



LOW-COST INNOVATIVE TECHNOLOGY FOR WATER QUALITY MONITORING
AND WATER RESOURCES MANAGEMENT FOR URBAN AND RURAL WATER SYSTEMS IN INDIA

Deliverable D5.5

Conceptual Design and Architecture of the Platform, revised version



Lead: UNEXE

Date: 28-02-2021

Public



LOTUS is co-funded by the European Commission under the Horizon 2020 research and innovation programme under Grant Agreement N° 820881 and by the Indian Government, Ministry of Science and Technology.



Project Deliverable

Project Number 820881	Project Acronym LOTUS	Project Title Low-cost innovative Technology for water quality monitoring and water resources management for Urban and rural water Systems in India
Instrument: Research and Innovation action		Thematic Priority EU-India water co-operation
Title D5.1 Conceptual design and architecture of the platform, first version		
Contractual Delivery Date February 2021 (M24)		Actual Delivery Date February 2021 (M24)
Start Date of the project February 1 st , 2019		Duration 48 months
Organisation name of lead contractor for this deliverable UNEXE		Document version V 1.0
Dissemination level Public X Confidential		Deliverable Type Document, Report X Demonstrator
Authors (organisations) UNEXE		
Reviewers (organisations) EGM		
Abstract This document captures the first version of the global system architecture, outlining all the components, their inputs and outputs and how they integrate with each other. It also identifies areas where existing technology will make up part of the solution and provides the foundations for designing a highly flexible, modular and		

expandable platform that will be suitable for the conditions in India and take advantage of the specific capabilities of the LOTUS sensor.

Keywords

Project architecture, Conceptual design, Platform, use cases

Disclaimer

This document is provided with no warranties whatsoever, including any warranty of merchantability, non-infringement, fitness for any particular purpose, or any other warranty with respect to any information, result, proposal, specification or sample contained or referred to herein. Any liability, including liability for infringement of any proprietary rights, regarding the use of this document or any information contained herein is disclaimed. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by or in connection with this document. This document is subject to change without notice.

LOTUS has been financed with support from the European Commission and the Indian Government, Ministry of Science and Technology.

This document reflects only the view of the author(s) and the European Commission and the Indian Government cannot be held responsible for any use which may be made of the information contained herein.



The LOTUS Project

LOTUS is a project funded by DG Environment under the European Union Horizon 2020 Research and Innovation Programme and by the Indian Government. It brings together EU and prominent Indian organisations with the aim to co-create, co-design and co-develop innovative, robust, affordable low-cost sensing solutions for enhancing India's water and sanitation challenges in both rural and urban area.

The LOTUS solution is based on an innovative sensor and includes tailor-made decision support to exploit the capabilities of the sensor as well as a specific approach to co-creation. LOTUS aims to be co-designed and co-produced in India, and have a wide, diverse and lasting impact for the water sector in India due to intense collaborations with commercial and academic partners in India.

Based on the low-cost sensor platform, solutions for the early detection of water quality problems, decision support for countermeasures and optimal management of drinking and irrigation water systems, tailored on the functionalities of the new sensor, will be developed and integrated with the existing monitoring and control systems.

This sensor will be deployed in five different use cases: in a water-network, on ground-water, in irrigation, in an algae-based wastewater treatment plant and water tankers. The packaging of the sensor, as well as the online and offline software tools, will be tailored for each of the use cases. These last will enable us to test the sensors and improve them iteratively.

The project is based on co-creation, co-design and co-production between the different partners. Therefore, an important stakeholder engagement process will be implemented during the project lifetime and involve relevant stakeholders, including local authorities, water users and social communities, and will consider possible gender differences in the use and need of water. Broad outreach activities will take place both in India and in Europe, therefore contributing to LOTUS impact maximization.

The further development and exploitation (beyond the project) of the novel sensor platform will be done in cooperation with the Indian partners. This will create a level playing field for European and Indian industries and SMEs working in the water quality area.

Table of Contents

1	Executive Summary	10
2	Introduction	11
3	Requirements and dependencies	12
3.1	Sensor functionalities.....	12
3.2	Strategic analytical tools and real time functionalities.....	12
3.3	End-user-specific requirements.....	13
3.4	Case study requirements	15
3.4.1	Use Case 1: Water distribution network Guwahati.....	15
3.4.2	Use Case 2: Tanker-based water distribution network	15
3.4.3	Use Case 3: Irrigation water distribution network	15
3.4.4	Use Case 4: Groundwater and river water monitoring	15
3.4.5	Use Case 5: Wastewater treatment	16
3.5	Other considerations	16
3.6	Summary of requirements	16
4	Conceptual design and architecture.....	19
4.1	State of the art	19
4.1.1	IoT architecture	19
4.1.2	Open Network Architecture	19
4.1.3	INSPIRE (2007/2/EC) directive for data interoperability	19
4.1.4	Application technology stacks	19
4.1.5	Personal Data Protection.....	20
4.1.6	Cybersecurity.....	20
4.2	Platform components	23
4.2.1	Conceptual platform.....	23
4.3	Platform requirements & assumptions.....	27
4.3.1	Data collection	27
4.3.2	Data management	27
4.3.3	Data processing	27
4.3.4	Data presentation (client & server).....	28
4.3.5	Platform considerations	28
4.4	Conceptual architecture	29

4.5	Applying the Conceptual Architecture to use case requirements	31
4.5.1	Data collection requirements	31
4.5.2	Data management, processing and use requirements	31
4.5.3	Dashboard requirements.....	32
4.6	Applying the Conceptual Architecture to the use cases.....	33
4.6.1	Use Case 1: Water Distribution network Guwahati	33
4.6.2	Use Case 2: Tanker-based water distribution network	33
4.6.3	Use Case 3: Irrigation water distribution network	33
4.6.4	Use Case 4: Groundwater and river water monitoring	33
4.6.5	Use Case 5: Wastewater treatment	34
5	Realising the conceptual design	35
5.1	FIWARE Context Information Broker	35
5.1.1	FIWARE as a platform	36
5.1.2	FIWARE as functionality.....	36
5.1.3	FIWARE data structures / formats.....	37
6	Platform demonstrator.....	39
6.1	Requirements.....	39
6.1.1	Goals	39
6.1.2	LOTUS Sensor data	39
6.1.3	Demonstrator functionality.....	39
6.1.4	User-centric Use Cases	40
6.2	Implementation	42
6.2.1	Server / Context broker	42
6.2.2	Administrator webapp.....	42
6.2.3	Customer mobile app	44
6.2.4	Control webapp	45
6.2.5	Anomaly detection	46
6.3	Results.....	47
6.3.1	Administrator webapp.....	47
6.3.2	Customer Mobile App.....	50
6.3.3	Control webapp	50
7	Conclusions and Next Steps.....	52
7.1	Conclusions	52

7.1.1	Proof of concept	52
7.1.2	Anomaly detection and localisation	52
7.2	Next steps	52
7.2.1	Integration with suitable FIWARE context brokers	52
7.2.2	LOTUS data collection.....	52
7.2.3	Anomaly detection integration	53
7.2.4	Security integration	53
7.2.5	SCADA integration	53
7.2.6	Enhanced mobile integration	53
8	References	54

List of Figures

Figure 1.	Prototypical MVC architecture	23
Figure 2.	Client-server architecture as MVC	24
Figure 3.	Conceptual architecture.....	24
Figure 4.	Conceptual architecture	30
Figure 5.	Conceptual architecture for groundwater and river water monitoring.....	34
Figure 6.	Typical IoT-inspired context broker architecture.....	35
Figure 7.	FIWARE device data structure showing multiple sensor data	37
Figure 8.	FIWARE device data structure showing single sensor data	38
Figure 1	- System use case collaborations.....	41
Figure 2	– Set Alert sequence (left), view alerts (right).....	43
Figure 3	- Geographic context, sequence diagram.....	43
Figure 4	– Graphing context, get graph params (left), build graph (right).....	44
Figure 5	- Mobile data sequence diagram.....	45
Figure 6	- Control context, get_control_data (left) , control_trigger (right).....	45
Figure 7	– Examples of anomaly detection (top) and localisation (bottom).....	46
Figure 8	– Set Alert Screen	47
Figure 9	- View Alert Screen	48
Figure 10	- Geographic context, map (left) and satellite (right).....	48
Figure 11	- Sensor status: working and properties in range (left), offline (centre) and working but at least one property out of range	48
Figure 12	- Sensor showing property details.....	49
Figure 13	- Graph view of sensor property	49

Figure 14 - Mobile app	50
Figure 15 - Control Panel.....	51

List of Tables

Table 1. Summary of desired system functionality and user interaction expressed by end users in the WP1 workshops.....	14
Table 2. General requirements relating to the conceptual design and architecture of the LOTUS platform.....	17
Table 3. Summary of data collection requirements for Use Case 4.....	17
Table 4. Summary of data management, processing and use requirements for Use Case 4	18
Table 5. Summary of administrator dashboard requirements for Use Case 4	18
Table 6. Table of case studies and relevant architectural components.....	26
Table 7. Data collection requirements mapped to conceptual architecture components.....	31
Table 8. Data management & processing requirements mapped to conceptual architecture components.....	31
Table 9. Administrator dashboard requirements mapped to conceptual architecture components ..	32

Acronyms and Definitions

Acronyms	Defined as
API	Application programming interface
CRUD	Create, read, update and delete
DSS	Decision support system
EC	Electrical conductivity
GPS	Global positioning system
IoT	Internet of things
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
LoRaWAN	Long range wide area network
LWM2M	Lightweight Machine-to-Machine
MVC	Model, view and controller
NGSIv2	Next Generation Service Interface v2
ONA	Open network architecture
OPC	Open platform communication
OPC-UA	Open Platform Communications Unified Architecture
OWASP	Open Web Application Security Project
PDPA	Personal Data Protection Act
SCADA	Supervisory control and data acquisition
SQL	Structured query language
TDS	Total dissolved solids
WDN	Water distribution network
WP	Work package
XSS	Cross-site scripting

XXE	XML External Entity
XML	Extensible Mark-up Language

1 Executive Summary

This document forms the revised version of the conceptual design and architecture for the LOTUS project, developing a conceptual architecture from use case requirements (Sections 3.3 & 3.4) whilst drawing on state-of-the-art architectural and operational practices (Section 4.1).

This iteration of the document concentrates on the design and delivery of the integrated platform demonstration (section 6). Given the low rate of uptake from the use cases and the limited availability of data, it was decided to develop functionality based on the Guwahati use case and covers the key use case collaborations for the demonstrator to implement.

2 Introduction

This document presents the revised version of the conceptual design and high-level architecture of the LOTUS platform, based on state-of-the-art internet of things (IoT) and open network architecture (ONA), the sensor functionalities (WP2), the specific user requirements (WP1), the strategic analytical tools (WP3) and the real time functionalities (WP4), combined with the specifics of the case studies (WP6).

For this iteration of the architecture, a prototypical use case has been developed that looks to demonstrate a broad range of platform functionality (data collection, data storage, data processing and visualisation through web and mobile apps) and is presented in Section 6.

In the architecture design, all components are outlined, along with their inputs and outputs and how they integrate with each other. It also identifies areas where existing technology will make up part of the solution and provides the foundations for designing a highly flexible, modular and expandable platform that are suitable for the conditions in India and take advantage of the specific capabilities of the LOTUS sensor.

The initial architecture proposed here will be updated later in the project as, and if, needed.

The key objective is to deliver an integrated framework to cope with the interoperability issues that arise in diverse environments and Use Cases (WP6, T6.1). The framework includes multiple modules communicating with each other through an application programming interface (API) developed in T5.1 and protocols developed in T2.2. Interfaces are specified with clear adherence to exchangeability and compliance with existing standards.

The conceptual design and high-level architecture of the LOTUS platform also took into account the outcomes from the co-creation workshops organised in WP1 (T1.2) in order to facilitate an end-user and pilot-driven approach. The integrated system has been developed, therefore, in strong collaboration with the demonstration use cases and the individual participants and actors.

3 Requirements and dependencies

WP5 provides a platform for integrated (real-time and offline) operational management of water systems. Broadly speaking, this is required to provide processing, real-time display and monitoring of system information, and an interface to strategic analytical tools developed for each use case. Specific requirements and dependencies that must be considered in the design and architecture of the platform are discussed in this section.

3.1 Sensor functionalities

The design of the platform needs to be tailored to the functionalities and capabilities of the LOTUS sensor. The LOTUS sensors are tuneable to various water use cases, but in all instances provide measurements of multiple parameters for chemical monitoring of water. Sensing capabilities initially cover temperature, conductivity, pH and chlorine, and later pesticides and heavy metals. Bacterial contamination is also intended to be measured via indirect detection methods. Additionally, operational information on the status of each sensor (relating to, for example, communication, energy and global positioning system (GPS) location) are available. The conceptual design and architecture of the platform must, therefore, facilitate the processing, display and monitoring of multiple measurements from each sensor and provide information on their operational status. Management of identifier information (IDs) for the sensors must also be considered. While the precise nature of the data processing, display and monitoring varies by use case, the processing is expected to transform sensor (and other) data quantitatively and qualitatively into formats that meet the needs of the respective stakeholders. Specific requirements related to the handling of data from the LOTUS sensor in each use case are further detailed in Sections 3.3 and 3.4.

