DunavNet EB700++ Mobile Sensor Device"

Annex K "South Korea Example System"



2 Access to CITI-SENSE Data Web Services

On a high level, the CITI-SENSE SEDS Platform provides two main interfaces with external actors: one for Data Ingestion and one for Data Publication/Data Access. Both interfaces have been realised as a range of web services that can be accessed via standard HTTP protocols.

2.1 Data Ingestion Services

The data ingestion services are web services that have been deployed on Amazon AWS. Currently it contains a single WFS endpoint which data providers can use to a) register their sensors, and b) upload their sensor observations. This interface is realised by implementing an ISO 19142/OGC Web Feature Service (WFS).

The WFS endpoint is located at <https://prod.citisense.snowflakesoftware.com/wfst> and supports the following WFS operations:

- GetCapabilities: provides information about the functionality that the WFS supports;
- DescribeFeatureType: provides a list of feature types that the WFS offers;
- GetFeature: allows the request and response of data via WFS request.

For security reasons, access to the Data Ingestions services is prohibited for clients without the correct credentials. This protects the SEDS Platform from any unauthorised ingestion of data.

The Data Ingestion services primarily consist of a WFS web service that allows client to post data ingest requests against. However, a small number of data providers in CITI-SENSE cannot provide this functionality. Therefore, an additional component has been added to the Data Ingestion services. This component connects periodically to a secure FTP server at the data provider and copies sensor data to the SEDS Platform. Here the datasets are converted into HTTP POST requests which are made against the main WFS Data Ingestion web service.

2.2 Data Publication Services

Two types of web interfaces have been realised on the Amazon Cloud AWS which allow users to get access to the sensor and sensor observation data stored in the SEDS Platform:

1. WFS
2. REST

Web Feature Service

A number of WFS service endpoints are available to end-users. After successfully connecting a WFS web service, an end-user can use standard WFS queries encoded using the OGC Filter Encoding Specification (FES)¹.

REST

Whilst the WFS web service provides a rich and comprehensive specification to interact with a web service, some end-users require a simpler, more lightweight specification for example to build mobile applications.

To satisfy the requirements of this user group in CITI-SENSE a number of REST web services have been deployed alongside the WFS web services. The RESTful webservices are capable of serving data formatted in either in XML or JSON.

¹ OGC 09-026r1 - OpenGIS Filter Encoding 2.0 Encoding Standard, 2010, Open Geo Spatial Consortium



Following the requirements of Work Packages 2 and 3, specific REST services have been implemented. Each of these web services fulfils a particular query pattern, such as the all observations for a particular sensor between two dates.

2.2.1 Publication Scenarios

In close cooperation with Work Package 5, a number of common query patterns (i.e. publication scenarios) have been identified and implemented. These scenarios form a logical consequence from the user specifications detailed in D7.5-Part 3.

These scenarios helped in focussing the development of a specific set of publication services that were required by data consumers.

The following scenarios have been agreed:

- **Scenario 1:** Publish all the observations for a specific sensor device.
- **Scenario 2:** Publish all observations for a specific sensor device at a specific point in time ("*snapshot*").
- **Scenario 3:** Publish all observations for a specific sensor device in a given time period.
- **Scenario 4:** Publish the latest observation for a specific sensor device.
- **Scenario 5:** Publish the latest observation for a specific sensor device for a specific pollutant.

All these scenarios have been implemented and tested for the WFS and the REST web services.

A sixth publication scenario was later agreed for the WFS service. This was achieved by creating a new WFS endpoint. This scenario was particularly useful for the widgets in visualising, on a map, the latest observation for all the sensors within a particular city. The endpoint can be queried spatially or using other filters such as by the city name. The gml:name element was used to publish the location names and could be queried using a simple PropertyIsLike filter (see section 2.2.2 below). The bounding box spatial filter could query the observations based on the spatial location. The spatial filter is limited to the observations that contain a geometry in the database.

Scenario 6: Publish the latest observation for all the sensors devices for a particular location.**Pre-cached bounding box spatial queries**

Web Feature Service (WFS) allows users to retrieve features stored in the backend database using the GetFeature request. The GetFeature request retrieves all the features of particular feature type or feature types as specified in the GetFeature request.

Most of the times users are not interested in getting all the features but are interested in getting the features of interest. These can be features that fall within a particular area of interest or features that meet certain criteria such as observations that have a 'finishtime' before a particular date.

A query for features within an area of interest is typically performed using a bounding box query. A bounding box is typically a box using user defined co-ordinates. The co-ordinates of the lower left corner and the upper right corner are passed to the bbox parameter.

Scenario 6 for the WFS returns the latest observations for each sensor within a particular city. Widgets and apps that would sit on top of the SEDS platform would query data from the WFS and present the data in a graphical way on a map. The users of the widgets could choose a city from a list



of cities. Once a city is selected, the widget fires off a BBOX queries to the WFS. The bounding box covers the extent of the city in question and returns the latest observation for all the sensors within the bounding box. As more and more observations were captured and the database started growing, and resulting in poor query performance which was a concern for the widgets.

Response caching is a technique for speeding up the response times for frequently used requests. The server side caching approach has been used for the CITI-SENSE project because it yielded substantial performance benefits. It helped reduce the load on the WFS server. 8 WFS requests have their responses cached. These are bounding box requests for 8 pilot cities that retrieve the latest observations from each sensor in those cities. These bounding box requests are the most commonly used requests. The widgets generate these WFS requests when users select the name of a city, to view data for, within the widget.

The response for each WFS request is saved as an XML file and these files are uploaded to an Amazon S3 bucket. The Amazon S3 bucket acts as the cache. Each of the 8 WFS requests gets redirected to the corresponding XML file on Amazon S3 bucket using Nginx configuration. The lifespan for the cache is 30 minutes, i.e. the XML files on the Amazon S3 bucket are refreshed every 30 mins.

2.3 Overview Data Publication services

The tables below provide samples of the REST and WFS request patterns for the 5 scenarios defined in the previous paragraph.

Note: Replace {sensor-id} with an actual sensor id.

REST services

Scenario 1: Give me all observations for a specific sensor-id
CSV:
https://prod.citisense.snowflakesoftware.com/csv/sensor/allobservations?sensorid={sensor-id}
JSON:
https://prod.citisense.snowflakesoftware.com/json/sensor/allobservations?sensorid={sensor-id}

Scenario 2: Give me all observations for a specific sensor-id at a specific point in time (time instant)
CSV:
https://prod.citisense.snowflakesoftware.com/csv/sensor/observationfinishtime?sensorid={sensor-id}&finishtime=2001-12-18T10:00:00.000
JSON:
https://prod.citisense.snowflakesoftware.com/json/sensor/observationfinishtime?sensorid={sensor-id}&finishtime=2001-12-18T10:00:00.000


Scenario 3: Give me all observations for a specific sensor-id in a specific time period
CSV:

<https://prod.citisense.snowflakesoftware.com/csv/sensor/observationfinishtime/between?sensorid={sensor-id}&from=2001-12-17T09:00:00.000&to=2001-12-18T09:00:00.000>

JSON:

<https://prod.citisense.snowflakesoftware.com/json/sensor/observationfinishtime/between?sensorid={sensor-id}&from=2001-12-17T09:00:00.000&to=2001-12-18T09:00:00.000>

Scenario 4: Give me the last observation for a specific sensor-id
CSV:

<https://prod.citisense.snowflakesoftware.com/csv/sensor/lastobservation?sensorid={sensor-id}>

JSON:

<https://prod.citisense.snowflakesoftware.com/json/sensor/lastobservation?sensorid={sensor-id}>

Scenario 5: Give me the last observation for a specific sensor-id for a specific observed property (eg. pollutant, temperature, etc.)
CSV:

<https://prod.citisense.snowflakesoftware.com/csv/sensor/lastobservation/observedproperty?sensorid={sensor-id}&observedproperty=NO2>

JSON:

<https://prod.citisense.snowflakesoftware.com/json/sensor/lastobservation/observedproperty?sensorid={sensor-id}&observedproperty=NO2>



WFS service

There are two distinct end points for the WFS:

WFS End Points	Purpose
https://prod.citisense.snowflakesoftware.com/wfs	General, all purpose, WFS endpoint
https://prod.citisense.snowflakesoftware.com/wfs/r	Endpoint specifically for accessing the latest observation.

Scenario 1: Give me all observations for a specific sensor-id

XML

```
https://prod.citisense.snowflakesoftware.com/wfs?service=wfs&version=2.0.0&request=GetFeature
&typename=cts:Observation&filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http://www.citi-sense.eu/citisense">
  <PropertyIsEqualTo>
    <ValueReference>cts:sensorID/@xlink:href</ValueReference>
    <Literal>{sensor-id}</Literal>
  </PropertyIsEqualTo>
</Filter>
```

Scenario 2: Give me all observations for a specific sensor-id at a specific point in time (time instant)

XML

```
https://prod.citisense.snowflakesoftware.com/wfs?service=wfs&version=2.0.0&request=GetFeature
&typename=cts:Observation&filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http://www.citi-sense.eu/citisense">
  <And>
    <PropertyIsEqualTo>
      <ValueReference>cts:sensorID/@xlink:href</ValueReference>
      <Literal>{sensor-id}</Literal>
    </PropertyIsEqualTo>
    <PropertyIsEqualTo>
      <ValueReference>//cts:finishtime</ValueReference>
      <Literal>2001-12-18T10:30:47.000</Literal>
    </PropertyIsEqualTo>
  </And>
</Filter>
```

Scenario 3: Give me all observations for a specific sensor-id in a specific time period

XML

```
https://prod.citisense.snowflakesoftware.com/wfs?service=wfs&
version=2.0.0&request=GetFeature&
```



Scenario 3: Give me all observations for a specific sensor-id in a specific time period

```

typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http://www.citi-sense.eu/citisense">
<And>
<PropertyIsEqualTo>
<ValueReference>cts:sensorID/@xlink:href</ValueReference>
<Literal>{sensor-id}</Literal>
</PropertyIsEqualTo>
<PropertyIsBetween>
<ValueReference>//cts:finishtime</ValueReference>
<LowerBoundary>
<Literal>2001-12-17T09:00:00.000</Literal>
</LowerBoundary>
<UpperBoundary>
<Literal>2001-12-18T09:00:00.000</Literal>
</UpperBoundary>
</PropertyIsBetween>
</And>
</Filter>

```

Scenario 4: Give me the last observation for a specific sensor-id

XML

```

https://prod.citisense.snowflakesoftware.com/wfsr?service=wfs&
version=2.0.0&
request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http://www.citi-sense.eu/citisense">
<PropertyIsEqualTo>
<ValueReference>cts:sensorID/@xlink:href</ValueReference>
<Literal>{sensor-id}</Literal>
</PropertyIsEqualTo>
</Filter>

```

Scenario 5: Give me the last observation for a specific sensor-id for a specific observed property (eg. pollutant, temperature, etc.)

XML

```

https://prod.citisense.snowflakesoftware.com/wfsr?service=wfs&
version=2.0.0&
request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http://www.citi-sense.eu/citisense">
<And>
<PropertyIsEqualTo>
<ValueReference>cts:sensorID/@xlink:href</ValueReference>
<Literal>{sensor-id}</Literal>

```


Scenario 5: Give me the last observation for a specific sensor-id for a specific observed property (eg. pollutant, temperature, etc.)

```

</PropertyIsEqualTo>
<PropertyIsEqualTo>
<ValueReference>//cts:observedProperty</ValueReference>
<Literal>NO2</Literal>
</PropertyIsEqualTo>
</And>
</Filter>

```

Scenario 6: Give me the last observation for all sensors for a particular location by name
XML

```

https://prod.citisense.snowflakesoftware.com/wfsI_r_location?service=wfs&
version=2.0.0&
request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:cts="http://www.citi-sense.eu/citisense">
<Or>
<PropertyIsLike escape="\\" singleChar="_" wildcard="*">
<ValueReference>gml:name</ValueReference>
<Literal>*oslo*</Literal>
</PropertyIsLike>
<PropertyIsLike escape="\\" singleChar="_" wildcard="*">
<ValueReference>gml:name</ValueReference>
<Literal>*Oslo*</Literal>
</PropertyIsLike>
<PropertyIsLike escape="\\" singleChar="_" wildcard="*">
<ValueReference>gml:name</ValueReference>
<Literal>*OSLO*</Literal>
</PropertyIsLike>
</Or>
</Filter>

```

Scenario 6: Give me the last observation for all sensors for a particular location using a spatial bounding box filter
XML

```

https://prod.citisense.snowflakesoftware.com/wfsI\_r\_location?service=wfs&
version=2.0.0&
request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http://www.citi-sense.eu/citisense"
xmlns:gml="http://www.opengis.net/gml">
<And>
<BBOX>
<ValueReference>cts:the_geom</ValueReference>
<gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">

```


Scenario 6: Give me the last observation for all sensors for a particular location using a spatial bounding box filter

```
<gml:lowerCorner>32.72 34.92</gml:lowerCorner>
<gml:upperCorner>32.85 35.06</gml:upperCorner>
</gml:Envelope>
</BBOX>
</And>
</Filter>
```

Scenario 6: Give me the last observation for all sensors for a particular location using a logical bounding box filter
XML

```
https://prod.citisense.snowflakesoftware.com/wfslr\_location?service=wfs&
version=2.0.0&
request=GetFeature&
typename=cts:Observation&
filter=
<Filter xmlns="http://www.opengis.net/fes/2.0" xmlns:cts="http://www.citi-sense.eu/citisense"
xmlns:gml="http://www.opengis.net/gml">
<And>
<And>
<PropertyIsGreaterThanOrEqualTo>
<ValueReference>//cts:latitude</ValueReference>
<Literal>32.72</Literal>
</PropertyIsGreaterThanOrEqualTo>
<PropertyIsLessThanOrEqualTo>
<ValueReference>//cts:latitude</ValueReference>
<Literal>32.85</Literal>
</PropertyIsLessThanOrEqualTo>
</And>
<And>
<PropertyIsGreaterThanOrEqualTo>
<ValueReference>//cts:longitude</ValueReference>
<Literal>34.92</Literal>
</PropertyIsGreaterThanOrEqualTo>
<PropertyIsLessThanOrEqualTo>
<ValueReference>//cts:longitude</ValueReference>
<Literal>35.06</Literal>
</PropertyIsLessThanOrEqualTo>
</And>
</And>
</Filter>
```



3 Access to SensApp Server

SensApp is a platform for sensor and sensor data handling. It is an open-source, service-based application for sensor registration, data collection, storage and visualization of sensor data.

3.1 Setup

The following describes how to access the CITI-SENSE SensApp server provided by SINTEF

3.2 Installing SensApp dependencies

SensApp is designed to run in Jetty, a lightweight application server that can easily be deployed on constrained servers running Unix-based systems (32Mb of RAM are sufficient). Here are the installation instructions for setting up of a SensApp cloud service. One SensApp cloud service is initially being provided by SINTEF, but more services can be set up if necessary.

When deploying SensApp in the cloud, the first step consists in creating a Virtual Machine running Linux as operating system. Ports 80 and 8080 should be open. The following steps have to be performed on the VM:

1. Install Java
2. Install MongoDB:
 - a. <http://www.mongodb.org/downloads>
 - b. Start the MongoDB server (run `mongod`)
3. Install jetty
 - a. <http://dist.codehaus.org/jetty/jetty-hightide-8.1.4/jetty-hightide-8.1.4.v20120524.tar.gz>
 - b. Add the jetty.sh script to you bin directory: `cp $JETTY_HOME/bin/jetty.sh /etc/init.d/jetty`

3.3 Deploying SensApp in the development environment

Once the environment properly configured, SensApp can be installed and started as follows:

1. Download <http://github.com/downloads/SINTEF-9012/sensapp/sensapp.war>
2. Move the sensapp.war to the "webapp2 directory: `cp $JETTY_HOME/bin/jetty.sh /etc/init.d/jetty`
3. Restart jetty: `service jetty restart`

3.4 Using Sensapp

In the following, we will describe how to create 3 atomic sensors associated to one bike. These three sensors will then be associated into a composite one. Finally, we will push data inside the system, and retrieve these data through SensApp.

3.4.1 Registering sensors:

In order to register a sensor, users need to send a HTTP POST request with a JSON body as follows:

```
{
  "id": "Bike1/gps_alt", "descr": "GPS altitude of the bike",
  "schema": { "backend": "raw", "template": "Numerical" }
}
```

The two other sensors can be created with similar requests:

```
{
  "id": "Bike1/gpsfix", "descr": "GPS fix status of the bike",
  "schema": { "backend": "raw", "template": "Numerical" }
}
```

```
{
  "id": "Bike1/ground_speed", "descr": "Ground speed measured on the bike",
  "schema": { "backend": "raw", "template": "Numerical" }
}
```

Existing templates are:

- Numerical
- Boolean
- String
- Summed

An example of such request is depicted in Figure 3-1.

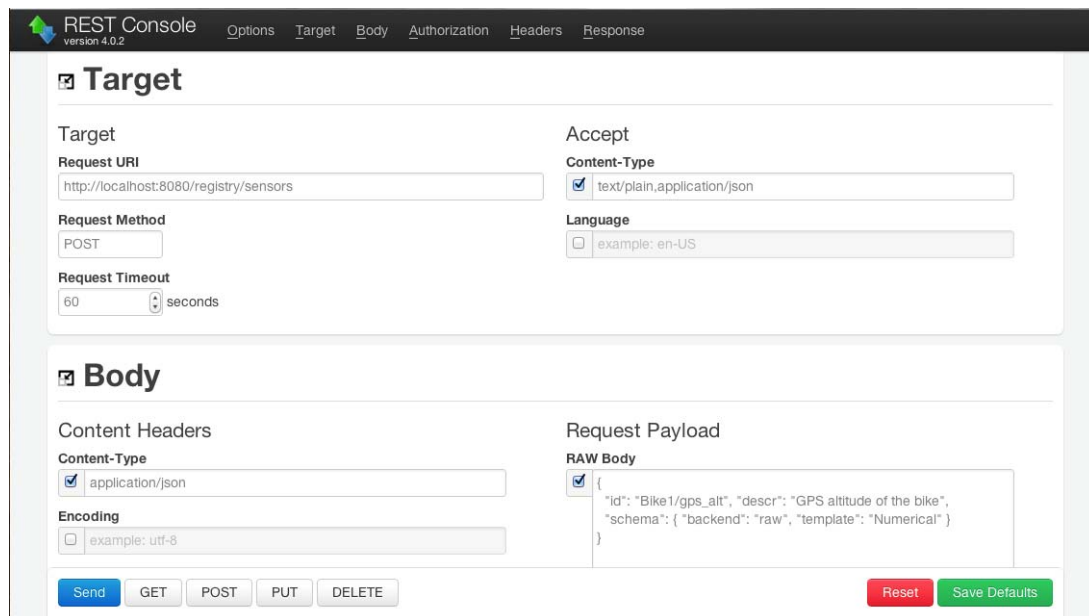


Figure 3-1 Using the RESTConsole to POST requests

3.4.2 Retrieve Sensor Description

In order to retrieve a sensor description, users need to send a HTTP GET request to an URL formatted as below:

- http://SERVER_URL/sensapp/registry/sensors/device_name

Accordingly, the result of the following request:



http://SERVER_URL/sensapp/registry/sensors/Bike1/gps_alt is a JSON describing the Bike1/gps_alt sensor.

3.4.3 Add metadata to registered sensors

In order to add metadata to a registered sensor, users need to send a HTTP PUT request to an URL formatted as below:

- http://SERVER_URL/sensapp/registry/sensors/device_name

The body of this request should be formatted in JSON as follows:

```
{ "tags": { "owner": "X", "license": "LGPL" } }
```

3.4.4 Create a composite sensor

In SensApp, a sensor is only for one type of data. If one wants to logically group several data types, then she has to create a composite sensor, which is a group of sensors.

In order to do so, send a HTTP POST request to:

- http://SERVER_URL/sensapp/registry/composite/sensors

With a JSON body following this structure:

```
{
  "id": "Bike1", "descr": "A Bike",
  "tags": { "owner": "X", "license": "LGPL" },
  "sensors": [
    "/registry/sensors/Bike1/gps_alt", "/registry/sensors/Bike1/gpsfix",
    "/registry/sensors/Bike1/ground_speed"
  ]
}
```

In this JSON structure, the array called "sensors" contains the list of sensors that will form the composite.

3.4.5 Push data inside SensApp

In order to push measurements into SensApp, users need to send a HTTP PUT request to the following URL:

- http://SERVER_URL/sensapp/dispatch

Sample measurements are available at the following address:

- https://github.com/SINTEF-9012/sensapp/blob/master/net.modelbased.sensapp.data.samples/CyclingData/Bike1/data/Bike1_1hz_gps_alt.senml.json

One can push data into the composite sensor by using a HTTP PUT request to the following URL:

- http://SERVER_URL/sensapp/dispatch



With a JSON body structured as follows:

```
{
  "bn": "Tmrt_site1", "bt": 1361521838,
  "e": [
    {"n": "/T", "u": "count", "v": 17.94},
    {"n": "/cloudiness", "u": "count", "v": 2},
    {"n": "/Tmrt", "u": "count", "v": 8.0}
  ]
}
```

Where:

- bn: composite name
- bt: basetime
- n: sensor name
- u: unit
- v: value

3.4.6 Retrieving sensor's data

In order to retrieve a sensor data, users need send to the server a HTTP GET request to an URL formatted as below:

- http://SERVER_URL/sensapp/registry/sensors/device_name

Follow the “dataset” link to access to the resource that contains the data:

- http://SERVER_URL/sensapp/databases/raw/data/device_name

Limit the retrieved data to only the 10 last measures:

- http://SERVER_URL/sensapp/databases/raw/data/device_name?limit=10

Limit the retrieved data with a time slot:

- http://SERVER_URL/sensapp/databases/raw/data/device_name?from=XXX&to=XXX

3.4.7 Retrieving data in CSV format

This service enables SensApp users to retrieve data from several sensors in CSV format. Users can specify through an HTTP POST request the set of sensors they are interested in as well as the character to be used as a separator between values. The listing below describes the typical content of a request in the Barcelona ExpoApp case. The objective of this request is to retrieve the GPS latitude, longitude, and the accelerometer values from a specific participant, each column being separated by a comma.

Example of content for a request to the encoder

```
{
  "datasets" : [{
    "url" : http:// SENSAPP\_URL /sensapp/databases/raw/data/ACC-z-ParticipantID001
  }, {
    "url" : http:// SENSAPP\_URL /sensapp/databases/raw/data/ACC-x-ParticipantID001
  }, {
    "url" : http:// SENSAPP\_URL /sensapp/databases/raw/data/ACC-y-ParticipantID001
  }, {
    "url" : http:// SENSAPP\_URL /sensapp/databases/raw/data/GPS-latitude-ParticipantID001
  }
]
```



```

    }, {
      "url" : http://SENSAPP\_URL/sensapp/databases/raw/data/GPS-longitude-ParticipantID001
    }
  ],
  "separator" : ",",
}

```

The result of such request consists in data formatted in CSV and organized as described in Table 3-1. Values are ordered and grouped according to their associated timestamp. Each line is composed of the value that has been collected at a specific time from each of the sensors specified in the request, each value being separated by a comma. The service represents the absence of measurement from a sensor within a line by a "-" in the CSV.

Table 3-1 Example of data aggregation

Time stamp	GPS lat	GPS long	ACC x	ACC y	ACC z
0:00:01	Lat	Long	x	y	z
0:00:02	Lat	Long	x	y	z
0:00:03	lat	Long	x	y	z

3.4.8 SensApp Admin web interface

The SensApp Admin GUI is accessible at the following address: http://SERVER_URL/SensAppGUI and can be used to perform the following actions:

- View the list of sensors and composite sensors registered in the SensApp server
- Create, update, delete sensors and composite sensors
- Observe the description of a sensor or a composite sensor
- Retrieve the JSON description of a sensor or a composite sensor
- Visualise sensors data in the forms of raw data tables or charts
- Export sensors data in SenML
- Export composite sensor data in the form of CSV

In the next sections we present how each of these actions can be performed.

3.4.8.1 Visualise the list of sensors and composite sensors

The main page of the SensAppAdmin GUI displays the list of sensors currently registered in the SensApp Server in a tabular form (see Figure 3-2). This page can always be reached by clicking on SensApp or by selecting "Administration>Sensors" in the menu.



SensApp Administration Visualisation Refresh page

Sensors

New Sensor

10 records per page

Search:

Name	Description	Creation date	Actions
A0001KTU84Q_86dca6ae980330f1_Sunnbuddy_UVA	TYPE_SUNBUDDY	2015-02-04 22:29:35	Edit Raw
A0001KTU84Q_86dca6ae980330f1_Sunnbuddy_UVB	TYPE_SUNBUDDY	2015-02-04 22:29:45	Edit Raw
ACC-x-ParticipantID002	ACC X axis	2015-02-17 10:44:55	Edit Raw
ACC-x-ParticipantID1111214	ACC X axis	2014-12-11 09:18:33	Edit Raw
ACC-x-ParticipantID1190115	ACC X axis	2015-01-19 12:46:26	Edit Raw
ACC-x-ParticipantID1260115	ACC X axis	2015-01-26 15:55:40	Edit Raw
ACC-x-ParticipantID2111214	ACC X axis	2014-12-11 09:17:39	Edit Raw
ACC-x-ParticipantID2181214	ACC X axis	2014-12-18 16:31:22	Edit Raw
ACC-x-ParticipantID2200115	ACC X axis	2015-01-20 18:05:23	Edit Raw
ACC-x-ParticipantID3090115	ACC X axis	2015-01-09 11:18:47	Edit Raw

Showing 1 to 10 of 859 entries

← Previous 1 2 3 4 5 Next →

Figure 3-2 SensApp Admin GUI main page

By default, the sensor table displays ten sensors; however, the number of records per page can be tuned as depicted in Figure 3-3.

Sensors

New Sensor

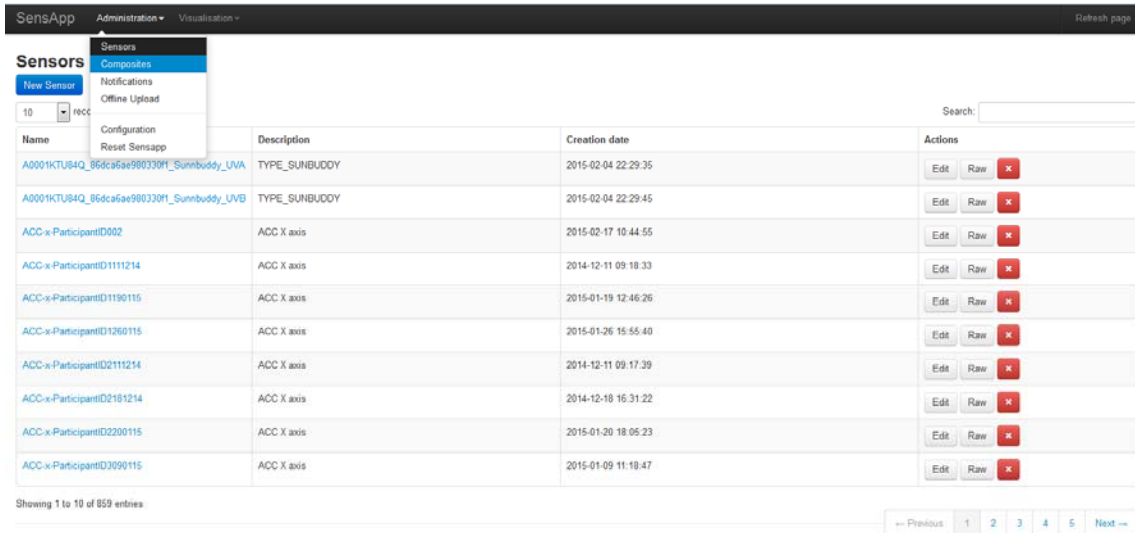
10 records per page

- 10
- 25
- 50
- 100

Name	Description
A0001KTU84Q_86dca6ae980330f1_Sunnbuddy_UVA	TYPE_SUNBUDDY
A0001KTU84Q_86dca6ae980330f1_Sunnbuddy_UVB	TYPE_SUNBUDDY
ACC-x-ParticipantID002	ACC X axis
ACC-x-ParticipantID1111214	ACC X axis
ACC-x-ParticipantID1190115	ACC X axis
ACC-x-ParticipantID1260115	ACC X axis

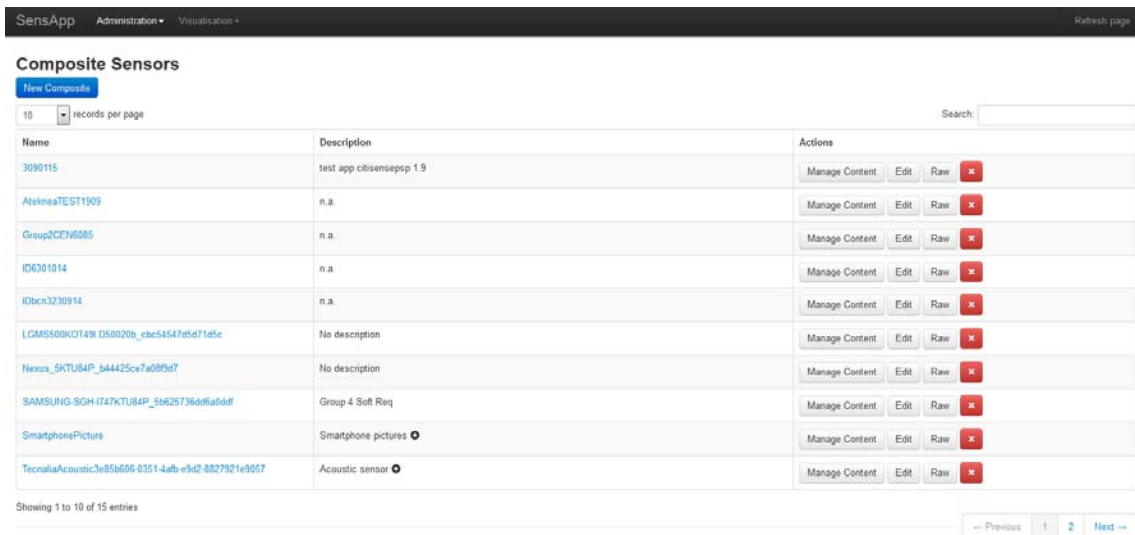
Figure 3-3 Change the number of records per page

Similarly, the list of composite sensors (see Figure 3-5) can be viewed by selecting "Administration>Composites" in the menu as shown in Figure 3-4.



Name	Description	Creation date	Actions
A0001KTU84Q_86dca5ae980330f1_Sunbuddy_UVA	TYPE_SUNBUDDY	2015-02-04 22:29:35	Edit Raw ✖
A0001KTU84Q_86dca5ae980330f1_Sunbuddy_UVB	TYPE_SUNBUDDY	2015-02-04 22:29:45	Edit Raw ✖
ACC-x-ParticipantID002	ACC X axis	2015-02-17 10:44:55	Edit Raw ✖
ACC-x-ParticipantID1111214	ACC X axis	2014-12-11 09:18:33	Edit Raw ✖
ACC-x-ParticipantID1190115	ACC X axis	2015-01-19 12:46:26	Edit Raw ✖
ACC-x-ParticipantID1260115	ACC X axis	2015-01-26 15:55:40	Edit Raw ✖
ACC-x-ParticipantID2111214	ACC X axis	2014-12-11 09:17:39	Edit Raw ✖
ACC-x-ParticipantID2161214	ACC X axis	2014-12-18 16:31:22	Edit Raw ✖
ACC-x-ParticipantID2200115	ACC X axis	2015-01-20 18:05:23	Edit Raw ✖
ACC-x-ParticipantID3090115	ACC X axis	2015-01-09 11:19:47	Edit Raw ✖

Figure 3-4 listing all composite sensors



Name	Description	Actions
3090115	test app citisensepsp 1 9	Manage Content Edit Raw ✖
AtelneaTEST1909	n.a.	Manage Content Edit Raw ✖
Group2CEN6085	n.a.	Manage Content Edit Raw ✖
ID6301014	n.a.	Manage Content Edit Raw ✖
IDbcn3230914	n.a.	Manage Content Edit Raw ✖
LGMS500K0T49i_050020b_ebe54547afd71d5c	No description	Manage Content Edit Raw ✖
Nexus_5KTU84P_b44425ce7a089d7	No description	Manage Content Edit Raw ✖
SAMSUNG-SGH-I747KTU84P_5b625736dd5a8d#f	Group 4 Soft Req	Manage Content Edit Raw ✖
SmartphonePictures	Smartphone pictures 🔍	Manage Content Edit Raw ✖
TecaliaAcoustic3e85b606-0351-4afb-e9d2-802792e19057	Acoustic sensor 🔍	Manage Content Edit Raw ✖

Figure 3-5 Composite sensors administration

3.4.8.2 Create, update, delete sensors and composite sensors

A new sensor or composite sensor can be created by clicking on "New Sensor" on their respective administration page. As a result, a form appears (see Figure 3-7) and has to be filled in order to create a new sensor. The set of mandatory fields are marked by a "*".

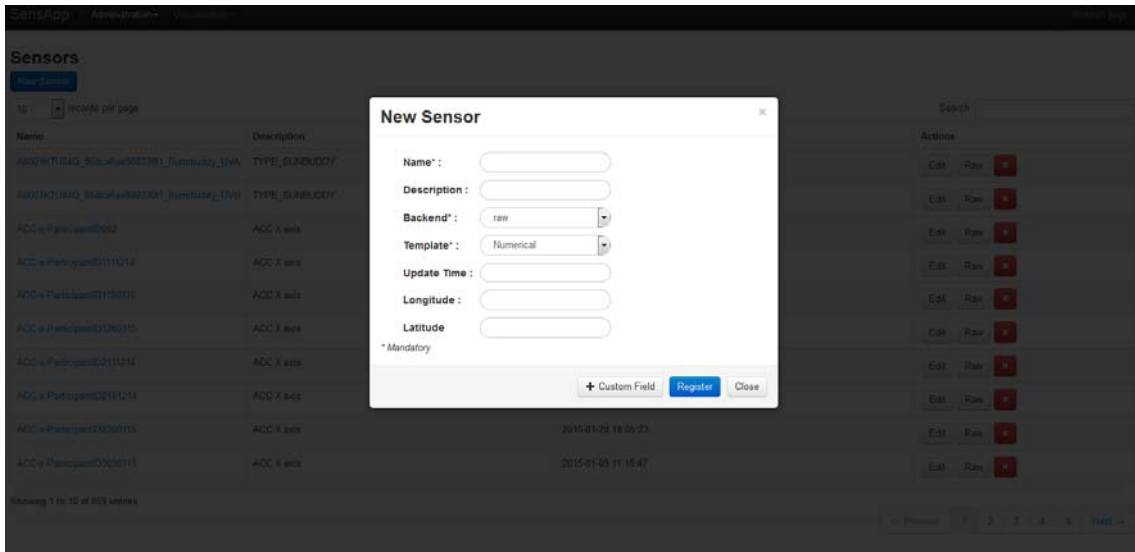


Figure 3-6 Sensor creation form

Each sensor can be deleted by clicking on the red button in the corresponding sensor row whilst it can be updated by clicking on the "edit" button.

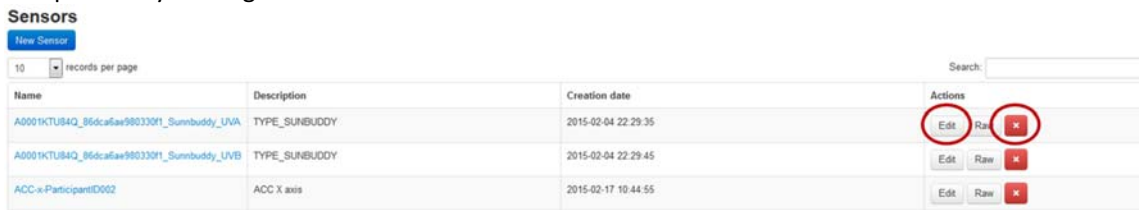


Figure 3-7 Managing sensors

3.4.8.3 Observe the description of a sensor or a composite sensor

The description of a sensor can be obtained by clicking on the sensor name (this also applies to composite sensors). The sensor description is thus displayed as depicted in Figure 3-8.

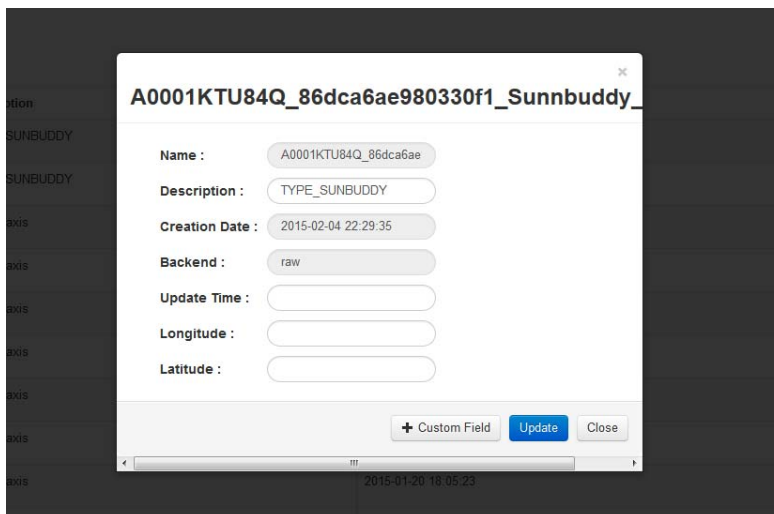


Figure 3-8 Sensor description

3.4.8.4 Retrieve the JSON description of a sensors or a composite sensor

In addition, users can retrieve the SenML description of a sensor (the same applies for a composite sensor) by clicking on the "Raw" button associated to the sensor of interest as depicted in Figure 3-9.

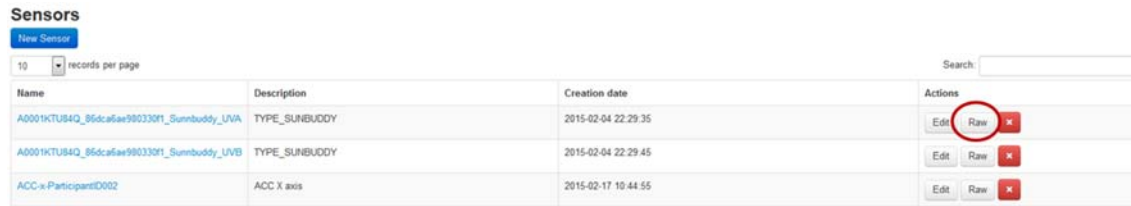


Figure 3-9 retrieve the raw description of a sensor

3.4.8.5 Visualise sensor data in the forms of a raw data table or charts

Sensor data can be visualised as a table associating the raw data to its timestamp or in the form of charts. The list of sensors whose data can be visualized can be obtained by selecting "Visualisation>Sensors" in the SensApp Admin menu as depicted in Figure 3-10.

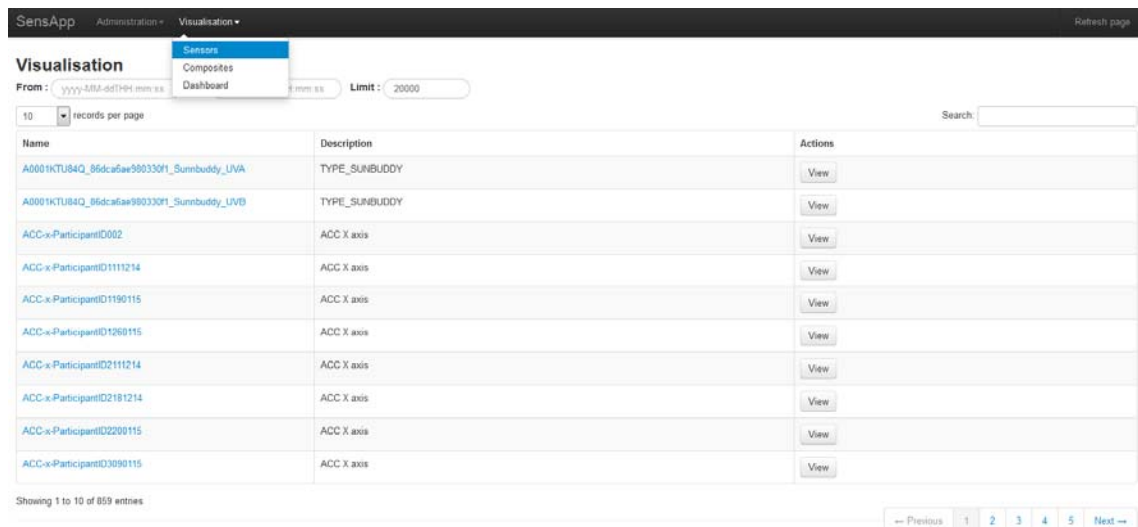


Figure 3-10 List of sensors whose data can be visualized

The list of sensors can be filtered by using the Search field (see Figure 3-11).



Figure 3-11 Filtering sensor data



Users can displays the data from a specific sensor by clicking on the corresponding "View" button as depicted in Figure 3-12.

Visualisation

From : yyyy-MM-ddTHH:mm:ss To : yyyy-MM-ddTHH:mm:ss Limit : 20000

10 records per page

Name	Description	Actions
A0001KTU84Q_86dca6ae980330f1_Sunnbuddy_UVA	TYPE_SUNBUDDY	View
A0001KTU84Q_86dca6ae980330f1_Sunnbuddy_UVB	TYPE_SUNBUDDY	View
ACC-x-Participant002	ACC X axis	View

Figure 3-12 View data from a specific sensor

By default the number of measurements displayed is limited to 20 000. This number can be changed by modifying the "Limit" field (see Figure 3-11) before clicking on "View". In order to visualize all the measurements from a sensor, this field has to be set to -1. In addition, it is possible to visualize data for a specific time window by using the fields "from" and "to" (see Figure 3-11). By default measurements are displayed in the tabular form shown in Figure 3-13.

A0001KTU84Q_86dca6ae980330f1_Sunnbuddy_UVA

Back Export

Table Chart Dynamic Chart Bar Chart

10 records per page

Time	Value (W/m2)
2015-02-12 18:44:57	0
2015-02-12 18:44:55	0
2015-02-12 18:44:53	0.0038836
2015-02-12 18:44:51	0
2015-02-12 18:44:49	0
2015-02-12 18:44:47	0
2015-02-12 18:44:45	0
2015-02-12 18:44:43	0
2015-02-12 18:44:41	0
2015-02-12 18:44:39	0

Showing 1 to 10 of 1.312 entries

Previous 1 2 3 4 5 Next

Figure 3-13 Displaying raw data

However, when sensors are registered with the numerical template (values are numeric), measurements can be displayed in the form of charts (see Figure 3-14) by clicking on one of the chart buttons.

