2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)

Monitoring the Smart City Sensor Data Using Thingsboard and Node-Red

Elham Okhovat
Computer Science Department
University of Western Ontario
London, Canada
eokhovat@uwo.ca

Michael Bauer
Computer Science Department
University of Western Ontario
London, Canada
bauer@uwo.ca

***Abstract*— The ubiquitous use of smartphones, smart devices, and low-cost sensors, has fostered much attention around the Internet of things (IoT). The Internet of Things has led to the manufacturing of more IoT-related products ranging from devices to applications. The sensors in an IoT network provide data on the environment in which they are deployed and interact with other devices and the applications in the broader IoT ecosystem. IoT components do not necessarily adhere to the same standard and so can be very heterogeneous in a particular environment. This can create challenges in the collection and monitoring of heterogeneous data in real-time as well as monitoring the infrastructure itself. Such data collection is not only important for the analysis of the environment but also for monitoring the health of the devices and operational components, such as software elements. In this paper, we introduce an architecture aimed at collecting, visualizing, and monitoring streams of data from sensors and from infrastructure components. The ThingsBoard IoT platform is used for data collection and visualization. Node-Red is used to categorize the data on the sensor name. Experiments using sensor data sets are provided to illustrate the approach, processing, and visualization.**

Keywords*— *Internet of Things, IoT Solution, Smart City, Monitoring, ThingsBoard, Node-Red

I. INTRODUCTION

The Internet of Things (IoT) is a combination of connected sensors, actuators and other smart objects, such as vehicles, smartphones, home appliances, wearable gadgets and even medical equipment, that can interact with each other, send or receive data through the internet or other communication technologies like MQTT [1], COAP [2], etc. Using this technology, people can manage their devices remotely from the comfort of their homes. Although this technology offers many advantages for its users and for companies, there are a number of operational issues. Because IoT often deals with large and continuous data, one of the main challenges for administrators is monitoring the current state of the environment and its data in real-time.

According to an ISO report [3], a smart city should aim to reach its goals by using resources in an efficient and consistent way to guarantee growth and innovation. What is needed in any smart city and IoT network is the constant monitoring of the network status and state of the city devices and sensors and enabling their management. How to address this is the main goal of this paper. In this research, we assume that the smart city infrastructure has several layers from sensors and devices at the base tier to complex IoT platforms, third-party applications and managerial units at the upper tiers as illustrated in Fig. 1. We introduce an architecture to enable the collection of data about the operational environment and illustrate how data collection and visualization can be integrated. We illustrate the process by simulating data collection in a simulated smart city environment. The data collected are then categorized using Node-Red [4] and feed ThingsBoard [5] with the telemetry data. In the end, we visualize the data through ThingsBoard dashboards and provide the management authority with the current state of all the sensory data and trigger alarm according to the pre-defined data status.

The rest of the paper is organized as follows. In section II, the key concepts such as the Internet of Things, smart city and IoT platform are reviewed and we elaborate on how the smart city infrastructure looks like. We then review some previous work in the field of smart cities and review some of the well-known IoT Platforms, their characteristics, and features. In Section IV, we describe our approach and discuss the technologies that we use. We then present the results of experiments and the final section will provide a summary and conclusion.

II. INTERNET OF THINGS AND SMART CITY

According to Cisco [6], the Internet of Things and its applications is the next evolution on the Internet and there would be a huge rise in the use of IoT in our everyday life because the number of connected devices per person is going up. The prediction is that the number of smart objects would be 25 to 50 billion by the year 2025.

The term "Smart City" was coined by Cisco and IBM to describe a city that benefits from information and communications technology and automation. This term was used to refer to "a city that makes a conscious effort to innovatively employ information and communication technologies (ICT) to support a more inclusive, diverse and sustainable urban environment'' [7]. As noted, we consider a smart city infrastructure to be one in which IoT is used to provide connectivity between its entities such as wearable devices, sensors, actuators, smartphones, and even applications, etc. to improve the quality of life for the citizens.