3.2 Strategic analytical tools and real time functionalities

The platform needs to implement the strategic analytical tools developed in WP3, as required for each case study. These tools provide the following functionalities;

- Creating historical time series data and outlining specific operational targets and chemical species to be monitored (T3.1)
- System modelling and simulation, including water quality demand, leakage and pressure management (T3.2). This includes modules for:
 - Optimizing intermittent operation of water distribution network (WDN) and transition towards 24x7 service delivery (T3.2.1)
 - Optimizing demand and asset management techniques and practices for effective and efficient operations and management of water distribution systems (T3.2.2)
 - Modelling and optimization of tanker-based water distribution system (T3.2.3)
 - Groundwater and river water system modelling and simulation (T3.2.4)

- Adaptation to the use cases, deployment and evaluation (T3.3)

Note, not all case studies have implemented all functionalities. Some use cases have even adopted alternative solutions that the conceptual architecture presented in this document is not applicable (as discussed in Section 3.4).

The platform also implements the real time functionalities developed in WP4 for Use Case 4. These include real time software tools with geo tagging for water system management (T4.2) and are adapted to the use cases, deployment and evaluation (T4.3).

Specific requirements from each use case relating to the implementation of strategic analytical tools and real time functionalities in the conceptual design and architecture of the platform are covered in Sections 3.3 and 3.4.

3.3 End-user-specific requirements

The specific requirements from the stakeholders in India should be addressed in the design of the platform. This includes not only the system operators or administrators, whose requirements are captured in Section 3.4, but also the end users of the platform (e.g. members of the general population). This section specifically addresses the requirements of the end users.

Under WP1, co-creation workshops involving the final (end) users of the LOTUS use cases and addressing user interaction and system functionality were held (as reported in D1.2). The development of the platform conceptual design and architecture have adopted an end-user and pilot driven approach that takes into account the outcomes from these workshops .

The workshops for Use Case 1 (WDN in Guwahati), Use Case 2 (tanker based WDN) and Use Case 4 (groundwater and river monitoring) involved members of the general population as final users. For Use Case 3 (irrigation WDN), farmers participated in the workshop, and for Use Case 5 (wastewater treatment), no workshop was held.

Commonly requested features included SMS notifications (Use Cases 1-5), information on water quality status (Use Cases 1, 2 and 4) and advice for safe water use (Use Cases 1 and 4). Specific parameters of interest were also identified by the users, along with requirements relating to monitoring frequency, data management and system maintenance.

An overview of all system functionalities and user interactions that the participants of the workshops expressed an interest in (as reported in D1.2) is given in Table 1. The section in D1.2 in which each is originally specified is indicated in brackets.

Table 1. Summary of desired system functionality and user interaction expressed by end users in the WP1 workshops

Use Case	System functionality	User interaction
1	<ul style="list-style-type: none"> Turbidity and bacterial contamination are 'must have' parameters (D1.2, Section 2.7) The sensor should be used to increase trust in piped water supply (D1.2, Section 2.6.2.3) 	<ul style="list-style-type: none"> SMS alert when groundwater does not meet safety standards (D1.2, Section 2.6.2.3) App with information on water quality status (e.g. using a colour coded system from 'good' to 'bad') and instructions for safe water use (e.g. a statement such as 'safe to drink after boiling/filtering/treatment') (D1.2, Section 2.6.2.3) Public display of groundwater quality (D1.2, Section 2.6.2.3)
2	<ul style="list-style-type: none"> Information on water quality, the quantity delivered by tankers and the time of filling the source should be provided (D1.2, Section 4.6.1.3) 	<ul style="list-style-type: none"> SMS notification providing information on water delivered (D1.2, Section 4.6.2.3)
3	<ul style="list-style-type: none"> pH, and electrical conductivity (EC) are 'must have' parameters (D1.2, Section 5.6.1.4) Sodium is a highly desirable parameter (D1.2, Section 5.6.1.4) High temporal resolution of monitoring (D1.2, Section 5.6.1.4) Data should be transferred to a system for storage and analysis (D1.2, Section 5.6.1.4) 	<ul style="list-style-type: none"> An advisory system for fertigation (D1.2, Section 5.6.1.4) Two data displays (D1.2, Section 5.6.2.3) SMS alerts (D1.2, Section 5.6.2.3) The system must require little maintenance (specifically, less than once a year) (D1.2, Section 5.6.2.3)
4	<ul style="list-style-type: none"> Fluoride, arsenic and iron are 'must have' parameters (D1.2, Section 3.7) Sensor should be used to encourage people to sign up to the government water supply (D1.2, Section 3.6.3) 	<ul style="list-style-type: none"> App that enables users to access water quality status and receive advice for safe water use (D1.2, Section 3.6.2.3) SMS alerts when groundwater does not meet safety standards (D1.2, Section 3.6.2.3) Public display (D1.2, Section 3.6.2.3)

3.4 Case study requirements

The platform design was meant to consider the specific requirements of individual case studies (WP6) such that its functionalities are adjusted for the tools needed in each Use Case (WP3 and WP4), where the requirements are detailed in the following sections, 3.4.1 to 3.4.5. Where applicable, this includes a summary of pertinent information from D6.1 [1], which includes information on the dashboards and interfaces, software / hardware component connectivity and software components required for each use case.

3.4.1 Use Case 1: Water distribution network Guwahati

Use Case 1 will be delivered by ABB such that it is not considered in the conceptual architecture, which requires the following run-time functionality:

1. Water quantity (pressure, flow) monitoring and anomaly detection in urban water distribution systems
2. Water quality monitoring and anomaly detection
3. Real-time mitigation measures for water quantity and quality alerts
4. Real-time alerts to the public

These will be considered further in WP4.

3.4.2 Use Case 2: Tanker-based water distribution network

Use Case 2 will be delivered by Suyati such that it is not considered in the conceptual architecture.

3.4.3 Use Case 3: Irrigation water distribution network

Use Case 3 will be delivered by JAIN / TU-Dortmund such that it is not considered in the conceptual architecture.

3.4.4 Use Case 4: Groundwater and river water monitoring

Use Case 4 will monitor groundwater quality in the cities of Guwahati, and Bangalore, and both groundwater quality and river water quality for the city of Varanasi.

Dashboard requirements identified for Use Case 4 in D6.1 are as follows:

- 1) An administrator dashboard (D6.1, Section 5.6) that includes:
 - Display of the concentration of different substances (including EC, total dissolved solids (TDS), temperature, pH, total hardness, chlorine, iron, arsenic, fluoride, nitrate and microbiological indicators) in the groundwater and the observed groundwater table (for Guwahati and Bangalore) (D6.1, Sections 5.4.1 and 5.6).
 - Display of different river water quality parameters (for Varanasi).

- A DSS displaying possible mitigation strategies for Guwahati, removal technologies for Bangalore, and removal methods or treatments for Varanasi (D6.1, Section 5.6).
- 2) The collected data to be represented with the graphical display (heat mapping type) with reference to the Indian water quality standards IS10500:2012
 - 3) Seasonal variations interlinking the external environmental parameters and its influence on ground water quality

Additional features specified in D6.1 include:

- 1) A numerical transient state contaminant model for Guwahati to identify an arsenic-safe aquifer or region for groundwater usage (D6.1, Section 5.1.1).
- 2) The river water quality monitoring in Varanasi to indicate the extent of water pollution and the presence of various pollutants to enable monitoring of the effectiveness of pollution control measures and assessment of the suitability of water for different uses (D6.1, Section 5.1.3).
- 3) Data to be collected from LOTUS sensors, both on-site and in lab, with an on-site storage capacity of at least 1000 entries and associated GPS data. Data are stored in the cloud via the LOTUS box (D6.1, Sections 5.4.3 and 5.5).
- 4) GPS location data must be integrated to capture the location of the sampling site for strategic planning and preparation of contaminant map (D6.1, Section 5.4.3).

3.4.5 Use Case 5: Wastewater treatment

Use Case 5 will be delivered by TU-Dortmund such that it is not considered in the conceptual architecture.

3.5 Other considerations

The platform must have the capacity to be exploited as a product, alongside the sensor, following the INSPIRE (2007/2/EC) directive for data interoperability [2]. It should also be modular, adaptable (to the specific requirements for each Use Case), expandable and interoperable with other existing systems at the utilities/users.

3.6 Summary of requirements

This section aims to provide a clear summary of the requirements identified in Sections 3.1-0 that will inform the conceptual design and architecture. Each requirement is given a unique ID so that it may later be referred to when checking that the conceptual design and architecture developed enables every requirement to be met.

Requirements identified in Sections 3.1, 3.2 and 3.3 can be considered general, as they are not specific to any individual use case. These requirements are summarised in Table 2.

Table 2. General requirements relating to the conceptual design and architecture of the LOTUS platform

ID	Description
RG1	Measurements of multiple parameters from each LOTUS sensor must be handled
RG2	Strategic analytical tools developed in WP3 for creating historical time series, outlining specific operational targets and monitoring chemical species must be implemented
RG3	Strategic analytical tools developed in WP3 for system modelling and simulation must be integrated.
RG4	The platform must integrate strategic analytical tools that are adapted to the specific use cases
RG5	The platform must integrate real time functionalities developed in WP4
RG6	The platform must have the capacity to be exploited as a product, following the INSPIRE (2007/2/EC) directive for data interoperability [2].
RG7	The platform should be modular, adaptable (to the specific requirements for each Use Case), expandable and interoperable with other existing systems at the utilities/users.

More detailed, tailored requirements can be identified for the individual use cases. As only Use Case 4 needs to be considered in the conceptual architecture (as discussed in Sections 3.4.1 to 3.4.5), the following requirements relate specifically to this use case. Firstly, the user requirements and case study requirements presented in Sections 3.3 and 3.4 identify a variety of sources from which the LOTUS platform needs to be able to collect data. Table 3, therefore, collates all the data sources that the conceptual design and architecture must consider for Use Case 4.

Table 3. Summary of data collection requirements for Use Case 4

ID	Description
RDC1	Collect data from LOTUS sensors (in bulk)

Secondly, multiple requirements related to the management, processing and use of this data can be identified from Sections 3.3 and 3.4. These include storage of data in meaningful, accessible and long-term repositories, processing of data pre/post storage, and use of data to create new insights (e.g. with analytical and decision support tools). Specific requirements applicable to Use Case 4 are summarised in Table 4.

Table 4. Summary of data management, processing and use requirements for Use Case 4

ID	Description
RDM1	A data logging system is required to store data collected from sources in Table 3
RDM2	Data collected needs to be processed for and used in advisory / decision support tools
RDM3	Data collected needs to be processed for and used in a contaminant transport model

Thirdly, the user requirements and case study requirements presented in Section 3.4 indicate that an administrator dashboard is needed for Use Case 4. Table 5, summarises the requirements for this.

Table 5. Summary of administrator dashboard requirements for Use Case 4

ID	Description
RD1	A dashboard for administrators is required
RD2	The observed groundwater table and concentration of different substances in the groundwater should be displayed
RD3	River water quality parameter concentrations should be displayed
RD4	A contaminant map should be provided, indicating the presence and extent of pollutants
RD5	A DSS displaying possible mitigation strategies and removal method(s) or treatment(s) should be included

4 Conceptual design and architecture

4.1 State of the art

State of the art areas of consideration include IoT architecture, ONA, the INSPIRE (2007/2/EC) directive for data interoperability [2], application technology stacks and cybersecurity. Further detail is given on each of these in the following sections, 4.1.1 to 4.1.5.

4.1.1 IoT architecture

The IoT architectural stack is considered to be comprised of the following components; sensing, network, data processing and application layers. This architectural stack allows fairly dumb sensing components to become part of complex applications, regardless of their location in the world relative to the application and user.

From Use Case 4, it can be seen that in some instances LOTUS sensors (sensing) are bundled with mobile phones (network) to create the lower core of an IoT stack. It is therefore expected that the platform architecture will provide the rest of the IoT stack (data processing & application layers) and will use a network layer to communicate with the LOTUS / mobile bundles.

4.1.2 Open Network Architecture (ONA)

Although ONA is normally considered as a feature of mobile telephony, it can be considered in more general terms as connectivity between networked systems. There is a role for ONA in Use Case 4 such that arbitrary (and undefined) components can be connected using open interfaces, typical based on networking protocols.

4.1.3 INSPIRE (2007/2/EC) directive for data interoperability

The INSPIRE (2007/2/EC) directive [2] is concerned with the use of open data formats for interoperability between users and systems. Given the need for Use Case 1 to use external run-time components (WP4), defining interfaces using a format to ensure interoperability is necessary.