A0001KTU84Q_86dca6ae980330f1_Sunnbuddy_UVA

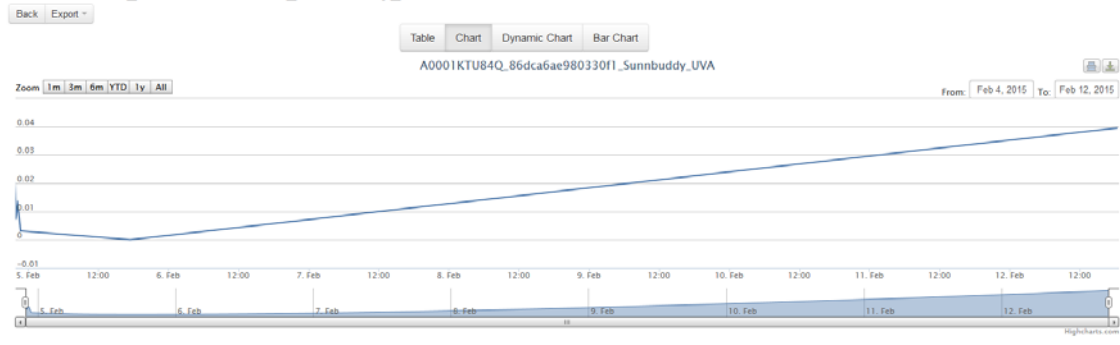
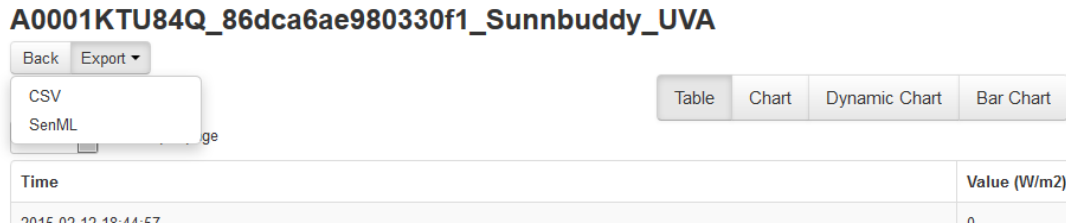


Figure 3-14 Displaying data as a chart

3.4.8.6 Export sensors data in SenML

It is also possible to retrieve sensors data as plain SenML. To do so, once a sensor has been selected for visualization, users can click on the "Export" button and select "SenML" as depicted in Figure 3-15.



A0001KTU84Q_86dca6ae980330f1_Sunbuddy_UVA

Back Export ▾

CSV
SenML

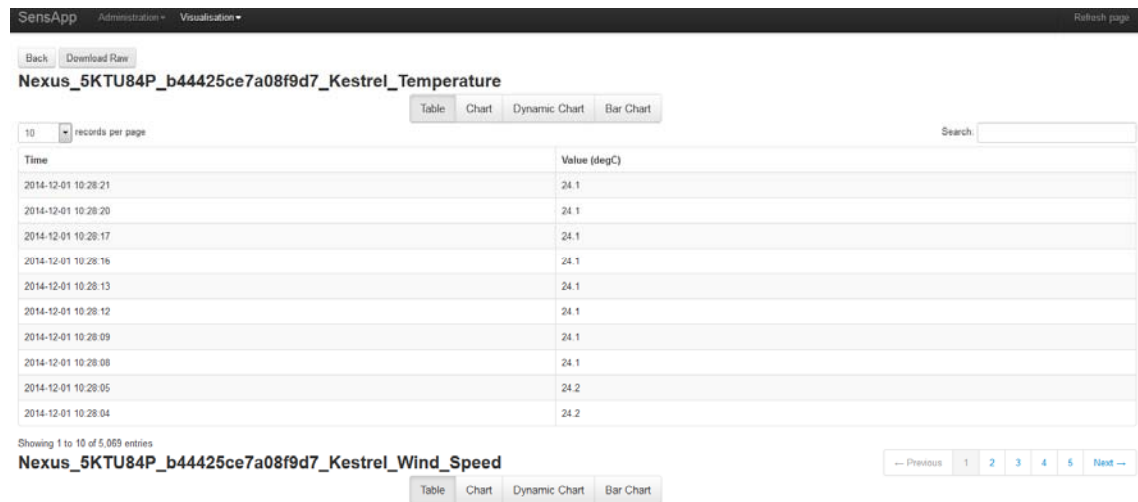
Table Chart Dynamic Chart Bar Chart

Time	Value (W/m2)
2014-12-10 10:44:57	0

Figure 3-15 Export sensor data as SenML

3.4.8.7 Export composite sensor data in the form of CSV

In addition, the measurements of all sensors forming a composite sensor can be aggregated in a CSV file. This file is generated on the SensApp server and can be downloaded from the client side. This can be done by visualizing the data of a composite sensor ("Visualization>Composites" and then "View") and then clicking on the "Download raw" button (see Figure 3-16).



SensApp Administration Visualization Refresh page

Back Download Raw

Nexus_5KTU84P_b44425ce7a08f9d7_Kestrel_Temperature

Table Chart Dynamic Chart Bar Chart

10 records per page Search: _____

Time	Value (degC)
2014-12-01 10:28:21	24.1
2014-12-01 10:28:20	24.1
2014-12-01 10:28:17	24.1
2014-12-01 10:28:16	24.1
2014-12-01 10:28:13	24.1
2014-12-01 10:28:12	24.1
2014-12-01 10:28:09	24.1
2014-12-01 10:28:08	24.1
2014-12-01 10:28:05	24.2
2014-12-01 10:28:04	24.2

Showing 1 to 10 of 5,069 entries

Nexus_5KTU84P_b44425ce7a08f9d7_Kestrel_Wind_Speed

Table Chart Dynamic Chart Bar Chart

← Previous 1 2 3 4 5 Next →

Figure 3-16 Generate CSV file with all measurements from of composite sensor

Depending on the amount of data stored for the selected composite sensor, the CSV generation can take several minutes. Once generated, a message displays the URL of the generated file and the file is automatically downloaded.

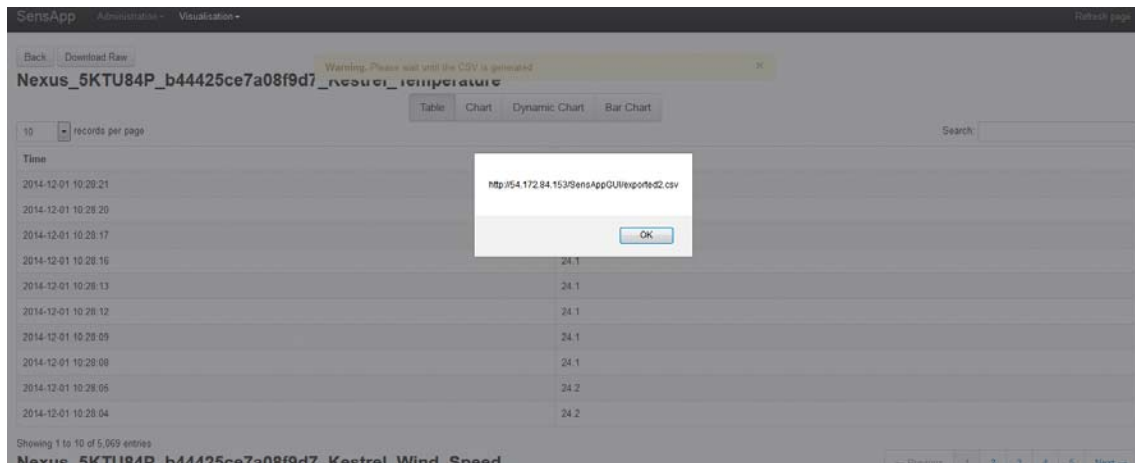


Figure 3-17 Download the generated csv file

3.4.9 SensAppHelper

This section shows a Java class describing how to interact with SensApp in Java.

```

package org.thingml.utils.http;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.StringWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

/**
 *
 * @author bmori
 * Inspired by http://www.gitpaste.com/paste/727/E1kSdBqXD7efNO4x5L1Z2T
 */
public class SensAppHelper {

    public static void main(String[] args) throws Exception {
        URL regURL = new
URL("http://Your_Installed_SensApp_ServerYour_Installed_SensApp_Server/sensapp
/registry/sensors");
        String status = SensAppHelper.registerSensor(regURL, "test", "just a test",
"raw", "Numerical");
        System.out.println(status);
    }

    public static String registerSensor(URL target, String id, String descr, String
backend, String tpl) throws Exception {
        StringBuilder req = new StringBuilder();
        req.append("{\"id\": \""+id+"\", \"descr\": \""+descr+"\"},");
        req.append("{\"schema\": { \"backend\": \""+backend+"\", \"template\": \""+
tpl +"\"}}");

        System.out.println(req.toString());

        //URL target = new URL("http", server, port, "/registry/sensors");
        HttpURLConnection c = (HttpURLConnection) target.openConnection();
        c.setDoOutput(true);
        c.setRequestMethod("POST");
        c.addRequestProperty("Content-type", "application/json");
        OutputStreamWriter wr = new OutputStreamWriter(c.getOutputStream());
        wr.write(req.toString());
        wr.flush();
    }
}

```



```

        BufferedReader rd = new BufferedReader(new
InputStreamReader(c.getInputStream()));
        String result = rd.readLine();
        rd.close();
        wr.close();
        return result;
    }

    public static String getSensorDetails(URL target, String url) throws Exception {
        //URL target = new URL("http", server, port, url);
        HttpURLConnection c = (HttpURLConnection) target.openConnection();
        BufferedReader in = new BufferedReader(new
InputStreamReader(c.getInputStream()));
        String inputLine;
        StringWriter result = new StringWriter();
        while ((inputLine = in.readLine()) != null)
            result.append(inputLine + "\n");
        in.close();
        return result.toString();
    }

    public static String pushData(URL target, String data) throws Exception {
        //URL target = new URL("http", server, port, "/dispatch");
        HttpURLConnection c = (HttpURLConnection) target.openConnection();
        c.setDoOutput(true);
        c.setRequestMethod("PUT");
        c.addRequestProperty("Content-type", "application/json");
        OutputStreamWriter wr = new OutputStreamWriter(c.getOutputStream());
        wr.write(data);
        wr.flush();
        BufferedReader rd = new BufferedReader(new
InputStreamReader(c.getInputStream()));
        String result = rd.readLine();
        rd.close();
        wr.close();
        return result;
    }

    public static String getData(URL target, String contentType) throws Exception {
        //URL target = new URL("http", server, port, url);
        HttpURLConnection c = (HttpURLConnection) target.openConnection();
        c.addRequestProperty("Accept", contentType);
        BufferedReader in = new BufferedReader(new
InputStreamReader(c.getInputStream()));
        String inputLine;
        StringWriter result = new StringWriter();
        while ((inputLine = in.readLine()) != null)
            result.append(inputLine + "\n");
        in.close();
        return result.toString();
    }
}

```

4 Visualisation components

4.1 Widget framework for Measurements by Dunavnet

The following widget framework for measurements has been designed and provided by the CITI-SENSE partner Dunavnet.

The documentation and demo source code has been provided so that mobile and web applications can easily be developed.

The framework has the following features:

- Based on Highcharts widgets (<http://www.highcharts.com/demo/>)
- Integrated with PHP in order to allow simple customization
- Demos with source code provided for iOS, Android and web platforms
- Supports mysql, mssql, postgresql and mongoDB
- Currently hosted on DNET server, but can be integrated on CITI-SENSE platform
- Documentation is here: <http://srv.dunavnet.eu/citisense/>
- Demo page is here: <http://srv.dunavnet.eu/citisense/demo/>



4.1.1 Installation

Download and unzip citisense to a project folder. Open blank index.php and simple 'include' or 'require' Citisense.php from /libs/

```
1 <?php
2 include_once("libs/Citisense.php");
```

4.1.2 Configuring data

There are two ways of using and processing data. First is to configure citisense Database.php class. Enter here your database information such as database driver, database hostname, user account, password etc. Citisense support several database types. Second and more convenient way is to use already hosted web services, and parse data in some format. In our case (citisense.dunavnet.eu) we use several services to parse data from xml and json files. For instance, for parsing live data we use rdf xml page, at



```
http://dunavnet.servehttp.com/rephandler/ecobus/355255040755580
```

For parsing historical data, we create encoded query directly into service. For example in

```
http://dunavnet.servehttp.com/rephandler/ecodb/
```

we append some query

```
SELECT co_ppm FROM history_data WHERE timestamp_db>= DATEADD(day, -1, GETDATE())and bus_identifier = 355255040039514
```

that after encoding looks like

```
SELECT%20co_ppm%20FROM%20history_data%20WHERE%20timestamp_db%3E%3D%20DATEADD(day%2C%20-1%2C%20GETDATE())and%20bus_identifier%20%3D%20355255040755580
```

and finally our xml link looks like

```
http://dunavnet.servehttp.com/rephandler/ecodb/SELECT%20co_ppm%20FROM%20history_data%20WHERE%20timestamp_db%3E%3D%20DATEADD(day%2C%20-1%2C%20GETDATE())and%20bus_identifier%20%3D%20355255040755580
```

In this example, we retrieve CO data for the 355255040755580 device.

4.1.3 Initialization

When citisense is included, simply make two instances of Citisense and GUI class. Citisense class contains all required methods to operate with pollution widgets. In other hand GUI class is responsible for drawing HTML block and widgets itself.

```
1 <?php
2 $citi = new Citisense();
3 $gui = new Gui();
```

4.1.4 Get data

When instances are created, it is very important to fill our variables with data. The framework currently supports mysql, mssql, postgresql and mongoDB. It is also possible to parse any kind of data from a service as an xml or json file. First step is to configure our Config.php file (/libs/Config.php). Enter your basic database info such as, hostname, dbname, port, username and password.

Example of parsing data directly from database

```
1 <?php
2 $pressure = $citi->getLastMeasurement('measurements', 'pritisak', 'measurement_id');
```

Parsing data from our live web service

```
1 <?php
2 $prWS = $citi->parseLiveXML('http://89.216.116.166/rephandler/ecobus/355255040755580', 'pressure');
```

The 'measurements' property is the database table name, 'pritisak' is the row name we want to parse and 'measurement_id' is the table's identification of the row.

4.1.5 Draw basic widget

After we have collected data into the variable, the next step is to simple draw something. First we need to setup a basic html structure with drawHeader() and drawFooter() methods. It is very important that everything else go between those two methods. Because this is only demo library, the framework can only draw Pressure, Humidity, Temperature and WindSpeed.

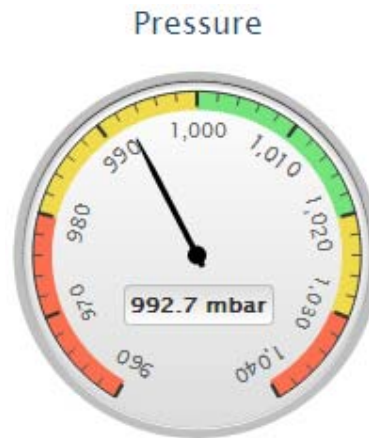
Example code for drawing pressure data

```

1 <?php
2 $gui->drawHeader();
3 $gui->drawPressureWidget($pressure);
4 $gui->drawFooter();

```

Result when page has rendered



4.1.6 Positioning and Sizing

Positioning and sizing the widgets are pure html and css. There is no need for extra php functions or javascript object. In your CSS file, reference #container id (this is the location where highcharts injects the javascript) and add a style width or height.

```

1 $(document).ready(function(){
2     $("#container").css("width","400px");
3     $("#container1").css("width","200px").css("height","600px");
4     $('#container2').css("margin-left","20%");
5 });

```

4.1.7 Process data

The framework can also process data, 'historical' or 'live' for calibration purposes. In this example, we pull out average values of the first 50 elements in database. First, we need to pull and store array of data in some variable.

```

1 <?php
2
3 $array_pr = $citi->getAllMeasurements('measurements', 'pritisak', '50');
4 $avg_pr = $citi->getAverage($array_pr, 'pritisak');
5 $gui->drawAverage($avg_pr, 'pressure');

```

4.1.8 Example code

Code below represent example with some basic functions.

```

1 <?php
2
3 include_once("libs/Citisense.php");
4
5 $citi = new Citisense();
6 $gui = new Gui();
7
8 // parsing data from database
9 // *****
10 $pritisak = $citi->getLastMeasurement('measurements', 'pritisak', 'measurement_id');
11 $vlaznost = $citi->getLastMeasurement('measurements', 'vlaznost', 'measurement_id');
12 $temperatura = $citi->getLastMeasurement('measurements', 'temperatura', 'measurement_id');
13 $brzina = $citi->getLastMeasurement('measurements', 'brzina', 'measurement_id');
14
15 // parsing data from web service
16 // *****
17 $prWS = $citi->parseLiveXML('http://89.216.116.166/rephandler/ecobus/355255040755580', 'pressure');
18 $so2WS = $citi->parseLiveXML('http://89.216.116.166/rephandler/ecobus/355255040755580', 'so2');
19 $no2WS = $citi->parseLiveXML('http://89.216.116.166/rephandler/ecobus/355255040755580', 'no2');
20 $coWS = $citi->parseLiveXML('http://89.216.116.166/rephandler/ecobus/355255040755580', 'co');
21 $co2WS = $citi->parseLiveXML('http://89.216.116.166/rephandler/ecobus/355255040755580', 'co2');
22 $o3WS = $citi->parseLiveXML('http://89.216.116.166/rephandler/ecobus/355255040755580', 'o3');
23 $noWS = $citi->parseLiveXML('http://89.216.116.166/rephandler/ecobus/355255040755580', 'no');
24
25 $niz = $citi->getAllMeasurements('measurements', 'temperatura', '50');
26 $avg_temp = $citi->getAverage($niz, 'temperatura');
27
28 $niz_pr = $citi->getAllMeasurements('measurements', 'pritisak', '50');
29 $avg_pr = $citi->getAverage($niz_pr, 'pritisak');
30
31 $niz_hum = $citi->getAllMeasurements('measurements', 'vlaznost', '50');
32 $avg_hum = $citi->getAverage($niz_hum, 'vlaznost');
33
34 $niz_ws = $citi->getAllMeasurements('measurements', 'brzina', '50');
35 $avg_ws = $citi->getAverage($niz_ws, 'brzina');
36
37

```

```
38 // draw html header structure
39 $gui->drawHeader();
40
41 // draw widgets parsed from web service
42 $gui->drawPressureWidget($prWS);
43 $gui->drawSO2Widget($so2WS);
44 $gui->drawNO2Widget($no2WS);
45 $gui->drawCOWidget($coWS);
46 $gui->drawCO2Widget($co2WS);
47 $gui->drawO3Widget($o3WS);
48 $gui->drawNOWidget($noWS);
49
50 // draw widgets parsed from database
51 $gui->drawHumidityWidget($vlaznost);
52 $gui->drawTemperatureWidget($temperatura);
53 $gui->drawWindSpeedWidget($brzina);
54
55 // draw average values (from database)
56 $gui->drawAverage($avg_temp, 'temperature');
57 $gui->drawAverage($avg_pr, 'pressure');
58 $gui->drawAverage($avg_hum, 'humidity');
59 $gui->drawAverage($avg_ws, 'wind speed');
60
61 // draw html footer structure
62 $gui->drawFooter();
```

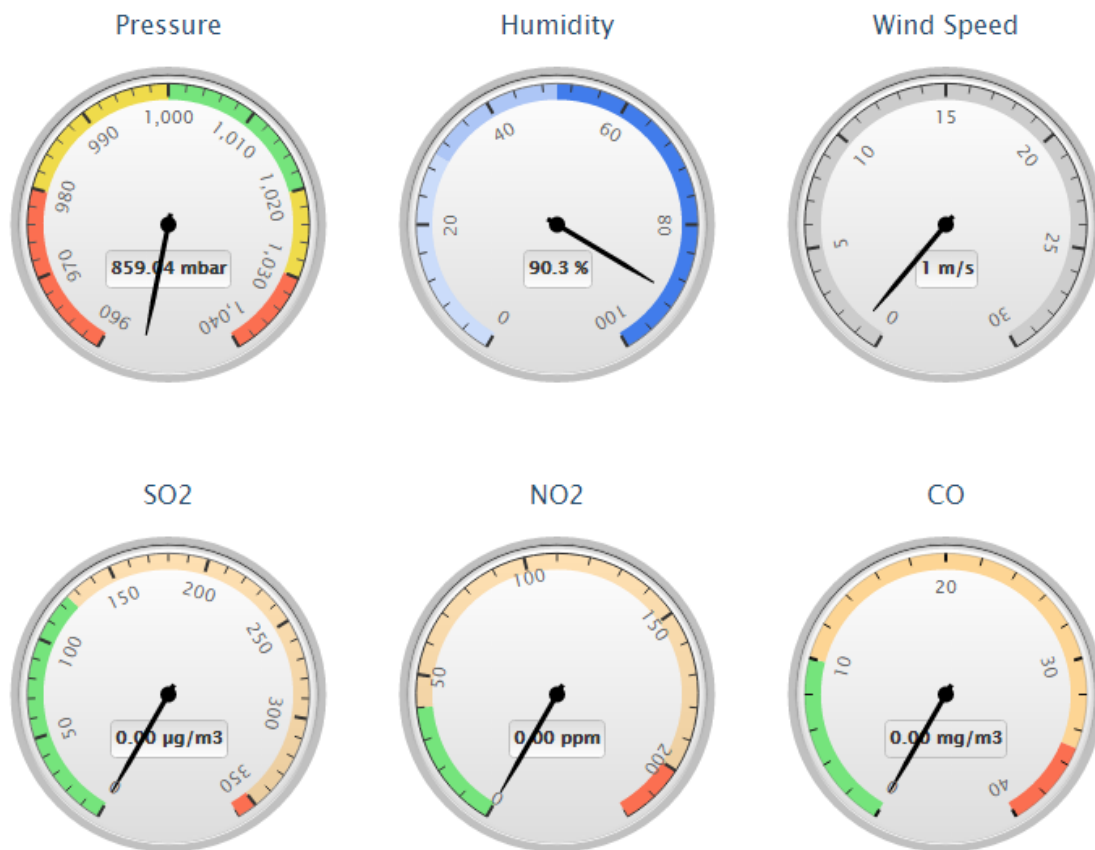
Example of result

Average temperature Value | 0.472

Average pressure Value | 901.21

Average humidity Value | 87.128

Average wind speed Value | 1.982



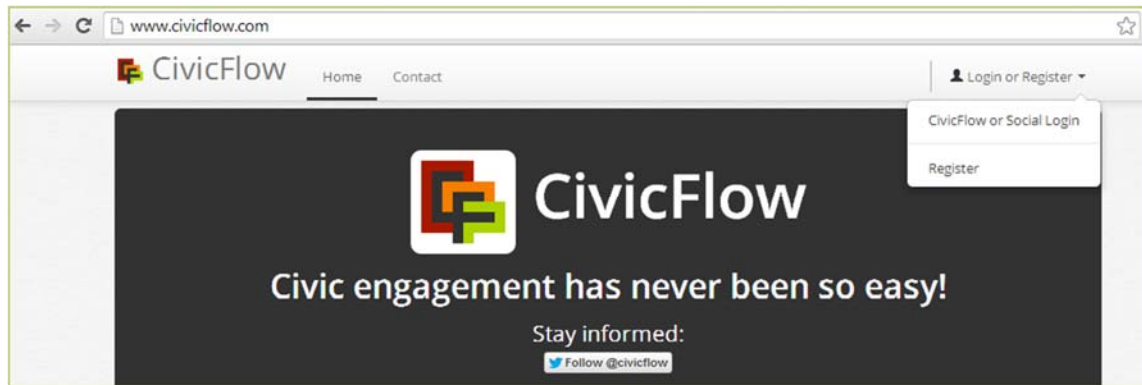
4.2 Widget framework for Questionnaires by U-Hopper

A widget framework for multi-channel questionnaires has been developed by U-Hopper.

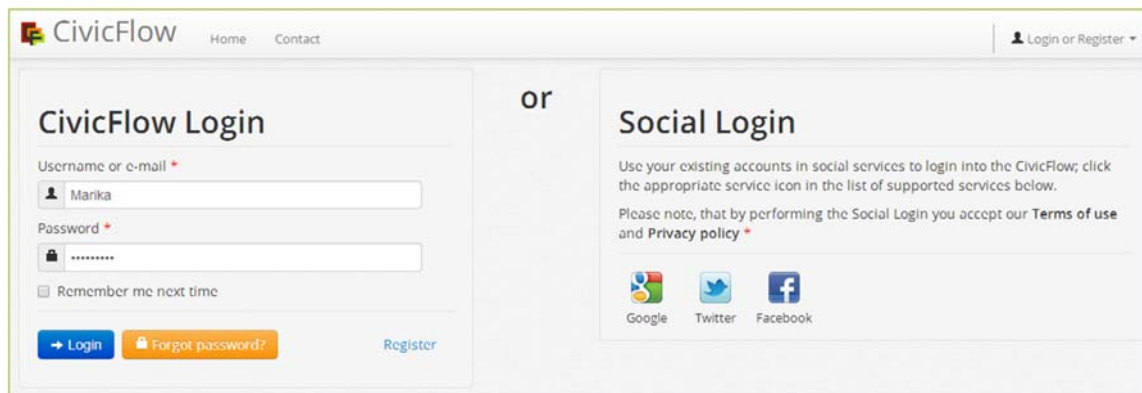
Go to www.civicflow.com and choose "CivicFlow or Social Login" or choose "Register" if you want or need to create a new account.

Registering as a user is uncomplicated and takes a minute or two.

If you choose "Social Login", you can log in with your Google, Twitter or Facebook account. This means CivicFlow will access your social profile information.

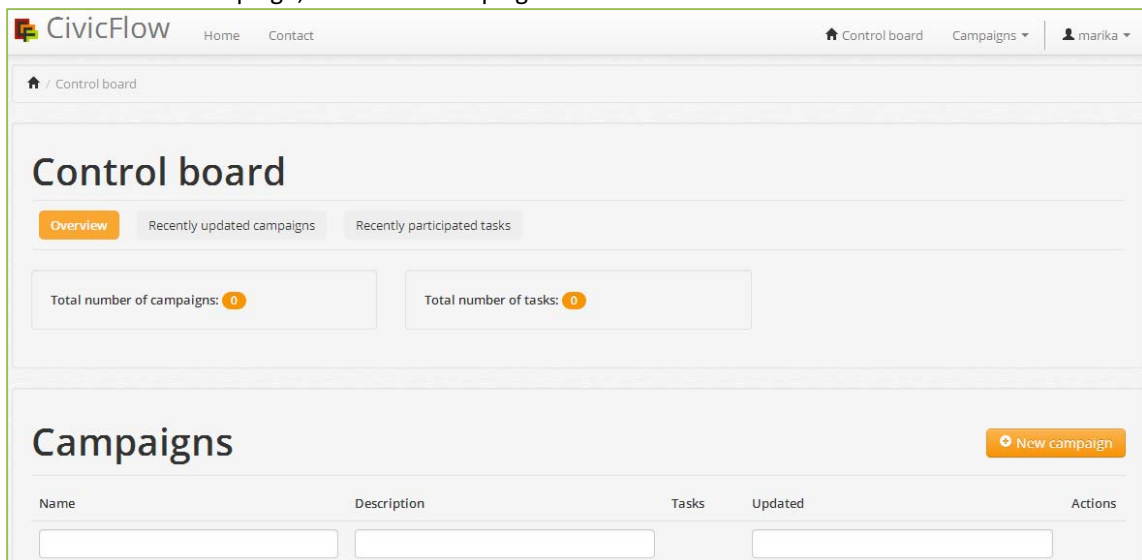


In the screenshot below, I choose to log in as a registered user.



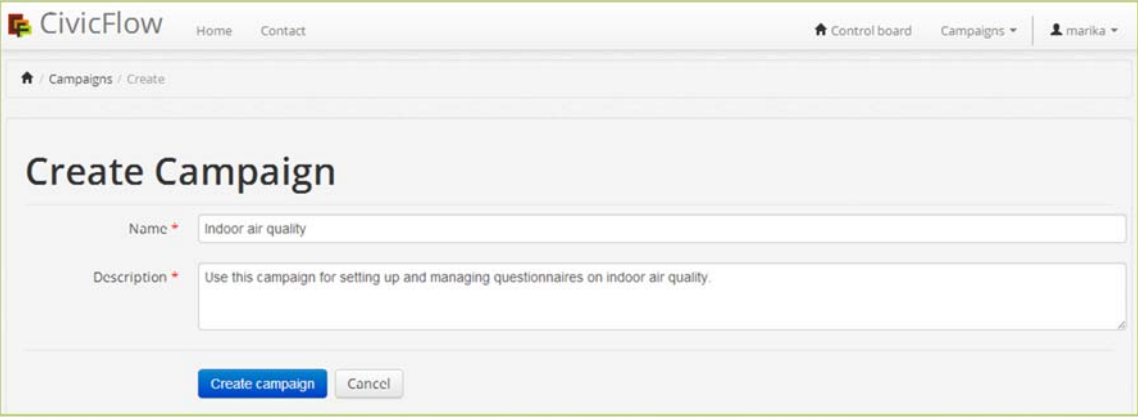
You need to **create a campaign** before you can create a questionnaire. You can add several questionnaires to one campaign.

To create a new campaign, click "New campaign"-button.



In this example, I create a campaign entitled "Indoor air quality" and add the description "Use this campaign for setting up and managing questionnaires on indoor air quality".

Click "Create campaign" when you have filled in name of campaign and description.



4.3 CITI-SENSE Sensors Visualization Widgets

The CITI-SENSE visualization widgets that has been presented above has extensively covered various possibilities for visualizing the data acquired by a single sensor. Given a sensor, a set of presented widgets allow to visually depict a real-time sensor measurement value (in a form of configurable Gauge chart), draw historical sensor measurements on a timeline (in a form of Line chart), and display historical sensor measurements on a map (for sensors publishing geographical coordinates of a measurement point). The widgets have been designed with the following set of requirements in mind: (1) configurability, allowing developers to easily configure the widgets for visualizing various aspects of measured data (e.g., range of measured values, coloring schemes for visual interpretation of data, etc.), (2) reusability, allowing developers to easily reuse the same widget for multiple measurements, and finally (3) embedding, allowing to easily plug widgets into 3rd party desktop and mobile applications and as such promote the CITI-SENSE data for the community. To reach that goal, the CITI-SENSE widgets has been developed using modern HTML5 techniques.

In the following, we present additional multi-sensor CITI-SENSE visualization widgets developed for depicting the data coming from multiple sensors. The widgets make use of the rich filtering capabilities of WFS storage in order to perform fine-tuned data retrieval. Similarly to single-sensor widgets, multi-sensor widgets target the same set of usage requirements and make use of HTML5 technologies.

The widgets share the common filtering configuration option, called “filter”, allowing definition of a sensor metadata look up criteria for retrieval of sensors of interest. A filter can specify multiple criteria, in such case the criteria will be connected with “AND” operand and hence will only retrieve sensors satisfying all criteria together. Look up criteria can be defined with the following attributes:

- “location” - for filtering by a name of the city a sensor installed in, e.g., Barcelona;
- “provider” – for filtering by an identifier of sensor provider, e.g., 2 will look up for Alphasense sensors;
- “status” – for filtering by sensor status, allowed values include “active” or “inactive”;
- “type” – for filtering by a type of a sensor, allowed values include “mobile” or “static”;
- “description” – for filtering by a partial containment of the given string in the sensor description metadata.

Special purpose “limit” attribute of a filter allows limiting the amount of extracted sensors.

In order to embed multi-sensor widgets in a 3rd party web page or a mobile application, a developer needs to include the widget script library and call the visualization rendering JavaScript code:

```
<script src="http://citisense.u-
hopper.com/widgets/sensors/jquery.sensors.lookup.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
  $("#YOUR_CONTAINER_ID").sensorslookup({
    ... other attributes specific to a visualization
    filters: {
      location: 'Barcelona'
      ,provider: 2
      ,status: 'Active'
      ,description: 'NO2'
      ,type: 'Mobile'
      ,limit: 1
    }
  });
</script>
```

The following is the list of CITI-SENSE widgets for visualization of multiple sensors:

4.3.1 Tabular visualization of sensors metadata

Tabular visualization allows extracting metadata of sensors satisfying filtering criteria and displaying them in a table. Such visualization is configured by a type “list” and can have additional attributes “title” and “subtitle” entitling the visualization. For example, in order to construct a table of *Active* sensors available in *Barcelona* a developer would use the following code:

```
<script src="http://citisense.u-
hopper.com/widgets/sensors/jquery.sensors.lookup.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
  $("#YOUR_CONTAINER_ID").sensorslookup({
    title: 'CITI-SENSE Barcelona Sensors'
    ,subtitle: 'available sensors'
    ,type: 'list'
    filters: {
      location: 'Barcelona'
      ,status: 'Active'
    }
  });
</script>
```

When the widget code is executed, the following visualization is constructed:

CITI-SENSE Barcelona sensors					
available sensors					
Identifier	Description	Type	Status	Location	Position
LEO-666C662E-bfdgh	NO2,O3,NO	mobile	active	Barcelona	
LEO-666C662E-cfhh	NO2,O3,NO	mobile	active	Barcelona	41.3525493, 2.0829019
LEO-666C662E- bvfgbgbg	NO2,O3,NO	mobile	active	Barcelona	
LEO-666C662E-hvcfgg	NO2,O3,NO	mobile	active	Barcelona	
LEO-666C662E-bzjdjd	NO2,O3,NO	mobile	active	Barcelona	

4.3.2 Map visualization of sensors metadata

Map visualization allows extracting metadata of sensors satisfying filtering criteria and displaying them on a map. Only sensors publishing the geographical position of measurements will be seen on a map. Such visualization is configured by a type “map”. For example, in order to construct a map of *Active* sensors available in *Barcelona* a developer would use the following code:

```

<script src="http://citisense.u-
hopper.com/widgets/sensors/jquery.sensors.lookup.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
  $("#YOUR_CONTAINER_ID").sensorslookup({
    type: 'map'
    , filters: {
      location: 'Barcelona'
      , status: 'Active'
    }
  });
</script>

```

When the widget code is executed, the following visualization is constructed:



Figure 4-1 Example of map with sensor data

Note that, when using the widget of type “map”, a developer can additionally specify a configuration attribute “layers” and give an array of URLs pointing to additional visualizations provided as KML format (Keyhole markup language), for example 3rd party noise maps, pollutions map, etc. For example, in order to overlay the above visualizations of sensors in Barcelona, a developer could use the following code:

```

<script src="http://citisense.u-
hopper.com/widgets/sensors/jquery.sensors.lookup.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
  $("#YOUR_CONTAINER_ID").sensorslookup({
    type: 'map'
    , filters: {
      location: 'Barcelona'
      , status: 'Active'
    }
    , layers: [
      url: 'http://pointer_to_kml_file_1'
      , url: 'http://pointer_to_kml_file_2'
    ]
  });
</script>

```

4.3.3 Tabular visualization of Air Quality Indexes of sensors

Tabular visualization widget of latest air quality indexes allows constructing a tabular representation of sensors satisfying filtering criteria and measuring air quality indexes. The visualization is configured by a type “list” and additional “measurement” configuration attribute with a “type” set to “aqi” or “caqi” and the “observedProperty” set to the gas the quality index needs to be extracted for. For

example, in order to construct a table of *Active* and “*Mobile*” sensors available in *Barcelona* performing measurements of *air quality index for NO2* a developer would use the following code:

```
<script src="http://citisense.u-
hopper.com/widgets/sensors/jquery.sensors.lookup.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
  $("#YOUR_CONTAINER_ID").sensorslookup({
    type: 'list'
    , title: 'Latest values of AQI for Barcelona'
    , subtitle: 'NO2'
    , filters: {
      location: 'Barcelona'
      , status: 'Active'
      , type: 'Mobile'
    }
    , measurement: {
      type: 'aqi'
      , observedProperty: 'NO2'
    }
  });
</script>
```

When the widget code is executed, the following visualization is constructed:

Latest values of AQI for Barcelona						
NO2						
Identifier	Description	Type	Status	Location	Position	AQI
LEO-666BA0BF	NO2,O3,NO	mobile	active	Barcelona	41.354713, 2.079287	4 (red)
LEO-666C662E	NO2,O3,NO	mobile	active	Barcelona	41.3524222, 2.0831551	1 (green)
LEO-666937AC	NO2,O3,NO	mobile	active	Barcelona	41.3545866, 2.078932	1 (green)
LEO-666B9D38	NO2,O3,NO	mobile	active	Barcelona	41.387383, 2.159409	2 (yellow)

4.3.4 Map visualization of Air Quality Indexes of sensors

Map visualization widget of latest air quality indexes allows displaying on a map points corresponding to sensors satisfying filtering criteria and colored with respect to the air quality index measured by a sensor. The visualization is configured by a type “map” and additional “measurement” configuration attribute with a “type” set to “aqi” or “caqi” and the “observedProperty” set to the gas the quality index needs to be extracted for. For example, in order to construct a map of *Active* and “*Mobile*” sensors available in *Barcelona* performing measurements of *air quality index for NO2* a developer would use the following code:

```
<script src="http://citisense.u-
hopper.com/widgets/sensors/jquery.sensors.lookup.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
  $("#YOUR_CONTAINER_ID").sensorslookup({
    type: 'map'
    , filters: {
      location: 'Barcelona'
      , status: 'Active'
      , type: 'Mobile'
    }
    , measurement: {
      type: 'aqi'
      , observedProperty: 'NO2'
    }
  });
</script>
```

When the widget code is executed, the following visualization is constructed:



Figure 4-2 Example of map with air quality index

The color of the point corresponds to the color assigned to the air quality index.

4.4 CITI-SENSE Perception Acquisition Campaigns

As noted in the previous version of the deliverable, besides the environmental measurements performed with physical sensors, such as various pollutions particles, levels of carbon monoxide, nitrogen dioxide, etc., the CITI-SENSE exploits the perception of environmental conditions experienced by citizens through the use of perception acquisition questionnaires. Such questionnaires focus on inquiring how citizens perceive the quality of the air, their impact on physical state of citizens, awareness of possible air quality hazards on citizens’ health, etc. For creation and delivery of questionnaires to citizens, the CITI-SENSE leverages the CivicFlow instrument (<http://www.civicflow.com>), developed by U-Hopper partner.

In order to execute similar perception questionnaires across multiple pilot sites, the following CivicFlow workflow has been adapted. Pilot officers agree upon the creation of the questionnaire template, which is set up in English. After that every pilot officer can “clone” this questionnaire in order to localize the template for his/her city, perform the translation of questions and answers in options:



In order to accommodate user-friendly way to answer a questionnaire, the conditional workflow has been implemented in the CivicFlow, allowing to encode conditions of types:



- If user answers an option X in a question Y, then hide or enable the question Z
- If user answers an option X in a question Y, then bring the user to the end of questionnaire

If question	is answered	then	Affected question(s)	Action
How often do you consciously look at air...	Never	Enable question(s)	If you have answered "Never" in the previous question, why do you never look at air quality information?	[Edit]
We also have two extra, more detailed...	No, sorry I don't have time now	Proceed to the end of questionnaire		[Edit]

Look and feel visualization of perception questionnaires on both, desktop computers and mobile devices has been unified and made user friendly by adding additional start page, explaining the scope of the questionnaire and the end page explaining the data usage policy of the CITI-SENSE project.



5 Summary

The updates in D7.6 compared to D7.5 are in particular an update of details of the data access in the chapters on data products and services and the CITI-SENSE data model.

Further the D7.6 is split into three parts, with part 1 focusing on the overall architecture and specifications and this part 2 focusing on the operational aspects including more details of the various sensor platforms and app usages of the CITI-SENSE architecture and platform. D7.5 part 3 - Citizen Observatory Toolbox - Developer perspective - contains information about how the various parts of the resulting toolbox, including methods, data management, web portals, widgets, sensors, surveys and mobile apps, - might be used as input to the development of new Citizen Observatories in the future.

This part 2 is based on the previous D7.5 part 2. Platform Operation – and contains detailed information about the operational use of the data platforms, with connected sensor platforms and apps.

The CITI-SENSE toolbox that has been created in the final phase of the project, until M48, has focused on description of the various services that are in use in the current architecture and platform, in such a way that it can be considered for use in future Citizen Observatory projects. This is further described in D7.6 part 3.

The following annexes A-K shows the practical usage of the CITI-SENSE Architecture and Platform for various relevant sensor platforms and apps.

6 Annex A: Geotech with AQMesh

6.1 Introduction to usage context

6.1.1 Introduction

Geotech, in partnership with Envirologger, provides the commercial sensor platform AQMesh. AQMesh is a static, wireless air quality monitor system to measure the main air pollution gases. It works through a network of arrayed monitors to measure a number of gases.

Sensor data is periodically transferred to a secure, centralized server within the AQMesh network, where it is stored and is used for further dissemination.

6.1.2 Reference to CITI-SENSE Pilot work packages

The AQMesh sensor network is mainly used as a static sensor observation platform for measuring outdoor air quality in Work Package 2.

6.1.3 Reference to CITI-SENSE Locations

AQMesh has been deployed in the following CITI-SENSE pilot cities: Barcelona, Oslo, Edinburgh, Haifa, Ljubljana, Vitoria and Vienna.

6.2 Architecture and interfaces used

The following shows an architectural picture of the data flow related to this sensor platform.

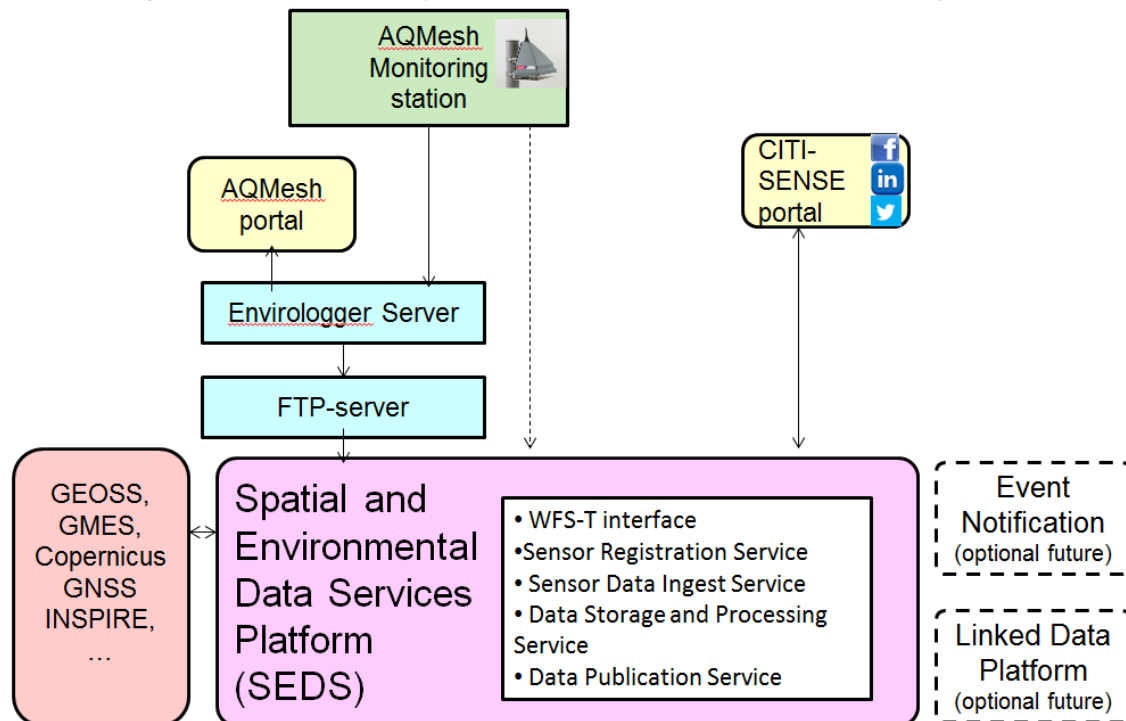


Figure 6-1 Data Flows for the CITI-SENSE Geotech AQMesh Sensor architecture

AQMesh monitoring stations make regular observations and send the resulting measurements to the AQMesh Server, using GPRS communication, where the data is stored on a secure FTP server for further dissemination.