978-1-6654-1236-0/21/$31.00 ©2021 IEEE
DOI 10.1109/SWC50871.2021.00064

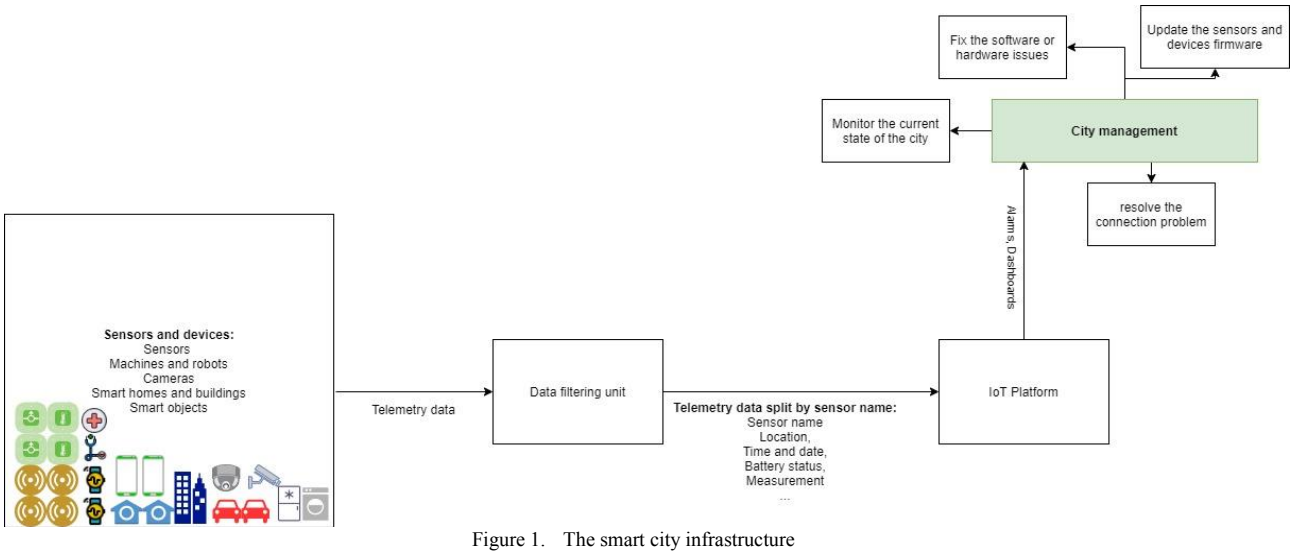


Figure 1. The smart city infrastructure

We assume the existance of a managerial entity responsible for the infrastructure, data, operations, etc. of a smart city. This entity can be a collection of software components, city administrators or both. There are sensors to measure different aspects of the environment which send that data to an IoT platform. We consider the IoT platform as a framework that acts as middleware between the devices, and other heterogeneous components of the IoT network and the cloud. It is responsible for provisioning the resources, receiving the telemetry data, managing and visualizing them using the dashboards, and triggering alarms according to the preset rules. In the next section, we will elaborate more on IoT platforms and look at some well-known ones. We also assume the presence of a data filtering unit that is in charge of categorizing the data based on sensor identifiers. At the top of the infrastructure are the city management entities which are responsible for fixing issues, updating the software and firmware versions and observing the current state of the city.

III. Related Work

A. Smart Cities

There have been many different research efforts related to the idea of smart cities. In the following, we focus on those that are most closely aligned with our work.

Sakhardande et al. [8] proposed a disaster management and smart city monitoring system that employs hardware components rather than software. Each module in the system consists of sensors, actuators, WiFi adapters, power supplies and Arduinos. Each node has a unique identifier and they form a star network. The system has two modes: monitoring and disaster management. In the first mode, the system collects the data and sends it to the dedicated cloud servers for processing. Each node, upon receiving the alert signal, terminates its monitoring task and changes its mode to disaster management. This node then transmits that signal to other networks that it is connected to and this process continues until the signal reaches the nodes that are located in the affected area. The use of an Arduino in each module does not seem reasonable for a smart city in which there are a massive number of sensors.

In [9], the authors created a single dashboard that provides the important measurements for the current state of the city of Bandung, Indonesia. These measurements include temperature, air and water quality, traffic and so on. Their architecture consists of the sensors deployed in the city to collect the data, and servers are connected to sensors using the internet. The user application is the unit that provides the dashboard on which the sensor data and the current state of the city are shown. The drawback of their solution is that the management sector cannot perform anything more than just monitoring the city's condition (air and water quality, temperature, traffic).

In [10], a city dashboard called My City Dashboard is introduced to process real-time data such as temperature and noise A vast variety of technologies are used in different layers of this architecture such as Apache Kafka and RabbitMQ that are used for the acquisition layer, Apache spark and Flink that are utilized for processing layer, MongoDB and PostGIS modules for PostgreSQL that are employed in persistence layer and dashboard layer is developed using RESTful services. The dashboard that is provided by this solution represents the city as a map that is divided into tiles and for each tile, statistics are shown, such as the average, maximum and minimum. This solution is limited to monitoring the state of each stream type, like noise, temperature and pollution, for a specific tile.

An IoT architecture called Souly is proposed in [11] that focuses on monitoring and management of smart buildings and hotels as part of the IoT environment. This system consists of sensors and actuators, IoT gateway, cloud platform, administration panel, API modules and mobile application. The main component of the system is a cloud platform which is responsible for data collection and processing, as well as providing reports and data monitoring. The data is stored in two databases, one for storing short-term data that can be used for

monitoring purposes and the other database that stores room configuration and tenant information. This system uses an MQTT broker and receives the data using this protocol. Souly can provide information about the state of the devices in the building through the mobile and web panels as well as logs related to problems in memory allocation, applications and even unsuccessful logins. This architecture is also capable of triggering notifications regarding device failures.

In [12] the authors used ThingsBoard and Spark to collect and analyze the data. In this work, data is first collected via ThingsBoard through MQTT and after that Apache Kafka helps to send the data to Spark Streaming. Spark receives the data and starts analyzing it after eliminating outliers and doing some data cleaning. This architecture is utilized in a smart health scenario to monitor the condition of patients who suffer from respiratory problems. The sensors can provide measurements such as temperature, humidity, level of oxygen, CO_2 and NO in the patient body.