Currently, FIWARE [3] is being extended to provide a data interoperability platform for water systems under the Fiware4Water project [4], and this should be strongly considered given the INSPIRE directive.

4.1.4 Application technology stacks

The conceptual architecture presented in this report is a client/server, which presents many technology choices.

4.1.4.1 Server-side

From the server-side, there is an assumption that the server will run Linux as this would appear to be an industry-standard and cost-effective approach, giving a wide choice of back-end development languages and frameworks, with technology interoperability allowing solutions to be developed using multiple languages and frameworks.

4.1.4.2 Client-side

Client-side development is largely concerned with the development of ‘dashboard’ applications that many of the case studies require, the assumption being that these would either be mobile native apps or web-based applications. This creates a wide range of technology stacks that could be employed.

For Use Case 4, suitable technology stack(s) can be chosen once there are clear requirements for run-time and visualisation / dashboard application in D4.1.

4.1.5 Personal Data Protection

The recent Personal Data Protection Act (PDPA) 2019 [5] enshrines personal data protection into Indian law and it should be noted that the act is largely geared around businesses collecting data rather than technology providers. It is also centred on data management for individuals and, given that case study 4 is centred on collecting, processing and storing river and ground water data, it should be exempt from the personal aspects of the PDPA, unless it becomes a requirement to store personal data for dashboard use, but this should be discouraged.

4.1.6 Cybersecurity

The Open Web Application Security Project (OWASP) Foundation maintain a ‘top 10’ of security risks for web applications [6]. These should be considered for a defensive architecture and are outlined below along with good practice that should be adopted for the LOTUS platform.

a. Injection

Injection flaws are concerned with injecting malware into commands sent from client to server, typically SQL queries. Within the LOTUS platform environment, this risk can occur whenever data is received by the server from a client, such as the platform dashboard clients and other services. This issue can be minimised by following a design pattern of not sending SQL, or any other plain text queries, to the server and sanitising inputs prior to executing plain text commands.

b. Broken Authentication

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users’ identities temporarily or permanently.

For Use Case 4, there is no consideration given to user authentication as there are no user or security requirements currently defined within the use case. This will be addressed in D5.2.

c. Sensitive Data Exposure

Many applications have been ‘hacked’, revealing sensitive data (user accounts and passwords, communication records, transactions and so on), due to application developers and operators under-estimating the importance of sensitive data and failing to secure it appropriately.

It is unlikely that Use Case 4 will contain sensitive data, but consideration will be given to protecting both short and long-term data.

d. XML External Entities (XXE)

XML External Entities relates to security issues with older XML parsers that can lead to unauthorised system access.

The LOTUS platform will use up-to-date XML parsers that do not suffer from XXE issues and will evaluate 3rd party software for Use Case 4 against this consideration.

e. Broken Access Control

Broken access control describes situations where users can access data that they shouldn’t due to issues with access control (rather than authentication issues as described previously).

f. Security Misconfiguration

Security misconfiguration is regarded as the most commonly seen security issue and describes multiple issues based around poor security, particularly running services with ‘default’ security, ignoring patch requests and avoiding system hardening.

g. Cross-Site Scripting (XSS)

XSS is a web-based hack that redirects a user’s browser to an untrusted site. For Use Case 4, this can be an issue for any clients that are realised through web services, rather than as bespoke apps.

h. Insecure Deserialization

Insecure deserialization describes transport level hacks where data is spoofed between client and server leading to many different issues. For the LOTUS platform, consideration will need to be given to robust client/server communication for the dashboard apps.

i. Using Components with Known Vulnerabilities

Many software components have, or have had, vulnerabilities that create attack vectors for hackers. Typically, as these weaknesses are discovered, they are fixed, or the components replaced for more secure components.

Within the LOTUS platform, care will need to be taken to keep the core platform up to date.

j. **Insufficient Logging & Monitoring**

The inability to log and monitor atypical behaviour is a core issue behind most cybersecurity issues, as they enable atypical behaviours to occur undetected, often for a long period of time. A logging component will be a core part of the LOTUS platform, all components should use it and operators should monitor its output.

4.2 Platform components

4.2.1 Conceptual platform

Section 3 details each use case and requirements for the LOTUS case study and is the source for the architectural activities within this report. Whilst the case studies are not identical there is a fair amount of functional overlap and commonality between them allowing for the creation of a re-usable core platform that can be customised by each use case to tailor required functionality.

Given the broad data collection, processing and presentation requirements of use case, it worth considering the model-view-controller (MVC) architecture for the conceptual platform, Figure 1. This architecture defines clear boundaries between data (as model), data processing (as controller) as user interaction (as view).

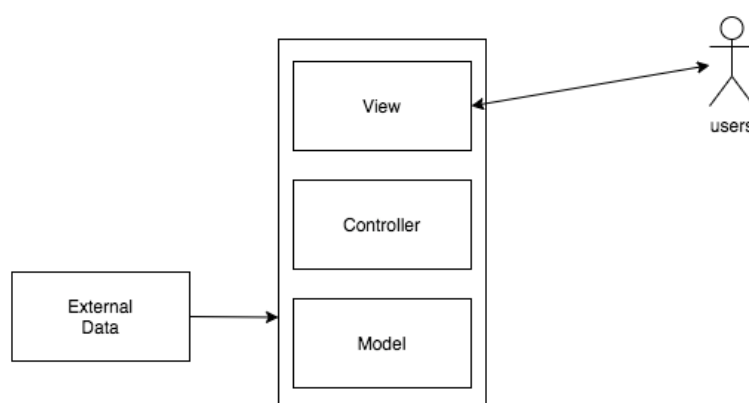


Figure 1. Prototypical MVC architecture

However, given the requirement in most case studies for concurrent user activity and, in some cases, different types of user to be able to interact with the systems developed, it makes sense to use a client/server architecture, with the assumption of multiple clients sharing data and functionality provided by a bespoke server. Figure 2 details the model-view-controller architecture implemented as client-server, with the server responsible for data storage (model) and data processing (controller) whilst the client, as dashboard(s), is responsible for presenting data (view) to the user and taking user input to be processed on the server.

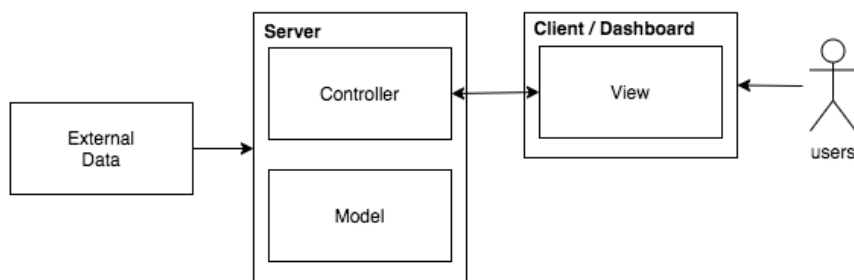


Figure 2. Client-server architecture as MVC

Figure 3 details the final iteration of conceptual platform with the introduction of IoT concepts, from Section 4.1 and the differentiation of external data types into LOTUS-sourced data (from the LOTUS sensor) and other data sources.

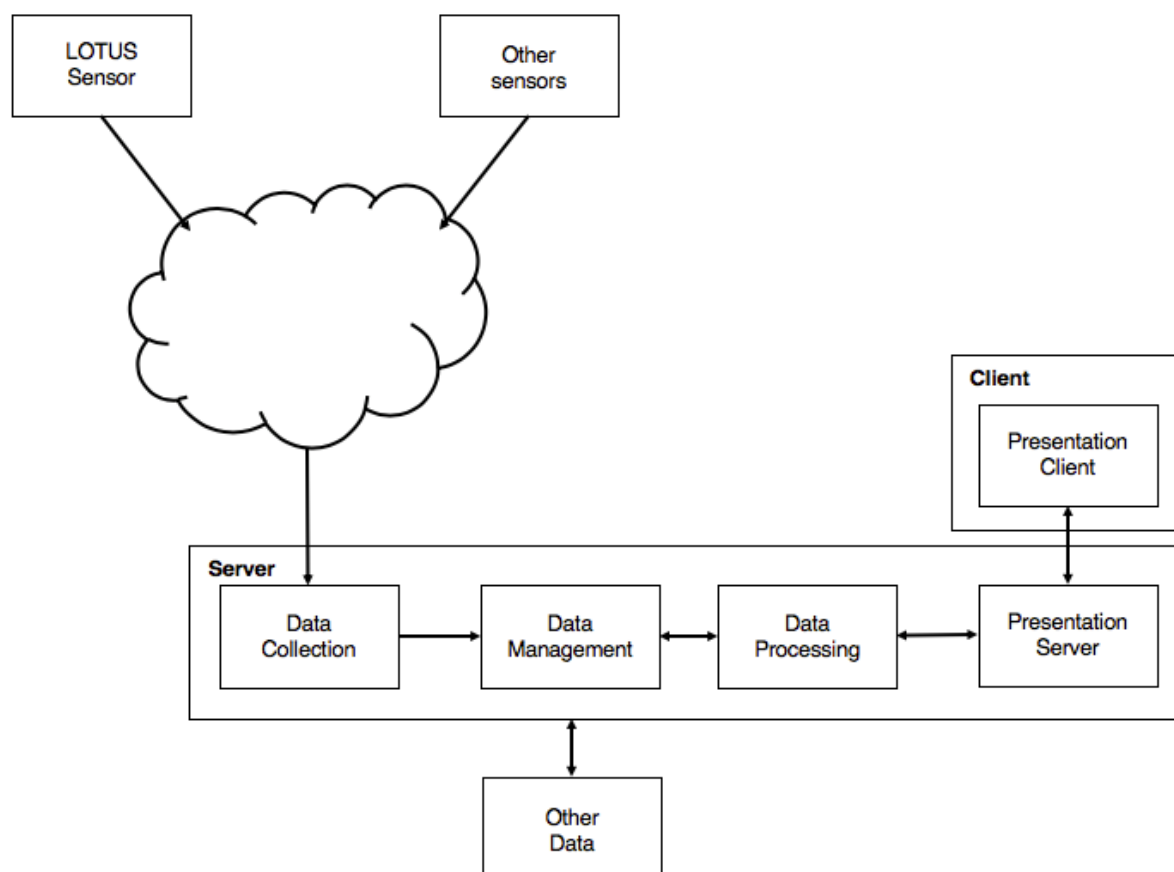


Figure 3. Conceptual architecture

Within these conceptual architecture components, there are the following roles and responsibilities:

- **LOTUS sensor**

These are devices as defined in Section 3.1 and are multi-function data collection devices. Their responsibility is to collect data which can be collected by the platform.

- **'Other' sensor**

These are any other kind of sensor that are not explicitly LOTUS sensors, typically GPS sensors. This data is collected alongside LOTUS sensor data and may be combined with LOTUS data in situations where it makes sense to.

- **Data collection**

The role of the data collection component is to collect sensor data and to prepare it for packaging into managed repositories. In Use Case 4, bulk LOTUS data will be presented for collection. It will be a responsibility of this component to log issues during the collection of data and to validate and verify incoming data where possible.

- **Data management**

The role of data management is the storage and manipulation of data. It is assumed that data will be managed in some form of repository in which data can be created, retrieved, updated and deleted within system policies.

Data management will also be responsible for the system-wide logging to record important incidents during platform operation.

- **Data processing**

This component is responsible for transforming data to provide some form of insight that can be shared with users via the presentation server and client components or sent to other systems via the other data component.

- **Presentation server & client**

These two components provide client / server communications such that client applications running on remote h/w can communicate with the server-based data and processing stack.

D6.1 Report on the detailed specification of the use cases, sensor requirements, and success criteria

Architectural Component	Use Case 1 Water Distribution Network Guwahati	Use Case 2 Tanker-based water distribution	Use Case 3 Irrigation water distribution network	Use Case 4 Groundwater and river water monitoring	Use Case 5 Wastewater treatment
Data collection	Data collected by SCADA	Data collected by tanker platform	Data collected by JAIN platform	Collect data from LOTUS boxes	Data collected by SCADA
Data management	Data managed by SCADA	Data managed by tanker platform	Data managed by JAIN platform	Manage sensor data Long-term data storage	Data managed by SCADA
Data processing	Water quantity monitoring and anomaly detection in urban water distribution systems Water quality monitoring and anomaly detection Real time mitigation measures for water quantity and quality alerts	Data processed by tanker platform	Data processed by JAIN platform	Decision support for mitigation, removal & concentration	Data processed by SCADA
Application Dashboard(s)	Real-time alerts to the public	Dashboard by tanker platform	Dashboard by JAIN platform	Dashboard for end user and CWC admins	Data visualised by SCADA

Table 6. Table of case studies and relevant architectural components

4.3 Platform requirements & assumptions

This section of the report examines the components of the platform in more detail in order to realise their requirements and assumptions.