It has been identified that the Data Ingestion component of the SEDS Platform requires a web service that connects to the AQMesh FTP service and pulls data into the SEDS Platform. This requirement and development has been designed using principle of adopting open standards for the data encoding and web service interfaces.

Data Collection (AQMesh)

AQMesh sensor devices (static pods) make a single observation at 15 minutes intervals. This data is stored on the pods and sent to the centralised AQMesh Server, via GPRS, on an hourly basis.



Figure 6-2 AQMesh station

Data Storage (AQMesh)

Within the AQMesh environment, the collected sensor data from the various AQMesh stations is stored on an internal, central AQMesh server. This server can be accessed by the SEDS Platform via a secure web connection.

For each individual sensor device, the data is stored as a CSV encoded ASCII file. Each file contains all observations over a 1-hour period for that specific sensor device.

```
Date,Time,Longitude,Latitude,1-NO Prescale - ppb,NO Slope,NO Offset,NO Final,NO Status,2-NO2 Prescale - ppb,NO2 Slope,NO2 Offset,NO2 Final,NO2 Status,4-CO Prescale - ppb,CO Slope,CO Offset,CO Final,CO Status,6-O3 Prescale - ppb,O3 Slope,O3 Offset,O3 Final,O3 Status,8-Temp-Celcius,9-RH-%,10-AP-mBar,210-Noise Level-Celcius,211-Noise Peak-%,200-Total Count-P/cm3,202-10 MC-ug/m3,203-2.5 MC-ug/m3,Flag,30/07/2015,20:00,10.767458,59.911752,-5.093,1.000,0.000,-5.093,Below LOD,36.955,1.000,0.000,36.955,,51.332,1.000,0.000,51.332,,10.502,1.000,0.000,10.502,,18.800,55.800,1001.600,688.000,805.000,,,,S
```



Data Ingest

Every hour a data ingestion web service on the SEDS Platform connects to the AQ Mesh FTP server and copies all existing CSV files to the SEDS Platform for ingestion into the database, which stores the sensor data.

The Data Ingest process consists of two components: 1) CSV Data Pull Service and 2) Data Ingest Service.

The 'CSV Data Pull Service' establishes a secure HTTPS connection to the remote AQMesh FTP server and copies all existing CSV datasets to the SEDS Platform. Here each dataset is converted to a standard HTTP POST Request that includes the sensor data as its payload.

This request is posted to the 'Data Ingest Service', which validates the request before ingesting the dataset into the relational SQL database, which forms the persistent data storage component.

The AQMesh Data Ingest Service uses a Python script to update the locations of particular sensors. For certain AQMesh sensors the location contained in the incoming data is the old/wrong location for the sensor. A simple CSV file contains the updated location values for the sensors. The Python script uses this CSV file as a lookup and updates the location in the payload that is included in the HTTP Post request. The updated locations for the sensors listed in the CSV file are not applied retrospectively on historic data. These are only applied to new data.

Data Storage (SEDS)

All ingested data is persistently stored in a centralised relational database. A single harmonised database schema has been designed and implemented in the database.

Data Publication

The stored data is made available to end-users through a number of web service interfaces. In particular, the SEDS Platform offers a standards-based Web Feature Service (WFS) that encodes datasets in the XML format.

In addition, a more light-weight REST interface is available, that provides datasets encoded in CSV or JSON.

Data Consumption

The data that is consumed from the SEDS Platform may be used in a variety of ways, such as visualisation in widgets and import in Excel.

6.3 Data (volume, velocity) being stored and retrieved

6.3.1 Data exchange between AQMesh and the SEDS Platform

The sensor pods measure five gases (NO, NO₂, O₃, CO and SO₂), plus temperature, atmospheric pressure and humidity. These are measured every 10 seconds and then constructed into 15 minute averages. On every hour the pods transmit this data to the AQMesh servers.

The sensor data is stored on a secure FTP server as CSV files, but also feeds into the AQMesh's proprietary web portal, AQMesh.net, for further analysis and visualisation to end users.

The Data Ingestion Web Service that resides on the centralised SEDS Platform polls the AQMesh FTP server at hourly intervals and pulls the latest available data into the SEDS Platform, processes it and stores it in the central database.

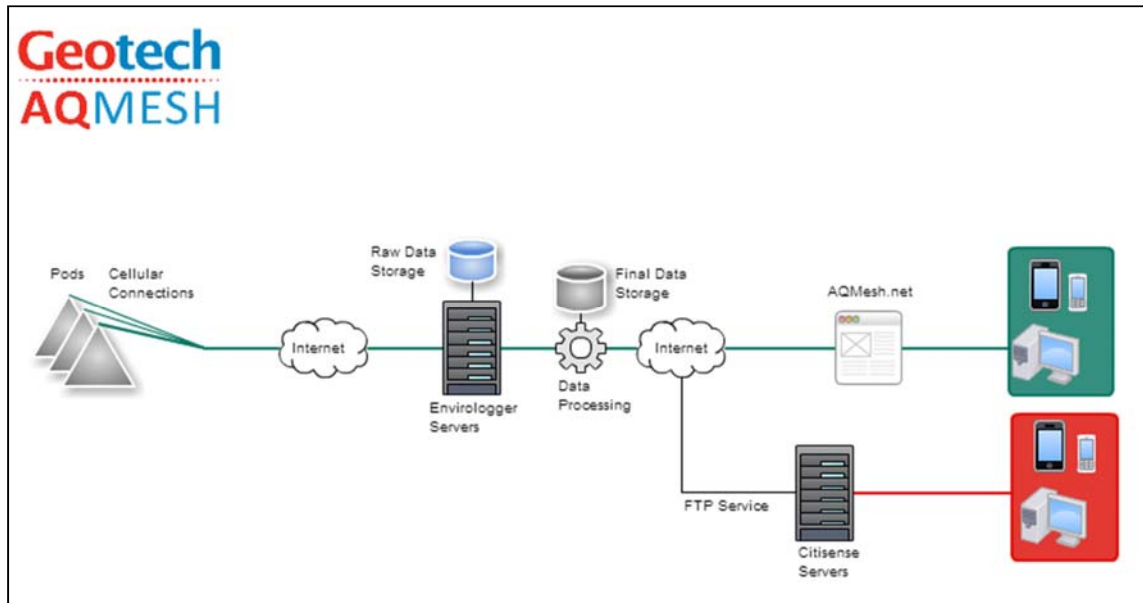


Figure 6-3 AQMesh data flow

6.3.2 Data volumes

It is scheduled that approximately 173 sensor pods will be distributed over the participating pilot cities. Based on this number, the expected total monthly data volumes will be over 8,500 observations, which is equivalent to approximately 175MB of data (in XML format).

6.4 User applications/apps/processing/visualisations

The AQMesh platform provides a web based data access facility: **AQMesh.net**. Via this secure web portal authorised users can access the data from different stations in their network.

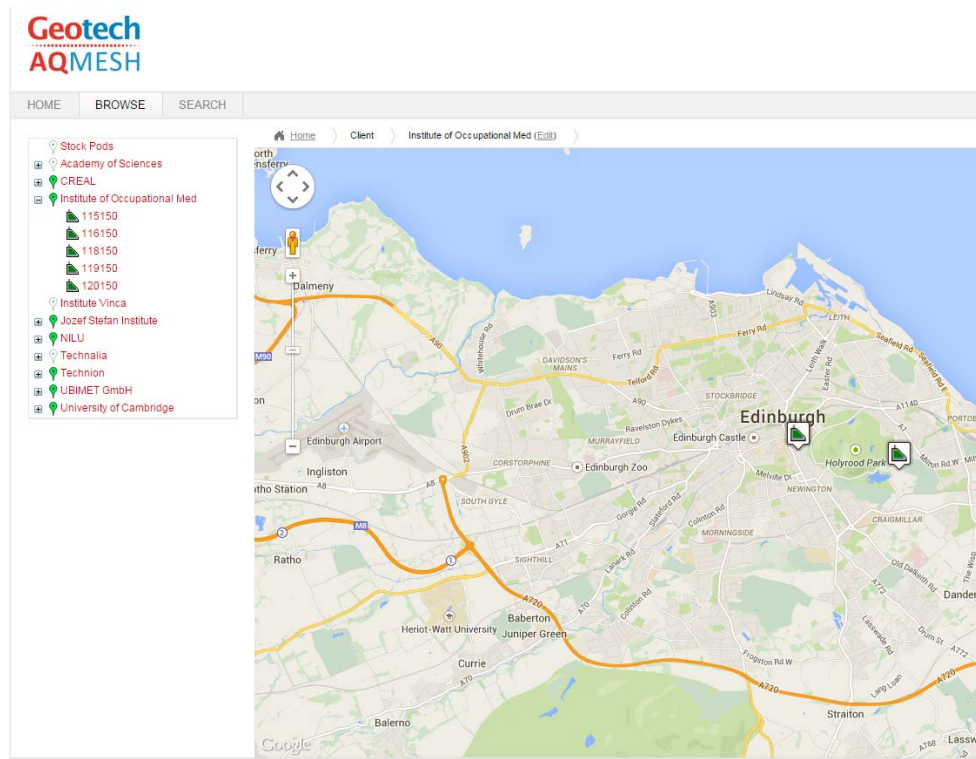


Figure 6-4 Screenshot of the AQMesh.net web portal

For each station, users can access, analyze and visualize the sensor data captured by that station. Graphs and tables can be created to show the sensor data over a period of time, such as the last 48 hours.

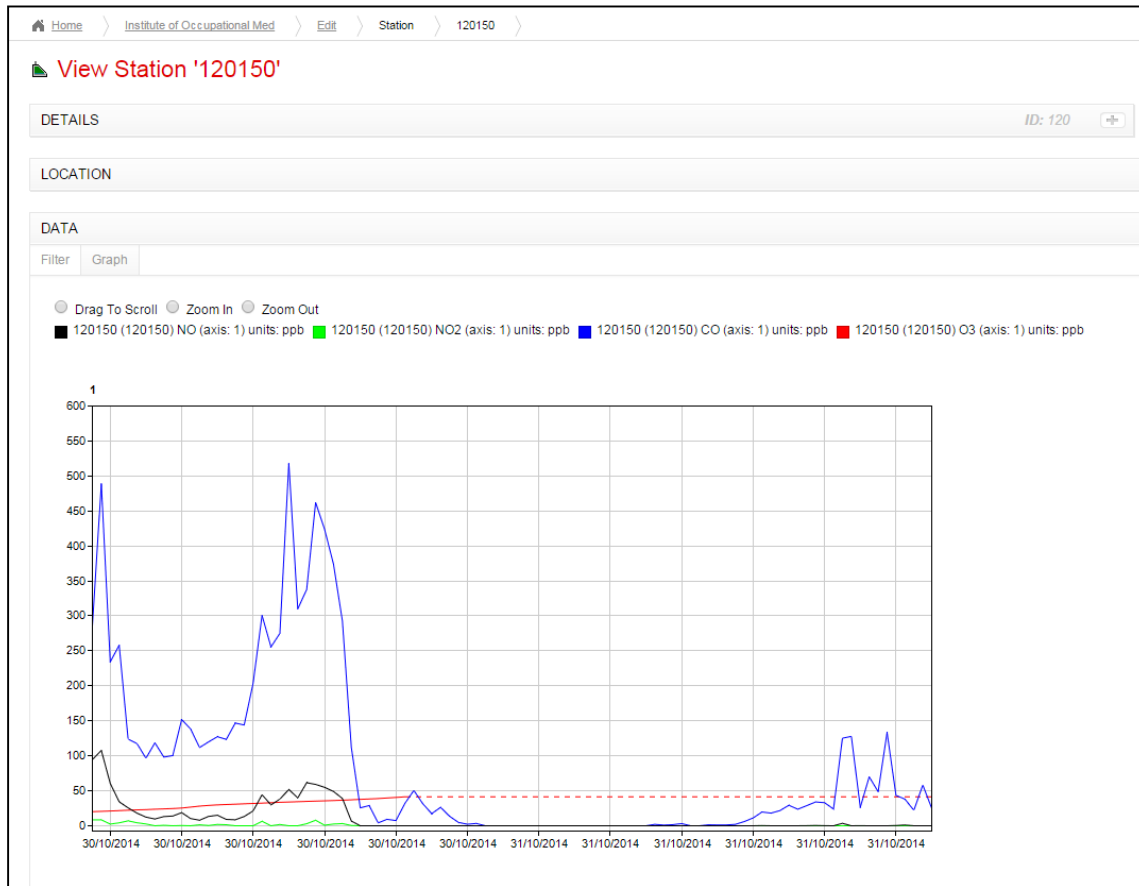


Figure 6-5 Displaying sensor data in a graph

AQMesh.net also allow users to export data selections to formats such as XML and CSV.

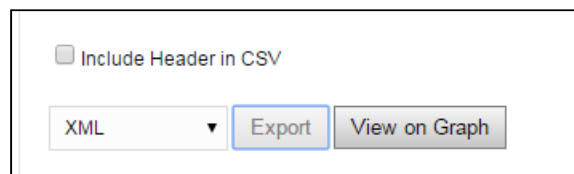


Figure 6-6 Exporting data

6.5 Suggested improvements and plans for next phase(s)

The plans considered for further development and evaluation of this for the next phases, i.e. for M48 is in particular to add support for measuring noise and particles in AQMesh.

7 Annex B: Atmospheric Sensors

7.1 Introduction to usage context

7.1.1 Introduction

Alphasense, in partnership with Atmospheric Sensors (ATMOS), provides sensor devices to CITI-SENSE participants. The sensor devices are mainly used to capture indoor air quality.

The ATMOS sensor devices capture the following data for each observation:

- NodeID
- Date and Time
- NO₂ concentration (ppb)
- NO concentration (ppb)
- CO concentration (ppb)
- O₃ concentration (ppb)
- PID (Volatile organic compounds - equivalent in ppb)
- CO₂ (ppm)
- Temperature (°C)
- Relative humidity (%)
- PM₁ (µg/m³)
- PM_{2.5} (µg/m³)
- PM₁₀ (µg/m³)
- Ambient sound level (not calibrated, for relative measurements)
- GPS location (latitude/longitude - only uploaded once per hour)

Every hour sensor data captured on the node is transferred to a secure, centralized server within the ATMOS network, where it is stored and used for further dissemination to the SEDS Platform.



Figure 7-1 Atmospheric Sensors node

7.1.2 Reference to CITI-SENSE Pilot work packages

The ATMOS sensor devices are mainly used as a static sensor observation platform for measuring indoor air quality in Work Package 3B Indoor Schools.

7.1.3 Reference to CITI-SENSE Locations

The sensor devices have been deployed in schools in the following locations: Oslo, Edinburgh, Ljubljana and Belgrade.

7.2 Architecture and interfaces used

The following shows an architectural picture of the data flow related to this sensor platform.

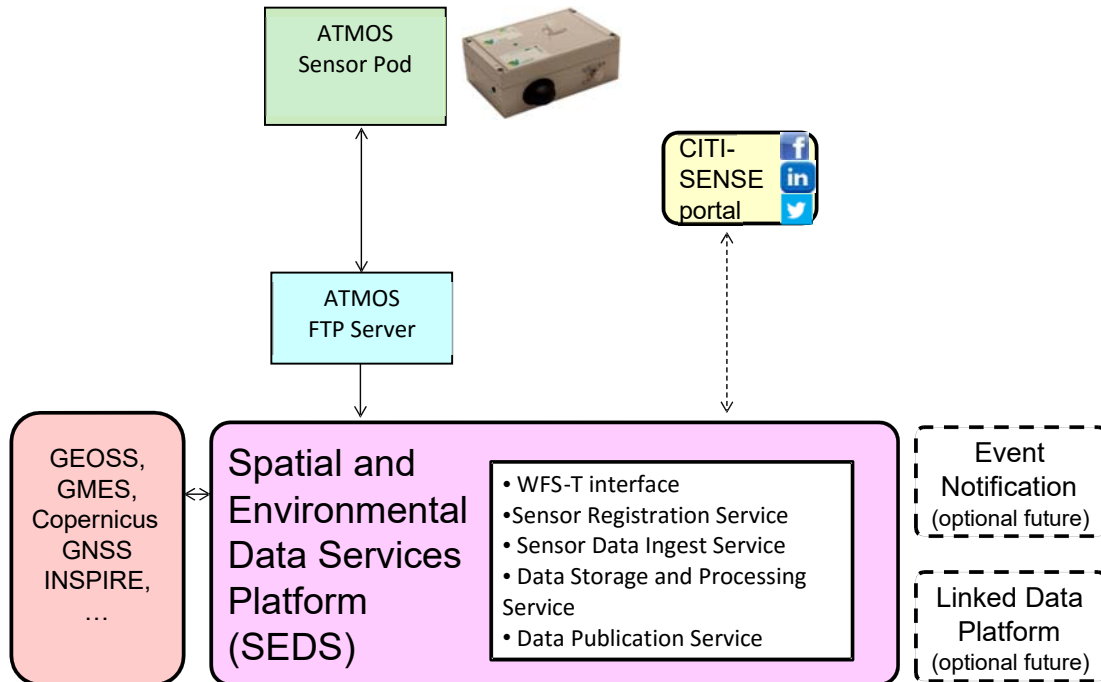


Figure 7-2 Data Flows for the CITI-SENSE Atmospheric Sensor architecture

7.2.1 Data collection (ATMOS)

The sensor devices from Atmospheric Sensors make an observation every 5 minutes. Every hour each sensor device sends the results of the observations via a GPRS connection to a central server at Atmospheric Sensors. Here the raw data is post-processed and the required data is extracted and converted to a file, which is stored on a secure FTP site at Atmospheric Sensors.

7.2.2 Data storage (ATMOS)

Each file contains the observations and measurements for a specific sensor device for a one-hour period. The data is encoded as comma separated values (CSV). Below is an example of the contents of such a file:

```
32,10/08/15,09:56:51,1,28,3,4956,-1729,3794,436,23.24,49.41,0.50,0.50,0.50,0.039139,22,0.00000,0.00000,0
32,10/08/15,10:01:51,2,37,57,1056,-1693,513,428,23.22,49.38,0.97,1.61,1.61,0.039520,22,0.00000,0.00000,0
32,10/08/15,10:06:51,3,29,45,683,-1132,0,428,23.24,49.35,0.79,0.79,0.79,0.039673,22,0.00000,0.00000,0
32,10/08/15,10:11:51,4,-168,39,534,-827,0,426,23.29,49.25,0.54,1.18,1.18,0.039825,22,0.00000,0.00000,0
32,10/08/15,10:16:51,5,-406,36,461,-657,0,423,23.36,49.14,0.11,0.11,0.11,0.039902,22,0.00000,0.00000,0
```



7.2.3 Data ingest (SEDS)

Every hour a data ingestion web service on the SEDS Platform connects to the ATMOS FTP server and copies all existing CSV files to the SEDS Platform for conversion and ingestion into the database.

The Data Ingest process consists of two components: 1) CSV Data Conversion Service and 2) Data Ingest Service.

The CSV Data Conversion Service establishes a secure HTTPS connection to the remote ATMOS FTP server and copies all existing CSV files to the SEDS Platform. After the files have been successfully copied, all CSV files on the FTP server are deleted.

Once the files have been copied to the SEDS Platform, the CSV Data Conversion Service converts each file to a HTTP Post request and posts it to the Data Ingest Service. The post request is an XML file that contains the observations and measurements from the original CSV file.

The Data Ingest Service is implemented as an open standard Web Feature Service (WFS) web interface. WFS is a standard web interface specification developed and managed by the Open Geospatial Consortium (OGC) and is also an ISO standard (ISO/DIS-19142).

The HTTP Post request is posted to the endpoint of the Data Ingest Service web service:

<https://prod.citisense.snowflakesoftware.com/wfst>

Apply corrections to CO2 values for ATMOS sensors (Section 6 Part 2 ATMOS)

Following analysis by WP3b, it was concluded that the ATMOS sensor data required a correction for CO2 values.

A correction formula was supplied. The correction formula was:

$$\text{NewCO2Value} = (\text{MeasuredCO2Value} + \text{OFFSET})/\text{SLOPE}$$

Each city provided the OFFSET and SLOPE values for CO2 for each node. The default value for OFFSET is set to 0 and the SLOPE to 1.

The following table lists all the sensor ids by city that have the CO2 correction factors applied. The table shows the OFFSET and SLOPE values for each sensor within each city. For all other ATMOS sensors the default values of OFFSET and SLOPE, mentioned above, are used.

Sensor Id	Offset	Slope	City
AT_3	-29.9084	1.0293	Oslo
AT_5	-55.5349	0.82432	Oslo
AT_6	-77.1947	0.83074	Oslo
AT_7	-47.6365	0.79157	Oslo
AT_8	-12.435	0.93143	Oslo
AT_9	-39.2376	0.93565	Oslo
AT_10	-22.4759	1.1084	Oslo
AT_11	-74.3624	0.95993	Oslo
AT_13	-66.7177	0.96443	Oslo
AT_14	-78.2772	0.87574	Oslo
AT_71	-3.1727	1.0676	Ljubljana
AT_72	-19.554	1.0571	Ljubljana
AT_73	-16.666	1.0798	Ljubljana
AT_75	-27.279	1.0977	Ljubljana
AT_77	-74.216	1.1126	Ljubljana
AT_78	-41.125	1.0596	Ljubljana
AT_79	-35.177	1.0785	Ljubljana
AT_80	-10.413	1.0859	Ljubljana
AT_81	-5.8694	1.0436	Ljubljana
AT_82	-33.165	1.1321	Ljubljana
AT_83	-12.447	1.0552	Ljubljana
AT_84	92.54723	1.405964	Belgrade
AT_86	-129.359	0.944334	Belgrade
AT_92	-83.8587	0.971562	Belgrade
AT_93	50.30511	1.176108	Belgrade
AT_94	25.26433	1.102721	Belgrade
AT_96	144.8644	1.598459	Belgrade
AT_97	-95.2308	0.965916	Belgrade
AT_98	-55.2177	0.945097	Belgrade
AT_99	340.2093	1.858132	Belgrade

A simple database lookup table was created with the above OFFSET and SLOPE values along with the sensor ids. The CSV Data Conversion Service reads the lookup table and applies the CO₂ correction using the formula mentioned above and then converts the incoming data into XML files that get posted to the Data Ingest Service.

The CO₂ corrections were being applied to new data captured by the ATMOS sensors and were not retrospectively applied to historic data.

7.2.4 Data Storage (SEDS)

All ingested data is stored and persisted in a centralised relational database. The database forms the data storage component of the SEDS Platform and is implemented as a PostgreSQL/PostGIS cloud



database instance on Amazon's AWS environment. The database stores data from all data providers in a single, harmonized database schema.

7.2.5 Data publication (SEDS)

The data ingested by the ATMOS sensor devices is made available to end-users through a number of web service interfaces. In particular, the SEDS Platform offers a standards-based Web Feature Service (WFS) that encodes datasets in the XML format. In addition, a more light-weight REST interface is available, that provides datasets encoded in CSV or JSON.

In addition to publishing the results of the observations made by the ATMOS sensor devices, the SEDS Platform also publishes the values for the Common Air Quality Index (CAQI)² and the Global Common Air Quality Index (Global CAQI). The CAQI value is calculated by converting the concentrations of a number of pollutants into one air quality index value. The higher the CAQI values, the worse the experienced air quality. The Global CAQI represents the highest (or worse) value of the individual CAQI values.

7.2.6 Data consumption

The sensor data from the ATMOS sensors is made available, through the Data Publication services on the SEDS Platform, to data consumers. Based on the consumer's requirements and application, data can be accessed via a variety of simple and complex web services in a variety of common data formats, such as XML, JSON and CSV.

7.3 Data (volume, velocity) being stored and retrieved

Each CSV file contains the observations and measurements for a 1-hour period. As the ATMOS sensor devices make an observation every 5 minutes, each file contains up to 12 observations.

A regular CSV file has a size of 1.2 KB.

From 1 September 2015, 64 sensor devices have been deployed to the participating pilot-cities in CITI-SENSE. With all 64 sensor devices activated, the total number of observations on a daily basis (24 hours) will be 18,432, which constitutes to a total file size of 22 MB.

7.4 Suggested improvements and plans for next phase(s)

The plans for further development and evaluation of this for the next phases, i.e. for M48 are as follows:

- If required, apply corrections on already ingested data.
- If required, increase frequency in which data is copied from the ATMOS environments into the SEDS Platform (currently once per hour).

² The CAQI was developed by the European Citeair project.



8 Annex C: OBEO MMA Radon and CO₂ Sensor

8.1 Introduction to usage context

8.1.1 Introduction

The OBEO MMA Wireless Radon and CO₂ Sensor make it easy to remotely monitor radon and/or CO₂ levels in indoor areas and buildings. The MMA utilizes the GSM/GPRS cellular network to relay the sensor data to a central server available from any PC/Mac/iPad. You may set up one MMA in one room/area or thousands in as many rooms/buildings as you require. Access to your wireless sensors is established through a secure login site. Ideal for building and property owners to monitor all their office buildings, schools, kindergartens, theatres and other indoor areas where people work, play, rest and live.

The radon sensor unit provided by OBEO has been tested and compared to different static radon sensors since M22 of the project. There has been some minor adjustments done regarding the firmware and at the moment the different cities are conducting a test locally in Oslo, Belgrade, Ljubljana and Edinburgh. The sensor data is transmitted via GPRS from sensor platform to the OBEO proprietary data server and then pushed to Snowflakes' server. GPRS communication is now working from all the cities.

The Obeo MMA makes it easy to remotely monitor radon and/or CO₂ levels in indoor areas and buildings. The MMA utilizes the GSM/GPRS cellular network to relay the sensor data to a central server available from any PC/Mac/iPad. Access to your wireless sensors is established through a secure login site.

Radon is a radioactive, odourless gas released from the normal decay of the elements uranium, thorium, and radium in rocks and soil. Radioactive particles from radon can damage cells that line the lungs and lead to lung cancer. The OBEO MMA is a device that can be considered as an electronic, automatic radon dosimeter. The MMA detects alpha particles from radon decay. A silicon semiconductor detects alpha particles that hit the surface of the sensor. The sensor is encapsulated by a positively charged housing (metering cell). Radon gas from the ambient air diffuses into the metering cell through a number of small apertures. These openings are covered by a filter keeping out dust and other radioactive particles. This filter also prevents radon decay products (daughters) present in the ambient air from entering the metering cell. In the metering cell radon decay will emit alpha particles. Radon atom decay produces the isotope Polonium-218. At its synthesis the Polonium-218 particle is ionized, holding a positive electric charge. The metering cell is held positively charged at a high voltage whereas the silicon sensor is grounded, thereby effectively depositing the particles on the surface of the sensor. The particles that are detected and registered by the sensor within a certain time indicate the radon concentration in the ambient air.



Figure 8-1 Picture of OBEO MMA

Figure 7-1 shows a picture of the enclosed unit of OBEO MMA, without a connected power adapter.

Table 8-1 Data characteristics of OBEO MMA

OBEO MMA	
Total Weight	510 g approx.
Dimensions	180x180x48.5 mm
Supply voltage range	250VDC
Accuracy/Resolution	0-5000 ppm, <50 ppm +/- 3%, 1 Bq/m ³
GSM Bands	850-1900 MHz, GSM/EGSM/DCPS/PCS
Ambient temp. range	+5 to +35 °C

Carbon dioxide (CO₂) is a gaseous component of the earth's atmosphere. The concentration of CO₂ in natural ambient air is about 0.04% or 400ppm. With each breath, humans convert oxygen (O₂) into carbon dioxide. Although carbon dioxide is invisible and odorless, an increased CO₂-content is apparent because humans will notice increased fatigue and reduced concentration. The CO₂-concentration is regarded as an important measure of indoor air quality. Non-Dispersive Infra-Red (NDIR) detectors are the industry standard method of measuring the concentration of carbon oxides. Each constituent gas in a sample will absorb some infrared at a particular frequency. By shining an infrared beam through a sample cell containing CO₂, and measuring the amount of infrared absorbed by the sample at the necessary wavelength, a NDIR detector is able to read the volumetric concentration of CO₂.

External sensors may be connected to the unit by means of an RJ11 expansion connector.

Also, the unit may be equipped from factory with other additional internal sensors as per request.

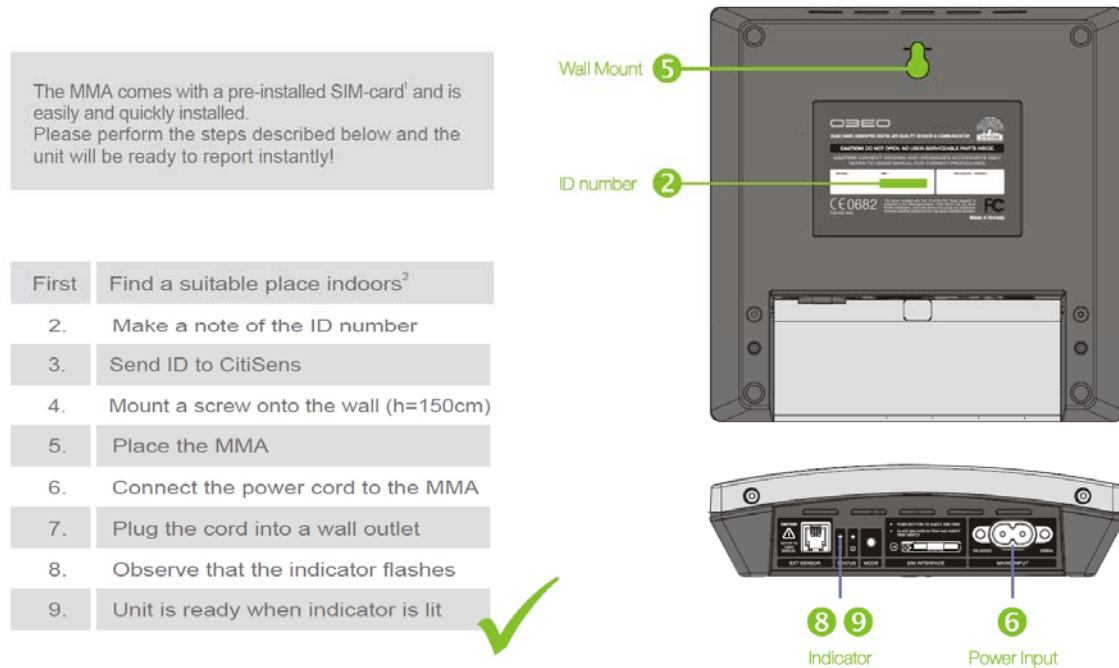


Figure 8-2 OBEO MMA Radon sensor installation process and connectors

With the OBEO MMA wireless GSM/GPRS sensor network platform, one can easily monitor the indoor air quality in a number of buildings with reliable sensor nodes that offer industrial ratings and local analysis capabilities. Each wireless network can scale from one to thousands of nodes and seamlessly integrate with existing databases and control systems.

8.1.2 Reference to CITI-SENSE Pilot workpackages

OBEO MMA is being further described and referred to in the following deliverables/work within the CITI-SENSE workpackages: WP7 D7.1, D 7.3., WP8 D8.1, D8.2

8.1.3 Reference to CITI-SENSE Locations

The **OBEO sensor** is used at the following CITI-SENSE locations: Oslo, Belgrade, Ljubljana and Edinburgh

8.2 Architecture and interfaces used

The following shows an architectural picture of the data flow related to this sensor platform.

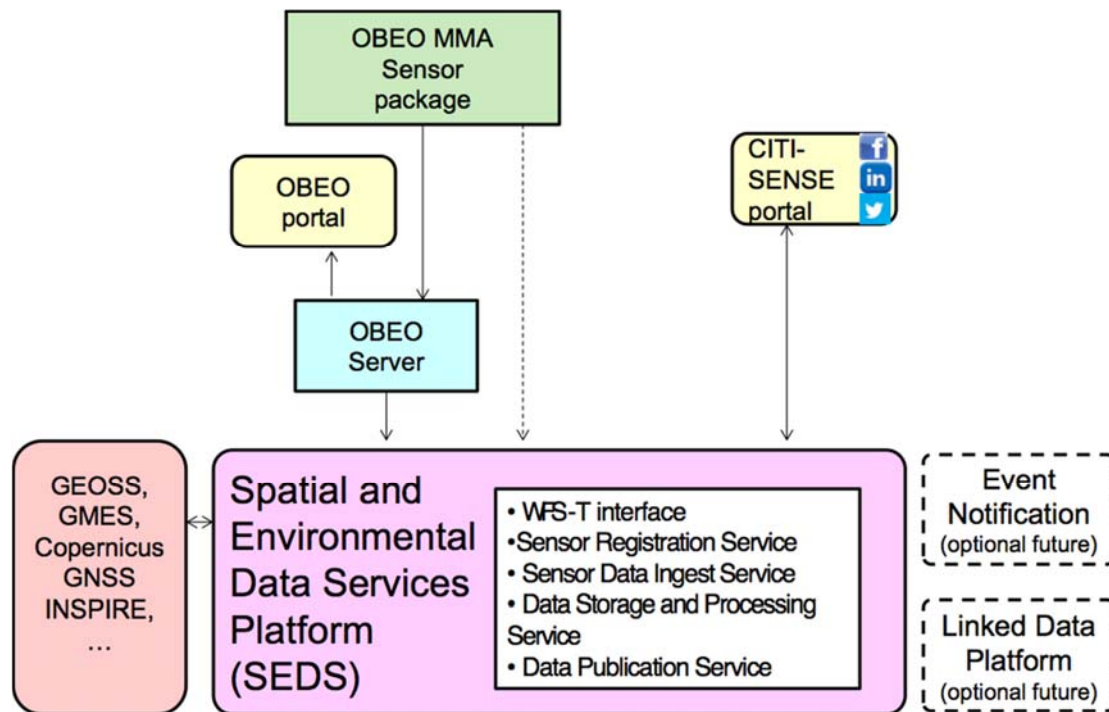


Figure 8-3 Data Flows for the OBEO architecture

OBEO establishes a GPRS connection with OBEO server, where the data are stored, processed and visualized. Users can log on to the web portal and access measured data,

The sensor data is transferred to the OBEO server directly from the device via GPRS and then pushed to the CITI-SENSE server using the appropriate web service. Ultimately, the data can be sent directly to the CITI-SENSE server at the later stage if required.

OBEO device data usage

Device sends raw data packed into tight formatted messages. This is done to save data cost.

8.2.1 OBEO server side infrastructure

The OBEO server side receives data from sensors via a XML protocol. All data is verified and posted onto the OBEO Oracle database.

8.3 Data (volume, velocity) being stored and retrieved

The data received from OBEO sensor platform for the CITI SENSE project is radon measurements.

Data are sent in XML format, and the length of data message is about 200 Bytes. Radon data is transferred once every hour. The time interval for other sensor data can be configured.

8.4 User applications/apps/processing/visualizations

As it is already explained the collected measurements are transferred to the back-end server via GPRS where they are stored and further processed. Data are stored in the centralized data storage for device position and sensors.



Application server is used for the visualization, administration and scheduling. The data can be visualized in a real-time using the appropriate Web application (portal) or mobile application. On the web portal, it is possible to see real-time and historical values. Measurements can be retrieved for specified defined period in the past. Visualization is done using third-party widgets.

The following data coming from the OBEO platform are stored on the server and can be viewed via WEB portal:

- Real-time sensor measurements
- Historical data (stored sensor measurements over the period of time)
- Graphical data presentation
- Data available as downloadable Excel file
- Data export in XML format available for different sensors

IMEI: 359394054128537

Radon: 320 Bq/m³, Temp.: 20° C. (68° F.)
 Last measured: 15-09-2015 at 15:06. Battery: .08 Volt. Hours counter: 4870.

	Last day		This month		Period: 2015	
	Radon	Temp.	Radon	Temp.	Radon	Temp.
Max.	560	20°	720	20°	1280	20°
Avg.	235	19,6°	243	19,9°	118	19,9°
Min.	80	15°	0	15°	0	15°



Figure 8-4 Visualisation of Radon and Temperature data



8.4.1 Suggested improvements and plans for the final phase

Radon level in indoor air will change with change in ventilation and outdoor temperature. On this background it is interesting to calculate radon level in schools when populated and when closed as two different values. This will show the effect of the ventilation system and the real radon level.

It is possible to do this based upon the hourly data stored in the system and to aggregate a value for opening hours and another for the time the building is out of use.



9 Annex D: Personal Sensor Platform with LEO

9.1 Introduction

This document extends the contents of previous version by adding information related to new Little Environmental Observatory (LEO) device introduced by Ateknea and its corresponding application. It is the upgrade of the initial CITI-SENSE PSP app.

The new Android app is called ExpoApp2, and both LEO and ExpoApp2 are related to the CITI-SENSE Architecture and platform in this annex.

9.2 Citisense PSP

CitiSensePSP is a mobile application that runs on Android devices. This application communicates with the Ateknea's Personal Sensor Pack (PSP) and sends the data to the SensApp Server. The PSP is composed by 5 different sensors: temperature, relative humidity, O₃, NO₂ and CO. The data obtained from these sensors is processed and then sent to the server, where it can be consulted with only some minutes of delay (depending on the configured upload frequency within the app).

9.2.1 Reference to CITI-SENSE Pilot workpackages

During the pilot studies, 20 personal sensor platforms developed by ATEKNEA were delivered to the EI cities related to Urban Quality. Barcelona, Edinburg, Haifa and Oslo were the cities involved in the pilot studies for Personal Sensors were. Each city received 5 personal sensor platforms (PSP) to be evaluated. The PSP were sent along with an Android app in order to control and read data from the PSP. More information about the pilots and units used can be found in D8.2.

9.2.2 Reference to CITI-SENSE Locations

For full deployment it is expected to deliver a total of 86 units, this time including Belgrade, Ljubljana, Vienna and Ostrava. This time, 8 different cities will participate in the deployment of the personal sensor platforms. A new version of ATEKNEA's PSP with smaller form factor and more reliable Bluetooth communication will be delivered.

During the pilot studies, several challenges have been identified for the personal units, that is the effect the relative humidity in the environment and temperature changes can modify the response of the gas sensors. Also, the time it takes for the sensor to adapt to a certain environment once it has been powered. Some mitigation features will be included in future versions of the app. More information about the distribution and features of the units to be used during full deployment can be found in D8.3

9.3 Architecture and interfaces used

The following shows an architectural picture of the data flow related to this sensor platform.

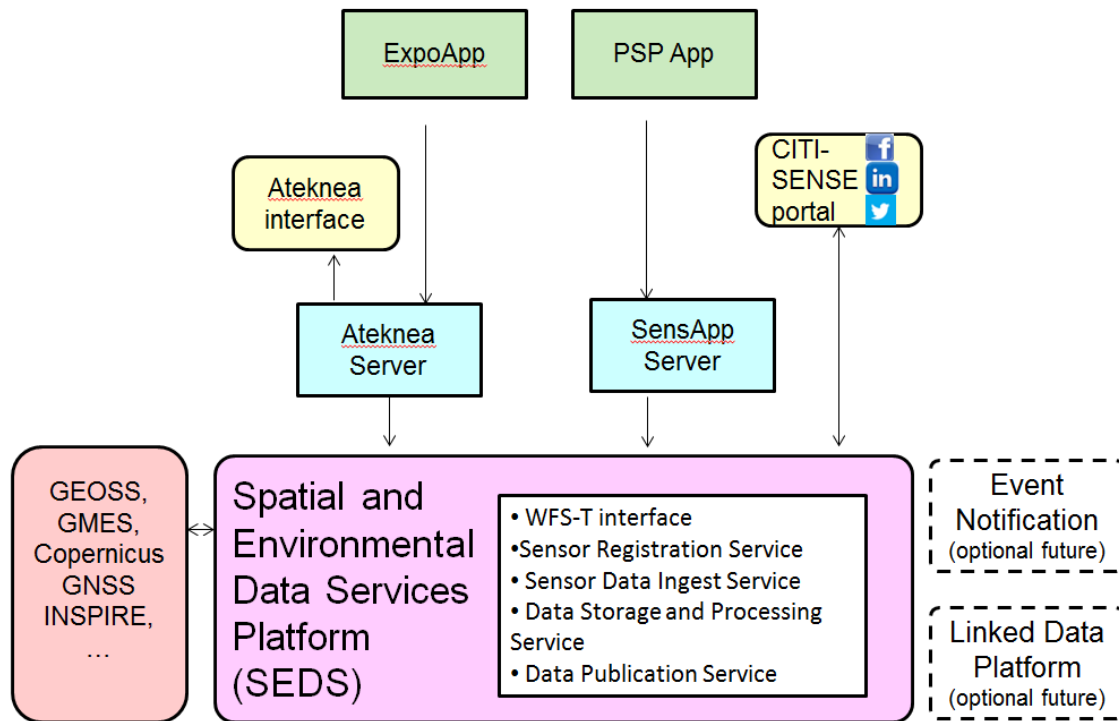


Figure 9-1 Data Flows for the CITI-SENSE PSP App and ExpoApp

In general terms, CitiSensePSP establishes a Bluetooth communication with the Ateknea Personal Sensor Pack (PSP) and reads all the data gathered from the gas and ambient sensors. This data is then processed by a specific Ateknea module and then saved in the smartphone's SD Card. Finally, the processed data is sent to the SensApp server where it can be visualized.

More specifically, the mobile application performs three differenced actions:

- Establish communication with the PSP through a Bluetooth socket, following a proprietary protocol defined by Ateknea to send commands and receive data from it.
- Process the data received from the PSP with the specific process module, and subsequently save it on the SD Card in different formats (raw data, processed data and data format to be sent to the server). Also there is a protocol log saved in the internal memory. This log contains the communication messages between the smartphone and the PSP (used for debugging purposes).
- Simultaneously, the data is uploaded to the SensApp server with a frequency chosen by the user. This action also contains the necessary initial sensor registration (with the associated and unique participant ID).

All this process is transparent for the user.

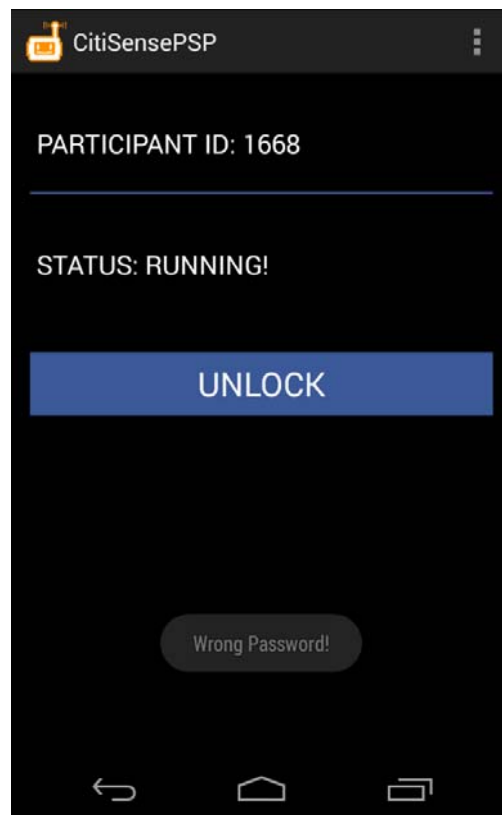


Figure 9-2 User Interface - monitoring

9.4 Data (volume, velocity) being stored and retrieved

The volume of data that is being collected during one sample of measurements is the following: The data received from the PSP consists of hexadecimal strings, which are processed to obtain the decimal values for the 5 sensors (temperature, relative humidity, O₃, NO₂ and CO).

