OGC SCIRA Pilot Engineering Report

Table of Contents

1. Subject . . . 4
2. Executive Summary . . . 5
   2.1. Document contributor contact points . . . 5
   2.2. Foreword . . . 6
3. References . . . 7
4. Terms and Definitions . . . 8
   4.1. Abbreviated Terms . . . 9
5. Overview . . . 13
6. Solutions Architecture . . . 16
   6.1. Scenarios . . . 16
   6.2. Use Case Diagrams . . . 17
   6.3. Class Diagrams . . . 22
   6.4. Reference Architecture . . . 26
7. SmartHub System . . . 28
   7.1. D1: First Responder Tracking Sensor . . . 29
   7.2. D2: First Responder Health Sensor . . . 31
      7.2.1. keys App (Apple Watch) . . . 31
   7.3. D3: First Responder Environment Sensor . . . 34
   7.4. D4: First Responder *SmartHub* . . . 35
   7.5. D5: First Responder Incident SensorHub . . . 40
   7.6. D6: Cloud-Based SensorHub and Catalog . . . 41
      7.6.1. Pulling Data from SensorHub . . . 42
      7.6.2. Pushing Data to SensorHub . . . 44
8. Command Communication System . . . 50
   8.1. D12: 3D Dashboard Application . . . 50
      8.1.1. Add 3D Data Service . . . 51
      8.1.2. Monitoring . . . 53
      8.1.3. Sensor Observations . . . 54
      8.1.4. Alerts . . . 55
      8.1.5. Incidents . . . 56
      8.1.6. Routing . . . 58
      8.1.7. Flood Model Output . . . 59
   8.2. D13: Command & Communication System . . . 61
   8.3. D17: Community Mobility Navigation, Alert & Sensing Application . . . 63
9. Indoor Navigation System . . . 64
   9.1. D8: First Responder Indoor Navigation Application . . . 64
   9.2. D18: RFID/BLE Indoor Navigation Beacons . . . 65
10. Street Navigation System . . . 69

10.1. D15: Traffic Management & Routing Model . . . 69
10.1.1. Architecture and Design . . . 70
10.1.2. Routing Engine Setup . . . 70
10.1.3. Extension to Routing API . . . 74
10.1.4. Road closures . . . 75
10.1.5. Main Findings and Recommendations . . . 76
10.2. D17: Community Mobility Navigation, Alert & Sensing App . . . 76
10.2.1. Example of a "Street Closure" Observation . . . 77
10.3. D20: 3D City Model . . . 78
11. Flood Sensing System . . . 79
11.1. D11/D16: Legacy Stream Sensors . . . 79
11.2. D14: Flood & Inundation Model . . . 82
11.2.1. Load Data . . . 83
11.2.2. Flow Direction Algorithm . . . 85
11.2.3. Alerts and Model Output . . . 86
11.2.4. Flood Working Model . . . 87
11.3. D17: Community Mobility Navigation, Alert and Sensing App . . . 92
12. Road Sensing System . . . 94
12.1. D10: Smart Traffic/Road Sensor . . . 94
12.1.1. Smart Traffic Sensor System Architecture . . . 95
12.1.2. Smart Traffic Sensor Components and Configurations . . . 97
12.1.3. Demos and Results . . . 104
13. Findings . . . 108
13.1. Application Development Findings . . . 108
Appendix A: Master Scenario Events List . . . 111
Scenario 1: River Monitoring . . . 112
Scenario 2: Flash Flooding . . . 120
Scenario 3: Assistance to Vulnerable Populations . . . 132
Scenario 4: Building Fire . . . 143
Scenario 5: Vehicle Accident . . . 162
Computing Environment . . . 179
App Interactions . . . 180
Appendix B: How to train YOLOv3 . . . 181
Using pre-trained YOLO model . . . 182
Training YOLO . . . 183
A) Training and testing dataset . . . 183
B) Network configuration . . . 184
C) Modifications to .data file . . . 184
D) Starting the training process . . . 185
E) Evaluation of the trained model . . . 185
Appendix C: Revision History . . . 187

Appendix D: Bibliography 188

Publication Date: 2020-05-04

Approval Date: 2020-04-29

Submission Date: 2020-03-01

Reference number of this document: OGC 20-011

Reference URL for this document: http://www.opengis.net/doc/PER/scira-pilot

Category: OGC Public Engineering Report

Editor: Sara Saeedi

Title: OGC SCIRA Pilot Engineering Report

OGC Public Engineering Report

COPYRIGHT

Copyright © 2020 Open Geospatial Consortium. To obtain additional rights of use, visit http://www.opengeospatial.org/

WARNING

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Public Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR’s sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Subject

This engineering report (ER) captures Smart City Interoperability Reference Architecture (SCIRA) Pilot implementation outcomes and findings to demonstrate the risk mitigation and safety capability of the SCIRA interoperable and standard-based architecture. SCIRA Pilot is an OGC (Open Geospatial Consortium) Innovation Program project sponsored by the US Department of Homeland Security (DHS) Science & Technology (S&T) in collaboration with the city of St. Louis, Missouri. The purpose of this project is to advance standards for smart and safe cities and develop open, interoperable design patterns for incorporating the Internet of Things (IoT) sensors into city services.

Chapter 2. Executive Summary

A lack of consensus on both common terminology and smart city architectural principles can result in divergent, proprietary and even contradictory specifications. Often, they fail to provide sufficient interoperability and scalability of the underlying IoT, and Cyber-Physical Systems (CPS) technologies to provide a suitable foundation for many smart cities applications.

SCIRA provides free deployment guides, a reusable design toolkit, and other resources to plan, acquire, and implement standards-based, cost-effective, vendor-agnostic, and future-proof smart city Information Technology (IT) systems and networks using technologies such as IoT, Sensor webs, and Geospatial frameworks. The objective of this Pilot is to research, design, and test the SCIRA as a reusable design toolkit that shows how to integrate proprietary IoT sensors into a common framework for public safety applications at the community level.

This Pilot focused on the City of St. Louis, Missouri to refine elements of interoperable smart city architecture through implementation and testing in functional, 'real world' public safety scenarios. The SCIRA Pilot concluded with on-site exercises and then a demonstration intended to establish a reusable example of SCIRA-based deployment. The operational application of the SCIRA design toolkit showed how cities can reap the benefits of standards-based interoperability.

2.1. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Table 1. Contacts

Name	Organization	Role
Sara Saeedi, PhD	University of Calgary	Editor
Josh Lieberman, PhD	OGC	Contributor
Steve Liang, PhD	University of Calgary	Contributor
Brian Hawkins	Coolfire	Contributor
Charles Chen	Skymantics, LLC	Contributor
Ignacio Correas	Skymantics	Contributor
Igor Starkov	Ecodomus	Contributor
Jason MacDonald	Compusult	Contributor
Marcus Alzona	keys / keys net LLC	Contributor
Mike Botts, PhD	Botts Innovative Research Inc.	Contributor
Mahnoush Mohammadi Jahromi	University of Calgary	Contributor
Sepehr Honarparvar	University of Calgary	Contributor
Sushmapriya Maddala	CYIENT Ltd.	Contributor

2.2. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 3. References

The following normative documents are referenced in this document.

NOTE: Only normative standards are referenced here, e.g. OGC, ISO or other SDO standards.

All other references are listed in the bibliography.

- OGC: OGC 15-078r6, OGC® SensorThings API Part 1: Sensing [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html]
- OGC® OGC® SWE Common [https://www.ogc.org/standards/swecommon]
- OGC® Sensor Observation Service Interface Standard [https://portal.opengeospatial.org/files/?artifact_id=47599]
- OGC: OGC 18-053r2, OGC® 3D Tiles Specification 1.0 [http://docs.opengeospatial.org/cs/18-053r2/18-053r2.html]
- OGC: OGC 12-019, OGC® City Geography Markup Language (CityGML) Encoding Standard [https://portal.opengeospatial.org/files/?artifact_id=47842]
- OGC: OGC 06-121r9, OGC® Web Services Common Standard (2015) [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- OGC: OGC 14-065, OGC® WPS 2.0 Interface Standard (2018) [http://docs.opengeospatial.org/is/14-065/14-065.html]
- OGC: OGC 09-025r2, OGC® Web Feature Service 2.0 Interface Standard – With Corrigendum (2014) [http://docs.opengeospatial.org/is/09-025r2/09-025r2.html]
- OGC: OGC 17-069r3, OGC® API - Features - Part 1: Core (2019) [http://docs.opengeospatial.org/is/17-069r3/17-069r3.html]
- OGC® Web Map Service (WMS) [https://www.ogc.org/standards/wms]
- OGC® Catalogue Services for the Web (CSW) standard [https://www.ogc.org/standards/cat]
- MQTT Version 5.0 OASIS Standard [https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html]
- IETF: The GeoJSON Format (RFC 7946), Internet Engineering Task Force (IETF) [https://tools.ietf.org/html/rfc7946]
- The JSON Data Interchange Syntax [http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf]
- Khronos Group, glTF 2.0 Specification, Ecma Standard [https://www.khronos.org/gltf/]

Chapter 4. Terms and Definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- 3DTiles

```
An OGC Community Standard for streaming large heterogeneous 3D geospatial datasets.
```

- Application programming interface

```
A standard set of documented and supported functions and procedures that expose the
capabilities or data of an operating system, application or service to other
applications (adapted from ISO/IEC TR 13066-2:2016)
```

- Deep Learning

```
It is a subset of data-driven Machine Learning algorithms trained by large
quantities of data.
```

- Feature

```
Abstraction of real-world phenomena (source: ISO 19101-1:2014)
```

- GeoJSON

```
"GeoJSON is a geospatial data interchange format based on JavaScript Object
Notation (JSON).  It defines several types of JSON objects and the manner in which
they are combined to represent data about geographic features, their properties,
and their spatial extents. GeoJSON uses a geographic coordinate reference system,
World Geodetic System 1984, and units of decimal degrees."
(https://tools.ietf.org/html/rfc7946[The GeoJSON Format])
```

- GeoPackage

```
An open, non-proprietary, platform- independent and standards-based data format for
geographic information implemented as a SQLite database container.
```

- gLTF

```
(GL Transmission Format) is a royalty-free specification developed and maintained
by Khronos for the efficient transmission and loading of 3D scenes and models by
applications.
```

- MQTT

```
Message Queuing Telemetry Transport is an open lightweight, network protocol that
transports messages between devices.
```

- REST

```
The Representational State Transfer (REST) style is an abstraction of the
architectural elements within a distributed hypermedia system.
It encompasses the fundamental constraints upon components, connectors, and data
that define the basis of the Web architecture, and thus the essence of its behavior
as a network-based application.
An API that conforms to the REST architectural principles/constraints is called a
RESTful API.
```

- Sensor

```
An entity capable of observing a phenomenon and returning an observed value. Type
of observation procedure that provides the estimated value of an observed property
at its output. [OGC 12-000]
```

- SensorThings

```
OGC Standard providing an open and unified framework to interconnect IoT sensing
devices, data, and applications over the Web.
```

4.1. Abbreviated Terms

- 3D 3 Dimensional
- AI Artificial Intelligence
- API Application Programming Interface
- APP Applications
- AVL Automated Vehicle Location
- AWS Amazon Web Services
- BLE Bluetooth Low Energy
- BYOD Bring Your Own Device

- CCTV Closed Circuit Television
- CFP Call for Participation
- CityGML City Geography Markup Language
- COP Common Operating Picture
- CPS Cyber-Physical Systems
- CPU Central Processing Unit
- CR Change Request
- CSI Camera Serial Interface
- CSW Catalogue Services for the Web
- CUDA Compute Unified Device Architecture
- CUDNN CUDA Deep Neural Network library
- DEM Digital Elevation Model
- DHS Department of Homeland Security
- DL Deep Learning
- DTM Digital Terrain Model
- DWG Domain Working Group
- EC2 Elastic Compute Cloud
- EMS Emergency Medical Service
- ER Engineering Report
- FCU Feng Chia University
- FLIR Forward Looking Infrared Radar
- FME Feature Manipulation Engine
- FP False Positive
- fps frames per second
- GeoJSON Geospatial JavaScript Object Notation
- GlTF Graphics Library Transmission Format
- GPS Global Positioning System
- GPU Graphics Processing Unit
- HD High Definition
- HTTP HyperText Transfer Protocol
- I3S Indexed 3D Scene layers
- ID/Id Identification
- IoT Internet of Things
- IoU Intersection of Union
- ISO International Organization for Standardization

- IT Information Technology
- JSON JavaScript Object Notation
- LTE Long Term Evolution
- mAP Mean Average Precision
- MIPI-CSI Mobile Industry Processor Interface Camera Serial Interface
- MicroSD Micro Secure Digital
- ML Machine Learning
- MO Missoury
- MQTT Message Queuing Telemetry Transport
- MSEL Master Scenario Events List
- NN Neural Network
- NGFR Next Generation First Responder
- NYS New York State
- OGC Open Geospatial Consortium
- ORM OGC Reference Model
- OSH Occupational safety and health
- OSM Open Street Map
- OWS OGC Web Services
- REST Representational state transfer
- RFID Radio-Frequency IDentification
- S&T Science & Technology
- SCIRA Smart City Interoperability Reference Architecture
- SMS Short Message Service
- SOS Sensor Observation Service
- STA SensorThings API
- SWE Sensor Web Enablement
- SWG Standards Working Group
- TIE Technology Integration Experiment
- TP True Positive
- UML Unified Modeling Language
- URL Uniform Resource Locator
- US United States
- USB Universal Serial Bus
- USGS United States Geological Survey
- UWB Ultra-WideBand

- WES Web Enterprise Suite
- WFS Web Feature Service
- WFS-T Web Feature Service - Transactional
- WG Working Group (SWG or DWG)
- WGS84 World Geodetic System 1984
- WiFi Wireless Fidelity
- WPS Web Processing Service
- WatchOS Apple Watch Operating SyStem

Chapter 5. Overview

The SCIRA Pilot includes various stakeholders, applications (App), data clouds and operational field sensors. A schematic overview of the SCIRA Pilot is shown in the following figure (Figure 1):

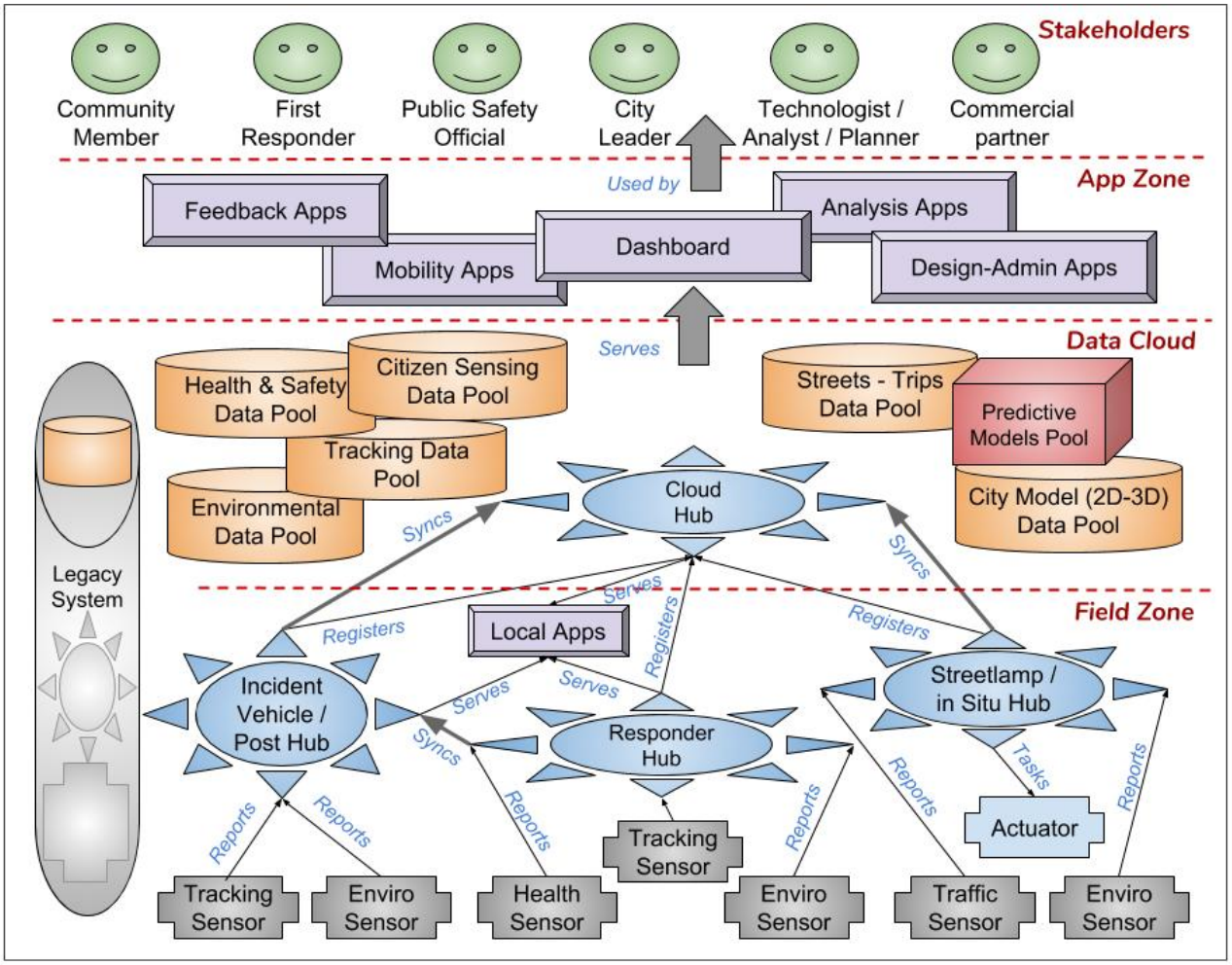

Figure 1. SCIRA Pilot Systems Overview

As shown in the above figure, the SCIRA Pilot systems are roughly divided into four segments:

- **Stakeholders**: Those who set up, oversee, provide data to, and/or make use of applications provided by the prototype systems;
- **App Zone**: Applications, whether desktop, mobile, or browser-based, that are provisioned by system data and services;
- **Data Cloud**: Storage and processing services that maintain an array of data "pools" accessible by the "Cloud Hub" ensemble of OGC Web services;
- **Field Zone**: Mobile, in-situ, and incident nodes that form a data and communications constellation reporting sensor measurements from various sensors and sensor platforms up towards the Data Cloud and provisioning local applications for first responders, commanders, and other field personnel.

The overall prototype system has been divided into six individual systems (listed in Table 1) that, while sharing some components, deliver distinct sets of functionality. In the second column, the list of deliverables from the SCIRA Pilot Call For Participation [https://portal.ogc.org/files/?artifact_id=83373].

Table 2. SCIRA Pilot Systems

System	Deliverables
SmartHub System	• D1: First Responder Tracking Sensor • D2: First Responder Health Sensor • D3: First Responder Environment Sensor • D4: First Responder SmartHub • D5: First Responder Incident SensorHub • D6: Cloud-based SensorHub and Catalog • Related Component: ◦ D13: Command and Communication System

Command Communication System	• D5: First Responder Incident SensorHub
	• D12: 3D Dashboard App
	• D13: Command and Communication System
	• D17: Community Mobility Navigation, Alert and Sensing App
	• Related Components:
	  ◦ D6: Cloud-Based SensorHub and Catalog
	  ◦ D16: Legacy Public Safety Information System Adapter
	  ◦ D20: 3D City Model

Indoor Navigation System	• D8: First Responder Indoor Navigation App • D18: RFID (Radio-Frequency IDentification)/BLE (Bluetooth Low Energy) Indoor Navigation Beacons • Related Component: ◦ D19: 3D Interior Building Model

Street Navigation System	• D15: Traffic Management and Routing Model
	• D17-Road: Community Mobility Navigation, Alert and Sensing App
	• Related Components:
	  ◦ D16: Legacy Public Safety Information System Adapter
	  ◦ D20: 3D City Model

System	Deliverables
Flood Sensing System	• D11/D16: Legacy Stream Sensors • D14: Flood and Inundation Model • D17-Flood: Community Mobility Navigation and Alert and Sensing App • Related Component: ◦ D20: 3D City Model
Road Sensing System	• D10: Smart Traffic/Road Sensor • Related Components: ◦ D11: Smart Weather Sensor ◦ D12: 3D Dashboard

This ER is organized as follows:

- Section 5 (Solutions Architecture) describes the use cases and scenarios, as well as the components developed and implemented for the SCIRA Pilot.
- Section 6 (SmartHub) describes SmartHub systems - supporting, accessing, and connecting wearable sensors with personal applications, as well as incident scene and Emergency Operations Center information resources.
- Section 7 (Command Communication) presents Command Communication systems for a Common Operating Picture (COP), as well as tasking between first responders, incident commanders, city personnel, and city leaders.
- Section 8 (Indoor Navigation) details an Indoor Navigation system for enabling first responders to navigate and be tracked inside of a building in the absence of GPS (Global Positioning System) reception.
- Section 9 (Street Navigation) shows solutions developed for advanced Street Navigation.
- Section 10 (Flood Sensing) discusses both predicted and actual flood inundation regions and street blockages
- Section 11 (Road Sensing) review road condition and traffic awareness for traffic monitoring and as input to response or evacuation routing.
- Section 12 (Findings) briefly discusses the key issues experienced during the SCIRA Pilot implementation. This section also provides a summary of the main findings and recommendations on preferred strategies and change requests.

Chapter 6. Solutions Architecture

The Solutions Architecture describes the use cases, scenarios and interfaces for the components developed and implemented for the SCIRA Pilot. This section provides a reference architecture to guide future implementations of Smart Cities and demonstrate the interoperability achievable using OGC standards. The St. Louis Pilot solution architecture, while specific to the St. Louis, MO municipality, constitutes many stakeholders and human interactions with redundant service implementations to provide sufficient confidence in the standards applied.

6.1. Scenarios

The five scenarios for SCIRA Pilot are listed in the following table (Table 2):

Table 3. SCIRA Pilot Scenarios

Scenario	Description
Scenario 1: River Monitoring	The City needs to monitor the river and anticipate the additional threat of flooding and needs to continue to take preparatory action while also dealing with potential responses related to the pending storm. Focus on River Des Pers and Riverview Dr. & Spring Gardens Area.
Scenario 2: Flash Flooding	Flash flooding has led to a high amount of standing water on streets in and around downtown. These streets are no longer passable and need to be blocked off. The City needs to detect where flash flooding is occurring, the severity, and respond accordingly (i.e. block off flooded streets).
Scenario 3: Assistance to Vulnerable Populations	The flash flooding has created a life and safety risk for vulnerable populations in the areas where the flash flooding is occurring. The health department needs to detect and verify the existence of vulnerable individuals and conduct outreach to get them to safety.
Scenario 4: Building Fire	Unknown to anyone, trenching in the area of T-Rex has damaged the basement and created a crack in the basement wall. Flash flooding and the flooding of trenches has led to flooding in the basement at T-Rex. The flooding in the mechanical room causes a short in the electrical system and starts a fire. Fire, Police, and EMS need to combat the fire and ensure safe and total evacuation of the building.
Scenario 5: Vehicle Accident	A speeding vehicle hits a patch of standing water, hydroplanes and collides with a fire hydrant and as a result knocks over the hydrant. Police, Water, Streets, and EMS need to respond to the incident, render assistance, and deal with the damaged fire hydrant and clear the accident.

The SCIRA Pilot follows a scenario flow as shown in Figure 2.

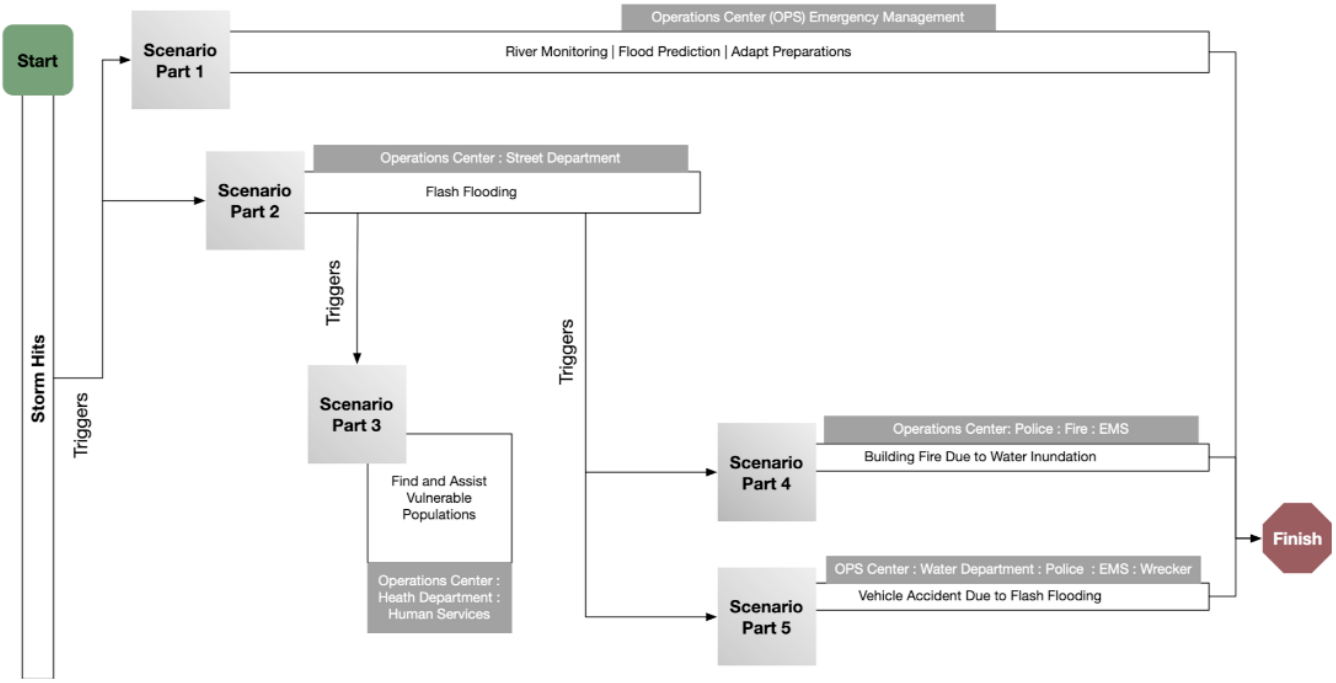

Figure 2. SCIRA Pilot Scenario Flow

The solution architecture is derived from a Master Scenario Event List (MSEL) which is a compilation of timelines and locations for all expected exercise events and actions that describe a scenario. The MSEL as provided in Master Scenario Events List contains the following items for the each scenario:

- Chronological listing that supplements exercise scenario
- Actor
- Location
- Event Description
  - Real or Simulated Event
  - Push or Pull data
  - Originating Device
  - Destination Device
  - Hardware
- Capability Set
- Required Training
- Test Result
- Notes

6.2. Use Case Diagrams

Each MSEL scenario is used to derive a Use Case diagram in Unified Modeling Language (UML). The following UML diagrams (Figure 3, to Figure 7) describe the interactions between the stakeholders and the activities performed using the architecture.

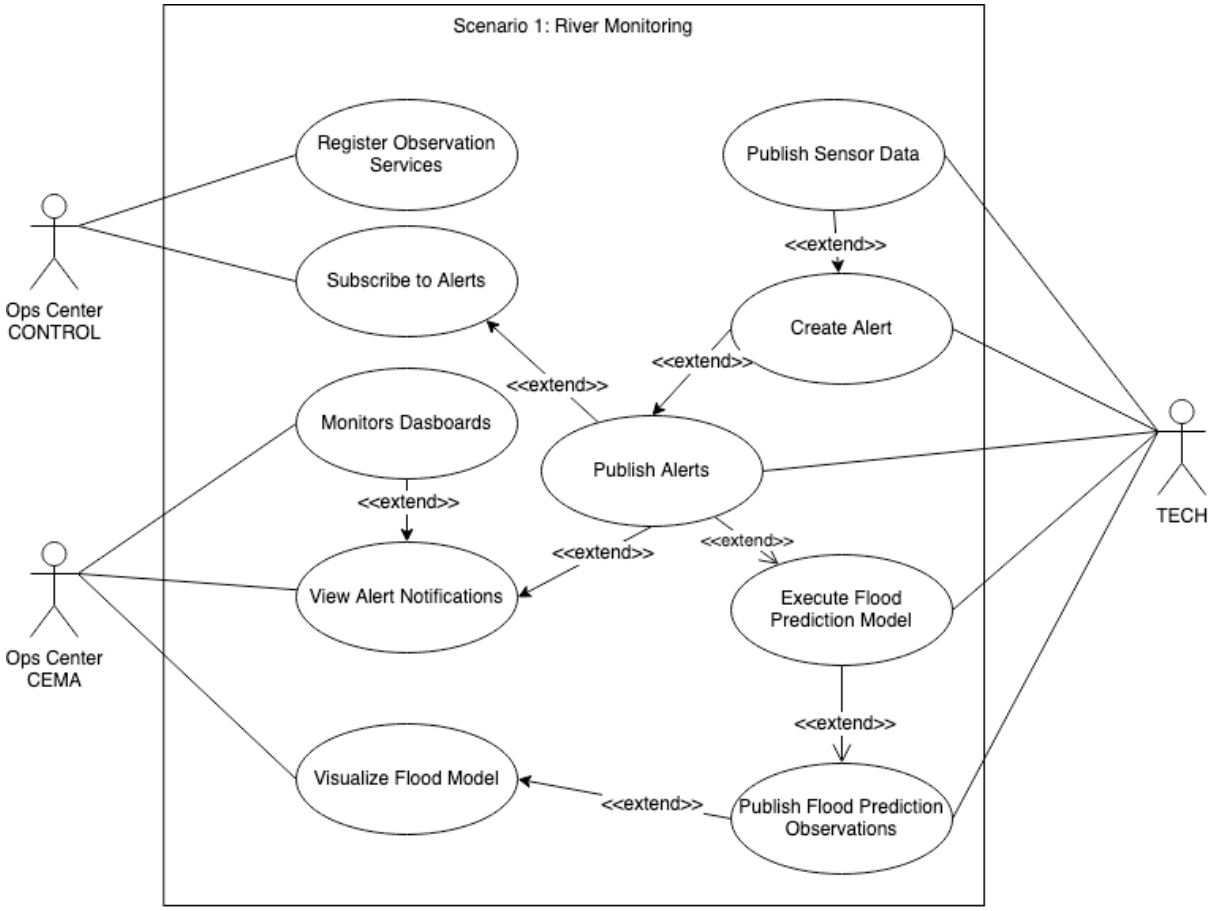

Figure 3. Scenario 1: River Monitoring

Figure 4. Scenario 2: Flash Flooding

Figure 5. Scenario 3: Assist Vulnerable Population

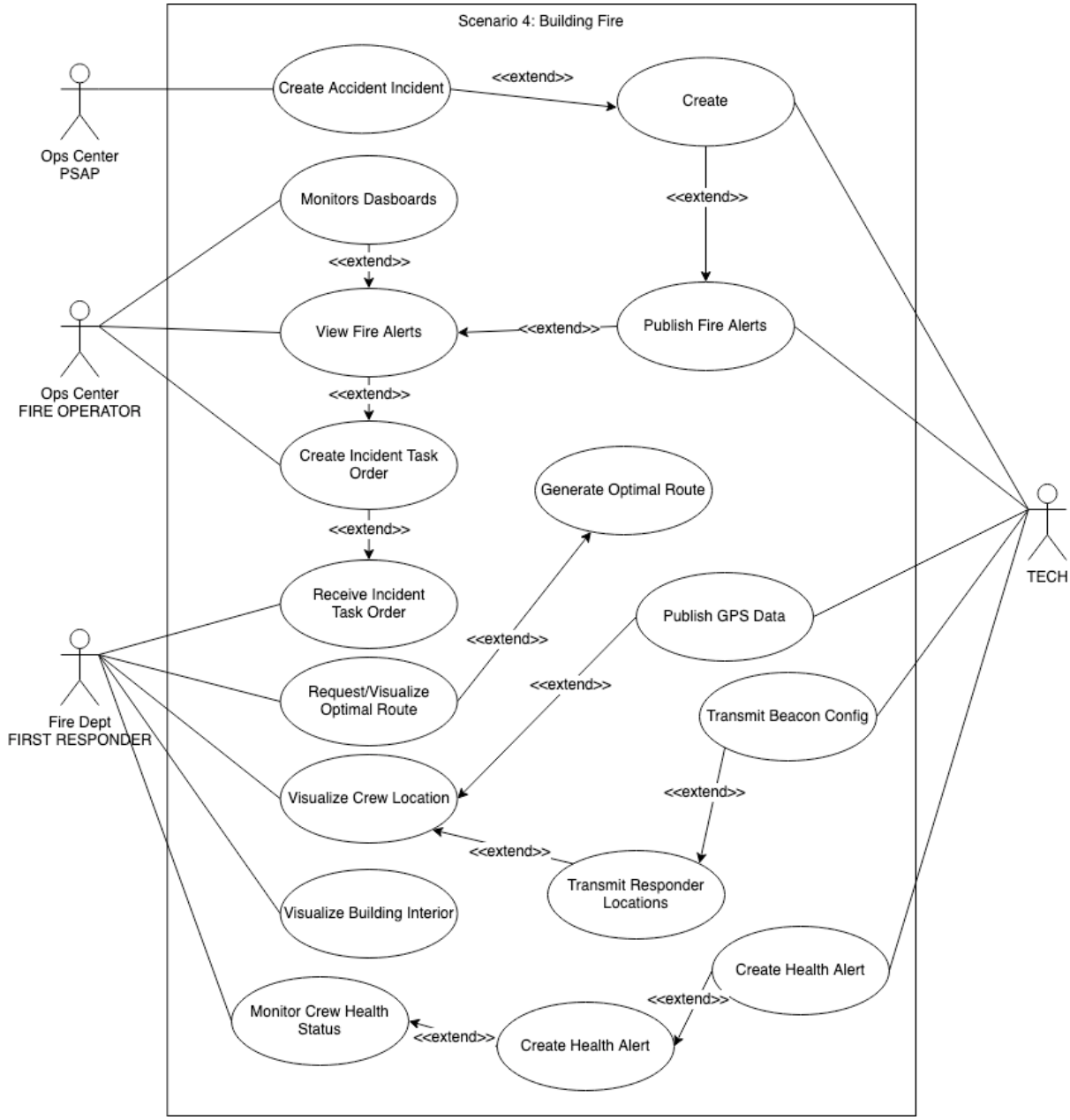

Figure 6. Scenario 4: Building Fire

Figure 7. Scenario 5: Vehicle Accident

6.3. Class Diagrams

Each of the scenarios can be derived into a Class Diagram as shown below (Figure 8, Figure 9, Figure 10, Figure 11, and Figure 12).