4.3.1 Data collection

This component is responsible for the collection and validation of data from LOTUS and other sensors.

The assumptions being that:

1. The platform can request updates / data from sensors
2. Sensors can send data to the platform on demand
3. Sensors can send erroneous data which required validation
4. Sensors can be broken and send no data or data that is garbage
5. Sensors can send the same data multiple times
6. LOTUS sensors are not the only sensor input
7. Data from multiple sensors (LOTUS + GPS) may need to be combined for storage

4.3.2 Data management

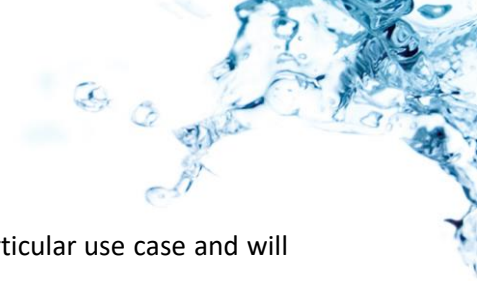
This component is responsible for the data lifecycle (create, read, update and delete, CRUD) within the platform. The assumptions being that:

1. The platform will realise its own storage solution
2. Sensor data collected through the data collection component will be stored for a relatively 'long' duration, potentially years
3. Other components within the platform will require data through searching
4. Data can be accessed from other data providers that are external to the platform and data can be added to platform-based repositories
5. Users of the platform can add additional data repositories
6. Data can be modified and/or deleted from the platform repository
7. Platform-wide logging will be managed through the data management component
8. EPANET models will be stored as part of data management

4.3.3 Data processing

This component is responsible for interpreting data from multiple sources; the data management component and external data sources and realises what is described as 'decision support' in several of the case studies.

The assumptions being that:



1. Data processing demands have the scope to be suitably unique to a particular use case and will be realised as bespoke components
2. Results from data processing may be stored in a platform repository.
3. Core use case functionality will be realised as interactions between data processing and data presentation.

4.3.4 Data presentation (client & server)

This component is responsible for realising functionality of the overall use case application and associated dashboard. The presentation component is realised in two parts: server and client.

The assumptions being that:

1. The presentation client will realise the dashboard component, or dashboard-like component of each use case.
2. The presentation client effectively forms the ‘view’ component of an MVC architecture and is responsible for taking user input and sending it to the server application and receiving server responses and presenting them back to the user.
3. The presentation server is effectively the ‘control’ component of an MVC architecture and will service user requests, leveraging the platform’s data management and processing components.
4. The overall architecture can support multiple clients to give different dashboard functionality.
5. Presentation client & server functionality is likely to be largely bespoke for each use case, given the diverse range of functionality required and the choice of client-side technology stacks available.
6. The client/server component will provide core communication functionality between client and server.

4.3.5 Platform considerations

1. The platform is extendable, with support for each use case to develop bespoke functionality across all components of the platform to meet their needs
2. The platform will collect LOTUS and associated sensor data
3. Data stored in the platform can be used by other systems
4. Data stored in other systems can be used (if accessible) by the platform
5. Dashboard applications can be developed using suitable hardware & technology stacks
6. FIWARE-compatible JSON-LD will be used as data format where possible
7. Cybersecurity will be considered as a core part of the architecture and the platform will be designed around current OWASP guidelines.



4.4 Conceptual architecture

The architecture is presented as Figure 4 and its refinement is effectively the combination of the architecture, presented in Figure 3, and the requirements and assumptions, presented in Section 4.3. The conceptual architecture enumerates a refinement of component-based functionality that details how component functionality can be realised.

The architecture realises key conceptual affordances:

1. Dashboard functionality is deliberately minimal to allow use cases to use technology stacks that meet their needs
2. Data collection is open-ended to support different methods of collecting data, from both sensors (LOTUS and others) and other data sources.

Section 4.6 models each use case with the conceptual architecture for functional fit and use case implementation.

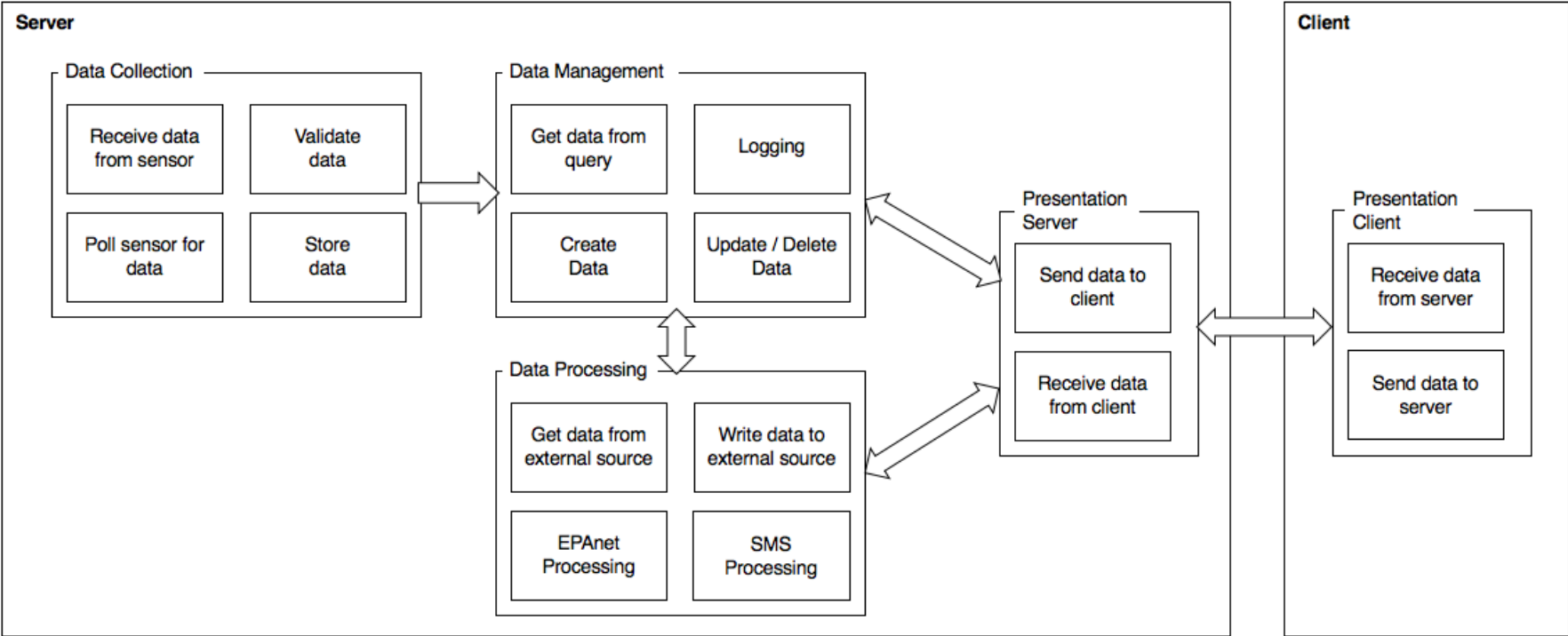


Figure 4. Conceptual architecture

4.5 Applying the Conceptual Architecture to use case requirements

This section of the report maps the conceptual architecture components and functions to the specific use case requirements identified in Section 3, to assess their fit. Table 7 addresses the data collection requirements (detailed in Table 3), Table 8 addresses the data management, processing and use requirements (detailed in Table 4), and Table 9 addresses the dashboard requirements (detailed in Table 5).

4.5.1 Data collection requirements

Table 7. Data collection requirements mapped to conceptual architecture components

ID	Description	Conceptual Architecture consideration
RDC1	Collect data from LOTUS sensors (in bulk)	LOTUS sensor data will be collected using the 'receive data from sensor' and/or 'poll sensor for data' functionality within the data collection component. It is assumed that sensors will either be polled for data or will automatically send data to the platform.

4.5.2 Data management, processing and use requirements

Table 8. Data management & processing requirements mapped to conceptual architecture components

ID	Description	Conceptual Architecture consideration
RDM1	A data logging system is required to store data collected from sources in Table 3	Data can be stored within the data management component, or externally through the 'write data to external source' functionality in the data processing component.
RDM2	Data collected needs to be processed for and used in advisory / decision support	Data (both internal and external) can be processed for decision support within the data processing component. Given the bespoke nature of each use cases advisory and support tools, it is expected that these components will be developed for each use case rather than solely relying on 'off-the-shelf' solutions, though there may be

	tools	scope for software re-use depending on use case needs.
RDM3	Data collected needs to be processed for and used in a contaminant transport model	Data can be collected from the data collection and data processing components and accessed through the data management component, or externally.

4.5.3 Dashboard requirements

Table 9. Administrator dashboard requirements mapped to conceptual architecture components

ID	Description	Conceptual Architecture consideration
RD1	A dashboard for administrators is required	Dashboard applications will be developed using server and client technologies that are appropriate for the use requirements presented.
RD2	The observed groundwater table and concentration of different substances in the groundwater should be displayed	
RD3	River water quality parameter concentrations should be displayed	
RD4	A contaminant map should be provided, indicating the presence and extent of pollutants	
RD5	A DSS displaying possible mitigation strategies and removal method(s) or treatment(s) should be included	

4.6 Applying the Conceptual Architecture to the use cases

This section of the report models each of the use cases in the conceptual architecture to investigate how the platform can be used and extended to best meet the use cases requirements.

4.6.1 Use Case 1: Water Distribution network Guwahati

This use case will be implemented largely outside of the LOTUS platform. It is expected that the ABB solution will integrate with leak detection functionality; this is covered in WP4.

4.6.2 Use Case 2: Tanker-based water distribution network

This use case will be implemented independently of the LOTUS platform

4.6.3 Use Case 3: Irrigation water distribution network

This use case will be implemented independently of the LOTUS platform

4.6.4 Use Case 4: Groundwater and river water monitoring

This use case is concerned with groundwater monitoring at two sites and river water monitoring. Although these site-based applications are currently considered as a single implementation, it is expected that they will be implemented as stand-alone solitons with some technology / functionality sharing. The conceptual architecture is presented as Figure 5.

- **Data Collection**
From the requirements presented, data will be collected from LOTUS sensors and then uploaded into the platform in bulk. This suggests different functionality from other user cases where data is collected one reading at a time or in bulk.
- **Data Management**
It is assumed that historic data will be stored within the platform for long-term comparison (T3.1) and water and GPS data combined to create geographic data. Data will also be archived to long-term storage where necessary.
- **Data Processing & Application Dashboard**
From the requirements for this use case, there are several instances of decision support required, geared around water quality mitigation and water removal, and these functional components will be shared between the three site applications (Guwahati, Bangalore & Varanasi).

SMS functionality has been requested as part of the client dashboard. With this approach, additional data management will need to maintain a repository of user phone nos. and provide SMS on demand. This may be easier to achieve through mobile notifications that can be broadcast to all users. However, this does rely on having the app running.

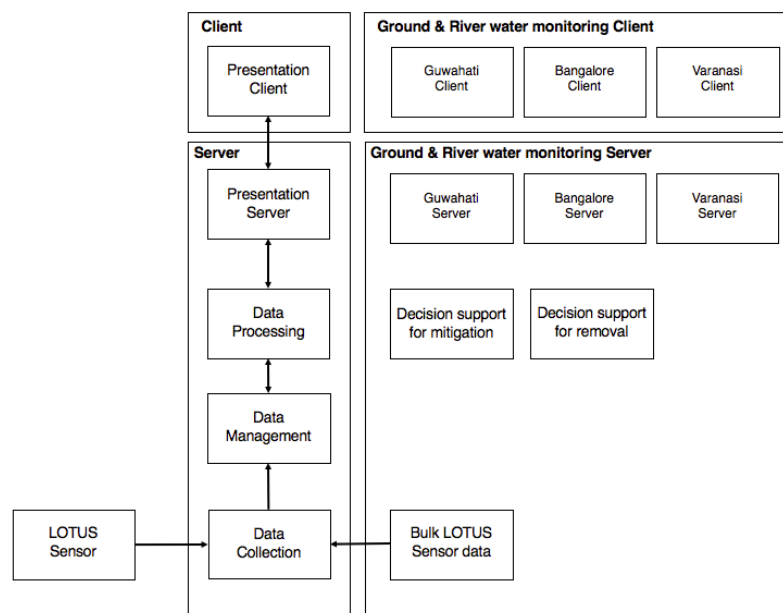


Figure 5. Conceptual architecture for groundwater and river water monitoring

4.6.5 Use Case 5: Wastewater treatment

This use case will be implemented independently of the LOTUS platform

5 Realising the conceptual design

This section of the report takes the conceptual architecture developed in Section 4.4 and applies the real-world components outlined in Section 4.1 to propose a physical design. At the core of this design is the philosophy of the ‘context broker’, Figure 6, or a centralised data hub, which relies on data being passed between components (typically component to broker and broker to component) using flexible and extensible data formats (JSON/JSON-LD) which reduces the design-time reliance on fixed data structures.