The data is being uploaded to the CITI-SENSE WFS at the following intervals: the application can upload an average of 130 samples every 5 minutes

The data received from the PSP consists on hexadecimal strings, which are processed to obtain the decimal values for the 5 sensors (temperature, relative humidity, O₃, NO₂ and CO). This decimal data, as well as the raw data, is sent to the SensApp server following an accorded format. Prior to the data sending, it is necessary to register each one of the sensors with the associated unique participant ID as well as the PSP id. This will allow to identify uniquely the data on the SensApp server.

Regarding the volume and velocity of data being stored and sent, it depends on the circumstances where the platform is running. In the case of the Bluetooth communication, the data is being read every 5 minutes (protocol convention). This is because the PSP samples each sensor every 2 seconds and provides an average of 15 samples every 30 seconds. The data upload frequency is chosen by the user in a range between 5 minutes – 3 hours or never. It is important to state that only the new data is sent to the server. Also, the data upload depends directly on the network status and velocity. As a

summary of this section, it has been calculated that in normal conditions the application can upload an average of 130 samples every 5 minutes (considering each sensor value per separated).

Finally, it is necessary to state that there is no retrieved information in this application, because users use the SensApp interface to see the data uploaded. The SensApp services used by the applications are:

- Sensor registration endpoint.
- Data uploading endpoint.

9.5 User applications/apps/processing/visualisations

The interface used on this solution consists only on a button to start the service, which starts the three actions previously commented. Additionally, there is the option to set some user settings, such as the server URL or the upload frequency.

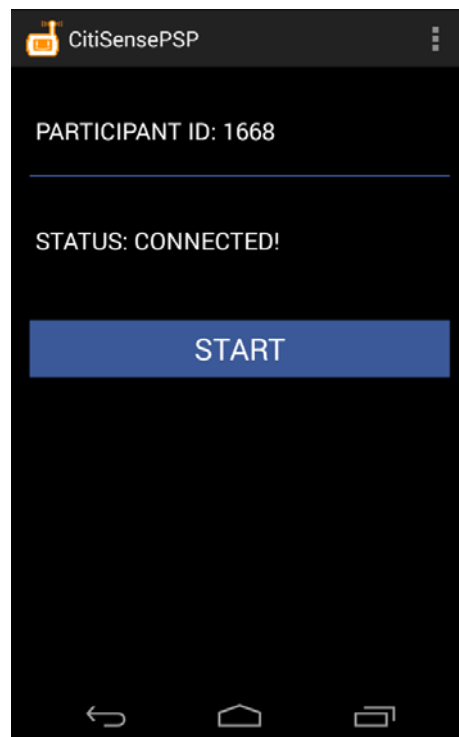


Figure 9-3 User Interface - Connected

Finally, if the involved cities decide it, they can give a password to the users that will unlock the live data user interface part, where there are showed the last values read and processed for each sensor. It is important to state that this live data interface only has the purpose of giving feedback to the user, and only shows the last values reads, so that can lead to some temporary differences with the SensApp server values.

As mentioned before, the users will have the CitiSensePSP Android application, which will allow them to start the services. These services include the data processing, with a specific module that converts the raw data from the PSP to readable and filtered values.

Regarding the visualizations, it is possible to difference two ways:

- Data uploaded through the SensApp interface. For each PSP there are 13 sensor registered (taking into account values raw and processed). These measures can be consulted in a table format, with the date and corresponding value. There is an extra possibility to visualize the processed values on this platform, since these are sent in decimal format making possible to see the data in a graph.
- Last sensors data read from the PSP through the CitiSense PSP Android application interface. This is used to provide direct feedback to the users, but it is necessary to state that this may have some difference with the last values of SensApp (there is a delay due to upload frequency).

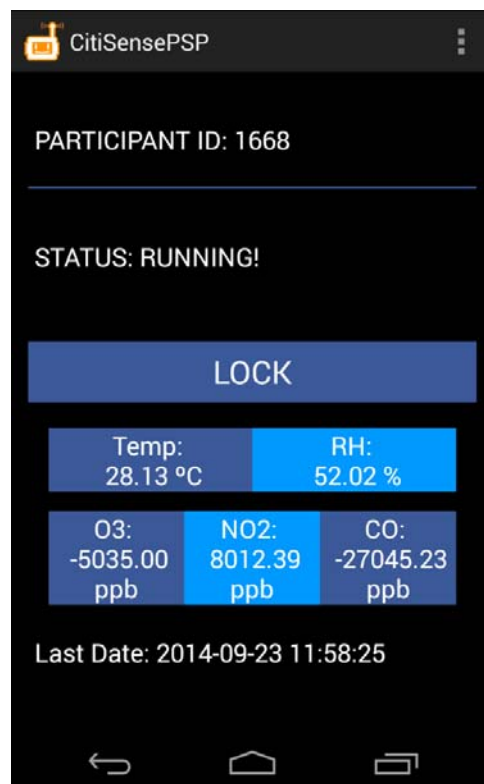


Figure 9-4: last captured sensor values

9.6 Suggested improvements and plans for next phase(s)

The development of this PSP app completed and version 1 has been released during the pilot studies. Different end users in the EI - Urban Quality are using the app and the first round of feedback became available at the beginning of the 3rd period.

Sensing & Control has collected feedback from end users and cities (through WP8 and WP6) and have further been defining the "service" that will provide direct and/or simplified information to end users about the environmental values collected.



At the end of the definition of the service, the functions defined in the service has been integrated in the current PSP app or additional application will be developed. This service has two different graphical user interfaces, one for the smartphone app, and another using a web browser. The former is a simplified version of the second one, due to the limitations of the screen size.

Among the different ideas for the service, we can highlight:

- 1) Plot historical values (useful for technicians) in ppm, °C and %RH
- 2) Show sensor values (useful for technicians) in ppm, °C and %RH
 - a. Current
 - b. Average
 - c. Min
 - d. Max
- 3) Show “environmental dose” based on activity (useful for end user) – value and/or colour (green, yellow and red – or other schema)
 - a. Hourly
 - b. Daily (maybe embedded in a calendar type interface)
- 4) Show “environmental dose” graph (useful for end user and technicians)
 - a. From date1 to date2 (default today)

Plot activity-environmental values on maps (useful for technicians and end users)

9.7 GPS/ACC Solution – Sensing & Control (with Ateknea)

9.7.1 Introduction to usage context

CitiSense is a mobile application that runs on Android devices. CitiSense reads all the data generated by ExpoApp and sends it to the SensApp server. This data is basically the ACC values (values for the three axis) and the GPS position encrypted.

9.7.2 Architecture and interfaces used

CitiSense is a service which is woken up every once in a while and reads the new available GPS and ACC data generated by ExpoApp on the SD card. The structure of the directory is set by ExpoApp as it's the application which leaves the data there.

CitiSense registers all the needed sensors for the concrete participant in SensApp server automatically; 3 for the accelerometer (X, Y, Z) and 2 for the GPS (latitude, longitude). Once the sensors are registered, it reads the data and sends it to SensApp.

All this process is transparent for the user. The only action he should do is to start the service. The interface used on this solution consists on the information screen, where some of the parameters are shown, and the menu, where the user can start/stop the service.

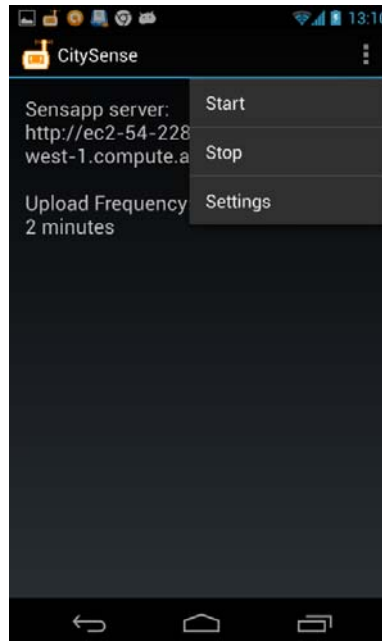


Figure 9-5: Configuration

9.7.3 Data (volume, velocity) being stored and retrieved

The data read from the ExpoApp folder consists on decimal values for the ACC sensors and encrypted values for the GPS. Prior to the data sending, it is necessary to register each one of the sensors with the associated unique participant ID. This will allow to identify uniquely the data on the SensApp server.

Regarding the volume and velocity of data being stored and sent, it depends on the circumstances where the platform is running. The data upload frequency is chosen by the user in a range between 5 minutes – 3 hours or never. It is important to state that only the new data will be sent to the server. Also the data upload depends directly on the network status and velocity. As a summary of this section, we have calculated that in normal conditions the application can upload an average of 21.600 ACC samples and 600 GPS samples, every 5 minutes.

Finally, it is necessary to state that there is no retrieved information in this application, because users use the SensApp interface to see the data uploaded. The SensApp services used by the applications are:

- Sensor registration endpoint.
- Data uploading endpoint.
- Encrypted data uploading endpoint.

9.7.4 User applications/apps/processing/visualisations

As mentioned before, the users will have the CitiSense Android application, which will allow them to start the service. This service include the data reading from the files created by the ExpoApp application, and also the data uploading to SensApp Server.

It is very important to state that due to privacy reasons, the GPS position is encrypted, and the CitiSense application sends encrypted data. The data decode is performed in the SensApp server.

Regarding the visualization, it is only possible to visualize the data through the SensApp interface. For each participant there are 5 sensors (3 ACC + 2 GPS), that can be consulted in a table format, with the date and corresponding value.

9.7.5 Suggested improvements and conclusions

From Sensing&Control there is no request for future work at the end of the project, but further enhancements to the services might be foreseen related to new versions of the sensor platform. The service that use GPS and Activity are already explained in section 10.4.

9.8 LEO Sensor Data Flow

LEO stands for Little Environmental Observatory. LEO device has been developed by ATEKNEA partner in WP8. LEO targets WP2 – Urban Quality, and provides sensors to capture NO (nitrate oxide), NO₂ (nitrate dioxide) and O₃ (ozone). As the device is target to be carried by an end user, it is used in combination with an Android application called ExpoApp2, a merge between former Sensing&Control's CitisensePSP and Ateknea's ExpoSomics android app. ExpoApp2 connects via Bluetooth channel with LEO, and combines smartphone's position and physical activity (calculated using accelerometer sensors) information with NO, NO₂ and O₃ from LEO device.

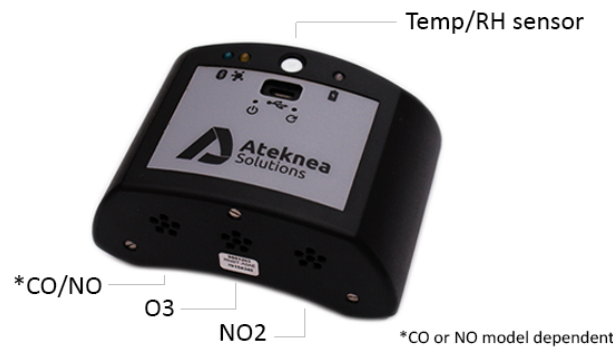


Figure 9-6: LEO Device

More information about the LEO sensor can be found at <http://citisense.ateknea.com/>

The data is being uploaded continuously to the CITISENSE SEDS server.

As mentioned above, the LEO sensors can act as a data logger by itself. Nevertheless, in order to engage a real time communication, a smartphone is used as a gateway. The LEO communicates with the smartphone via Bluetooth.

In order to reduce the amount of data uploaded by the smartphone, only RAW data is sent to the ATEKNEA server. This data is then post- processed and sent to the CITISENSE SEDS platform via a WFS requests. The entire chain is shown in the figure below.



Figure 9-7: Dataflow for LEO device

This architecture was decided over a direct transfer to WFS for the following reasons:

- The data transmitted (JSON format) on the ATEKNEA server is around 10% of the size of the XML transaction needed to send the same amount of data directly to WFS.
- The post processing algorithm is evolving. Hence, only one implementation of the algorithm will be running on the ATEKNEA server and being transparent to the version of ExpoApp2
- The entire RAW data is being saved on the server which allows to re-apply the post processing algorithm if necessary.

In order to provide help on the use of the LEO sensor and related applications, several video tutorials have been posted online. These can be accessed following the next links:

1. How to install the APK <https://www.youtube.com/watch?v=gFALRa3vhTM>
2. How to use with the sensors https://www.youtube.com/watch?v=cLdi_ATVpMM
3. How to Stop a session <https://www.youtube.com/watch?v=grdDn-J3Rmc>
4. How to Kill ExpoApp2 if something goes wrong <https://www.youtube.com/watch?v=-PLoA8PWwyM>

9.8.1 ExpoApp2

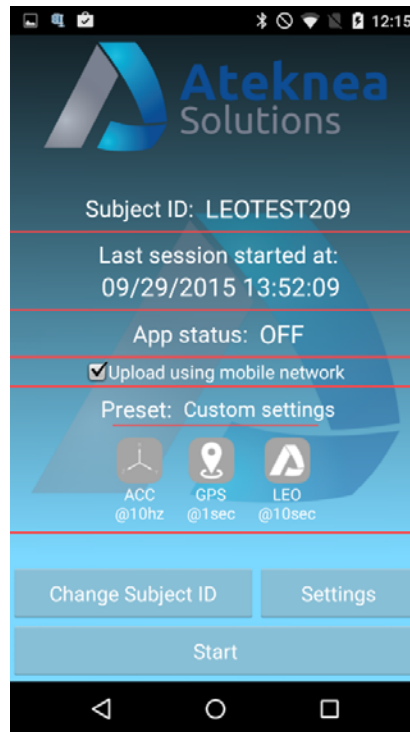


Figure 9-8 ExpoApp2 Main screen

ExpoApp is a mobile application that runs on Android devices; initially written and tested on a Samsung Galaxy SIII running Android 4.1.2, but tested successfully on a Samsung Galaxy Y running Android 2.3.6 and a Samsung Galaxy Nexus running Android 4.2.2. ExpoApp logs location and physical activity information throughout the day.

9.8.2 How does ExpoApp work?

ExpoApp uses the GPS receiver and accelerometer built in the smart phone to track physical activity and the participant. This information is stored in the smart phone SD card or in the internal storage (in case that there is no SD card). Each sensor (Location, Accelerometer, Altimeter, etc.) stores the information in a different file within a folder called “ExpoApp”.

9.8.3 How to Start a New Collection

To start a new collection, open ExpoApp touching the icon on the home screen. Go to settings to “Configure your ExpoApp Session”. You can also select the default profile, which will automatically define the parameters for the different sensors.

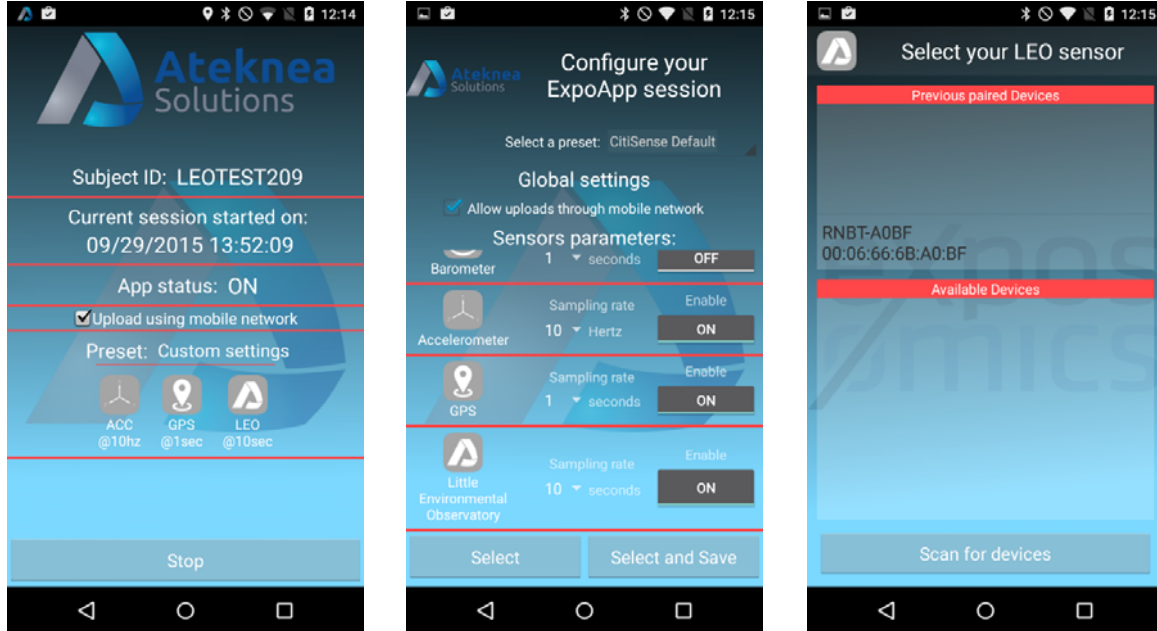


Figure 9-9 ExpoApp2 configuration interface

Then you will have to select the desired LEO sensor, just by tapping twice on the sensor icon. Scan the available Bluetooth devices and select the correct one.

Lastly, you can Change your Subject ID to something meaningful. When you are done setting up you session you can hit on Start and the new session will start.

9.9 Suggested improvements and conclusions

The LEO devices and associated applications (including client apps for AQI, sensor levels and activity [03]) have been provided to partners in charge of Urban Quality (WP2). We are providing troubleshooting on the use of the different applications, and will wait feedback for potential improvements and enhancements.

Bug correction will be done as soon as requested by end users. Nevertheless, continuous test on lab and real deployments managed by Sensing & Control and Atekneas are ongoing to detect potential bugs.



10 Annex E: LEO air quality measurement integration with enControl

10.1 Introduction to usage context

10.1.1 LEO device and Exposomics app

LEO stands for Little Environmental Observatory. LEO device has been developed by ATEKNEA partner in WP8. LEO targets WP2 – Urban Quality, and provides sensors to capture NO (nitrate oxide), NO₂ (nitrate dioxide) and O₃ (ozone). As the device is target to be carried by an end user, it is used in combination with an Android application called ExpoApp2, a merge between former Sensing & Control's CitisensePSP and Ateknea's ExpoSomics android app. ExpoApp2 connects via Bluetooth channel with LEO, and combines smartphone's position and physical activity (calculated using accelerometer sensors) information with NO, NO₂ and O₃ from LEO device.

The data is being upload continuously to CITI-SENSE main servers.

More information about can be found at <http://citisense.ateknea.com/>

10.1.2 enControl™

EnControl™ is a white label B2B product by Sensing & Control Systems providing a smart home solution. At a glance, EnControl has four main functions:

- 1) Comfort
- 2) Security
- 3) Energy Management
- 4) Automation

Users of EnControl are able to monitor and control remotely their homes & business using smartphones or any device running internet browsers like tablets or PCs.

EnControl can be divided into three main components, (i) home devices, (ii) IoT Platform and (iii) graphical user interfaces (as shown in the next figure).

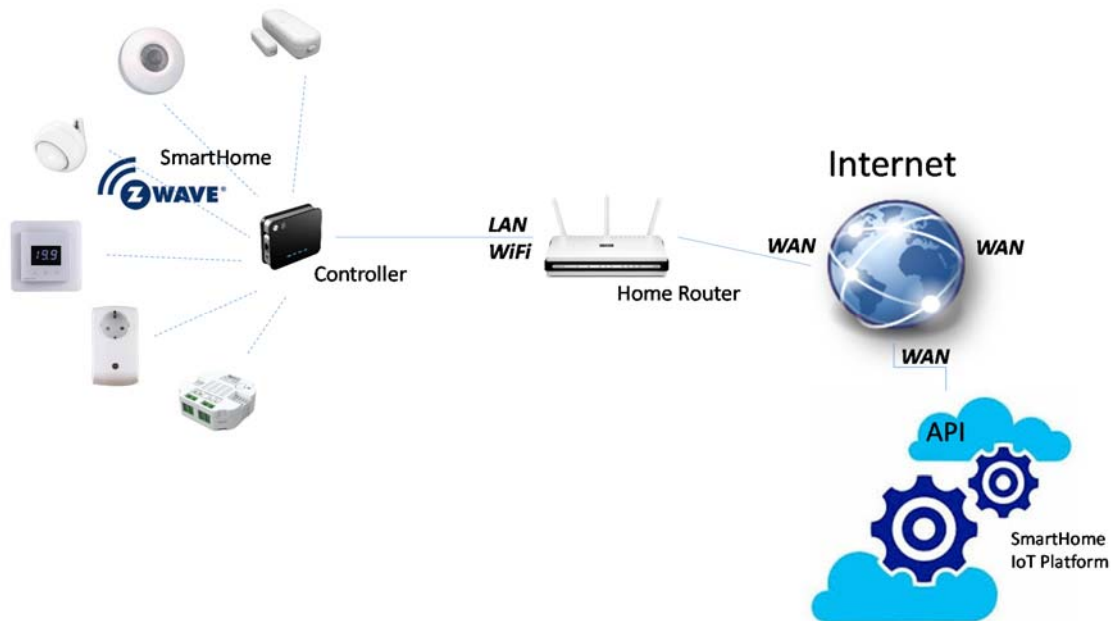


Figure 10-1: Smart home architecture

At home level, EnControl is composed by (i) end devices (sensors and actuators) and (ii) home controller (Gateway). The gateway supports communication using different protocols, but mostly uses ZWAVE radio technology for the communication of off-the-shelf end devices and home controller.

There are more than 300+ companies providing ZWAVE more than 1300 products³ which enable enControl™ to deliver the four main functionalities highlighted above, which are extended in following short list⁴ (from the point of view of the information being triggered by the end devices):

- 1) Comfort
 - a. Climate monitoring
 - b. Climate control
 - c. Temperature, Humidity, CO2 (etc...) levels
- 2) Security
 - a. Detection of door/window opening
 - b. Detection of movement
 - c. Detection of Smoke
 - d. Detection of CO
 - e. Detection of water basement
- 3) Energy Management
 - a. Energy consumption
 - b. Energy control (switch on/off electricity, water, gas, etc...)
- 4) Automation
 - a. Switch on/off appliances
 - b. Switch on/off lights
 - c. Open/Close doors, curtains, shutters

³ According to <http://z-wavealliance.org>

⁴ The list does not pretend to cover all possibilities; the Reader should understand that the smart home solution can integrate any ZWAVE standard product, thus enabling the functionality delivered by a particular product. For full list of product, please visit ZWAVE alliance web page.

The IoT platform contains the core of the smart home solution. It provides an open REST API enabling the home controllers to exchange information bidirectionally, based on synchronous or asynchronous actions triggered by IoT or end users through EnControl interfaces.

The IoT platform serves as well as main repository of information, keeps historical information about end devices data (information and status) as well as basic actions triggered by different actors, so users of the smart home solution knows in real time who-when-what of actions monitored.

The open REST API is used by user interfaces through web clients and smartphone apps in order to represent to end users the information being acquired from home end devices, and the action that can be triggered to them, so it encapsulates the smart home functions offered.

It is important to notice that smart home clients (smart home apps and web interfaces, 3rd party services, etc...) are able to interact with the smart home through the API.

The main home controller target is to timely exchange information between devices and IoT platform (and thus with end users), and information being generated by the home controller itself. However, additionally it provides three important functions:

- 1) Management of ZWAVE network and devices
- 2) Basic pre-processing capabilities (mainly energy calculations)
- 3) Local information repository in order to deal with temporal internet cut offs (so historical information is kept to be sent when internet connectivity is resumed).

The current smart home solution provides an intuitive and friendly user interface, which is similar for tablets and PCs (web interface) and smart phones. Figure 10-2 shows the look and feel of the dashboard, (a) for web browser and (b) for smart phone.

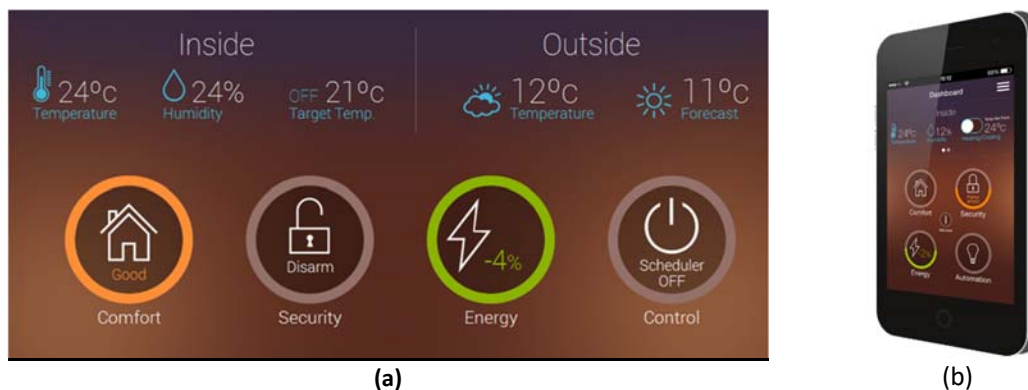


Figure 10-2: enControl™ Dashboard

10.2 Architecture and interfaces used

The following shows an architectural picture of the data flow related to the integration of LEO devices within S&C's Smart Home solution.

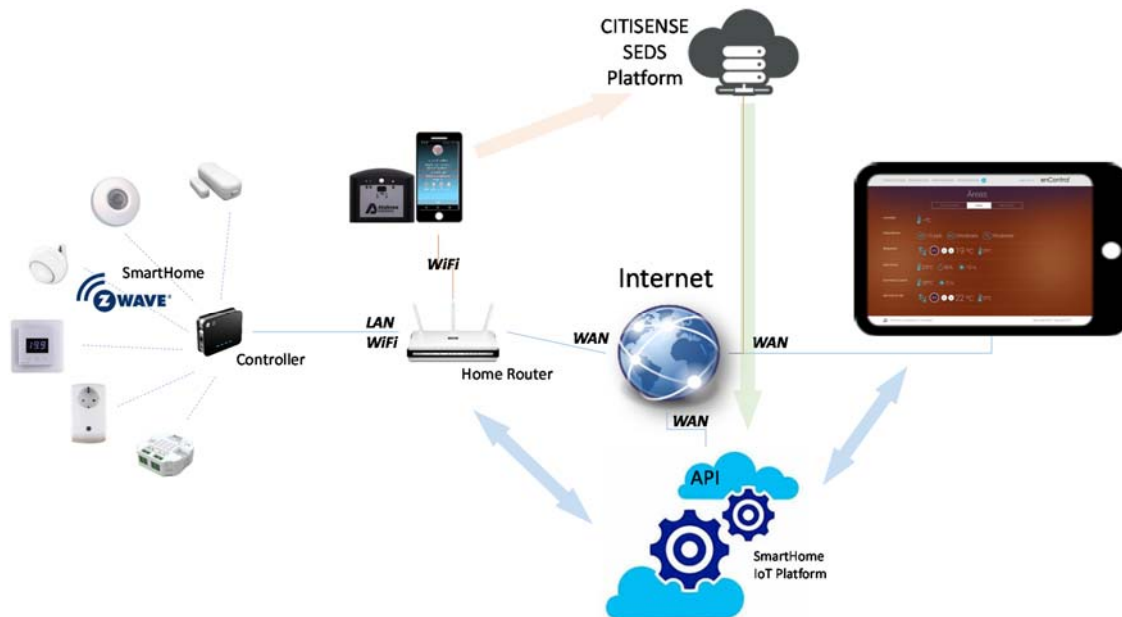


Figure 10-3 Data Flow Architecture for the LEO device, SEDS and enControl™

In general terms, LEO is used as static sensor within the home which is installed. A smart phone with ExpoApp2 application installed is used to upload data to CITISENSE SEDS platform in real time. The LEO device is capable to act as a data logger, so data being stored in device memory is upload once the smart phone connects with it. In this second case, enControl™ will not show data in real time, while in the former case the data will be shown to user in real time.

The smart home IoT platform (backend) has been upgraded to connect at data base level and web services and functions. The database has been extended to support air quality features, thus the platform allows storage of raw data and air quality index values. Thus now enControl™ is able to store CITI-SENSE air quality measurements and keeps an historical information that will be able to display to end users through web or smartphone clients.

A service in the SmartPhone IoT platform fetch data from CITISENSE SEDS platform at a predefined intervals. The interval has been fixed to 5 minutes, and we load the last values pushed by ExpoApp2 to SEDS. In normal case, real time, the service loads the last five measurements, as LEO acquires one sample every minute, and ExpoApp2 uploads them every 5 minutes.

In order to download data from the LEO sensors installed at home from SEDS, the worker uses the LEO's ID that the user can find in a label stick on the device, and uses following web services:

[https://prod.citisense.snowflakesoftware.com/wfs?service=wfs&version=2.0.0&request=GetFeature&typename=cts:Observation&filter=<Filterxmlns='http://www.opengis.net/fes/2.0'xmlns:cts='http://www.citi-sense.eu/citisense'><And><PropertyIsEqualTo><ValueReference>cts:sensorID/@xlink:href</ValueReference><Literal>"+LEOSENSORID+"</Literal></PropertyIsEqualTo><PropertyIsBetween><ValueReference>//cts:finishtime</ValueReference><LowerBoundary><Literal>"+LastDateValueUtc.ToString\("s"\)+"</Literal></LowerBoundary><UpperBoundary><Literal>"+DateTime.UtcNow.ToString\("s"\)+"</Literal></UpperBoundary></PropertyIsBetween></And></Filter>](https://prod.citisense.snowflakesoftware.com/wfs?service=wfs&version=2.0.0&request=GetFeature&typename=cts:Observation&filter=<Filterxmlns='http://www.opengis.net/fes/2.0'xmlns:cts='http://www.citi-sense.eu/citisense'><And><PropertyIsEqualTo><ValueReference>cts:sensorID/@xlink:href</ValueReference><Literal>)⁵

⁵ SEDSv2 2 – PROD – Data Access-SampleRequests.pdf (SnowFlake)



Besides LEO sensors capture multiple sensor data, our web client is only interested on following sensors:

- 1) NO – nitrogen oxide (ppb, parts per billion)
- 2) NO₂ – nitrogen dioxide (ppb, parts per billion)
- 3) O₃ – ozone (ppb, parts per billion)

When AQI⁶ is available from SEDS, the client also shows

- 1) NO₂ Air Quality Index (Health relation)
- 2) O₃ Air Quality Index (Health relation)

Which provides following mapping:

AQI Colour	Index	Health relation	PM10 (µg/m ³)	PM2.5 (µg/m ³)	NO ₂ (µg/m ³)	SO ₂ (µg/m ³)	O ₃ (µg/m ³)	CO
Red	4	Unhealthy	>200	>100	>200	>350	>240	
Orange	3	Unhealthy for sensitive groups	100-200	50-100	150-200	250-350	180-240	
Yellow	2	Moderate	50-100	25-50	100-150	150-250	100-180	
Green	1	Good	<50	<25	<100	<150	<100	

10.3 User applications/apps/processing/visualisations

enControl™ web client has been updated to provide feedback to the smart home user about the air quality provided by LEO device.

When available, AQI is preferred information to be shown to smart home user instead of ppb (part per billion) values, as provide more “clean” information to them. AQI is available for NO₂ and O₃, but not for NO. This is shown in Figure 10-4, a screenshot of enControl™ web client.

⁶ <http://confluence.nilu.no/pages/viewpage.action?pageId=48955415>

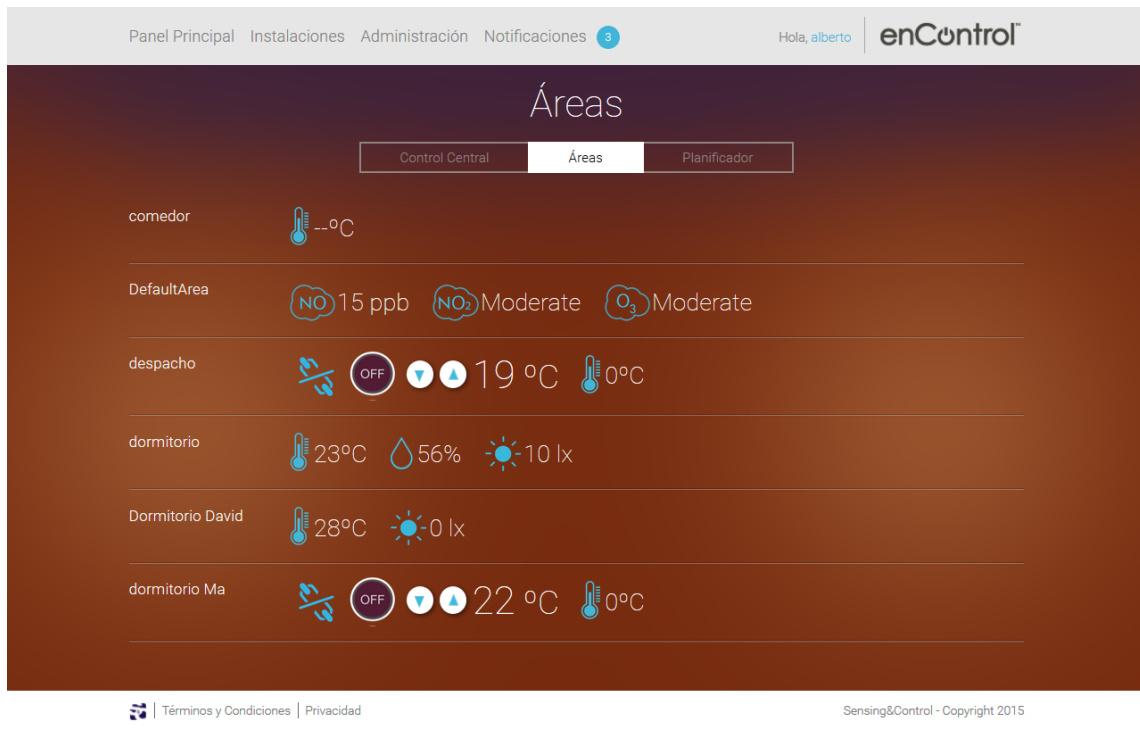


Figure 10-4 enControl™ user interface with LEO device measurement in area called “DefaultArea”.

Figure 10-5 shows last raw data (NO) or AQI/Health relation (NO₂ and O₃). The user can then click on each of the values, and is redirected to plot view of today’s values. The values are aggregated by 30 minutes (avg), then the user can:

- 1) Zoom in/out current plot values
- 2) Plot historical values by selecting initial day and final day.

Following screenshots gives a flavour of the information being available to end user through enControl™ client.

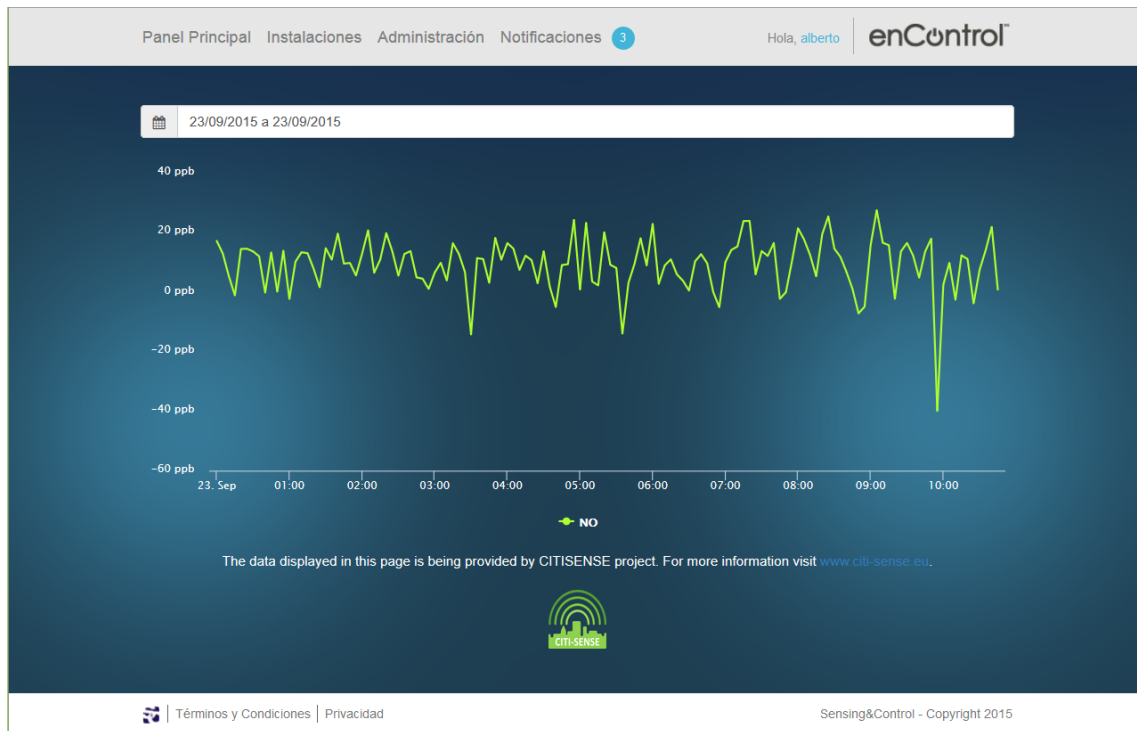


Figure 10-5: Nitrogen Oxide (NO) raw data aggregated (avg) on 5 minute interval

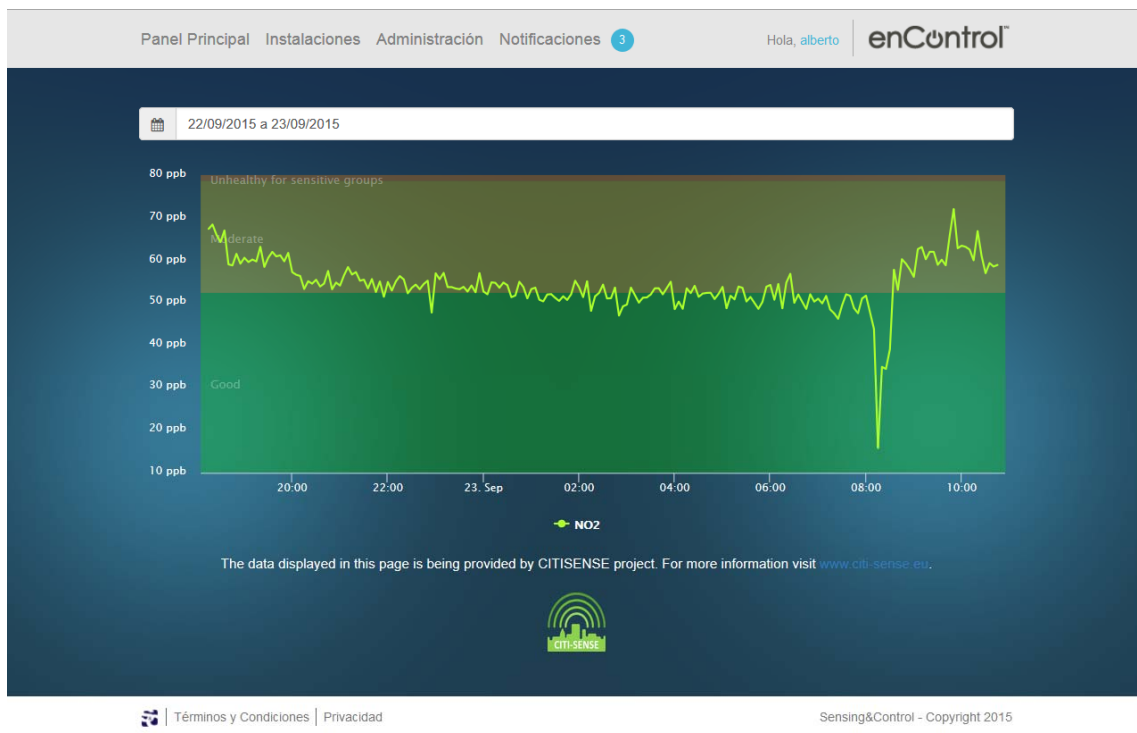


Figure 10-6: Nitrogen Dioxide (NO₂) raw data aggregated (avg) on 5 minute interval with AQI



Figure 10-7: Ozone (O₃) raw data aggregated (avg) on 5 minute interval with AQI



Figure 10-8: Ozone (O₃) raw data aggregated (avg) on 30 minute interval with AQI on background.



10.4 Suggested improvements and conclusions

The current version assumes a LEO device at home. The ID of the device is then used for the configuration for the web service call.

LEO is used for inside/outside measurements, the user is responsible to know where he has installed the device at home, and he can define a name for an area, and used as reference of the place installed. Currently the name has been defined as “DefaultArea”, but could be any name like “living room” or “garden”.

One of the main improvements would be to map a wide area measurement (for instance a city or city block) that is computed by CitiSense and available by SEDS platform. In this way, the user may not need to install outside home a LEO sensor, but get aggregated information around the zone his/her house is place.

If a SEDS provide a web service for this, the integration of this feature is straightforward within enControl™.

Other aspects to consider for the future includes:

- 1) Notifications
 - a. To add notifications to enControl users based on AQI/Health Relation levels change.
- 2) Rule engine
 - a. To perform control commands triggered by AQI/Health Relation Levels over actuators at home to improve indoor air quality.



11 Annex F: SENSE-IT-NOW Multi-platform Smartphone App

11.1 Introduction to usage context

11.1.1 Introduction

SENSE-IT-NOW is a multi-platform smart phone app and currently tested on Android platforms. The SENSE-IT-NOW smartphone application is the user's front-end for different applications and services. It reads and displays data from different sensors by doing http requests to the CITI-SENSE platform. It gives the user the possibility to add subjective observations about their current environment and act as a human sensor.

At this stage of the project, access point for the data is both a web service provided by U-Hopper for requesting answers on questionnaires, and the SensApp product from SINTEF for easy storage and retrieval of simple sensor data.

The application integrates the questionnaire product CivicFlow from U-Hopper.

11.1.2 Reference to CITI-SENSE Pilot workpackages

The SENSE-IT-NOW application is offered as a tool in the CITI-SENSE toolbox and can be used across all locations and workpackages. It was used as part of the main tools for the Vitoria Case study in WP3a.

SENSE-IT-NOW is further described and referred to in the following deliverables/work within the CITI-SENSE work packages: D3.1, D3.2, D2.1 D2.2, D6.2 and it has been extensively documented in D6.4 and tested in D6.5.

11.1.3 Reference to CITI-SENSE Locations

Some specific adjustments have been created to support the Vitoria case study in WP3.

11.2 Architecture and interfaces used

11.2.1 Data flow

The following shows the current architectural picture of the data flow related to the SENSE-IT-NOW application.