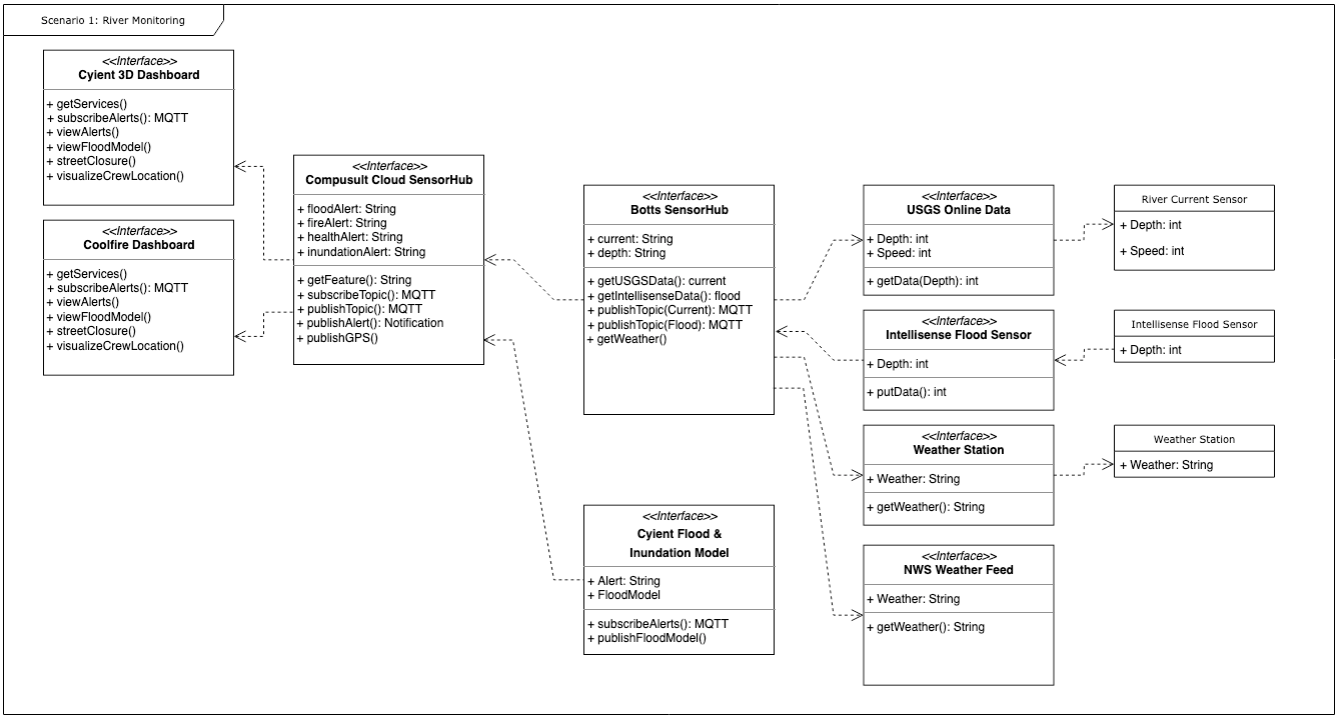

Figure 8. Scenario 1 Class Diagram

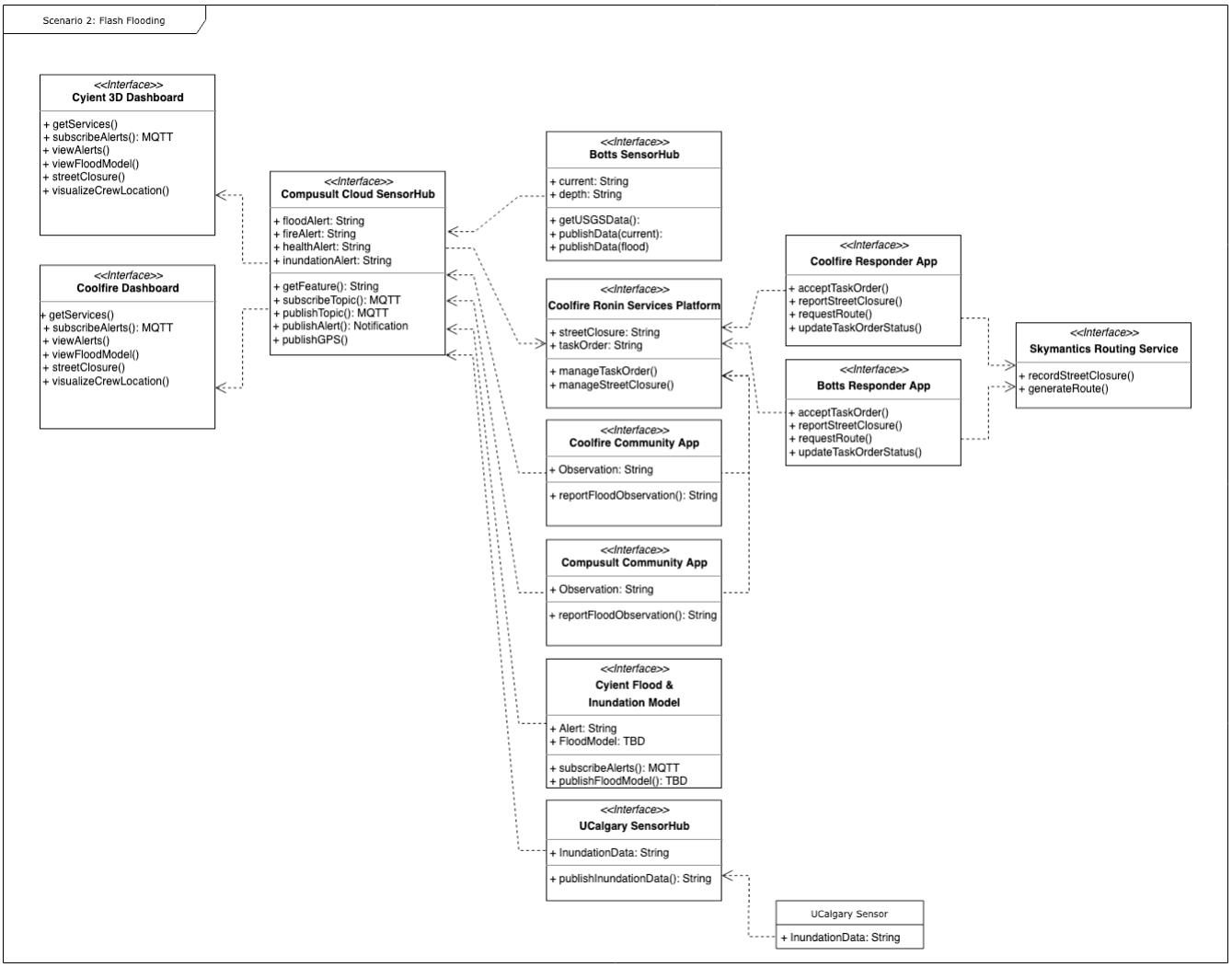

Figure 9. Scenario 2 Class Diagram

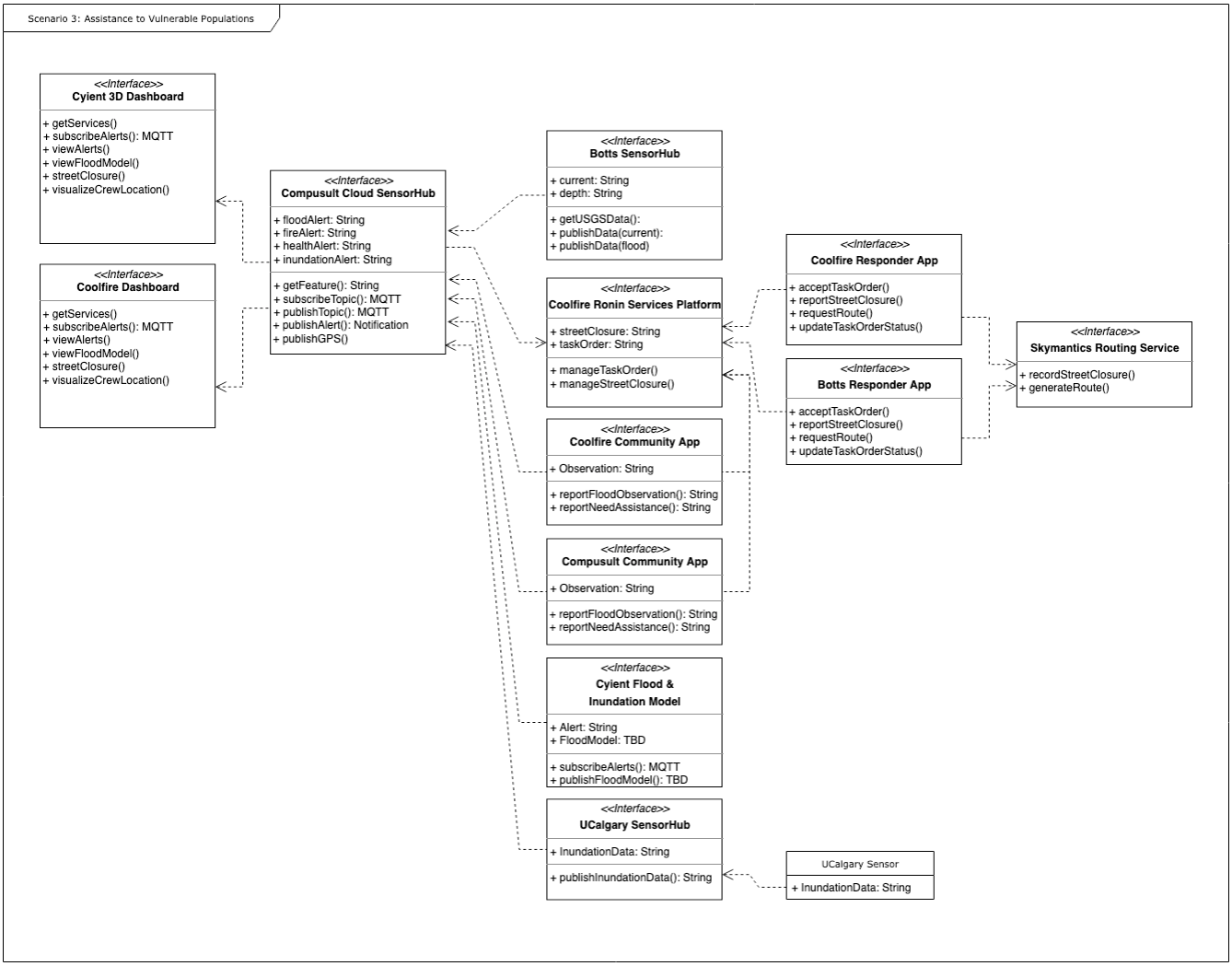

Figure 10. Scenario 3 Class Diagram

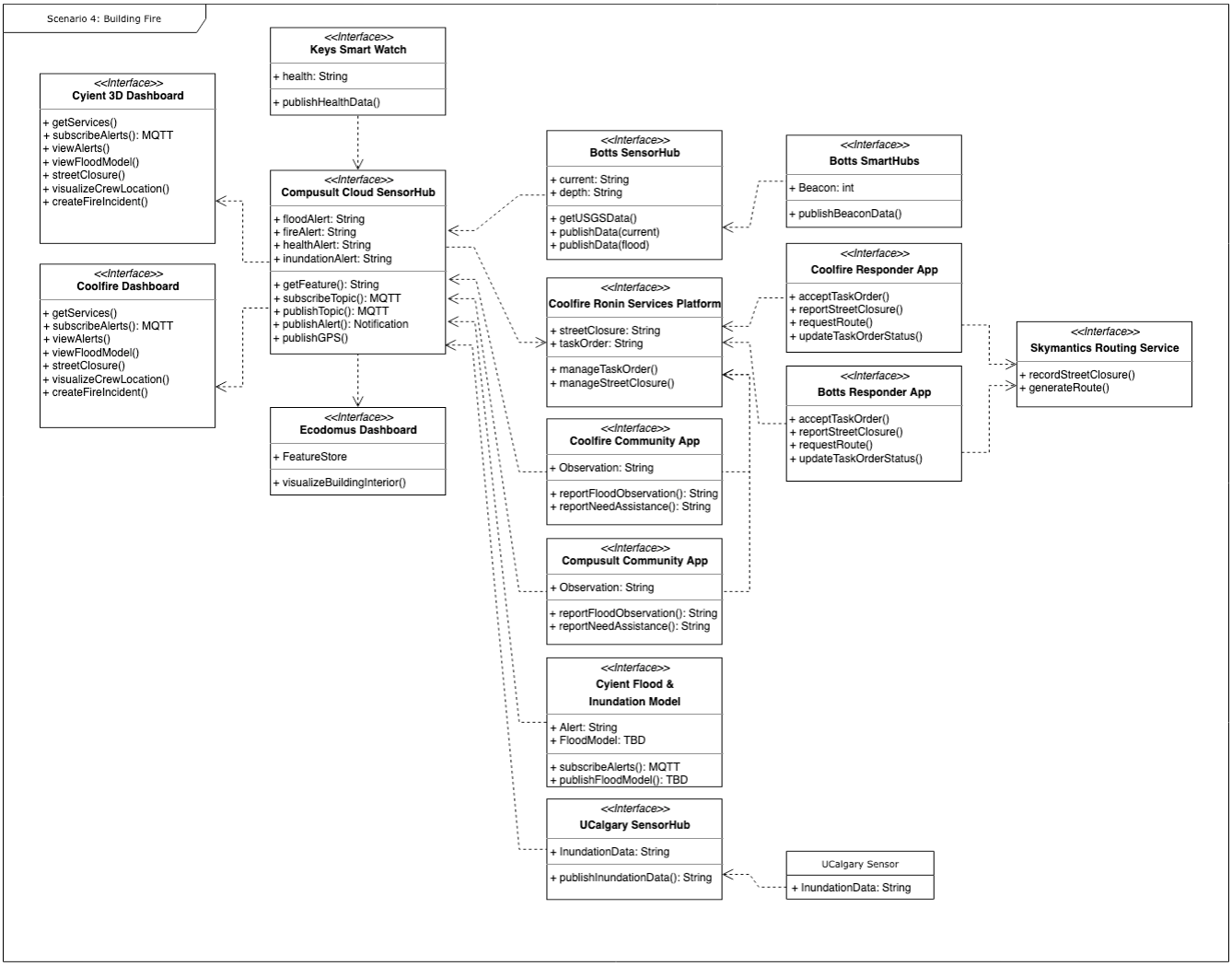

Figure 11. Scenario 4 Class Diagram

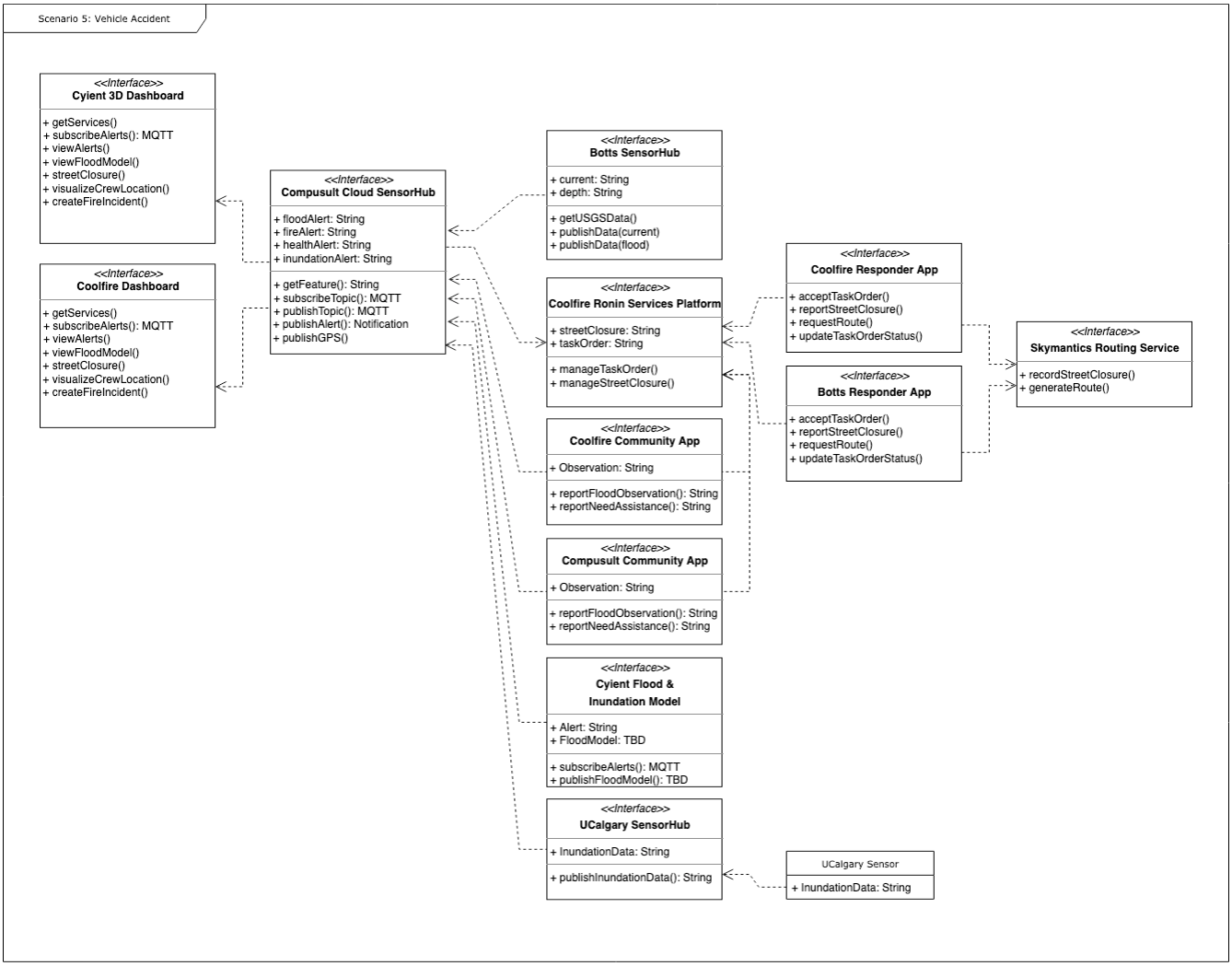

Figure 12. Scenario 5 Class Diagram

6.4. Reference Architecture

The final reference architecture is a generic view of a singular platform of services to perform a set of communications between applications for use in the Smart Cities architecture. For the scope of this Pilot, the reference architecture depicts the minimum subset of components and interfaces needed to fulfill the need of the St. Louis SCIRA Pilot which addresses particularly the emergency management activities of the city (Figure 13). These components can be viewed as being a reference for the implementation of future emergency management components in a Smart City infrastructure for the appropriate use cases described herein.

NOTE Future work could include expanding use cases to include Smart City communications for transportation services, utilities, and other aspects of Smart City architecture.

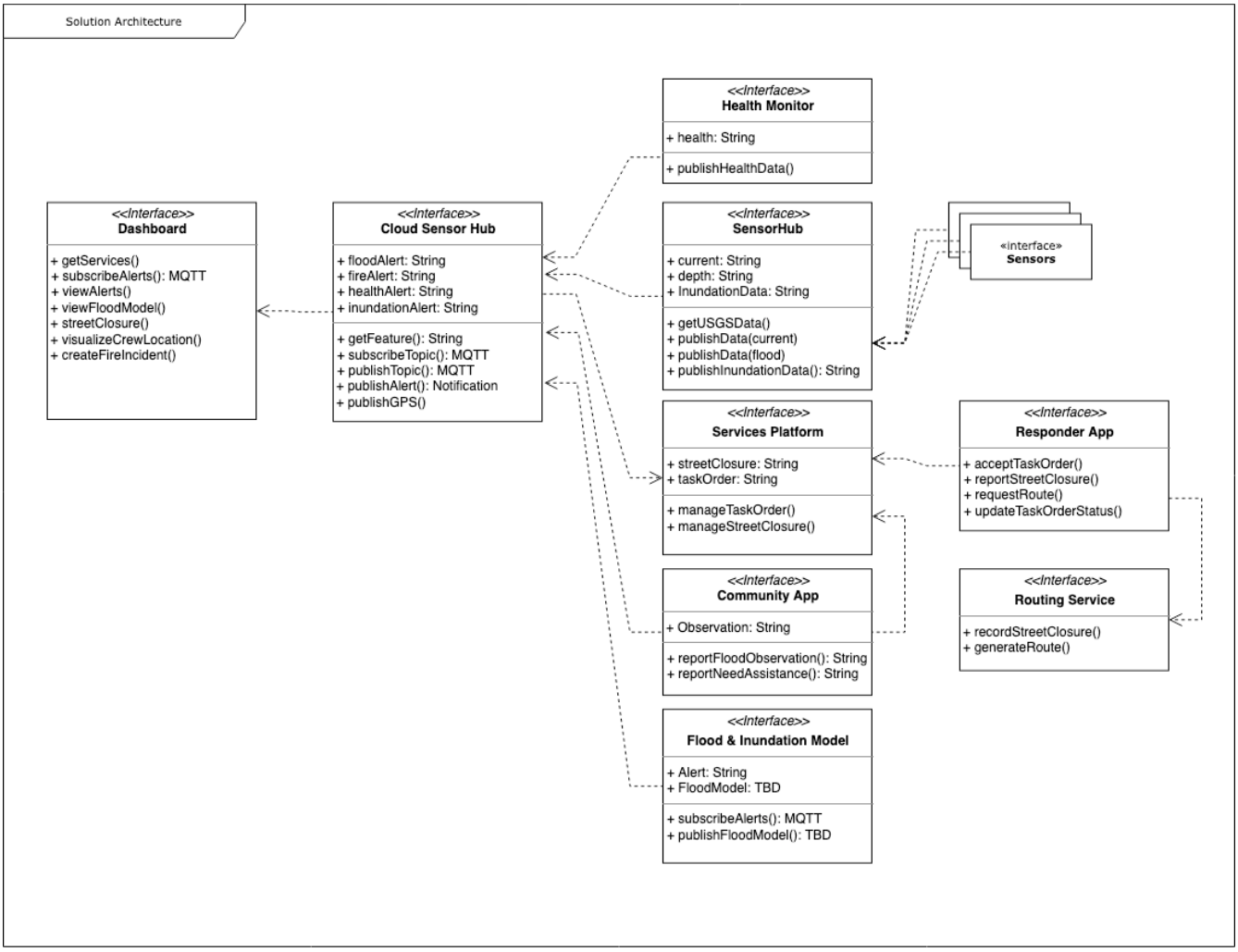

Figure 13. Smart Cities Generic Solution Architecture

Chapter 7. SmartHub System

SmartHub systems center around supporting, accessing, and connecting body-worn sensors with personal applications, as well as incident scene and Emergency Operations Center information resources. Further background on SmartHub systems is found in the Next Generation First Responder (NGFR) [https://www.dhs.gov/science-and-technology/ngfr/handbook] Integration Handbook. The wiring of a SmartHub system is shown in the figure below (Figure 14):

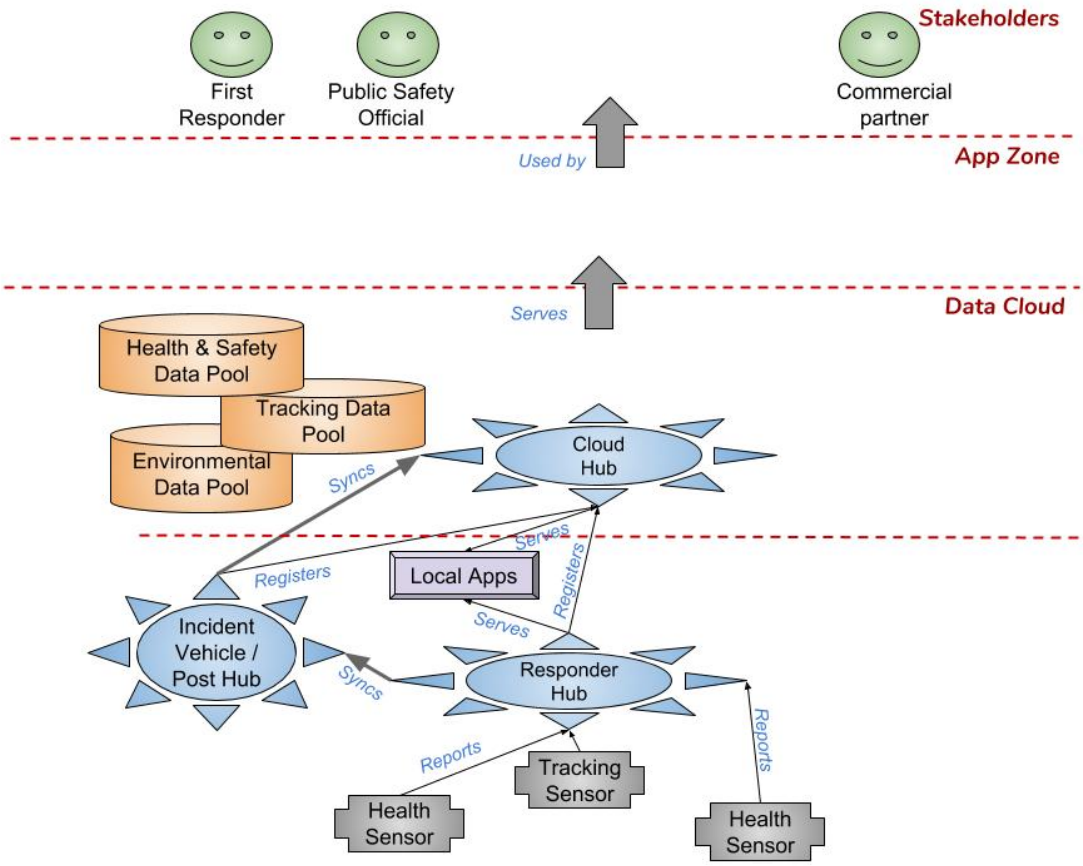

Figure 14. SmartHub System Overview

The OGC® Sensor Web Enablement (SWE) [http://docs.opengeospatial.org/wp/07-165r1/] and OGC® SensorThings API [https://www.ogc.org/standards/sensorthings] (Application Programming Interface) standards were both used in the SCIRA Pilot in support of the SensorHub. These standards provide an interoperable approach in support of virtually any sensor system. In essence, the suite of OGC® SWE and OGC® SensorThings API (STA) standards provide standard means for the discovery of sensors, tasking sensors and actuators, requesting robust descriptions of sensors and sensor observations, and for accessing real-time or archived observations and messages. Those standards can support sensor systems in a Cloud environment, within a field hub, or as part of the sensor itself or on the sensor platform ("on-the-edge").

OGC® SWE and OGC® STA standards can support the insertion of sensors and sensor observations through transactional mechanisms, thereby allowing small apps on the edge to push observations to sensor hubs residing in the cloud or on a field node. Some implementations of the OGC® SWE and OGC® STA standards enable the complete deployment of a SensorHub within any environment. One such implementation is OpenSensorHub (OSH) [https://opensensorhub.org/] an open source software platform supporting OGC® SWE and OGC® SensorThings API standards. As depicted in

the figure below (Figure 15), OSH can be fully deployed at all levels of the sensor environment, from the sensor or sensor platform itself to Android phones and tablets, to local workstations, and to the Cloud. This means that all functionality of the OSH platform, from storage, processing, real-time streaming of observations, and tasking of sensors can be supported on any of these devices.

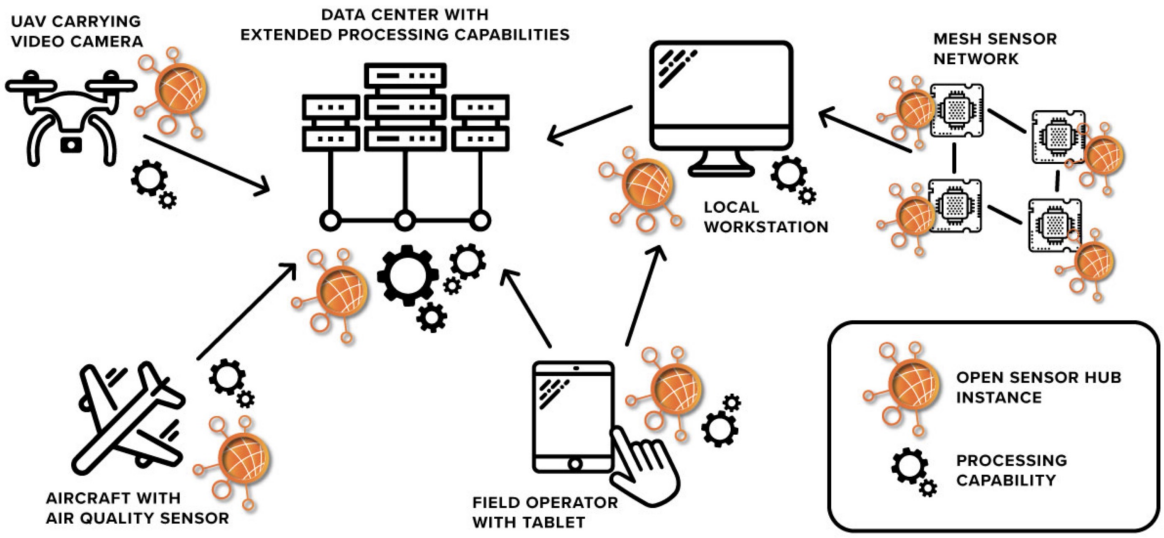

Figure 15. OpenSensorHub (OSH) Cross-Platform Functionality

Within the DHS domain, the SensorHub domain is often divided into three main components:

1) First Responder *SmartHub*: the personal mobile sensor hub, sometimes referred to as a *SmartHub*;

2) First Responder Incident SensorHub: the incident field hub domain supported by perhaps a laptop or desktop within a local network;

3) Cloud-based SensorHub & Catalog: the enterprise-level deployed in the Cloud.

Often there are restrictions in the network bandwidth within these three domains such that Cloud deployments support the widest breadth of connectivity and available resources. The incident field hubs support a local network shared with local devices (sensors, databases, and *etc.*) but may have limits to its connectivity to Cloud resources. Finally, the personal *SmartHub* may be body-worn, support sensors that are personal to the wearer (*e.g.*, location, heartbeat, body-cam, and *etc.*), and may be limited in connectivity with a local team or squad. In some cases, the amount of data exchanged in real-time between these three domains may be limited, and may be configured to get only the most relevant data up to the Cloud or incident hub. Other times, connectivity may not be a challenge at all, allowing for complete exchange of observations between any parties who desire it.

7.1. D1: First Responder Tracking Sensor

A basic but important capability for monitoring First Responders and other assets is to be able to access and display their locations in real-time. A SensorHub treats *location* as simply another observation derived from the GPS, beacon, Laser RangeFinder sensors or other positioning devices. First Responder tracking involves tracking the vehicle carrying the First Responders or tracking the location and movement of the First Responder themselves. In either case, tracking locations outside

typically involves GPS sensors. When the First Responder moves inside a building, GPS often fails or becomes highly inaccurate.

An ideal device for reporting First Responder location is a smartphone (Figure 16 and Figure 17). The smartphone not only provides GPS-based location but also provides geospatial orientation using a combination of a geomagnetic compass and accelerometer. In addition, the smartphone provides a ready-means to push locations to a remote server through its LTE (Long-Term Evolution) or WiFi (Wireless Fidelity) connection. Several First Responder Tracking approaches were used for this exercise including pushing locations to the *CoolFire* Ronin platform, as well as pushing locations from an Apple Watch and an OSH *SmartHub* to the *Compusult SensorHub* in the Cloud.

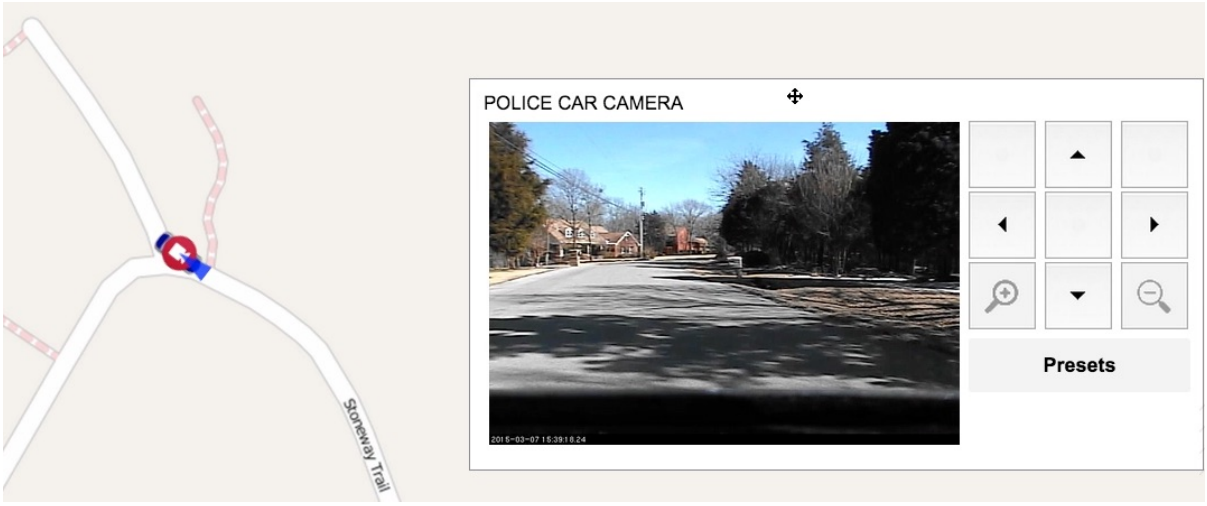

Figure 16. Police Car Tracking using GPS with Android Smartphone Running OSH

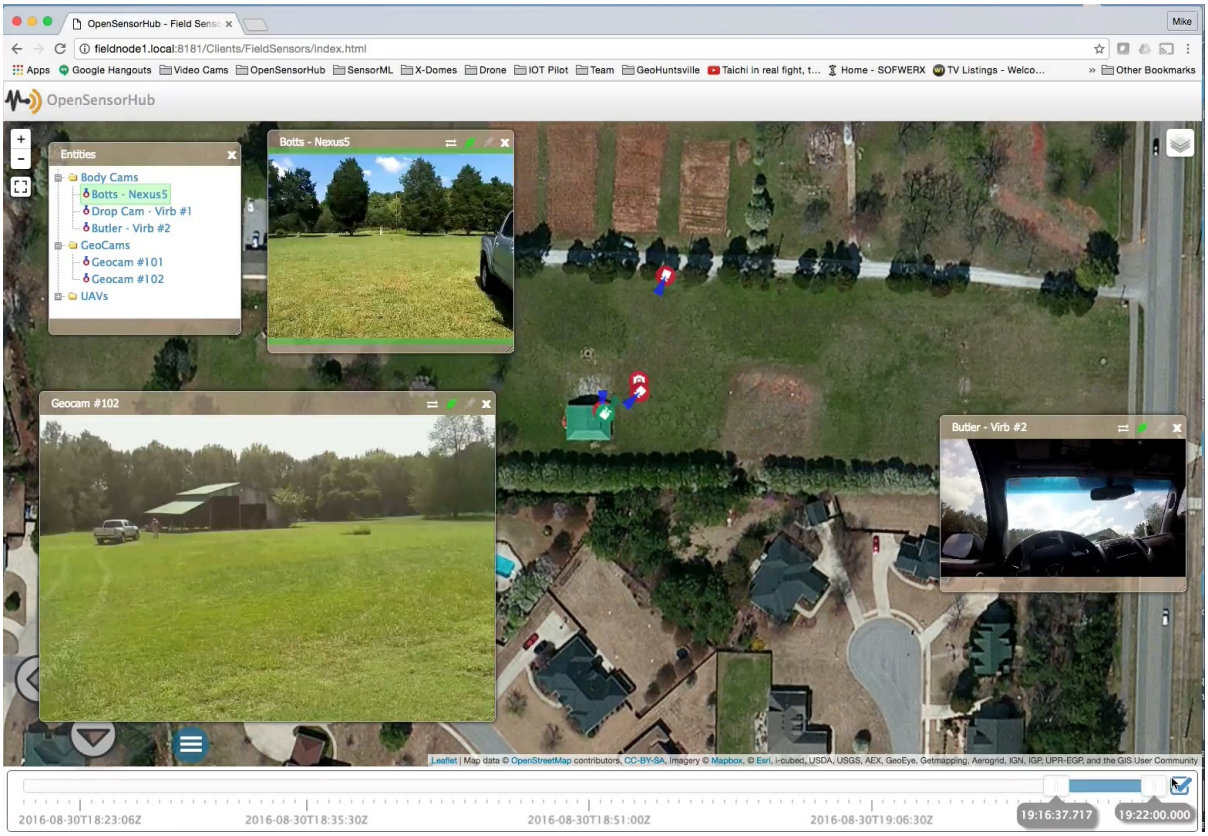

Figure 17. Tracking Vehicles and Personnel with Android Smartphone Running OSH

The SCIRA Pilot incorporated three sensor systems for tracking the location of First Responders:

- ***Botts Innovative Research* Indoor Navigation**: *Botts Innovative Research*'s solution focused on indoor navigation, utilizing indoor beacons detected by SONIM [https://www.sonimtech.com/products/devices/] phones issued to the First Responders to derive and communicate location. Please see the Indoor Navigation section for more details;
- ***keys App* (Apple Watch)**: *keys*' solution reported position data utilizing the GPS information provided by Apple Watches worn by the First Responders - a mixture of both watches issued to the First Responders, as well as watches already owned by First Responders (representing a Bring-Your-Own-Device (BYOD) deployment model). Please see the First Responder Health Sensor section for more details;
- **Coolfire Mobile Response**: Coolfire's solution reported position data utilizing the GPS information provided by SONIM phones issued to the First Responders. See the Command & Communication section for details.

7.2. D2: First Responder Health Sensor

In Public Safety incident scenarios, knowing the status of your most important asset - the First Responder - is vital for implementing an effective, efficient, and safe response. In addition to location, awareness of the current health of First Responders while performing their duties could save lives, helping identify a First Responder that is in distress, that has fallen, or even that has had a heart attack.

7.2.1. keys App (Apple Watch)

keys' solution reported heart-rate data and health incident alerts utilizing the heart-rate information provided by Apple Watches (Figure 18). These Watches worn by the First Responders in the SCIRA Pilot exercise- a mixture of two watches issued to the First Responders, as well as watches already owned by First Responders (representing a BYOD deployment model).

Figure 18. keys App on WatchOS (Apple Watch Operating SyStem) with a High Heart Rate Alert

The best camera/sensor/platform is the one you always have with you – using the *keys* App, First Responders & Incident Commanders can leverage the devices they already carry to provide location, heart rate, and other sensor data into their current operational systems, as well as receive

incident alerts (visual/audio/vibration), select sensor data, and an incident/team map – all from their smartwatch via FirstNet [https://www.firstnet.com/devices/phones.html]-compatible phones, LTE or Incident Area Networks.

keys App Platform - Consumer Hardware & Frameworks

The *keys* App is a software-based solution that utilizes mass-market consumer wearable computing and biometric platforms (such as the Apple Watch) to both send and receive location, sensor data, and alert notifications.

keys App Deployment

The *keys* App allows for both organization-provided and user-provided (BYOD) implementations.

keys App Communication

The *keys* App leverages the network capabilities of its hardware platform (Apple Watch) to both send and receive sensor data and messaging/alerts (Figure 19). This includes BLE (connecting via its paired iPhone), WiFi (Incident Area Network, Vehicle or First Responder Hotspot, and *etc.*), LTE (Apple Watches with LTE service), and FirstNet (Apple Watch with service or via paired iPhone). The communication channel is chosen automatically based on both network availability and power consumption costs (BLE is much less "expensive" than WiFi, which in turn is less expensive than LTE).

Figure 19. keys App Communication Networking - BLE WiFi LTE FirstNet

keys App Interoperability

The *keys* App utilizes non-proprietary, platform-independent, and royalty-free open standards (OGC® SensorThings API [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html], MQTT [https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html] (Message Queuing Telemetry Transport), GeoJSON [https://tools.ietf.org/html/rfc7946] (Geo JavaScript Object Notation) and *etc.*) for communication, to enhance interoperability and compatibility, and reduce costs.

keys App Pilot Instantiation

For the final Pilot exercise, the *keys* App published location, heart-rate and health alert data to the Compusult Cloud-based SensorHub & Catalog using the OGC® SensorThings API [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html] (with Pilot-specific additions as specified in the Cloud SensorHub section). Health data was able to be visualized within the primary pilot scenario dashboards and datastreams (Figure 20).

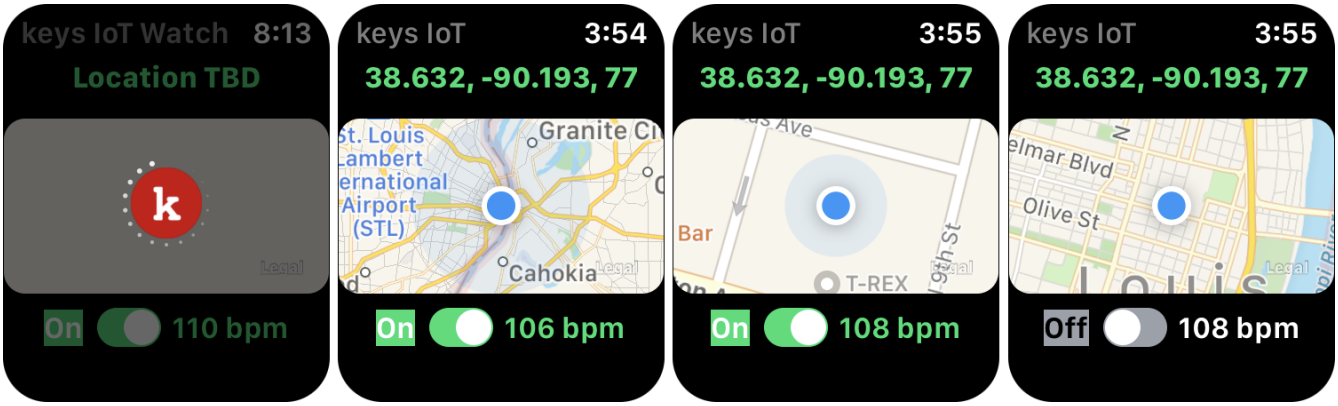

Figure 20. keys App Apple Watch Interface and Zoom Levels

In addition, all data published to the Cloud SensorHub also was able to be visualized in the hub's standard OGC® SensorThings [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html] data visualization tools, as shown in the figure below (Figure 21).

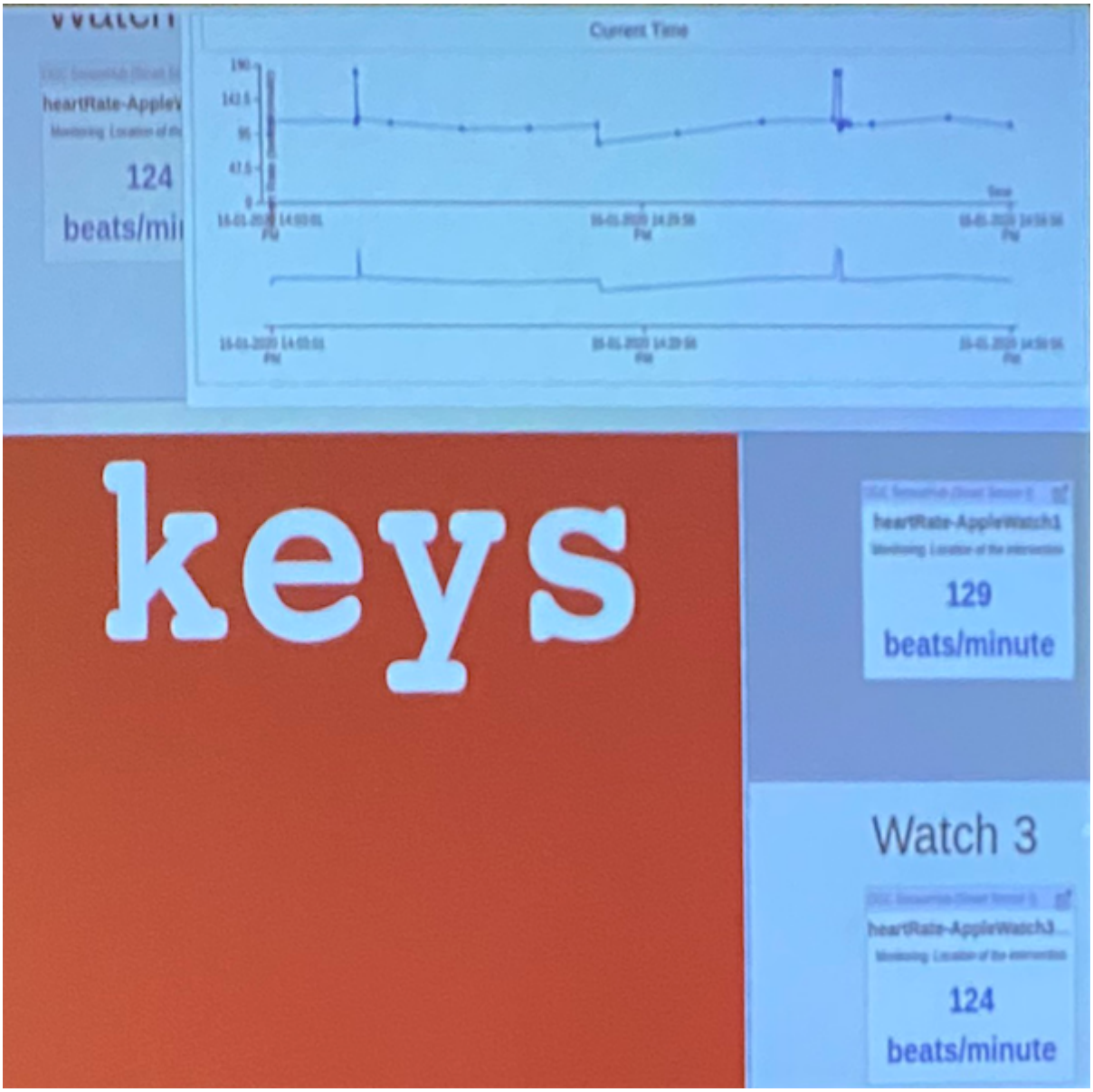

Figure 21. keys App Watch Data Visualized on Compusult Cloud SensorHub

Also to note, development and integration coordination was performed with *Botts Innovative Research* for possible sensor publishing to the First Responder *SmartHub* as well as for the *keys* App Apple Watches to assist/act as an additional Indoor Navigation Beacon sensor for their implementation. The final integration and exercise provided insight for potential future integration of these capabilities.

7.3. D3: First Responder Environment Sensor

During a response to an incident, the responder may depend on warnings and situation awareness provided by a collection of sensors. Some, such as monitors for heartbeat, temperature, O2, and the presence of toxins in the atmosphere might be monitored by body carried sensors. Others, such as a local mobile weather station, a small surveillance drone, shot detection sensors, drop cams, or motion detectors (*e.g.*, left in cleared areas following a breach), may be portable but not body-worn sensors. Still, all of these sensors should be managed by a SensorHub that complies with OGC standards, whether a *SmartHub* or Incident Field Hub (Figure 22).

Figure 22. Unmanned Arial Vehicle Monitoring a Simulated Incident

7.4. D4: First Responder *SmartHub*

The First Responder *SmartHub* serves multipurpose roles. A smartphone with OSH installed serves as an excellent *SmartHub*. It can fully support the management of local sensor observations while being able to push observations to a field or cloud-based SensoHub (using LTE or WiFi), it has on-board sensors providing geolocation, geo-orientation, and High-Definition (HD) video, it can connect through Bluetooth or WiFi with other helpful sensors (*e.g.*, Laser Rangefinder, FLIR (Forward Looking Infrared Radar), throwable camera for breaching risky environments), it can provide digital forms for inputting incidence reports, and it can provide a geospatial mapping environment for visualization of single sensor display or a 3D Common Operating Picture (COP).

- ***SmartHub* Providing Situation Awareness**: The figures below (Figure 23), demonstrates the use of a *SmartHub* to not only provide location awareness to the First Responder who is wearing it (Figure 24), but also enables the real-time streaming of the First Responders location, orientation, and video to any authorized viewer such as the incident commander.

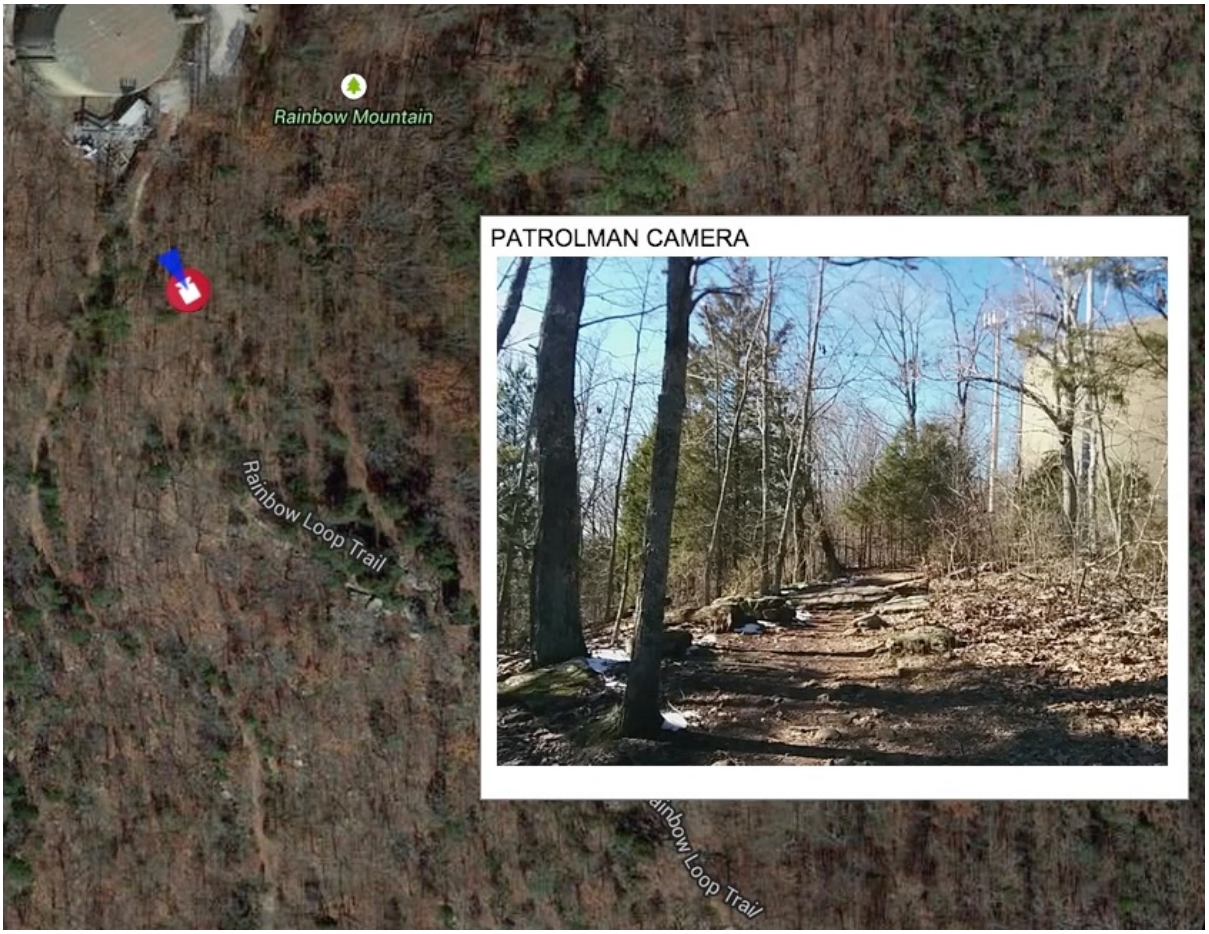

Figure 23. OSH-enabled SmartHub Streaming Location, Orientation, and Video

Figure 24. Body-Mounted SmartHub Worn on Chest

A laser RangeFinder connected to *SmartHub* through bluetooth allows one to tag distant or risky locations for transmission of situation awareness, or to be used to task a camera or drone to look at or fly to a remote location. This also provides one of many examples of the importance of incorporating on-demand processing within the *SmartHub*, as such a process chain combines the GPS location from the phone and the measurements from the Laser RangeFinder and determines the target geolocation (Figure 25).

Figure 25. Laser RangeFinder Connected to SmartHub

- ***SmartHub* Providing Spot Reports**: The *SmartHub* provides an excellent environment for enabling the entry and publishing of time and location-tagged incident reports (Figure 26). These reports, once entered, become observations managed by the *SmartHub* and can be configured to be pushed to other SensorHub nodes or to a message broker such as one that implements the MQTT standard.

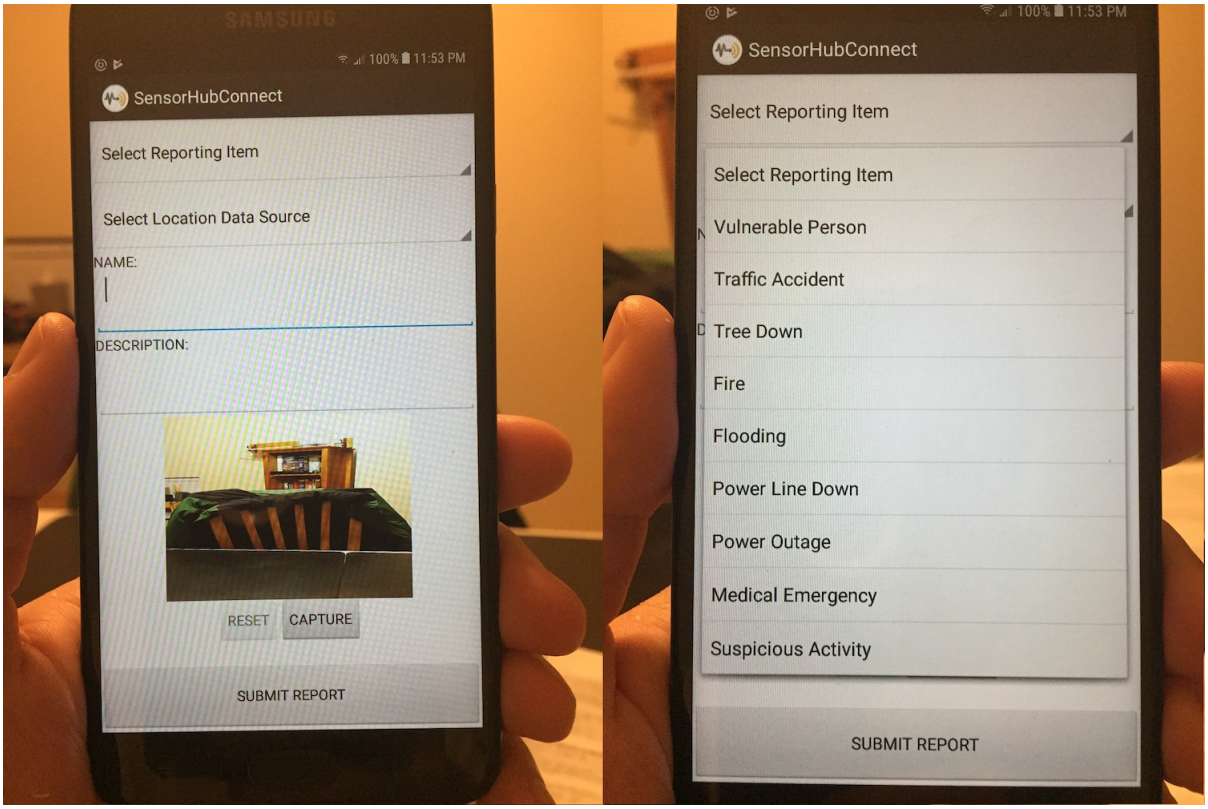

Figure 26. OSH Spot Report - Used to capture an incident or event

The SmartHub provides the ability to tag the location of the incident using the phone's GPS, the location of a Laser RangeFinder target, or the location of a BLE beacon (Figure 27).