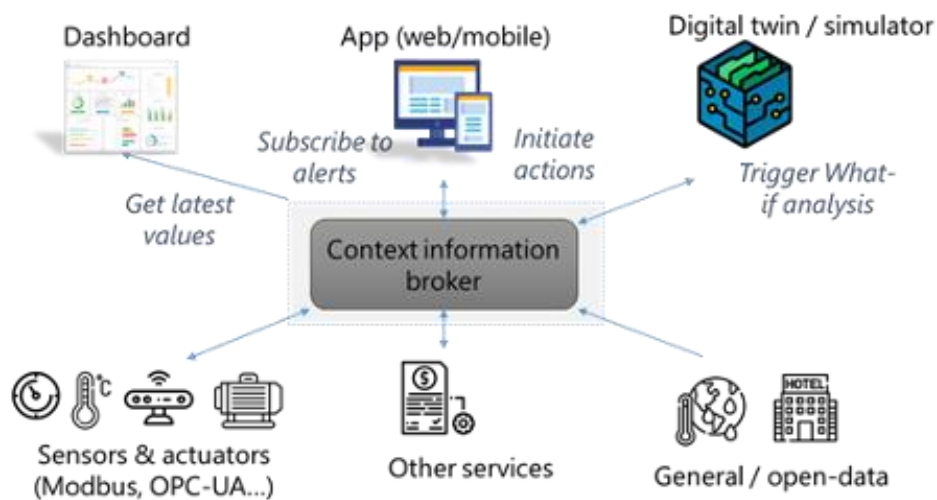


Figure 6. Typical IoT-inspired context broker architecture

The key benefit of the context broker approach is that the only part of the architecture that needs to be explicitly defined is that of the message format between broker and clients of the broker. This approach typically uses open data transfer protocols (http requests) with platform-agnostic data formats, e.g. JSON and JSON derivatives. Whilst this approach is not necessarily efficient in terms of data packet size, modern network performance does not generally make this an issue, even with ‘low performance’ networks, whilst the flexibility of key-value pair data packets makes it relatively trivial to extend component interoperability as new data providers and consumers are added to a broker system.

5.1 FIWARE Context Information Broker

FIWARE [3], as mentioned in Section 4.1.3, is being extended to provide a data interoperability platform for water systems under the FIWARE4Water project [4], and is currently being used as a context broker in multiple water system research programmes that UNEXE is part of, making it an ideal candidate for use in LOTUS.

5.1.1 FIWARE as a platform

As a platform, FIWARE [9] is a curated framework of open source platform components which can be assembled together and with other third-party platform components to accelerate the development of Smart Solutions. These components address the following areas of functionality: interfaces to IoT, core context management and, context processing, analysis and visualisation. This approach allows users of FIWARE to take a ‘mix and match approach’, with multiple components being provided for each area of functionality and allows users to engage with as much (or as little) of the FIWARE stack as required.

5.1.2 FIWARE as functionality

FIWARE effectively provides three functional components that map to the conceptual platform’s data collection, data management, data processing and presentation components. As a context broker architecture, the ‘interfaces to IOT’ component is responsible for packaging data from sensing devices and delivering them to the context broker (‘core context management’). The ‘core context management’ component is responsible for the short, medium and long-term management of data and the ‘data processing and presentation’ components are responsible for interrogating and displaying data from the context broker.

5.1.2.1 Interfaces to IOT

FIWARE provides a number of IOT Agents [10], supporting different data formats (JSON, LWM2M, Ultralight, LoRaWAN, OPC-UA and Sigfox) as well as a core agent library that is designed to be extended for custom data formats. The goal of these interfaces is to deliver data to the context manager using the NGSIv2 API [10] and associated data formats.

Given Use Case 4’s requirement to collect sensor data in bulk (RDC1), LOTUS sensors could be configured to download bulk data through their USB interface with a PC implementing the IOT Agent. Conversely, the LOTUS sensor could be configured as an IOT Agent to communicate directly with the context broker.

5.1.2.2 Core context management

FIWARE provides a number of context brokerages [13] depending back-end databases and data lifetimes. Typically, the Orion-LD context broker [12] is used for the collection and short-term management of data with Cygnus, Comet and Quantum Leap context managers providing longer-term, time-sliced storage for a variety of SQL and non-SQL databases.

Given Use Case 4’s requirement to store sensor data (RDM1 & RDM2) over a potentially long-time (given the need for seasonal data), it would make sense to pair an Orion-LD broker with Quantum Leap to provide long-term data storage.

5.1.2.3 Context processing, analysis and visualisation

FIWARE provides a number of generic tools for analysis and visualisation [14], whilst the NGSiv2 API [10] and JSON data formats provide users with a flexible interface for building tools from scratch, which is something UNEXE has done on other projects.

Given Use Case 4's dashboard requirements (RD1-RD5) and requirement for heat mapping, it's likely that a combination of FIWARE's visualisation tools and bespoke tools will meet these requirements, depending on target platform(s).

5.1.3 FIWARE data structures / formats

FIWARE's reference data structures are defined in a set of GitHub repositories [7]. For LOTUS sensors, datamodel.device [8] can be used as a data format that describes a physical device that can perform a particular activity, typically sensing some property from the environment (pressure, pH, salinity etc).

The flexible format of the device structure allows a device to define multiple 'controlled properties' that the device will capture, enabling this data structure to work with the LOTUS sensor's multiple sensing capabilities. Figure 7 details this with a device containing an array of 18 controlled properties in one sensor unit.

```

0 = Object {id: "urn:ngsi-ld:Sensor:dolni okol test sensor_1", type: "https://uri.etsi.org/ngsi-ld/default-context/Sensor",
  id = "urn:ngsi-ld:Sensor:dolni okol test sensor_1"
  type = "https://uri.etsi.org/ngsi-ld/default-context/Sensor"
  https://uri.etsi.org/ngsi-ld/name = Object {type: "Property", value: "dolni okol test sensor:1", metadata: }
  https://uri.fiware.org/ns/data-models#controlledProperty = Object {type: "Property", value: , metadata: }
    type = "Property"
    value = Array(18)
      0 = "https://uri.etsi.org/ngsi-ld/default-context/pressure transmitter"
      1 = "https://uri.fiware.org/ns/data-models#temperature"
      2 = "https://uri.etsi.org/ngsi-ld/default-context/ammonium"
      3 = "https://uri.etsi.org/ngsi-ld/default-context/chlorine"
      4 = "https://uri.fiware.org/ns/data-models#pressure"
      5 = "https://uri.fiware.org/ns/data-models#flowrate"
      6 = "https://uri.fiware.org/ns/data-models#conductivity"
      7 = "https://uri.fiware.org/ns/data-models#free chlorine"
      8 = "https://uri.fiware.org/ns/data-models#ph"
      9 = "https://uri.fiware.org/ns/data-models#iron"
      10 = "https://uri.fiware.org/ns/data-models#arsenic"
      11 = "https://uri.fiware.org/ns/data-models#nitrates"
      12 = "https://uri.fiware.org/ns/data-models#pesticide"
      13 = "https://uri.fiware.org/ns/data-models#fluoride"
      14 = "https://uri.fiware.org/ns/data-models#turbidity"
      15 = "https://uri.fiware.org/ns/data-models#hardness"
      16 = "https://uri.fiware.org/ns/data-models#carbonate"
      17 = "https://uri.fiware.org/ns/data-models#bicarbonate"
    length = 18

```

Figure 7. FIWARE device data structure showing multiple sensor data

In comparison, Figure 8, shows a device with a single sensor that captures pressure data, (<https://uri.fiware.org/ns/data-models#pressure>). This demonstrates the flexibility of the key-value pair approach to FIWARE's data format. It should also be noted that 'controlled properties' will contain key-value pairs that relate to data (and keys) that are relevant to their properties (sensors).

```
▼ 0 = Object {id: "urn:ngsi-Id:Device:003", type: "https://uri.fiware.org/ns/data-models#Device", https://uri.etsi.org/ngsi-Id/name: , https://uri.fiware.org/ns/data-  
  01 id = "urn:ngsi-Id:Device:003"  
  01 type = "https://uri.fiware.org/ns/data-models#Device"  
  ▶ https://uri.etsi.org/ngsi-Id/name = Object {type: "Property", value: "Isonzo Bridge sensor", metadata: }  
  ▼ https://uri.fiware.org/ns/data-models#controlledProperty = Object {type: "Property", value: "https://uri.fiware.org/ns/data-models#pressure", metadata: }  
    01 type = "Property"  
    01 value = "https://uri.fiware.org/ns/data-models#pressure"
```

Figure 8. FIWARE device data structure showing single sensor data

This is particularly attractive for the LOTUS platform as it effectively builds the context broker around a single data structure, in that the interfaces to IoT will create data using this structure, context management will store it and context processing & visualisation will process and visualise using this structure.

6 Platform demonstrator

6.1 Requirements

This section of the report covers the requirements of the current demonstrator iteration, given the COVID-related challenges of sensor development and data availability.

6.1.1 Goals

The over-arching goal for this iteration of the platform demonstrator is to take a step back from the LOTUS use-case centric logical architecture reached in the D5.1 deliverable and develop a platform that meets the broader goals of the for the platform, ‘to design a platform that will enable the consortium to demonstrate the advanced capabilities of the LOTUS sensor with the management solutions’, (Annex 1).

With that in mind, the platform demonstrator has four broad areas of functionality to consider: collection of data from LOTUS sensors and other sources, storage / management of data, processing of data and dashboard application(s) enabling users to interact with data.

6.1.2 LOTUS Sensor data

Given that the LOTUS sensor is not yet in a position to be able to generate on-site data, sensor data will be synthesized to give typical LOTUS sensor properties and readings.

6.1.3 Demonstrator functionality

For this iteration of demonstrator, the Guwahati case study has been taken as a starting point to provide the following functional goals:

6.1.3.1 Data collection

Appropriate sensor data will be generated and collected from multiple virtual sensors within the Guwahati area. Given that the sensors are virtual, their properties will be drawn from suitable LOTUS sensor properties and any other relevant properties and measures that support the demonstrator.

Sensor data will be generated periodically to simulate typical sensor operation, though this can be made more or less frequent as required.

Sensor data is assumed to be valid.



6.1.3.2 Data management

Once collected, sensor data will be stored in a time-series context broker, allowing temporal access. Stored sensor data will be persistent and will survive context broker restarts.

6.1.3.3 Data processing

Stored data will be processed on demand for user applications. It is assumed that data processing will not update data stored in the context broker.

6.1.3.4 Application dashboard

The demonstrator will provide two sets of application dashboard functionality: admin role & customer role.

The 'admin role' will provide a web-based application (browser) that will enable administrators to view and manage aspects of the demonstrator. Users will be able to set alarm limits for sensor properties, view the status of sensors geographically, as graphs and textually.

The 'customer role' will provide a mobile application that reports the status of the system to 'increase trust in piped water supply' (use case). The mobile app will be presented as a proof of concept that data can be shared between the platform and a mobile app rather than a 'final quality' mobile application.

6.1.3.5 Control dashboard

In order to inject and test 'edge cases' the demonstrator will include a control dashboard that will allow users to set the state of sensors and the value of their properties in order to show how the demonstrator responds to situations.

6.1.4 User-centric Use Cases

Much of the platform demonstrator is server-based, therefore, to give a clearer view of functionality, the following user-centric use cases have been considered.

6.1.4.1 Administrator webapp

- User sets alarm limits for sensor property
System updates sensor property alarm limits
- User views alarm state for sensor properties (textually)
System generates a list of sensors that are currently generating alarms and presents them to user through webapp.

- User views current status of sensors in geographic context
System builds sensor information and presents to user in a map context through webapp
- User views historical status of a sensor property in graphical context
System builds sensor information and presents to user in a graph context through webapp

6.1.4.2 Customer mobile app

- Customer receives up-to-date information about state of water network / quality
Mobile app requests network state information, system responds with information for mobile app user.