The work in [13] proposed a system that provides real-time monitoring data of water quality. This system is based on a wireless sensor network and developed using BIO (Bristol Is Open) to provide an experimental environment. Their proposed architecture is made of several modules from data acquisition, power supply and data transmission to data storage and redistribution module. The data is sent to data storage using WiFi and TCP/IP. For visualization purposes, they used Grafana, which is a web application that is able to analyze and visualize the data through charts, graphs, etc. Although this monitoring system is very good in terms of measuring water quality parameters and visualizing them, it does not produce alarms and does not filter the data before sending it to the database.

In [14] the authors proposed a monitoring system to measure th PH and moisture in the soil. PH sensors and soil moisture sensors send their measurements to the client node which consists of a LoRa transceiver, Arduino microcontroller, Wemos d1, and power supply. Then client node transmits that data to the master via a LoRa protocol every 8 hours. After that, the master unit that is connected to the Internet transfers the data to the cloud for monitoring. Lastly, the sensor values are presented to the user in three different forms: the latest value, table and chart. This system also has a status icon that informs the user about the soil condition.

A monitoring and alarm system is proposed in [15] that is developed for drainage and waste management. This system can detect the blockage or high amount of toxic gases and generates alarms. It has several hardware and software components such as a DC power supply, an ultrasonic sensor, a gas sensor, GPS, Driver circuit, IoT development kit, Arduino microcontroller and blunk app for creating graphical dashboards. Unfortunately, the authors did not provide detail about their proposed system. Based on the block diagram provided it seems that the system sends the sensor data to the cloud using MQTT but it does not have any data cleaning step before using the data.

The last related paper studied is [16] which is the most complete monitoring system we could find on this topic. This paper introduces a framework that is a combination of monitoring systems that can monitor every aspect of a smart city from air pollution to temperature and so on. Their framework uses data analysis techniques and it can even predict how the data affects the smart city. In their framework for each smart city property, they created a separate monitoring system which includes air and noise monitoring and control system, a speed and web monitoring system (for vehicles), temperature and weather monitoring system (UV, wind,…), fire detection system, waste management system, geographic information system. Their architecture also consists of a power supply, a display, a central control room and an alarm system. In their smart city model, all of the sensors and their corresponding systems are connected to a microcontroller unit(MCU) and the data are processed in that unit. Data analysis is done in this central unit to check whether the data are at dangerous levels and, if so, the system creates alarms, otherwise the data is stored and displayed to the user.

Although the aforementioned works are used for monitoring and observing smart city conditions, each has some limitations. For example, in the Sakhardande et al. [8] approach they focused on hardware and each module of their approach had to have all hardware elements from sensors to the Arduino. In the work done in the city of Bandung [9], there is no means for triggering alarms based on changes in the stream of data. My City Dashboard [10] can present the statistics for each tile on the map or cluster, but the manager is unable to extract the information for a specific sensor or building in that cluster, therefore, they cannot locate the origin of the problem to fix the issue. Although Souly [11] is quite similar to our approach and is able to monitor the sensors, detect issues and even trigger alarms, it is created specifically for hotels and buildings and focuses on one building or hotel at a time. Even though in [10] the authors used the same IoT platform as ours, they directly feed it with their data and after that, they do data cleaning and eliminating the outliers which could have been done before data reaching to the ThingsBoard.

In contrast, our approach aims to monitor multiple assets and entities which is required in broader smart city scenarios. The other benefits of our approach include the existence of a data filtering unit between the devices and the platform to save the resources, no need to add any extra components to the infrastructure, an approach that is scalable which means that the data can be shown for a specific sensor, a building, a group of buildings or even the whole city, that the management component has access to the monitoring dashboards and the current state of the city in real-time and it can also be notified of any dangerous situation.

B. IoT Platforms

IoT platforms are mostly responsible for collecting, managing and monitoring the data coming from the sensors and devices in an IoT network. Some of the more advanced platforms can even trigger alarms based on the data or control the actuators and devices in the network. In the following, we look at some of the common IoT platforms and their features.

FIWARE [17] is an open-source platform that can be used to develop smart solutions for the Internet of Things such as smart cities, smart industry, smart energy and so on. The main component of FIWARE is the context broker. Some third-party components can be added to the context broker to make it more

compatible with different business needs. These components enable FIWARE context brokers to deal with context data, processing, analysis and visualization of data, and even publication or monetization [17]. This platform does not provide message encryption that negatively affects its security [18].

Snap4City [19] is another solution that can help developers to implement the IoT application for a smart city. This platform not only meets the functional requirements but also satisfies the non-functional requirements such as scalability, Interoperability, heterogeneity, robustness, security and so on which are important in an IoT network. Snap4City adopts the ontology for data representation and approaches for detecting the new sensors and actuators from Km4City and makes it scalable to be able to handle the massive number of events that happen in the smart city. Snap4City architecture includes components for data collection, data storing and management, IoT application/microservice and service development, smart city process control and execution, etc..

The next IoT platform is thethings.io [20] which claims flexibility and scalability and it is not dependent on specific hardware or protocol. It offers a wide variety of capabilities such as device management, data visualization and monitoring using customized dashboards, analysis, integration, alert generation, security and privacy, etc.