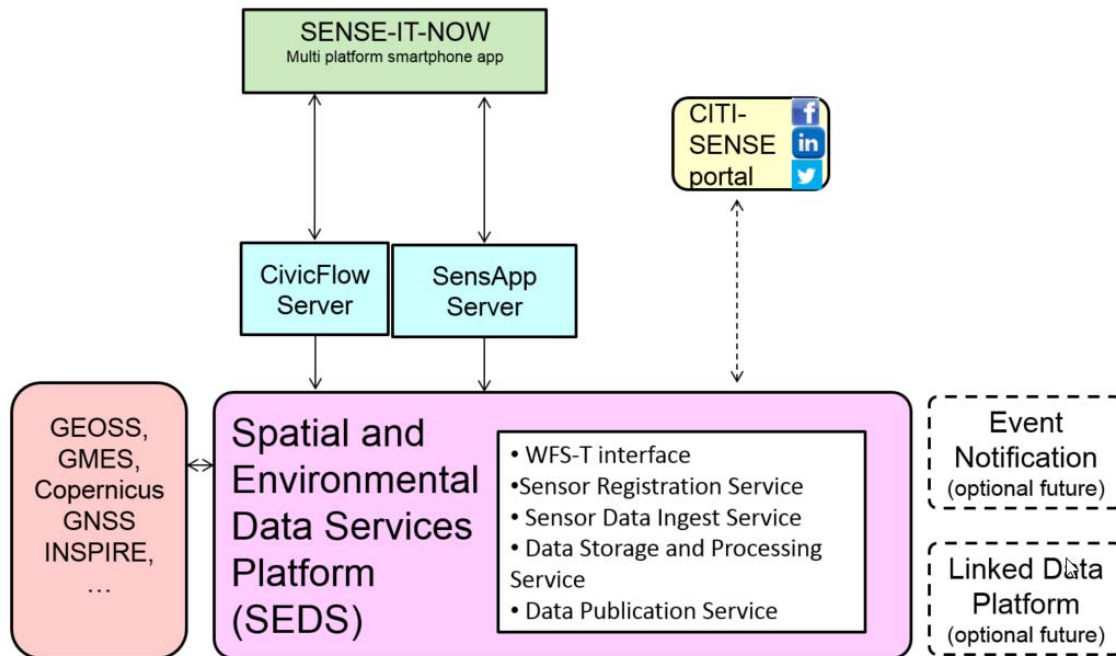


Figure 11-1 Data Flows for the SENSE-IT-NOW architecture

In general, the application can show measurements of any sensor data stored in the CITI-SENSE platform. It also uses the platform for storage of observations of the surroundings (photos taken by the smartphone with a perception tag of pleasant or unpleasant). The CivicFlow application is an integrated part of the SENSE-IT-NOW application and provides an interface for answering questionnaires. Some of these answers are required for the application to calculate comfort indexes and present that to the user.

11.2.2 Architecture

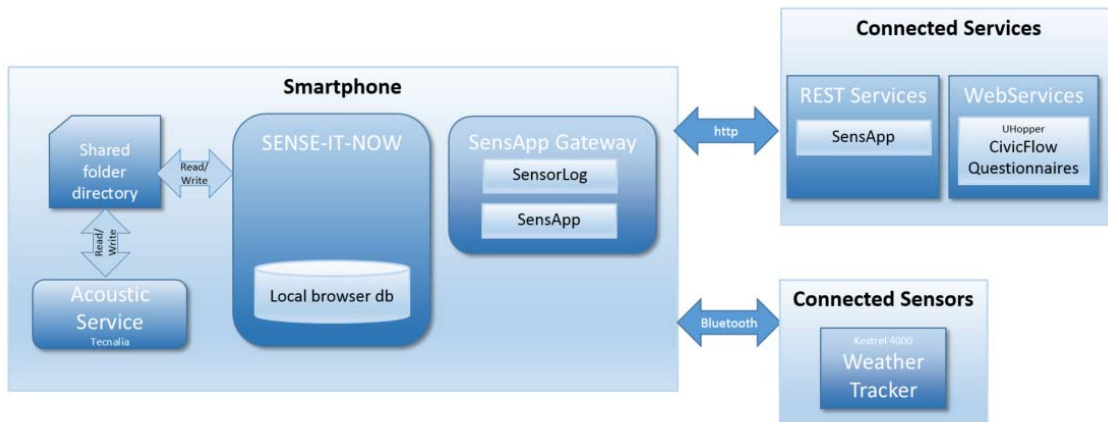


Figure 11-2 SENSE-IT-NOW Interface architecture

The SENSE-IT-NOW application works as a user interface between different applications, products and services developed by CITI-SENSE to support the case studies in the fieldwork.

The application can perform different actions:

- Displays measured or calculated sensor data stored on the CITI-SENSE platform
- Start and stop a measured period for the user to follow measured values for a sensor during a time period
- Calculate different comfort indexes for the acoustic sensor software from Tecnalia and the Kestrel weather meter sensor in combination with answers from questionnaires. For the acoustic sensors the application also support for sound alert detection.

Provides a possibility to take a photo and add a perception of unpleasant or pleasant at a specific GPS position and offers a map that displays the previous observations.

11.2.3 Framework Architecture

The SENSE-IT-NOW application was built with PhoneGap/Cordova. This is an open source framework that uses standardized web APIs (JavaScript, HTML5, CSS3) to develop applications for different mobile platforms.

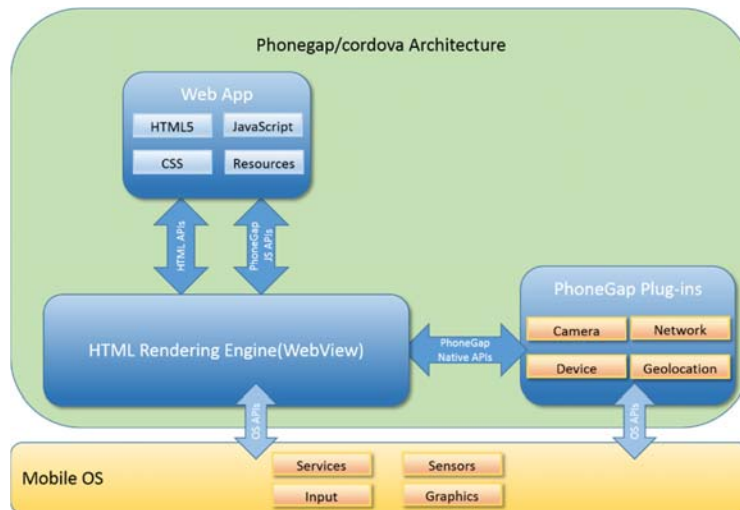


Figure 11-3 PhoneGap/Cordova SmartPhone App architecture

PhoneGap/Cordova offers different plugins for accessing the native platform it runs on. Currently the application has implemented support for four plugins:

- Camera
 - Use of the phones camera
- Network
 - Access Wi-Fi and cellular network
- Device
 - Access information about the device`s hardware and software
- Geolocation
 - Access location data based on GPS or network signals

11.3 Data (volume, velocity) being stored and retrieved

Pictures uploaded and stored in SensApp are scaled down to 800x800 pixels. The application uses a local web db to pre store the photos and uploads them into SensApp when there is mobile or Wi-Fi connection.

SENSE-IT-NOW displays sensor data from the CITI-SENSE platform. When a session for displaying measured real time values are started, the application does a request every 10 second to get the latest uploaded data. Currently four sensors are supported

- Acoustic
- Temperature
- Relative humidity
- Wind speed

To display continues graphs of measured values on each sensor, the application requests data within a short time period (depending on the last time the values could be retrieved) every 30 second. The amount of data transferred on each request are limited.

11.4 User applications/apps/processing/visualisations

11.4.1 SENSE-IT-NOW

The SENSE-IT-NOW application consists of different elements to assist the Empowerment initiatives. The supported sensors can be selected and added in a settings page and will reflect what data that will be requested and displayed in the graphs available. All settings are stored locally on the smartphone using a web db.

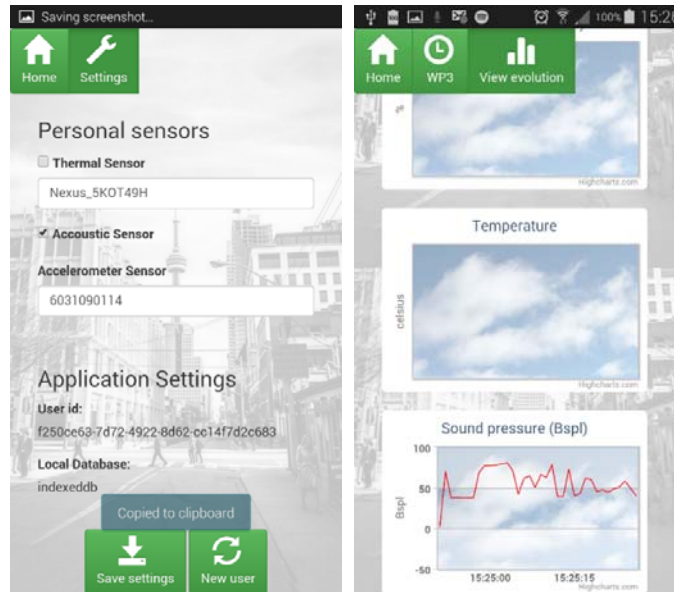
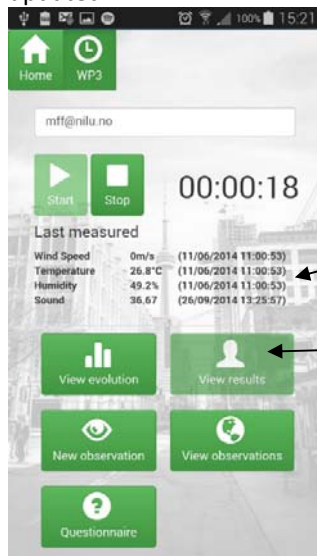


Figure 11-4 SENSE-IT-NOW Settings page and visualisation

The user needs to start a measured period for the graphs and measurements to be displayed and updated.



Displays last measuring value and time

Here you will see the last measured values from the Kestrel and the acoustic sensor.

Figure 11-5 SENSE-IT-NOW Selection page

Adding observations gives the possibility for the user to contribute with his/her own perception of the environment and can later be viewed on a map displaying only this specific person's photos and perceptions.

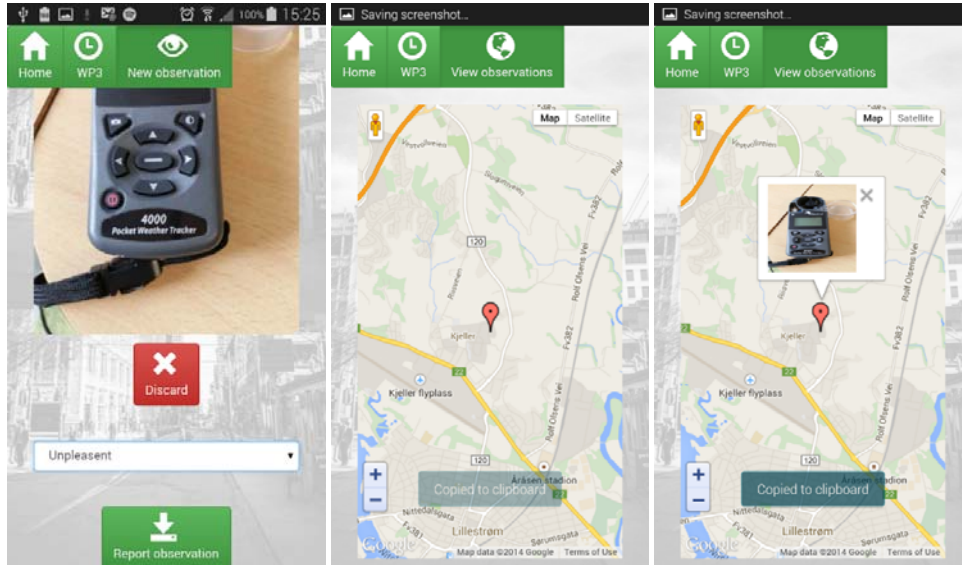


Figure 11-6 SENSE-IT-NOW Observation pages

The green marker displays pleasant photos taken and the red one is for unpleasant photos.

The SENSE-IT-NOW application also works directly with the native android Acoustic sensor application. This app from Ateknea needs to be installed on the smartphone and be enabled in the settings of SENSE-IT-NOW. When certain sound varieties happens in the surrounding this will be detected and the user can select it's source and perception. This is will later be used to calculate a sound comfort index after the user has chosen to end the measured period.

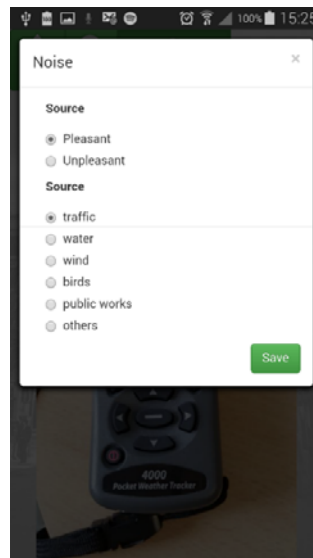


Figure 11-7 SENSE-IT-NOW Noise page



SENCE-IT-NOW integrates the CivicFlow app from U-Hopper to add more needed information from the user.

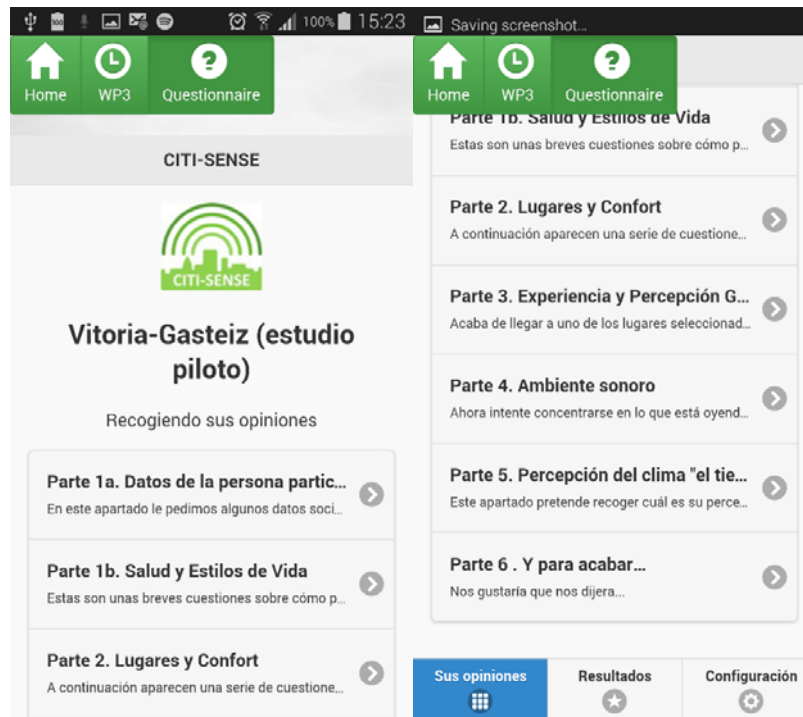


Figure 11-8 SENSE-IT-NOW Questionnaire pages

After stopping a measurement period, the application offers different calculated result depending on the sensors chosen.

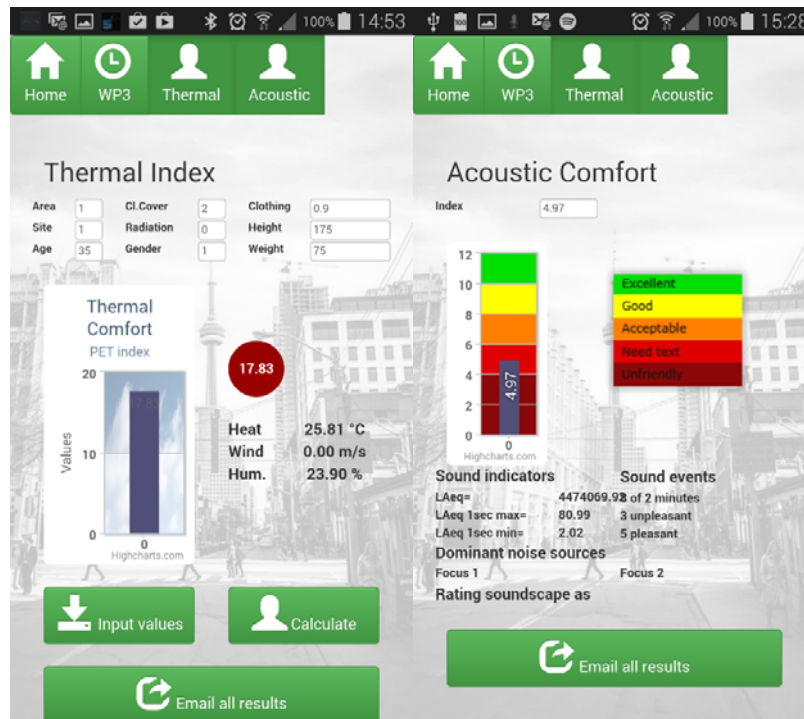


Figure 11-9 SENSE-IT-NOW Thermal Index and Acoustic Comfort

11.4.2 SENSE-IT-NOW – WP2 – Urban Quality based on LEO sensor

SENSE-IT-NOW app has been upgraded to support the urban quality use case developed in WP2, in particular, the application is supporting the LEO (Little Environment Observatory) device provided by project partner Atekeia.

The LEO device captures NO, NO₂ and O₃. As the device is target to be carried by an end user, it is used in combination with an Android application called ExpoApp, which connects via Bluetooth channel with LEO, and combines smartphone's position and physical activity (calculated using accelerometer sensors) information with NO, NO₂ and O₃ data.

The data is being upload continuously to the CITISENSE main servers, thus being available by SENSE-IT-NOW application to provide feedback to end user.

ExpoApp requires the user to configure a 'Subject ID'. This parameter is the one that has to be configured in SENSE-IT-NOW in order to display data being collected by ExpoApp for a particular user. Following picture shows the parameter for both applications as shown in following picture.

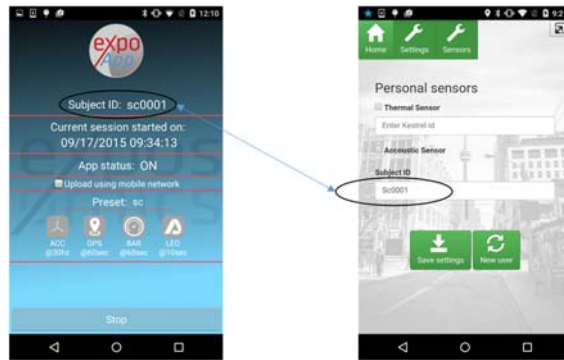


Figure 11-10 SENSE-IT-NOW with ExpoApp

In order to setup the Subject ID in SENSIT-NOW application, just tab ‘Settings’ button and then ‘Sensors’ button to arrive at the interface shown in previous picture.

Once the subject ID has been configured, the user can have feedback about the air quality levels and the physical activity of the person as a function of time and as a function of position.

The two possibilities are available after the user tabs on WP2 button.

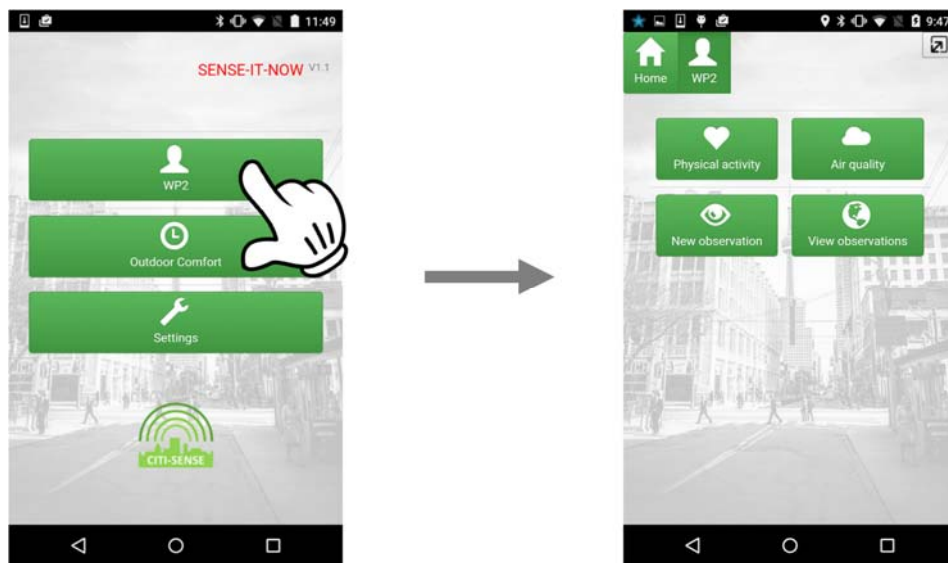


Figure 11-11 SENSE-IT-NOW Data selection pages

If the user tabs ‘Air Quality’ button, he will be requested to select a range of dates and the sensor measurement: NO, NO₂ or O₃.

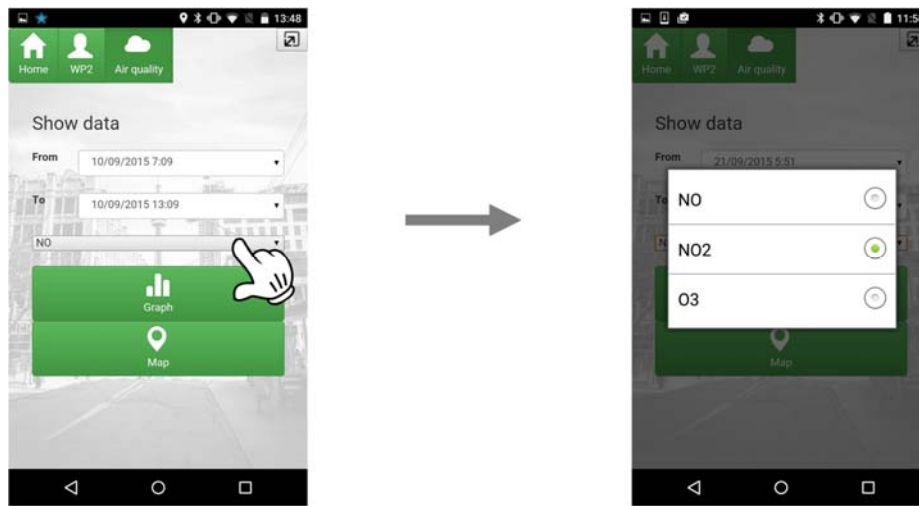


Figure 11-12 SENSE-IT-NOW Air Quality data selection pages

Once the user has selected dates and sensor measurement, the application will be able to display related data. Following figure shows an example of data for NO₂ sensor as function of time.

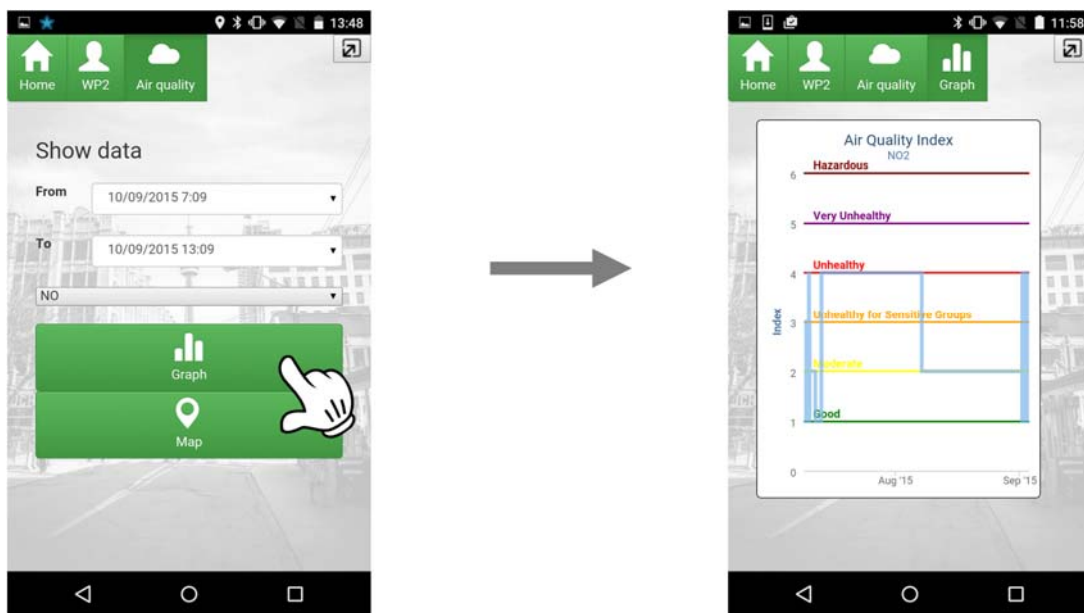


Figure 11-13 SENSE-IT-NOW Selecting graph of Air Quality Index

The following figure shows an example of data for NO₂ sensor as function of position. Notice that the colours on the map are related with air quality index colours shown in previous picture.

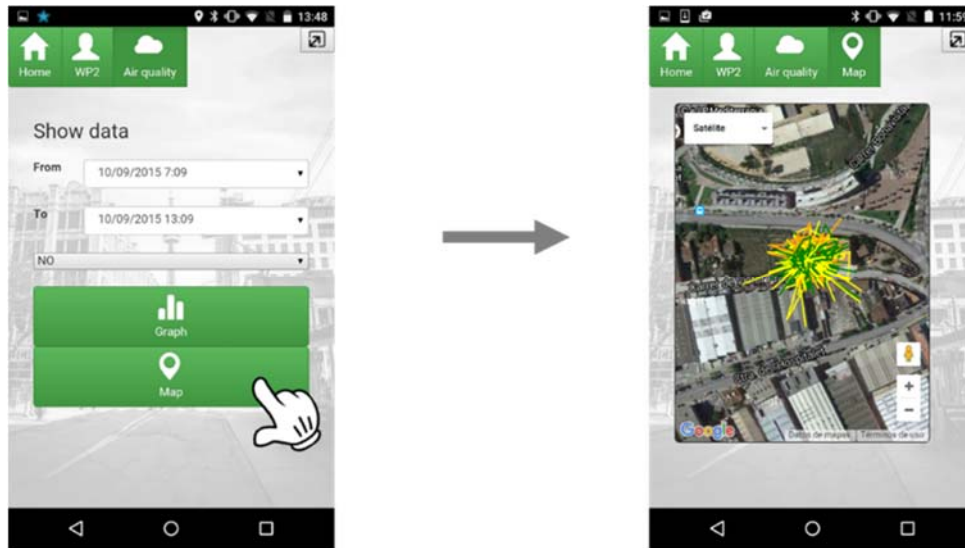


Figure 11-14 SENSE-IT-NOW map widget with position based data

11.5 Suggested improvements and conclusions

Activities in WP6 – task 6.4 Test products and services (using preliminary data) for the EIs addressed in WPs 2-3.

The testing has been of mobile apps and their ability to provide information from sensors to the user. There has also been testing of widgets on web portals. Finally, there has been tests of the user friendliness of the products and services, technical tests, and tests to check that the products and services are compliant with requirements from WP2 and WP3.

Further to implement and document tools in CS toolbox for the EIs addressed in WPs 2-3 and test with real data has been done in the final pilot cases.

11.6 Service for calculating physical movement

11.6.1 Introduction to usage context

The CitiSense.ExpoApp is a windows service running on a server at NILU that calculates physical movement indexes on a specific position based on gps and acclerometry data produced by the ExpoApp android application from Sensing & Control and Ateknea.

11.6.2 Architecture and interfaces used

This windows service reads data created by the ExpoApp from Sensing & Control and Ateknea from the SensApp instance on the Amazon cloud and calculates an indicative value representing the index of physical movement on a specific position. The results are registered and stored back in SensApp.

11.6.3 Data (volume, velocity) being stored and retrieved

The service is invoked every 3 minute and finds data on all sensors registered in SensApp. The service works on data given for a 10-second period and aggregates and stores the MET values as 1 minute data.



11.6.4 User applications/apps/processing/visualizations

There are no graphical interface for this windows service.

11.6.5 Suggested improvements and plans for next phase(s)

This application needs to be testes for larger amount of data and installed and configured to work in a production environment. This service currently runs on a test machine.

11.7 PET calculation

11.7.1 Introduction to usage context

The java script library provides a function with input parameters and produces a PET value to indicate a thermal comfort index.

11.7.2 Architecture and interfaces used

The PET calculation is a conversion of a Fortran 90 code into a Java Script library. The Fortran code is originally developed by Tecnalía.

The input parameters to the function are:

- Wind speed average of a time period
- Temperature average of a time period
- Relative humidity average of a time period
- Tmrt (Mean Radiant Temperature) value
- Person`s age
- Person`s weight
- Person`s height
- Person`s cloud cover

11.7.3 Data (volume, velocity) being stored and retrieved

No data to be retrieved or stored

11.7.4 User applications/apps/processing/visualizations

No user interfaces

11.7.5 Suggested improvements and conclusions

There is no further work planned to be done directly on the library.

The JavaScript is made available as part of the CITI-SENSE toolbox and is accessible using the following URL: <http://toolbox.citi-sense.eu/tecnalia.PETCalculation.js>

12 Annex G: SENSE-IT-NOW: Acoustic Service

12.1 Introduction to usage context

12.1.1 Introduction

The Acoustic Service is a service that runs on Android devices. It has been developed to measure the citizens' acoustic comfort at public outdoor spaces in Vitoria (Spain).

The Acoustic Service, through the smartphone's microphone, analyze all the acoustic signals in order to detect the acoustic events. Such events are noticed to the user in order to ask for her perception.

The smartphone's microphone is provided with a windscreen. The Edutige EIM-003 microphone was chosen as part of the acoustic sensor. The accuracy of the noise levels measured by the microphone are analyzed in terms of its frequency response and the differences in global noise levels (dBA), in comparison with a class 1 microphone. The purpose of the sensor is not reaching the accuracy of a sonometer class 1, but to give a global noise level with a difference of ± 2 dB. The analysis considers the whole measurement chain: from microphone to Acoustic Service, since the comparison is made considering the final results obtained.

The final aim of the acoustic service is to calculate the ESEI (Environmental Sound Experience Indicator). This indicator was developed by Tecnalía in previous research projects to measure the Acoustic Comfort and it is based on the sound events (defined by an own developed algorithm) and the user's perception.

All the acoustic measures, the events and ESEI calculations are sent to the SensApp Server.

12.1.2 Reference to CITI-SENSE Pilot workpackages

The Acoustic Service is being further described and referred to in the following deliverable/work within the CITI-SENSE project: deliverable D3.2/workpackage 3.

12.1.3 Reference to CITI-SENSE Locations

The Acoustic Service is used at Vitoria (3 public outdoor spaces).

12.2 Architecture and interfaces used

The Acoustic Service does not store computed data, so during the measurement period the data flow is as follows: (see Figure 12-1).

- The Acoustic Service uses SensApp as repository for the L_{Aeq} (equivalent noise level with the A weighting over a given period of time) calculation (every 60 seconds).
- The Acoustic Service sends the detected events to the CITI-SENSE App. As response for each event, the SENSE-IT-NOW App sends the user's perception to the Acoustic Service.
- The final calculated values (ESEI-Acoustic Comfort Index and L_{Aeq} maximum, minimum and average) are sent to the SENSE-IT-NOW App to be displayed to the user performing the measurement period.

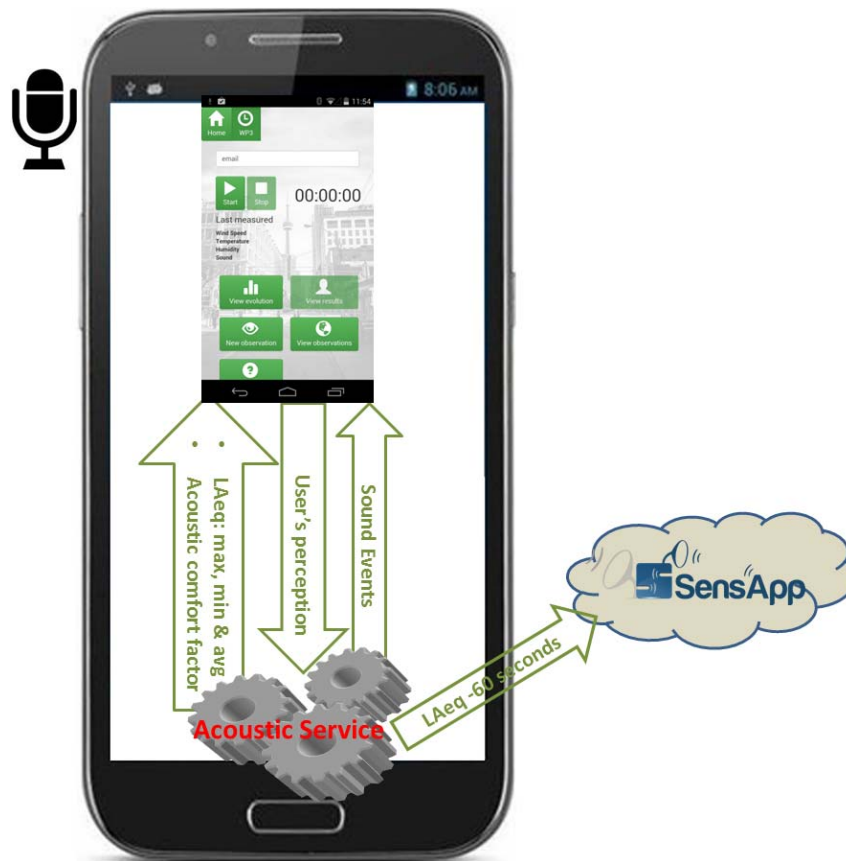


Figure 12-1 Acoustic Service – Data flow

12.3 Data (volume, velocity) being stored and retrieved

The volume of data that is being collected during one sample of measurements is the following:

- The signal received by the microphone is processed to obtain the L_{Aeq} every second.
- The Acoustic Service sends the L_{Aeq} measures each 60 seconds to SensApp, which means 60 L_{Aeq} values every minute.
- For each relevant event detected, the Acoustic Service communicates with the SENSE-IT-NOW App to ask the user about her perception, in order to incorporate it in the ESEI calculation.
- The value of the ESEI along with the maximum, the minimum and the average values of L_{Aeq} are sent to the CITI-SENSE App at the end of the measurement period.

12.4 User applications/apps/processing/visualisations

The acoustic service is invocated from the Citisense App, so no user interface is provided from the service itself, although the Citisense App provides the interface to the acoustic service.

The next figure shows how the Citisense App shows to the user the measurements as they are gathered by different services (as their corresponding sensors), being one of them the Acoustic Service:

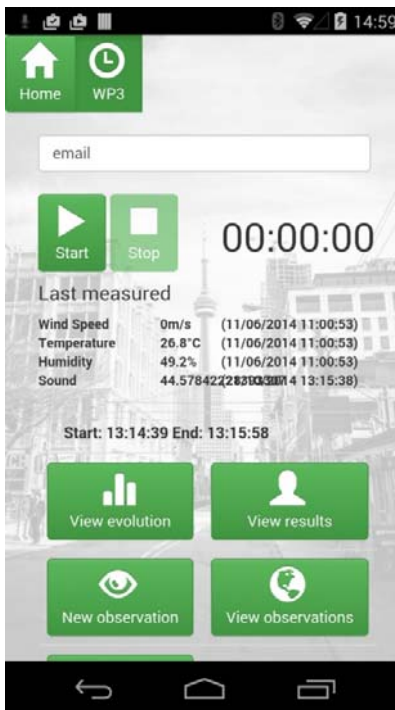


Figure 12-2 Citisense App User Interface

The figure shows how the user is alerted when the acoustic service detects an acoustic event. The user is asked about the sound source and her perception of the event. The answer to this questionnaire is sent back to the acoustic service for the ESI calculation.

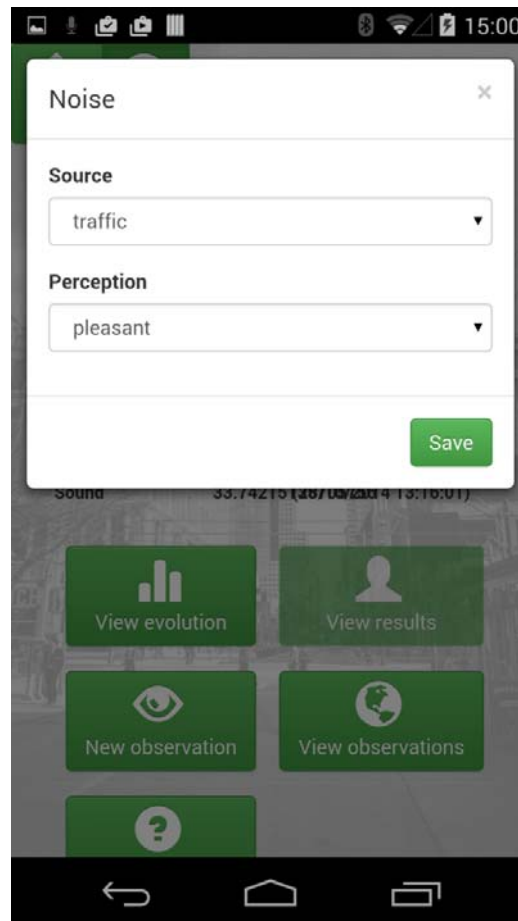


Figure 12-3 Event Notification Questionnaire

12.5 Suggested improvements and conclusions

During the pilot study developed to define the protocol for phase 2, some aspects to be improved were identified linked to the acoustic service. These aspects are related with the functionality of the service to obtain, the required technical data to assess acoustic parameters during the pilot implementation phase and with the empowerment initiative in Vitoria-Gasteiz:

1. Poor stability in the process of the received signal by the microphone: this aspect implies a series of failures:
 - The sound events are not correctly⁷ detected.
 - The descriptor of the sound level (LAeq) is not correctly⁸ evaluated by the acoustic service.
2. The calculation of the ESEI index (acoustic comfort) is not totally accurate: there are some problems in the data transfer between the acoustic service, the APP and the U-Hopper

⁷ The reference to “correct” is linked to the comparison of the results obtained by the acoustic service with data provided by a sound meter (calibrated with type 1 microphone).

⁸ The reference to “correct” is linked to the comparison of the results obtained by the acoustic service with data provided by a sound meter (calibrated with type 1 microphone).



questionnaire. So, to solve this aspect the process for data transfer will be reanalyzed in the following terms:

- For the acoustic Indicator (ESEI) assessment the answers and their values of the questions 2 (Q4.02) and 4 (Q4.04) of part 4 of questionnaire will be differently treated:
 - a. The variables range in the questionnaire is from 1 to 5.
 - b. This variables range must be adapted to the following process:
 - i. $((Q4.02) + 1) + (Q4.04) \rightarrow$ range from 3 to 11
 - 1. This new range must be encoded as follows:
 - a. Values $< 6 \rightarrow -1$
 - b. Between 6-8 $\rightarrow 0$
 - c. Values $> 8 \rightarrow +1$
 - ii. The new code (-1, 0, 1) must be sent to the " Acoustic Service" where the Acoustic indicator (ESEI) is calculated.

From a general point of view, the first testing done on the data flow and the communication between the acoustic service and the SENSE-IT-NOW APP provides correct results. This aspect has been confirmed by the comparison of the data provided by the acoustic service and the information displayed by the SENSE-IT-NOW APP.

13 Annex H: CityAir

13.1 Introduction to usage context

13.1.1 Introduction

The main purpose of the CityAir smartphone app is to collect people's perception on air quality and allow the user to register how he/she perceives the air quality where they are at any time. It also gives the people a possibility to view perceptions and comments made by others.

The app's intention is to be easy and simple to use. The app is not location or country specific, and can therefore be used at a global scale.

The users can give their perception about the air quality where they currently are located, by adding an icon of a man. The colour of the icon decides the perception where others than green indicates that the current air quality is not just good. The users can then choose to suggest the source of the pollution.

The CityAir app is an app that has been developed based on part of the work done in the Citi-Sense-MOB project. CityAir is developed using the same code base and are simplified and modified to be used in a more generic way to support all EIs in the CITI-SENSE project.

In deliverable D6.4, the CityAir will be extensively documented and D6.5 will describe the test phases involved in the process.

13.1.2 Reference to CITI-SENSE Pilot workpackages

WP2 came up with the requirements for a smartphone app that could collect people's opinion and perception about the air quality.

13.1.3 Reference to CITI-SENSE Locations

CityAir is designed for usage across the world. It is multilingual and is currently supporting the following languages

- English
- Serbian
- Catalanian
- Spanish
- Czech
- Slovenian
- Norwegian

13.2 Architecture and interfaces used

13.2.1 Data flow

The following figure shows the current architectural picture of the data flow related to the CityAir application and CITI-SENSE's overall architecture.

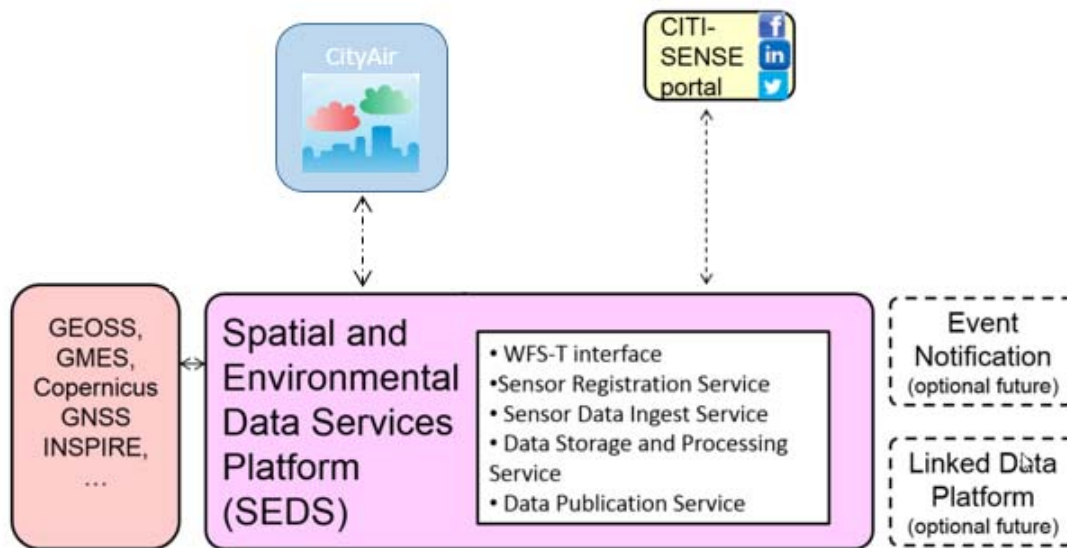


Figure 13-1 Data Flows for the CityAir architecture

13.2.2 Architecture

CityAir stores all information in a local database on the smart phone. When network is available, it will store the perceptions and the comments from the user on the SEDS server hosted by Snowflake using HTTPS (a secure HTTP protocol that requires username and password).

CityAir can also download and visualize perceptions and comments made by other users.

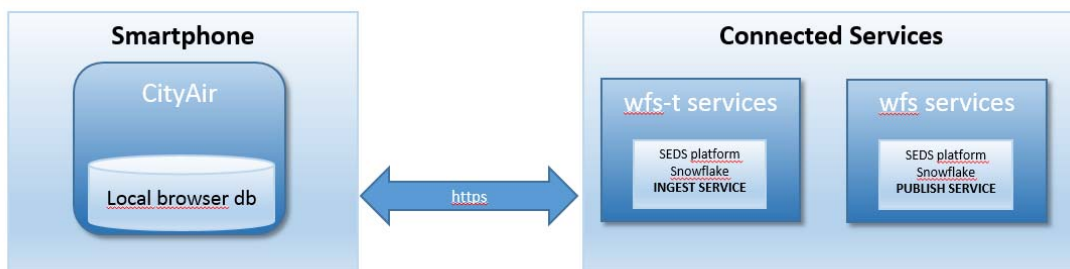


Figure 13-2 CityAir Architecture

13.2.3 Framework Architecture

The CityAir application is built with PhoneGap/Cordova, similarly to the WE-SENSE-IT application. This is an open source framework using standardized web APIs such as JavaScript, HTML5 and CSS3 to develop applications for different mobile platforms.