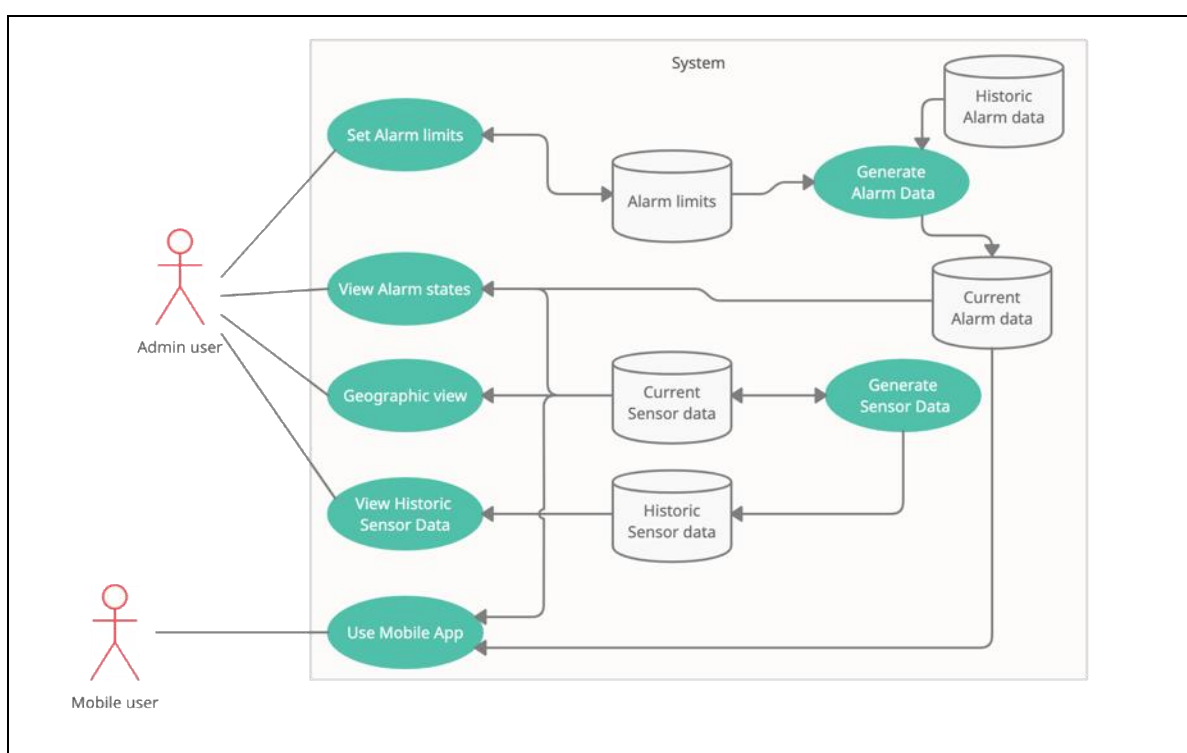


Figure 9 - System use case collaborations

6.2 Implementation

6.2.1 Server / Context broker

For this iteration of platform demonstrator, server and context broker functionality has been combined and realised through Flask. This approach was undertaken primarily to improve development iteration times, making it a simpler process to deploy, build and refine context broker contexts and content without the need to go through FIWARE installation and management procedures. However, the next iteration of platform will use a full FIWARE installation.

To simulate LOTUS sensor data, the server runs a periodic threaded process that creates plausible FIWARE device records every 15 minutes, though this value can be changed in code. Users can make use of the control screen to switch the state of sensors (on or offline) and to set sensor properties to read in or out of spec. Sensor reading specifications are currently set in-code.

6.2.2 Administrator webapp

The administrator webapp is implemented as a collection of html pages that are dynamically served through Flask and its jinja mark-up language. In addition, the html files use local JavaScript functionality to dynamically build content in situations where it does not make sense to serve layout content from the server, e.g. updating map view markers, updating graphs and switching between view and set modes in the alert screen.

In general, client data is requested through http requests which are served up through the Flask server, the general request handling approach being that the server is responsible for all the ‘heavy lifting’ leaving the client to just visualise the response.

6.2.2.1 View and Set alerts

The view and set alerts screen allows users manage sensor properties by setting the minimum and maximum values for sensor properties, view any properties that are outside those ranges and any sensors that are off-line.

To set an alert, the alert screen will send a `post_alert` request to the server with minimum, maximum and alert enabled attributes for the given sensor property. On receipt, the server will update the alert settings and process the alerts, Figure 10.

In contrast, the view alert screen will request sensor alert data (`get_alert_data`). On receipt, the server will collect the current alert settings and currently active alerts using both data sources to create a client-side response which will show out of range sensor properties, their expected ranges and their current readings.

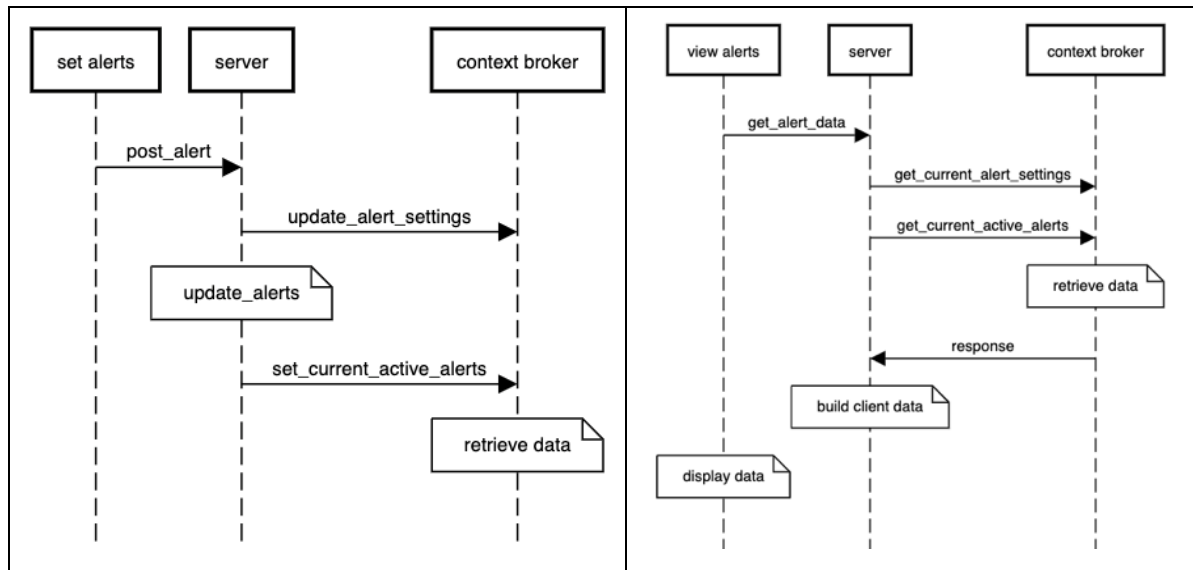


Figure 10 – Set Alert sequence (left), view alerts (right)

On entering each value or setting the active flag, the client sends the current property state to the server. On receipt, the server updates the current alert settings, generating new alerts (where necessary).

6.2.2.2 Geographic Context

The geographic context screen makes periodic http requests (`get_device_data`) to the server to receive up-to-date sensor information containing their position, current state and textual information relating to each property that the sensor is recording.

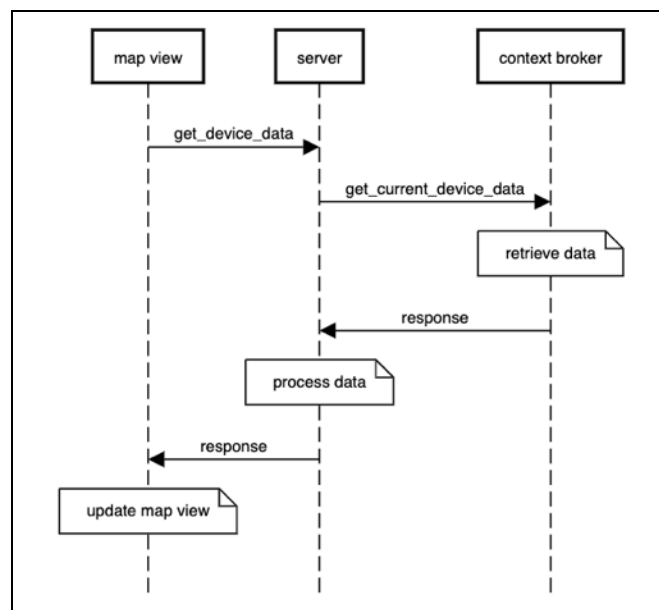


Figure 11 - Geographic context, sequence diagram

On receiving the request, the server will query the context broker for the current state of all devices and use this data to create a client-side response, Figure 11.

6.2.2.3 Graphing context

The graphing context screen allows users to display time-based graphs based on any properties of any sensors. To do this, the screen makes two requests. The first, `get_graph_params`, will collect a list of devices and their parameters, Figure 12. This is then used to build the two dropdowns in the screen with the client responsible for updating the right-hand properties dropdown to reflect the properties for the currently selected sensor in the left-hand dropdown.

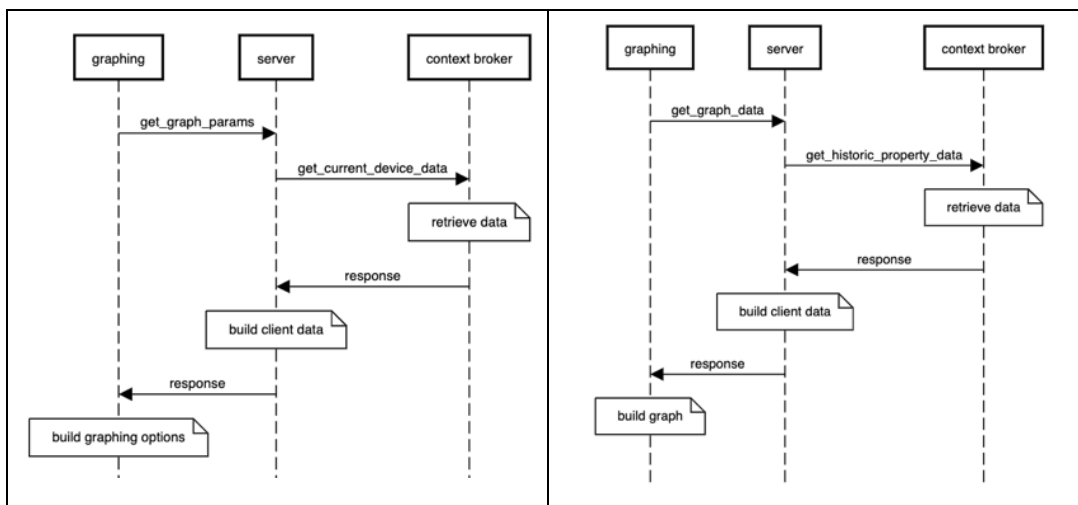


Figure 12 – Graphing context, get graph params (left), build graph (right)

When the right-hand dropdown is selected, the client will request `get_graph_data` from the server, passing the selected device and property. The server will respond with a package of data containing time-based reading values and headings that are passed into a highcharts map for rendering.

6.2.3 Customer mobile app

The customer mobile app is implemented as a Cordova-wrapped webapp as a data delivery proof-of-concept, i.e. it is possible to communicate with the Flask server using http requests on a mobile device.

Unlike the administrator webapp, the customer mobile app does not receive dynamic html content, all the content for the mobile app is bespoke, with the exception of the table generation, which is handled on the server and sent to the mobile app.

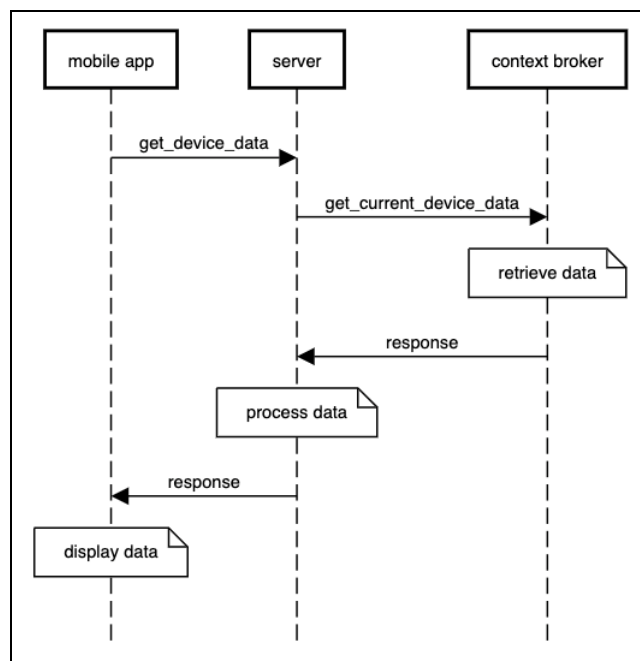
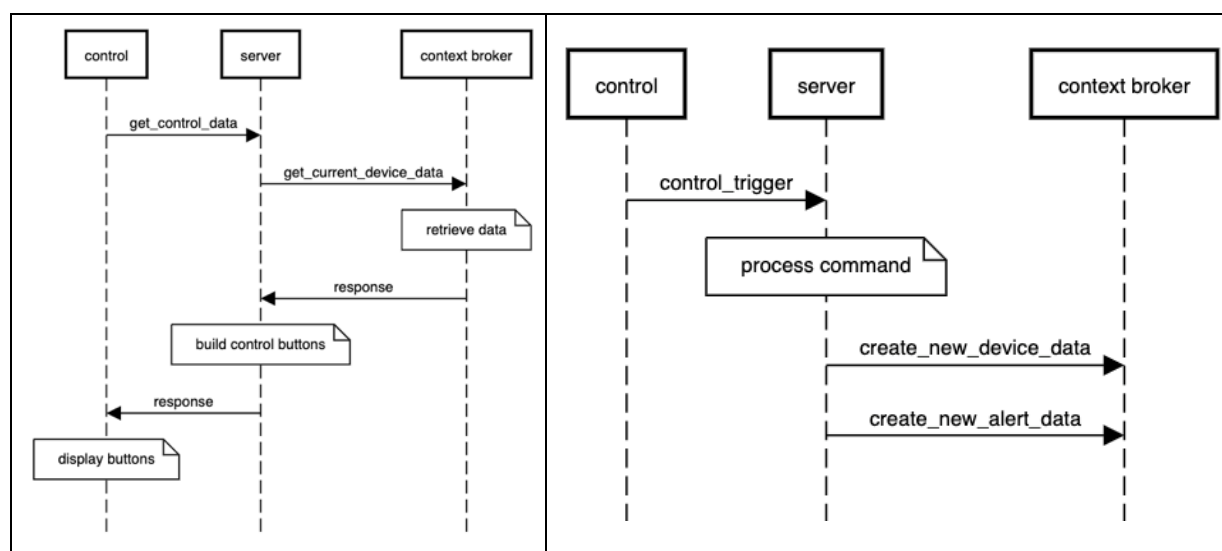


Figure 13 - Mobile data sequence diagram

The mobile app re-uses the `get_device_data` request (as used by the graphing context) to collect current sensor data that can be displayed on the mobile screen, Figure 13.