Another multi-functional platform for data collection, processing and sharing is Orchestra Cities [21] which is an extension of the FIWARE IoT Platform. Orchestra Cities is an open-source and open API platform that allows defining and monitoring IoT applications and services in an IoT network and is designed to connect smart devices and citizens in a collaborative environment. This platform not only focuses on the citizens and individuals but also tries to consider the industry and public sectors. It can support current communication protocols, cloud services and offers management aspects of a smart city such as security management, device management, data management, dashboard management, and data integration management.

OpenIoT [22] is an open-source cloud solution that is designed to discover and manage sensors, devices and actuators and support communication between the entities and applications in an IoT network. This platform can configure the algorithms for collecting and filtering the data as well as generating the events. Unfortunately, the latest update of this java-based project goes back to Nov 2015.

Thinger [23] is also another open-source IoT platform that is developed by a Spanish company and provides a console to easily manage and monitor the devices in the cloud environment and it can support different hardware such as Arduino, Raspberry Pi and so on. This platform can offer bidirectional communications between heterogeneous devices, store and show and share the data with the dashboards, and trigger alarms. One of the negative points of this platform is that it does not support messaging protocols other than HTTPS [18].

Linksmart (Hydra) [24] is a free European IoT platform that like the other IoT platforms offers device management, communication and data analysis and monitoring. This platform can be used in smart building, smart energy, waste management and other projects in the realm of the smart city. Its architecture consists of device integration and abstraction layer, service provisioning, data management and processing, network and security, and human-computer interaction [25]. Linksmart components include a service-oriented architecture (SOA), model-driven approach, 3-layered discovery architecture, p2p-based network architecture, dynamic runtime architecture, context management, self-* management, security and trust enabled, and storage management [26].

Thingsboard [5]. is a flexible and scalable open-source IoT platform which is written in Java 8 and can offer a wide variety of capabilities that are needed in an IoT network such as provisioning and management of actual and virtual sensors and devices, data collection and analysis, visualizing the real-time data with widgets on the dashboards, triggering alarms based on the data fluctuations, controlling the devices, and so many more. This platform is very scalable and customizable and can support different standard IoT communication protocols, like MQTT, COAP and HTTP, also it can define and manage users, customers, devices, assets, alarms and dashboards.

ThingsBoard has been selected as our IoT platform because it outweighs the other platforms due to the following reasons. It is an open-source platform, its community on GitHub is more active compared to the other ones, its user interface is consistent and clear. Moreover, ThingsBoard is very extensible and allows creating custom widgets to be added to the dashboards, therefore you are not bound to use the default widgets. The deployment options for this platform can be on-premises such as Linux, Windows, Raspberry Pi, docker, or in the cloud such as AWS, google cloud and so on. In addition, this platform can support HTTP, CoAP, MQTT and even other protocols such as LoRaWAN and SNMP [27]. If there is a need for more features, it can be upgraded to the Professional edition which provides more facilities. Also, ThingsBoard offers PaaS to support cloud-based architecture as well.

IV. Approach

For our prototype architecture, we assumed that our smart city contains several sensor objects whose data are stored in data files. Our data filtering unit receives the sensor data from sensor objects via communication protocols. Our architecture consists of several components that help the city management to monitor the current state of the city and its sensors and devices, address problems, and take action based on high-level goals. This architecture, as shown in Fig. 2, provides the structure to enable components to collect the telemetry data from the sensors deployed in the smart city, categorize, and visualize them and provide a complete and real-time view of what is going on in the city. The telemetry data is composed of the measurement information about the environment in which the sensors are deployed and sometimes the details about the sensors and devices themselves. Our prototype architecture works as follows. Initially, the sensor data are collected and logged in a dataset then Node-Red receives the dataset, filters, and categorizes the data based on the sensor name and transmits each data stream to its corresponding sensor in the ThingsBoard. Finally, the ThingsBoard IoT Platform is utilized to provide data visualization, analyze the data, and trigger alarms based on data fluctuations; this procedure is described in more detail in section IV.C. The descriptions of Node-Red and ThingsBoard are presented in the following section.