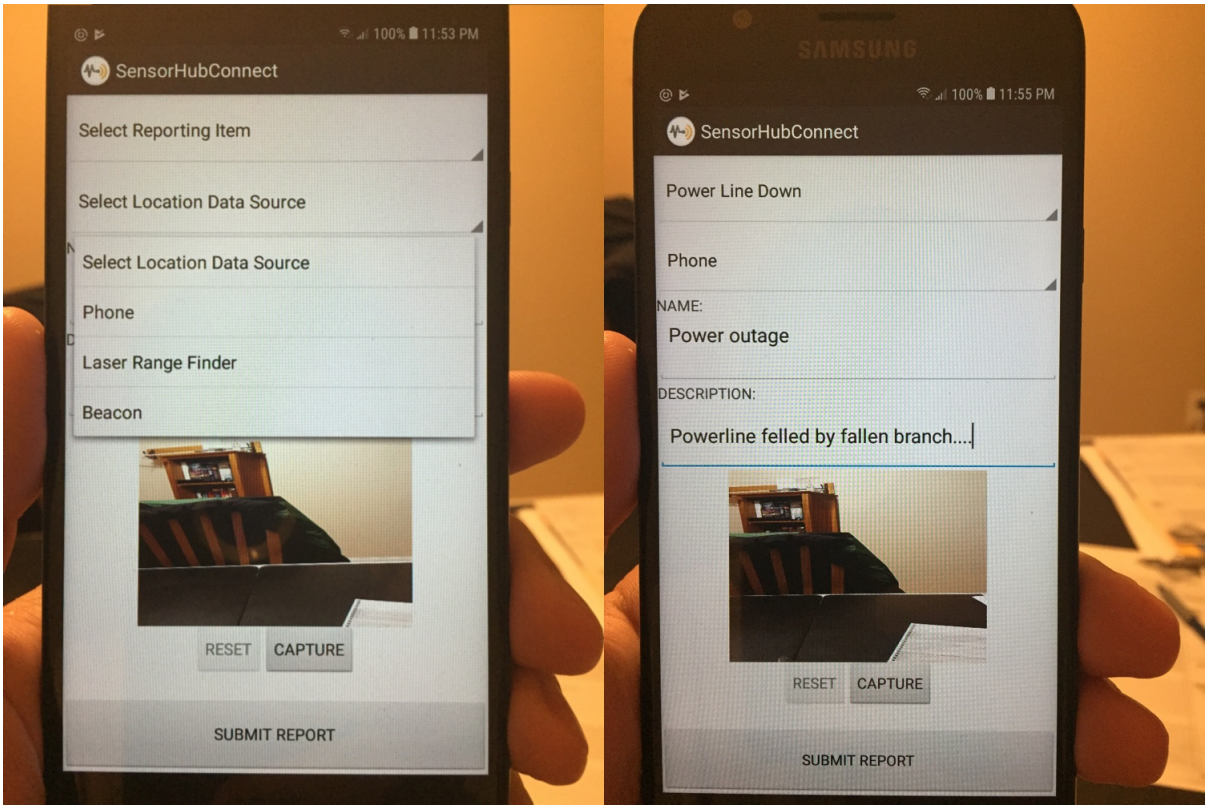

Figure 27. OSH Spot Report - Pick Location Source, Take Image and Submit

- ***SmartHub* Providing Geospatial Mapping.** The *SmartHub* device can also serve as a visualization display for mapping and displaying any desired observations managed by the local or other SensorHubs. The figures below (Figure 28) show the ability to access the AVL

(Automatic Vehicle Location) and traffic cameras from NYS (New York State) SensorHubs to locate a particular dump truck and perhaps verify its location on a traffic cam (Figure 29).

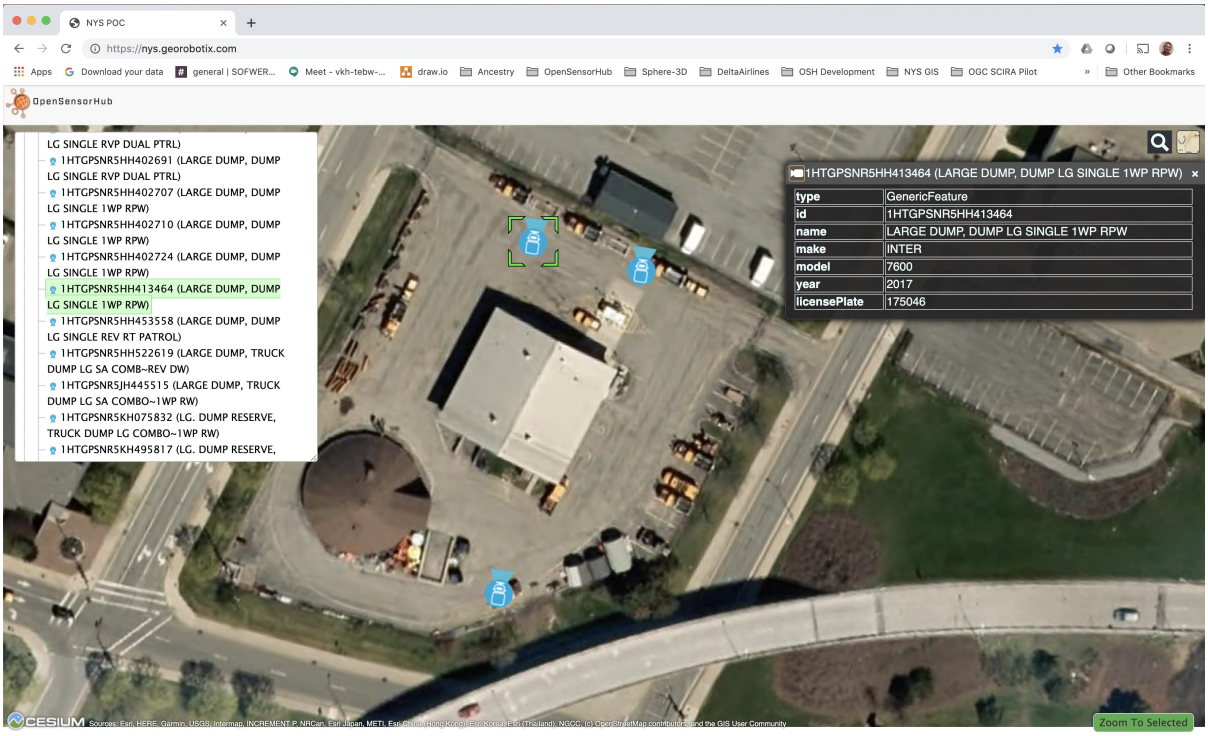

Figure 28. NYS Dump Truck Located at Facility

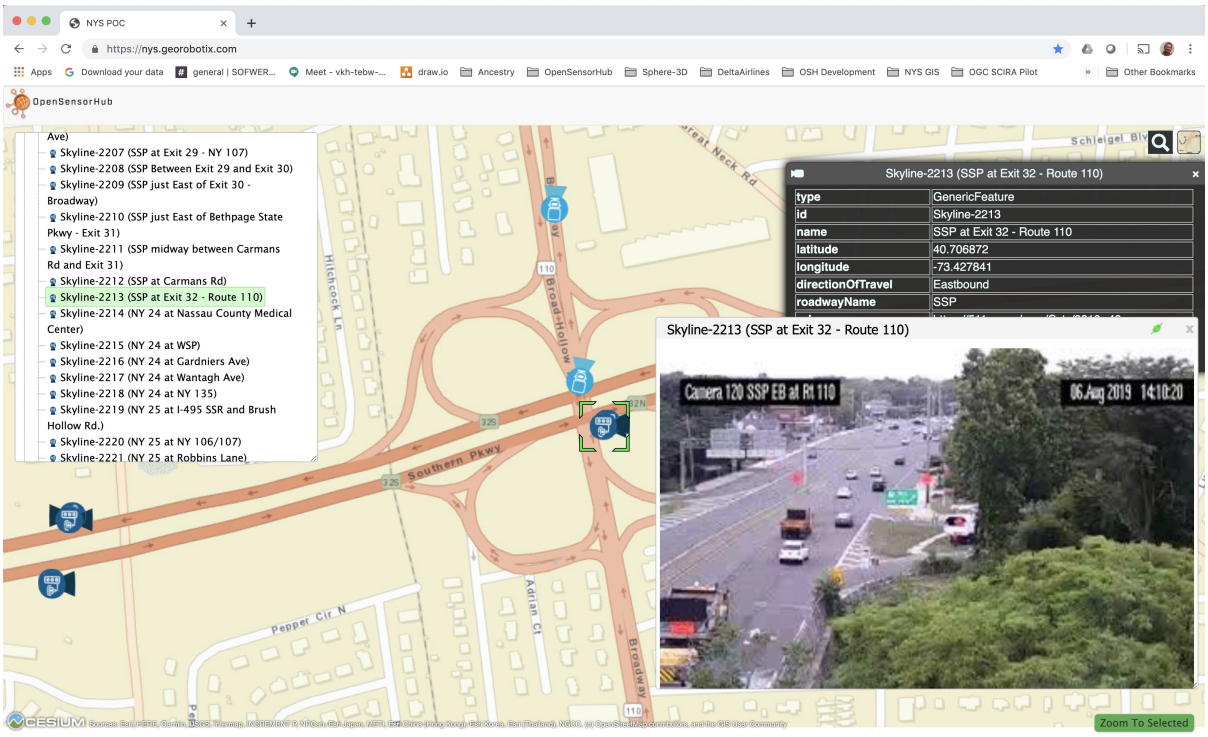

Figure 29. NYS Dump Truck Located Driving Past a Traffic Camera

In addition to accessing sensor data, the *SmartHub* can also access and display routing information to direct the city operators to the next incident. In the figures below (Figure 30 and Figure 31), the routing requested from the Skymantics Web Processing Service (WPS). This routing information could be combined in a COP with sensor data such as the location of other city vehicles or hazards such as flooded streets.

Figure 30. Routing Display for Road Information

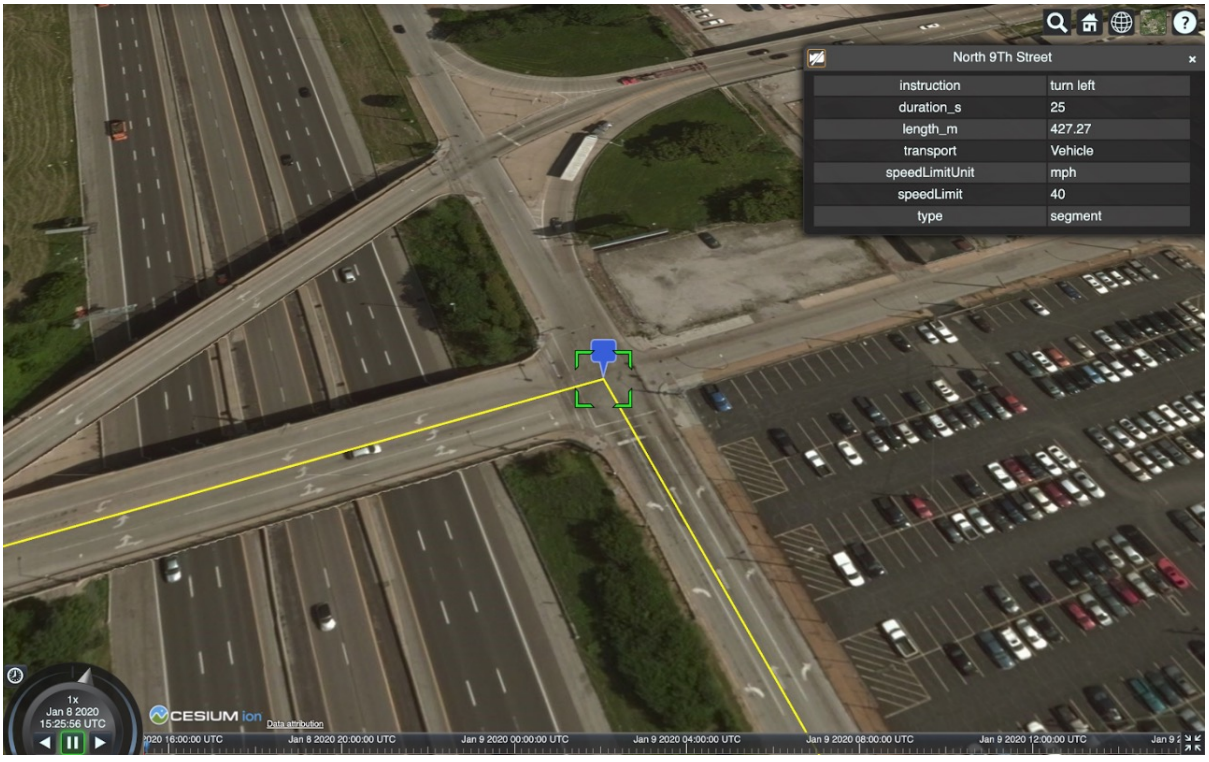

Figure 31. Routing Display for Direction to Turn

7.5. D5: First Responder Incident SensorHub

The First Responder Incident SensorHub was eliminated from the scenario, but as discussed previously, it serves as a local SensorHub that maintains connection to all sensors within a given geospatial area of a particular incident. It stores the local sensor observations and makes them available to any authorized First Responder within its local network. If so configured, the Incident SensorHub can also push observations up to the Cloud SensorHub for more general access and

monitoring.

7.6. D6: Cloud-Based SensorHub and Catalog

Compusult follows the OGC® SensorThings API [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html] (v1.0) Standard to push and pull data to/from SensorHub. In the sections below the 'push' and 'pull' functions are described, links to the API are provided, as well as a few examples of each method.

The catalog service, based on the OGC® Catalog Services for the Web (CSW) [https://www.ogc.org/standards/cat] standard, allows external clients to publish and discover SensorHubs, *SmartHubs* and other content. The CSW catalog service provides a standard-based data access interface to Incident SensorHubs and *SmartHubs*, other data pools, and model processing capabilities. The catalog service also provides access to legacy data zone systems equipped with their own OGC interfaces.

SmartBase SensorHub (Figure 32) serves as a centralized cloud platform where all participant hubs communicate with each other.

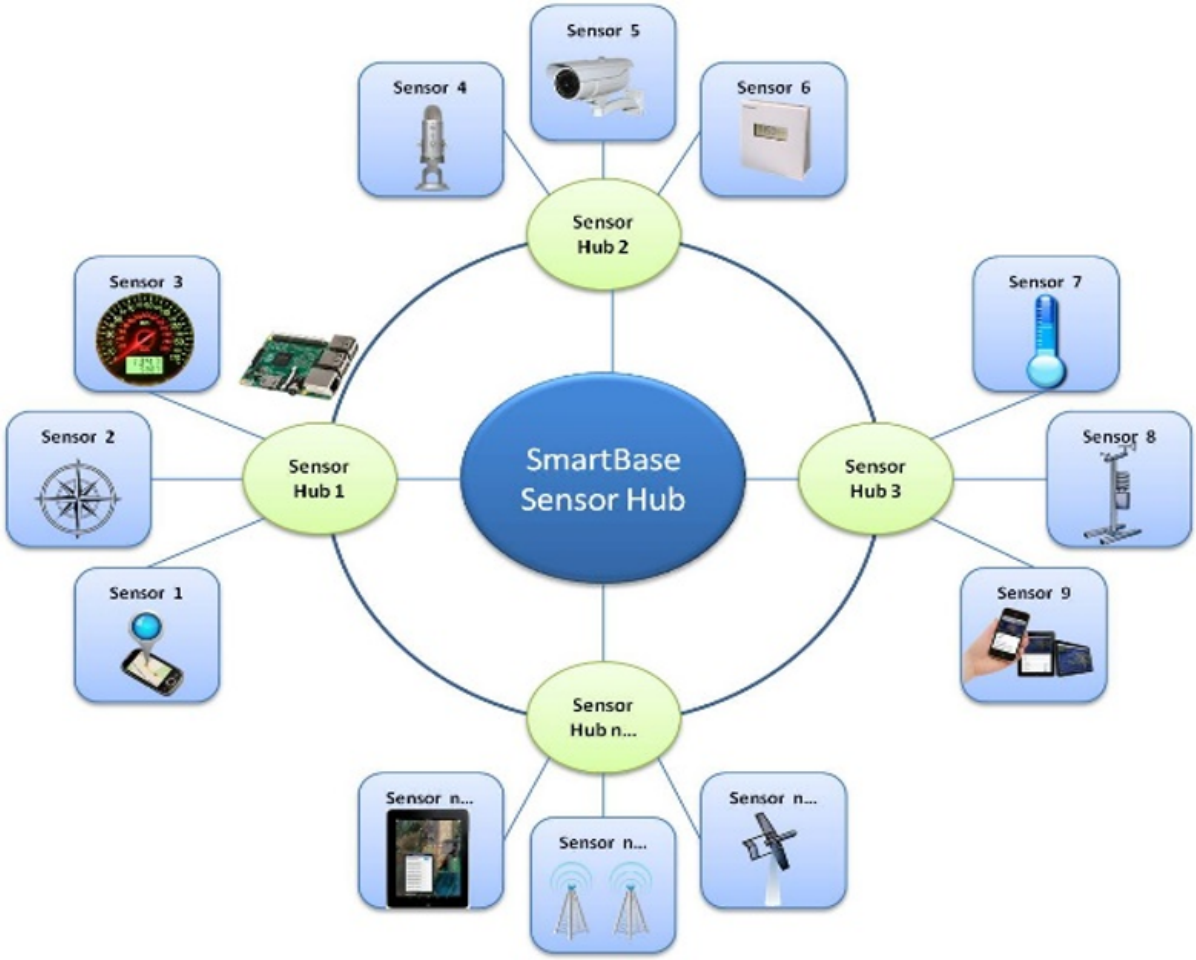

Figure 32. SmartHub Overview

Response teams made use of modern sensing and communications technology to achieve situational awareness, provide the ability to task sensors and create alerts. An alert example would be setting up a notification if a heart rate sensor indicates a drop below a certain threshold. The alerting system would then send a notification to subscribers in the form of an email or text message.

SensorHub supports the OGC® SensorThings API [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html] and MQTT [https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html]. End users are required to authenticate with SensorHub to allow them to register, discover, access, and visualize Incident SensorHubs & *SmartHubs*, access sensor data and other data pools, monitor and manage sensor readings, create tasks and alerts, view sensor locations and historical data graphs/lists, and combine data from multiple sources.

SensorHub's MQTT broker was used as a central location that multiple SCIRA participants could connect their MQTT clients to. The participants could then push data directly to the OGC® SensorThings API [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html] Observations topic and have it displayed on-demand on both SensorHub dashboards and in the MQTT client windows of other subscribers.

SensorHub's Tasking and Alerting module was used as an originator of alert messages. *Compusult* conferred with SCIRA partners to set up rules that would trigger when conditions were anomalous (*e.g.*, when flood waters exceeded a certain height). When they did, they automatically pushed MQTT warning messages to predefined topics, and any subscribed clients received them on an event-driven basis (Figure 33).

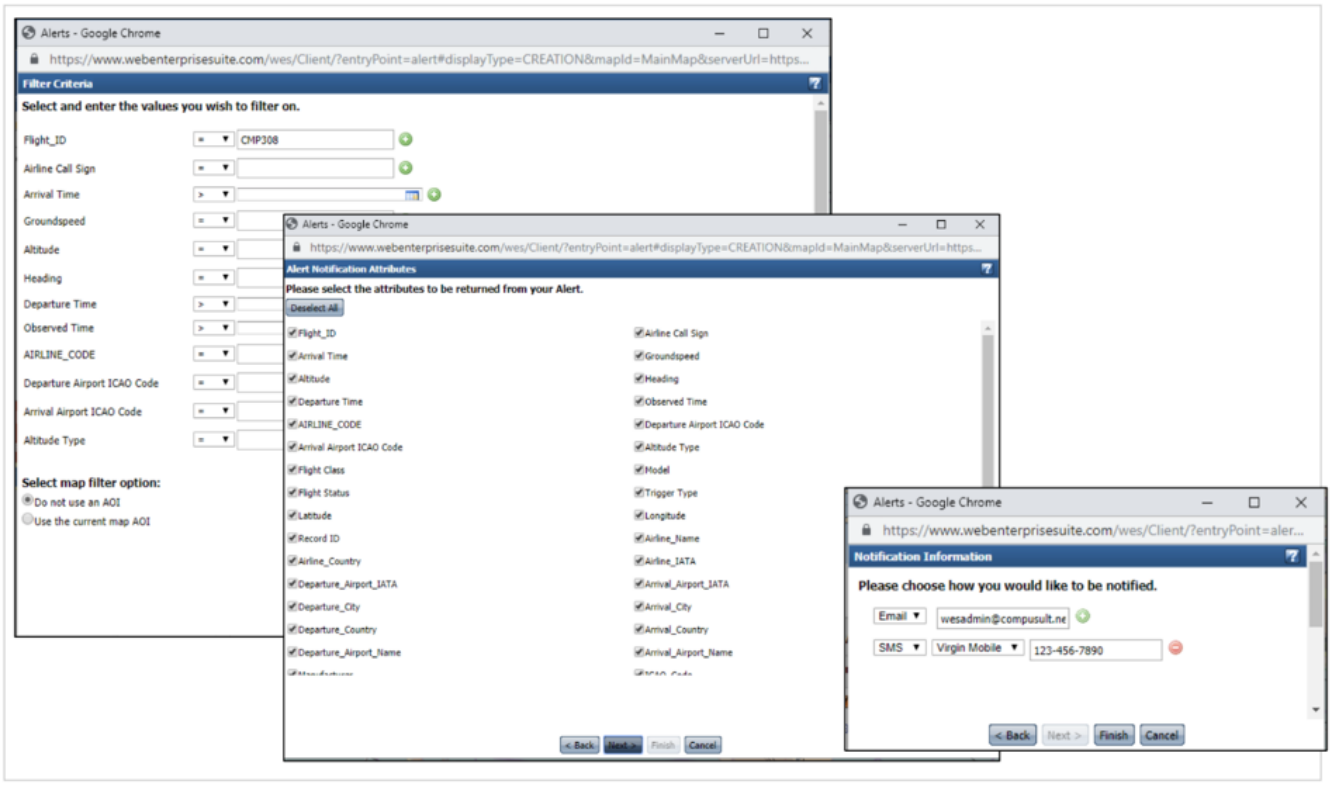

Figure 33. SmartHub Alerting System

The SensorHub MQTT service is protected through secure access control. Each SCIRA participant was given user login credentials for SensorHub, which they had to input into their MQTT clients to access the service.

7.6.1. Pulling Data from SensorHub

HyperText Transfer Protocol(HTTP) GET and MQTT can be used to pull existing data (Things, Datastreams or Observation) from the OGC® SensorThings API [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html]. Both HTTP and MQTT are used in this pilot as described in the following

sections:

Pulling Data With HTTP

Existing data (Things, Datastreams or Observation) can be retrieved using HTTP GET requests. GET Requests can be formulated according to the entity type requested:

```
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Things
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Datastreams
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Observations
```

Individual entities can be requested by supplying an ID (IDentification) such as `(1)` in this case:

```
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Things(1)
```

Nested elements can be returned in the same query. The following query will get all Things with all Datastreams nested inside.

```
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Things?$expand=Datastreams
```

Linked entities can be requested. This query will get all the Observations for the Datastream with ID 1:

```
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Datastreams(1)/Observations
```

If more than 100 results are returned SensorHub will return a paging link.

Pulling Data With HTTP with Paging Results

For large queries paging can be used to break the results into multiple pages, as follows:

```
https://ogc-
hub.compusult.com/SensorHub/SensorThings/v1.0/Datastreams(1)/Observations?$top=100&$sk
ip=100
```

- `$top` changes how many results are shown
- `$skip` is how many entities will be skipped

For example, `$top=100` and `$skip=100` will be the second page of 100 results.

Pulling Data with MQTT

MQTT can be used to accept new data (Observations) from external sensors. To start the flow of data from a sensor enabled with MQTT, the user subscribes to entity topics. Once a subscription is made to a topic, all new observations posted to that topic will be forwarded to the subscriber. Only

new/updated entities will be sent over MQTT, existing ones will not.

The following items can be used as topics:

```
- Things
- Datastreams
- Observations
```

Subscribing to linked entities is also possible. To get all Observations for a Datastream the following topic can be used:

```
- Datastreams(1)/Observations
```

7.6.2. Pushing Data to SensorHub

Data pushed to SensorHub follows the OGC® SensorThings API [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html]. To get started pushing data, it should be understood that each observation (or reported data) needs an associated Thing, DataStream, Sensor, ObservedProperty and FeatureOfInterest object created in advance (please see the following examples). Data or observations can be pushed to SensorHub following the POST or MQTT methods as outlined in the API.

Submitting Data With HTTP POST Requests Workflow

- **Step 1.** Create a Location entity (Optional):
  - POST JSON (JavaScript Object Notation) data to the following URL (Uniform Resource Locator):

```
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Locations
```

```
HTTP
{
    "name":"Example Location",
    "description":"Example Location Description",
    "encodingType":"application/vnd.geo+json",
    "location":{
        "type":"Feature",
        "geometry":{
            "type": "Point",
            "coordinates": [-52.8047,47.5185]
        }
    }
}
----
```

- The request will return a header "location" with a URL:

```
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Locations(1)
```

This is a direct link to the created Location, with the ID (1).

- **Step 2.** Create a Thing entity:
  - POST JSON data to the following URL:

```
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Things
```

```
{
    "name":"Example Thing",
    "description":"Example Thing Location",
    "properties":{},
    "Locations":[{
        "@iot.id":1
    }]
}
```

Location ID is the ID returned in the header of Step 0.

TIP Location does not need to exist, but it would be nice to reference a location.

- The request will return a header "location" with a URL:

```
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Things(2)
```

This is a direct link to the created Thing, with the ID

- **Step 3.** Create a Datastream, Sensor, ObservedProperty and FeatureOfInterest:
  - POST JSON data to the respective URLs, getting IDs from headers as needed:

```
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Sensors
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/ObservedProperties
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Datastreams
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/FeaturesOfInterest
```

Sensor Example

```
{
   "name":"Example Sensor",
   "description":"Example Sensor Description",
   "encodingType":"application/pdf",
   "metadata":"http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf"
}
```

ObservedProperty Example

```
{
   "name":"Example ObsProp",
   "definition":"http://sweet.jpl.nasa.gov/ontology/property.owl#Example",
   "description":"Example ObsProp Description"
}
```

Datastream Example

```
{
   "name":"Example Datastream",
   "description":"Example Datastream Description",
   "unitOfMeasurement":{
      "name":"degree Celsius",
      "symbol":"°C",
      "definition":"http://unitsofmeasure.org/ucum.html#para-30"
   },
   "observationType":"http://www.opengis.net/def/observationType/OGC-
OM/2.0/OM_Measurement",
   "ObservedProperty":{
      "@iot.id":1
   },
   "Sensor":{
      "@iot.id":1
   },
   "Thing":{
      "@iot.id":1
   }
}
```

FeatureOfInterest Example

```
{
   "name":"Feature Example",
   "description":"Feature Example Description",
   "encodingType":"application/vnd.geo+json",
   "feature":{
      "type":"Feature",
      "geometry":{
         "type":"Point",
         "coordinates":[
            -52.8047,
            47.5185
         ]
      }
   }
}
```

- **Step 4.** POST Observations:
  - To post an observation, the user sends JSON to a URL through HTTP POST:

```
https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Observations
```

```
{
   "result":"0",
   "FeatureOfInterest":{
      "@iot.id":1
   },
   "Datastream":{
      "@iot.id":1
   }
}
```

- **Step 5.** Alternative: POSTing all data except Observations at once using nested structure:
  - POST JSON data to Thing URL;
  - Post FeatureOfInterest JSON separately.

```
{
   "description":"Example Thing Description",
   "name":"Example Thing",
   "Locations":[
      {
         "name":"Example Location",
         "description":"Example Location Description",
         "encodingType":"application/vnd.geo+json",
         "location":{
            "type":"Feature",
            "geometry":{
               "type":"Point",
               "coordinates":[
                  -52.8047,
                  47.5185
               ]
            }
         }
      }
   ],
   "Datastreams":[
      {
         "name":"Example Datastream",
         "description":"Example Datastream Description",
         "unitOfMeasurement":{
            "name":"degree Celsius",
            "symbol":"°C",
            "definition":"http://unitsofmeasure.org/ucum.html#para-30"
         },
         "observationType":"http://www.opengis.net/def/observationType/OGC-
OM/2.0/OM_Measurement",
         "ObservedProperty":{
            "name":"Example ObsProp",
            "definition":"http://sweet.jpl.nasa.gov/ontology/property.owl#Example",
            "description":"Example ObsProp Description"
         },
         "Sensor":{
            "name":"Example Sensor",
            "description":"Example Sensor Description",
            "encodingType":"application/pdf",
            "metadata":"http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf"
         }
      }
   ]
}
```

Submitting Data with MQTT Workflow

- Data is submitted in the same JSON formats as POSTed data;

- Data submitted through MQTT will not return IDs through headers;
  - Subscribe to the same MQTT topics to view the IDs as they are created;
  - Or check the SensorThings Representational State Transfer (REST) interface after submissions;
- Instead of URLs, use entity types as topics;
  - REST: `https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Observations`;
  - MQTT Topic: Observations.

Chapter 8. Command Communication System

The Command Communication system establishes communications, a common operating picture, and tasking between first responders, incident commanders, city personnel, and city leaders. The structure of a command communication system is summarized in the following wiring diagram (Figure 34):

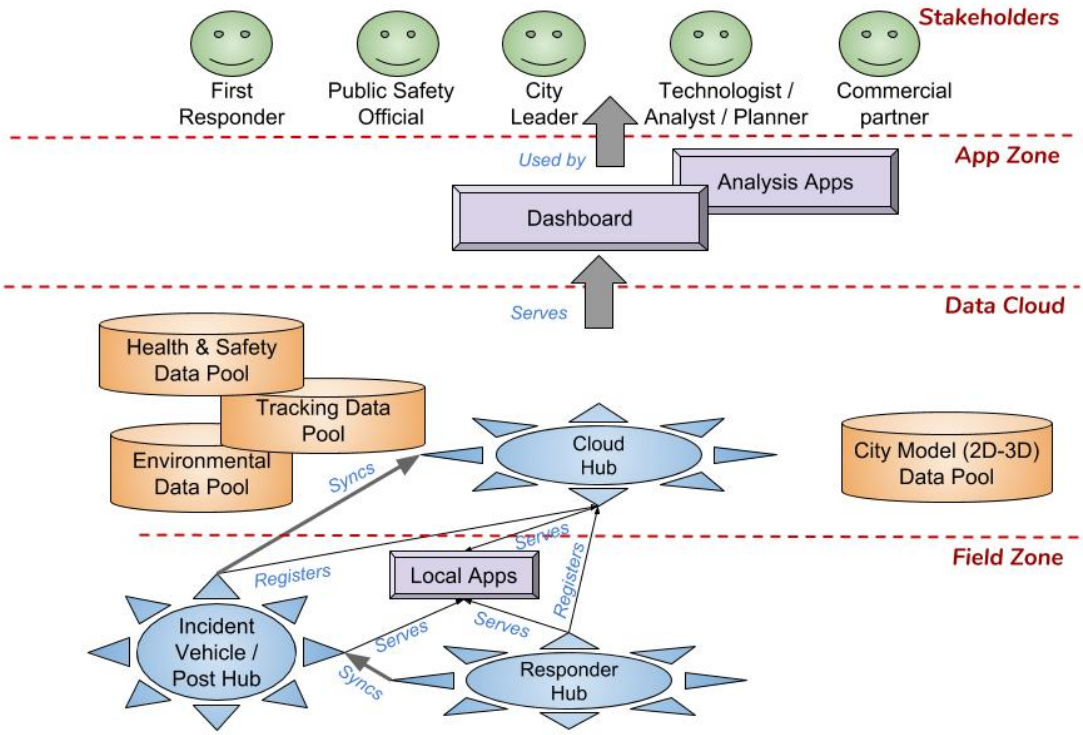

Figure 34. Command Communication System Overview

8.1. D12: 3D Dashboard Application

The 3D Dashboard application provides a common reference framework enabling the decision making and visualization capabilities, *i.e.* a decision support facility, a client for *Flood & Inundation* model and *Traffic & Routing* model. This dashboard becomes the foundation for all decisive actions derived using sensor information, traffic, and incidents through the Cloud Sensor Hub and OGC conformant Legacy systems.

The primary motivation for addressing the concept of 3D Dashboard application is to provide a generalized visualization of geospatial data, sensor feeds, analytics, decision support tools, with a mechanism for incident management. As a generalized data visualization, 3D Dashboard provides a common reference framework to integrate multiple streams of data and business rules and enable the decision making capabilities. The data flow architecture of the dashboard is shown in Figure 35.

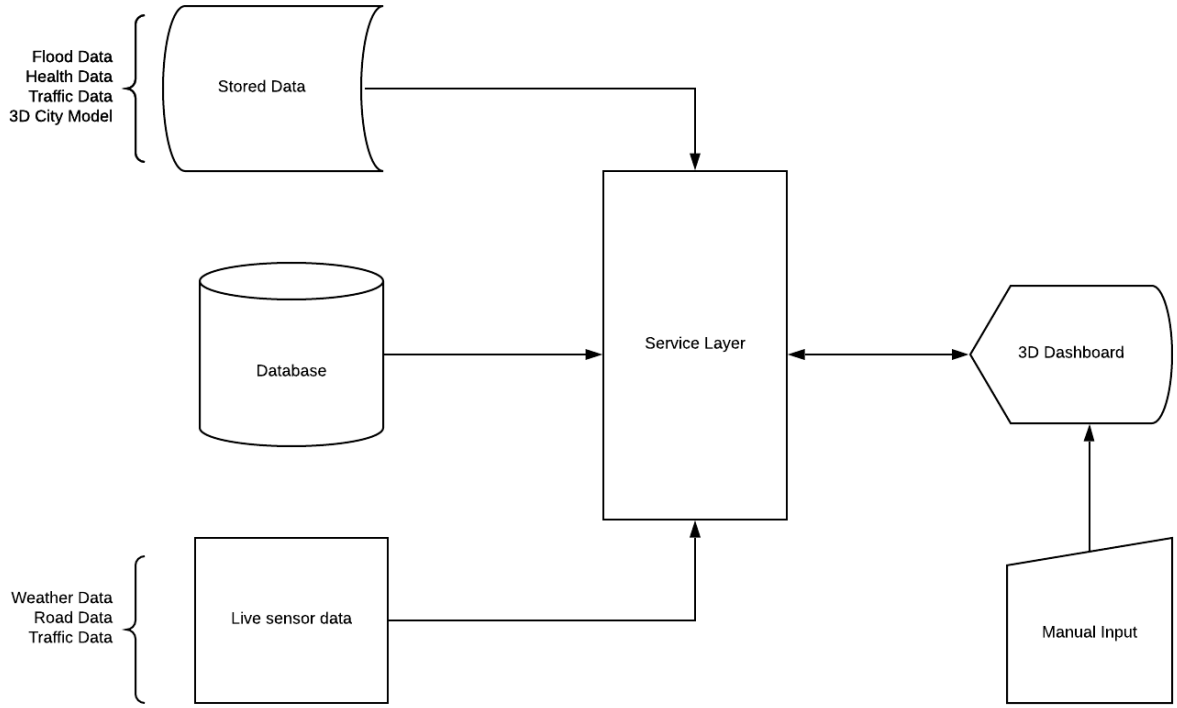

Figure 35. Data Flow Architecture of Dashboard

NOTE

Benefits of the 3D dashboard

- Benefits of the 3D dashboard for end-users:
  - First responders benefit from the new tools and applications provided by the improved decision making system which offers a comprehensive common operating picture;
  - Emergency evacuation processes can be expedited through better tools;
  - Live video feeds provide better community outreach.
- Benefits for developers:
  - It is designed based on the open standards and specifications are truly interoperable for future product releases;
  - Discovery of possible improvements for OGC standards and lessons learned;
  - Lessons learned can assist with future research and reduce the learning curve.

8.1.1. Add 3D Data Service

Add Data Service enables data upload provision. This dashboard uses several data sources for testing and experimentation including glTF [https://www.khronos.org/gltf/] (GL Transmission Format) and 3D Tiles. The glTF specification is a file format for 3D scenes and models [https://en.wikipedia.org/wiki/3D_modeling] using the JSON [https://en.wikipedia.org/wiki/JSON] standard. This format is an efficient, interoperable format with minimum file size and runtime processing by Apps. The glTF and 3D Tiles [https://www.ogc.org/standards/3DTiles] formats are directly supported by the Open Source CesiumJS [https://cesium.com/cesiumjs/] 3D mapping library.

OGC® 3D Tiles [https://www.ogc.org/standards/3DTiles] is an OGC Community Standard for streaming massive heterogeneous 3D geospatial datasets. A single glTF data set, referred to as a Scene Layer, is a container for arbitrarily large amounts of heterogeneously distributed 3D geographic data.

The datasets include glTF, 3D Tiles.

The user can browse, upload and view the 3D Tiles and glTF data formats on the 3D dashboard. In the background, the file is stored on the server path specified where it is processed and retrieved for the visualization. The user must browse a zip file containing a subfolder that has all the supported files to upload, the zip file name and subfolder name must be the same.

The Add data panel (Figure 36) provides options to select the type of data to upload. Upon selection, a user can browse the file(s) and provide the required information.

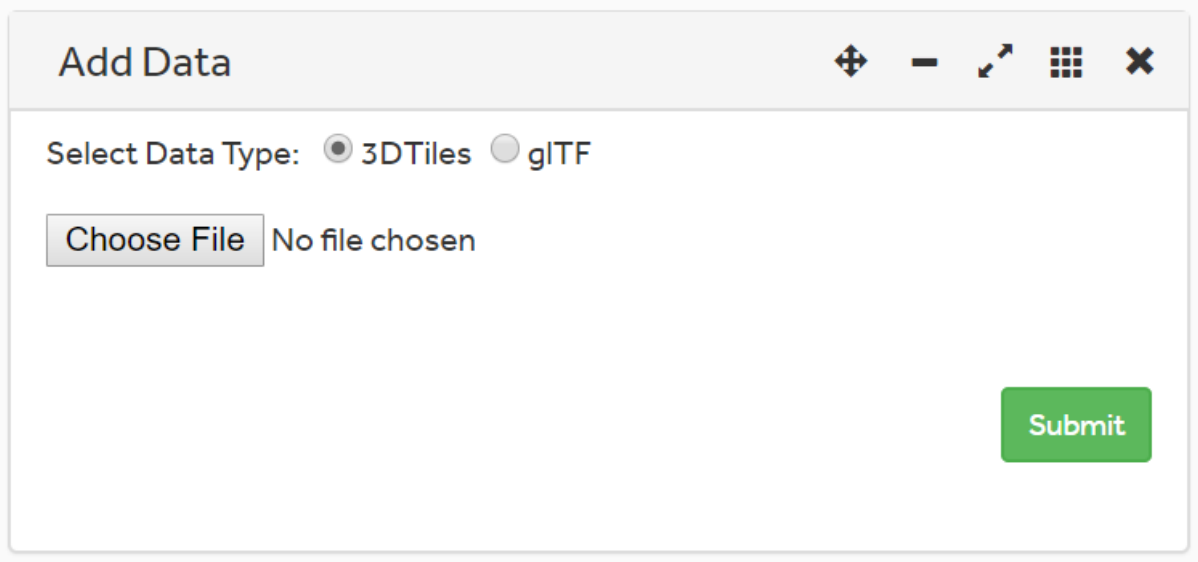

Figure 36. Add 3D Data Service

The below figure (Figure 37) shows the uploaded 3D Tiles through Add Data service.

Figure 37. 3D Tiles

The below figure (Figure 38) shows the uploaded glTF model through *Add Data* service.

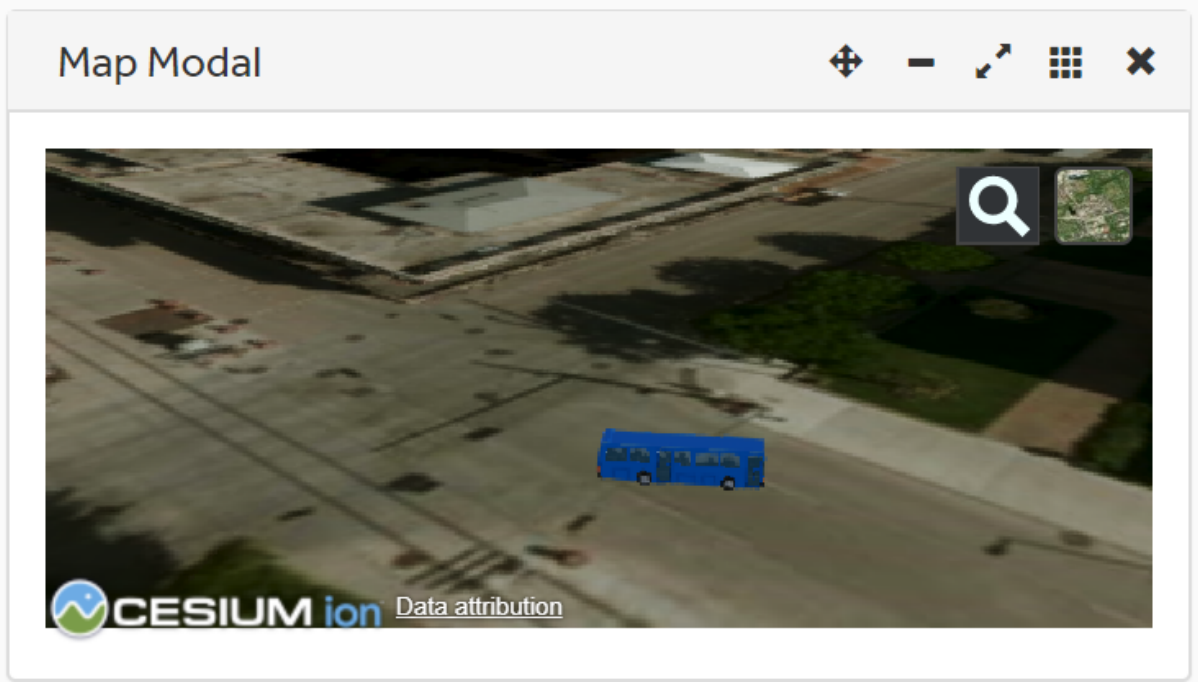

Figure 38. Uploaded glTF Model

8.1.2. Monitoring

Monitoring helps in trend prediction and modeling study. The 3D Dashboard monitors and compares current or past incidents at a time to understand and model the changes or impacts.

- **Stream Gauge Observations**: The stream gauge (alias stream gage) data is being provided by the sensors from another SCIRA pilot participant, *Botts Innovative Research*. This information is fetched by polling their endpoint through REpresentational State Transfer (REST) services (Figure 39).

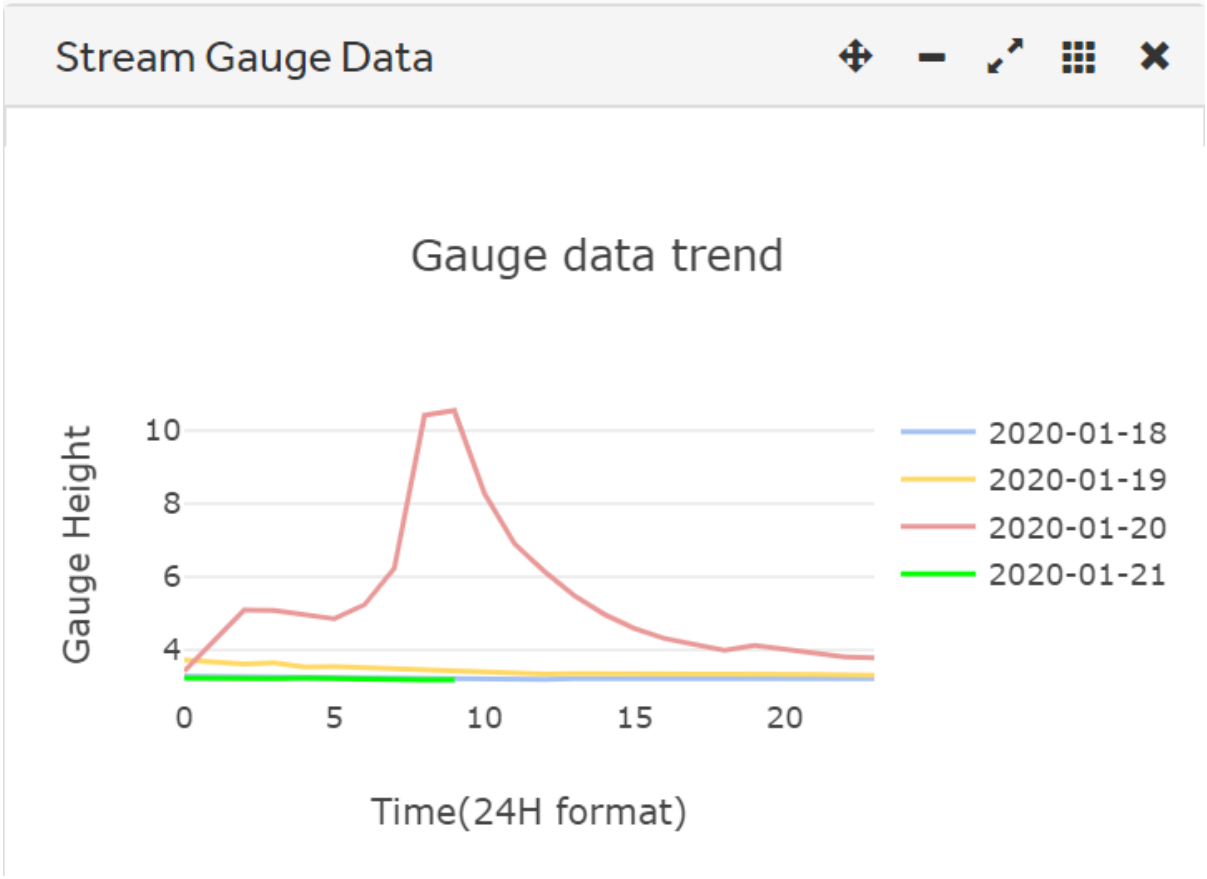

Figure 39. Stream Gauge Trend

- **Weather Observations**: The weather observations are fetched from *Compusult SensorHub* (datastream IDs: – 174, 179, 180, 181, 184) through REST services and are represented as shown below (Figure 40) for the ease of monitoring.