6.2.4 Control webapp

The control webapp, Figure 14, allows users to control the state sensors and value of their properties.

Figure 14 - Control context, `get_control_data` (left) , `control_trigger` (right)

To create the screen of buttons, the control screen sends a `get_control_data` request to the server. On receipt, the server iterates through all sensors and properties to create a set of command records. This data is sent back to the control screen and is used to create the button instances.

When a button is clicked, Figure 14 (right), a `control_trigger` request is sent to the server containing attributes relating to the sensor, property and desired property state. This data is parsed by the server and used to generate a new set of device data, effectively updating the server at that point.

6.2.5 Anomaly detection

Work has been undertaken to bring the water quantity and quality research (WP4) into the integrated platform to provide anomaly detection and localisation, Figure 15.

Time	Message
2021-01-27 09:22:44	Anomaly detected in device GT-1 Pressure measurement , based on 4 consecutive measurements out of expected range: <ul style="list-style-type: none"> • Expected lower limit of 61.472 breached at 2021-01-27 09:22:36 with a value of 53.003 • Expected lower limit of 61.472 breached at 2021-01-27 09:22:38 with a value of 54.266 • Expected lower limit of 61.472 breached at 2021-01-27 09:22:41 with a value of 53.084 • Expected lower limit of 61.472 breached at 2021-01-27 09:22:43 with a value of 53.773
2021-01-27 09:22:42	Anomalies localised: Potential network links for cause of Pressure anomaly in device(s) ['GT-1'] identified. Likely candidates: ['3092019_1882.3092019_1883.1', '3092019_1883.3092019_1902.1', '3092019_12063.3092019_1903.1', '3092019_7650.3092019_12063.1', '3092019_7636.3092019_7635.1', '3092019_1885.3092019_1905.1', '3092019_7647.3092019_12080.1', '3092019_12073.3092019_7641.1', '3092019_12085.3092019_7631.1', '3092019_12072.3092019_1885.1', '3092019_12086.3092019_7631.1', '3092019_7646.3092019_7647.1', '3092019_7636.3092019_7641.1', '3092019_1882.3092019_12073.1', '3092019_7634.3092019_7636.1', '3092019_12080.3092019_7640.1', '3092019_7630.3092019_7637.1']

Figure 15 – Examples of anomaly detection (top) and localisation (bottom)

The process works by taking an historic set of sensor property readings to form a baseline of operations. New property readings can be compared with the baseline to determine if a sensor property is anomalous, in comparison with being in, or out, of alert range.

This has not been included in the current demonstration but will be included in the next iteration.

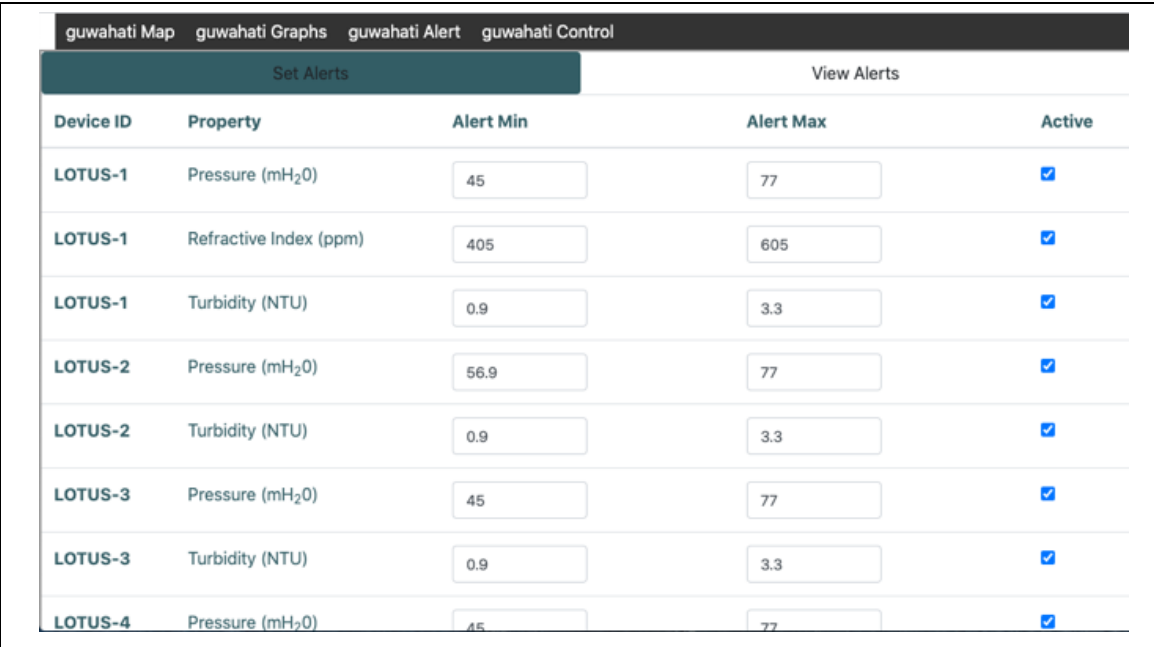
6.3 Results

The platform demonstrator is current running as a Flask server running on <http://167.172.49.166:5100/>. The platform synthesizes sensor data and serves web content for the administrator webapp and handles http requests for the customer mobile app.

6.3.1 Administrator webapp

6.3.1.1 Set alerts

The set alerts screen allows users to set the minimum and maximum values for each property that each sensor consists of, Figure 16.



guwahati Map guwahati Graphs guwahati Alert guwahati Control				
Set Alerts			View Alerts	
Device ID	Property	Alert Min	Alert Max	Active
LOTUS-1	Pressure (mH ₂ O)	45	77	<input checked="" type="checkbox"/>
LOTUS-1	Refractive Index (ppm)	405	605	<input checked="" type="checkbox"/>
LOTUS-1	Turbidity (NTU)	0.9	3.3	<input checked="" type="checkbox"/>
LOTUS-2	Pressure (mH ₂ O)	56.9	77	<input checked="" type="checkbox"/>
LOTUS-2	Turbidity (NTU)	0.9	3.3	<input checked="" type="checkbox"/>
LOTUS-3	Pressure (mH ₂ O)	45	77	<input checked="" type="checkbox"/>
LOTUS-3	Turbidity (NTU)	0.9	3.3	<input checked="" type="checkbox"/>
LOTUS-4	Pressure (mH ₂ O)	45	77	<input checked="" type="checkbox"/>

Figure 16 – Set Alert Screen

On entering each value or setting the active flag, the client sends the current property state to the server. On receipt, the server updates the current alert settings, generating new alerts (where necessary).

6.3.1.2 View alerts

The view alerts screen allows users to view the current active alerts. Alerts are generated under two conditions: 1. a sensor property is outside of the minimum / maximum ranges defined for it or 2. a sensor is offline (in fiware terms, its device state is 'Red').

guwahati Map guwahati Graphs guwahati Alert guwahati Control		
Set Alerts		View Alerts
Device ID	Property	Value
LOTUS-1	Pressure:Device Unavailable	66.5128
LOTUS-1	Refractive Index:Device Unavailable	530.1908
LOTUS-1	Turbidity:Device Unavailable	2.7091
LOTUS-2	Pressure:Out of Range: (56.9-77.0)mH ₂ O	36.2328
End of triggered alert list		

Figure 17 - View Alert Screen

6.3.1.3 Geographic Context

The geographic context screen provides an environment for users where sensor data is displayed in a map environment, Figure 18. The map can be viewed in a stylised vector format or as a traditional satellite image.

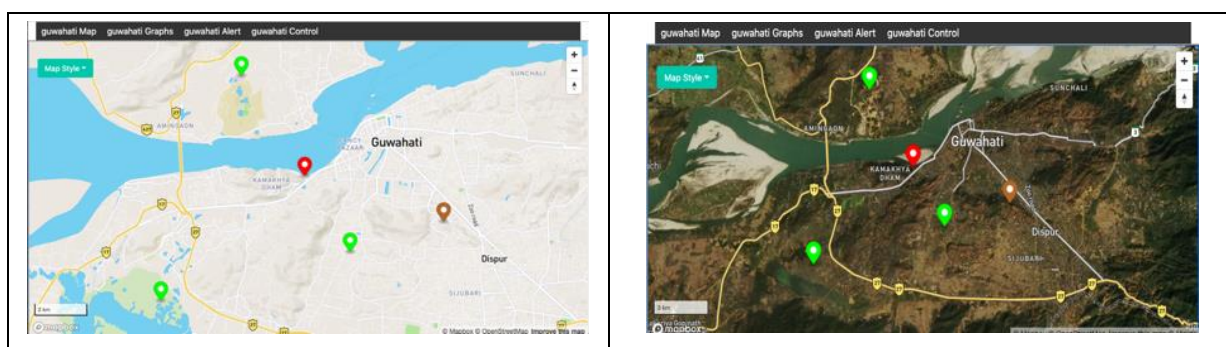


Figure 18 - Geographic context, map (left) and satellite (right)

To aid evaluating sensor state, the sensors are colour-coded (Figure 19). Clicking on a sensor marker will reveal sensor details, Figure 20

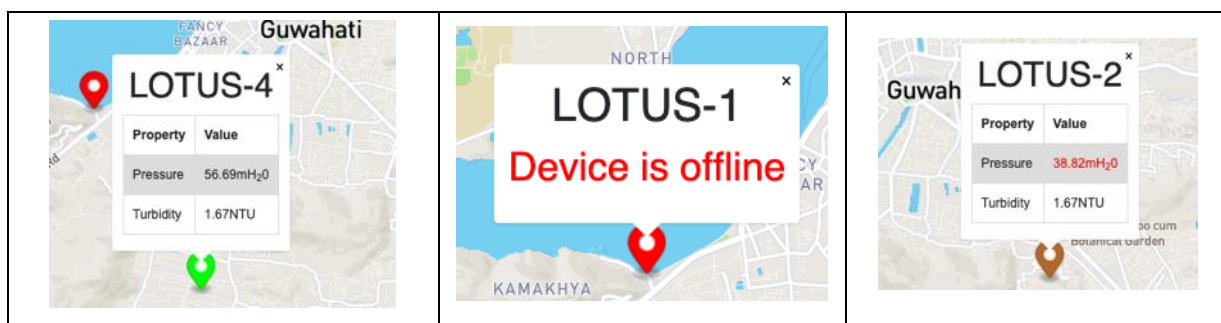


Figure 19 - Sensor status: working and properties in range (left), offline (centre) and working but at least one property out of range

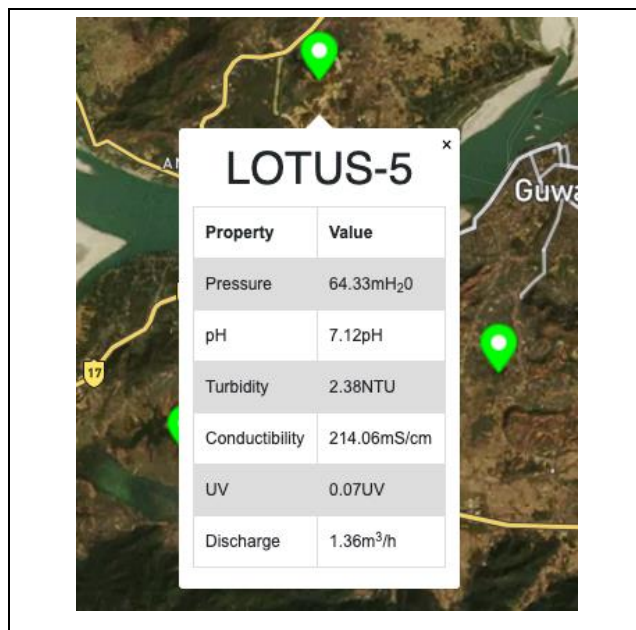


Figure 20 - Sensor showing property details

6.3.1.4 Graphing context

The graphing screen allows users to view sensor property data over an historic timeline, Figure 21. Sensors may be chosen from the left dropdown whilst their individual properties can be selected with the right dropdown.