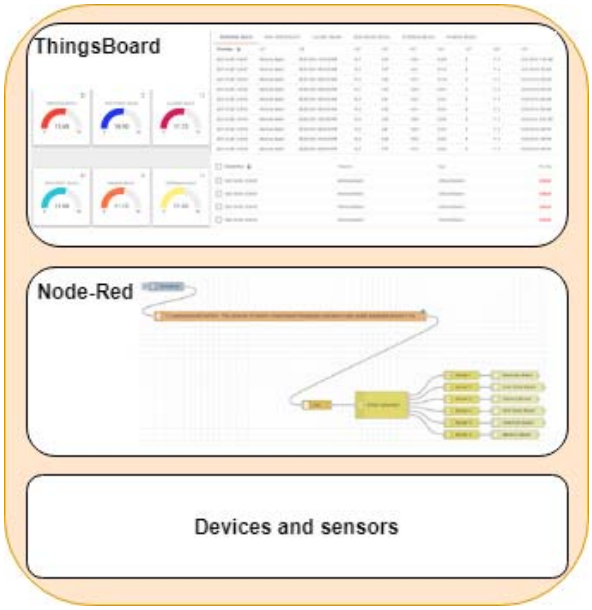


Figure 2. Proposed Architecture

A. *Node-Red*

Node-Red flow [4], which was developed by IBM, is a flow-based programming tool that makes the presentation of the behaviour of the system or application and it is built on the node.js runtime. The Node-Red flow contains specific blocks of code called nodes that are wired together, and the data passes through the nodes. This open-source tool is ideal for event-driven applications and can be applied to a variety of system architectures from the on-premises to the cloud environment like IBM Cloud, AWS, and Microsoft Azure. The Node-red editor provides some pre-built nodes, but new nodes can be created or installed to extend their capabilities. These nodes can be dragged and dropped from the palette to the workspace and then they should be wired together to create a flow.

B. *ThingsBoard*

We briefly described ThingsBoard in the previous section, but here we will provide a short description of its architecture, the different editions, and a comparison between them. ThingsBoard architecture consists of four main components as specified and described below [28]:

- The ThingsBoard Transport component is responsible for receiving the messages from the sensors and devices in the network using communication protocols, parsing the message and pushing it into the message queues.
- ThingsBoard Core that handles REST API calls, WebSocket subscriptions on entity telemetry and attribute changes, storing the information about device sessions, and monitoring device connectivity state.
- ThingsBoard Rule Engine is considered as the main component of the system and it subscribes to incoming messages and processes them. This component uses Node-Red for creating the rules and flows.
- The last component is ThingsBoard web UI that is stateless and is written in Express.js.

Because ThingsBoard is very flexible it not only can be launched on a local computer but also can be deployed on a cloud, such as AWS, Azure and so on. ThingsBoard offers two types of architectures: Monolithic and Microservices. Based on the amount and type of the data, it provides different storage options like SQL (PostgreSQL, HSQLDB), NoSQL (Cassandra) and a hybrid approach that stores all the data related to ThingsBoard itself in PostgreSQL and time-series in Cassandra or timescale DB.

Thingsboard provides two on-premises editions. Community edition which is the limited version and free, and the Professional Edition with some extended features. For this research we use the Community Edition that provides the following features [29]:

- Attributes, that are the information about the sensors or devices such as name, location, battery life, and so on.
- Data collection. This data is the measurement data coming from the sensors or devices. For example, a pressure sensor gives information about the pressure of gas or liquid.
- Entities and relations. Supported entities are devices, tenants, assets, users, etc. Using ThingsBoard we can define the relation between two entities for example a building and the sensors deployed in it.
- Real-time data visualization. The data can be shown in the form of charts, tables and even in customized widgets.
- Rule engine, which is a framework to create event-based workflows.
- Remote procedure calls that are APIs and widgets to push commands and control the devices
- Tracking and auditing the user's actions and storing them as an audit log.

In this research, our deployed ThingsBoard uses PostgreSQL for both entities and telemetry data and it can handle about 5,000 telemetry data points per second but for projects with a larger scale, Cassandra can be used which can handle more than 20,000 data points per second [30].

C. *Sensor data classification and visualization*

In this paper, the purpose is to visualize telemetry data coming from the sensors. For experimentation purposes, we use data collected from sensors used to monitor the quality of water deployed at Chicago beaches [31]. This dataset provides information about beach-name, water temperature, turbidity, Transducer-depth, wavelength, wave-period, and battery life and contains more than 34,900 records. Because ThingsBoard itself does not offer some of the crucial nodes for classification purposes in its rule chain, we installed Node-Red on the local computer to split and filter the data. As shown in Fig. 3, a flow is created that starts with reading our sensor data from a file and classifying the data based on the sensor name using the switch node. Then each stream of data is guided to the corresponding output. After that, this output is sent to the related sensor in ThingsBoard using the HTTP protocol. To determine the specific sensor on ThingsBoard, the device access token is used that can be obtained from ThingsBoard device details.

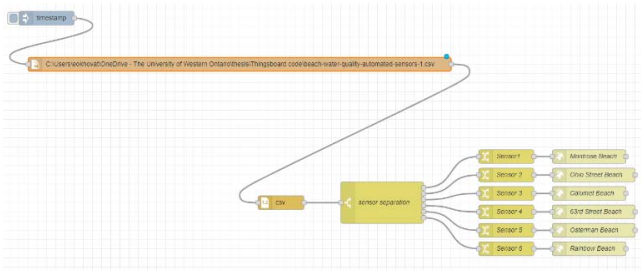


Figure 3. Node-Red flow to send the data to ThingsBoard

On the ThingsBoard side, first, an asset is created for each beach. An asset is a specific entity such as a building or a house that has several sensors and can be managed using ThingsBoard. Next, the same number of sensors are added to the device management section and they are assigned to an asset. Then each sensor in ThingsBoard receives the telemetry data that belongs to it and shows it in the latest telemetry tab. This is one of the advantages of ThingsBoard that it can receive several streams of telemetry data at the same time.