PhoneGap/Cordova offers different plugins for accessing the native platform it runs on. Currently the application has implemented support for three plugins

- Network
 - Access Wi-Fi and cellular network
- Device
 - Access information about the device`s hardware and software
- Geolocation
 - Access location data based on GPS or network signals

See further information on this architecture described for the WE-SENSE-IT app previously described.

13.3 Data (volume, velocity) being stored and retrieved

The SEDS platform offers different components for storing and retrieving data

- WFS-T – data ingest component, https transaction service for inserting data
- WFS – data publication component, http service for accessing data, xml
- Web services – simplified web services for accessing data, json and csv formats

For uploading and storing the users` perceptions and comments, the SEDS platform`s WFS-T ingestion component requires a predefined xml structure.

One perception or comment will be built using the following xml structure.

```
<wfs:Transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:cts="http://www.citi-sense.eu/citisense"
xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gco="http://www.isotc211.org/2005/gco"
xmlns:gss="http://www.isotc211.org/2005/gss" xmlns:gts="http://www.isotc211.org/2005/gts"
xmlns:gsr="http://www.isotc211.org/2005/gsr" xmlns:wfs="http://www.opengis.net/wfs/2.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:fes="http://www.opengis.net/fes/2.0"
version="2.0.0" service="WFS">
<wfs:Insert>
<cts:Observation gml:id="LOCAL_ID_0">
<!-- M constant-->
<cts:sensorID xlink:href="#AQAP_e90886b8-80cf-4ee8-fb4b-8bee3f0e6d6d"/>
<cts:contains>
<cts:Measurement>
<cts:measurementID>#PopulatedAutomatically</cts:measurementID>
<cts:value/>
<cts:uom>PCN</cts:uom>
<cts:observedProperty>1970;male;2;white;Outside</cts:observedProperty>
<cts:measuretime>2015-09-11T11:14:41</cts:measuretime>
<cts:latitude>59.95787</cts:latitude>
<cts:longitude>11.04713</cts:longitude>
</cts:Measurement>
</cts:contains>
<cts:canHave>
<cts:Questionnaire>
<cts:includes>
<cts:Question>
<cts:has>
```



```

<cts:Response>
  <cts:source>xxx</cts:source>
  <cts:timestamp>2015-09-11T11:14:41</cts:timestamp>
</cts:Response>
</cts:has>
<cts:needs>
  <cts:Answer>
    <cts:value>1970</cts:value>
  </cts:Answer>
</cts:needs>
<cts:parent_id>1</cts:parent_id>
<cts:values>xxx</cts:values>
<cts:required>0</cts:required>
<cts:label>Year Born</cts:label>
</cts:Question>
</cts:includes>
<cts:includes>
  <cts:Question>
    <cts:has>
      <cts:Response>
        <cts:source>xxx</cts:source>
        <cts:timestamp>2015-09-11T11:14:41</cts:timestamp>
      </cts:Response>
    </cts:has>
    <cts:needs>
      <cts:Answer>
        <cts:value>male</cts:value>
      </cts:Answer>
    </cts:needs>
    <cts:parent_id>1</cts:parent_id>
    <cts:values>xxx</cts:values>
    <cts:required>0</cts:required>
    <cts:label>Gender</cts:label>
  </cts:Question>
</cts:includes>
<cts:includes>
  <cts:Question>
    <cts:has>
      <cts:Response>
        <cts:source>xxx</cts:source>
        <cts:timestamp>2015-09-11T11:14:41</cts:timestamp>
      </cts:Response>
    </cts:has>
    <cts:needs>
      <cts:Answer>
        <cts:value>2</cts:value>
      </cts:Answer>
    </cts:needs>
    <cts:parent_id>1</cts:parent_id>
    <cts:values>xxx</cts:values>
    <cts:required>0</cts:required>
    <cts:label>Education</cts:label>
  </cts:Question>
</cts:includes>

```



```

<cts:includes>
  <cts:Question>
    <cts:has>
      <cts:Response>
        <cts:source>xxx</cts:source>
        <cts:timestamp>2015-09-11T11:14:41</cts:timestamp>
      </cts:Response>
    </cts:has>
    <cts:needs>
      <cts:Answer>
        <cts:value>Outside</cts:value>
      </cts:Answer>
    </cts:needs>
    <cts:parent_id>1</cts:parent_id>
    <cts:values>xxx</cts:values>
    <cts:required>0</cts:required>
    <cts:label>Comment</cts:label>
  </cts:Question>
</cts:includes>
<cts:includes>
  <cts:Question>
    <cts:has>
      <cts:Response>
        <cts:source>xxx</cts:source>
        <cts:timestamp>2015-09-11T11:14:41</cts:timestamp>
      </cts:Response>
    </cts:has>
    <cts:needs>
      <cts:Answer>
        <cts:value>white</cts:value>
      </cts:Answer>
    </cts:needs>
    <cts:parent_id>1</cts:parent_id>
    <cts:values>xxx</cts:values>
    <cts:required>0</cts:required>
    <cts:label>Color</cts:label>
  </cts:Question>
</cts:includes>
<cts:campaign_id>0</cts:campaign_id>
<!-- M Int -->
<cts:created>2015-09-11T11:14:41</cts:created>
<!-- M DateTime -->
</cts:Questionnaire>
</cts:canHave>
<cts:idquestionnaire>#PopulatedAutomatically</cts:idquestionnaire>
<!-- M -->
<cts:starttime>2015-09-11T11:14:41</cts:starttime>
<!-- M Date Time-->
<cts:finishtime>2015-09-11T11:14:41</cts:finishtime>
<!-- M Date Time -->
</cts:Observation>
</wfs:Insert>
</wfs:Transaction>

```



The logical structure of the data to transfer to the server is an observation with a questionnaire object. This questionnaire may contain several questions with a corresponding response and answer pair. This follows the CITI-SENSE Data Model:

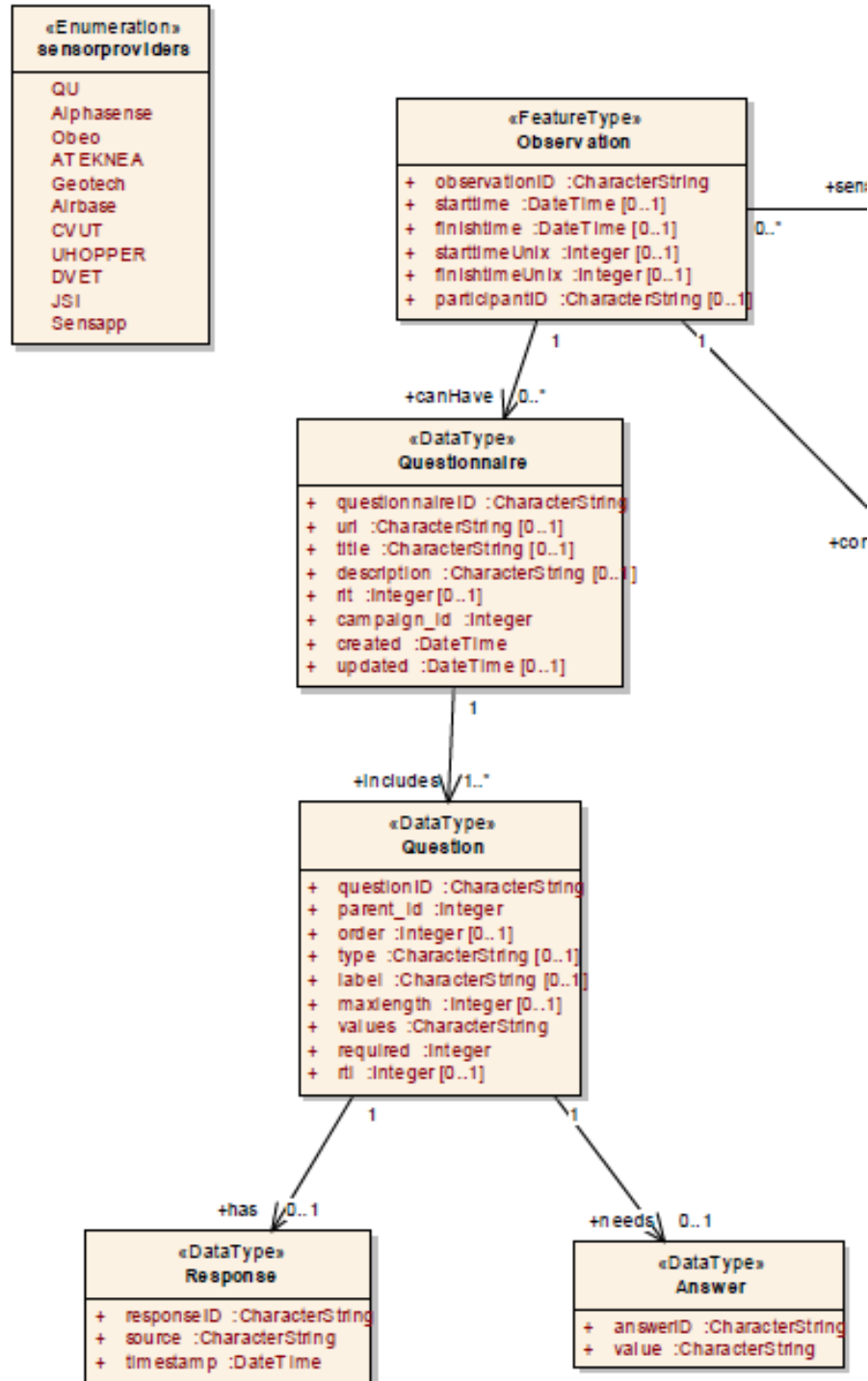


Figure 13-3 Part of CITI-SENSE Data model used by CityAir

Depending on the length of the comments, the size of the transferred and uploaded xml to the server will vary. Tests indicate that each transfer will be under 10KB.



The upload stream will use cellular network or Wi-Fi for sending the data. The user can choose only to upload data if she or he is connected to Wi-Fi.

To retrieve others' perceptions and comments the SEDS platform offers web services that provides the data on JSON format. JSON is text based data and formatted in key-value pairs and is in this case a better format to consume for a mobile device due to its smaller size.

An example of an HTTP GET query from CityAir is to download the perceptions from another CityAir app within a chosen time period

https://prod.citisense.snowflakesoftware.com/json/sensor/observationfinishtime/between?sensorid=AQAP_a5c8313f-f4ea-4202-e3de-a7ec1c2d93bd&from=2015-10-10T13:46:50&to=2015-11-10T13:46:50

The response from the server is a JSON object that can be read and visualized by the CityAir app.

```
[
  { "aqi_colour": null, "aqi_value": null, "caqi_colour": null, "caqi_value": null, "finish_time": "2015-10-27T20:48:16.000", "global_caqi_colour": null, "global_caqi_value": 0, "latitude": 46.17678, "longitude": 14.32264, "measure_time": "2015-10-27T20:48:16.000", "measurementid": 253850000.0, "observationid": 253848000.0, "observedproperty": "1988;male;3;green;", "participantid": null, "sensorid_feature_id": "AQAP_a5c8313f-f4ea-4202-e3de-a7ec1c2d93bd", "start_time": "2015-10-27T20:48:16.000", "uom": "PCN", "value": null }
]
```

The size of one typical response with one perception or one comment is about 500 bytes.

13.4 User applications/apps/processing/visualisations

13.4.1 CityAir – smart phone app

CityAir can collect a user's perception of the air quality at its current location using the smartphone's GPS. The user can choose to place a green, yellow, orange or red person to mark the area according to the air quality experienced. The user can also choose what pollution sources might be the cause of a polluted area and to leave a comment for others to view.

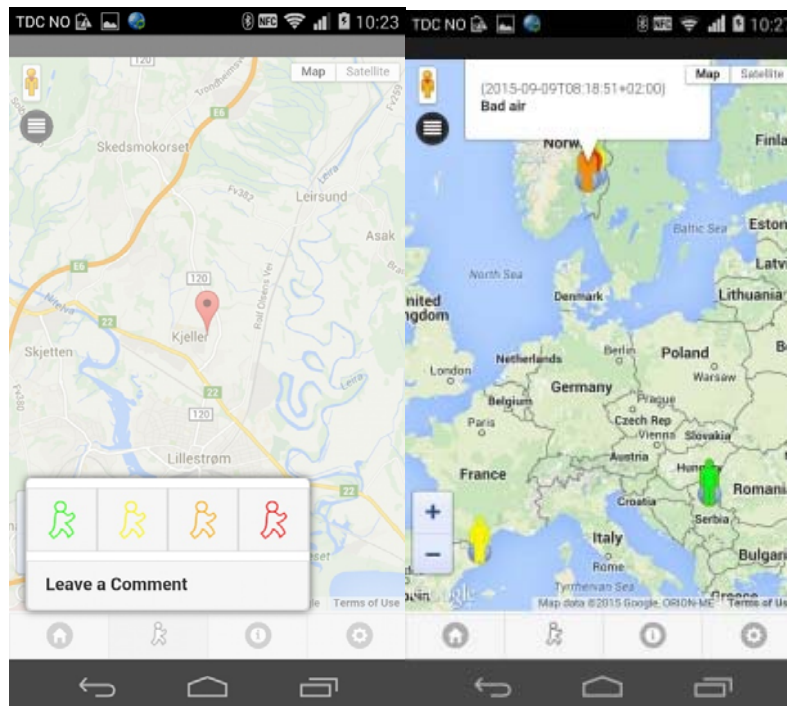


Figure 13-4 CityAir App Interface

13.4.2 Visualization services

Web pages displaying collected comments and perceptions have been set up to support each EI's local web portal, with clickable icons for information about date and time for the contributions, the suggested pollution sources and comments.

At globale scale

<http://toolbox.citi-sense.eu/UserPerceptions.html>

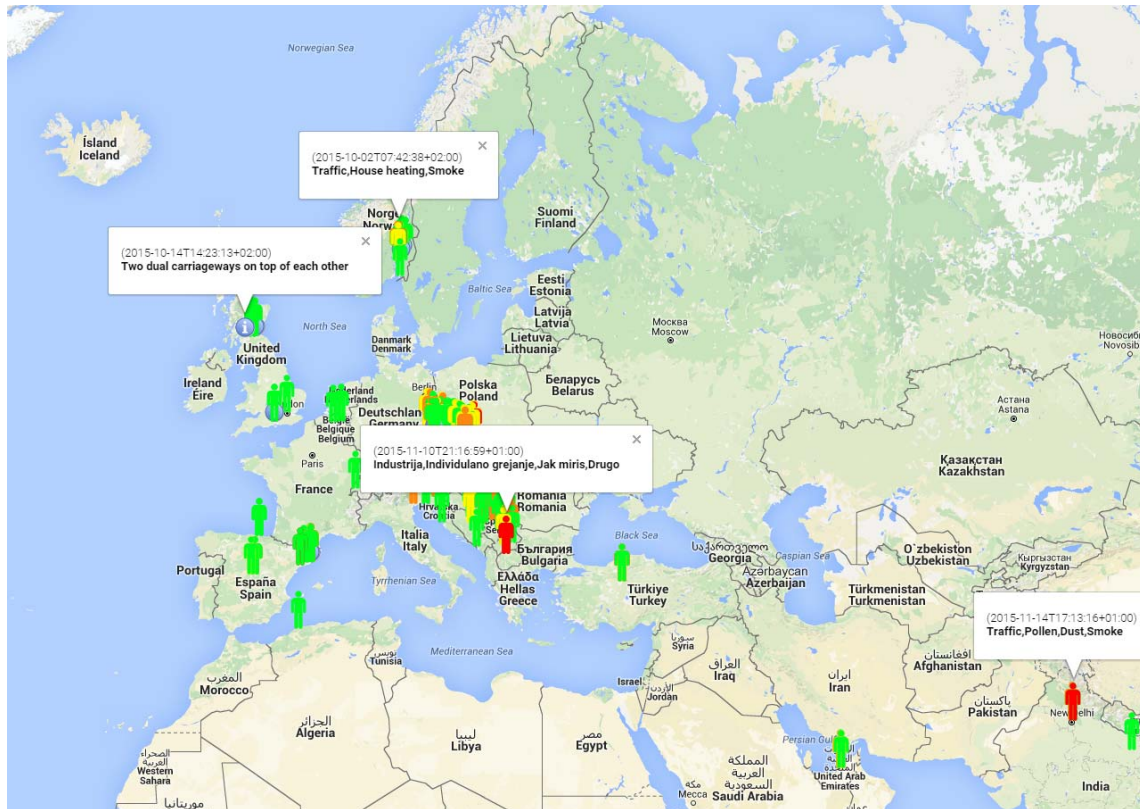


Figure 13-5 CityAir Map Visualisation

At locale scale

<http://toolbox.citi-sense.eu/UserPerceptionOslo.html>

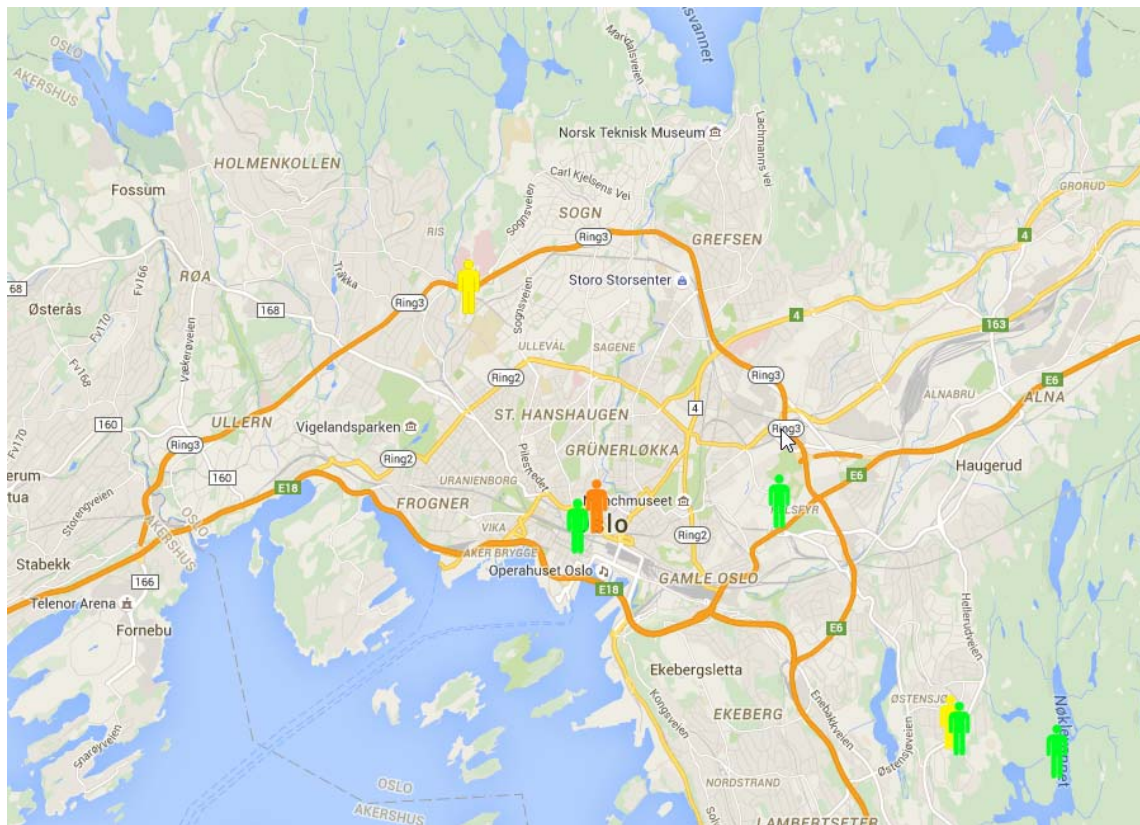


Figure 13-6 CityAir Map on local scale

13.5 Suggested improvements and conclusions

The most recent work has been to create the final version of CityAir products and services, including further improvements to visualization. The app is currently in a stable and production version, available thru Google Play and Apple's App Store.

During various trials CityAir was tested in real time environment with live data and environment by the location-officers from the EIs. Issues and problems were collected, registered and documented on the project's internal confluence space and the project's bug and issue tracking system JIRA. These experiences has been the foundation for the decisions on which improvements to make for the final version.



CITISense /... / Issue tracker
Issue Specifications
Created by mff on Oct 16, 2015

Edit Watch Share Tools


Key	Summary	T	Created	Updated	Due	Assignee	Reporter	P	Status	Resolution
CITISENSE-34	Not download ALL perception markers	🔴	Oct 16, 2015	Oct 22, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Fixed
CITISENSE-9	Vlasta_phone do not remeber user info	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Open	Unresolved
CITISENSE-12	GPS issues - not getting gps from phone	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Won't Fix
CITISENSE-3	Project info and app info in locale language	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Fixed
CITISENSE-2	Missing translation of two new tekst fields	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Fixed
CITISENSE-1	Change of text for the Only-wifi description	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Fixed
CITISENSE-6	Add extra "perception" to app description	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🟡In Progress	Unresolved
CITISENSE-5	Can't see the screen correctly when I choose the report tab	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Won't Fix
CITISENSE-4	Choose language popup screen	🔴	Oct 16, 2015	Oct 21, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Fixed
CITISENSE-7	Home button - remove others perceptions	🔴	Oct 16, 2015	Oct 22, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Fixed
CITISENSE-15	Update of Czech Language	🟢	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Fixed
CITISENSE-13	Info page on the perception markers	🟢	Oct 16, 2015	Oct 22, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Fixed
CITISENSE-14	Make more obvious the option to see all other users observations.	🟢	Oct 16, 2015	Oct 22, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🔴	🔴Resolved	Fixed
CITISENSE-21	Remove www.citi-sense-mob for other countries than Norway	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🟢	🔴Open	Unresolved
CITISENSE-18	Map focus when perception added	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🟢	🔴Open	Unresolved
CITISENSE-17	Have comments together with perception	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🟢	🔴Open	Unresolved
CITISENSE-16	Change to google api v2	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🟢	🔴Open	Unresolved
CITISENSE-20	Translate "map" "satellite"	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🟢	🔴Open	Unresolved
CITISENSE-19	Translate Backbutton	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🟢	🔴Open	Unresolved
CITISENSE-22	Have CITI-SENSE web page link as a hyperlink (for all language versions)	🔴	Oct 16, 2015	Oct 16, 2015		Mirjam Fredriksen	Mirjam Fredriksen	🟢	🔴Open	Unresolved

13.6 User steps for operation – reference necessary for usage

13.6.1 Installation

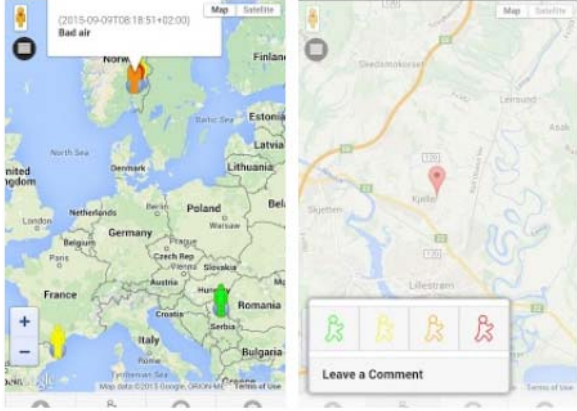
CityAir on Google Play Store

<https://play.google.com/store/apps/details?id=io.cordova.CityAir>



CityAir
Software department, NILU Sosial
PG-klassifisering

Legg til i ønskelisten **Installer**



Vil du oversette beskrivelsen til norsk med Google Oversetter? **Oversett**

CityAir will allow you to report on how you perceive the air quality where you are. This information will help to create a citizens air quality map, and answer the question, how do citizens perceive the air pollution in their city? Join us, and help us to colour the city. It is very easy!

You can rate the air quality next to you using four colours:

Green, meaning that the air quality is very good

Figure 13-7 CityAir description from Google Play store




CityAir on iTunes

<https://itunes.apple.com/us/app/cityair-perception/id1045646666?l=nb&ls=1&mt=8>

CityAir Perception

By NILU

Open iTunes to buy and download apps.



Description

CityAir will allow you to report on how you perceive the air quality where you are. This information will help to create a citizens air quality map, and answer the question, how do citizens perceive the air pollution in their city? Join us, and help us to colour the city. It is very easy!

[CityAir Perception Support](#) ...More

What's New in Version 1.2

- You can view perceptions made by other CityAir users reported last month, last week or today
- Language can be chosen from first popup screen
- Bug fixes and improvements

[View More by This Developer](#)

[View in iTunes](#)

+ This app is designed for both iPhone and iPad

Free

Category: Lifestyle

Updated: Nov 09, 2015

Version: 1.2

Size: 11.6 MB

Language: English

Seller: Nilu – Stiftelsen Norsk Institutt for Luftforskning © 2015 NILU

Rated 4+

Compatibility: Requires iOS 6.0 or later. Compatible with iPhone, iPad, and iPod touch.

Customer Ratings

We have not received enough ratings to display an average for the current version of this application.

Screenshots iPhone | iPad

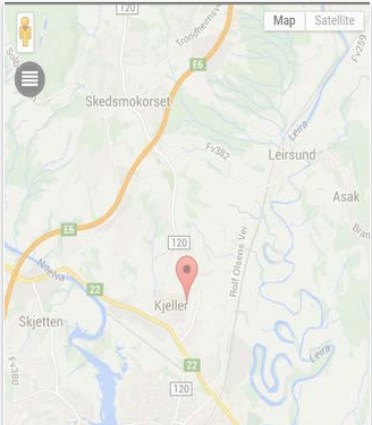
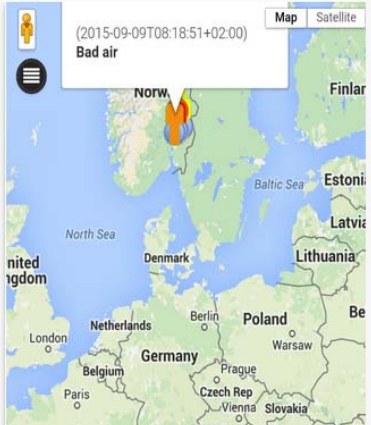
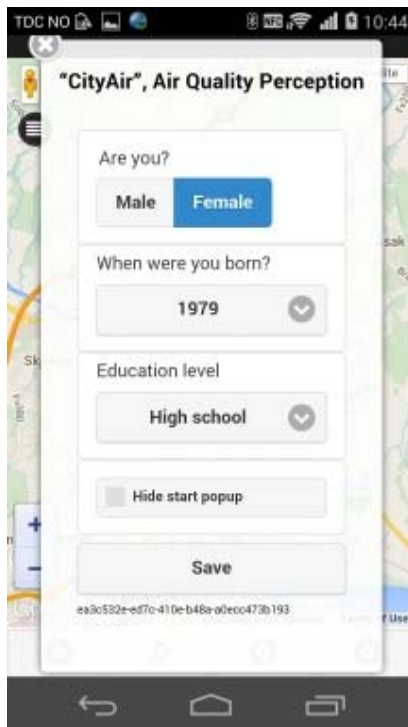



Figure 13-8 CityAir description from Apple iTunes App Store

13.7 Usage

This description of how to use the CityAir smartphone app, can be found on <http://oslo.citi-sense.eu/learnmore/cityairapp.aspx>



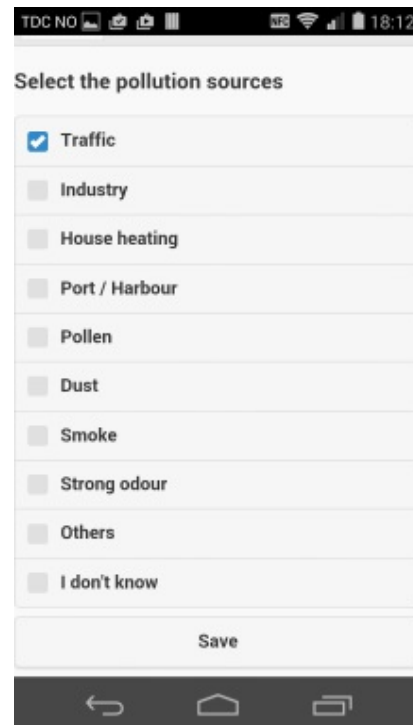
Screen 1: The first screen is always a popup screen with input fields for the user. The user can choose to stop this popup for being displayed every time, by checking the "Hide start popup" box



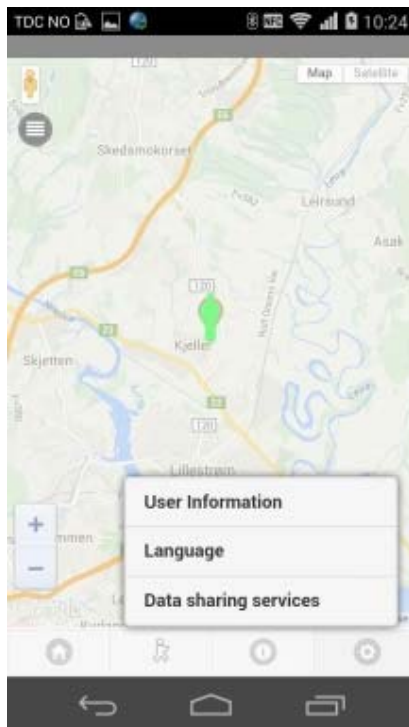
Screen 2: This is the second screen (or the first if the user chose to hide the popup). It contains all perceptions and comments the user has reported. The markers are clickable and will display the comment or the pollution source.



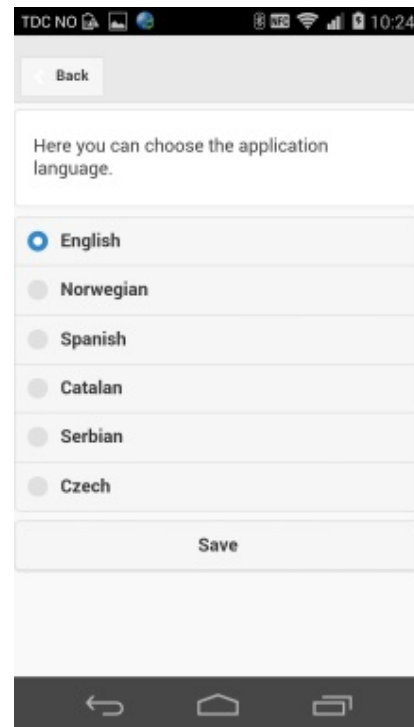
Screen 3: On the bottom of the screen, there are four buttons. The man is the report option. When clicked the user can choose between adding a coloured marked man according to the Air Quality perception, or leave a comment.



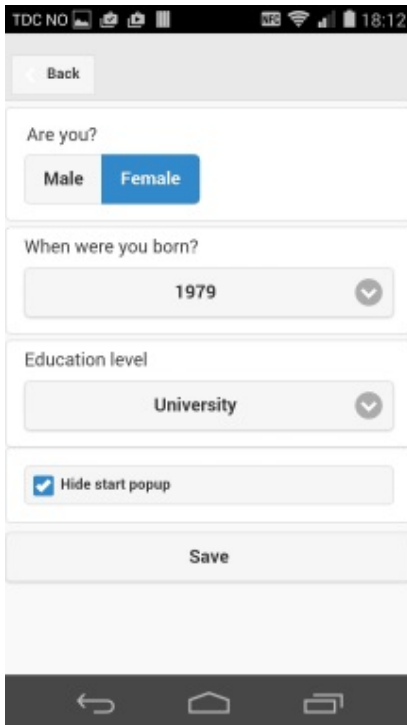
Screen 4: If the user chooses the yellow, orange or red man, he/her will be given a list of pollution sources to suggest before saving. Select the colour of the marker/man you think describe the current air quality at your location.



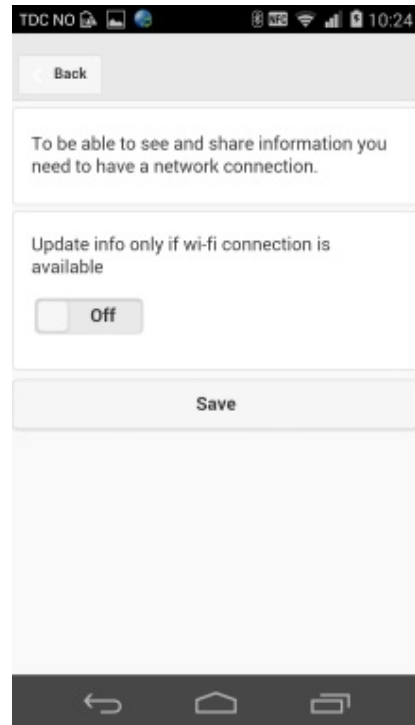
Screen 5: Under the gear button, the user can choose to change language, edit the user information or choose to only share data when on Wi-Fi under the data sharing services.



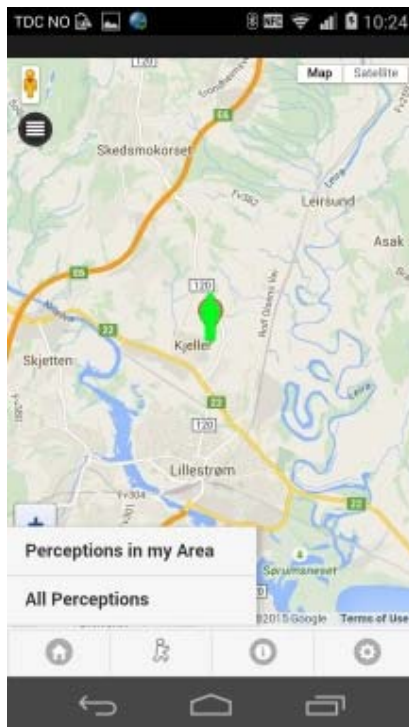
Screen 6: Select your preferred language and click on the Save button.



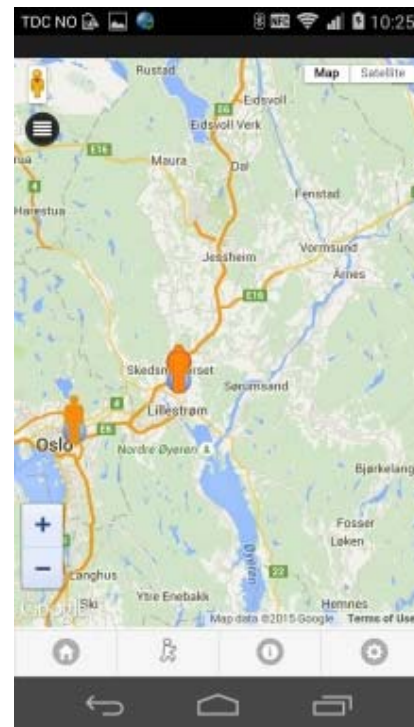
Screen 7: Edit user information such as gender, when you were born, and whether or not to hide the popup.



Screen 8: Data sharing services: If you want to share your information and upload it to the server only when you are using Wi-Fi, drag the button to "ON".



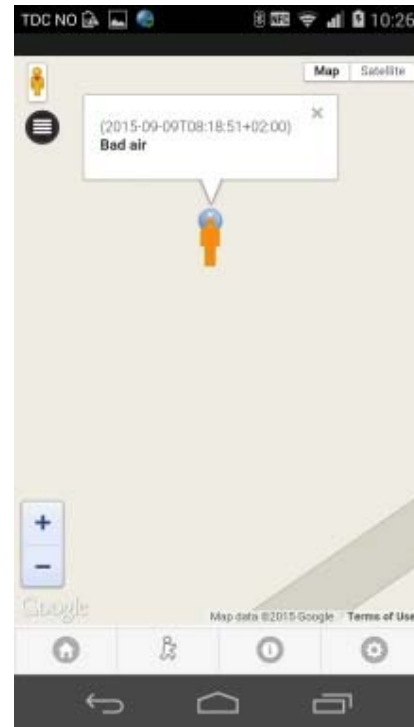
Screen 9: The user can choose to view just perception markers and comments that it local or all reported.



Screen 10: Click the icon in the upper left corner and select the perception you want to look at.



Screen 11: The markers are clickable and will display the date and time the measurement was done and the pollution source or the comment made.



Screen 12: Example of screen showing the details of a marker.



14 Annex I: VESNA Personal Air Quality Pack

14.1 Introduction to usage context

14.1.1 Introduction

The VESNA-based platforms as planned within the project were primarily designed for the use as personal/portable sensor units in selected pilot cities (Belgrade, Ljubljana and Vienna). Functionally similar personal sensor unit with some further sensors for gasses and context enrichment was developed after the Phase 1 tests which could also be used as a reference static outdoor platform for relative calibration and, if equipped with different set of sensors, it could also be used as an indoor platform. The new personal sensor unit VESNA-PAQ is equipped with the following sensors:

- VOC, H₂S, SO, NO, NO₂, O₃, CO (ppb)
- Temperature (°C) & Relative Humidity (%) - (Sensirion SHT21)
- Pressure, Accelerometer, Lightning

The personal sensor unit supports wireless connection to an Android smartphone and/or tablet via Bluetooth Smart Low Energy (BLE). The smartphone will in turn serve as the communication gateway towards the server using any of available data connections. Besides gateway functionality its role is to

enrich the data coming from the personal sensor unit with the GPS location, timestamps and user defined context.

The personal sensor unit is battery operated and includes Li-Po rechargeable battery providing 1300 mAh capacity. Charging of the battery is provided via micro USB connector. The autonomy of the personal sensor pack is in the range of 24 hours but depends on the mode of operation.

The personal sensor unit is housed in a plastic box with silicone shield. For calibration purposes raw data can also be downloaded via USB, which needs to be parsed and imported into a spread sheet. Alternatively, access to data was also provided via BLE interface and a purposely developed application.



Figure 14-1 The VESNA v2.0 Personal Air Quality (PAQ) Sensor Pack

14.1.2 Reference to CITI-SENSE Pilot workpackages

VESNA-based platform – PAQ is being further described and referred to in the following deliverables/work within the CITI-SENSE work packages: D8.2

14.1.3 Reference to CITI-SENSE Locations

VESNA-based platform – PAQ is planned to be used in some empowerment initiatives in Ljubljana pilot city, particularly in the school case study, as well as at some demonstrations at international events and conferences.

14.2 Architecture and interfaces used

14.2.1 Vesna WiFi system

The VESNA air quality monitoring system (Figure 14-2) comprises the VESNA personal sensor unit, smartphone app and the remote server. The smartphone app implements our custom LCSP (Lightweight Client Server Protocol) protocol which is used to send requests to the sensor node. VESNA



is set to listen on a specific TCP port to which the smartphone connects. After connection, the LCSP protocol is used to exchange information. When the data is downloaded from the sensor network, the mobile application has an option to visualize the data on graphs or as raw values. It also has an option to forward the data to the server in our custom JSON structure. The server stores the data in the database and translates the data to WFS and forwards it to the Snowflake over WFS-T in the XML format over HTTP POST request.

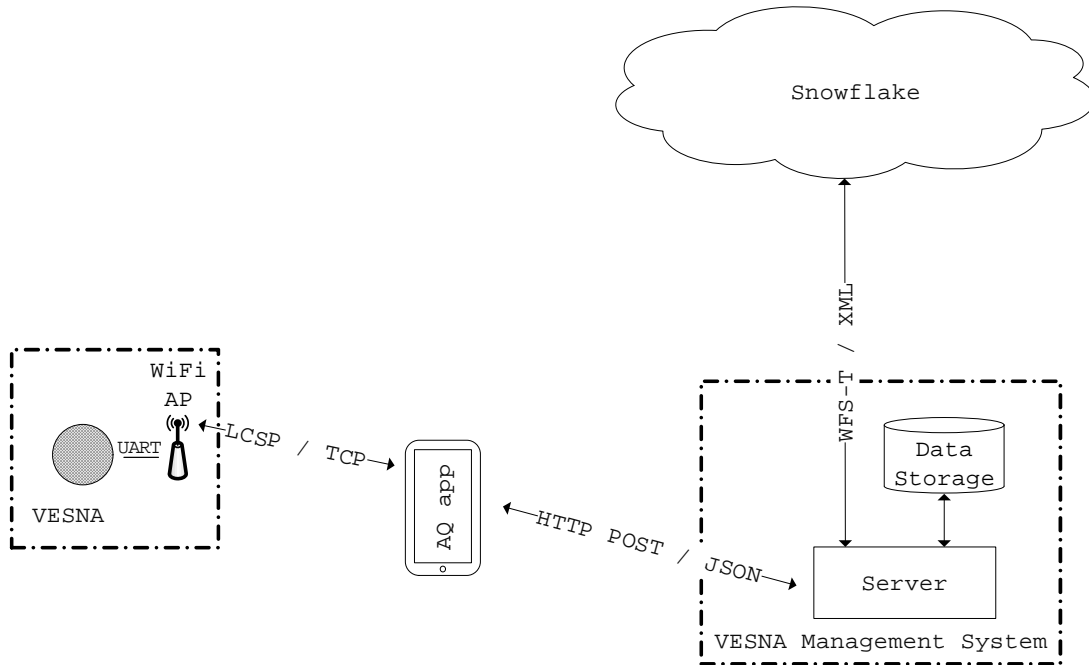


Figure 14-2 VESNA Air Quality Monitoring System Architecture

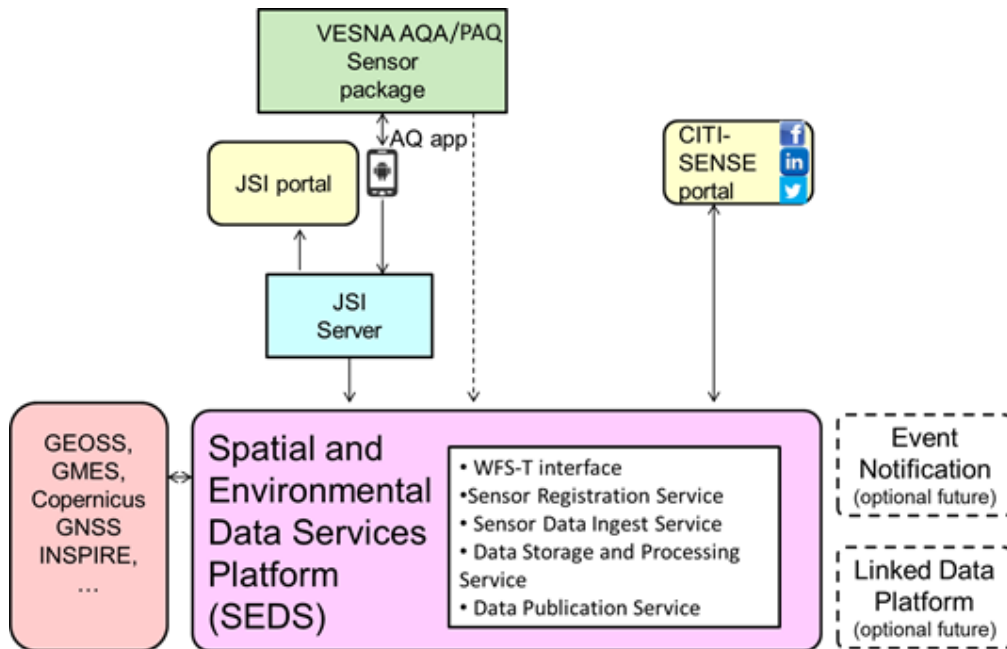


Figure 14-3 Data Flows for the VESNA PAQ architecture



For the purpose of communication between sensor node and our server hosted at JSI we developed a new protocol (Figure 14-4) called LCSP (Light-weight Client Server Protocol) which was inspired by HTTP protocol and is simple enough for a fast implementation on the VESNA platform. The protocol defines two requests, GET and POST which are understood by each VESNA sensor node. The GET is used for “safe” requests which do not change the state of the system and POST for “unsafe” requests which change the state of the system. The response is considered to be in a binary format, although it is normally in text format. Every response ends with an OK\r\n sequence and this is how the client recognizes the end of the response.

```

GET:
  GET resource?arg1=val1&arg2=val2&...&argN=varN\r\n

  resource:  abstract resource identifier
  examples:  - firmware/version
              - sensors/temperature
  arg1:      parameter 1 name
  val1:      value of parameter 1
  ...
  argN:      parameter N name
  valN:      value of parameter N

POST:
  POST resource?arguments\r\n
  Length=len\r\n
  <data, having len bytes length>\r\n
  crc=crc_value\r\n

  resource:  abstract resource, for example: firmware
  arguments: arguments given to the handler of POST
  example:   2.34/firmware
  len:       length of the data to send
  data:      possibly binary data, to be transmitted
  crc_value: value of CRC calculated on all the previous content
              except the line starting with crc=;
              value represented as an unsigned decimal number

Responses from the coordinator have the general form:

  <response to a specific request>\r\n
  OK\r\n

```

Figure 14-4 Resource access protocol

The protocol includes simple and efficient error handling mechanism (Figure 14-5). There are two types of errors. The first type of errors is JUNK-INPUT, which is the more common situation when the client mistypes the resource name and the parser on the node does not recognize it. Following this response the parser on the node expects 5 new lines, which reset the parser, and only after that a new attempt can be made to access the resource. The second type of errors is CORRUPTED-DATA, which means that CRC check was not successful indicating that an error happened somewhere on the line between the infrastructure and the gateway. The last situation will occur with very low probability.

```
<output from node, description of error>\r\n
<JUNK-INPUT or CORRUPTED-DATA>\r\n
\r\n
OK\r\n
```

Figure 14-5 Error handling

Figure 14-6 depicts our custom data JSON structure which is used for sensor data transport from the custom developed AQ mobile app to the server. The structure can include zero or more measurements for each sensor. Each sensor measurement value is annotated with a timestamp, latitude and longitude. Furthermore each collection of measurements is annotated with the free text context which is defined by the user of the AQ app.

```
{
  "sensors": [
    {
      "sensor_node_id": "CS001",
      "sensor_type": "AFE19024041",
      "measured_phenomenon": "carbon_monoxide",
      "unit_of_measurement": "ppb",
      "context": "test",
      "measurements": [
        {
          "timestamp": 1396361281,
          "latitude": 46.042437,
          "longitude": 14.487779,
          "value": 0
        }
      ]
    },
    {
      "sensor_node_id": "CS001",
      "sensor_type": "AFE19024041",
      "measured_phenomenon": "ozone",
      "unit_of_measurement": "ppb",
      "context": "test",
      "measurements": [
        {
          "timestamp": 1396361281,
          "latitude": 46.042437,
          "longitude": 14.487779,
          "value": 0
        }
      ]
    }
  ]
}
```

Figure 14-6 JSON Data Structure used by AQ app

14.2.2 VESNA Bluetooth Low Energy system

Bluetooth Low Energy (BLE) system is comprised of so called services which are taking care of their own functionalities e.g. heart rate monitoring, current time synchronization, temperature measurement etc. We are using modified Heart Rate Service for measurement transmission from BLE sensor node to mobile app on Android smartphone. We chose Heart Rate service because of early support on all BLE devices and therefore best compatibility with other devices and also because of similarities in data rate and purpose.