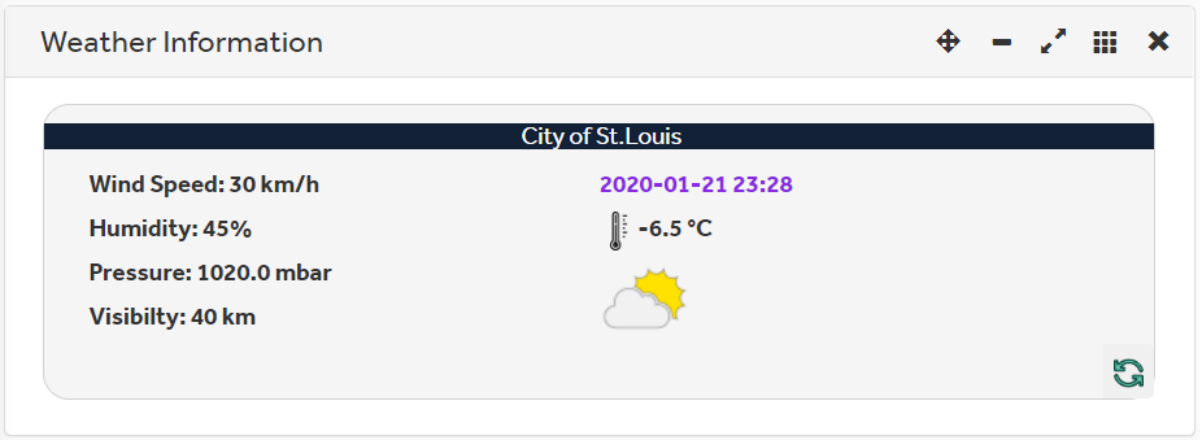

Figure 40. Weather Board

8.1.3. Sensor Observations

The services offering sensor observations are registered and are accessible on the 3D Dashboard. Users can input any other endpoint based on his/her requirements. This section allows the user to query each sensor's observations and visualize, which is used to compare predicted with current weather, flow observations (Figure 41).

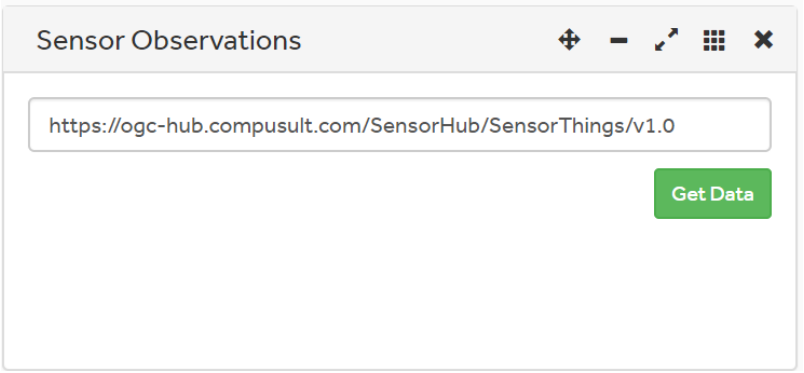

Figure 41. Interface

User can visualize the historical and live sensor readings on different charts with respect to time. This interface provides functionality to generate charts based on user input information for ease of visualization (Figure 42).

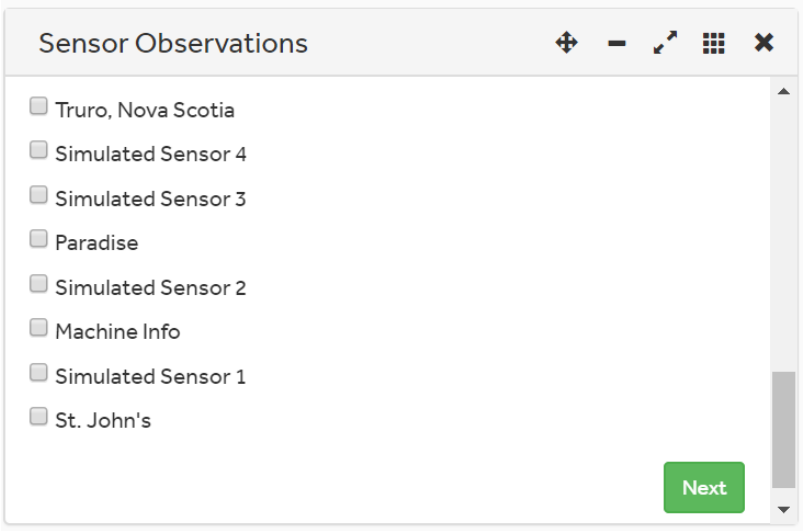

Figure 42. Selected Sensor DataStream Observations

8.1.4. Alerts

The 3D Dashboard allows alert subscriptions related to flooding (both river and flash), health, and street closures. The subscription is achieved using MQTT protocol and to topics such as FloodTrend, FlashFlood, PersonAssist, and RoadClosure (Figure 43).

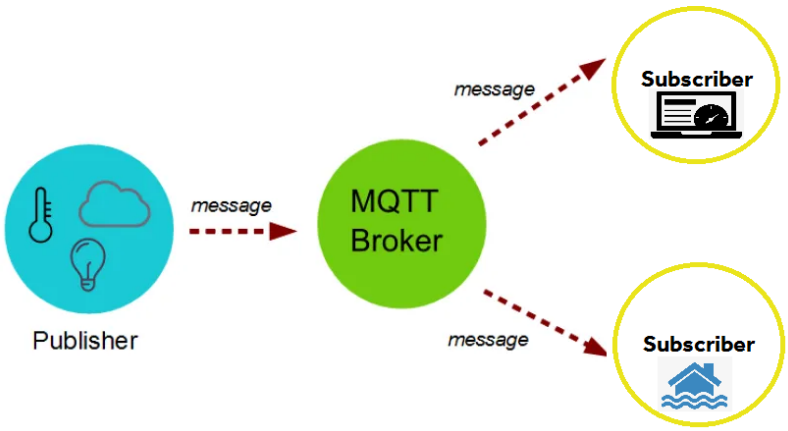

Figure 43. MQTT Protocol

MQTT is an open lightweight, network protocol that transports messages between devices. In this case the *Compusult SensorHub* is the publisher and the 3D Dashboard is the subscriber (Figure 44).

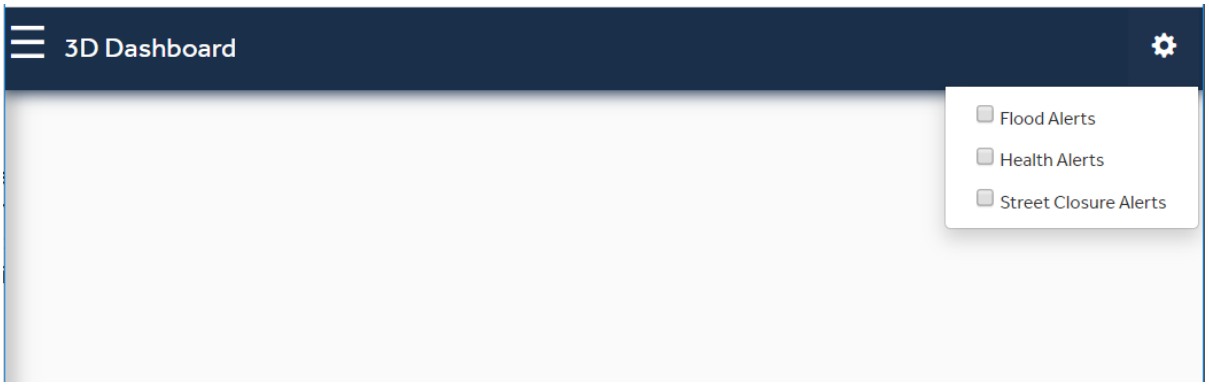

Figure 44. Subscription to Alerts From the Dashboard

Alerts, when published, pop up on the dashboard with a certain message based on the predefined thresholds and the identified risk. Once the output from the responsible models is fed to the *Compusult SensorHub*, that information is fetched and displayed accordingly. These alerts further facilitate taking preparatory actions in a decision making support system.

8.1.5. Incidents

While monitoring the scenario, if the user identifies a risk, she/he can set up the predefined threshold values to alert/notify the stakeholders about the identified risk (Figure 45). Also, the Operator receives alerts for the incidents such as public assistance, street closure, fire or vehicle accident. Then she/he can locate or track that incident, add tasks and act upon. The alerts are fetched from the *Coolfire API* through REST calls.

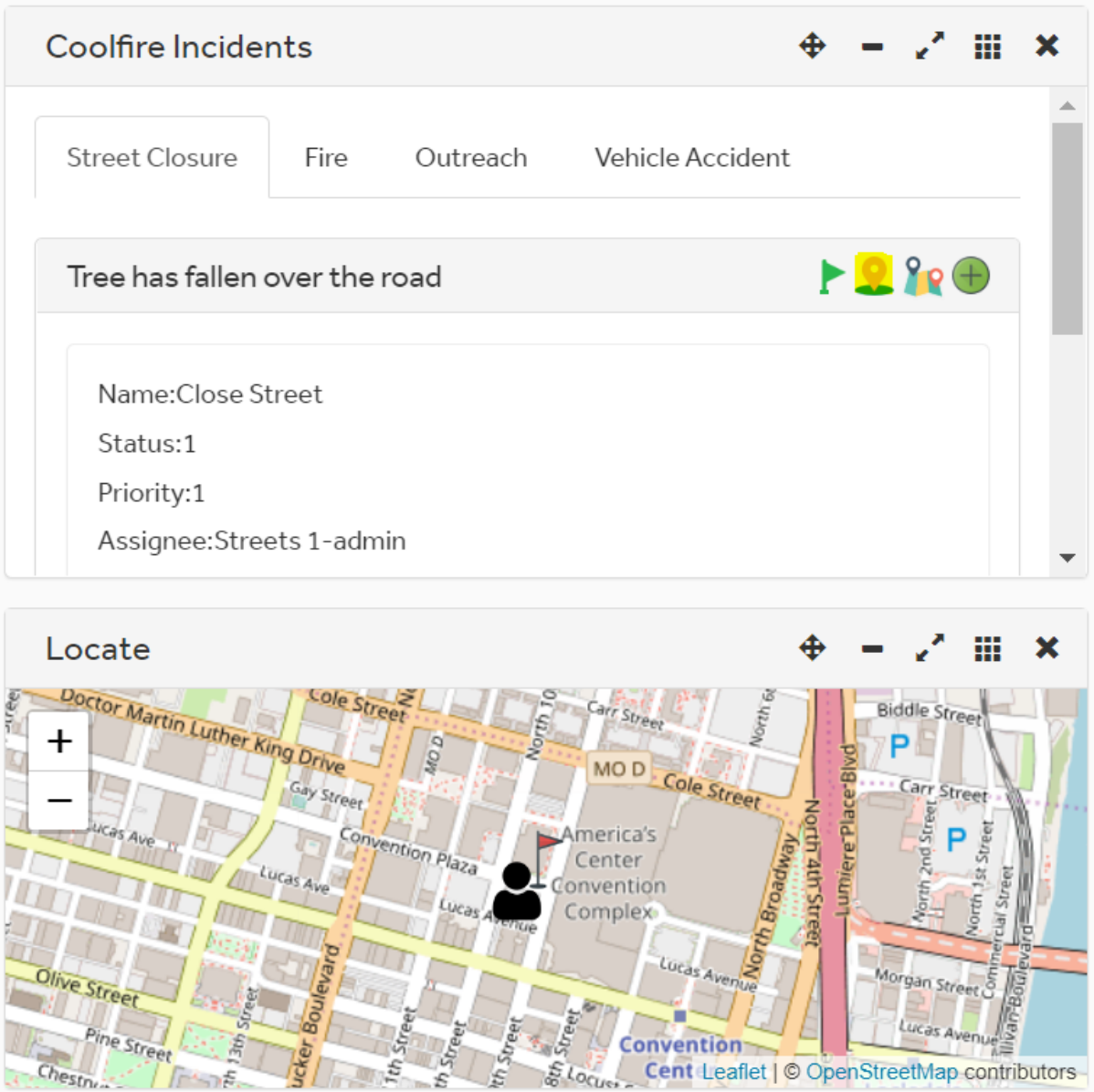

Figure 45. Incidents, Tasking and Tracking

The operator visualizes the team location by fetching location coordinates by sending a recurring request to the Ronin platform followed by computing route.

- **Create New Incidents:** In the SCIRA pilot, once the call center received a call reporting a fire/vehicle accident, an incident was raised in the 3D Dashboard. After confirming the incident creation, a dispatch task was added to the record. In these situations, the 3D Dashboard incorporates the *Coolfire API* services such as sessions in the incident creation (Figure 46).

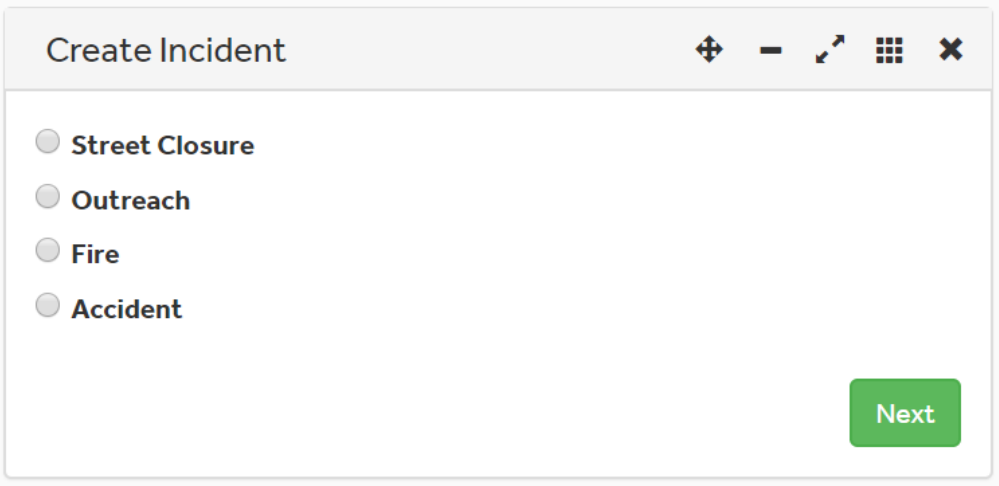

Figure 46. Create New Incidents

For example, when the operator receives an inundation alert or a call starts creating an incident event. Based on the type, the operator creates an incident in the following panel (Figure 47).

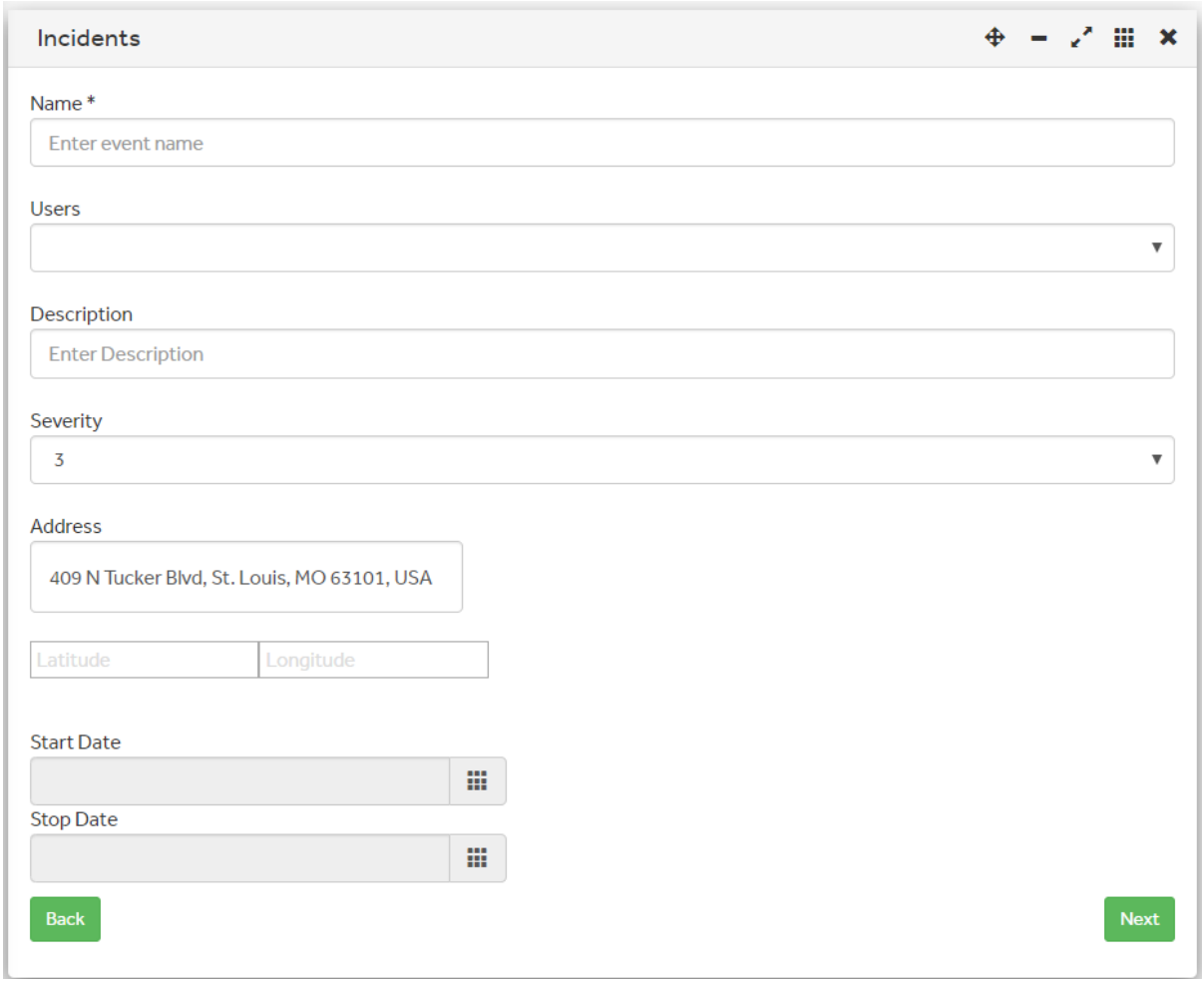

Figure 47. Incident Parameters

8.1.6. Routing

The Routing service allows a user to request a new route computation by considering the street closure. The user enters the source and destination coordinates, or selects their location from a map, then fetches the generated routes using the *Skymantics* routing service. The following figure (Figure 48) shows the requested new route based on a fire incident scenario.

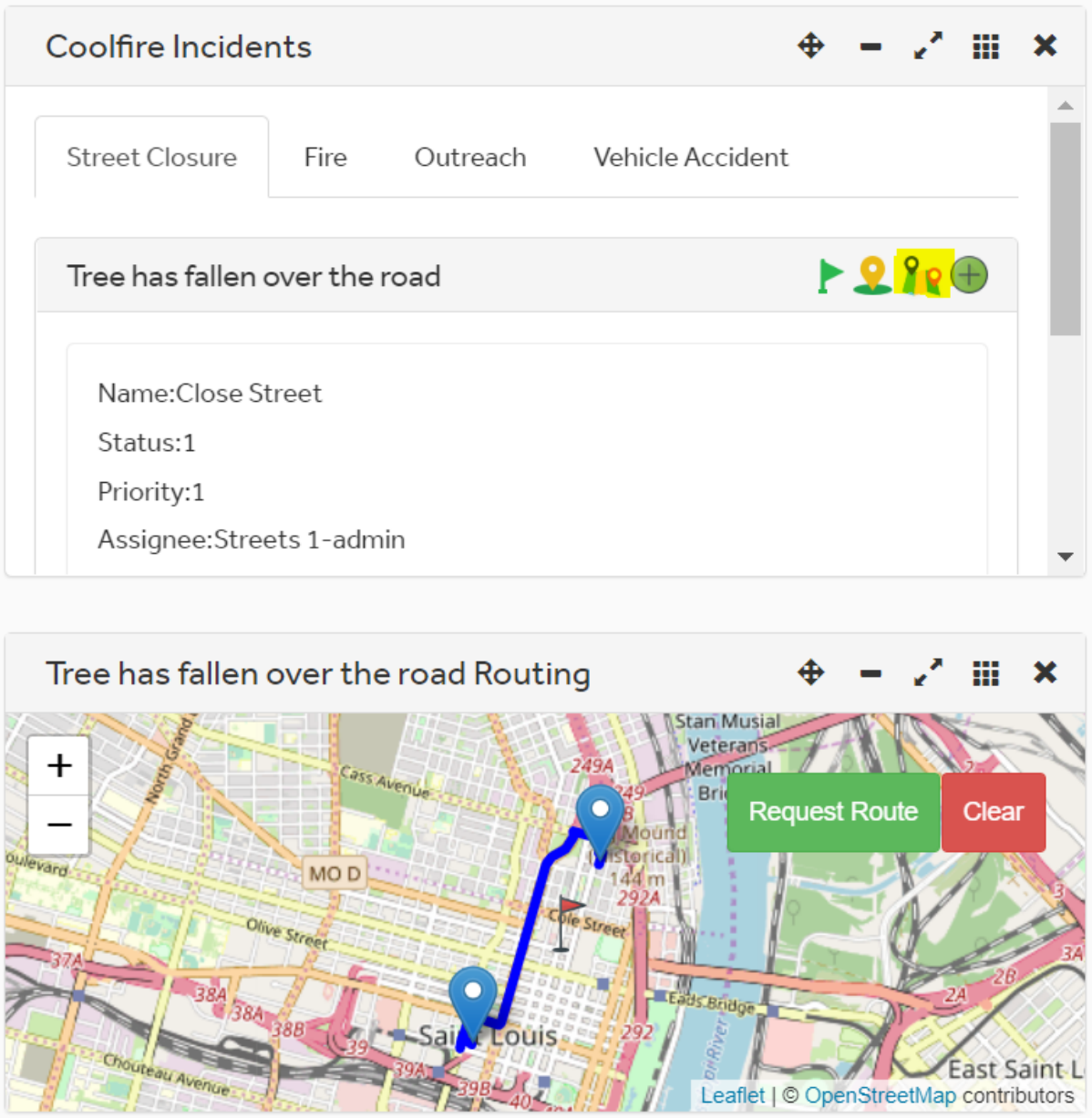

Figure 48. Requesting Route

8.1.7. Flood Model Output

Flood model prediction observations from *Compusult SensorHub* are published on to the 3D Dashboard. The flood simulation model output consists of a 3D map is shown in the following figure (Figure 49).

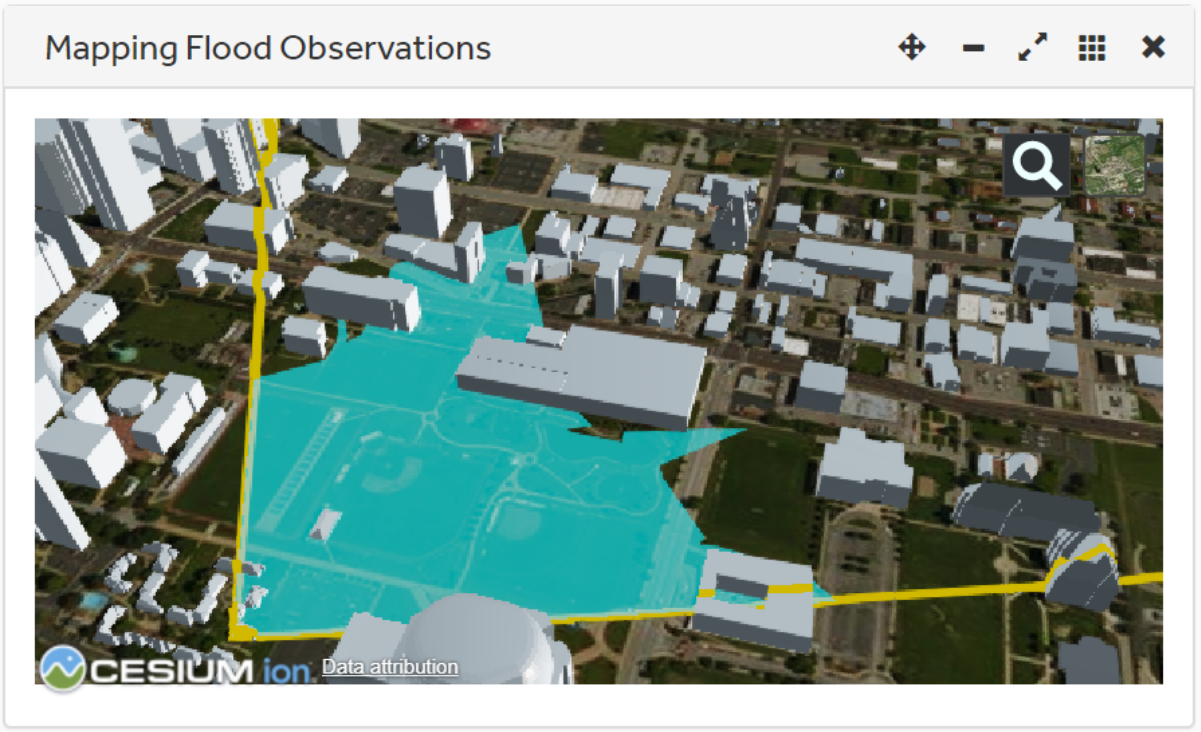

Figure 49. Mapping Flood Observations

A report (Figure 50) also shows the number of impacted buildings, area (`in square meter`), number of active cameras in the site and their corresponding feed.

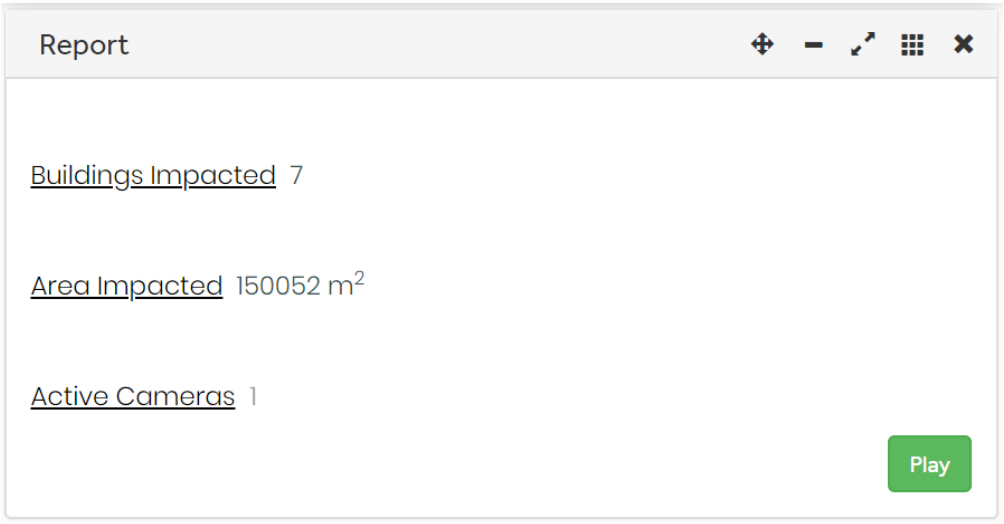

Figure 50. Flood Observations Report

Finally, the following section (Figure 51) shows a provision to the current observations such as pedestrian count, vehicle count, and traffic flow within the camera's view is fetched from the *Compusult SensorHub* and generated from the *University of Calgary Smart Camera* sensor (datastreams IDs: 196,197,199).

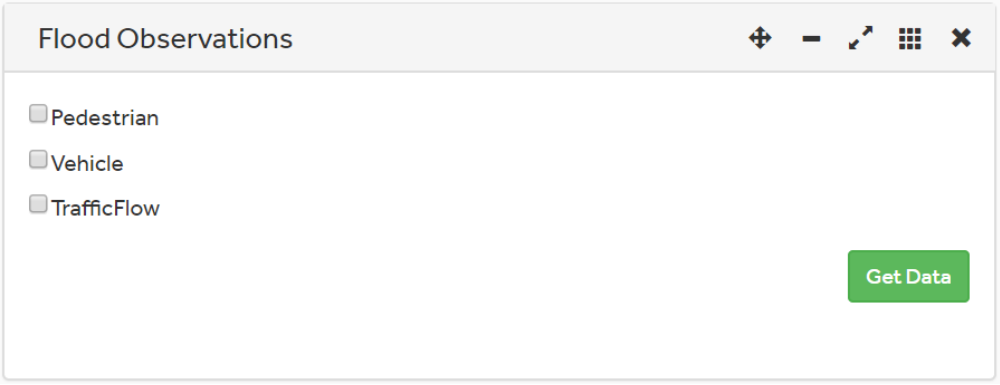

Figure 51. Query Flood Observations

While monitoring the scenario, if user identifies the risk and set up the predefined threshold values, the stakeholders will be alerted/notified with the identified risk.

Operator visualizes the live data for traffic and videos, inundation areas, overall incident and task status in the 3D dashboard (Figure 52).

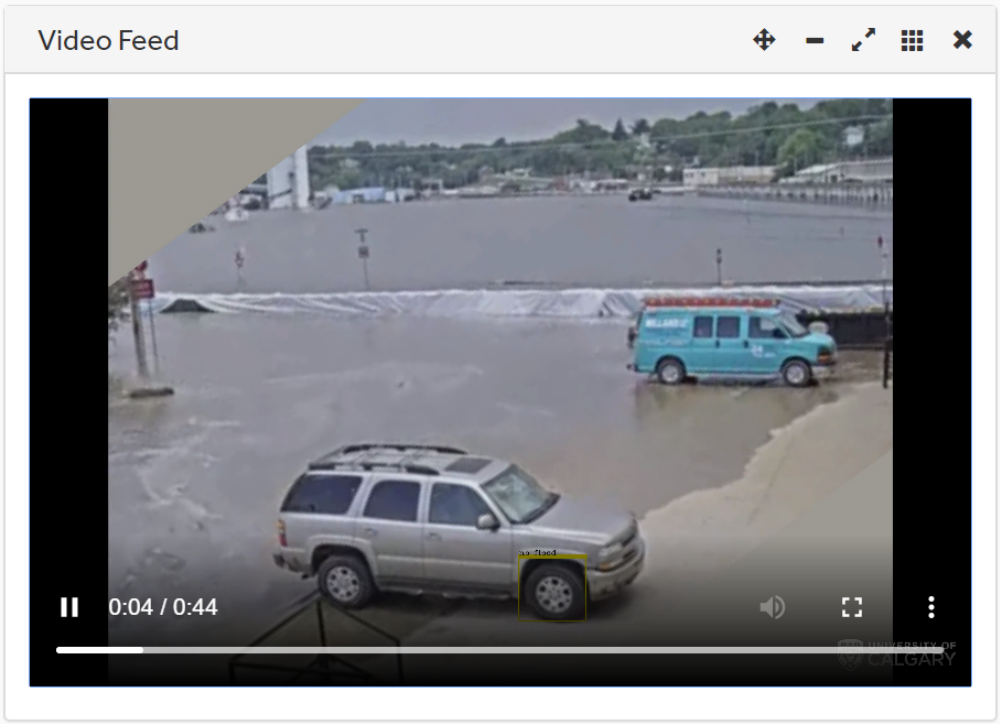

Figure 52. Video Feed

8.2. D13: Command & Communication System

In the SCIRA pilot, *Coolfire* dashboard software gathered information from *Compusult SensorHub* as well as directly from dispatch operators. The information that was gathered consisted of data from the smartwatches, flood information (data from both predictions and camera data), as well as location and communications from the parties involved. This derived information was made available to users of both the *Coolfire* dashboard and *Coolfire* mobile response applications. Users of *Coolfire* applications were able to take action and perform their duties using the gathered information that was provided to them (Figure 53).

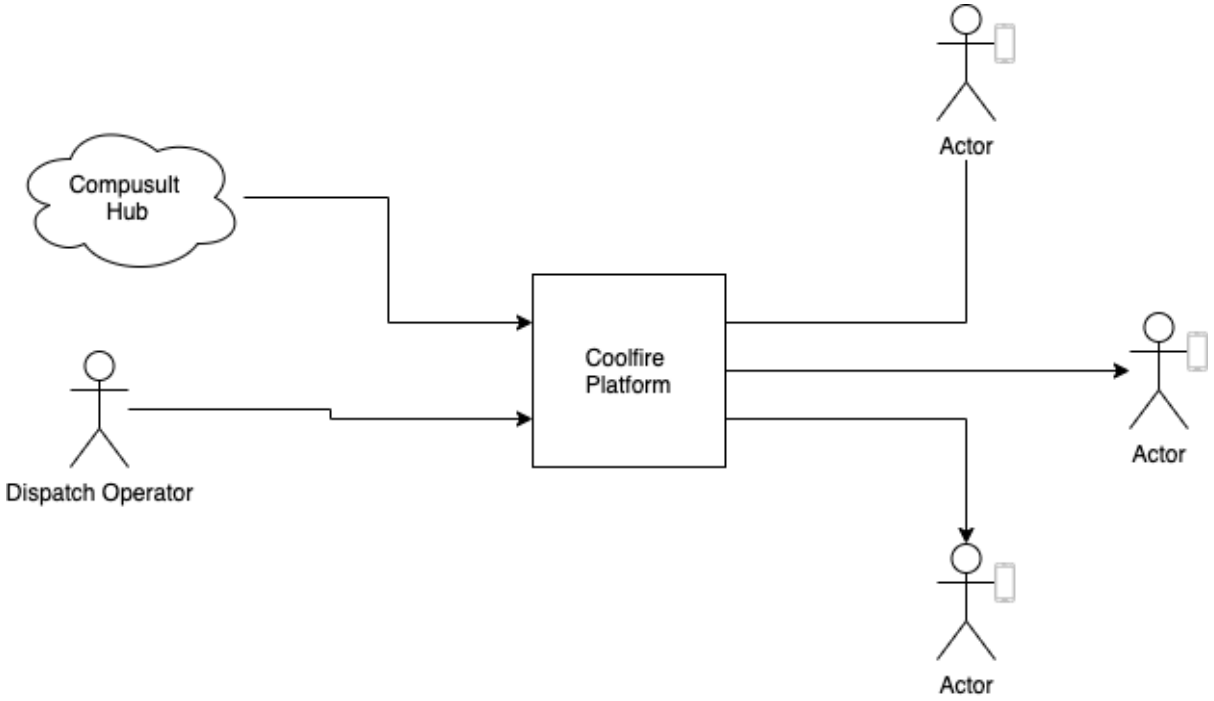

Figure 53. Command Communication System Activity Diagram

Compusult GoMobile and *Web Enterprise Suite (WES)* applications were used as the Command and Communication System. GoMobile was used to collection observation data using the sensors available on the Android tablet or phone. The observations were synchronized to the WES and displayed geospatially on a map interface using OGC® GeoPackage [https://www.geopackage.org/] as the transfer mechanism. The WES would in turn use MQTT to send the observation to the SensorHub where it was then available to all parties (Figure 54).

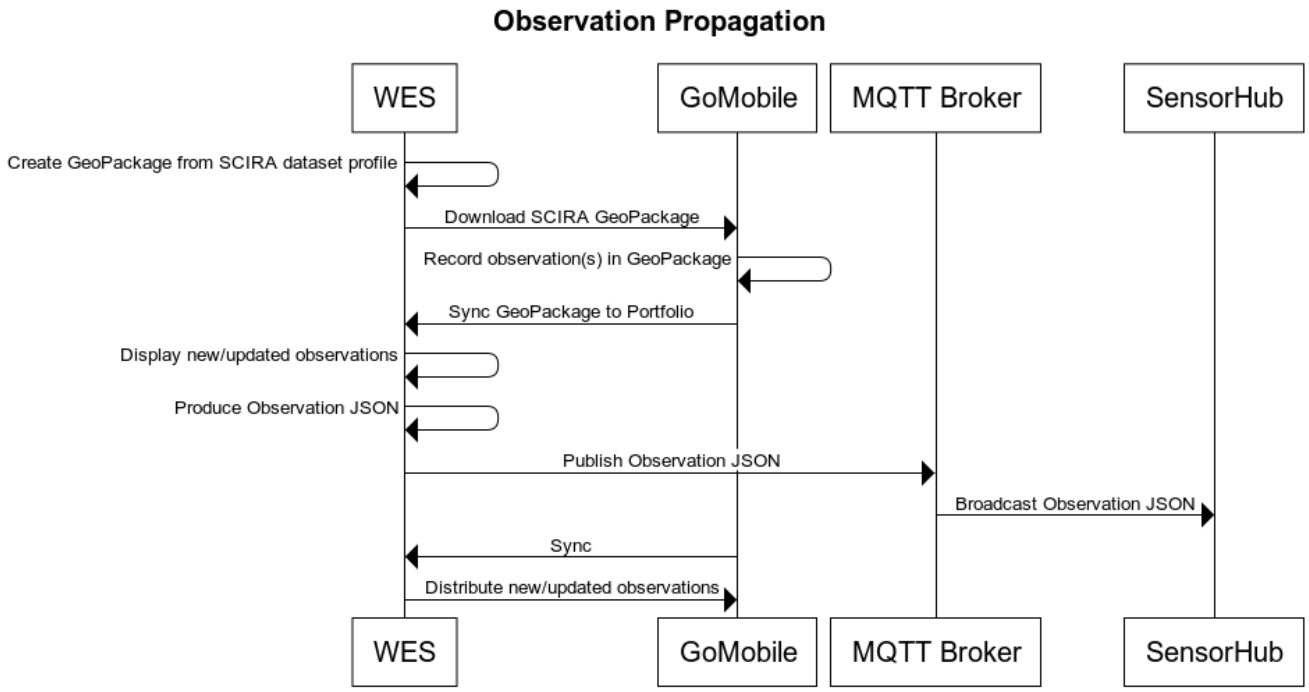

Figure 54. Compusult Command Communication System Overview

When an OGC® GeoPackage [https://www.geopackage.org/] containing SCIRA features is synced, the features will be extracted from the GeoPackage and encoded as a JSON observation. The resulting JSON would be pushed to the SensorHub using MQTT and a datastream depending on the

observation type. Alerts, based on values/conditions, are provided within a live service that is compliant with the OGC® Web Map Service (WMS) [https://www.ogc.org/standards/wms] standard using the *SensorHub*. Alerts are transmitted over SMS (Short Message Service) to ensure mobile device support in addition to specific mobile App and web browser support. *SensorHub* is able to monitor sensors on *SmartHubs* and be able to alert and task *Incident Management Personnel* based on sensor values.

Example 1. Command Communication System Workflow:

- The command and communication system will provide details of populated areas that may be affected by the flooding and dispatch mobile users to those areas.
- As the buildings are checked for people, mobile users will collect data indicating building marked as safe or checked.
- The feature data is then synced across all connected mobile users and command central.
- Observations are created and pushed to SensorHub where alerts can be triggered to notify subscribed parties of the situation as it happens.
- If additional services are required at a location, those can be dispatched via messaging across mobile devices and command and communication services.

8.3. D17: Community Mobility Navigation, Alert & Sensing Application

Community users were able to use the *Coolfire* community application to submit information regarding a person in need or a flood observation incident. The information that was gathered, includes the location of the flood or the person in need and a photo. This information was used by the dispatch operator using the *Coolfire* dashboard software to dispatch emergency personnel who were using the *Coolfire* responder application.

Chapter 9. Indoor Navigation System

Indoor NAvigation System components are summarized in the following wiring diagram (Figure 55):

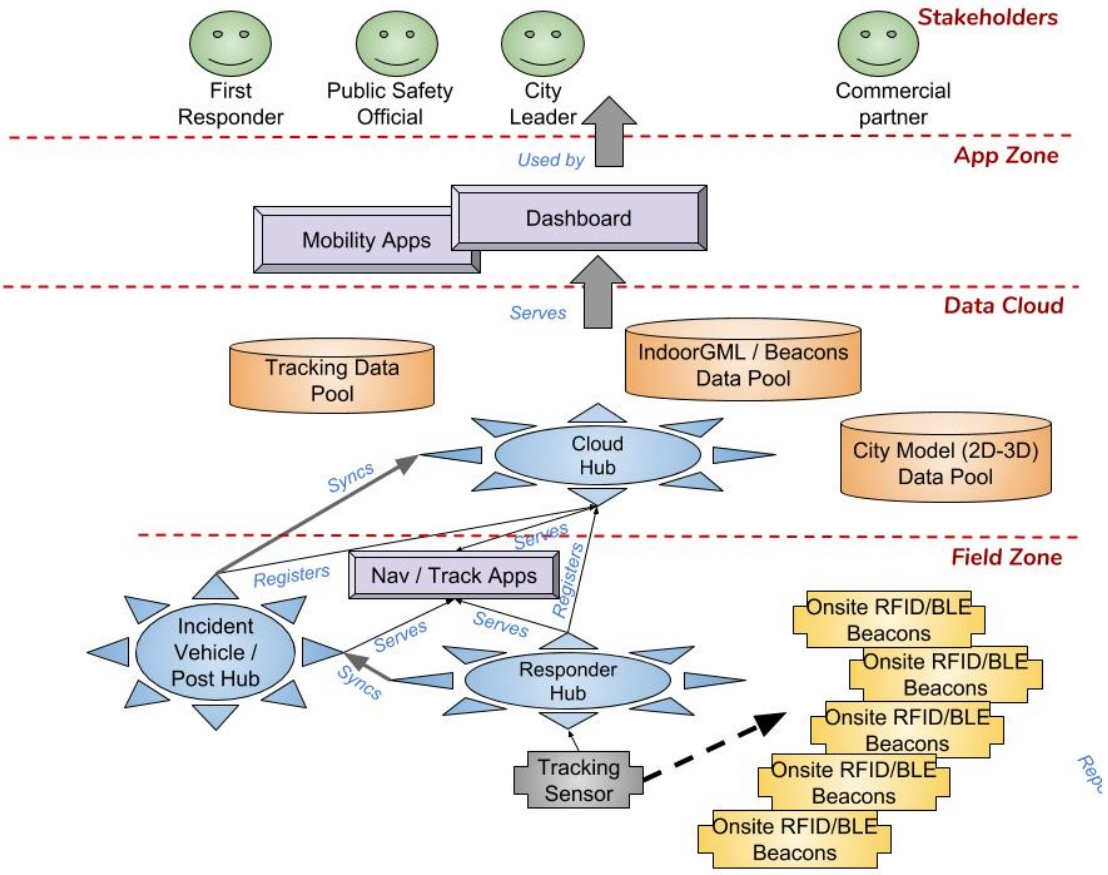

Figure 55. Indoor Navigation System Overview

The GPS system provides current location positioning for civilian applications with an accuracy of one meter or better. This enables a wide range of applications from local awareness of people and places, to tagging the location of various events (*e.g.* accidents, flooded streets, *etc.*), to the ability to route from any location to the location of the emergency event. However, GPS signals are often lost or highly inaccurate inside buildings, making in-door navigation and tracking much more difficult. An Indoor Navigation system enables first responders to navigate and be tracked inside a building in the absence of GPS reception. There are several evolving technologies that are being investigated for their potential ability to aid in tracking locations inside of buildings, and therefore being able to know where people are and how to navigate them to where they want to be. These technologies include Bluetooth Low-Energy (BLE) beacons, triangulation using WiFi signals, RFID, ultra-wideband (UWB) devices, just to name a few. This project focused on the use of BLE Beacons as described below.

9.1. D8: First Responder Indoor Navigation Application

In the OGC SCIRA Pilot scenario, the challenge was to determine and report the location of a first responder inside of a burning building and to transmit that location over LTE (Long Term Evolution) in real-time to a commander in the field or to *Command Communication System*. These locations could be used to track the progress of the first responder, guide the first responder to

where they have to go by using verbal commands or allow the commander to identify which room the first responder was in when he declared "Clear". The SCIRA Pilot exercise accomplished this through the combination of pre-deployed BLE beacons and an Android phone running *OSH SmartHub* technology.