ThingsBoard offers a wide variety of widgets to create and customize a dashboard to visualize the data in real-time. It is also possible to develop a widget using HTML, CSS, and JavaScript in its built-in IDE. In the next section, some experiments are described along with visualization of data.

Other than the aforementioned advantages of our approach presented at the end of Section III.A, the novelty of our work also lies in having a data filtering unit which cleans the data before reaching the IoT platform. Most of the works that have been done on this topic do not have any data filtering unit between the devices and the IoT platform. This feature ensures that the data is clean and free of anomalies. Also, in this unit, we can apply machine learning techniques to the data before feeding ThingsBoard.

V. EXPERIMENTS

In this section, we present experiments done using ThingsBoard and our prototype architecture. We start with the representation of the state of one of the sensor attributes for all the deployed sensors. Then we continue by providing all the data for each sensor respectively. Lastly, we configure ThingsBoard to create an alarm when a certain condition is met. Our dataset is taken from [31] that consists of the sensory data coming from the sensors deployed in the water at several beaches in Chicago.

A. Dashboard to show the state of an attribute for all sensors

First, we focus on just one attribute that is common among all the sensors. There are six sensors and we want to monitor their battery life. The dashboard contains six digital gauges that represent the state of the battery for each sensor. For example, in Fig. 4 the battery life of all the sensors can be seen in real-time and the administrator can check them regularly to ensure that they are in a good condition. Alarms could also be defined and generated when the battery level goes below a specific threshold. In the next section, we will talk about the alarms and how they are shown on the ThingsBoard dashboard.

Figure 4. ThingsBoard dashboard to show battery life of the sensors

B. Visualization of all of the telemetry data by sensor location

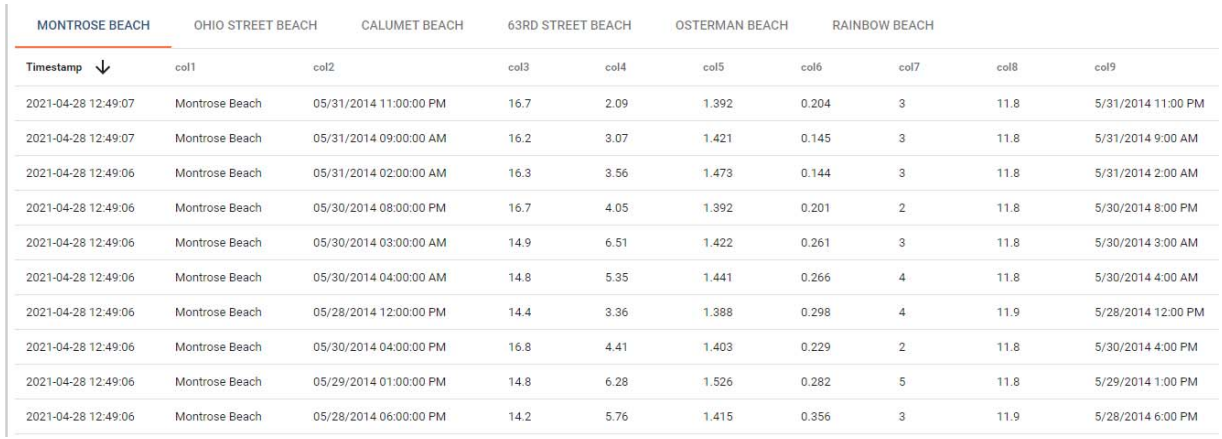
For our second experiment, we want to show all the data coming from all of the sensors deployed on the beaches. For each asset, a separate dashboard can be created to show the data for that specific asset. To present and visualize the data in the ThingsBoard dashboard, a time-series table widget is used to get the telemetry data from the sensors and show them in the table organized in separate sections split by tabs with the sensor location as shown in Fig. 5. After starting the Node-Red flow, in this dashboard, the data records start to appear one after the other.

C. Triggering alarms and generate notifications based on the sensor measurements

Finally, we create alarms based on the sudden changes in the data. For example, there is a sensor in every beach responsible for measuring wave height. By monitoring this sensor, we can create alarms to notify the people who live near the sea to be ready for an imminent tsunami or even to evacuate their homes in case of a disastrous situation. In this use case, we define a limit for the wave height. When the wave height exceeds that limit an alarm is created with the time and severity. Fig. 6 depicts the alarm dashboard which is created for one of the beaches.