Modified Heart Rate Service data packet is comprised of 32-bit UNIX timestamp, which represents time of capture, unique 32-bit sensor ID, which represents actual sensor in database and 32-bit measurement value.

UNIX Timestamp (32 bit)	Sensor ID (32 bit)	Sensor Data (64 bit)
-------------------------	--------------------	----------------------

Figure 14-7 BLE packet

We solved problem of time synchronization on BLE sensor node with Current Time Service, where Android smartphone acts as current time server. The first thing application on BLE sensor node checks after connection establishment is current time on BLE sensor node. If there is a mismatch between current time on BLE sensor node and current time on Android smartphone, Current Time Service pulls current time information from server (phone).

After measurement start application on BLE sensor node periodically collects sensor values from all sensors and sends measurements to mobile application on phone, phone then collects all measurements and pushes them to data storage on remote server from where it can download data back for analysis and visualization.

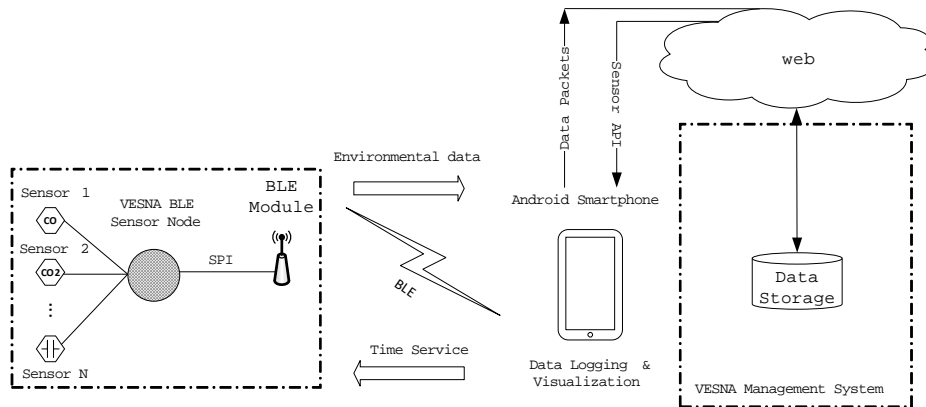


Figure 14-8 VESNA BLE system

14.2.3 JSI server side infrastructure

On the server at JSI we implemented the WFS-T 2.2 support in the form of a PHP server script in a combination with a PostgreSQL database used as a sensor register. The PHP code consists of four parts:

- Data parser and translator from JSI JSON to WFS-T XML format
- Database lookup (checks if the node has already been registered into Snowflake)
- WFS-T HTTPS authentication with provided username and password (WFS-T v2.2)
- WFS-T sensor registration
- WFS-T data insert

The flowchart depicted in Figure 14-9 describes the process from our server receiving the data to the WFS-T data insert. In parallel to the process that has been described we store the data in our local database on the PostgreSQL database server.

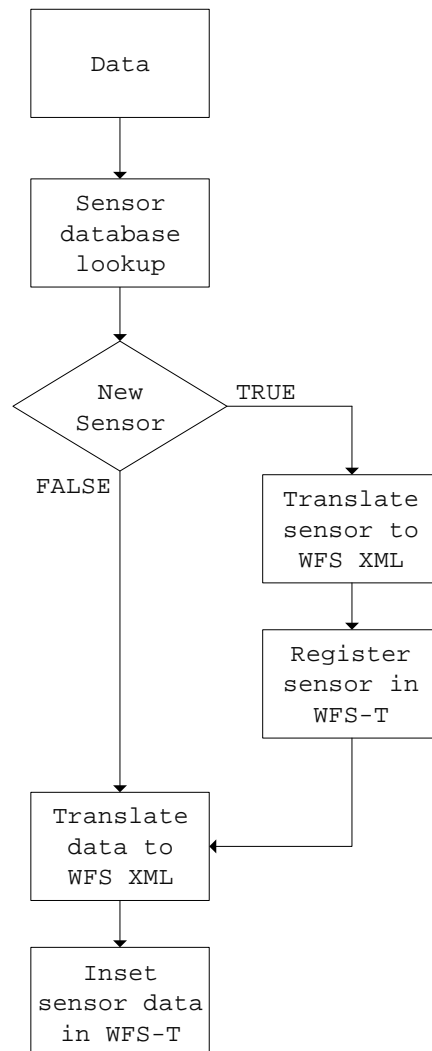


Figure 14-9 WFS-T support flowchart

14.3 Data (volume, velocity) being stored and retrieved

14.3.1 VESNA AQA v1.0

AQA portable sensor units are used in pilot cities Belgrade, Ljubljana and Vienna. Each city has 5 units in testing phase 1, so together there are 15 units in regular testing use. Each unit can collect at most 8 measurements (one from each sensor) each 5 seconds (5 second being the shortest sampling interval) and at least 8 measurements every 5 minutes (5 minutes being the longest sampling interval). Possible sampling intervals are also 30 second and 1 minute. Each measurement is composed of a timestamp, latitude, longitude and value. A rough estimation of 24h measurement collection would thus be for a single AQA unit between 2304 and 138,240 measurements. Therefore the highest amount of data to be stored per day from all 15 units in testing phase would be around 2 million sensor measurements.

14.3.2 VESNA PAQ v2.0

PAQ portable sensor units will be used in pilot cases in Ljubljana and there are 5 PAQ units altogether. Each unit can collect at most 12 measurements (one from each sensor) each 5 seconds (5 second being the shortest sampling interval). Each measurement is composed of a timestamp, latitude, longitude and value. A rough estimation of 24h measurement collection would thus be for a single unit around 207360 measurements. Therefore the highest amount of data to be stored per day from all 5 units in testing phase would be around 1 million sensor measurements.

14.4 User applications/apps/processing/visualisations

JSI AQ mobile app for users was developed for Android smartphones and is needed for connecting and collecting data from the VESNA PAQ and AQA devices, for basic visualization and for the transfer of data to the remote server. The application supports two operating modes. The first is Wi-Fi mode and the second is Bluetooth Smart in low energy (BLE) mode. WiFi mode is used with VESNA-AQA units whereas BLE mode is used with VESNA-PAQ units.

The application consists of 5 tabs: Pair, Data, Post, Context and Log. The tab PAIR is used to set up the connection with a VESNA unit and the tab DATA is used to visualize the data, which is stored locally on the smartphone/tablet. Simple graphs of subsequent samples can be drawn for each of the supported sensors. The aim of these graphs is primarily for initial in-field cross check if the unit is working according to expectations, while for actual end user visualization the use of a common CITI-SENSE mobile app is foreseen.

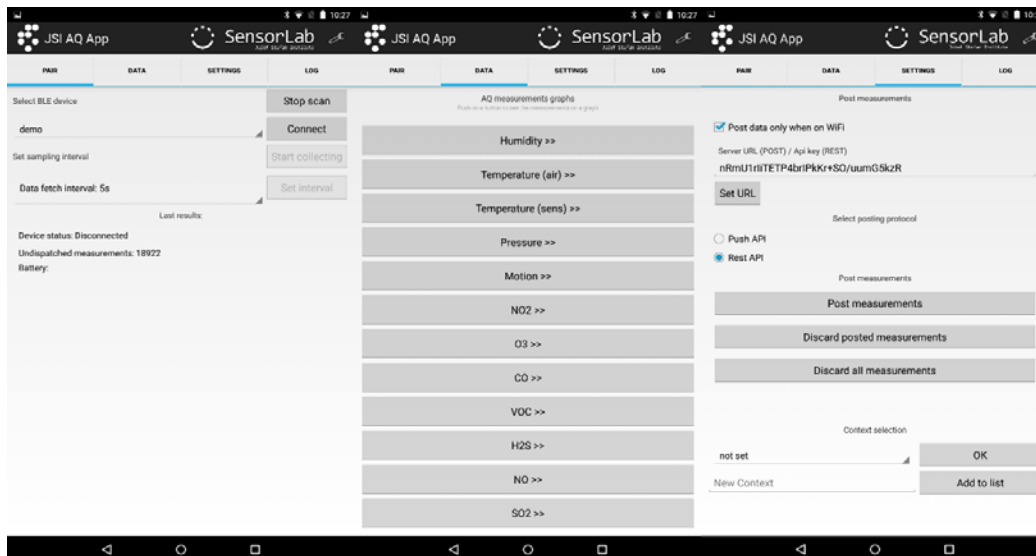


Figure 14-10 JSI AQ App v2.0

The POST tab is used to forward the collected measurements to the online database at the remote server, which stores the data and forwards it further to the Snowflake platform. In this tab a user has to choose the appropriate type of connection. If using a flat data plan on the smartphone, any internet connection type can be selected, but user can also restrict data uploading only via Wi-Fi connections, when in the range of a known Wi-Fi access point. When appropriate type of connection is chosen one can *post measurements*. There is also an option to clean up the posted or all measurements stored locally, so that they no longer appear in the DATA tab visualization.



14.4.1 Suggested improvements and conclusions

We came to the following conclusions during the WFS-T support implementation:

- The documentation on Confluence was sufficient for understanding the system
- The documentation on Confluence was sufficient for successful implementation of WFS-T support
- The documentation on Confluence was sufficient for successful testing of the WFS-T implementation



15 Annex J: DunavNet EB700++ Mobile Sensor device

15.1 Introduction to usage context

15.1.1 Introduction

The DunavNet **EB700++ (EkoBUS700++)** device enables measurement of air pollution and atmospheric conditions at the location of a vehicle determined by a built-in GPS module. Also, the device can be mounted at any fixed indoor/outdoor location for monitoring local environment parameters. The collected measurements are transferred to the back-end server via GPRS where they are stored and further processed.

The main monitoring environmental parameters are: CO, CO₂, NO, NO₂, SO₂, O₃, air pressure, temperature, humidity, with ability to extend the monitored parameters per specific requirements. There is a possibility to connect external devices and additional sensors via USB or RS232 communication port. Dylos DC 1700 device for Particulate Matter measurements is integrated with EB700++, therefore P10 measurements are being sent via EB700++ device to the back-end server.

- Gas sensors are Alphasense B4 family electrochemical types and IRC-A1 for CO₂ (Infrared). T+%RH is Sensiron SHT11. Air pressure sensor is MPXA6115AC6U.
 - **CO₂** (0-5000ppm)
 - **O₃** (0-2ppm)
 - **NO-B4** (0-20ppm)
 - **NO₂-B4** (0-20ppm)
 - **SO₂-B4** (0-20ppm)
 - **CO-B4**(0-50ppm)
 - **MPX4115** (15 - 115 kPa)
 - **SHT71**(-40°C – 123°C, 0-100%)
- It is a possible to connect external devices and additional sensors via USB or RS232 communication port. Dylos DC 1700 device for Particulate Matter measurements is integrated with EB700++, so measurements obtained from Dylos DC 1700 are sending via EB700++ device to the back-end server.
- A true Laser Particle Counter with 2 size ranges (>0.5 & >2.5 microns) - small (bacteria, mold, etc) large (pollen, etc.).

DNET ekoBUS700++	
Total Weight	500 g approx.
Dimensions	- Indoor: 105x145x30mm + sensors board 80x100mm - Outdoor without Dylos DC 1700 : 200x130x75 mm - Outdoor with Dylos DC 1700: 315x205x100mm
Supply voltage range	- For auto industry 8-28V DC - For fixed/indoor locations 12V DC
Power consumption	max 10W
Ambient temp. range	-30 to +50 °C



Figure 15-1 The DNET EB700++

In order to improve the quality of the device, a new version of device is developed, namely EB800. The following improvements are addressed:

- Electronic design is improved in order to reduce noise introduced in the device
- OPC Alphasense sensor is added instead of Dylus;
 - o OPC-N2 (PM₁, PM_{2.5} and PM₁₀ measurements)
- Noise sensor is added
- New box design is used
- An option for battery supply is added
- In the previous version measurements were taken only from one sensor pin, now measurements are taking from two pins, and also temperature compensation is added (PT1000 sensors)



Figure 15-2 The DNET EB800

15.1.2 Reference to CITI-SENSE Pilot workpackages

EB700++ is being further described and referred to in the following earlier deliverables/work within the CITI-SENSE workpackages: WP7 D7.1, D 7.3/D7.4/D7.5 WP8 D8.1, D8.2

15.1.3 Reference to CITI-SENSE Locations

The EB700++ is used in Belgrade.

15.2 Architecture and interfaces used

The following shows an architectural picture of the data flow related to this sensor platform.

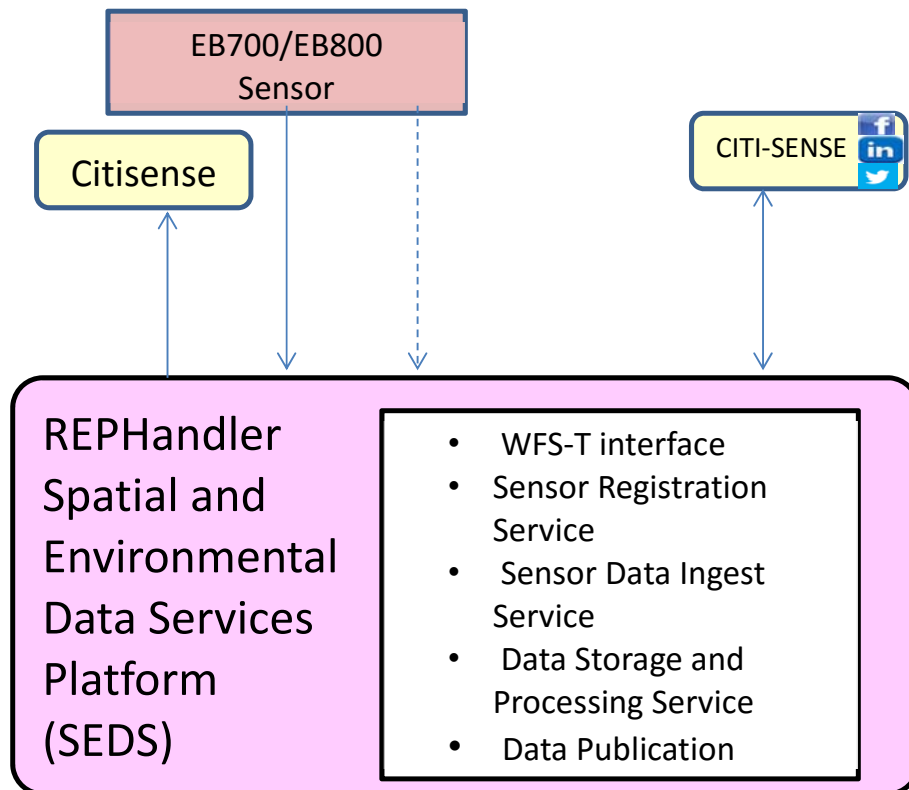


Figure 15-3 Data Flows for the CITI-SENSE DNET architecture

EB700++ establishes a GPRS connection with DNET server, where the data are stored, processed and visualized. Users can log on to the web portal and access measured data.

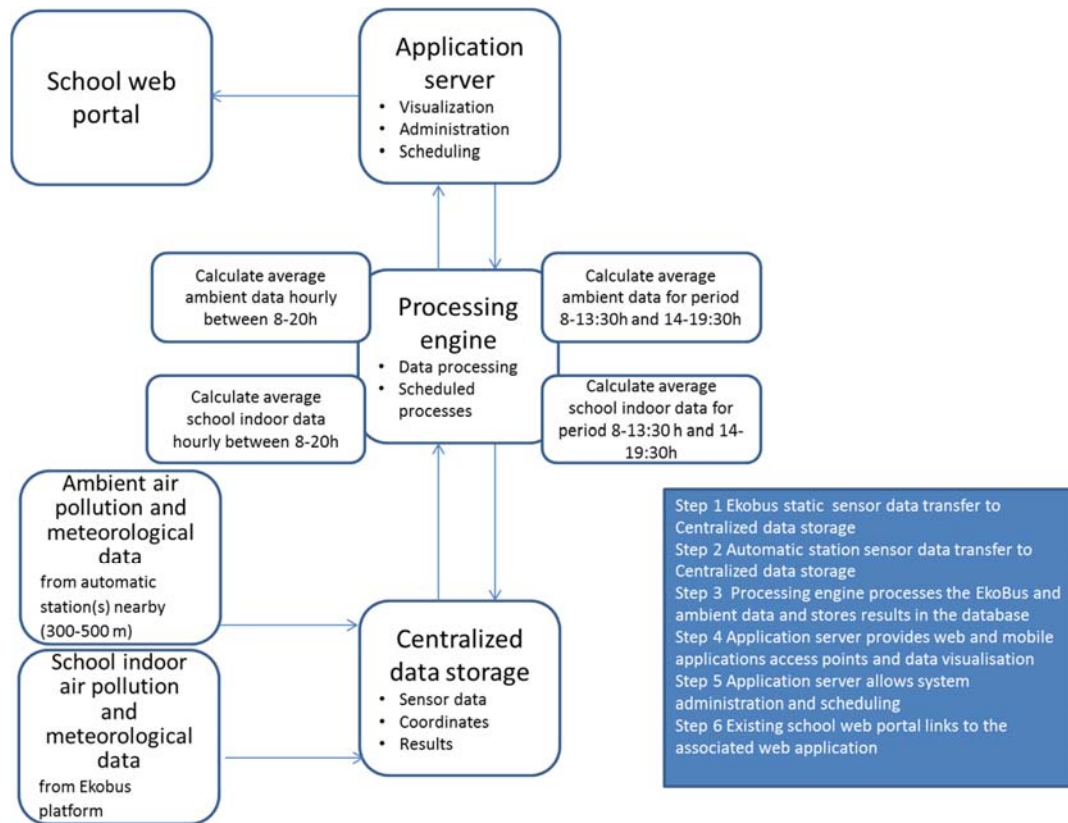


Figure 15-4 EB700++ System Architecture

Here is the summary of overall system building blocks and features:

The EB700++ devices with sensors are connected to the platform serving as the back-end cloud infrastructure. The collected data from the sensors is transmitted to the back-end infrastructure (Communication Server component) via the mobile network using the CoAP protocol. The cloud platform and associated building blocks enable the core features such as:

- the permanent storage of data (data storage within the Data Server component),
- data cleaning and processing functionality (Data Server and Data Processing components)
- a set of visualization widgets (Visualization Engine component).

Web server enables creation of the associated interfaces in order to allow communication of the web and mobile applications with the rest of the system.

Analytics engine enables smart interpretation of real-time and historical data from combination of data sources.

Data aggregation engine enables creation of additional services through combination of data from different services.

Data processing engine provides advanced data interpretation and preparation for visualization including the AR content with the help of Augmented Reality content generator.

Alternative platform adapters enable the services to be deployed and provisioned on the third-party platforms using the existing infrastructure within the enterprises or cities.



The sensor data is transferred to the DNET server directly from the device via GPRS and then pushed to the CITI-SENSE server using the appropriate web service. Ultimately, the data can be sent directly to the CITI-SENSE server at the later stage if required.

EB700++ device data usage

Device sends raw data packed into the XML formatted messages.

Sample message:

```
<EC>
  <n>EB700</n>
  <ei>355255040039514</ei>
  <si>220032302262587</si>
  <d>
    <sD>
      <h>0036</h>
      <t>0034</t>
      <p>0992</p>
      <CO>3595</CO>
      <O3>0003</O3>
      <CO2>1098</CO2>
      <NO2>0987</NO2>
      <NO>2061</NO>
      <SO2>2041</SO2>
      <AUD>2058</AUD>
    </sD>
    <gD>095537.000,4515.0020N,01950.4004E,0.8,110.2,3,28.73,0.14,0.07,1802
    11,10</gD>
  </d>
</EC>
```

Where:

- Root element is <EC>
- Device name is in the text content of the <n> element
- IMEI and IMSI for the device modem and SIM card are stored in <ei> and <si> elements respectively.
- Data are placed in the <d> element.
- Sensor readings are placed as child elements of the <sD> element. Sensor reading contains measurements for relative humidity <h>, temperature <t>, atmospheric pressure <p>, gases levels (CO, CO₂, O₃, NO, NO₂, SO₂). Sound pressure level is in the text content of the <AUD> element.
- GPS data are stored in the <gD> element in the following format:



- {Time from GPS},{Latitude},{Longitude},{Dilution of precision},{Altitude},{Fix type},{Course},{Speed km/h},{Speed knots},{Date},{#of satellites}
- Server component for the data processing is REP Handler (RH). RH is designed for bus vehicles monitoring system.
- Server receives data on the following address: **{RH_HOST:RH_PORT}/rephandler/**
- Data may be accessed through ecobus service on the following address: **{RH_HOST:RH_PORT}/{IMEI}**

Example: <http://89.216.116.166/rephandler/ecobus/355255040040264>

RH response (XML formatted RDF message):

```
<rdf:RDF xmlns:mo="http://www.ict-sensei.org/MeasurementAndObservation#"
xmlns:nmib="http://www.ict-sensei.org/NodeMIB#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<nmib:Location mo:hasTimeStamp="23 Aug 2013 14:15:42" mo:latitude="-1.0" mo:longitude="-1.0"/>
<nmib:SO2Level mo:hasDecimalValue="13.0" mo:hasDescription="Sulfur dioxide SO2"
mo:hasUnit="ppm"/>
<nmib:COLevel mo:hasDecimalValue="10.0" mo:hasDescription="Carbon monoxide CO"
mo:hasUnit="ppm"/>
<nmib:NO2Level mo:hasDecimalValue="1.5" mo:hasDescription="Nitrogen dioxide NO2"
mo:hasUnit="ppm"/>
<nmib:CO2Level mo:hasDecimalValue="350.0" mo:hasDescription="Carbon dioxide CO2"
mo:hasUnit="ppm"/>
<nmib:TemperatureLevel mo:hasDecimalValue="26.0" mo:hasDescription="Temperature"
mo:hasUnit="C"/>
<nmib:HumidityLevel mo:hasDecimalValue="53.0" mo:hasDescription="Humidity"
mo:hasUnit=""/>
<nmib:PressureLevel mo:hasDecimalValue="101.63" mo:hasDescription="Pressure"
mo:hasUnit="kPa"/>
</rdf:RDF>
```

15.2.1 DNET server side infrastructure

System contains two web .NET MVC 5 applications and .NET web API. They are running on IIS (7.5 and above). Databases are two MS SQL Server 2008 R2 (or newer).

First application, "Administration" is for administering users and devices. Second application, EkoNet has role in preparation and presentation of data collected from devices. Web API contains functions for communication between OCB and database.

Averaged data for 1h and 24h are prepared with the help of FIWARE component Orion Context Broker.

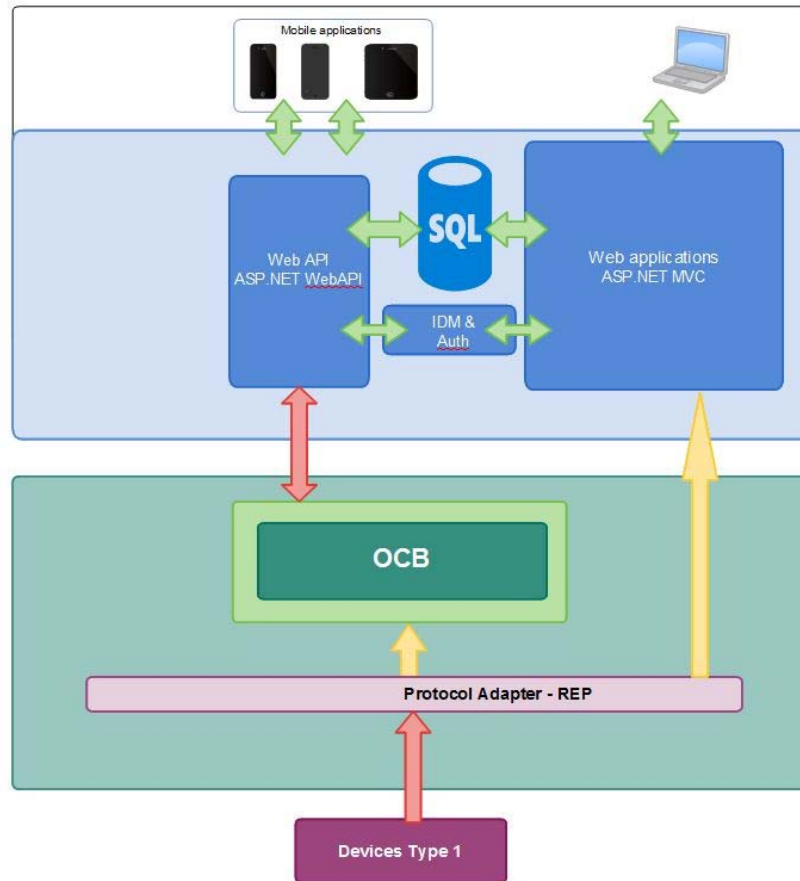


Figure 15-5 Server side infrastructure

15.3 Data (volume, velocity) being stored and retrieved

The data received from EB700++ consists of data collected during the sampling period. Sampling period is adjustable, and for the purposes of pilot in Belgrade it was fixed on 1 minute. So data from sensors were collected and sent every one minute to the server via GPRS. Measurements are collected from the following sensors: CO, CO₂, NO, NO₂, O₃, RH, Temperature, Air pressure, PML and PMS. Also GPS coordinate of the device is sent, and ID of the device (IMEI of the GPRS modem).

Data are sent in XML format, and the length of data message is about 300 Bytes.

15.4 User applications/apps/processing/visualisations

As it is already explained, the collected measurements are transferred to the back-end server via GPRS where they are stored and further processed. Data are stored in the centralized data storage for device position and sensors.

Application server is used for the visualization, administration and scheduling. The data can be visualized in a real-time using the appropriate Web application (portal) or mobile application. On the web portal it is possible to see real-time, and historical values. Measurements can be retrieved for specified defined period in the past. Visualization is done using third-party widgets.



Basic Operation and features *within* the ekoBUS700++

- Centralized data storage for coordinates and sensors data
- Data processing of the sensors data
- Application server is used for visualization, administration and scheduling
- Web application (portal), possible use on mobile devices as well
- Customized for CITI-SENSE pilot
- Shows real-time, and historical values
- Measurements could be retrieved for the desired defined period in the past
- Visualization is done using third-party widgets
- Highly customisable, can be modified to suit use-case or specific requirements
- Data tables and graphs
- Secure data storage

The following data coming from the EB700++ platform are stored on the server and can be viewed via WEB portal:

- Real-time sensor measurements
- Historical data (stored sensor measurements over the period of time)
- GPS data

Available at <http://srv.dunavnet.eu/ModuleEkoNet/Data/Index>

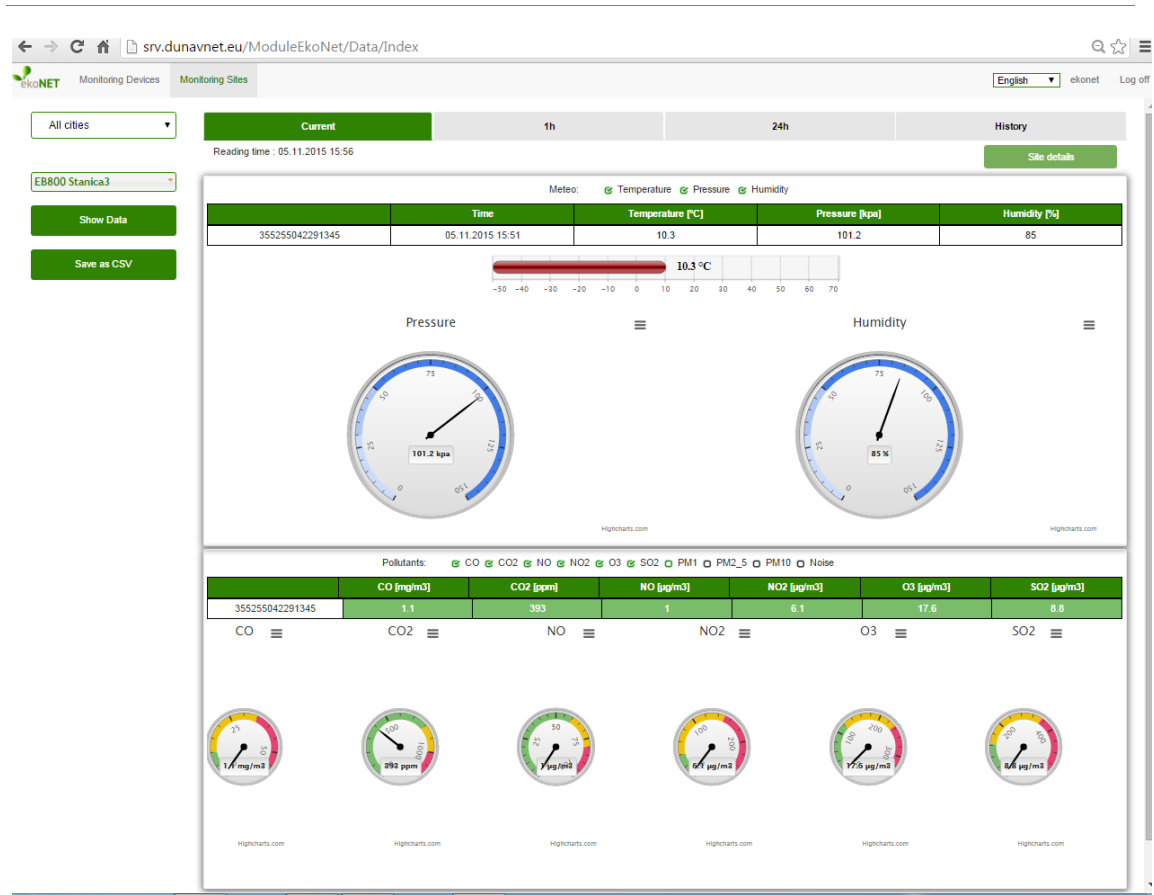


Figure 15-6 Web portal

16 Annex K: South Korea UPA example system

16.1.1 System Architecture

Error! Reference source not found. shows the architecture overview of an example system from Seoul, which has been developed by the CITI-SENSE partner KICT, demonstrating a practical example of the use of the ISO19154, “Geographic information – Ubiquitous public access – reference model (UPA)” compliant UPA architecture described in D7.5 part 1 chapter 10.

The architecture is described by dependency relationships according to the call relationship between the main elements and the elements. The architecture is largely classified into presentation layer, logic layer, and data layer, and each layer consists of components. In addition, the Sensor Data Acquisition interface is implemented by the components from the data layer.

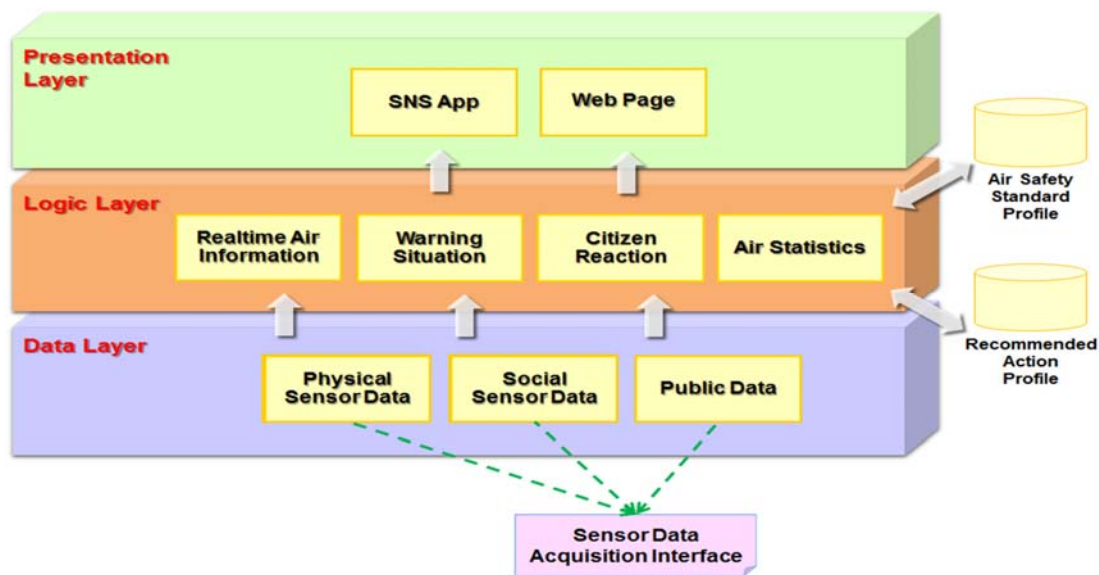


Figure 16-1 Architecture of the Seoul Example System

16.1.2 Software Configuration

The software overall configuration is as follows: Database stores the data, Web Server collects and provides data, and Web Client provides the UPA service. Here, the entire software is configured based on OpenSource.

Database

With regard to the database, PostgreSQL, which is the DBMS provided as OpenSource, is used to save data, and the PostGIS plug-in is installed and used in order to process the Geometry data.

Web Server

The web server is established based on Apache/Tomcat. To provide the overall web service, the electronic government-framework (egov-framework: spring-framework-based), which is used as a public business development framework, is used. Furthermore, data are provided through web client and HTTP communication using Restlet, which provides RESTful service; a Socket module and HTTP

Client module are included in the configuration to connect with external systems. To provide service regarding Geographic Information System (GIS) data, Web Map Service (WMS) and Web Feature Service (WFS) are provided using GeoServer, and GeoServer is used in connection with the OpenLayers of the web client.

Web client

The web client is developed on the basis of JSP and Javascript. Its User Interface (UI) is supplemented by adding the JQuery library, and map data rendering is facilitated using the OpenLayers library.

Data exchange

Ordinary data delivered from the web server to the web client is provided using a data type that can be easily managed with Javascript, such as JSON and SML, through the RESTful web service. Geospatial data is provided through GeoServer using a type of WMS and WFS data, which are the Open Geospatial Consortium (OGC) standards.

In the case of collecting sensor data, such information is gathered using the data types SenML, XML, JSON, etc., depending on the connection method. When providing data to the CITI-SENSE Platform is necessary, such data are provided using a data type (SenML, etc.) required by the Platform.

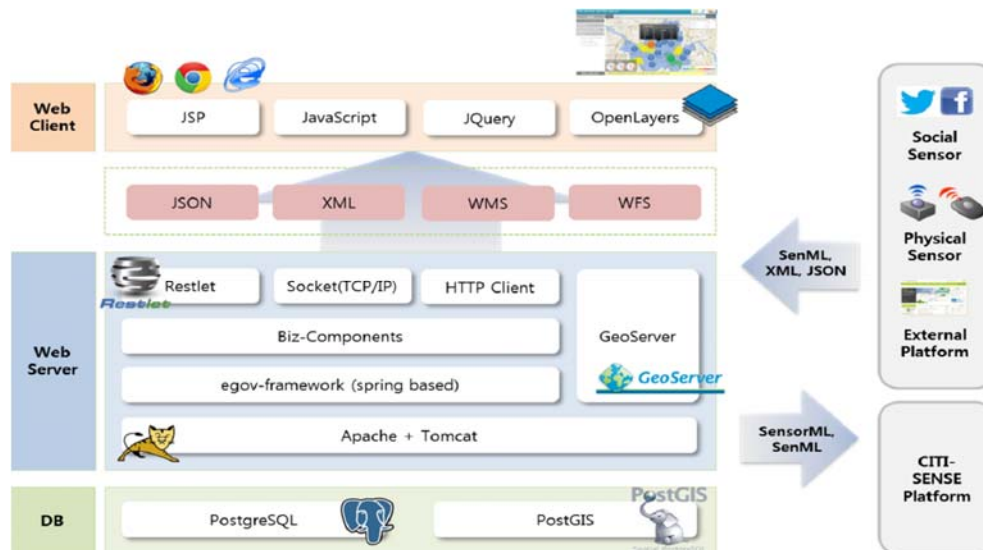


Figure 16-2 The web-based pilot system under development

16.1.3 Sensor Data Collection

The pilot system aims to collect data from physical sensors and social sensors, in which a physical sensor measures air pollutants and a social sensor concerns citizens' reaction to air quality. Since the air quality information, which is air pollutant measurements, is used to be obtained from the OpenAPI, air pollution data is collected through an interface with the related system.



Also, SALT LUX, which is a partner organization in Korea, has a role to collect and analyze social data and to describe how this can have an interface with the CITI-SENSE platform. Thus, Seoul citizens' reaction to air pollution can be collected with an interface service with the CITI-SENSE platform.

Thus, to collect sensor data, the web-based pilot system supports an integrated interface service under various communication environments. Also, prior to store each sensor onto a database, the data processing method analyzes and validates the collected data to determine whether it has a correct format or not.

Component	Description
Data Interface Policy Management	<ul style="list-style-type: none"> • Management of a data collection schedule and period.
Log Management	<ul style="list-style-type: none"> • Management of a collection log of the sensor data.
Sensor Data Management	<ul style="list-style-type: none"> • The data processing method according to the sensor interface policy • The data management and storage for monitoring air pollution

The pilot system, the web-based GIS, is able to manage and analyze the air pollution data from each physical sensors, for which air pollution data from each of those physical sensors follows the spatial information standard such as WMS, WFS and GeoJSON formats.

Sensor information is organized for including the air environment information and citizens' reactions and also, it is configured to provide information to users through the RESTful web service, which is based on the HTTP protocol.

To provide a location-based analysis, the integration of sensing information and analytical information such as statistical data is interfaced with external services utilizing the RESTful Web service, similar to the sensor information, and considers the data expression and provides a data structure that can be easily used by external systems.

Component	Description
GIS Information management	<ul style="list-style-type: none"> Offers a function providing spatially informative data in a service format, and uses the WMS and WFS formats.
Sensor Information Management	<ul style="list-style-type: none"> Offers a function providing spatially informative data in a service format, and manages and provides information through the RESTful Web service using the JSON format.
Analysis Information Management	<ul style="list-style-type: none"> Provides service by converting analyzed data into a user-convenient data structure, and provides information through the RESTful Web service using the JSON format using XML, JSON format.

The location-based data are analyzed when an analysis service related to the location requested by users is required, and the results are provided.

When new data are generated through the integration of sensing information, such as the SNS Human-Comprehensive Air-quality Index (H-CAI), the results are provided by applying the related algorithm.

The data required by users, such as yearly and monthly statistical data, and the long-term data trends are analyzed, and the results are provided.

Component	Description
Location-based Analysis	Provides a function for analyzing related information based on the input location, and returns the result.
Integration of sensing information	Produces a new index (i.e., H-CAI) or information by analyzing the air environment information and citizens' reactions (social sensor).
Statistical management	Analyzes the average environment information or long-term trends based on the data stored in a DB, and provides statistical information.

To collect the air environment information, public data on Seoul (data.seoul.go.kr) are used. The Seoul public data list has a total of 4,007 items, 247 of which are environment information on Seoul; of these 247 items, 20 are related to air and can be used in this service and data list. The 20 items related to the air environment of Seoul, and the data to be provided for each public service, are investigated.