A BLE beacon is a device that repeatedly sends out a radio signal at some configured frequency and at some configured power. For most such beacons, the signal sent is also simple in that it usually consists of a unique ID identifying the device or a URL linking to a more complete description or message. Thus, the beacon itself does not detect anything (such as its location or purpose), it merely sends out a signal that can be

In the SCIRA Pilot, the detector device was an Android-based SmartHub phone running OpenSensorHub (OSH), which includes a driver for detecting, recording, processing, and streaming out BLE beacon messages in real-time. The processing algorithm also utilized data from beacons (*e.g.*, transmitted signal strength and received signal strength), in order to determine which beacon was the closest beacon. *SmartHub* was then able to retrieve the device ID from the beacon and used that ID to access that beacons information from the web (or from a local database running on the SmartHub). In this exercise, the message included the pre-determined latitude, longitude, altitude, and room number associated with the beacon.

9.2. D18: RFID/BLE Indoor Navigation Beacons

Twenty BLE beacons were deployed throughout the T-Rex building where the SCIRA Pilot was located. Mainly for "room clearing" exercise, BLE beacons were installed at the street entrances, throughout the basement, on the stairs leading from the basement to the third floor, and throughout the third floor (Figure 56).

Figure 56. BLE Beacon Deployed Near Basement Stairs

The beacons used were Eddystone Social Retail BLE beacons costing around $20 each (Figure 57).

Figure 57. Eddystone Social Retail BLE Beacons Used for This Pilot

Once the beacons are deployed throughout the building, the database is updated for each beacon to assign a geolocation and room number to that beacon ID (Figure 58).

Figure 58. Placement of BLE Beacons on Third Floor

As the first responder moves throughout the building the commander's dashboard receives a geolocation coordinate and the room number of the closest beacon. Thus, the location of the first responder can be indicated by a position icon or highlighting of the room as shown in the figures below (Figure 59 and Figure 60).

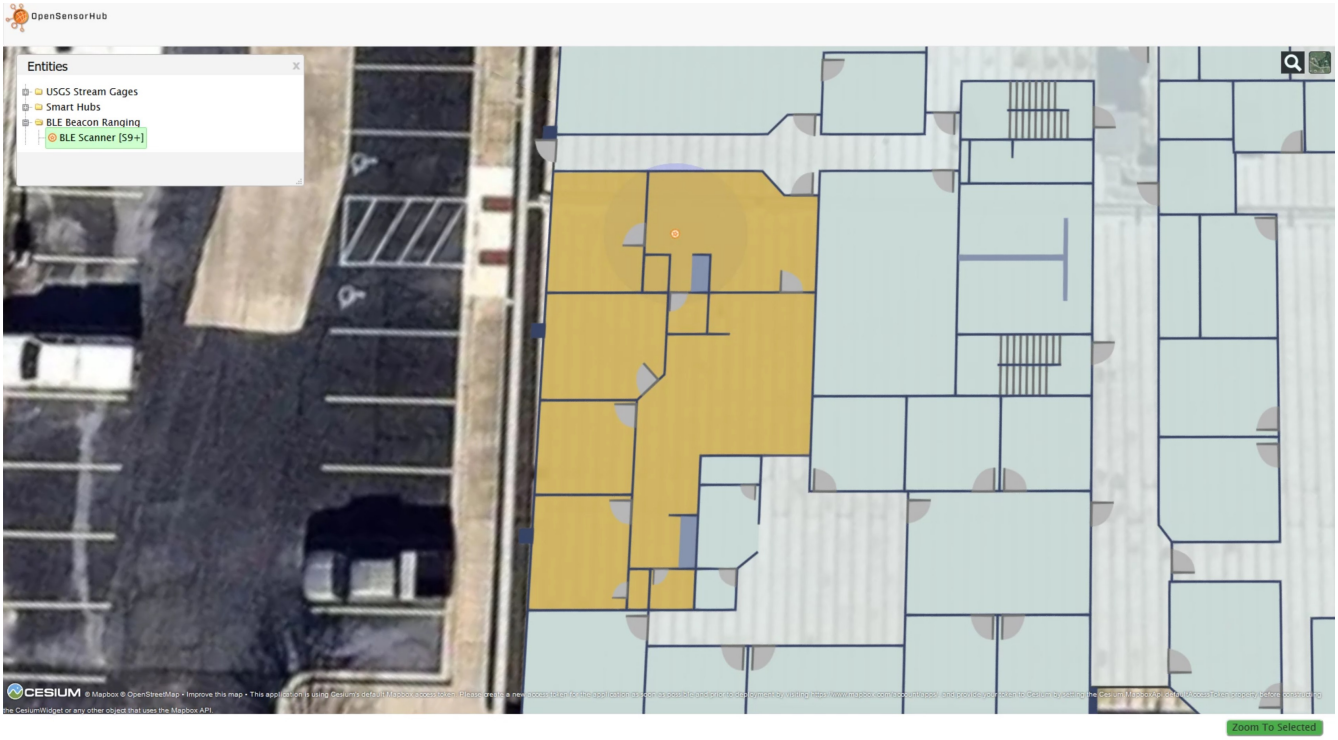

Figure 59. Geolocation Detection Testing

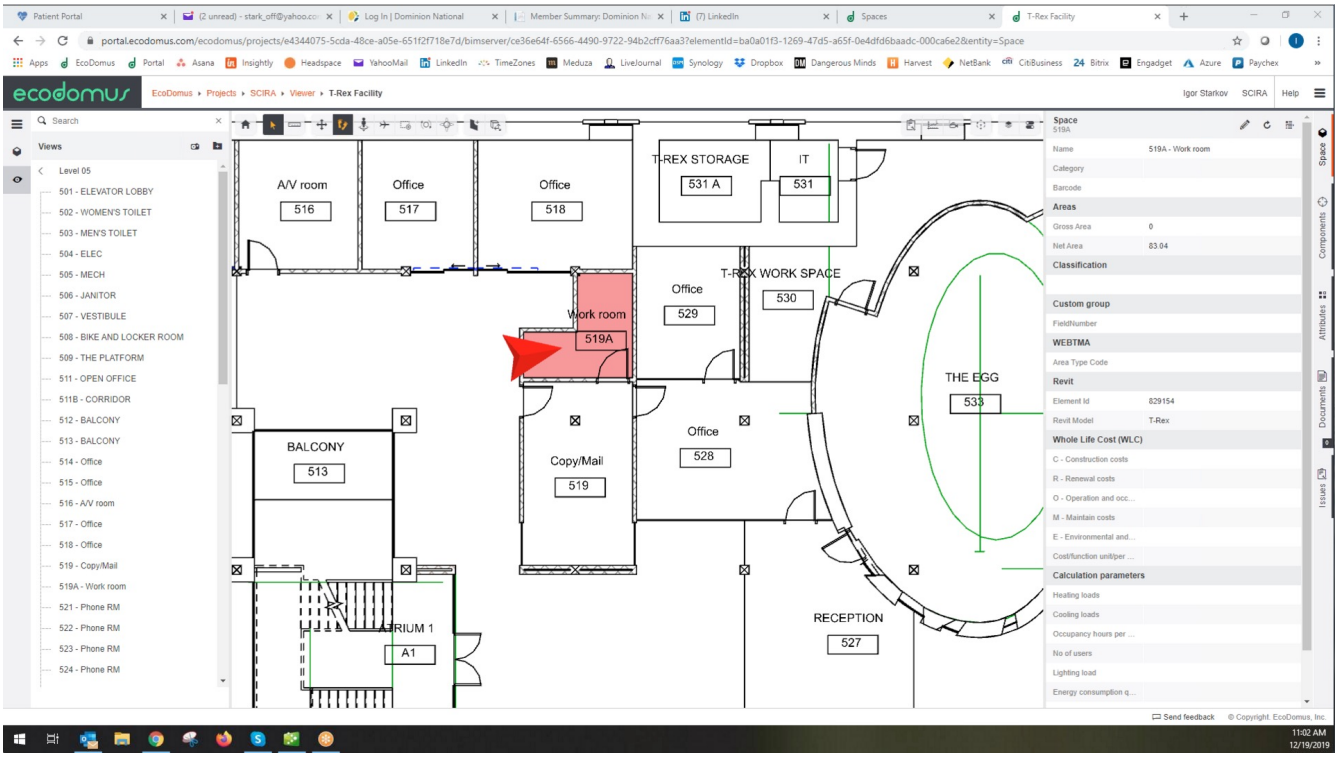

Figure 60. Room Highlight Indicating Presence of First Responder

Chapter 10. Street Navigation System

The Street Navigation system is summarized in the following wiring diagram (Figure 61):

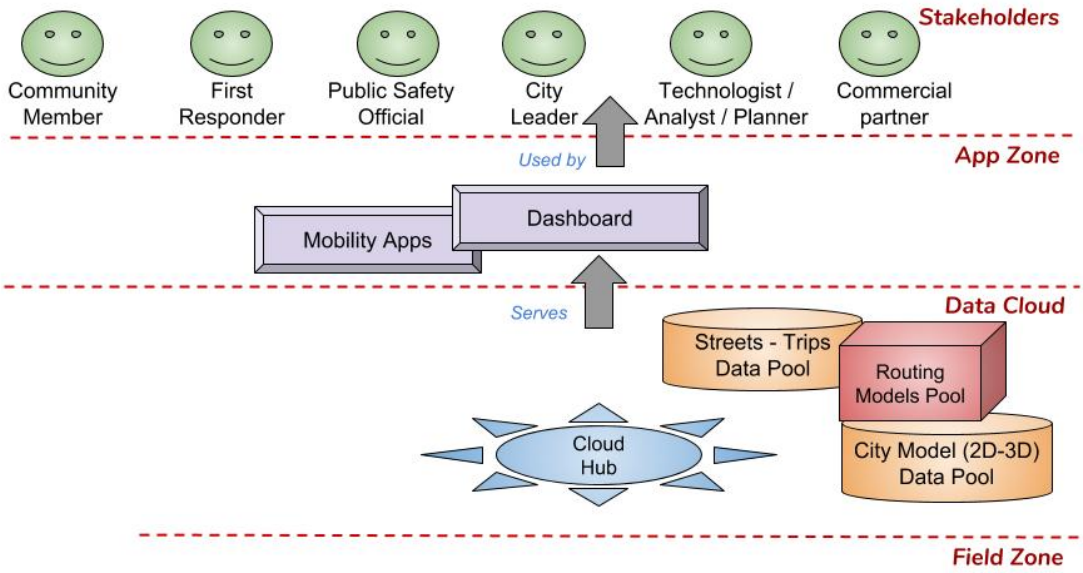

Figure 61. Street Navigation System Overview

10.1. D15: Traffic Management & Routing Model

This deliverable is implemented using *Skymantics Universal Routing Engine*. This routing engine is used in the following SCIRA scenarios (please find more information in the Master Scenario Events List:

- Scenario 2: Flash flooding
- Scenario 3: Assist vulnerable population
- Scenario 4: Building fire
- Scenario 5: Vehicle accident

The routing engine interacts with the following Apps:

- *Cyient 3D Dashboard*: provides optimal route and historical routes;
- *Botts Responder App*: provides optimal route in response to task notification;
- *Compusult Cloud SensorHub*: subscribes to published road closure observations.

10.1.1. Architecture and Design

The image below (Figure 62) shows the main use cases of the routing engine in the SCIRA Pilot and the Apps with which it interacts.

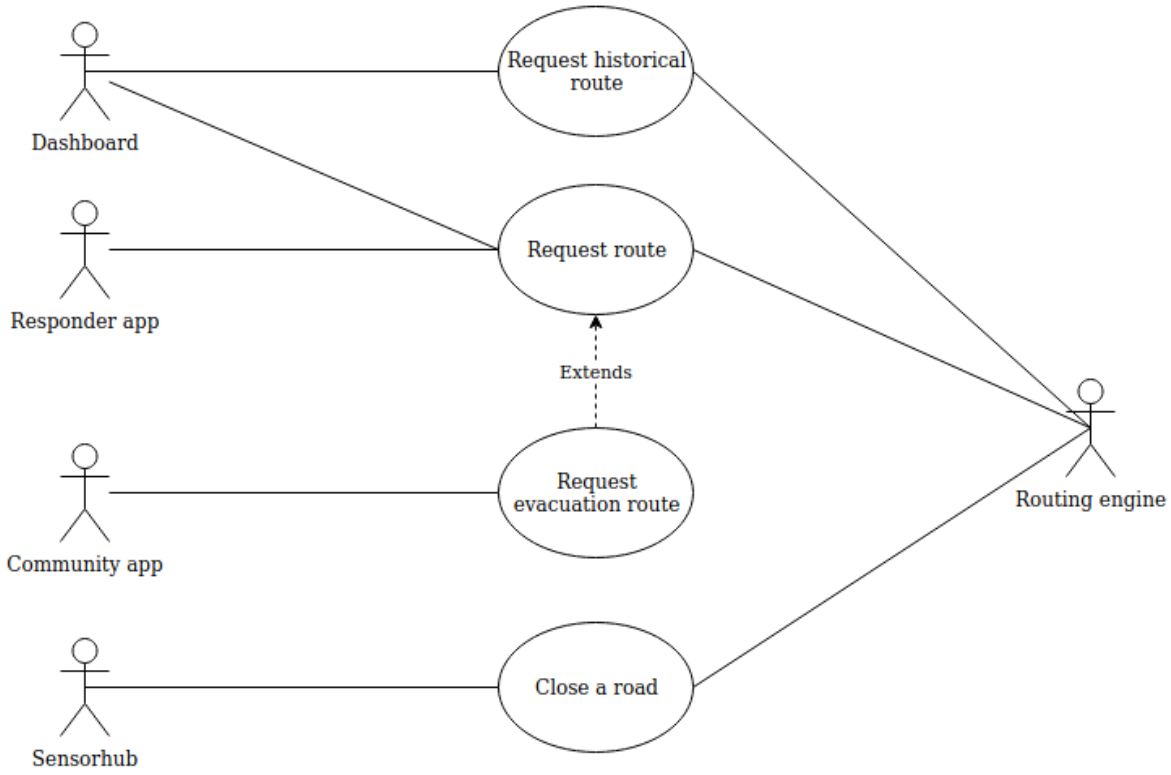

Figure 62. Traffic Management & Routing Model Use Cases

- **Request route**: an app or dashboard requests a route using the Open Routing API [https://www.opengeospatial.org/projects/initiatives/routingpilot], the routing engine computes the route and delivers it in the format of a Route Exchange Model [http://docs.opengeospatial.org/per/19-041r3.html#RoutingExchangeModel].
- **Request evacuation route**: an extension of the standard request route, the routing engine computes several alternatives for the same route and selects one randomly, to avoid creating traffic jams when evacuating an area.
- **Request historical route**: this use case makes use of a feature implemented in the OGC® Web Processing Service (WPS) [http://docs.opengeospatial.org/is/14-065/14-065.html] of *Skymantics*' routing engine that keeps a cache of past routes and can be queried using the *Open Routing API*.
- **Close the road**: the routing engine subscribes to_Compusult SensorHub_'s road closure datastream (using a MQTT protocol), which forwards a road closure observation every time a responder app triggers the event. The routing engine uses the data forwarded in the observation to generate future routes avoiding closed streets. Road closures can only be for public vehicles (that is, evacuation routes) or for all vehicles (including emergency and first responders).

10.1.2. Routing Engine Setup

Skymantics' Universal Routing Engine, developed in the Open Routing Pilot [http://skymantics.com/2020/01/15/building-a-routing-engine-part-1-about-ogcs-open-routing-api/], is compatible with three different

dataset formats, two algorithm families, in three different modes, and two different preferences.

The routing engine for SCIRA Pilot uses *Open Street Map* (OSM) dataset (maps of the states of Missouri and Illinois) to extract the map data and build the graphs for routing in the Greater St-Louis metropolitan area. OSM datasets are well maintained by a large community of users and they offer a great level of street capillarity. On the other hand, they lack some details on traffic rules. For the purposes of the SCIRA Pilot, it was considered that the pros outweigh the cons.

The algorithm chosen for the SCIRA Pilot was *Dijsktra with Contraction Hierarchies*, which is an optimization method using bi-directional Dijkstra algorithm as the basis that greatly increases the computation speed. In order to improve the accuracy, the street hierarchies were flattened to only 5 levels, with motorways and primary streets at the top level and non-vehicle streets at the lowest. In this manner, pedestrian segments of the routes will be considered for the start and end as a native part of the route calculation, but the routing algorithm will ignore them for the rest of the computation.

The specs for the SCIRA routing engine required some performance improvements of the existing implementation, in particular for the use case “Request evacuation route”, as it needs to calculate several alternative routes in a reasonable amount of time. The main effort focused on the data preprocessing, generating shortcuts linking lower hierarchy streets with higher hierarchy ones and shortcutting high hierarchy nodes. Additionally, the code generating the GeoJSON-based Route Exchange Model was optimized. Overall, at the end of this optimization process, the route generation was about four times faster and the generated route output about half its size than the initial results and routes were fully processed in between 0.2 to 0.5 seconds, which is considered a reasonable amount of time for this pilot.

The algorithm to calculate evacuation routes uses Dijkstra with Contraction Hierarchies to calculate the optimal route, in the same way as calculating any other route, but it then continues the route computation until it has found enough sufficiently-different alternatives or completed enough route computations. The algorithm then selects randomly one route and sends it back to the client.

Skymantics’ Universal Routing Engine has a built-in feature to provide obstacles that should be avoided when generating a route. This feature has been adapted for the “Close a road” use case. The observations received from the SensorHub are transformed into obstacles and kept persistent in the routing engine. The road closures can be only for public vehicles (in which case, they only affect the computation of evacuation routes) or for all vehicles (in which case, they affect all route computations).

Below is a simple example to visualize the operation of the routing engine. More complex examples were designed for scenarios 2 to 5 in the SCIRA Pilot MSEL.

The following image (Figure 63) shows the context to manage: a street segment (blue square) has been closed for public vehicles and another street segment (red square) has been closed for all vehicles. An emergency unit (yellow dot) needs to arrive at an incident (orange dot) and help evacuate people to destination (green dot).

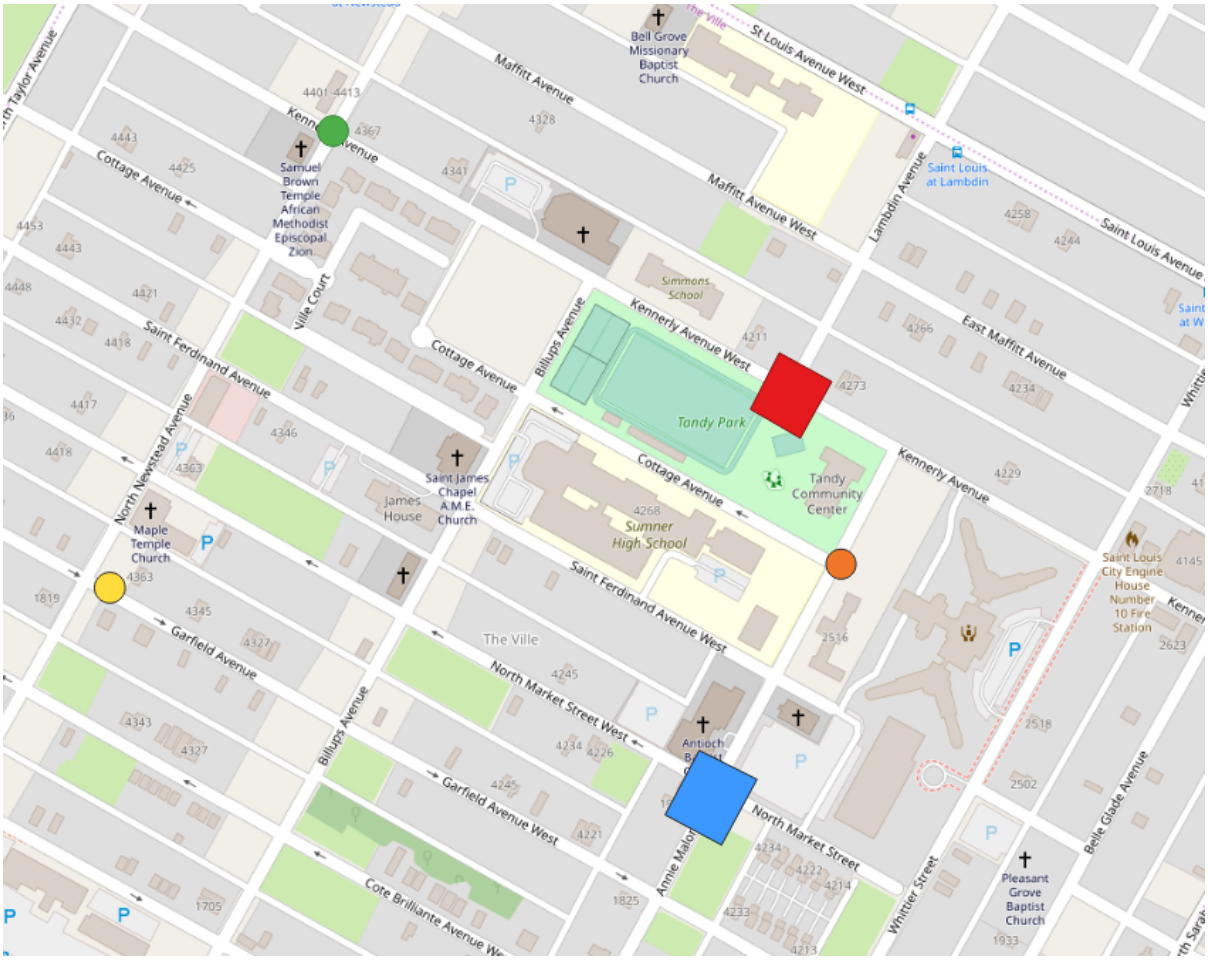

Figure 63. Traffic Management & Routing Model Example: Context

The emergency unit requests a route. As the closed street segment on its way is only blocked for public vehicles, it is routed through it (Figure 64).

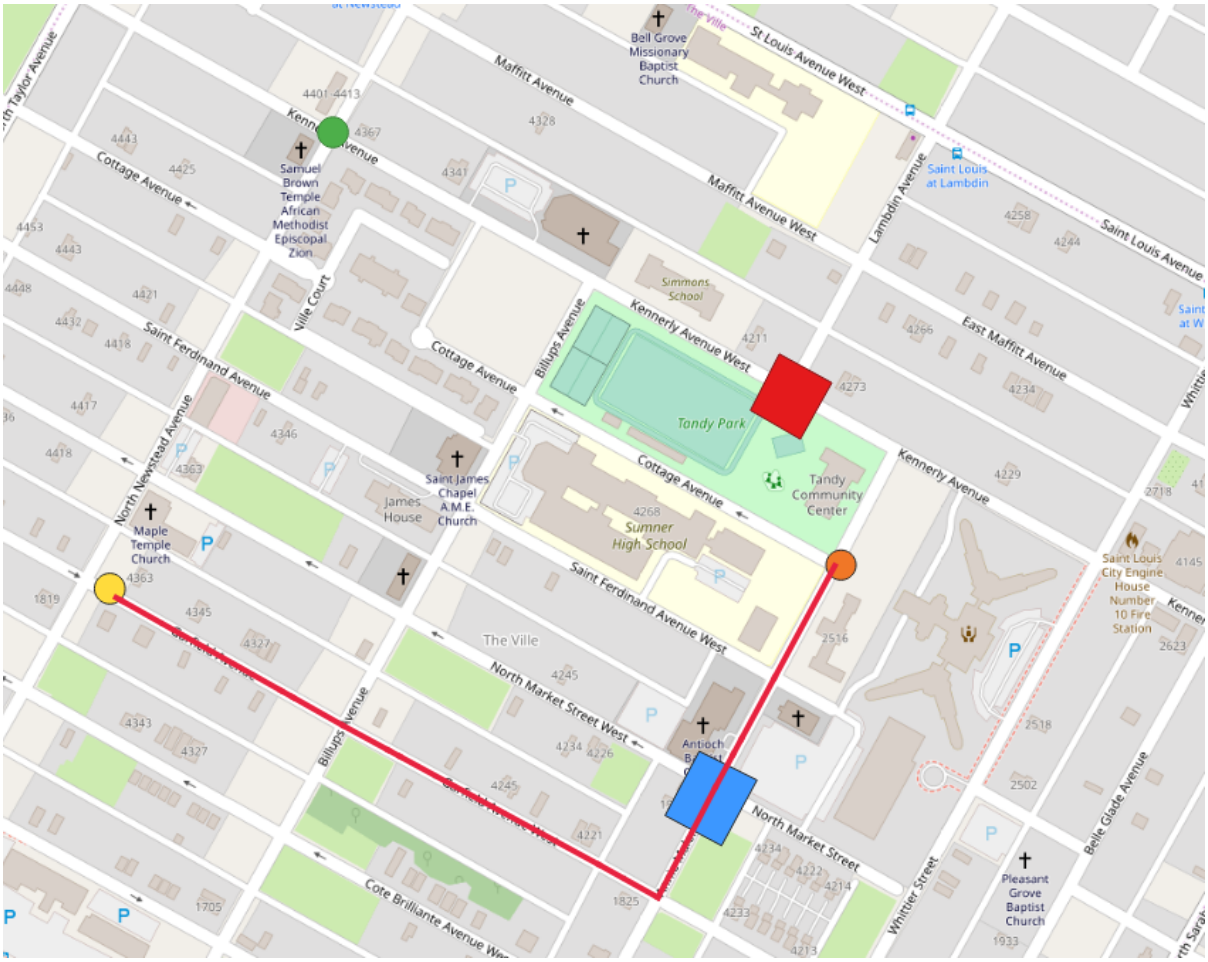

Figure 64. Traffic Management & Routing Model Example: Emergency Routing

From the incident spot, citizens request for evacuation routes to the destination. Three different routes are generated and randomly assigned to different citizens (Figure 65). Note that no route going through any of the closing streets is generated.

Figure 65. Traffic Management & Routing Model Example: Evacuation Routing

10.1.3. Extension to Routing API

The endpoint deployed for SCIRA is available at: http://skymanticswps.eastus.azurecontainer.io:8080/scira/

Specs of the original Routing API can be found in SwaggerHub: https://app.swaggerhub.com/apis/cportele/wps-routing-api/1.0.0

The following extensions have been added to the Routing API:

- POST /routes:
  - New parameter "context", with three possible values:
    - "standard": default value. It generates a standard route, without additional rules
    - "emergency": It generates a route for an emergency vehicle, taking into consideration road segments and maneuvers only allowed for emergency vehicles
    - "evacuation": It generates a series of possible routes from A to B and randomly selects one
- Route schema
  - New property "transport" in route-segment features, with two possible values:
    - "vehicle": default value. Indicates that the route segment is for vehicles

- "foot": Indicates that the route segment is for pedestrians

Here is an example of a request to compute a route. Note that for this pilot, the routing engine's algorithm is hardcoded to "dijkstra-ch" and the preference is hardcoded to "fastest", thus, they do not need to be specified.

```
{
  "name": "Test Route",
  "waypoints": {
    "type": "MultiPoint",
    "coordinates": [
      [
        -90.286654,
        38.566002
      ],
      [
        -90.219345,
        38.663900
      ]
    ]
  },
  "dataset": "SCIRA",
  "context": "evacuation"
}
```

10.1.4. Road closures

In order to receive the information on road closures, Skymantics' routing engine connects to the Compusult SensorHub (available at ogc-hub.compusult.com, port 1883), using MQTT protocol, and subscribes to datastream 232.

A new GeoJSON observation type, Street Closure, is defined for the SCIRA Pilot, with the following structure:

```
• obsType ("streetclosure")
• id
• timeStamp
• Params
  - closureType (public, all)
  - action (close, open)
  - closureReference (ID of closure to remove or None)
  - location
    * type (Feature)
    * geometry
      - type (Polygon)
      - coordinates [ [ [long, lat], [long, lat], ... ] ]
```

When the SensorHub receives a road closure event, it is forwarded as an observation to all

applications and devices that were properly subscribed to the datastream, such as Skymantics' routing engine, which then apply this information for future route calculations.

10.1.5. Main Findings and Recommendations

Here are the recommendations drawn from D15:

- The *Open Routing API* developed in the *Open Routing Pilot* has proven to be efficient, accessible and flexible, being able to adapt to specialized use cases. The first recommendation would be to continue the development and testing of this API and to find new use cases where it can be put in practice.
- One possible extension to the *Open Routing API* is related to routes with multiple transport means (in SCIRA Pilot, vehicle and by foot). This could be a research area with great potential for the API, not only to be able to specify the transport means of a route, but to apply the API for multi-modal passenger transportation (finding the most efficient transportation from point A to B, combining public and private options such as taxi, bus, tram, metro, ferry, train or plain) or even for logistics and freight transportation. The API and Route Exchange Model already support many of the basic elements required to do so and it would vastly expand its applicability.
- A second extension to evaluate for the *Open Routing API* (coded in SCIRA Pilot with the parameter "context") allows the routing engine to use different behaviors than standard. In the SCIRA Pilot this was applied in two use cases: 1) for emergency vehicles that, in addition to eventual traffic rules for emergency encoded in the map dataset, were permitted to go through roads closed only for public vehicles and 2) for public vehicles requesting an evacuation route selected randomly from a pool of possible options. A deeper evaluation of this extension would allow the *Open Routing API* to better adapt to vertical routing applications, in combination with the myriad of sensors and web applications offering information that could affect a route calculation.
- The option to review historical routes, although useful in pilots, might cause security and privacy issues if not properly implemented. Further revisions of the API should take it into consideration.

10.2. D17: Community Mobility Navigation, Alert & Sensing App

This deliverable is linked to the "Road Sensing, Flood Sensing, and Command Communication Systems" as well.

Mobile applications were used to collect observation data regarding street closure observation. The following attributes are collected during a street closure observation.

- obsType (streetclosure)
- id
- timeStamp
- point (x,y)
- radius (ft)
- closureType (public, all)
- closureAction (close, open)
- closureReference (ID of closure to remove)

10.2.1. Example of a "Street Closure" Observation


```
{
  "id": "8af56b06-66a5-42cf-82f7-d4932eac0b71",
  "timeStamp": "2019-11-18T05:00:25.013Z",
  "observationType": "streetclosure",
  "params": {
    "closureAction": "open",
    "closureType": "public",
    "location": {
      "type":"Feature",
      "geometry":{
        "type": "Circle",
        "coordinates": [-52.8047,47.5185],
        "radius":250,
        "properties": {
            "radius_units": "ft"
        }
     }
    }
  },
    "encodingType":"application/vnd.geo+json"
}
```


Participants involved in the street navigation scenarios could subscribe to alerts to get notified when road closures occurred. As observations are pushed to the SensorHub those matching the criteria would trigger an alert so that the subscribers are notified. The subscribers could then check the SensorThings results to find out details such as the locations of the closures.

Interfaced with Ronin, an API developed by Coolfire, for purposes of this scenario. Ronin is a portable platform that members of the public can use to self-report incidents, mark the locations of the incidents on a map, and post messages on the subjects of the incidents. SensorHub was updated to interface with the Ronin API developed by Coolfire, receiving both messages and geospatial data (*e.g.*, the location of a road closure) directly from the Coolfire Ronin webpage and translating them into SensorHub observations. The intent of this API was to permit Ronin users to forward data through to the SensorHub to be displayed and trigger alerts.

10.3. D20: 3D City Model

This deliverable is linked to the "Road Sensing, Command Communication, and Street Navigation Systems" as well.

Chapter 11. Flood Sensing System

The Flood Sensing system provides both predicted and actual flood inundation regions and street blockages, summarized in the following wiring diagram (Figure 66):

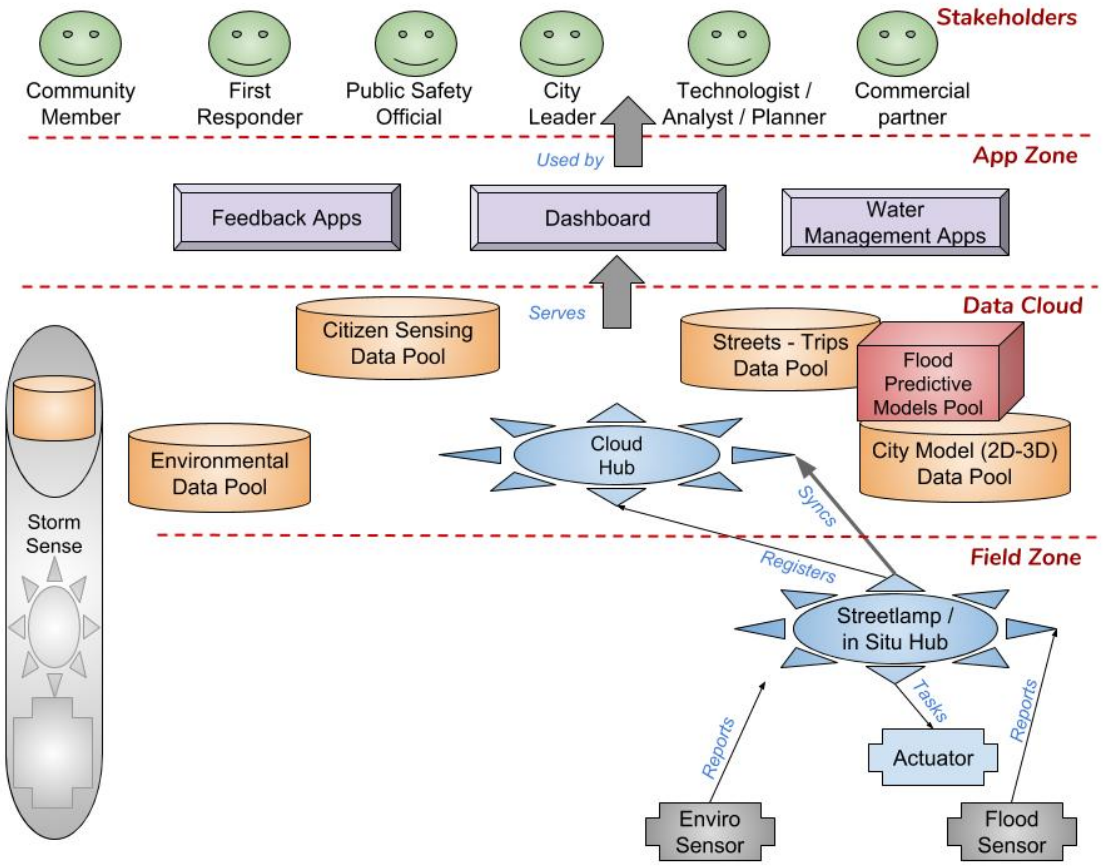

Figure 66. Flood Sensing System Overview

11.1. D11/D16: Legacy Stream Sensors

Flood models require access to real-time stream gauge (alias stream gage) data, providing stream height and discharge measurements. Also, ready access to such data can assist first responders by providing immediate awareness of potential flooding conditions along both major rivers and minor streams and channels. Such data for all of the United States (US) is provided online through web-based access for 10,000+ stream gauges. This is a very valuable collection of real-time and archived data.

Unfortunately, the USGS (United States Geological Survey) stream gauge data is supported by a custom solution that requires using a new method for finding, accessing, and parsing the relevant data. One of the benefits of OGC standards is that they provide a common language for requesting and receiving sensor data, as well as describing the sensors and sensor systems. An OpenSensorHub (OSH) adapter allows us to support this valuable collection of data, through OGC standard and web service interfaces that improve interoperability between the stream gauge data and other similar or disparate sensor observations (*e.g.* measured and predictive weather measurements, ship/boat navigation data, ocean and lake buoys, soil moisture monitoring etc). Using OGC standards also complements the existing system without having to replicate data and establish a completely new web capability.

Such adaptions to legacy sensor systems allow the SCIRA Pilot to quickly and effectively enable other data sets in NYS, such as traffic cameras, AVL, mesonet weather networks, ocean buoys and lake buoys. Having all of these datastores speaking a "common language" increases interoperability. For example, stream gauges can get relevant alerts about potential flooding in a localized area, while traffic and security cameras can allow the user to check the streets in the area for initial flooding. At the same time, a moving school bus can be notified of a potentially serious hazard involving school children in the flood zones.

Figure 67 indicates the location of stream gauges (measuring river, stream, and channel flow conditions) provided by the OSH within the area waterways.

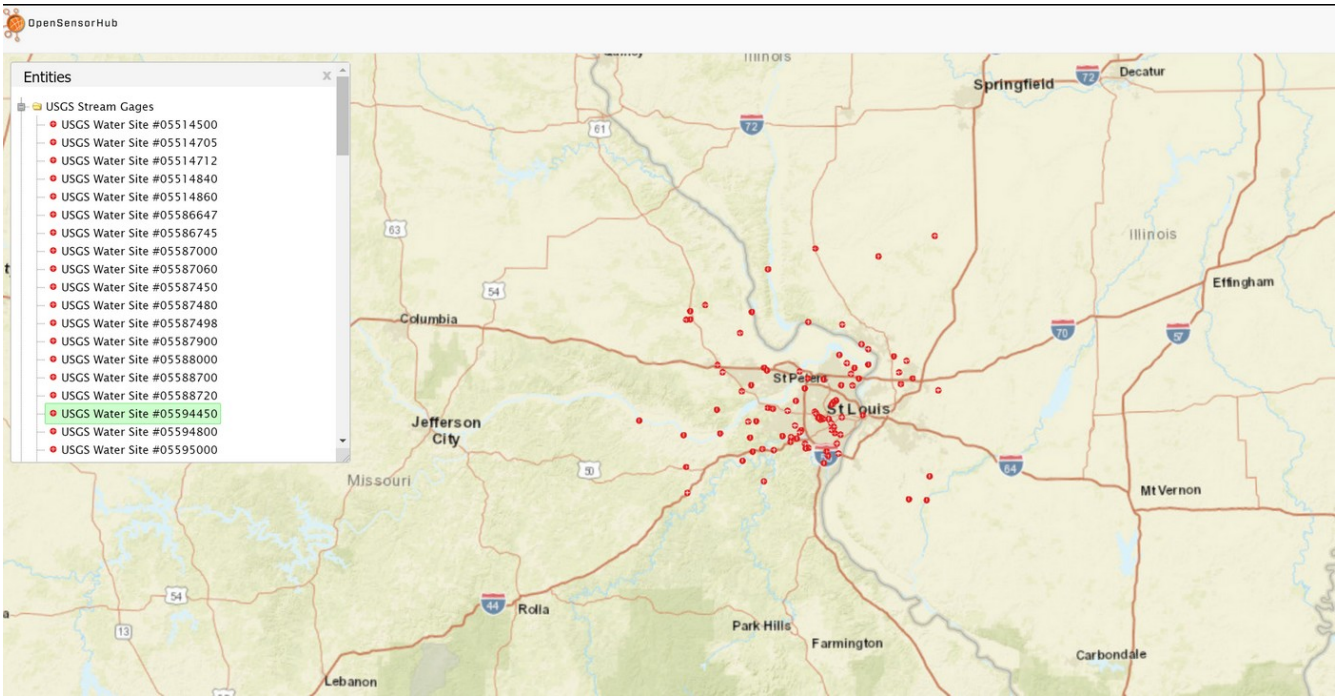

Figure 67. Stream Gauge Locations around St. Louis as Provided by the OSH Web Services

Each of these sensors provides new data every 15 minutes. These real-time and archived observations can be requested from Web services based on OGC standards. The sensor observations can be displayed on favourite visualization or processing applications, as shown below (Figure 68 and Figure 69).

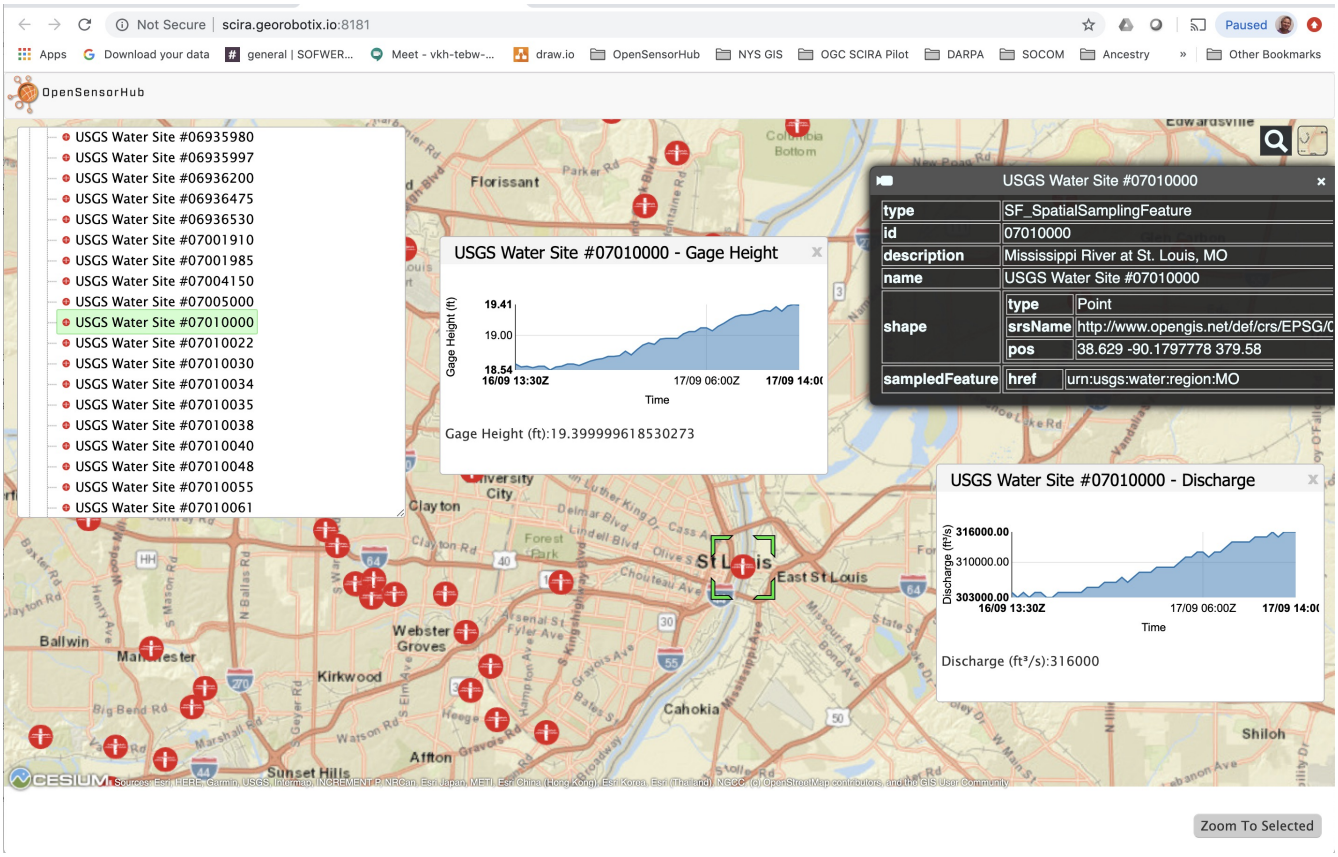

Figure 68. Stream Gauge Data for the Mississippi River at the Arch

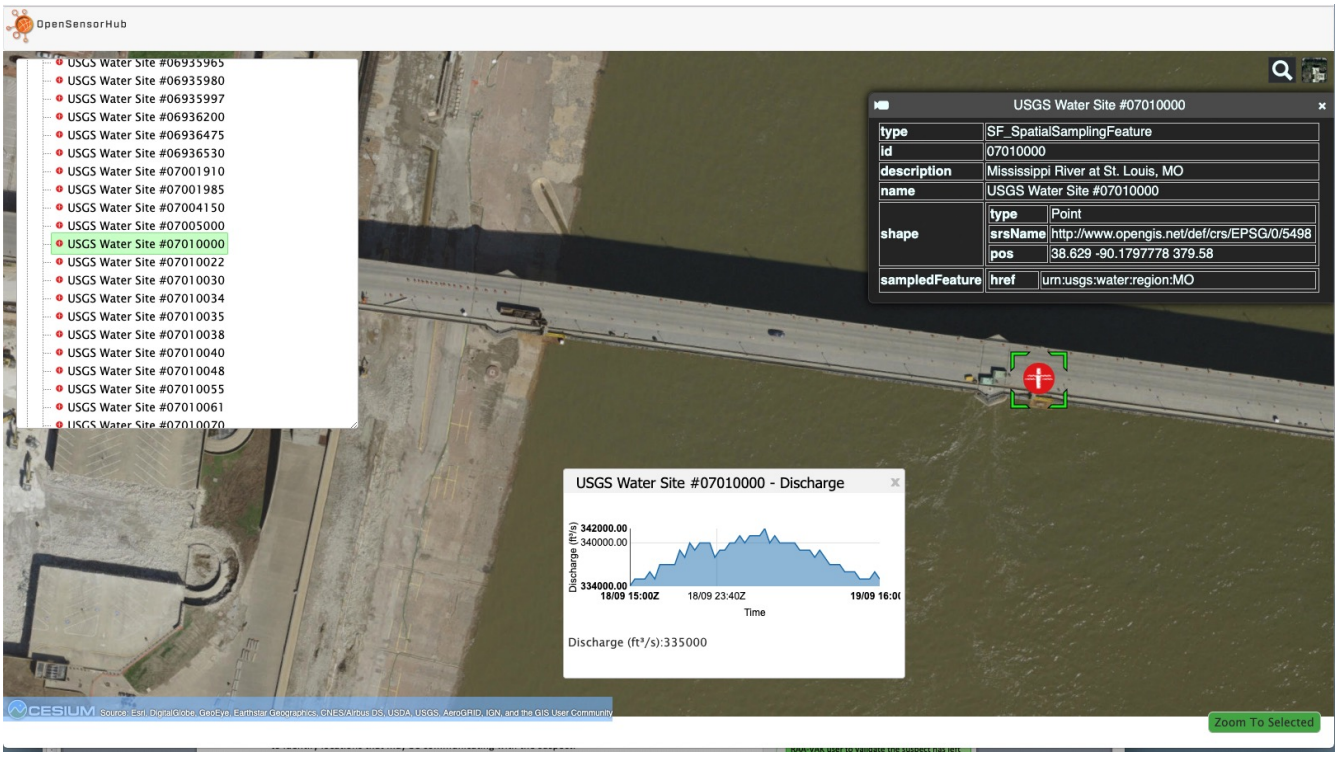

Figure 69. Stream Gauge at the Martin Luther King Bridge

The stream gauges also report observations at more minor waterways, such as the gauge at Tenmile River shown below (Figure 70).