D. Filtering the data and ignoring the data records with null values

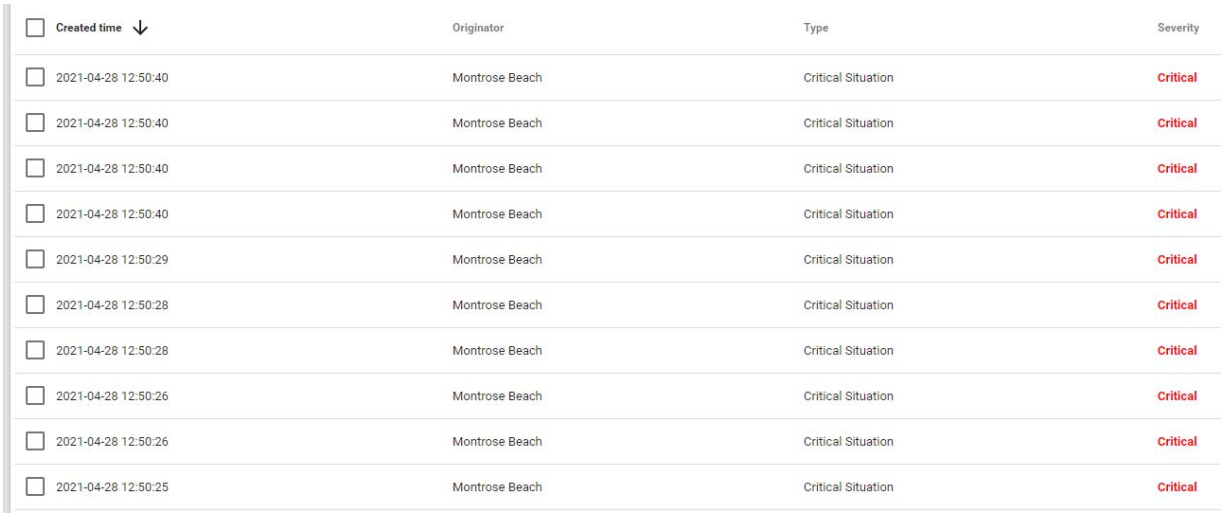
By looking at the data tables we noticed that there are some null records in our dataset in which only the beach name is available, and the rest of the record is empty. Because null values adversely affect the performance of any system, we decided to get rid of them before they reach our IoT platform to prevent data redundancy in the ThingsBoard database and save our resources which is very critical in any smart city. In order to ignore them as shown in Fig.7, we added the “clean data” node after splitting the data in Node-Red and then the data is navigated to the related sensor in ThingsBoard.



MONTROSE BEACH | OHIO STREET BEACH | CALUMET BEACH | 63RD STREET BEACH | OSTERMAN BEACH | RAINBOW BEACH

Timestamp	col1	col2	col3	col4	col5	col6	col7	col8	col9
2021-04-28 12:49:07	Montrose Beach	05/31/2014 11:00:00 PM	16.7	2.09	1.392	0.204	3	11.8	5/31/2014 11:00 PM
2021-04-28 12:49:07	Montrose Beach	05/31/2014 09:00:00 AM	16.2	3.07	1.421	0.145	3	11.8	5/31/2014 9:00 AM
2021-04-28 12:49:06	Montrose Beach	05/31/2014 02:00:00 AM	16.3	3.56	1.473	0.144	3	11.8	5/31/2014 2:00 AM
2021-04-28 12:49:06	Montrose Beach	05/30/2014 08:00:00 PM	16.7	4.05	1.392	0.201	2	11.8	5/30/2014 8:00 PM
2021-04-28 12:49:06	Montrose Beach	05/30/2014 03:00:00 AM	14.9	6.51	1.422	0.261	3	11.8	5/30/2014 3:00 AM
2021-04-28 12:49:06	Montrose Beach	05/30/2014 04:00:00 AM	14.8	5.35	1.441	0.266	4	11.8	5/30/2014 4:00 AM
2021-04-28 12:49:06	Montrose Beach	05/28/2014 12:00:00 PM	14.4	3.36	1.388	0.298	4	11.9	5/28/2014 12:00 PM
2021-04-28 12:49:06	Montrose Beach	05/30/2014 04:00:00 PM	16.8	4.41	1.403	0.229	2	11.8	5/30/2014 4:00 PM
2021-04-28 12:49:06	Montrose Beach	05/29/2014 01:00:00 PM	14.8	6.28	1.526	0.282	5	11.8	5/29/2014 1:00 PM
2021-04-28 12:49:06	Montrose Beach	05/28/2014 06:00:00 PM	14.2	5.76	1.415	0.356	3	11.9	5/28/2014 6:00 PM

Figure 5. ThingsBoard time-series dashboard to show sensor telemetry data



Created time	Originator	Type	Severity
2021-04-28 12:50:40	Montrose Beach	Critical Situation	Critical
2021-04-28 12:50:40	Montrose Beach	Critical Situation	Critical
2021-04-28 12:50:40	Montrose Beach	Critical Situation	Critical
2021-04-28 12:50:40	Montrose Beach	Critical Situation	Critical
2021-04-28 12:50:29	Montrose Beach	Critical Situation	Critical
2021-04-28 12:50:28	Montrose Beach	Critical Situation	Critical
2021-04-28 12:50:28	Montrose Beach	Critical Situation	Critical
2021-04-28 12:50:26	Montrose Beach	Critical Situation	Critical
2021-04-28 12:50:26	Montrose Beach	Critical Situation	Critical
2021-04-28 12:50:25	Montrose Beach	Critical Situation	Critical