The graph shows property readings over time, with the red horizontal lines showing the minimum and maximum alert values for the selected property. This instance, the property has been generating 'out of range' warnings, given that most of its values are under its minimum alert threshold.

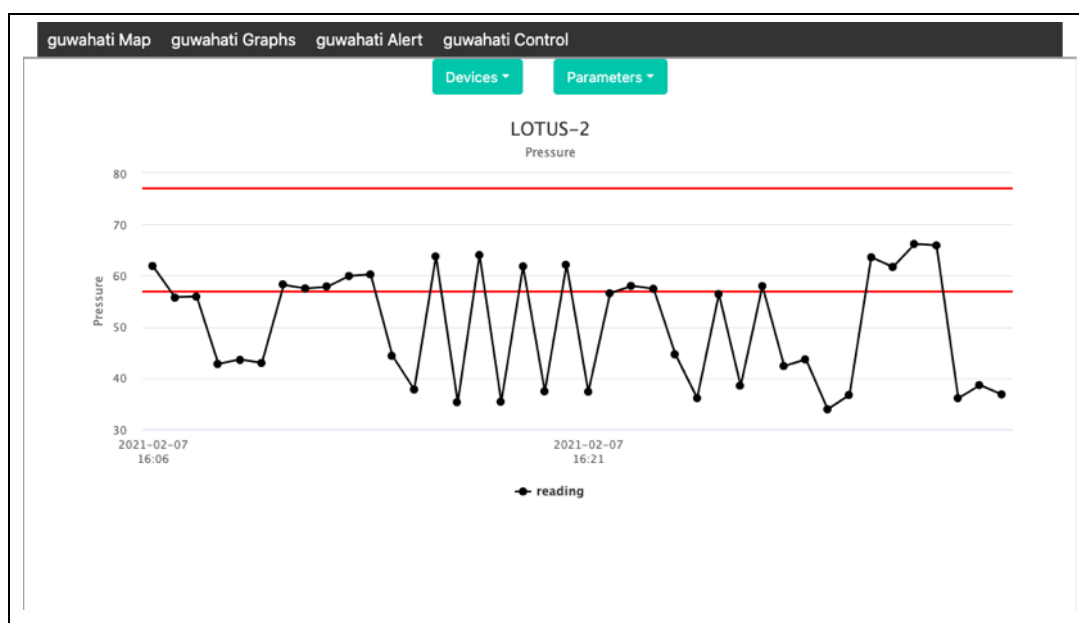


Figure 21 - Graph view of sensor property

6.3.2 Customer Mobile App

The customer mobile app is a proof-of-concept app that demonstrates that data can be polled (via http requests) from the platform to a remote device.

Currently, the mobile app receives sensor data on-demand from a user, Figure 22. For a more refined mobile application, user data would be processed either on the platform or another service and passed to the mobile app using an appropriate transport (http requests, sockets etc)

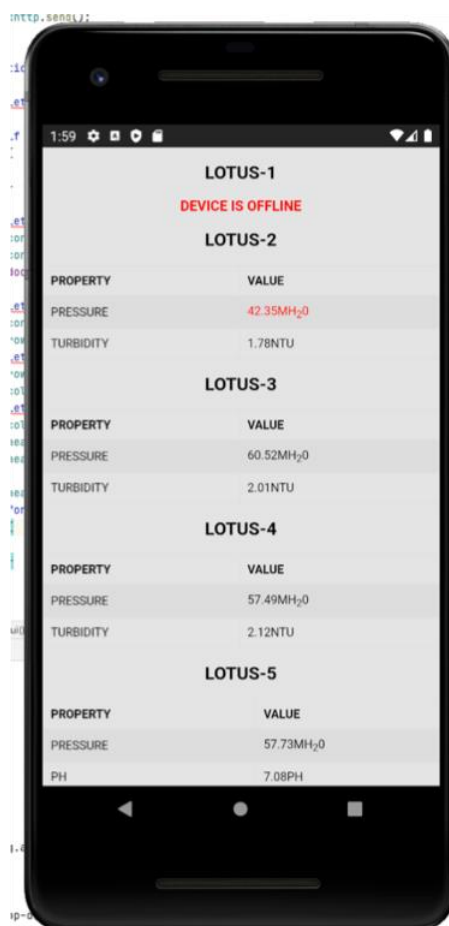


Figure 22 - Mobile app

6.3.3 Control webapp

The control webapp, Figure 23 is currently part of the admin webapp but exists as an html link on the webapp and could easily be relocated to a separate webapp.

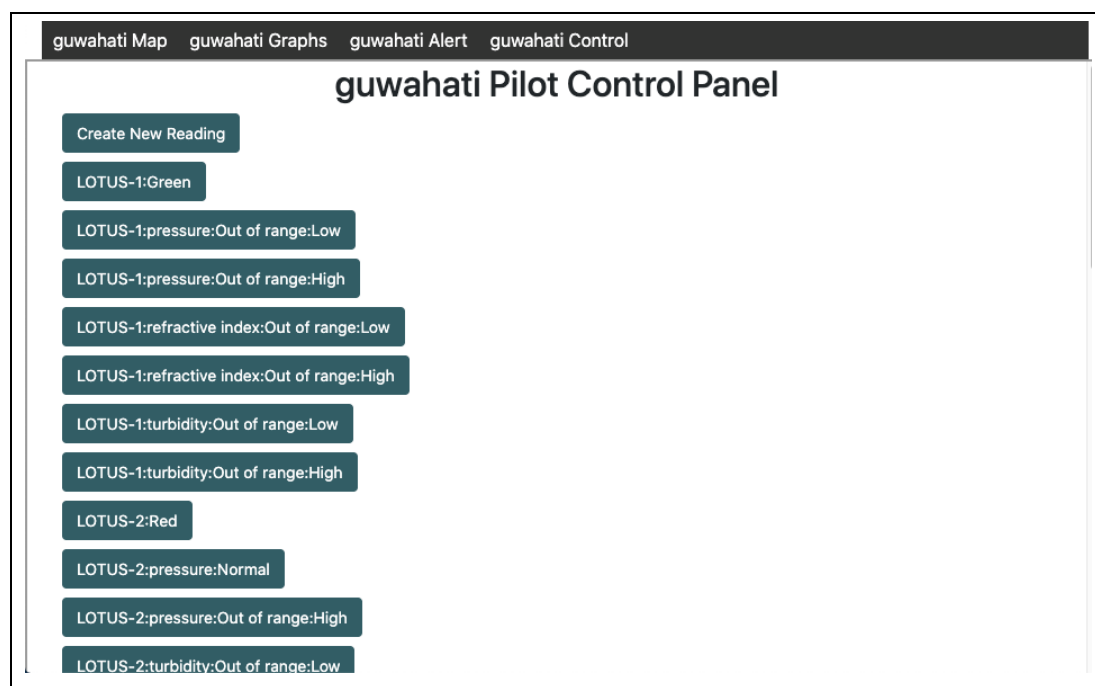


Figure 23 - Control Panel

The role of the control webapp is to inject the context broker with new states for sensors and their properties allowing situations to be created without the need to write explicit code.

The buttons are split into three different types:

- Create new reading
This button will force the platform to create a new set of sensor readings
- Sensor Green | Red
These buttons will toggle the operational state of a sensor: red implies off-line, green implies on-line
- Sensor Property Normal | Out of Range: Low | Out of Range: High
These buttons will toggle modes of data generation. Normal implies that the property will remain 'in range' of the values that are set for it (in code rather than the alert limits). Out of Range: Low will return a value that is beneath the minimum alert value, whilst Out of Range: High will return a value that is above the maximum alert value.

For the sensor and sensor property buttons, the current state of the button is not displayed, for example if a sensor is currently on-line, only the Red (off-line) state will be shown.

7 Conclusions and Next Steps

7.1 Conclusions

7.1.1 Proof of concept

The client/server context broker demonstration presented show that the platform approach is valid, albeit with some caveats that are discussed in section 7.2. As presented, the demonstration manages persistent data, simulates LOTUS sensor data collection, processing and storage and provides functionality for a range of users (admin and end-users) on a range of devices (web and mobile).

7.1.2 Anomaly detection and localisation

Currently, the anomaly detection and localisation system performs well with synthesized data. However, the platform has recently moved to an entirely persistent data model which is causing some issues with this system.

7.2 Next steps

7.2.1 Integration with suitable FIWARE context brokers

This iteration of the demonstrator was developed without the use of FIWARE context brokers, but with FIWARE data types. This approach was taken for several reasons: Firstly, there are a range of context brokers available and we didn't want to commit the project to a particular technology stack before necessary. Secondly, much of the work undertaken in this iteration of development wasn't predicated on FIWARE context brokers, just on the ability to store and retrieve data. This is particularly so in the cases where different datasets and functionality are being worked on in parallel – having a core context broker can slow that process down, as we have found on other projects.

However, FIWARE context brokers are clearly the solution for LOTUS moving forward and they will form the core of the next iteration of development (D5.7).

As part of this work, we will develop the platform to be more closely aligned to a context broker model, with multiple services rather than the current monolithic approach.

7.2.2 LOTUS data collection

In this iteration of development, it has been necessary to work with synthetic data collection as none of the LOTUS use cases have been in a position to provide live data and the LOTUS sensor is not in a position to provide data either.

For the next stage of development, we will look to work with real end-user data, in particular to meet the goal of having a full-cycle context broker and to address the deliverable goal of ‘signal processing tools (cleaning and error detection)’.

7.2.3 Anomaly detection integration

The anomaly detection algorithms have been developed from Fanlin’s initial proof of concept to a solution that worked well in our initial platform demonstrator. However, since moving to a fully persistent data model, the anomaly baseline property generation functionality is producing erroneous results.

This will be addressed as part of the D5.7 work.

7.2.4 Security integration

The current platform demonstrator has no considerations for security. This will be addressed as part of the D5.7 work.

7.2.5 SCADA integration

The current platform demonstrator has no considerations for SCADA integration. This will be addressed as part of the D5.7 work.

7.2.6 Enhanced mobile integration

The current platform demonstrator has a simple http request interface for interacting with mobile applications. Whilst this works, it is not ideal and a websocket interface will be provided as part of the D5.7 work.

8 References

- [1] Open Water Analytics. "OWA-EPANET Toolkit 2.2." http://wateranalytics.org/EPANET/toolkit_versions.html (accessed 15th May 2020).
- [2] USEPA. "EPANET Application for Modeling Drinking Water Distribution Systems." <https://www.epa.gov/water-research/epanet> (accessed 18th May 2020).
- [3] European Parliament "Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE)," ed, 2007.
- [4] FIWARE. "FIWARE: The Open Source Platform for Our Smart Digital Future." <https://www.fiware.org/> (accessed 23rd June 2020).
- [5] European Commission. "FIWARE for the Next Generation Internet Services for the WATER sector." <https://cordis.europa.eu/project/id/821036> (accessed 23rd June 2020).
- [6] OWASP. "Top 10 Web Application Security Risks." <https://owasp.org/www-project-top-ten/> (accessed 20th May 2020).
- [7] Smart Data Models – FIWARE <https://github.com/smart-data-models> (accessed 1st September 2020).
- [8] FIWARE Device specification <https://smart-data-models.github.io/dataModel.Device/Device/doc/spec.md> (accessed 1st September 2020).
- [9] FIWARE Developers introduction <https://www.fiware.org/developers/> (accessed 1st September 2020).
- [10] FIWARE - INTERFACE WITH IOT, ROBOTS AND THIRD-PARTY SYSTEMS <https://www.fiware.org/developers/catalogue/> (accessed 1st September 2020).
- [11] FIWARE-NGSiv2 Specification <http://fiware.github.io/specifications/ngsiv2/stable/> (accessed 1st September 2020).
- [12] FIWARE Context.Orion-LD <https://github.com/FIWARE/context.Orion-LD> (accessed 1st September 2020).
- [13] FIWARE Core Context Management <https://github.com/FIWARE/catalogue/blob/master/core/README.md> (accessed 1st September 2020).
- [14] FIWARE Context processing, analysis and visualisation <https://github.com/FIWARE/catalogue/blob/master/processing/README.md> (accessed 1st September 2020).