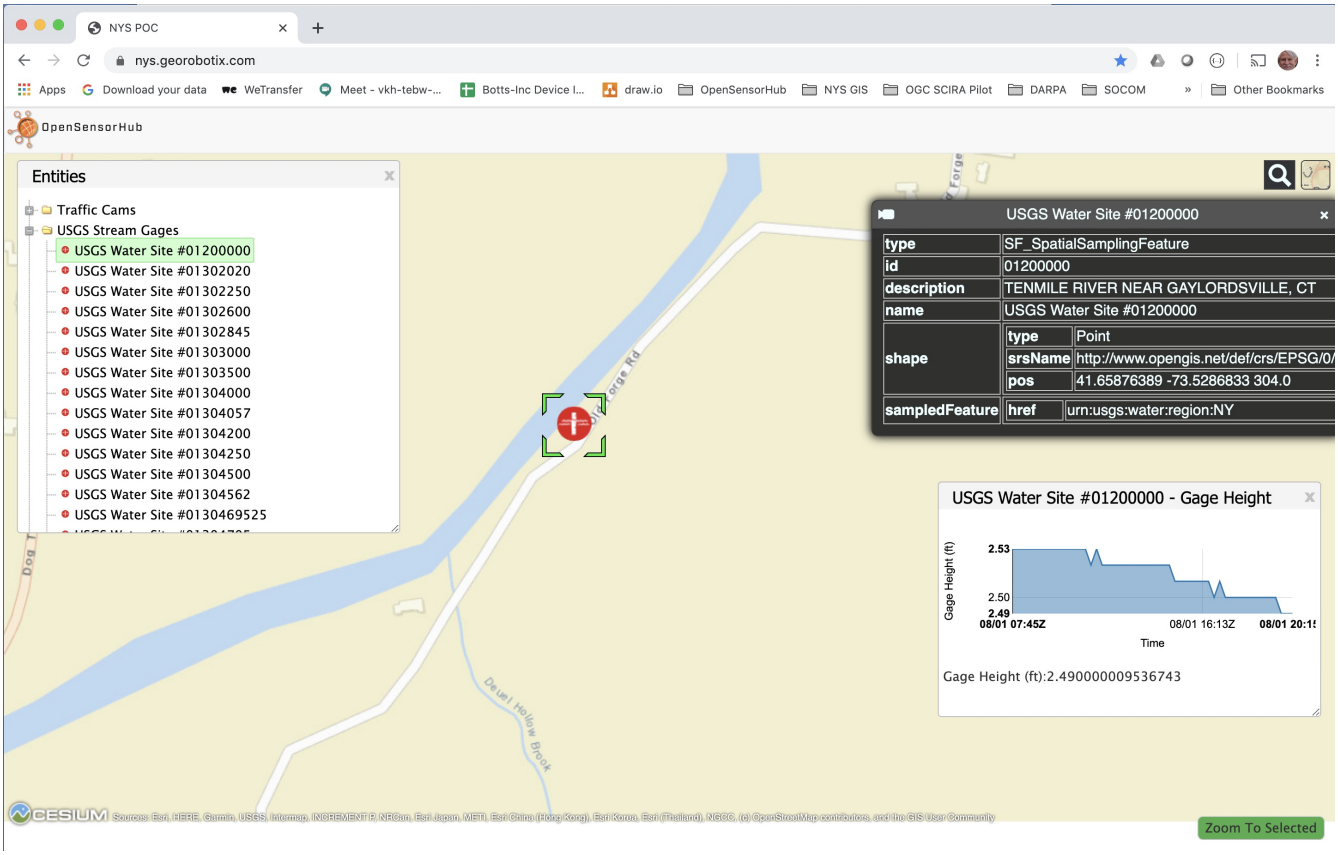

Figure 70. Stream Gauge at Tenmile River

The stream gauge data is very useful for visualizing the water level in the stream, to quickly determine whether the stream is rising or subsiding. The data provided by the OGC Sensor Hubs (*e.g. Compusult SensorHub*) can also be requested by flood models to be fed into the processing workflow. These stream gauge observations served as input data to the flood models described in the next section.

11.2. D14: Flood & Inundation Model

The Flood and Inundation model is a system/user-driven model that would be able to simulate the scenario, visualize the possible impact of the flood and derive predictive analysis. This model provides a framework for deriving the response matrix and preparedness plans in place. Frequent rains put an abundance of land stress-causing inundation in areas that are prone to flooding. The data flow in the Flood and Inundation model is shown in Figure 71.

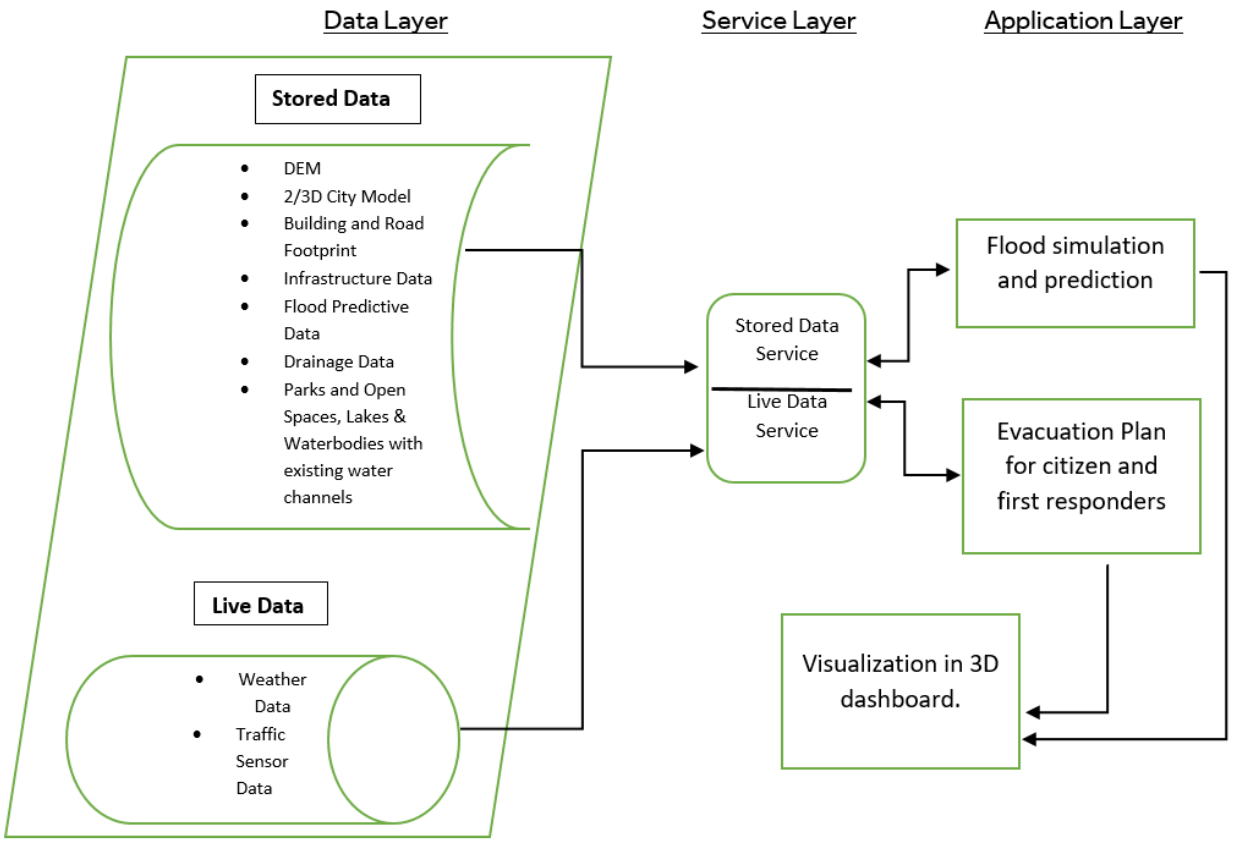

Figure 71. Flood Data Flow in the Flood and Inundation Model

Benefits of the Flood & Inundation model in the SCIRA Pilot are listed below:

NOTE

Benefits of the the Flood & Inundation Model

- Benefits for end-users:
  - First responders are directed towards the building via the safest and most rapid routes;
  - The community can be evacuated or diverted away from the area with minimal congestion.
- Benefits for developers:
  - Ensure specifications are truly interoperable for future standards and product releases;
  - Discovery of possible improvements to the OGC standards;
  - Development of new algorithms or applications for water flow direction using the Pour Point Model;
  - Lessons learned can assist with future research and reduce the learning curve.

11.2.1. Load Data

This section describes the data that is loaded for the Flood model to work. This model uses several data sources for testing including Digital Elevation Model (DEM)/Digital Terrain Model (DTM), OGC® CityGML [https://portal.opengeospatial.org/files/?artifact_id=47842] 3D Model, *Line* feature data, *Point*

feature data, *Polygon* feature data.

Pre-loaded data is required to perform flood modeling through Cesium ion [https://cesium.com/cesium-ion/]. During the flood model prediction analysis, OGC® 3D Tiles [http://docs.opengeospatial.org/cs/18-053r2/18-053r2.html] helps to provide better visualization. For the benefit of end-users, the visualization is color-coded based on the effect of flood buildings (while considering each tile as a building).

Visualizing 3D Tiles

To visualize a 3D City Model, the OGC® CityGML [https://portal.opengeospatial.org/files/?artifact_id=47842] encoding is converted to OGC® 3D Tiles [http://docs.opengeospatial.org/cs/18-053r2/18-053r2.html] and is stored and streamed to maps. Figure 72 shows the conversion process and Figure 73 shows visualized 3DTiles.

Figure 72. Conversion Process of CityGML to 3D Tiles

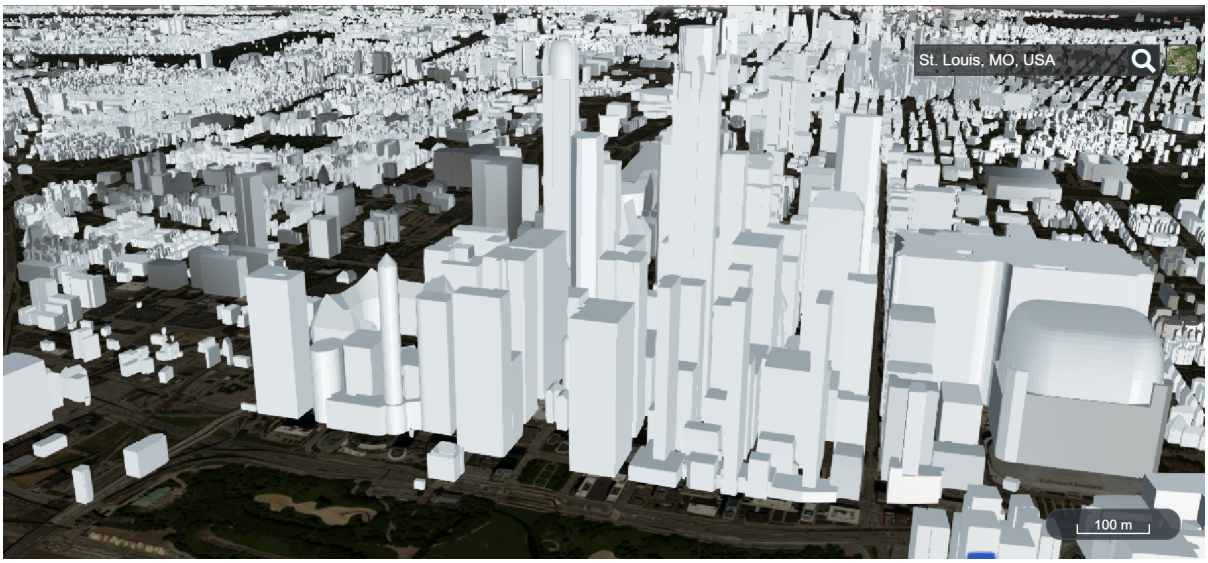

Figure 73. Visualizing 3D Tiles

The boundary of the City of St.Louis is incorporated for a better view and also to highlight the area of interest(Figure 74).

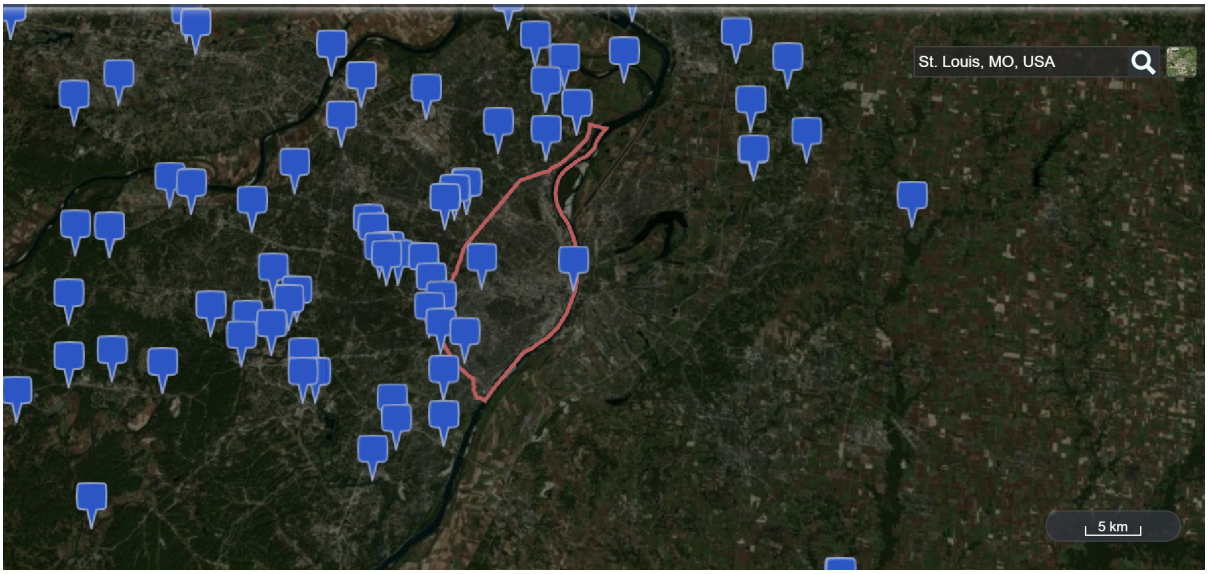

Figure 74. Visualizing Boundary and Infused Stream Gauge Data

11.2.2. Flow Direction Algorithm

The Flow Direction algorithm for the area of interest is based on a DEM/DTM. The Pour Point Model is used to show how and which direction water travels [1].

The below figure (Figure 75) shows the terrain data flow to delineate the Hydro model.

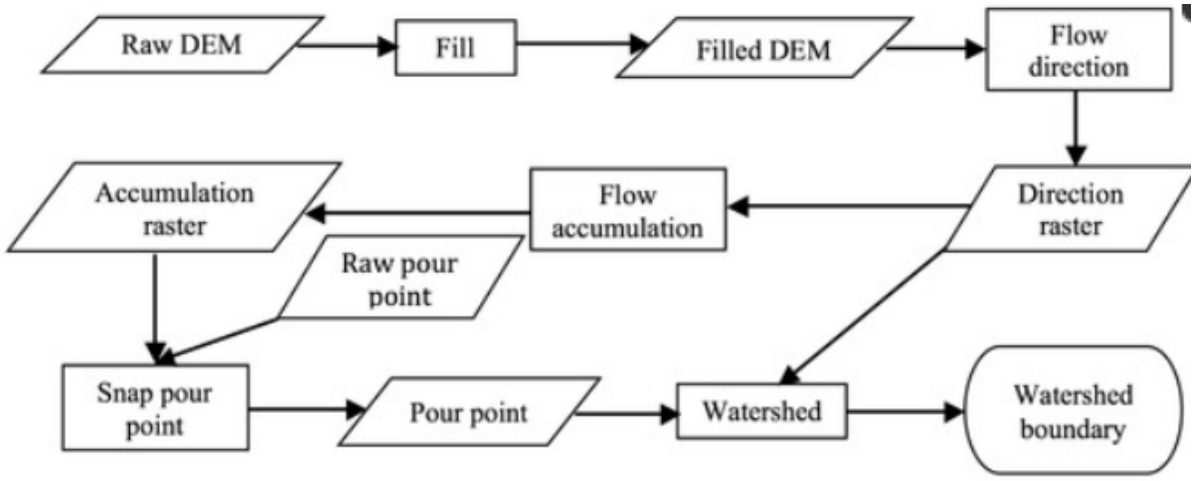

Figure 75. Terrain Data Flow to Delineate the Hydro Model

All eight adjacent cells in the Pour Point model have a value expressing how water flows as in the following figure (Figure 76):

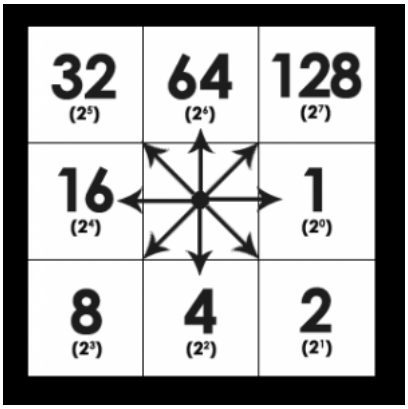

Figure 76. Pour Point Model

In this algorithm, we create two grids. The first grid contains the flow direction from each grid cell to one of its adjacent or diagonal neighbors, calculated using the direction of steepest descent. The second grid contains the slope, as evaluated in the direction of steepest descent, and is reported as drop/distance (*i.e.* tangent of the angle). In flat areas, flow directions are assigned away from higher ground and towards the lower ground using the method of Garbrecht and Martz [1].

Figure 77. Accumulation Points and Flow Directions

In the above figure (Figure 77), accumulation points (represented in red) are the lowest elevation value cells from which there is no outflow of water to any adjacent cells where water accumulation is anticipated.

11.2.3. Alerts and Model Output

The Flood model subscribes to the alerts from the *Compusult SensorHub*. As soon as the model starts in the background, an automated process runs for subscribing to flood related topics using MQTT [https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html] (Figure 78). MQTT is an open lightweight publish subscribe [https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern] network protocol that transports messages between devices.

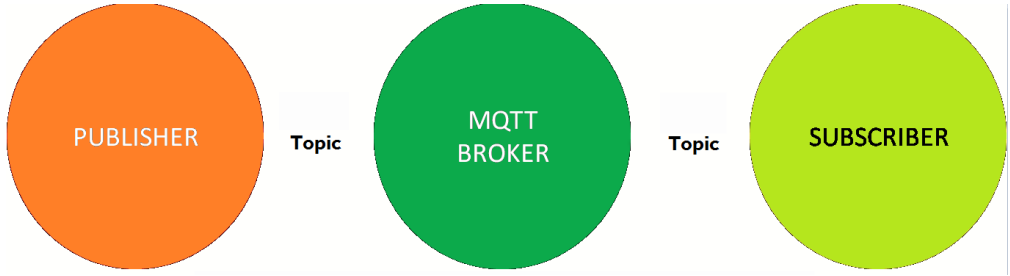

Figure 78. MQTT Protocol for Flood Alerts

In this case, the subscriber is the Flood model and the publisher is the *Compusult SensorHub*. The model subscribes to two topics (*i.e.* FloodTrend for river monitoring and FlashFlood for flash flooding). When the *Compusult SensorHub* publishes alerts to the Flood model based on the topic and type of the alert's message/body, the model is trained to run on the extent of an area of interest. The model considers sensor observations, weather predictions and *etc.*

The output generated is pushed to the *Compusult SensorHub* datastream (datastream ID: 230) in a JSON encoding as presented below:

JSON Encoding Results:


```
\{"attributes": \{
  "noOfBuildingsImpacted":"157",
  "areaImpacted":"689723",
  "cameraId":"16",
  "encodingType":"application/vnd.geo+json",
  "feature":\{
    "type":"Feature",
    "geometry":\{
      "type":"Polygon",
      "coordinates":[
      [[-90.3010082244873,38.647377488268724],[-90.30418395996094,38.63356712505498],[
-90.26839256286621,38.62974534092597],[-90.26538848876953,38.64382459083146],[-
90.3010082244873,38.647377488268724]]
      ]
    }
  }
}
}
```


11.2.4. Flood Working Model

The Flood Working model leverages sensor observations, weather predictions, and 2.5/3D landscape data to make on-going predictions of timing, extent, and severity of stream flooding and land surface inundation. In the SCIRA Pilot, providing and maintaining emergency routes safe from flooding was used by the community that must travel or evacuate during the event. The Flood Working model becomes a standing model for emergency management during the SCIRA Pilot. Also provides an OGC compatible interface (*e.g.* OGC® SOS [https://portal.opengeospatial.org/files/?artifact_id=47599], STA [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html], and WPS [http://docs.opengeospatial.org/is/14-065/14-065.html]) and interoperates with the *Compusult SensorHubs*.

The Flood Working model, developed on hydrodynamics, simulates the flood scenarios based on user inputs and also runs the results on the sensor infused data.

- The model considers the extent drawn either by the free-hand method or pre-defined in the area of interest, which is further divided into equal parts.
- The water-logging is simulated by the algorithm based on terrain elevations, flow direction and sensor data.
- Affected buildings and impacted areas are detected by calling the chain of functions that performs analysis for the area with flood water-log and buildings submerged.
- The simulated output is displayed on the map showing the affected area (Figure 79).

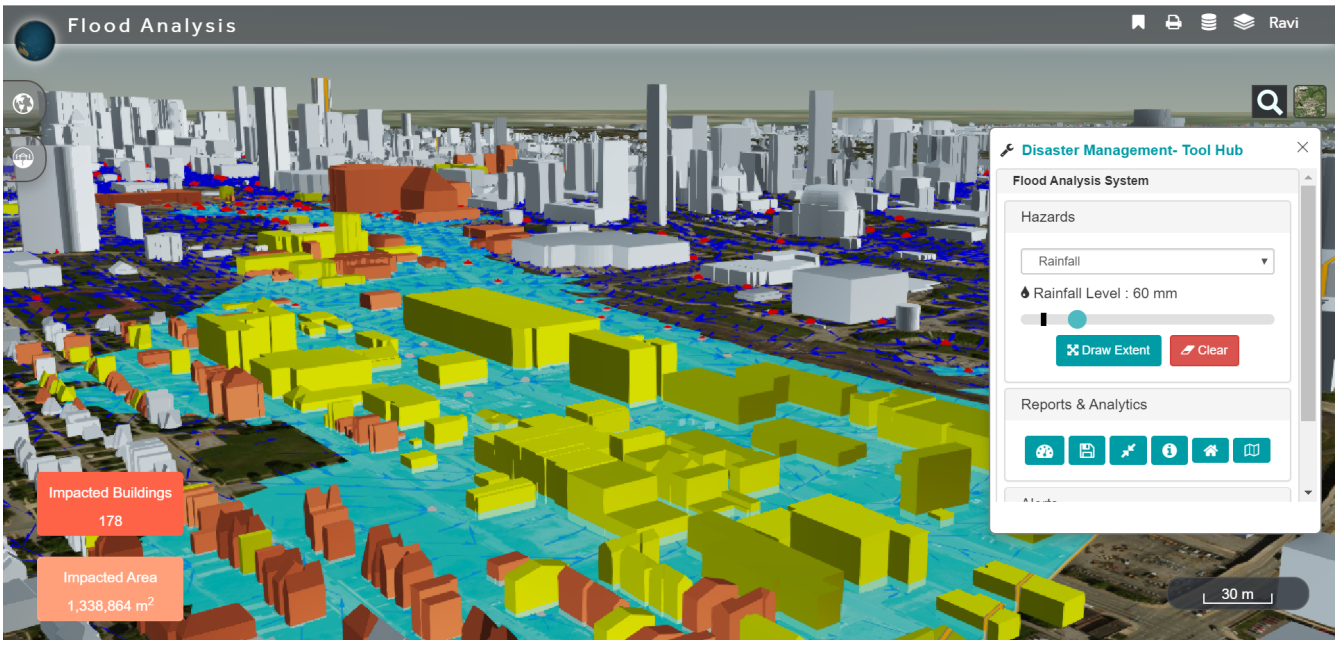

Figure 79. Flood Simulation Output

The color coding of buildings helps to identify the affected buildings for different rainfall levels. This predictive model output is pushed to the *Compusult SensorHubs* and displayed in the dashboard. This acts as a foundation for the City's preparedness plan for flood events by all stakeholders (*e.g.* City officials, rescue teams, and public).

Rainfall Forecast

Data is fetched from an open ource weather API, which is further used to forecast rainfall for the coming week beforehand (Figure 80). This forecasting acts as a base for a flood predictive model.

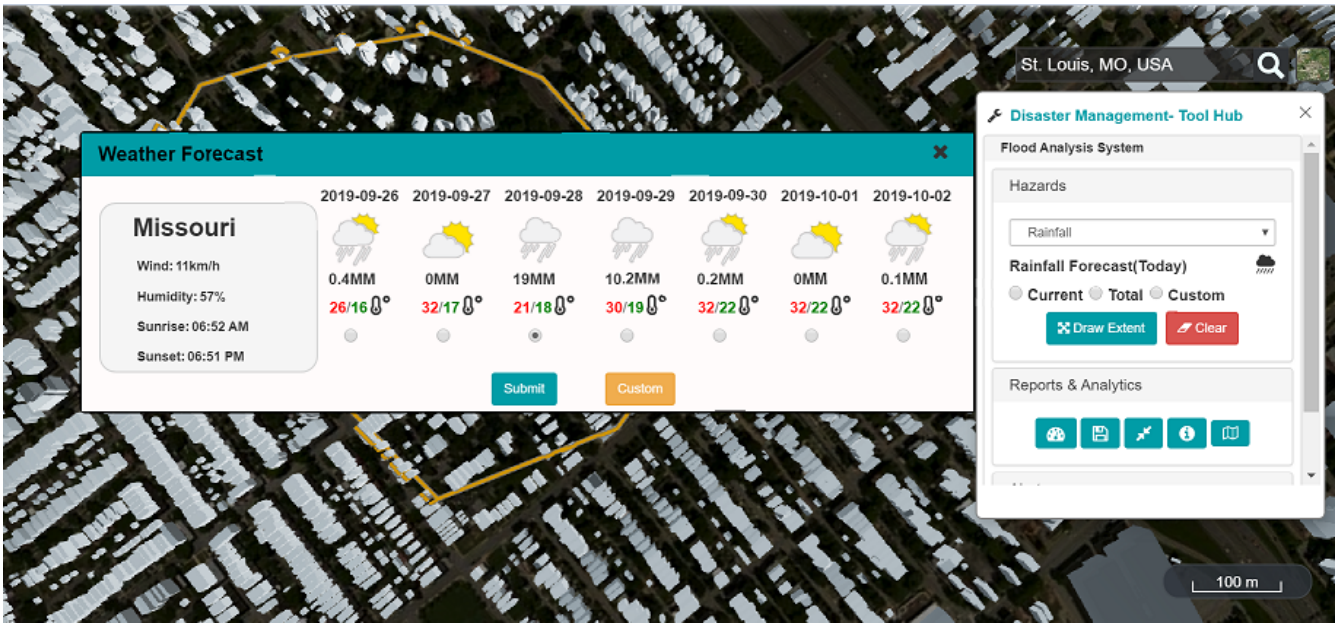

Figure 80. Weather Trend for a Week

Trend and Predictive Analysis

Trend analysis is another important step for decision making. Using this, the user can observe the stream gauge readings for past trends. Flood prediction is also based on trend analysis.

Predictive analysis is another algorithm, which uses historic stream data for a period and predicts the stream readings used as an input for the predictive flood model.

Figure 81 shows the stream gauge data trend for the past four days in a 24-hour time format with respect to gauge height. This gauge data is provided by another SCIRA Pilot participant (*Botts Innovative Research*).

Figure 81. Stream Gauge Trends

The system considers all these parameters and performs the flood analysis on the selected extent if the flash flooding has led to a high amount of standing water on streets in and around the area.

These streets are no longer passable and need to be blocked off. The alerting system helps to alert the responders and acts accordingly.

River Monitoring

To monitor the river and anticipate the additional threat of flooding, the City needs preliminary actions while dealing with the responses on the most anticipated areas or areas of interest. Emergency management needs to monitor the flood threat as it changes due to this latest storm system and may need to alter preparation activities based on the changing threat. The river gauge sensor feeds are considered in order to monitor the changes in the river stream level. The system performs predictive analysis based on historical data and forecasts flooding over the next 24 hours (Figure 82 and Figure 83). Based on this analysis, preparatory actions can be taken including alerting all the stakeholders.

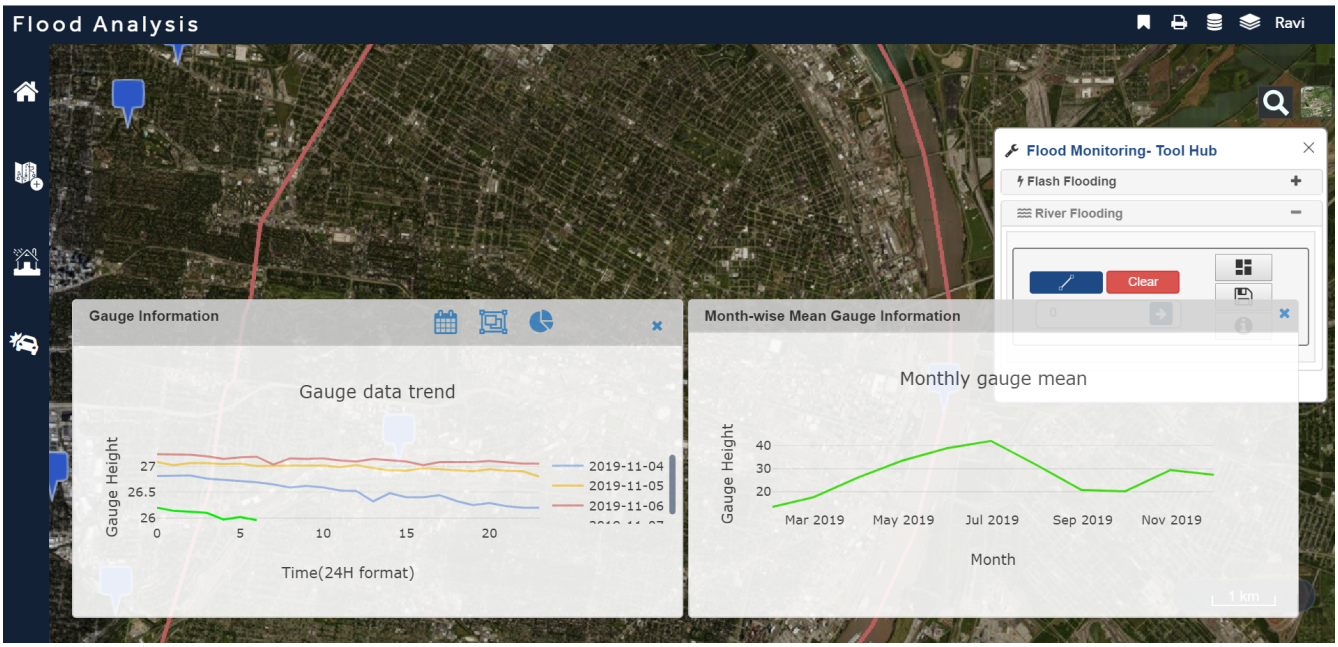

Figure 82. Monthly Gauge Mean Data Representation

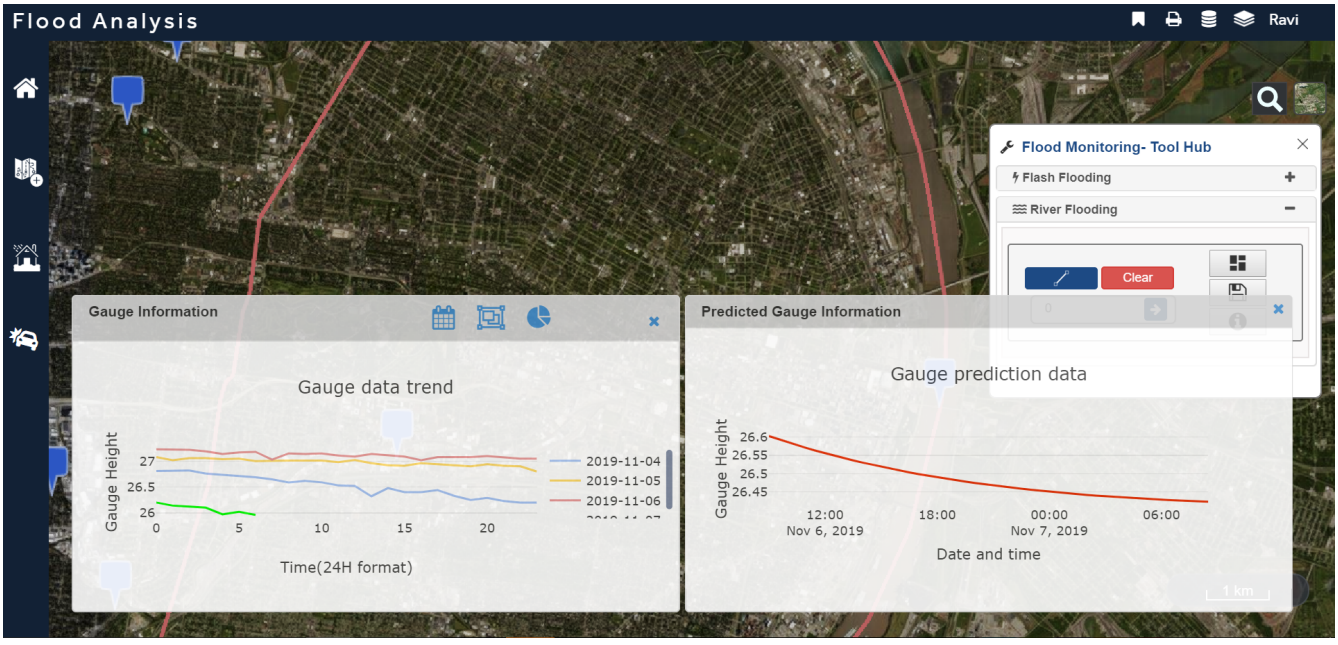

Figure 83. 24 Hour Gauge Data Representation

Based on the sensors' feed thresholds (such as stream and weather), the *Compusult SensorHub* publishes alerts to the flood model based on predictive analysis. If the system forecasts that the river is at flood stage, this is taken as an input and the flood model runs in the area of interest. Observations at points along the river are collected and the inundation is simulated as shown in Figure 84 and Figure 85.

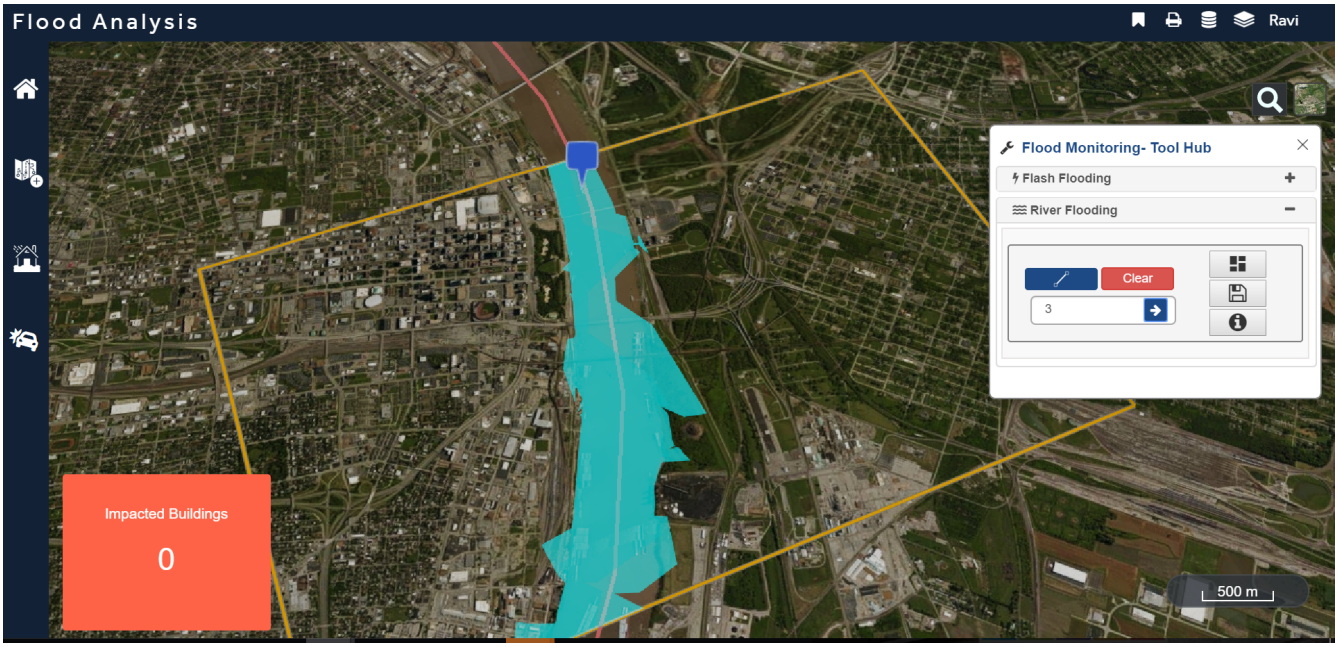

Figure 84. Selecting the Area of Interest

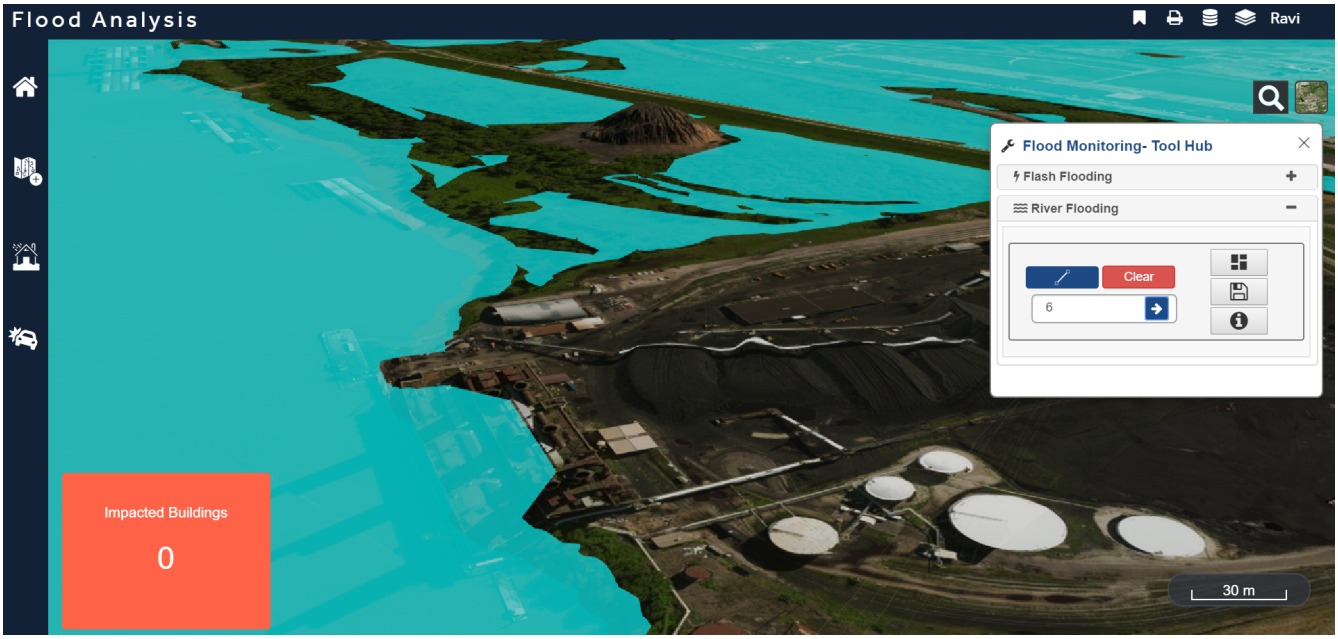

Figure 85. River Flooding

The Flood & inundation model subscribes for alerts to be published and runs on different sensors observations as the input and performs predictive analysis, flash flooding and river flooding simulations. The model also incorporated a few other datastream observations (*e.g.* camera feeds, weather observations, stream gauge readings *etc.*). The model sends its output to the *Compusult SensorHub* that can be further consumed in decision making and preparatory actions.

11.3. D17: Community Mobility Navigation, Alert and Sensing App

Mobile applications were used to collect observation data regarding flooding incidents. In the event of a flood, the following data parameters can be collected using the Community Mobility Navigation, Alert and Sensing App.

Flood Observation Parameters in a Community App

- obsType (flood)
- id
- validTime
- obsTime
- Point (x,y)
- Radius (ft)
- featureType (channel, drainage, land surface)
- obsDepth (ft)
- obsMode (meter, visual, model)

Example of a "Flood" Observation:


```
{
  "id": "8af56b06-66a5-42cf-82f7-d4932eac0b71",
  "timeStamp": "2019-11-18T05:00:25.013Z",
  "observationType": "flood",
  "params": {
    "featureType": "drainage",
    "obsMode": "visual",
    "obsDepth": 3.5,
    "obsTime":"2019-11-18T05:00:25.013Z",
    "validTime":"2019-11-18T05:00:25.013Z",
    "location": {
      "type":"Feature",
      "geometry":{
        "type": "Circle",
        "coordinates": [-52.8047,47.5185],
        "radius":250,
        "properties": {
          "radius_units": "ft"
        }
      }
    }
  },
  "encodingType":"application/vnd.geo+json"
}
```


In a community reported flood case, participants involved in street navigation scenarios can subscribe to alerts to get notified when a flood has occurred. As observations are pushed to the

Compusult SensorHub matching the criteria to trigger an alert, the subscribers will be notified. The subscribers can check the SensorThings API results to find out details such as the locations of the flooding.

Chapter 12. Road Sensing System

The Road Sensing system provides road condition and traffic awareness for traffic monitoring and as input to response or evacuation routing. The following diagram (Figure 86) shows its wiring with other subsystems.

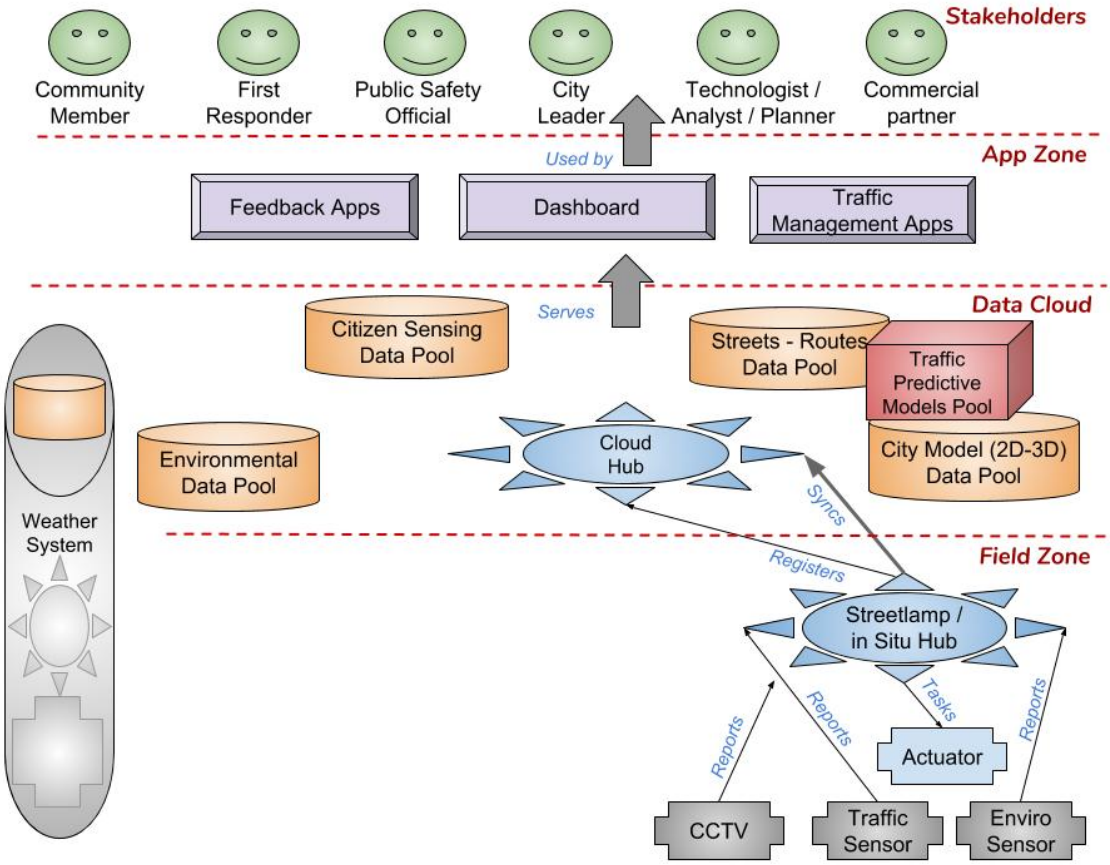

Figure 86. Road Sensing System Overview

12.1. D10: Smart Traffic/Road Sensor

In this deliverable a smart traffic/road video sensor is developed to monitor the intersections in real-time and send extracted information to the serve. In the scenario of flooding, for example, road status awareness in terms of flash flooding detection, water level estimation, and traffic count is vitally important. This information can help first responders to make better decisions and save vulnerable people more efficiently. Based on the scenario requirements, a real-time intersection monitoring system is designed and developed in D10. This system monitors the number of pedestrians, vehicles, the average speed of the vehicles, and in the case of flash flooding, it detects the water existence and estimates the level of water. Two different architectures (edge-based and cloud-based) are proposed in this deliverable (D10) to fulfill these criteria.