No	Service Name	Output Data
1	Time-averaged air pollution level	<ul style="list-style-type: none"> Measured time and date, measured location, concentration of nitrogen dioxide (ppm), ozone concentration (ppm), concentration of carbon monoxide (ppm), sulfur dioxide (SO₂) (ppm), fine dust ($\mu\text{g}/\text{m}^3$), ultrafine dust ($\mu\text{g}/\text{m}^3$)
2	Status of ozone alert issuance per year	<ul style="list-style-type: none"> Year, number of issuances, issuance date, maximum concentration (ppm)
3	Status of fine dust alert issuance per year	<ul style="list-style-type: none"> Year, number of issuances, issuance date, maximum concentration ($\mu\text{g}/\text{m}^3/\text{hour}$)
4	Status of yellow dust issuance alert per year	<ul style="list-style-type: none"> Year, number of viewed issuances, number of issuance viewing dates, number of alert issuances, number of observed dates, maximum concentration ($\mu\text{g}/\text{m}^3/\text{hour}$)
5	Daily weather observation information	<ul style="list-style-type: none"> Observed date (°C), name of the location (°C), average temperature (°C), minimum air temperature (°C), maximum air temperature (°C), average humidity (%), minimum humidity (%), maximum humidity (%), average wind velocity (m/s), maximum wind velocity (m/s), amount of precipitation (mm)
6	Real-time weather observation information per weather station	<ul style="list-style-type: none"> Measured date and time, name of the location, temperature (°C), humidity (%), wind direction 1, wind direction 2, wind velocity (m/s), precipitation (mm), sun radiation (M/m^2), sunshine (hour)
7	Roadside daily average air environment information per period	<ul style="list-style-type: none"> Measured date, identification of roadside, name of the location, fine dust ($\mu\text{g}/\text{m}^3$), ozone (ppm), concentration of nitrogen dioxide (ppm), concentration of carbon monoxide (ppm), sulfur dioxide (SO₂) (ppm)
8	Roadside time-averaged air environment information per period	<ul style="list-style-type: none"> Measured date and time, area code, name of area, location code, name of location, 1 hour of fine dust ($\mu\text{g}/\text{m}^3$), 24 hours of fine dust ($\mu\text{g}/\text{m}^3$), ultra-fine dust ($\mu\text{g}/\text{m}^3$), ozone (ppm), concentration of nitrogen dioxide (ppm), concentration of carbon monoxide (ppm), sulfur dioxide (SO₂) (ppm)
9	Daily average air environment information per period	<ul style="list-style-type: none"> Measured date, name of area, name of place, fine dust ($\mu\text{g}/\text{m}^3$), ultra-fine dust ($\mu\text{g}/\text{m}^3$), ozone (ppm), concentration of nitrogen dioxide (ppm), concentration of carbon monoxide (ppm), sulfur dioxide (SO₂) (ppm)
10	Real-time weather observation information per area	<ul style="list-style-type: none"> Measured date, name of area, name of place, fine dust ($\mu\text{g}/\text{m}^3$), ultra-fine dust ($\mu\text{g}/\text{m}^3$), ozone (ppm), concentration of nitrogen dioxide (ppm), concentration of carbon monoxide (ppm), sulfur dioxide (SO₂) (ppm), integrated air environment level, integrated air environment index, index decision material

No	Service Name	Output Data
11	Daily average air pollution level	<ul style="list-style-type: none"> Measured date, name of place, concentration of nitrogen dioxide (ppm), concentration of carbon monoxide (ppm), sulfurous acid gas (SO₂) (ppm), fine dust ($\mu\text{g}/\text{m}^3$), ultra-fine dust ($\mu\text{g}/\text{m}^3$)
12	Monthly average air pollution level	<ul style="list-style-type: none"> Measured monthly, name of place, concentration of nitrogen dioxide (ppm), concentration of carbon monoxide (ppm), sulfurous acid gas (SO₂) (ppm), fine dust ($\mu\text{g}/\text{m}^3$), ultra-fine dust ($\mu\text{g}/\text{m}^3$)
13	Yearly average air pollution level	<ul style="list-style-type: none"> Measured year, name of place, concentration of nitrogen dioxide (ppm), concentration of carbon monoxide (ppm), sulfurous acid gas (SO₂) (ppm), fine dust ($\mu\text{g}/\text{m}^3$), ultra-fine dust ($\mu\text{g}/\text{m}^3$)
14	Fine dust forecast/alert for Seoul	<ul style="list-style-type: none"> Announcement time (YYYYMMDDHHMI), forecast/alert identification (f, forecast; a, alert), contaminated material type, forecast grade, action tips (forecast), alert level, action tips (alert)
15	Ozone forecast/alert for Seoul per area	<ul style="list-style-type: none"> Announcement time (YYYYMMDDHHMI), forecast/alert identification (f, forecast; a, alert), contaminated material type, forecast level, action tips (forecast), alert level, action tips (alert)
16	Yellow sand alert for Seoul	<ul style="list-style-type: none"> Contaminated material type, alert level, action tips (alert)
17	Real-time average air environment of Seoul	<ul style="list-style-type: none"> Integrated air environment grade, integrated air environment index, index decision material, nitrogen dioxide (ppm), carbon monoxide (ppm), sulfurous acid gas (SO₂) (ppm), fine dust ($\mu\text{g}/\text{m}^3$), ultra-fine dust ($\mu\text{g}/\text{m}^3$)
18	Real-time average air environment per local district	<ul style="list-style-type: none"> Integrated air environment grade, integrated air environment index, measured date, admin code of measurement station, name of measurement station, index decision material, nitrogen dioxide (ppm), carbon monoxide (ppm), sulfurous acid gas (SO₂) (ppm), fine dust ($\mu\text{g}/\text{m}^3$), ultra-fine dust ($\mu\text{g}/\text{m}^3$)
19	Ultra-fine dust forecast/alert of Seoul	<ul style="list-style-type: none"> Announcement time (YYYYMMDDHHMI), forecast/alert identification (f, forecast; a, alert), contaminated material type, forecast level, action tips (forecast), alert level, action tips (alert)
20	Roadside real-time air environment per measuring station	<ul style="list-style-type: none"> Measured date, identification of roadside, fine dust ($\mu\text{g}/\text{m}^3$), ozone (ppm), concentration of nitrogen dioxide (ppm), concentration of carbon monoxide (ppm), sulfurous acid gas (SO₂) (ppm)

The Open Data Plaza of Seoul provides data to the RESTful Web service based on HTTP, and the situation recognition model based Web module requests data through the HTTP protocol.

The Open Data Plaza of Seoul returns the relevant data for the requested service in either XML or JSON format, and the collection procedure of such air pollution data of Seoul is shown in the diagram below.

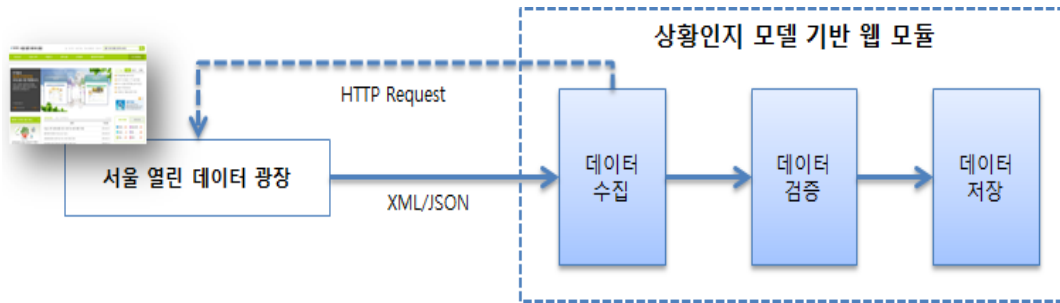


Figure 16-3 Data collection method for Seoul

Actual physical sensors are installed, and the sensing data are transmitted to the web server. The USN-based data collection technique is applied. The physical sensors collect the primary data from the gateway through the sensor network. The collected data are transmitted to the web server in SenML format. Transmitted data are stored after verification. The data collection procedure of a physical sensor is shown in the diagram below.

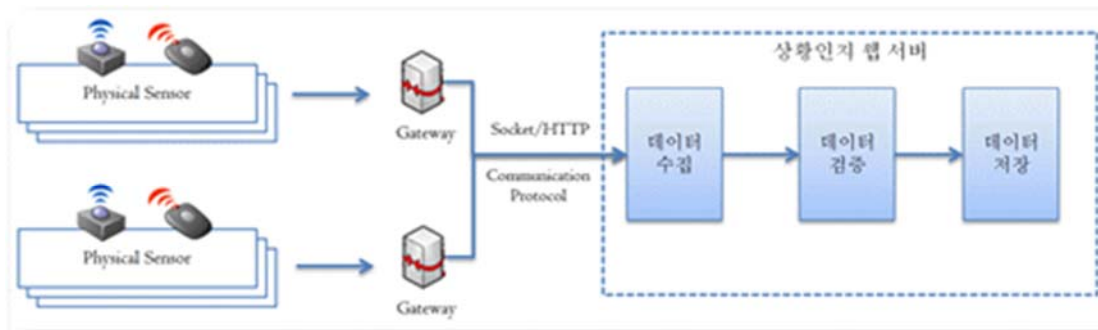


Figure 16-4 Physical Sensor Interface

SenML is applied, which is used to transmit the sensing data of a physical sensor to the web server. SenML is a data model made for simple types of meta-data or measured values from the measurement equipment or device, where the data consist of a single object, including the sequence, and each item of the sequence consists of elements that recognize the sensors. The format can be implemented using JSON, XML, and an Efficient XML Interchange (EXI), and in addition, can be implemented using a compressed binary format. The character set uses UTF-8 (allowing only the ASCII character set), and the transmission protocol is either TCT-based HTTP or UDP-based CoAP.

Table 16-1 Data representation of SenML data types

SenML	JSON	Type	Role
Base Name	bn	String	Name of tester or equipment that manages the sensors
Base Time	bt	Number	Reference time
Base Units	bu	Number	Reference unit
Version	ver	Number	Version
Measurement Parameters	e	Array	Sequence (at least more than once)
Name	n	String	Sensor or name of parameter
Units	u	String	Measurement unit
Value	v	Floating point	Number-type value expression
String Value	sv	String	Character-type value expression
Boolean Value	bv	Boolean	Yes- or no-type value expression
Value Sum	s	Floating point	Sum of measured value according to the time variation
Time	t	Number	Measured time for sensor storage
Update Time	ut	Number	Updated time

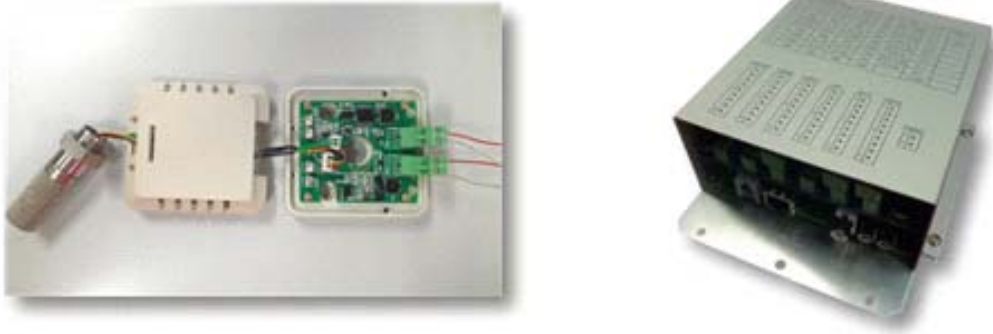


Figure 16-5 (Left) Sensor for temperature and humidity, and (right) gateway

The corresponding physical sensor transmits temperature and humidity data at one-minute intervals. The MAC address, measurement value, measurement unit, measurement time, latitude, and longitude of the gateway are transmitted. The physical sensors measure the temperature and humidity, and convert them into the current (0 to 20 mA) and transmit them to the ADC converter of the gateway.

The gateway converts the received current values into digital format and transmits them to the Situation Recognition Web server. The MAC address of the gateway is used for the Base Name during transmission, and the sensor identification is set to the MAC address_{sensor name}_{serial number}. The transmitted data are as below.

Transmission data (MAC, 0007c2307d2d; Timestamp, 1434318141, 26°C, 34.3%RH)

```
{ "e": [
  {"n": "0007c2307d2d_temp001", "v": 26, "u": "Cel"},
  {"n": "0007c2307d2d_humid001", "v": 34.3, "u": "%RH"},
  {"n": "latitude", "v": 0, "sv": "37.482887", "u": "lat"},
  {"n": "longitude", "v": 0, "sv": "126.896078", "u": "lon"}
],
"bn": "0007c2307d2d", "bt": 1434318141
}
```

The measured temperature and humidity values are confirmed and expressed as below in the Real-Time Sensor Information of the UPA Service Test Site.

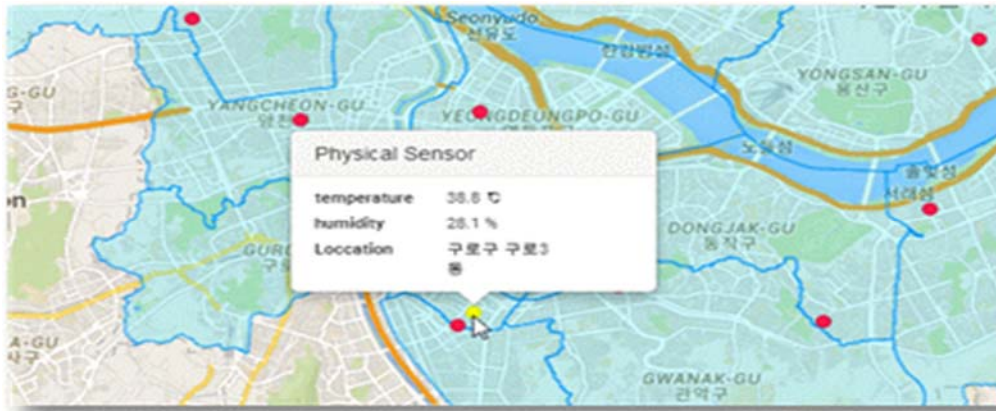


Figure 16-6 Data expression of physical sensor

The Table 16-2 summarizes the service name, request period, cron format, requested parameter type, and update cycle of the public data for 20 items related to the air environment of Seoul. The cron was set for separate requests at 3-s intervals because requesting and receiving requests at same time causes a server load. Each cron performs the request in 3 seconds of interval from 0 to 57 based on 15 minutes of period.

Table 16-2 Service names with cycle and parameters

Name of Service	Request Cycle	cron	Type of Parameter	Update Cycle
Daily average air pollution level	15 min	00/15 * * * *	yyyymmddHH24	Irregular (frequently)
Daily average air environment per period	15 min	03/15 * * * *	yyyymmdd	Irregular (frequently)
Roadside daily average air environment information per period	15 min	06/15 * * * *	yyyymmdd	Irregular (frequently)
Daily weather observation information	15 min	09/15 * * * *	yyyymmdd	Irregular (frequently)
Fine dust forecast/alert for Seoul	15 min	12/15 * * * *	X	Regular (1 hour)
Ozone forecast/alert per area for Seoul	15 min	15/15 * * * *	X	Regular (1 hour)
Ultra-fine dust forecast/alert for Seoul	15 min	18/15 * * * *	X	1 hour
Real-time average air environment per local district	15 min	21/15 * * * *	X	Regular (1 hour)
Real-time average air environment of Seoul	15 min	24/15 * * * *	X	Regular (1 hour)



Name of Service	Request Cycle	cron	Type of Parameter	Update Cycle
Monthly average air pollution level	15 min	27/15 * * * * *	yyyymm	Irregular (frequently)
Real-time average air environment per area	15 min	30/15 * * * * *	X	Irregular (frequently)
Roadside real-time air environment per measuring station	15 min	33/15 * * * * *	X	Irregular (frequently)
Real-time weather observation information per weather station	15 min	36/15 * * * * *	X	Irregular (frequently)
Air pollution level per time average	15 min	39/15 * * * * *	yyyymmddHH2400	Irregular (frequently)
Time average air environment information per period	15 min	42/15 * * * * *	yyyymmddHH2400	Irregular (frequently)
Yellow sand alert for Seoul	15 min	45/15 * * * * *	X	Regular (1 hour)
Yearly average air pollution level	15 min	48/15 * * * * *	YYYY	Irregular (frequently)
Status of ozone alert per year	15 min	51/15 * * * * *	X	Irregular (frequently)
Status of fine dust alert per year	15 min	54/15 * * * * *	X	Irregular (frequently)
Status of yellow dust alert per year	15 min	57/15 * * * * *	X	Irregular (frequently)

16.1.4 System Functions

The Korean example system, which uses an air pollution sensor network and a location-based context information model, aims to provide air-pollution warning services to citizens who are sensitive to the air environment, respiratory patients, and air-pollution-related agency employees. The air pollution sensor network monitors the air pollution substance status and its changes in real time, and saves them in a database to provide the basic statistics data for establishing air environment policies. The context information model is designed to provide air quality information depending on the user's location and areas of interest.

The proposed system uses the Korean CAI (Comprehensive Air Quality Index) to provide air quality information and action tips depending on the user's health status. The Korean CAI was developed to help its users easily understand the air pollution measurement values such as a fine dust (PM-10), ozone (O₃), a nitrogen (NO₂), a carbon monoxide (CO) and a sulfurous acid gas (SO₂). The CAI is classified into 'good,' 'moderate,' 'slightly bad,' 'bad,' and 'severely bad.' The 'good' phase indicates that the air quality does not negatively affect any people, and the 'moderate' phase is the level at which the air-

quality-related patient group can be slightly negatively affected. The phase ‘slightly bad’ is the level at which the health of the patient and sensitive (children, elderly, and infirm) groups can be compromised when they are exposed to the air environment. The phase ‘bad’ is the level at which not only the health of the patient and sensitive groups can be compromised, but normal people can also experience physical discomfort. The phase ‘severely bad’ is the level at which the patient and sensitive groups are seriously harmed by the air, and normal people can be slightly affected if they are exposed to the air environment. Furthermore, emergency treatments may be needed by the patient and sensitive groups.

The Figure 16-7 shows the GUI (Graphic User Interface) of the example system, in which the small circle in each district shows the air information as a value depending on the choice shown in the information window. In addition, the colour varies at four levels based on the value. Once the mouse is placed on the circle in the district, all of the air information is displayed. The selections provided on the menu are as follows:

- Real-time Air Quality Info
- Alert Status
- Citizen Reaction Status
- Statistics
- Annual Average Air Pollution
- Pollution trends
- Annual Statistical Alert
- Citizen Reaction Statistics

If users select air quality information from the map on the right, real-time information is displayed on the map for each district in Seoul. Here, different colours are displayed according to pollution range. Furthermore, if the mouse is placed on the symbol marked on the map, comprehensive information regarding air quality in the relevant district appears in a pop-up window. At the bottom left of the map, a gauge shows the feelings of Seoul citizens so that citizen reaction according to pollution level can be verified.

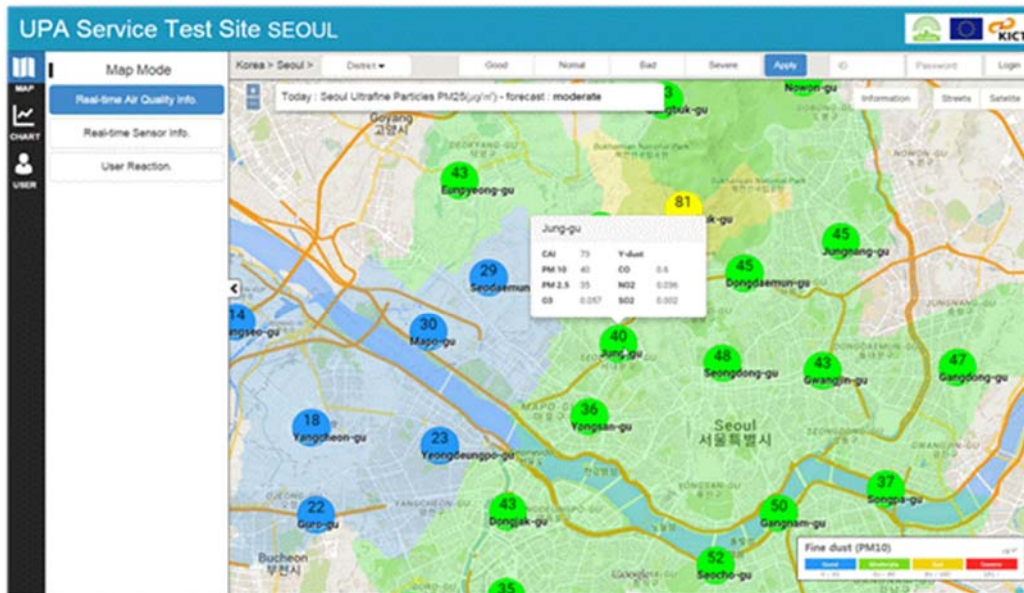


Figure 16-7 GUI of the pilot system under development

Table 16-3 Services from the Pilot System

Service Name	Descriptions
Air Quality Info.	<ul style="list-style-type: none"> To provide air quality information for each district along with citizens' reactions To provide air quality information including pollutant information and air pollution forecast
Real Time Air Pollution Info.	<ul style="list-style-type: none"> To provide real-time air quality information including PM, CO, CO₂, O₃, and SO₂. To provide action tips depending on the air-pollution level
Air Pollution Prediction Info.	<ul style="list-style-type: none"> To provide an one day forecast of air pollution To provide action tips depending on the air-pollution level
Citizens' Reaction Info	<ul style="list-style-type: none"> To provide citizens' reactions collected from citizens' opinions and SNS
Air Pollution Statistics	<ul style="list-style-type: none"> To provide air pollution statistics based on open data To provide a series of graphs including an air pollution value and its trend and air pollution warning frequencies To provide statistics of citizens reactions to air pollution
Citizens' Opinion	<ul style="list-style-type: none"> To allow citizens to input their opinions about air pollution Citizens can input their opinions with a text and an emotional motion icon
Others	<ul style="list-style-type: none"> To recommend a list of best districts for house moving To provide a list of hospital nearby a user's location

Table 16-4 Functions in Seoul Example Air quality Information System

Category	Function Name	Description
Map	Base map	<ul style="list-style-type: none"> Google layers is used as a base map to display a street map and a satellite image
	Seoul district map	<ul style="list-style-type: none"> Seoul's admin. map is used to show its districts' boundary
	Map manipulation	<ul style="list-style-type: none"> Users can use functions of zoom In/out and move
	Air quality info.	<ul style="list-style-type: none"> Air quality Information is shown with a legend
Air Pollution Info.	selection	<ul style="list-style-type: none"> Users can select an air pollutant from a list
	Real-time air pollution	<ul style="list-style-type: none"> A real-time air pollutant info is shown over a district map
	CAI (pop-up)	<ul style="list-style-type: none"> A new window is popped up to show comprehensive information for a district
	User location	<ul style="list-style-type: none"> User's Locations is shown over a map
Physical Sensor	Location	<ul style="list-style-type: none"> Locations of physical sensor are shown over a map
	Information	<ul style="list-style-type: none"> Real-time air quality information from a physical sensor is shown over a map

Citizens' Reaction	Search	<ul style="list-style-type: none"> Citizens' reactions to air pollution is shown on a display
Forecast & Alert	search	<ul style="list-style-type: none"> Information of real-time air pollution forecast/alert is shown on the bottom of a display
	Action tips	<ul style="list-style-type: none"> Action tips depending on air pollution level is shown with a pop-up window
Statistics	Average of CAI	<ul style="list-style-type: none"> Average of CAI is shown for each district
	Average of air pollutant	<ul style="list-style-type: none"> Average of each air pollutant is shown with a table and a graph
	Trend	<ul style="list-style-type: none"> Trend of each air pollutant is shown with a graph
	Forecast/alert	<ul style="list-style-type: none"> Frequencies of air pollution alert is shown with a graph and a table
	Citizens' reaction	<ul style="list-style-type: none"> Statistics of citizens' reaction is shown with a graph

16.2 Examples of Air Pollution Warning Service

The Seoul example system for air pollution warning service is implemented by integrating the environment data and social data, and this GIS-based system uses a map to display real-time air information, the forecast/alert status, citizen's reactions, and the Seoul CAI (Comprehensive Air-quality Index).

The main screen consists of a menu on the left and a map on the right. The menu on the left side of the screen can be used by the user to select the required information. The map on the right side then provides the selected information such as real-time forecast and alert information.

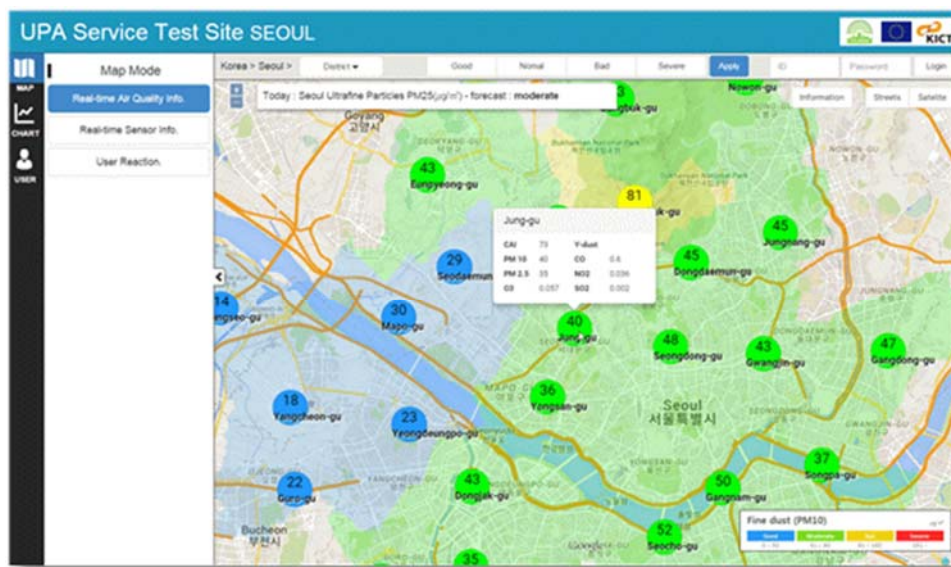


Figure 16-8 Start-up screen of UPA service

The air quality information consists of the Seoul CAI, PM, ozone, nitrogen dioxide, carbon monoxide, and sulfurous acid gas for each district. The user can obtain information by choosing one of the eight contaminated materials in the information window, from which the desired information can be selected.

The small circle in each district shows the air quality information as a value depending on the choice shown in the information window. In addition, the colour varies at four levels based on the value. Once the mouse is placed on the circle in the district, all of the air information is displayed (Figure 16-8).

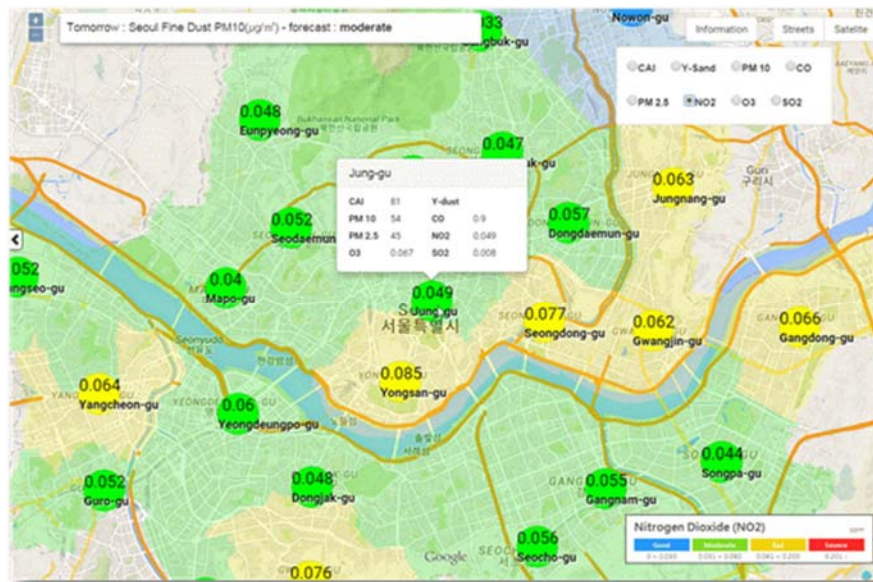


Figure 16-9 Display of real-time concentration of contaminated materials

Air quality information, which is available from this example system is closely related with the Open data Space of Seoul that provides the current status of the air environment forecasts/alerts. Also, the system is designed to provide users with the action tips based on the forecast/alert as a DB. The user can retrieve the action tips on the corresponding forecast/alert by clicking the Action Tips button in the forecast/alert status menu (Figure 16-10). It is expected that, in the future, such a service providing more valuable information to users with high sensitivity to their air environment, such as patients with a respiratory disease, can be used.

Forecast & Warning Status

Item : All Month : 06 Search

Item	Level	Date	Grade	Standard	Area	Action Tips
Fine Dust	forecast	2015-06-17 17:00	moderate	31 - 80	Seoul	Action Tips
Ozone	forecast	2015-06-17 17:00	moderate	0.031 - 0.090	Seoul	Action Tips
Ultrafine Particles	forecast	2015-06-17 17:00	moderate	16 - 50	Seoul	Action Tips
Ultrafine Particles	forecast	2015-06-17 11:00	moderate	16 - 50	Seoul	Action Tips
Fine Dust	forecast	2015-06-17 11:00	moderate	31 - 80	Seoul	Action Tips
Ozone	forecast	2015-06-17 11:00	bad	0.091 - 0.150	Seoul	Action Tips
Ultrafine Particles	forecast	2015-06-17 05:00	moderate	16 - 50	Seoul	Action Tips

< 1 2 10 20 26 >

Figure 16-10 Status of air pollution forecast/alert



Figure 16-11 Action tips for each air pollution material

The air environment statistics information service was implemented, and it displays various graphs and tables for each period and district for an analysis of the pollution level within Seoul. The air environment statistics information falls into three categories, i.e., the yearly average pollution level per district of Seoul, an expression of the pollution variation for each district and contaminated material, and the status of the yearly issuance of forecasts/alerts in Seoul.

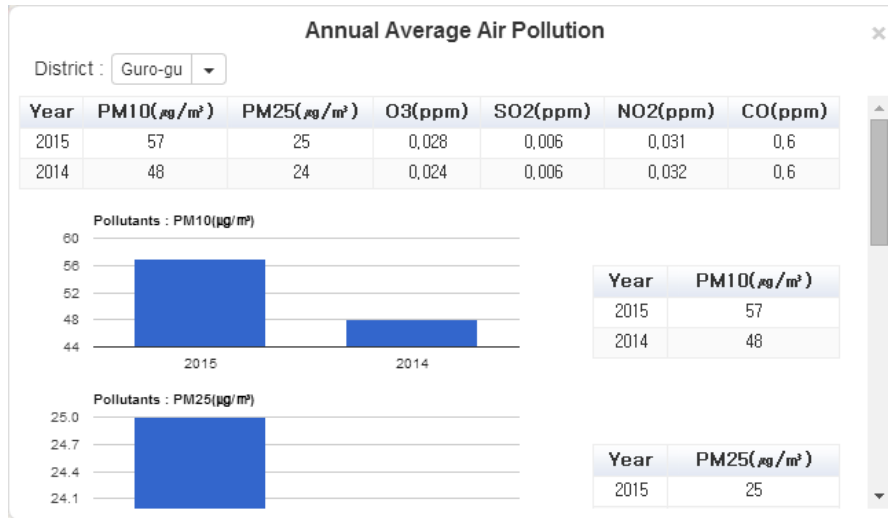


Figure 16-12 Yearly Average Air Pollution Level per District

The diagram below shows the yearly average of the contaminated materials in Guro-gu. Because the concentrations of fine dust, ultra-fine dust, ozone, sulfurous acid gas, nitrogen dioxide, and carbon monoxide are expressed through graphs and tables, the trend in their variation for the two most recent years and the amount of contaminated materials can be verified.

Such data are useful for a monthly or seasonal analysis of the characteristics because they show the long-term trends. In addition, the predictions can be made because meaningful yearly and quarterly patterns can be extracted from these data. The graph represents the pollution variation for a nine-month period. Once the user selects the pollution type and area, the results are expressed as a line graph, which can be useful for analyzing the pollution trend.

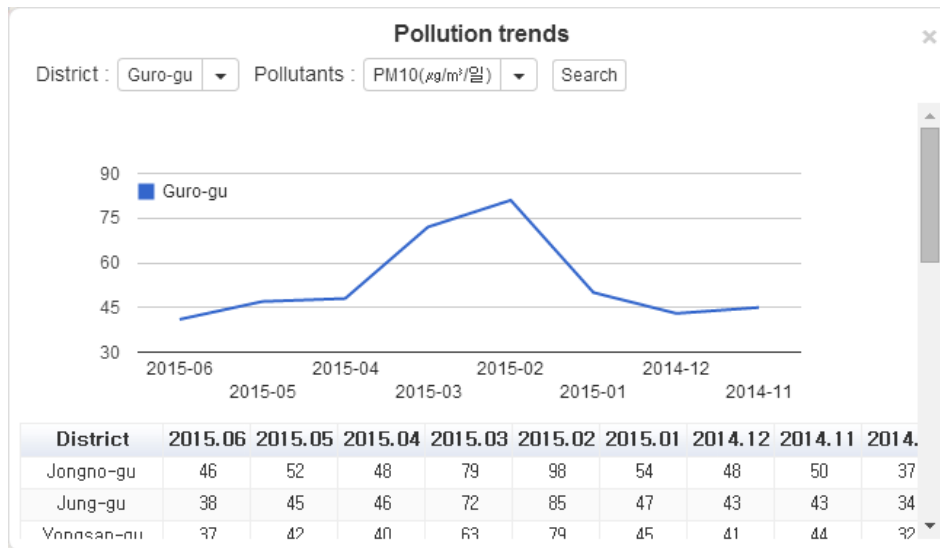


Figure 16-13 Variation trend of contaminated material concentration per district



A service for evaluating the status of yearly forecasts/alerts of Seoul was implemented. The forecast/alert data are represented through a bar graph and table allowing the pollution level to be analyzed by viewing the yearly forecast/alert status. Once the mouse is placed on a vertical bar, a tool tip that shows the data information appears; this tip is also available in other charts.

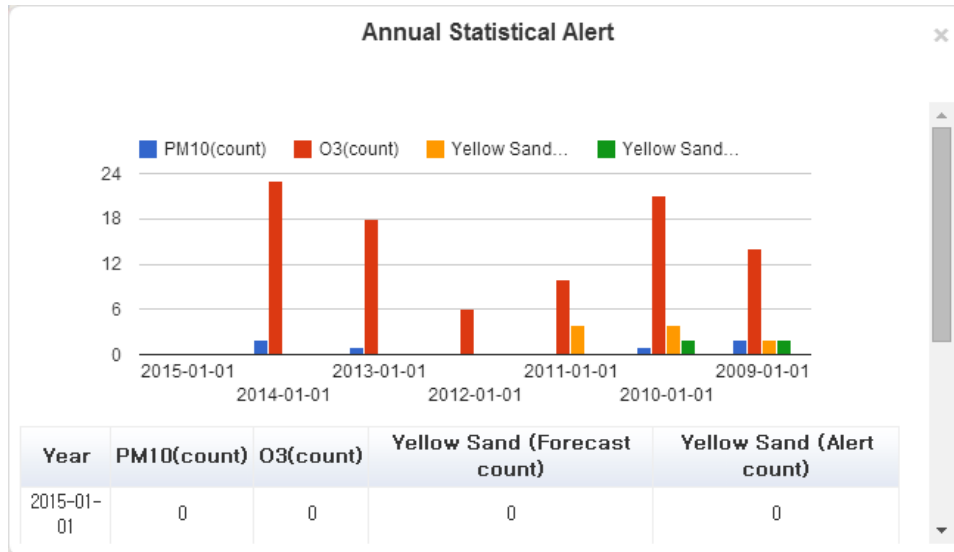


Figure 16-14 Status of forecasts/alerts per year

A service for showing the citizens’ reaction regarding the air environment was implemented. It suggests an expression method for collected SNS data related to the air environment through the cooperation with other related organizations. It also suggests interfacing with both the CITI-SENSE platform and the related organizations as a method to interface with SNS data (called a social sensor hereafter). If any data to be utilized in the CITI-SENSE platform exist, interfacing with the CITI-SENSE platform is an effective means of data management.

The social sensor expresses the awareness/emotions that citizens actually feel, which a physical sensor cannot provide. The data from the physical sensor express the environment as a value, and the social sensor expresses the awareness/emotions that cannot be expressed as a number through a comparison with the value from a physical sensor.

A social sensor measures fine dust, temperature, humidity, satisfaction/happiness, and dissatisfaction/depression on a weekly basis, and expresses the calculated index based on the frequency of tweet versus the regional population.

In addition, the reactions of users of a domestic POC site based on the pollution level can be analyzed, and the expressed data are the accumulated reactions of the citizens from the previous day.

The contaminated data are also expressed as the average data for the previous day. Once a mouse is placed on the desired admin area, the average pollution and accumulated reaction of the previous day can be verified. The most common reaction of the citizens is reflected at the right of the number through an emoticon.

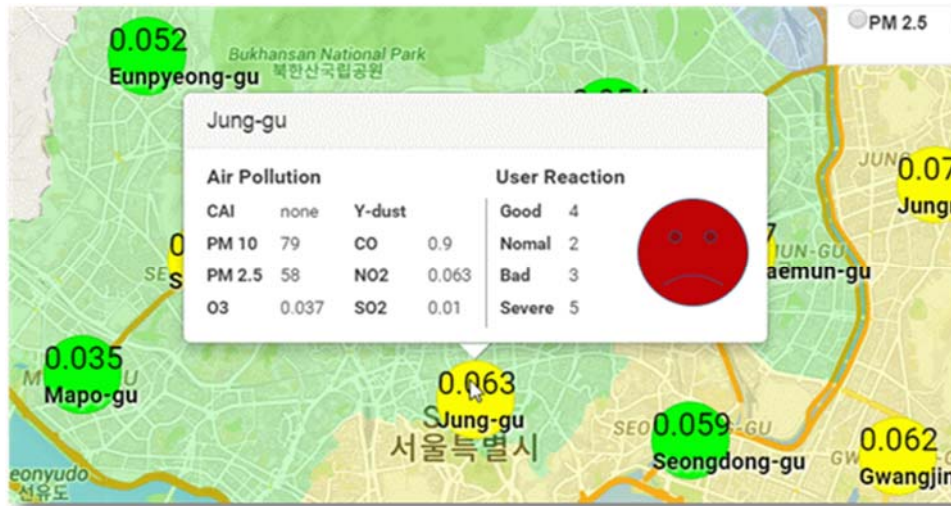


Figure 16-15 Status of user reactions

Various types of air-pollution related information is provided using the air environment integrated information service. The next scenario proceeds based on the assumption that a person with asthma is in a location where the SO₂ level as increased to worse than Bad and is considering moving to a new area.

First, the user with asthma logs in. A warning message appears if the No. 2 value increases to worse than Bad. The user then clicks the sentence “Please Click Here to find Hospital” in the corresponding forecast/alert message.

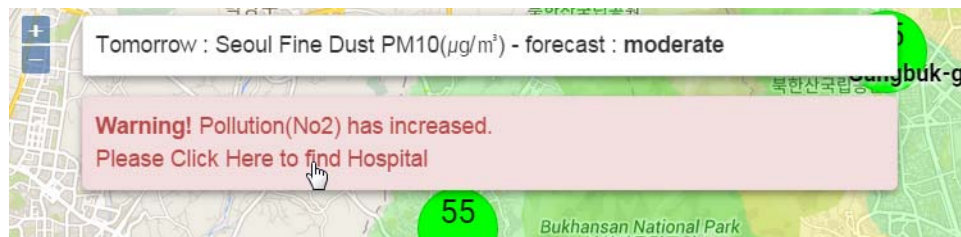


Figure 16-16 Warning of increase in pollution material value

When the current position is selected on the map, all hospitals and health centers within a 2-km radius are displayed. Clicking on the desired hospital shows how the hospital can be reached.

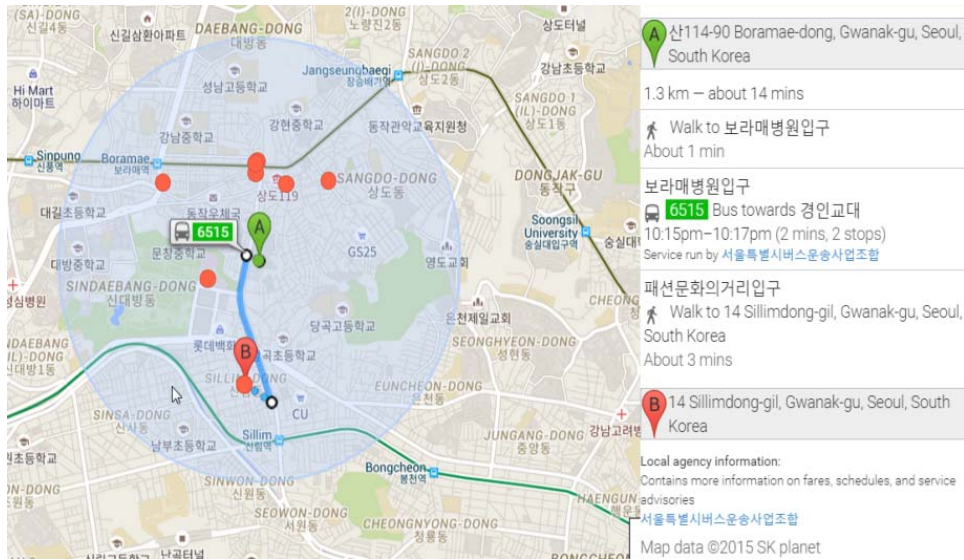


Figure 16-17 Hospital location and path guide

Next, consider a scenario in which a person with asthma is suggested to relocate to another area. First, the person with asthma logs in. The air quality information system displays the average three-year data on the pollution materials related to asthma (e.g., SO₂ and NO₂) through graphs and tables, and recommends three districts in order of the lowest amount of pollution materials present. The colors of the recommended areas vary on the map.

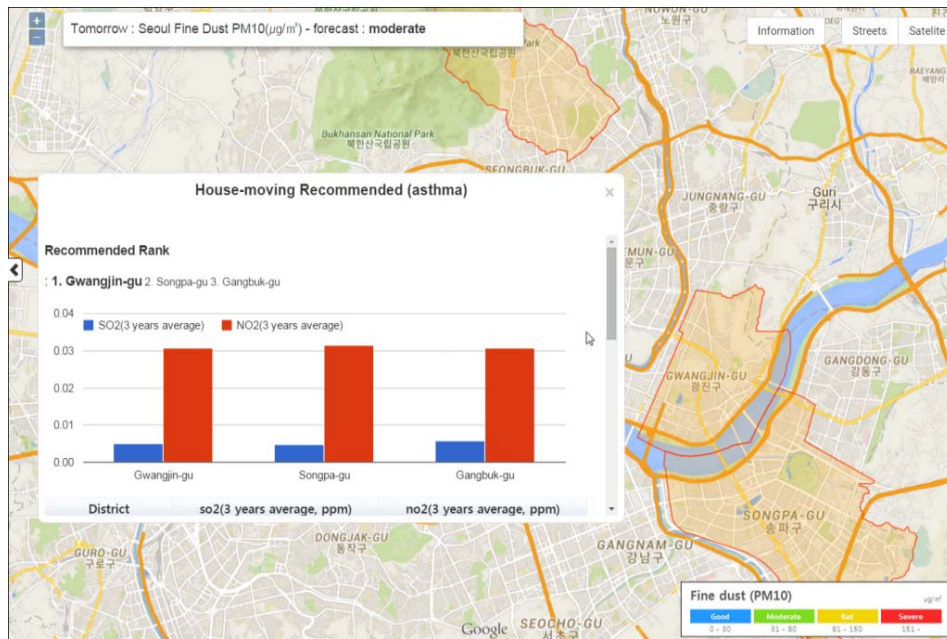


Figure 16-18 Recommendation of best district for home movement