To develop a smart video monitoring sensor, the NVIDIA Jetson Nano Developer Kit is used. One of the key features of this kit is a powerful hardware configuration in terms of Graphics Processing Unit (GPU) and Central Processing Unit (CPU), which makes it suitable for *Deep-Learning (DL)* and *Artificial Intelligence (AI)* applications.

AI, Machine Learning (ML) and DL are often used interchangeably in the context of Data Science

and Big Data, although they have different meanings. AI is a broader concept which is used in different applications since 1950's to mimic the cognitive functions of humans. For example, Artificial Neural Networks (ANN) have an architecture similar to the human brain to recognize patterns and help computers for categorizing information. ML which is used since 1980's is a subset of AI focuses on the ability of machines to learn from data (supervised or unsupervised). The goal of a ML algorithm is to improve modeling and predictions by updating the learning capabilities. DL is another buzzword that started to be used since 2010's and belongs to the data-driven ML algorithms trained by large quantities of data. The concept of DL in this report is referred to as a *deep Convolutional Neural Networks (CNN)* which has many layers involved in an ANN similar to a nested hierarchy of related concepts or decision trees.

12.1.1. Smart Traffic Sensor System Architecture

In this section, the architecture of the system in terms of the software modules and their relationship is presented. Based on the project requirements, two architectures are developed: The first approach is based on an edge-computing paradigm. Moving all the processing load to the edge would increase the performance of the system, as it would save the required time for transferring data. Additionally, edge-computing would dramatically reduce the sending data size to the server, as it only needs to transfer the extracted knowledge. Consequently, it would also decrease the volume of high rate publishing/subscribing through the cloud network. The second architecture allocates most of the processing loads to the server side by sending all the image frames to the cloud. Consequently, devices consume less energy and perform less complicated computations. The following describes these architectures.

Edge Computing Architecture

Based on the definition proposed by Satyanaranayan *et al.* ([2]):

"Edge Computing is a new paradigm in which substantial computing and storage resources —variously referred to as cloudlets, micro data centers, or fog nodes— are placed at the Internet's edge near mobile devices or sensors."

Edge Computing restrains latency in cloud services, as the sensing and processing tasks are both conducted in devices. Plus, it provides more scalability through publishing a wide range of information from raw data. By storing data results in a local database on the edge, errors of pushing data to the cloud could be prevented and handled. In terms of fitness for use and interoperability, Edge Computing provides more resilience for SensorHub to manage data dissemination.

In the first architecture, Edge Computing is used to make a smart traffic sensor which is capable of detecting traffic and flood. Because of the extensive functionalities of Jetson Nano board, the object detection and tracking processes implemented on the edge. Also, Jetson Nano board directly posts results to the SensorHub Internet of Things (IoT) cloud. Figure 87 illustrates the general architecture of the edge computing scenario for D10. In the following figure, the Sensor layer is responsible for producing video streams and preparing the video for the edge node (Jetson Nano). All the image processing algorithms are managed in the edge layer. Additionally, this layer is the connection to the cloud services and the network. The last layer is the cloud, which publishes, subscribes and stores all IoT data of the SCIRA project.

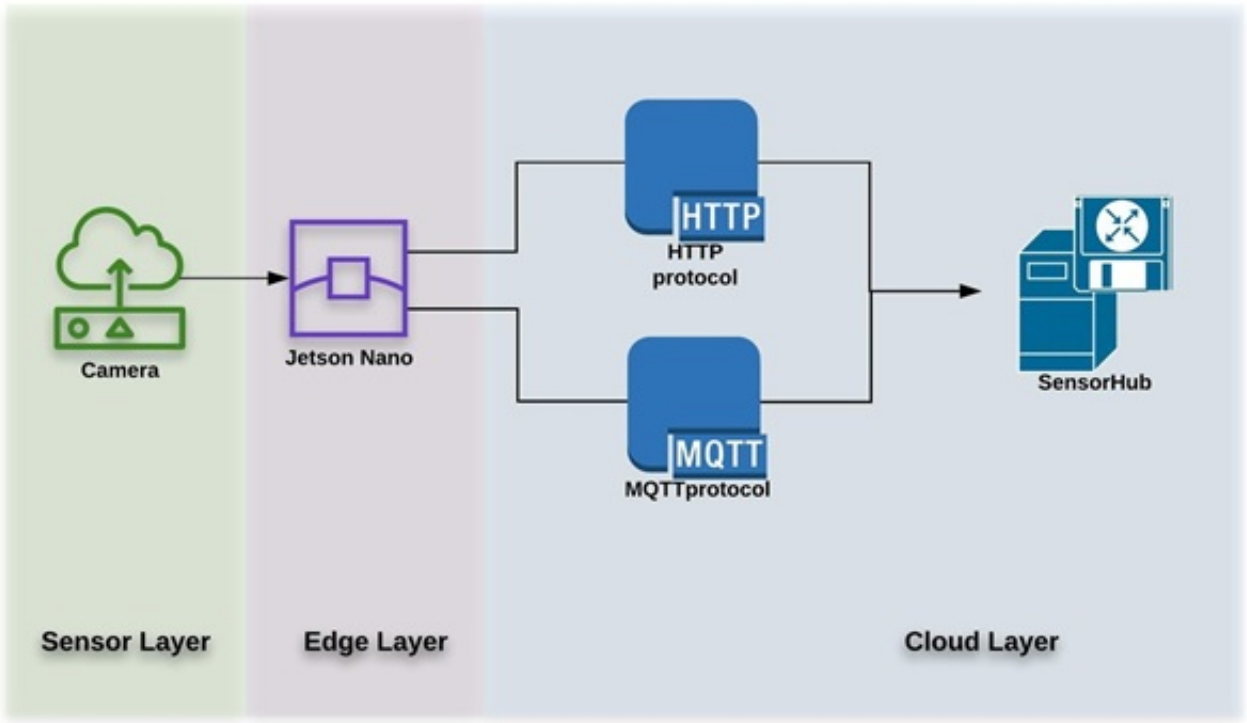

Figure 87. The Designed Edge Computing Architecture for the Smart Traffic Sensor

To fulfill the criteria of this project, two Jetson Nano devices are used. The first one (Jetson Nano #1) is specifically trained for the flash flood severity detection, and the second one (Jetson Nano #2) is used for traffic monitoring (detecting pedestrians, vehicles, and estimate the average speed of the vehicles). For the flood detection training, the sample image dataset was captured from the sideview, therefore, the DL module is able to detect flood from the side road camera. However, the traffic monitoring is based on the cameras installed on top of the intersection. The reason for having two different Jetson Nano devices is mounting them in the way that they could have two different viewpoints (i.e. side view for flood detection and top view for traffic monitoring).

Cloud Computing Architecture

To use a variety of camera sensors in this pilot, a cloud computing architecture is also designed. Mell and Grance ([3]) assert:

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

This architecture is independent of the platform and supports developers to use a wide range of libraries and APIs. Moreover, system administration would be easy as services always are accessible from everywhere. The general architecture of the cloud service is depicted in Figure 88. The sensor layer converts images to base64 format and post results to the cloud layer. The cloud layer includes a processing service, which gets images. ML module developed on the cloud analyzes the video frames and provides results for SensorHub endpoints.

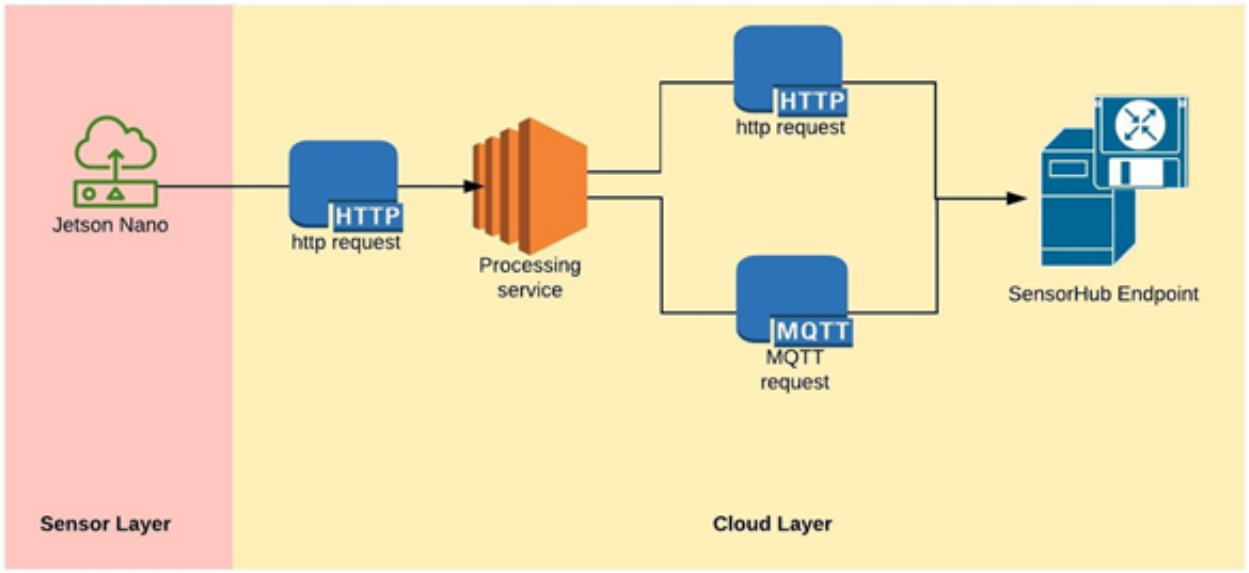

Figure 88. The Designed Cloud Computing Architecture for the Smart Traffic Sensor

12.1.2. Smart Traffic Sensor Components and Configurations

Smart Traffic Sensor consist of an NVIDIA Jetson Nano Developer Kit as the core of the sensor, various video cameras, ML algorithms libraries, and the SensorHub endpoints.

NVIDIA Jetson Nano Developer Kit

Jetson Nano is a small computer board that is capable of running multiple ANNs in parallel. This feature would make Jetson Nano a suitable option for deploying ML on the edge. Jetson Nano uses CUDA and GPU which enables developers to increase the performance of ML processes such as object detection, image processing, and etc. Its Input/Output (I/O) connectors are illustrated through Figure 89.

Figure 89. NVIDIA Jetson Nano Developer Kit Containing Different I/O Connectors

Some preliminary specs for the Jetson Nano module are mentioned in Table 3:

Table 4. Some Preliminary Specs of the NVIDIA Jetson Nano Developer Kit

Items	Features
GPU	128 CUDA Core Maxwell Architecture

Items	Features
CPU	4 core ARM A57
Memory	4 Giga-Byte (GB) 64-bit
Storage	16 GB

The Jetson Nano uses a Micro Secure Digital (MicroSD) card to hold the operating system. The Jetson Nano Developer Kit SD-Card image file can be downloaded from here [https://developer.nvidia.com/jetson-nano-sd-card-image-r3221] and flashed to the microSD card. There is a great tutorial on NVIDIA website to setup and boot the Jetson Nano at this link [https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#intro].

The next step would be installing the dependencies by the following codes:

```
sudo apt-get update
sudo apt install python3-pip

sudo pip3 install Cython
sudo pip3 install opencv-python
sudo apt-get install python-opencv
sudo pip3 install scikit-image
sudo pip3 install pandas
sudo pip3 install pillow
```

One of the benefits of using Jetson Nano is using pre-installed CUDA, which is a parallel computing platform designed by NVIDIA. Through using CUDA, NVIDIA's graphic card is employed, which will accelerate the computationally-intensive tasks, and as a result, the execution time will decrease significantly.

Camera Input

The performance of the sensor is tested using video frames captured by a Logitech Webcam connected to a USB port of the Jetson Nano. hardware features of this webcam are illustrated in Table 4.

Table 5. Hardware Features of Logitech Webcam

Items	Features
Maximum resolution	1080 p/30 frames per second (fps) - 720 p/30 fps
Focus type	autofocus
Field of view	78°

Deep Learning Module

For this pilot, YOLO is used as the core of ML algorithms for detecting pedestrians and vehicles, and later, it is trained for detecting standing water on the street and estimating the level of water. YOLO is a fully convolutional ML neural network (NN), designed for detecting objects in real-time. YOLOv3 is the latest version of YOLO (at the time of this report), designed by Redmon and Farhadi ([4]). In this version, the number of hidden layers of the model is increased. This has improved its accuracy, and yet it is still fast and suitable for real-time object detection. The original implementation of YOLO is in the Darknet framework [https://github.com/pjreddie/darknet], which is an open-source neural network (NN) framework, written in C and CUDA, and it supports both CPU and GPU computations.

Traffic Monitoring Methodology

Traffic Monitoring is used to detect pedestrians, vehicles, and estimate average traffic speed. The proposed method is a combination of YOLO object detection, Kalman Filtering and object correlation. After tracking and identifying every object in consecutive frames, every object is labelled. Then, labelled objects are used for speed estimation, loop detection, and closure detection.

A brief scheme of the Traffic Monitoring Methodology is presented in Figure 90. This procedure consists of several modules. First, an image is captured from the camera and sent to the object detector to detect vehicles and pedestrians. These objects are passed to the object tracker, and this module would assign a unique id to each vehicle and track them in the following frames until the object exits the frame. Then the location of each identified vehicle is used for estimating the average speed of each of them. There are two scheduler managers for triggering the HTTP request builder for pushing the results to the SensorHub.

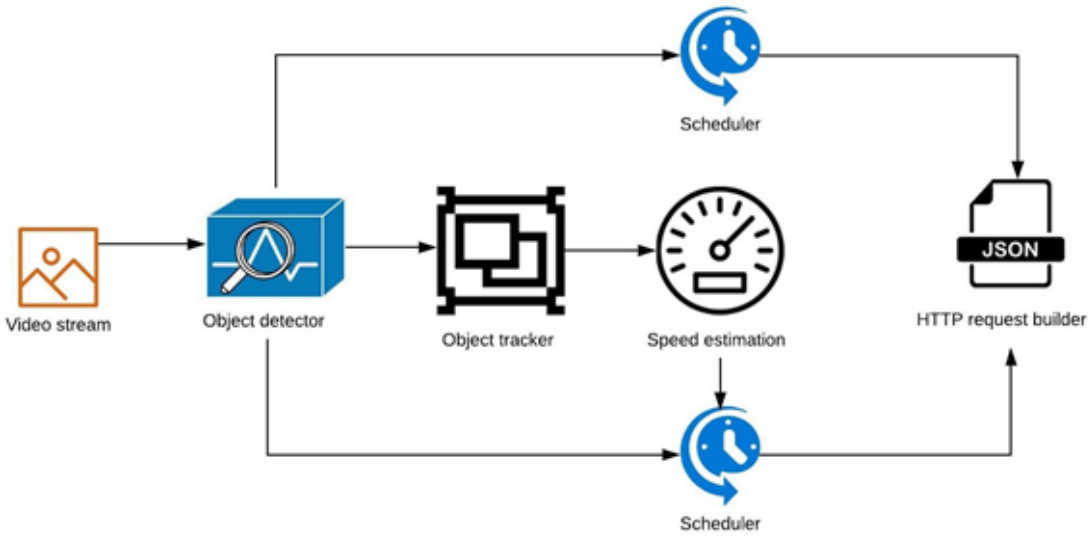

Figure 90. The General Scheme of the Process for Calculating the Average Speed of Each Vehicle, Counting the Number of Pedestrians and Vehicles per Frame

- **Object Detection:**

A pre-trained YOLOv3 model is used for detecting vehicles and pedestrians. The main algorithm deployed for YOLOv3 is CNN, which is developed by Darknet. OpenCV has been employed for image processing, reading, and writing. Object tracking employs both CPU cores and GPU for processing.

The main challenge of the task is to balance the accuracy and speed of the object detection process. Regarding the literature and former experiences of the lab about object detection performance, using an integration of CPU cores and GPU would soar the speed of detecting objects while keeps the accuracy high. The University of Calgary GeoSensorWeb lab research team leveraged several traditional and innovative methods to increase detection performance speed while keeping an eye on accuracy.

- **Vehicle Tracking:**

To monitor the traffic flow, vehicles are tracked between consecutive image frames. The entire procedure is divided into detecting and tracking steps. In the first step, objects of every frame are detected by the ML trained model. The second step is to identify objects and track them through frame series using deterministic mathematical models. As the second part devours most of the processing unit capacity, the developing team broadly focused on increasing the speed of the object tracking using fast numerical function libraries (such as Numba [http://numba.pydata.org/]). Also, the algorithm takes advantage of the highly reliable estimation-prediction process of the Kalman filter algorithm. Similar to object detection, object tracking employs both CPU cores and GPU for processing.

- **Speed Estimation:**

A mathematical model is developed for average object speed estimation. The mean value of vehicle speed is calculated by matching every identified vehicle through a consecutive pair of video captured frames. Then using correlation and distance measurements between the matched objects through the frames, rational speed is estimated.

The lab research team investigated several solutions for object tracking from video frames. The best solution must generate the most precise results while keeping the speed high as well.

Multiple object detection and tracking. This is the most credible candidate regarding project fitness for use. Although the tracking speed is not as fast as the other two methods, it generates the most accurate results. In the first step, all objects are identified in every video frame. Then every couple of consecutive video frames are selected. Then bounding boxes of every object is compared with the previous frame boxes. The most correlated box is reserved for the object and is stored for the next frame in a similar process. Eventually, all objects are identified and tracked in all frames. Some mathematical algorithms, such as Kalman Filter and programming libraries such as Numba, are used for increasing the speed of more than one frame per second for low processing unit devices. Figure 91 shows a screenshot of a demo application for a loop detector. In the demo, all vehicles are tracked and counted as they cross a virtual line.

In order to detect the speed of vehicles, a loop detector approach has been applied to find the period of vehicles crossing the virtual line. An example of a speed detection process is illustrated in Figure 91.

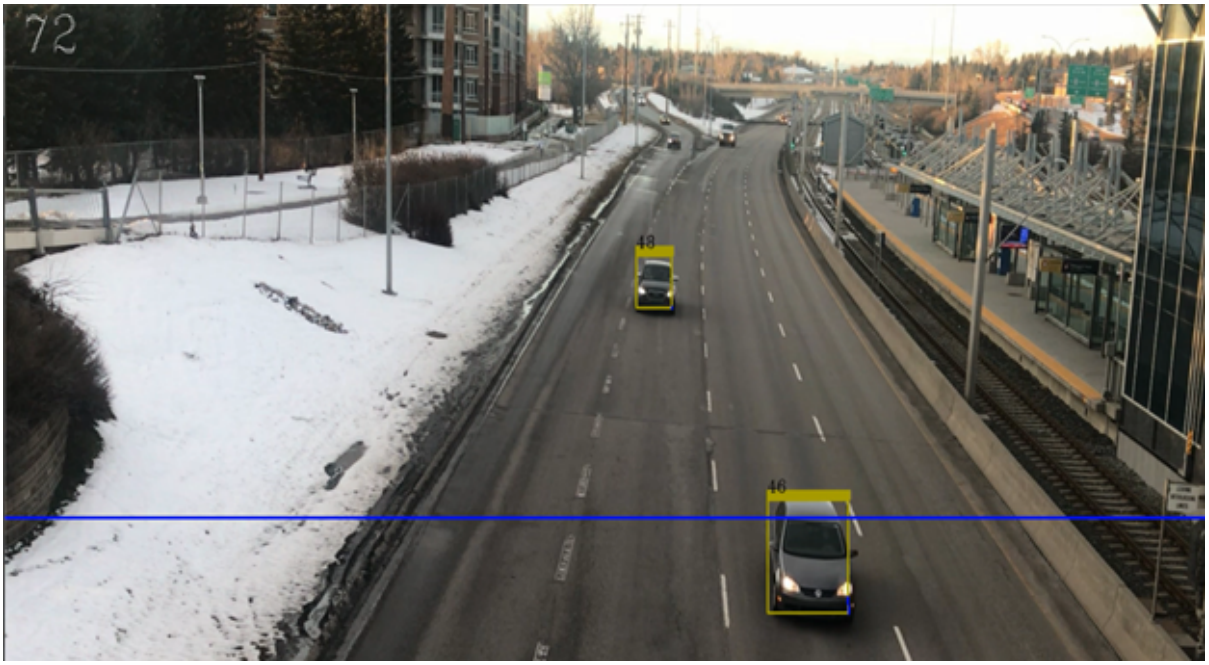

Figure 91. The Sample Speed Detection Using Loop Detector Approach (The average speed is on the top left corner of this photo)

Flood Monitoring Methodology

This section explains the flood detection process in this pilot. Figure 92 shows a general scheme of estimating the severity of a flash flood. The input of this process is a frame captured from the camera. The frame is resized later to be prepared for the ML module, where a Convolutional Neural Network (CNN) is specially trained for four different classes of objects: water splash, low-level of water, high-level of water, and no flash flood. The change in the level of water is detected based on the level of tire sinking in the water. Then, a scheduler is used to call a HyperText Transfer Protocol (HTTP) request builder. This module is responsible for transforming the result into JSON format and push it to an OGC® SensorThings API [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html] endpoint.

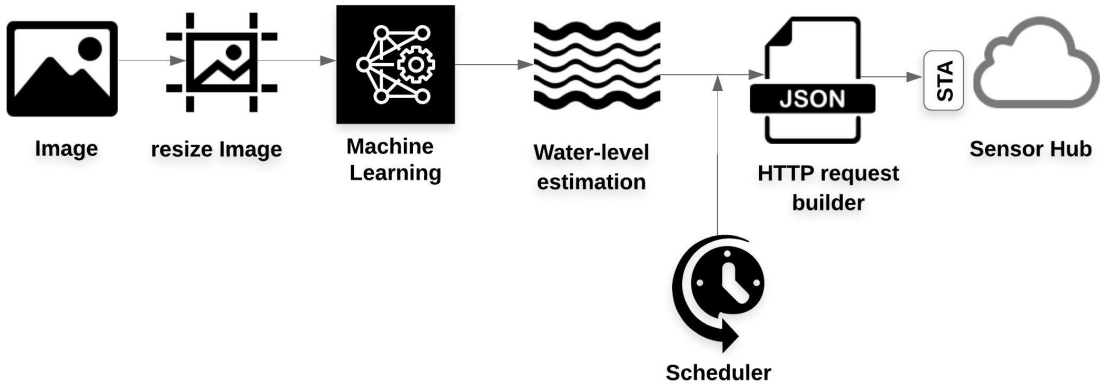

Figure 92. The General Scheme of the Flood Detection Process

For detecting standing water in the street and estimating the severity of the flash flood, a model is trained using flood datasets from the GIS Research Center at Feng Chia University, Taiwan.

Regarding flash flood detection, since the severity of the flash flood is estimated using the portion of tire sinking in water, the camera should be installed at a height that it can capture the side view

of the vehicle (Figure 93 shows two examples of such a viewpoint).

Figure 93. Some Samples Camera Viewpoint From Jetson #1 for Flood Severity Detection (The tires of vehicles should be visible in the frames to be detected)

For this pilot, a fork of implemented YOLO (AlexeyAB darknet repository [https://github.com/AlexeyAB/darknet.git]) is used. The trained YOLO model is tested further, using video files from YouTube that are new to the model. The output of the trained ML model for detecting water splashes as well as estimating the severity of the flood is presented at this link [https://drive.google.com/file/d/1SSLqH_6td7ZaHh24ZJnKRcrQrOI1Aq6C/view?usp=sharing]. Figure 94 is a screenshot of the output of the model. The following figure shows that the model successfully detected the splashes and the level of water.

Figure 94. An Example of the DL Algorithm in Detecting Water Splashes and Sinking Tires (The confidence values of each detection are shown in parenthesis)

Data Publisher

The observations of the smart traffic sensor are pushed to the SensorHub endpoint.

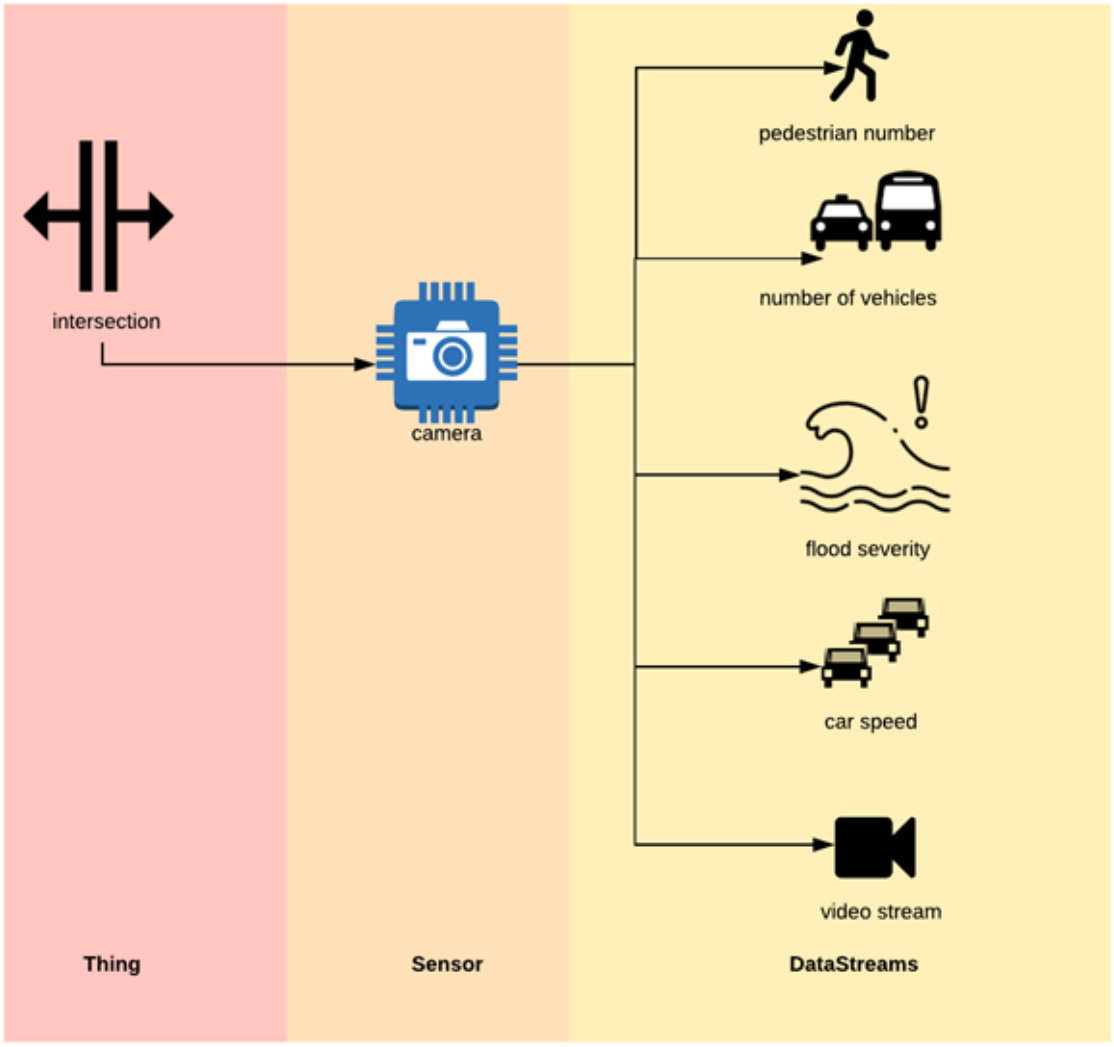

Figure 95. University of Calgary Endpoint Data Model

Figure 95 illustrates the primary data model for the University of Calgary data results. For the model, Things are street intersections, a set of cameras is considered as a sensor, datastreams include numbers of pedestrians, number of vehicles, flood occurrence and average speed. The presented data model completely follows the OGC STA. In this model, the Thing is the intersection that the camera is pointing at, the Sensor is the smart sensor or the location of the camera, and the Datastreams present the number of pedestrians, number of vehicles, water existence, and vehicle average speed. Based on the SensorHub endpoint, our model details are articulated in Table 5.

Table 6. University of Calgary Endpoint Details

Item	ID	Name	Link
Thing	23	The intersection of 32 Ave Nw and Collegiate Blvd NW	Things(23) [https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Things(23)]
Thing	24	The intersection of 32 Ave Nw and 33 ST NW (for traffic monitoring)	Things(24) [https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Things(24)]

Item	ID	Name	Link
Feature of Interest	21	The intersection of 32 Ave Nw and Collegiate Blvd NW	FeaturesOfInterest(21) [https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/FeaturesOfInterest(21)]
Feature of Interest	28	The intersection of 32 Ave Nw and 33 ST NW (for traffic monitoring)	FeaturesOfInterest(28) [https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/FeaturesOfInterest(28)]
Data Stream	196	Pedestrian traffic count	Datastreams(196) [https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Datastreams(196)]
Data Stream	197	Vehicle traffic count	Datastreams(197) [https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Datastreams(197)]
Data Stream	192	The severity of the flash flood	Datastreams(192) [https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Datastreams(192)]
Data Stream	199	Average Speed	Datastreams(199) [https://ogc-hub.compusult.com/SensorHub/SensorThings/v1.0/Datastreams(199)]

All results are converted to JavaScript Object Notation (JSON) format based on the OGC® SensorThings API [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html] standard. As the SensorHub cloud accepts MQTT and HTTP requests and messages, two functions are coded for transferring results to the Occupational Safety and Health (OSH) endpoint datastreams. The edge processing results are considered as observations and sent to the OSH cloud service. Every smart device involves several asynchronous and synchronous threads. To avoid crossing tasks, a scheduling function is defined to manage tasks based on their priorities and also fitness for use. As there would be two Jetson sensors, one for detecting flood and one for processing traffic results, there is no need to schedule tasks between flood and traffic tasks. Therefore, the scheduling task involves sending different data streams. For instance, vehicle and person numbers are sent every 5 seconds and in the separate parallel processes while the speeds are processed and posted to the server every 10 seconds.

12.1.3. Demos and Results

Two online demos for each of the Jetsons to demonstrate each model's capabilities is prepared for this pilot. In the online demos, we tried to simulate the real scenario in the GeoSensorWeb lab at the University of Calgary (Figure 96). In order to illustrate the real-time performance of the sensor, Jetson's camera is pointed to a pre-recorded video file. Two video files are prepared for this pilot: 1) a highway with vehicles and pedestrians, 2) a flash flooding in the streets ([5], [6]). Jetson captures real-time images from monitor I, where the prepared video is displayed, and passes it to the ML

model. The results of the ML model are visualized in a screen connected to the Jetson (monitor II), and at the same time, they are pushed to SensorHub.

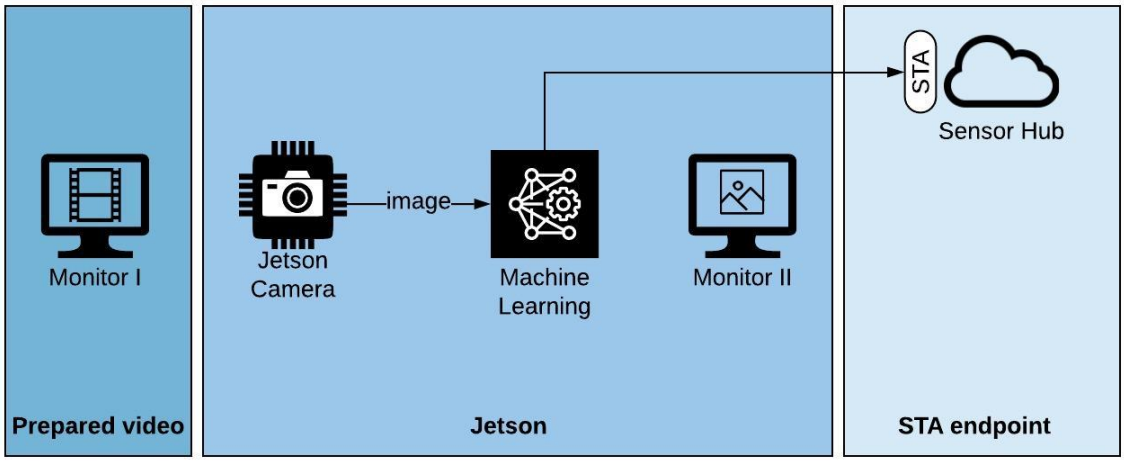

Figure 96. Simulating the Real-Time Scenario for the Online Demo

The online demo for Jetson #2, for traffic monitoring, is shown in Figure 97. The screen of monitor II and the corresponding Datastreams on the SensorHub are illustrated in this figure. The observations are pushed with an interval of less than 1 second. The related video to Figure 97 is available at this link [https://drive.google.com/file/d/1haNEFHnoIM3fqHyzf2iB76DjEzwfGP3c/view].

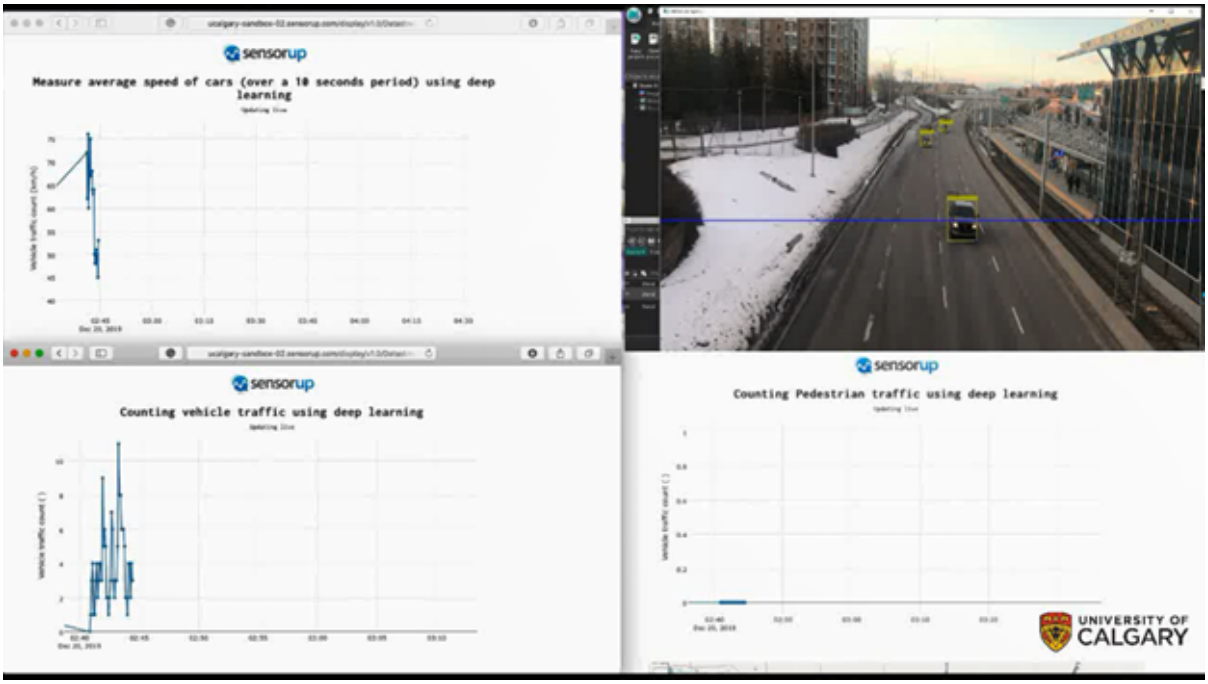

Figure 97. The Traffic Demo for the Smart Traffic Sensor. (Top-left: Observations of the average speed of vehicles (SensorHub endpoint), Top-right: result of the ML model (monitor II), Bottom-left: Observations of vehicle count (SensorHub endpoint), Bottom-right: Observations of the pedestrian count (SensorHub endpoint))

The video of second online demo, for flash flood detection is also available at here [https://drive.google.com/file/d/1DwpGUSojCpQ-uE9FI0vMe18w13ZhR2Au/view?usp=sharing]. Figure 98 shows a screenshot of this demo. The small window in this image shows the live updating observations on the SensorHub.

Figure 98. A Screenshot of the Online Demo for Flash Flooding Scenario

NOTE

Architecture Design Performance

In the SCIRA pilot exercise, cloud service and edge-computing architectures have been developed for monitoring traffic:

The cloud traffic monitoring service, in comparison with the other architecture, is developed based on the CPU processing algorithms and is independent of the Operation System. In a stand-alone architecture implemented on a laptop (with NVIDIA GPU model of GTX2070), the CPU processing algorithm demonstrated five frames per second (fps) while leveraging GPU soared it into 50 fps. The performance involves tracking, speed estimating, and object detection workflows. Therefore, the cloud service will be an adequate option in case of a fast analysis. The service is fitted for central computing uses when the high rate of data transmission is not a problem.

Regarding the latest tests, the performance of the Jetson services (using GPU processing) is ten times as fast as the cloud service performance. Jetson, as an edge computing approach, will be used when high rate IoT monitoring results are required. Moreover, distributing computation loads to the network nodes provides high overall performance for the IoT network. However, reconfiguring and any further developments on the sensors are hard and expensive.

NOTE

Image Processing Library Performance

The object detection algorithms of vehicles and pedestrians' detection quality include detection accuracy and speed. These two criteria grow in reverse (i.e. improvement in execution time of the algorithm would lead to a decrease in the accuracy of the detection and vice versa). YOLOv3 can provide two fps speed and 15-pixel accuracy (the smallest object size which is recognizable by the ML component) in Jetson. Although the TinyYOLOv3 speed is 15 fps, the accuracy is 30% less than YOLOv3. Regarding the priority of accuracy to the execution time, for this project, YOLOv3 is used for object detection at a 5-sec rate.

NOTE

Image Processing Lesson Learned

The speed estimation algorithm accuracy vitally depends on the virtual line (loop detector) position in the frame. The location of the line should be configured based on the camera point of view. Based on experiments, the best line position for Closed Circuit Television (CCTV) Traffic cameras is a horizontal line that crosses the half-frame height.

Relative speed outcomes are reliable in most cases. Therefore, it is highly recommended to use the algorithm for finding street congestion or monitoring traffic flows. At the same time, the absolute results demonstrate an estimation of real speed values, which is not suggested for precise speed detection use cases.

Regarding the flash flood detection, since the dataset used for training YOLO was the side view of the vehicle, it is crystal clear that the performance of the algorithm would increase should the input of the model include a side view of the vehicle.

Moreover, using splashes as an indicator for detecting water in the street would make the model dependent on vehicles passing through water.

To improve flash flood detection, a bigger dataset with a variety in the view angle, light and different levels of water is needed.

Chapter 13. Findings

Typically, smart city projects involve complex distributed systems with multiple stakeholders and diverse applications. To make well-informed decisions, it is essential to have a proper data integration strategy, which must allow working with heterogeneous data sources and platforms in interoperable ways.

SCIRA Pilot ER describes how interoperability can be established in the common operational SCIRA Pilot over the heterogeneous data sources (*e.g.* sensors, IoT platforms, simulators and other data models and encodings), platforms, applications and stakeholders. On the other hand, SCIRA architecture allows querying and visualizing observations from data sources using widely adopted international standards such as the OGC SensorThings API.

Multiple data sources can be attached to these infrastructures and their interfaces will always be common for different applications. The SCIRA Pilot provides several deployment possibilities to meet the interests of different types of stakeholders. A medium skilled Java programmer is capable of implementing a new adapter within a day based on the already provided examples. This would allow the user to retrieve sensor observations from all the connected data sources.

NOTE

SCIRA Operational Pilot Findings

- Interoperability based on standards and on a broad paradigm of observations & measurements is a valuable “awareness multiplier” for public safety operations.
- Preparation and execution of an operational demonstration show the importance and difficulty of interoperating between operational awareness (observations, predictions) and operational management (incidents, tasks).
- Public safety operations rely on well-rehearsed responsive protocols and procedures driven by events. They are best served by event-driven, subscription-based IT architectures.
- Operational exercises require concentration on the way that operational participants do their jobs, so technical capabilities and interoperability arrangements (e.g., observation types) need to be established earlier and more stably than in initiatives targeting lower technical maturity.
- Public safety is an on-going activity in every city, so smart city architectures need not only “As-is” and “To-be” components, but also “Becoming” guidance.

13.1. Application Development Findings

The following findings were derived from the developed applications:

- **3D Dashboard:** The 3D Dashboard achieved a level of interoperability through implementation of different interfaces to communicate, fetch, infuse and visualize data through REST request/response and MQTT publication-subscription services:
  - Fetching data from the data sources is the first step, building a generic interface which acts as a common platform to fetch data from different data sources providing a common visualizing platform.

- Automating the process of fetching the live data at multiple intervals.
- Automating the process of displaying the API response for visualization of data.
- Injecting the fetched data into different scenario by applying conditions/schemas.

- **Traffic management and Routing:**
  - The *Open Routing API* developed in the *Open Routing Pilot* has proven to be efficient, accessible and flexible, being able to adapt to specialized use cases. The first recommendation would be to continue the development and testing of this API and to find new use cases where it can be put in practice.
  - One possible extension to the *Open Routing API* is related to routes with multiple transport means (in the SCIRA Pilot this included vehicles and by foot). This could be a research area with great potential for the API, not only to be able to specify the transport means of a route, but to apply the API for multi-modal passenger transportation (finding the most efficient transportation from point A to B, combining public and private options such as taxi, bus, tram, metro, ferry, train or plain) or even for logistics and freight transportation. The API and Route Exchange Model probably support already most of the basic elements required to do so and it would vastly expand its applicability.
  - A second extension to evaluate for the *Open Routing API* (coded in the SCIRA Pilot with the parameter "context") allows the routing engine to use different behaviors, other than typical ones. In the SCIRA Pilot this was applied in two use cases: 1) for emergency vehicles that, in addition to eventual traffic rules for an emergency encoded in the map dataset, emergency vehicles were permitted to go through roads closed only for public vehicles and 2) for public vehicles requesting an evacuation route selected randomly from a pool of possible options. A deeper evaluation of this extension would allow the *Open Routing API* to better adapt to vertical routing applications, in combination with the myriad of sensors and web applications offering information that could affect a route calculation.
  - The option to review historical routes, although useful in pilots, might cause security and privacy issues if not properly implemented. Further revisions of the API should take this into consideration.
- **Flood and Inundation Models:** The Flood and Inundation model subscribes for alerts to be published and runs on different sensors observations as the input and performs predictive analysis, flash flooding and river flooding simulations. The model also incorporated a few other datastream observations (*e.g.* camera feeds, weather observations, stream gauge readings *etc.*). The model sends its output to the *Compusult SensorHub* that can be further consumed in decision making and preparatory actions.
  - Transforming OGC® CityGML [https://portal.opengeospatial.org/files/?artifact_id=47842] into OGC® 3D Tiles [http://docs.opengeospatial.org/cs/18-053r2/18-053r2.html] using Cesium ion which is the supported format to display 3D City Model on the map.
  - Successful in retrieving the stream gauge, weather sensor data from the OGC-supported SensorHub (*Compusult SensorHub*) and using it as an input to the flood model.
  - Incorporating a boundary (shape file) of the City of St.Louis to limit the area of sight and provide a better view for the end-user.
  - Automated flood modeling based on alerts published by the *Compusult SensorHub*.
- **Smart Traffic/Road Sensor:** There are a cloud service and edge-computing architectures for

Smart Traffic/Road Sensor.

- The cloud service has better performance, easier software configuration and maintenance; but, it needs a high rate of data transmission.
- Edge processing provides high overall performance for the IoT network, less bandwidth, similar performance (using GPU) as cloud; However, the maintenance and update of the software is hard and expensive.
- Flash flood detection accuracy depends on the training step.
- To improve flash flood detection, a bigger dataset with a variety in the view angle, light and different levels of water is needed.

Appendix A: Master Scenario Events List

This appendix is the latest version of the MSEL document after the finalized corrections during the pilot preparation sessions on January 2020.

Scenario 1: River Monitoring

The City needs to monitor the river and anticipate the additional threat of flooding and needs to continue to take preparatory action while also dealing with potential responses related to the pending storm. Focus on River Des Pers and Riverview Dr. & Spring Gardens Area

Table 7. MSEL for the river monitoring scenario

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
1-001		START EX	N/A	Ops Center	Scenario start									N/A	
1-002		EA	CONTROL	Ops Center	Controller confirms that sensor observation services are registered and accessible	Real	Pull	• Cyient 3D Dashboard	Compusult Cloud Sensor Hub	Laptop		SA/COP	No		May just need to show this in Compusult dashboard

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
1-003		EA	CONTROL	Ops Center	Controller subscribes to flood-related alerts	Real	Pull	• Cyient 3D Dashboard	Compusult Cloud Sensor Hub	Laptop		SA/COP	No		MQTT subscription capability
1-004	1.1	EA	CEMA	Ops Center	Operator monitors dashboards during flood event	Real		• Cyient 3D Dashboard	N/A	Laptop		SA/COP	Yes		Cyient Dashboard fetches observations, Coolfire fetches from Ronin

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
1-005	1.2	INJ	TECH	TECH	River gauge sensor publishes current measurements	Real	Pull	USGS online data	Botts SensorHub	Cloud		SA/COP	No		Botts SensorHub pulls data from USGS
1-006		EA	TECH	TECH	SensorHub publishes data	Real	Push	Botts SensorHub	Compusult Cloud Sensor Hub	Cloud		SA/COP	No		Capability is in place, not yet activated
1-007	1.2	INJ	TECH	TECH	Intellisense flood sensor transmits data	Sim	Push	Intellisense flood sensor	• Botts SensorHub	Cloud		SA/COP	No		Botts working with Intellisense on driver

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
1-008		EA	TECH	TECH	SensorHub publishes data	Real	Push	• Botts SensorHub	Compusult Cloud Sensor Hub	Cloud		SA/COP	No		Waiting on data
1-009	1.2	INJ	TECH	TECH	Weather station transmits data	Real	Pull	Weather station	Botts SensorHub	Cloud		SA/COP	No		
1-010		INJ	TECH	TECH	National Weather Service (NWS) publishes weather data	Sim	Pull	NWS online weather feeds	Botts SensorHub	Cloud		SA/COP	No		Possible to show real with Weatherbit or other API.

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
1-011		EA	TECH	TECH	SensorHub publishes data	Real	Push	Botts SensorHub	Compusult Cloud Sensor Hub	Cloud		SA/COP	No		
1-012		INJ	TECH	TECH	Creation of potential flooding alert	Real			Compusult Cloud Sensor Hub	Cloud	Topic: "FloodTrend"	SA/COP	No		Awaiting flood observation
1-013		EA	TECH	TECH	Hub publishes alert	Real	Push	Compusult Cloud Sensor Hub	• Cyient 3D Dashboard	Laptop			No		Available through MQTT but not Ronin API
1-014		EA	CEMA	Ops Center	Operator views alert on dashboard	Real			• Cyient 3D Dashboard	Laptop		SA/COP	Yes		Automated due to subscription to alert

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
1-015		EA	TECH	TECH	Cyient model receives alert	Real	Push	Compusult Cloud Sensor Hub	Cyient Flood and Inundation Model	Cloud		SA/COP	No		Automated due to subscription to alert - 3 depths categorized: 0, 6", 12"
1-016	1.2	INJ	TECH	Ops Center	Cyient runs flood prediction model and publishes prediction observations	Real	Push	Cyient Flood and Inundation Model	Compusult Cloud Sensor Hub	Cloud		SA/COP	No		Automated after receipt of alert or continuous / periodic

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
1-017		EA	TECH	TECH	Hub publishes model outputs		Push	Compusult Cloud Sensor Hub	• Cyient 3D Dashboard	Laptop					Automated publishing to Cyient Dashboard (see above); Coolfire Dashboard <→ Compusult in progress

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
1-018	1.1	EA	CEMA	Ops Center	Operator visualizes flood model outputs	Real		• Cyient 3D Dashboard	N/A	Laptop		SA/COP	Yes		Cyient is able to fetch an visualize flood observations, also is sending geofences to Ronin via API at present
1-019		END EX	N/A	Ops Center	End of scenario										

Scenario 2: Flash Flooding

Flash flooding has led to a high amount of standing water on streets in and around downtown. These streets are no longer passable and need to be blocked off. The City needs to detect where flash flooding is occurring, severity, and respond accordingly (i.e. block off flooded streets).

Table 8. MSEL for the Flash Flooding scenario

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/SIM for Demo	PUSH/PULL	ORIGINATING DEVICE/APPLICATION	DESTINATION DEVICE/APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
2-001		STARTEX	N/A	Ops Center	Scenario start									N/A	
2-002		EA	CONTROL	Ops Center	Controller confirms that sensor observation services are registered and accessible	Real	Pull	• Cyient 3D Dashboard	Compusult Cloud Sensor Hub	Laptop		SA/COP	No		May just need to show this in Compusult dashboard

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
2-003		EA	CONTROL	Ops Center	Controller subscribes to flood-related alerts	Real	Pull	• Cyient 3D Dashboard	Compusult Cloud Sensor Hub	Laptop		SA/COP	No		MQTT subscription capability
2-004		EA	STREETS	Ops Center	Operator monitors dashboards during flood event	Real	Pull	• Cyient 3D Dashboard	N/A			SA/COP	Yes		Cyient Dashboard fetches observations, Coolfire fetches from Ronin
2-005	2.1	INJ	TECH	Field	U of Calgary sensors transmit inundation data	Sim	Push	UCalgary SensorHub	Compusult Cloud Sensor Hub	Cloud		SA/COP	No		"Text" Observation Type

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
2-006	2.1	INJ	CITIZENS	Field	Community members report flooding observations	Real	Push	• Coolfire Community App	Compusult Cloud Sensor Hub	Andriod Phone / Tablet		SA/COP	Yes		
2-007	1.1	EA	STREETS	Ops Center	Operator visualizes observations on dashboard	Real	Pull	• Cyient 3D Dashboard	Compusult Cloud Sensor Hub	Laptop		SA/COP	Yes		Both sensor and community observations

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
2-008	2.2	INJ	TECH	TECH	Creation of flash flooding alert	Real		Compusult Cloud Sensor Hub	N/A	Cloud	Topic: "FlashFlood"	SA/COP	Yes		Alert is created by the Sensor Hub based on received observations (Water depth 6"+)
2-009		EA	TECH	TECH	Hub publishes alert	Real	Push	Compusult Cloud Sensor Hub	• Cyient 3D Dashboard	Laptop		SA/COP	No		
2-010		EA	STREETS	Ops Center	Operator views alert on dashboard	Real		N/A	• Cyient 3D Dashboard	Laptop		SA/COP	Yes		Automated due to subscription to alert

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
2-011		EA	TECH	TECH	Cyient receives alert and/ or observations	Real	Push + Pull	Compusult Cloud Sensor Hub	Cyient Flood and Inundation Model	Cloud		SA/COP	No		model receives observations,
2-012		INJ	TECH	TECH	Cyient model sends flash flood prediction observations	Real	Push	Cyient Flood and Inundation Model		Cloud		SA/COP	No		Triggered by receipt of alert or continuous

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
2-013		EA	TECH	TECH	Hub publishes flash flood prediction observations		Push	Compusult Cloud Sensor Hub	• Cyient 3D Dashboard	Laptop	Topic: "FlashFlood"	SA/COP	No		Automated publishing to Cyient Dashboard (see above); Coolfire Dashboard receives from Compusult
2-014	1.1	EA	STREETS	Ops Center	Operator visualizes flood model outputs	Real		• Cyient 3D Dashboard	N/A	Laptop		SA/COP	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
2-015	2.4	EA	STREETS	Ops Center	Operator creates street closure task order	Real	Push	• Cyient 3D Dashboard	Coolfire Services Platform	Laptop		CAD	Yes		Closure observation
2-016	2.4	EA	STREETS	Station	Street crew receives task order	Real	Push	Coolfire Services Platform	• Coolfire Responder App	Android Phone / Tablet		CAD	Yes		
2-017	2.4	EA	STREETS	Station	Street crew acknowledges and accepts task order	Real	Push	• Coolfire Responder App	Coolfire Services Platform	Android Phone / Tablet		CAD	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
2-018.0		EA	STREETS	Station	Street crew requests route	Real	Pull	• Coolfire Responder App	Skymantics routing service	Andriod Phone / Tablet		ROUTING	Yes		***Coolfire app does not connect to Skymantics: either use Coolfire HERE subscription or use Botts for routing
2-018	2.4	EA	TECH	Station	Skymantics provides optimal route	Real	Pull	Skymantics Routing Service	• Coolfire Responder App	Andriod Phone / Tablet		ROUTING	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
2-019	2.4	EA	STREETS	Field	Streets crew departs for tasked location	Real		N/A	N/A			ROUTING	No		Route visualized in app with or without turn-by-turn
2-020	2.4	EA	TECH	Field	Device publishes GPS data	Real	Push	Device	Botts SensorHub	Cloud		ROUTING	No		Tracking observation
2-021	2.4	EA	TECH	Field	SensorHub publishes data	Real	Push	Botts SensorHub	Compusult Cloud Sensor Hub	Cloud		ROUTING	No		**Tracking directly through Coolfire Services Platform to Dashboard***

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
2-022	2.4	EA	STREETS	Ops Center	Operator requests and visualizes crew location	Real	Pull	Compusult Cloud Sensor Hub	• Cyient 3D Dashboard	Laptop		ROUTING	Yes		Could also be a subscription
2-023	2.4	EA	STREETS	Field	Streets crew arrives at flood location	Real		N/A	N/A			WORKFORCE MOB	No		
2-024	2.4	EA	STREETS	Field	Streets crew closes flooded street	Sim		N/A	N/A			WORKFORCE MOB	No		
2-025	2.4	EA	STREETS	Field	Streets crew reports road closure	Real	Push	• Coolfire Responder App	Compusult Cloud Sensor Hub	Andriod Phone / Tablet	Topic: "RoadClosure"	WORKFORCE MOB	Yes		Closure observation (Botts)

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
2-026	2.4	EA	STREETS	Field	Streets crew adds update to task order	Real	Push	• Coolfire Responder App	Coolfire Services Platform	Android Phone / Tablet		WORKFORCE MOB	Yes		Task update (Coolfire)
2-027	2.4	EA	STREETS	Ops Center	Operator requests route computation to confirm street closure	Real	Pull	• Cyient 3D Dashboard	Skymantics Routing Service	Laptop		ROUTING	Yes		Selects arbitrary waypoints to illustrate routing around closure
2-028	2.4	EA	TECH	Ops Center	Skymantics provides route computation	Real	Push	Skymantics Routing Service	• Cyient 3D Dashboard	Laptop		ROUTING	No		*

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING DEVICE/ APPLICATION	DESTINATION DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
2-029	2.4	EA	STREETS	Ops Center	Operator visualizes provided route	Real		• Cyient 3D Dashboard	N/A	Laptop		ROUTING	Yes		
2-030		END EX			End scenario										

Scenario 3: Assistance to Vulnerable Populations

The flash flooding has created a life and safety risk for vulnerable populations in the areas where the flash flooding is occurring. The health department needs to detect and verify the existence of vulnerable individuals and conduct outreach to get them to safety.

Table 9. MSEL for the assistance to vulnerable populations scenario

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
3-001		STARTEX	N/A	Ops Center	Scenario start										
3-002		EA	CONTROL	Ops Center	Controller confirms that sensor observation services are registered and accessible	Real	Pull	• Cyient 3D Dashboard	Compusult Cloud Sensor Hub	Laptop		SA/COP	No		May just need to show this in Compusult dashboard

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/SIM for Demo	PUSH/PULL	ORIGINATING/RESPONDING DEVICE/APPLICATION	DESTINATION/REQUESTING DEVICE/APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
3-003		EA	CONTROL	Ops Center	Controller subscribes to flood-related and health-related alerts	Real	Pull	• Cyient 3D Dashboard	Compusult Cloud Sensor Hub	Laptop		SA/COP	No		MQTT subscription capability
3-004		EA	HEALTH	Ops Center	Operator monitors dashboards during flood event	Real	Pull	• Cyient 3D Dashboard	N/A	Laptop		SA/COP	Yes		Cyient Dashboard fetches observations, Coolfire fetches from Ronin

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/SIM for Demo	PUSH/PULL	ORIGINATING/RESPONDING DEVICE/APPLICATION	DESTINATION/REQUESTING DEVICE/APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
3-005		INJ	CITIZEN	Field	Citizen reports individual in need of assistance	Real	Push	• Coolfire Community App	Compusult Cloud Sensor Hub	Andriod Phone / Tablet		SA/COP	Yes		Aid observation — probably 2 or more of them in order to generate 2 tasks for 2 teams
3-006		INJ	TECH	TECH	Creation of citizen in need alert	Real	Push	Compusult Cloud Sensor Hub	N/A	Cloud	Topic: "PersonAssist"	SA/COP	Yes		Alert is created by the Sensor Hub based on received observations

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
3-007		EA	TECH	TECH	Hub publishes citizen in need alert	Real	Push	Compusult Cloud Sensor Hub	• Cyient 3D Dashboard	Laptop		SA/COP	No		
3-008		EA	HEALTH	Ops Center	Operator reviews citizen reports	Real	Pull	Compusult Cloud Sensor Hub	• Cyient 3D Dashboard	Laptop		SA/COP	Yes		
3-009		EA	HEALTH	Ops Center	Operator decides that flooded areas will impact vulnerable individuals	Sim		N/A	N/A			SA/COP	No		Uses inundation observations from S2

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
3-010		EA	HEALTH	Ops Center	Operator creates incident and task order for outreach teams to assist vulnerable individuals	Real	Push	• Cyient 3D Dashboard	• Ronin	Laptop		CAD	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
3-011		INJ	HEALTH	Health Dept	Outreach teams receive task order	Real	Push	• Ronin	• Coolfire Responder App	Android Phone / Tablet		CAD	Yes		2 teams can be staffed per Robert: 111319. Figure out the task multiplicity
3-012		EA	HEALTH	Health Dept	Outreach team acknowledges and accepts task order	Real	Pull	• Coolfire Responder App	• Ronin	Android Phone / Tablet		CAD	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
3-013		EA	HEALTH	Health Dept	Outreach team requests route	Real	Pull	• Coolfire Responder App	Skymantics routing service	Android Phone / Tablet		ROUTING	Yes		
3-014		EA	TECH	Health Dept	Skymantics provides optimal route	Real	Push	Skymantics routing service	• Coolfire Responder App	Android Phone / Tablet		ROUTING	No		
3-015		EA	HEALTH	Field	Outreach team departs for tasked location	Real		N/A	N/A			ROUTING	No		Visualizing route +/- turn-by-turn in mobile app
3-016	2.4	EA	TECH	Field	Device publishes GPS data	Real	Push	Device	BottsSensorHub	Cloud		ROUTING	No		Smarthub +/- Cradlepoint

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
3-017	2.4	EA	TECH	TECH	SensorHub publishes data	Real	Push	Botts SensorHub	Compusult Cloud Sensor Hub			ROUTING	No		
3-018	2.4	EA	HEALTH	Health Dept	Operator visualizes crew location	Real	Push or Pull	Compusult Cloud Sensor Hub	• Cyient 3D Dashboard	Laptop		ROUTING	Yes		
3-019		EA	TECH	TECH	Creation of new street closure	Real	Push	• Coolfire Responder App	Compusult Cloud Sensor Hub	Andriod Phone / Tablet + Laptop		SA/COP	Yes		Closure observation.

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
3-020	2.2	INJ	TECH	TECH	Creation of closure alert	Real		Compusult Cloud Sensor Hub	N/A	Cloud	Topic: "RoadClosure"	SA/COP	No		Alert is created by the Sensor Hub based on received observations
3-021		EA	TECH	TECH	Hub publishes street closure alert	Real	Push	Compusult Cloud Sensor Hub	N/A	Cloud		SA/COP	No		
3-022		INJ	HEALTH	Field	Outreach team receives alert of street closure	Real	Push	Compusult Cloud Sensor Hub	• Coolfire Responder App	Android Phone / Tablet		SA/COP	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
3-023		EA	HEALTH	Field	Outreach team requests new route	Real	Pull	• Coolfire Responder App	Skymantics routing service	Android Phone / Tablet		ROUTING	Yes		
3-024		EA	TECH	Field	Skymantics provides updated route	Real	Push	Skymantics routing service	• Coolfire Responder App	Android Phone / Tablet		ROUTING	No		
3-025	2.4	EA	HEALTH	Field	Outreach team arrives at first location	Real		N/A	N/A			WORKFORCE MOB	No		
3-026	2.4	EA	HEALTH	Field	Outreach team reports observations	Real	Push	• Coolfire Responder App	Compusult Cloud Sensor Hub	Android Phone / Tablet		WORKFORCE MOB	Yes		med observation to Sensor Hub

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
3-027		EA	HEALTH	Field	Outreach team updates task status	Real	Push	• Coolfire Responder App	Ronin	Android Phone / Tablet		WORKFORCE MOB	Yes		Update to Ronin
3-028		EA	HEALTH	Ops Center	Operator visualizes and tracks team location	Real	Pull or Push	• Cyient 3D Dashboard	N/A	Laptop		WORKFORCE MOB	Yes		
3-029		END EX			End scenario										

Scenario 4: Building Fire

Unknown to anyone, trenching in the area of T-Rex has damaged the basement and created a crack in the basement wall. Flash flooding and the flooding of trenches has led to flooding in the basement at T-Rex. The flooding in the mechanical room causes a short in the electrical system and starts a fire. Fire, Police, and EMS need to combat the fire and ensure safe and total evacuation of the building.

Table 10. MSEL for the building fire scenario

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
4-001		STARTEX	N/A	Ops Center	Scenario start										
4-002	1.1	EA	FIRE	Ops Center	Operator monitors dashboards during flood event	Real		• Cyient 3D Dashboard	N/A	Laptop		SA/COP	Yes		Cyient Dashboard fetches observations, Coolfire fetches from Ronin

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
4-003		INJ	FIRE	Ops Center	Call taker receives 911 call reporting fire at T-REX	Sim		N/A	N/A			SA/COP	No		
4-004		EA	FIRE	Ops Center	Call taker creates fire incident	Real	Push	• Cyient 3D Dashboard	Ronin	Laptop		SA/COP	Yes		
4-005		EA	FIRE	Ops Center	Operator receives alert of fire incident	Real	Push	Ronin	• Cyient 3D Dashboard	Laptop		SA/COP	Yes		Dispatcher does not take calls, call taker does not dispatch units

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/SIM for Demo	PUSH/PULL	ORIGINATING/RESPONDING DEVICE/APPLICATION	DESTINATION/REQUESTING DEVICE/APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
4-006		EA	FIRE	Ops Center	Operator adds dispatch task(s) to the fire record	Real	Push	• Cyient 3D Dashboard	Ronin	Laptop		SA/COP	Yes		
4-007		INJ	FIRE	Ops Center	Call taker receives second 911 call reporting fire	Sim		N/A	N/A			SA/COP	No		
4-008		EA	FIRE	Ops Center	Call taker confirms incident already in system	Real	Pull	• Cyient 3D Dashboard	Ronin	Laptop		SA/COP	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
4-009		EA	FIRE	Station	Responding units acknowledge and accept dispatch task(s)	Real	Pull	• Coolfire Responder App	Ronin	Android Phone / Tablet		SA/COP	Yes		
4-010.0		EA	FIRE	Station	Responding unit requests route	Real	Pull	• Coolfire Responder App	Skymantics routing service	Android Phone / Tablet		ROUTING	Yes		
4-010		INJ	FIRE	Station	Skymantics provides optimal route	Real	Push	Skymantics routing service	• Coolfire Responder App	Android Phone / Tablet		ROUTING	Yes	Responder app request from present location	4-011

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
	EA	FIRE	En route	Responding units depart for incident scene	Real		N/A	N/A			ROUTING	No			4-012

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
	EA	FIRE	En route	Responding units request building interior view	Real	Pull	Compusult Cloud Sensor Hub (Catalog) + Feature Store	• Ecodomus Dashboard	Laptop		IN BUILDING NAV			Building model is catalogued by Compusult, but may be stored separately in a WFS (Web Feature Service) or locally due to size	4-013

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
	EA	FIRE	En route	Responding units visualize building interior	Real		Ecodomus Dashboard	N/A			IN BUILDING NAV	Yes			4-014
2.4	EA	TECH	TECH	Responding unit device publishes GPS data	Real	Push	Device	BottsSensorHub	Andriod Phone / Tablet or CP		ROUTING	No		Either Cradle-point or SmartHub device GPS (Android phone)	4-015
2.4	EA	TECH	TECH	SensorHub publishes data	Real	Push	BottsSensorHub	Compusult Cloud Sensor Hub	Cloud		ROUTING	No			4-016

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
2.4	EA	FIRE	Ops Center	Operator visualizes team location	Real		• Cyient 3D Dashboard	N/A	Laptop		ROUTING	Yes			4-017
	EA	FIRE	Scene	Responding units arrive at incident scene	Real		N/A	N/A			WORKFORCE MOB	No			4-018
	EA	FIRE	Scene	Responding units report arrival (task update) on scene	Real	Push	• Coolfire Responder App	Ronin	Andriod Phone / Tablet		WORKFORCE MOB	Yes		Update of dispatch task	4-019

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
	INJ	TECH	TECH	Bottsmarthubs request building beacon configuration data	Real	Pull	BottsSmartHubs	Compusult Cloud Sensor Hub	Andrroid Phone / Tablet		IN BUILDING NAV	No		Identities and corresponding location of beacons requested by firefighters prior to building entrance	4-020

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
	EA	FIRE	Scene	Responding units (A) enter building and begin fire response operations	Sim		N/A	N/A			WORKFORCE MOB	No			4-021
	EA	FIRE	Scene	Responding units (B) begin to clear building	Sim		N/A	N/A			WORKFORCE MOB	No			4-022

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
	INJ	TECH	TECH	Bottsmarthubs transmit responder locations in building	Real	Push	Bottsmarthubs	Compusult Cloud Sensor Hub	Android Phone / Tablet		IN BUILDING NAV	No		SmartHubs detect and identify beacons, compute location within building	4-023

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
	INJ	TECH	TECH	Firefighter smart watches transmit physiological data	Real	Push	Keys Smart Watch	Compusult Cloud Sensor Hub	Apple Watch		IN BUILDING NAV	No		may be pushed to Botts SmartHub and then to Compusult Cloud Hub depending on connectivity.	4-024

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
	EA	FIRE	Scene	Command staff monitor firefighter locations in building	Real	Pull	Compusult Cloud Sensor Hub	Ecodomus Dashboard	Laptop		WORKFORCE MOB	Yes		Need to confirm what location information is needed for tracking in Ecodomus Dashboard	4-025
	EA	FIRE	Scene	Command staff guide units to incident tasks	Real	Push	Audio device	Audio device	LMR		WORKFORCE MOB	Yes		LMR + earpiece	4-026

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
	EA	FIRE	Scene	Responding units (A) report fire under control to incident command	Real	Push	Audio device	Audio device	LMR		WORKFORCE MOB	Yes		Could report status to Siri for those with a lot of patience	4-027
	EA	FIRE	Scene	Incident command updates incident record to fire under control	Real	Push	• Coolfire Responder App	Ronin	Andriod Phone / Tablet		WORKFORCE MOB	Yes		Only Coolfire app presently accesses Ronin API	4-028

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/SIM for Demo	PUSH/PULL	ORIGINATING/RESPONDING DEVICE/APPLICATION	DESTINATION/REQUESTING DEVICE/APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
	EA	FIRE	Scene	Responding units (B) report rooms cleared to incident command	Real	Push	Audio device	Audio device	LMR		WORKFORCEMOB	Yes			4-029
	EA	FIRE	Scene	Incident command updates room status	Real	Push	Ecodomus Dashboard		Laptop		WORKFORCEMOB	Yes		Either building attribute or occupancy observation	4-030

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
	EA	FIRE	Scene	Incident command updates incident record to all cleared	Real	Push	• Coolfire Responder App	Ronin	Andriod Phone / Tablet		WORKFORCE MOB	Yes			4-031
	EA	FIRE	Scene	Command staff continue to monitor firefighter locations	Real	Pull	Compusult Cloud Sensor Hub	Ecodomus Dashboard	Laptop		IN BUILDING NAV	Yes			4-032

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
	INJ	TECH	Scene	Firefighter physiological monitor reports heart rate outside of set parameters	Sim	Push	Keys Smart Watch	Compusult Cloud Sensor Hub	Apple Watch		SA/COP	No		med observation	4-033
	EA	TECH	TECH	Creation of physiological alert	Real		Compusult Cloud Sensor Hub	N/A		Topic: "MedAlert"	SA/COP	No		Alert is created by the Sensor Hub based on received observations	4-033

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ ?	TEST STATUS R/Y/ G	NOTES
	EA	TECH	TECH	Hub publishes alert	Real	Push	Compusult Cloud Sensor Hub	Ecodomus Dashboard	Laptop		SA/COP	No			4-034
	INJ	FIRE	Scene	Command staff review alert of firefighter in distress	Real	Pull	Compusult Cloud Sensor Hub	Ecodomus Dashboard	Laptop		SA/COP	Yes			4-035
	EA	FIRE	Scene	Command staff guide paramedics to firefighter	Real	Push	Audio device	Audio device	LMR		SA/COP	Yes			4-036

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
	EA	FIRE	Scene	Command staff continue to monitor progress of units	Real	Pull	Compusult Cloud Sensor Hub	Ecodomus Dashboard	Laptop		WORKFORCE MOB	Yes			4-037
	EA	FIRE	Scene	Command staff updates incident record	Real	Push	Ecodomus Dashboard Coolfire Responder App	Ronin	Laptop		WORKFORCE MOB	Yes			4-038

Scenario 5: Vehicle Accident

A speeding vehicle hits a patch of standing water, hydroplanes and collides with a fire hydrant and as a result knocks over the hydrant. Police, Water, Streets, and EMS need to respond to the incident, render assistance, and deal with the damaged fire hydrant and clear the accident.

Table 11. MSEL for the vehicle accident scenario

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/SIM for Demo	PUSH/PULL	ORIGINATING/RESPONDING DEVICE/APPLICATION	DESTINATION/REQUESTING DEVICE/APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-001		START EX	N/A	Ops Center	Scenario start										
5-002		INJ	PSAP	Ops Center	Call taker receives 911 call reporting vehicle accident	Sim		N/A	N/A			SA/COP	No		
5-003		EA	PSAP	Ops Center	Call taker creates accident incident	Real	Push	• Cyient 3D Dashboard	Ronin	Laptop		SA/COP	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-004		EA	PSAP	Ops Center	Operator receives accident incident	Real	Push	Ronin	• Cyient 3D Dashboard	Laptop		CAD	Yes		
5-005		EA	PSAP	Ops Center	Operator creates dispatch tasks within incident	Real	Push	• Cyient 3D Dashboard	Ronin	Laptop		CAD	Yes		Create a task for each crew responding
5-006		INJ	PSAP	Ops Center	Call taker receives second 911 call reporting accident	Sim		N/A	N/A			SA/COP	No		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-007		EA	PSAP	Ops Center	Call taker confirms incident in system	Real	Pull	Ronin	• Cyient 3D Dashboard	Laptop		CAD	Yes		
5-008		EA	LE	Station	Law enforcement units acknowledge / accept dispatch orders	Real	Push	• Coolfire Responder App	Ronin	Andriod Phone / Tablet		CAD	Yes		Works for Coolfire app, coming for Botts app

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
5-009		EA	WATER	Station	Water department crew acknowledges / accepts dispatch orders	Real	Push	• Coolfire Responder App	Ronin	Andriod Phone / Tablet		CAD	Yes		Works for Coolfire app, coming for Botts app
5-010		EA	EMS	Station	EMS units acknowledge / accept dispatch orders	Real	Push	• Coolfire Responder App	Ronin	Andriod Phone / Tablet		CAD	Yes		Works for Coolfire app, coming for Botts app

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
5-011		EA	STREETS	Station	Streets department crew acknowledges / accepts dispatch orders	Real	Push	• Coolfire Responder App	Ronin	Andriod Phone / Tablet		CAD	Yes		Works for Coolfire app, coming for Botts app
5-012.0		EA	LE	Station	Lawenforcement unit requests route	Real	Pull	• Coolfire Responder App	Skymantics routing service	Andriod Phone / Tablet		ROUTING	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
5-012		INJ	LE	Station	Skymantics provides optimal routes to law enforcement units	Real	Pull	Skymantics routing service	• Coolfire Responder App	Android Phone / Tablet		ROUTING	No		Manual request
5-013.0		EA	WATER	Station	Water crew requests route	Real	Pull	• Coolfire Responder App	Skymantics routing service	Android Phone / Tablet		ROUTING	Yes		
5-013		INJ	WATER	Station	Skymantics provides optimal routes to water department crew	Real	Push or Pull	Skymantics routing service	• Coolfire Responder App	Android Phone / Tablet		ROUTING	No		Part of task acceptance or manual request

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-014.0		EA	EMS	Station	EMS requests route	Real	Pull	• Coolfire Responder App	Skymantics routing service	Android Phone / Tablet		ROUTING	Yes		
5-014		INJ	EMS	Station	Skymantics provides optimal routes to EMS units	Real	Push or Pull	Skymantics routing service	• Coolfire Responder App	Android Phone / Tablet		ROUTING	No		Part of task acceptance or manual request
5-015.0		EA	STREETS	Station	Streets crew requests route	Real	Pull	• Coolfire Responder App	Skymantics routing service	Android Phone / Tablet		ROUTING	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
5-015		INJ	STREETS	Station	Skymantics provides optimal routes to streets department crew	Real	Push or Pull	Skymantics routing service	• Coolfire Responder App	Andriod Phone / Tablet		ROUTING	No		Part of task acceptance or manual request
5-016		EA	ALL	En route	Responding units depart for incident scene	Real		N/A	N/A			ROUTING	No		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-017		EA	TECH	TECH	Lawenforcement devices transmit GPS data	Real	Push	Device	Botts SensorHub → Compusult Cloud SensorHub	Android Phone / Tablet		ROUTING	No		Botts SmartHub GPS
5-018		EA	TECH	TECH	Water department devices transmit GPS data	Real	Push	Device	Botts SensorHub → Compusult Cloud SensorHub	Android Phone / Tablet		ROUTING	No		Botts SmartHub GPS
5-019		EA	TECH	TECH	EMS devices transmit GPS data	Real	Push	Device	Botts SensorHub → Compusult Cloud SensorHub	Android Phone / Tablet		ROUTING	No		Botts SmartHub GPS

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-020		EA	TECH	TECH	Streets department devices transmit GPS data	Real	Push	Device	Botts SensorHub → Compusult Cloud SensorHub	Android Phone / Tablet		ROUTING	No		Botts SmartHub GPS
5-021	2.4	EA	ALL	Ops Center	Operator requests and visualizes crew locations	Real	Push	Compusult Cloud Sensor Hub	• Cyient 3D Dashboard	Laptop		ROUTING	Yes		Push subscription. Could have different topics for different resource types (e.g. firetruck)

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
5-022		EA	LE	Scene	Lawenforcement units arrive on scene	Real		N/A	N/A			WORKFORCE MOB	No		
5-023		EA	LE	Scene	Lawenforcement units close street	Real		N/A	N/A			WORKFORCE MOB	No		
5-024		EA	LE	Scene	Lawenforcement units report public street closure	Real	Push	• Coolfire Responder App	Compusult Cloud Sensor Hub	Android Phone / Tablet		WORKFORCE MOB	Yes		Closure observation

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-025		EA	WATER	Scene	Water dept crew arrives on scene	Real		N/A	N/A			WORKFORCE MOB	No		
5-026		EA	WATER	Scene	Water dept crew shuts off water to hydrant	Real		N/A	N/A			WORKFORCE MOB	No		
5-027		EA	WATER	Scene	Water dept crew update their task	Real	Push	• Coolfire Responder App	Ronin	Android Phone / Tablet		WORKFORCE MOB	Yes		
5-028		EA	EMS	Scene	EMS units arrive on scene	Real		N/A	N/A			WORKFORCE MOB	No		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-029		EA	EMS	Scene	EMS units treat driver of crashed vehicle	Real		N/A	N/A			WORKFORCE MOB	No		
5-030		EA	EMS	Scene	EMS units update their task	Real	Push	• Coolfire Responder App	Ronin	Andriod Phone / Tablet		WORKFORCE MOB	Yes		
5-031		EA	STREETS	Scene	Streets dept crew arrives on scene	Real		N/A	N/A			WORKFORCE MOB	No		
5-032		EA	STREETS	Scene	Streets dept crew shuts off power to downed light pole	Real		N/A	N/A			WORKFORCE MOB	No		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-033		EA	STREETS	Scene	Streets dept crew update their task	Real	Push	• Coolfire Responder App	Ronin	Android Phone / Tablet		WORKFORCE MOB	Yes		
5-034		EA	CITIZEN	Scene	Citizens request optimal evacuation routes	Real	Pull	• Coolfire Community App	Skymantics routing service	Android Phone / Tablet		ROUTING	Yes		

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAININ G REQ?	TEST STATUS R/Y/ G	NOTES
5-035		INJ	TECH	Scene	Skymantics provides optimal routes to citizens (around accident location, flooding closures from S2, S3, building fire closure from S4)	Real	Push	Skymantics routing service	• Coolfire Community App	Android Phone / Tablet		ROUTING	No		Route based on role respects all public closures. Multiple routes to spread out evacuation

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-036		EA	TOWTRUCK	Scene	Tow truck driver requests optimal route	Real	Pull	• Coolfire Responder App	Skymantics routing service	Andriod Phone / Tablet		ROUTING	Yes		
5-037		INJ	TOWTRUCK	Scene	Skymantics provides optimal routes for tow truck driver to accident location	Real	Push	Skymantics routing service	• Coolfire Responder App	Andriod Phone / Tablet		ROUTING	No		Route based on role ignores accident closure (these should be the responder apps)

STEP	OBJ	TYPE	ACTOR	LOCATION	DESCRIPTION	REAL/ SIM for Demo	PUSH /PULL	ORIGINATING/ RESPONDING DEVICE/ APPLICATION	DESTINATION/ REQUESTING DEVICE/ APPLICATION	Client Hardware	ALERTS	CAPABILITY SETS	TRAINING REQ?	TEST STATUS R/Y/G	NOTES
5-038		EA	PSAP	Ops Center	Operator visualizes inundation areas, overall incident and task status, tracking, possibly route overviews in dashboard (stretch)	Real	Pull or Push	Ronin & Compusult Cloud Sensor Hub	• Cyient 3D Dashboard	Laptop		CAD	Yes		Added overall operating picture at end
5-039		END EX			End scenario										

Computing Environment

SCIRA Pilot has been deployed and used on various computing environments and hardware systems such as Android Phone, Laptop, iPad, Android Tab, and Apple Watch. The following table (Table 11) list the hardware used for different scenarios in this pilot.

Table 12. SCIRA pilot computing hardware

Scenario/Resource	Android Phone	Laptop	iPad	Android Tab	Apple Watch
Scenario 1		1			
Scenario 2	1	1			
Scenario 3	2	1		1	
Scenario 4	More than 2	1	1		More than 2
Scenario 5	5	4		1	
Total	More than 9	8	1	2	More than 2

App Interactions

Interactions between different apps developed and used in this pilot are listed in the following table.

Table 13. SCIRA pilot app interactions

App/ Resource	Navigation Route	Tracking Observation	FloodTrend Observation	FlashFlood Observation	RoadClosure Observation	PersonAssist Observation	MedAlert Observation	Flood Incident	Street Closure Task	Vulnerable Pop Incident	Person Assist Task	Fire Incident	Fire Response Task	Accident Incident	Accident Response Task
Dashboard	Request	Subscribe	Subscribe	Subscribe	Create, Subscribe	Subscribe	Subscribe (Ecodomus)	Create, Update	Create	Create, Update	Create	Create, Update	Create	Create, Update	Create
Responder	Request	Create*			Create, Subscribe				Subscribe, Update		Subscribe, Update		Subscribe, Update		Subscribe, Update
Community	Request			Create		Create									
Flood Model			Create	Subscribe, Create											
Routing Service	Respond				Subscribe										
Watch App							Create								

*automated

RGC: "POLICE, EMS, STREETS, and WATER will always use the Coolfire dashboard. CEMA, HEALTH and HUMAN will always use the Cyeint 3D dashboard. I am backing off on FIRE. FIRE will always use whatever ops center dashboard is best suited for Scenario 4"

Appendix B: How to train YOLOv3

In this appendix, the steps through installing and using YOLO is explained briefly.

Using pre-trained YOLO model

Before the Darknet installation, the 'Makefile' should be modified based on the system configurations. For example, OpenCV should be set to '1' if the OpenCV library is installed on the system, or both GPU and CUDNN should be set to '1' if a CUDA GPU is used in the system.

After installing Darknet, a pre-trained YOLO weight file can be used for detecting objects listed in COCO dataset [https://github.com/AlexeyAB/darknet/blob/master/data/coco.names] (considering the scope of this pilot, this weight file is only used for detecting cars and pedestrians). The weight file can be downloaded from here [https://pjreddie.com/media/files/yolov3.weights], or by running the following command:

```
  wget https://pjreddie.com/media/files/yolov3.weights
```

Finally, for running YOLOv3 on webcam the following command is used:

```
  ./darknet detector demo <path-to-the-.data-file> <path-to-the-.cfg-file> <path-to-
weight-file>
```

For example:

```
  ./darknet detector demo cfg/coco.data cfg/yolov3.cfg yolov3.weights
```

Training YOLO

For detecting the existence of water on the street, YOLO is trained to detect water splashes and tires with different levels of sinking in water as an object. The sinking tires have been categorized into three groups of no-water, water-low, and water-high. For this purpose, the flood dataset from FCU <?>, Taiwan, is used. Figure 99 shows some samples of this dataset.

Figure 99. Some sample of flood dataset used for this pilot. (FCU flood dataset)

For training YOLOv3, the following files are required:

- labelled image dataset (image files and their corresponding labels should be in the same folder)
- pre-trained weights for the convolutional layers darknet53.conv.74 [https://pjreddie.com/media/files/darknet53.conv.74]
- 'train.txt' (contains a list of training images and their directory)
- 'test.txt' (contains a list of testing images and their directory)
- configure file (.cfg)
- '.names' file (contains the semantic name of objects - each in a new line)
- A folder for weights files to be saved in
- '.data' file (indicates the number of classes, and directory of train.txt, test.txt, and .names files, and weights folder)

All of these files, except the one mentioned in item 2 <?>, should be changed based on the custom objects. This process is explained in the following subsections.

A) Training and testing dataset

One of the main requirements for training YOLO is a labelled dataset, which is a folder containing the set of images and a .txt file for each corresponding image. The name of the .txt file should be the same as the image file. For instance, for image_1.jpg, the text file should be image_1.txt. For more information on how to label images visit https://github.com/AlexeyAB/Yolo_mark [https://github.com/AlexeyAB/Yolo_mark].

In the next step, the dataset is split into training and testing groups. The path to each training/testing image is saved into either train.txt or test.txt files. For example:
~/darknet/trainYolo/image/image_1.jpg

B) Network configuration

The next step before training the model is modifying the network configuration. The number of custom classes of objects matters in setting the network parameters. For this purpose, the following changes (Table 13) should be applied to a copy of yolov3.cfg:

Table 14. Required modifications on the configure file

Variable	Line number in yolov3.cfg	Comments	Example for 2 classes of object
Batch	6	Should be set to 64	64
Subdivisions	7	Should be set to 8 If during the training process, memory becomes full, this value should be increased to 16, 32 or 64	8
Max_batches	20	Equals to number of classes times 2000 (It shouldn't be less than 4000)	max_batches=4000
	Steps	22	steps=A, B where: A= (0.8 * max_batches), and B= (0.9 * max_batches)
steps=3200,3600	Classes	610, 696, 783	Number of classes
classes=2	Filters	603, 689, 776	Equals to: (classes + 5) * 3

C) Modifications to .data file

This file contains the following information regarding the model:

```
classes = 2
train = data/train.txt
valid = data/test.txt
names = data/obj.names
```

backup = backup/

The "classes" is the total number of classes, "train" shows the directory of the train.txt file, "valid" indicates the route to the test.txt file, "names" is the directory of the .names file, and "backup" is the

location where the generated weight files are going to be saved in.

D) Starting the training process

Go to the directory of darknet-alexyab [https://github.com/AlexeyAB/darknet?files=1] and run the following command to start the training process:

```
./darknet detector train <path-to-the-.data-file> <path-to-the-.cfg-file> <path-to-darknet53.conv.74> -map
```

For example:

```
./darknet detector train cfg/coco.data cfg/yolov3.cfg darknet53.conv.74 -map
```

During the training process, a graph of mean-Average Precision (mAP) and average loss for each iteration is plotted (Figure 100).

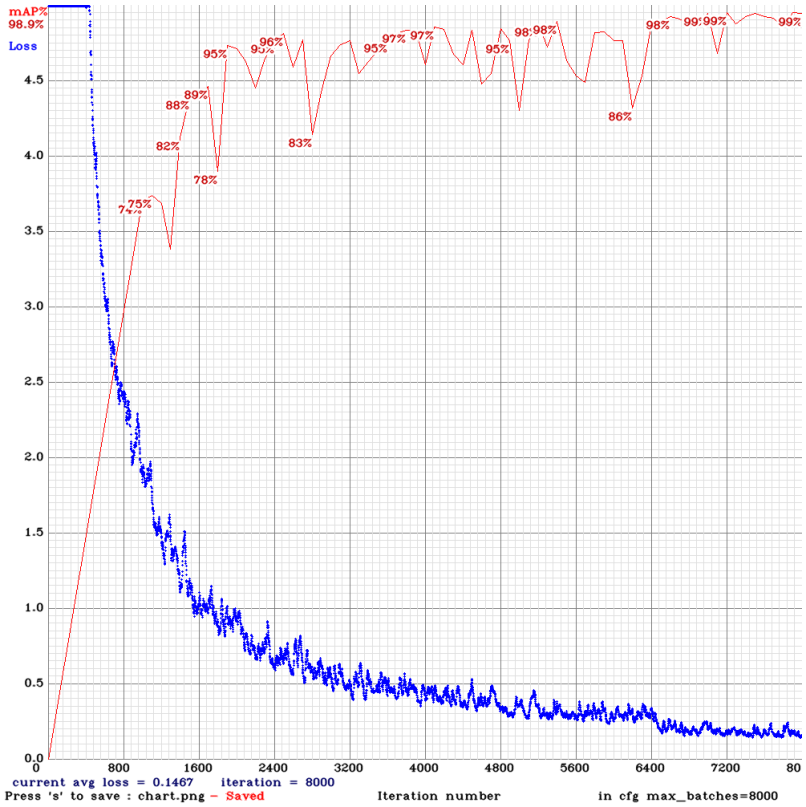

Figure 100. Average loss (blue graph) and mAP (red graph) during the training process for the trained model

E) Evaluation of the trained model

The model is evaluated using the test image dataset (listed in test.txt), and Table 14 shows the evaluation parameters of the trained model with a confidence threshold of 0.25. In this evaluation, the model detected 870 objects in total with a unique-truth-count of 680.
Precision is the ratio of correctly predicted observation of the total observations. Recall ks another

evaluation metric for the model, which is a ratio of correctly predicted observations to all the observations in an actual class (summation of True Positive and False Negative).
F1-score, which is the weighted average of Precision and Recall, gives a better understanding of the model in uneven class distribution.
True-Positive value for this model shows the total number of detections where the model was successful, and the False Positive value indicates how many times the model has failed in detecting objects.
The intersection of Union (IoU) is another evaluation metric that is used to measure the accuracy of the model. The value range of IoU is between 0 and 1: "0" means that there are no overlapping pixels between the two bounding boxes, while "1" indicates that the two bounding boxes are equal. For calculating this metric, two values are needed: A ground-truth bounding box (which is a hand-labelled bounding box from the testing set, that specify where in the image the object is), as well as a predicted bounding box from the model. IoU is the overlap of these two bounding boxes divided by their area of union.

Table 15. Training evaluation parameters for confidence threshold of 0.25

Training Evaluation Metrics	**Values**
Precision	98%
F1-score	95%
Recall	93%
True Positive (TP)	633
False Positive (FP)	16
Intersection of Union (IoU)	80.48%

Appendix C: Revision History

NOTE

Example History (Delete this note).

replace below entries as needed

Table 16. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
December 15, 2019	S. Saeedi	.1	all	initial version
January 31, 2020	S. Saeedi	.5	all	complete version
February 28, 2020	S. Saeedi	.9	all	submittal version
March 4, 2020	S. Saeedi & J. Lieberman	1.0	all	editorial changes & summary
April 28, 2020	S. Saeedi & J. Lieberman	1.0	all	editorial changes

Appendix D: Bibliography

1. Garbrecht, J., Martz, L.W.: The Assignment of Drainage Direction Over Flat Surfaces in Raster Digital Elevation Models. Journal of Hydrology. 193, 204–213 (1997).
2. The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing. 8, 14–23 (2009).
3. Peter Mell, T.G.: The NIST definition of cloud computing. (2011).
4. Joseph Redmon, A.F.: Yolov3: An incremental improvement. (2008).
5. Tourscotland: FloodingVideo: Cars Driving Through Flooding Road Perth Perthshire Scotland, https://www.youtube.com/watch?v=tCqEARMI-_k, (2012).
6. Okraftwerkmod: FloodingVideo: Vehicles crossing a flooded ford 2016 - river crossing, https://www.youtube.com/watch?v=NEkt1QdFHIs&t=104s, (2016).