Figure 6. ThingsBoard alarm dashboard

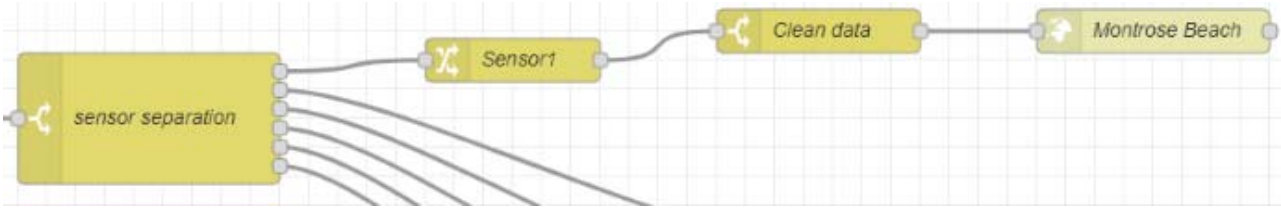


Figure 7. Data cleaning sub-flow in Node-Red

VI. Conclusion

In this paper, we proposed an architecture consisting of Node-Red and ThingsBoard IoT Platform to collect the smart city data, clean and categorize them, and monitor them in real-time. At last, three different visualizations were applied to the data to provide in-depth monitoring of the IoT network and Smart City ecosystem.

Our future plan is to monitor not only the data coming from the sensors and devices in the city but also the performance metrics of the whole architecture such as CPU, throughput, memory usage , response time and so on. Moreover, as is evident, in a smart city we must deal with a massive number of applications and devices, and they might encounter failures, get disconnected or even face delays and bottlenecks. As the network and smart city grow, it is preferable that the management tasks be done autonomously without any human intervention. Therefore, we consider going further and provide a complete autonomous management system to manage the whole ecosystem and fix any issue right away 24/7.

REFERENCES

[1] "MQTT - The Standard for IoT Messaging", Mqtt.org, 2021. [Online]. Available: https://mqtt.org/.

[2] Z. Shelby, K. Hartke and C. Bormann, "The Constrained Application Protocol (CoAP)," June 2014.

[3] I. t. ISO/IEC JTC 1, "Smart cities," International Organization for Standardization, Geneva, 2014.

[4] "Node-RED", Nodered.org, 2021. [Online]. Available: https://nodered.org/.

[5] "ThingsBoard - Open-source IoT Platform", ThingsBoard, 2021. [Online]. Available: https://thingsboard.io/.

[6] D. Evens, "The Internet of Things - how the next evolution of the internet is changing everything," CISCO white paper, 2011.

[7] U. Rosati and S. Conti, "What is a Smart City Project? An Urban Model or A Corporate Business Plan?," *Procedia - Social and Behavioral Sciences,* vol. 223, pp. 968-973, 2016.

[8] P. Sakhardande, S. Hanagal and S. Kulkarni, "Design of disaster management system using IoT based interconnected network with smart city monitoring," in *2016 International Conference on Internet of Things and Applications (IOTA)*, 2016.

[9] S. Suakanto, S. H. Supangkat, Suhardi and R. Saragih, "Smart city dashboard for integrating various data of sensor networks," in *International Conference on ICT for Smart Society*, 2013.

[10] C.-C. Usurelu and F. Pop, "My City Dashboard:Real-time Data Processing Platform for Smart Cities," *Journal of Telecommunication and Information Technology,* pp. 89-100, 2017.

[11] M. Dryjanski, M. Buczkowski, Y. Ould-Cheikh-Mouhamedou and A. Kliks, "Adoption of Smart Cities with a Practical Smart Building Implementation," *IEEE Internet of Things Magazine,* vol. 3, no. 1, pp. 58-63, 2020.

[12] L. T. De Paolis, V. De Luca and R. Paiano, "Sensor data collection and analytics with thingsboard and spark streaming," in *2018 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS)*, 2018.

[13] Y. Chen and D. Han, "Water quality monitoring in smart city: A pilot project," *Automation in Construction,* vol. 89, pp. 307-316, 2018.

[14] A. F. Rachmani and F. Y. Zulkifli, "Design of IoT Monitoring System Based on LoRa Technology for Starfruit Plantation," in *TENCON 2018 - 2018 IEEE Region 10 Conference*, Jeju, Korea, 2018.

[15] A. M and B. A, "Iot Based Drainage and Waste Management Monitoring and Alert System for Smart City," *Annals of the Romanian Society for Cell Biology,* vol. 25, no. 3, pp. 6641-6651, 08 March 2021.

[16] S. Jha, L. Nkenyereye, G. P. Joshi and E. Yang, "Mitigating and Monitoring Smart City Using Internet of Things," *Computers, Materials & Continua,* vol. 65, no. 2, pp. 1059-1079, 2020.

[17] "Developers - FIWARE", FIWARE, 2021. [Online]. Available: https://www.fiware.org/developers/.

[18] S. Hill, *Scalable IoT platforms,* Stuttgart, 2019.

[19] C. Badii, E. Belay, P. Bellini, d. Cenni, M. Marazzini, M. Mesiti, P. Nesi, G. Pantaleo, M. Paolucci, S. Valtolina, M. Soderi and I. Zaza, "Snap4City: A Scalable IOT/IOE Platform for Developing Smart City Applications," *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI),* 2018.

[20] "thethings.iO – The Most Simple Enterprise IoT Platform", thethings.iO, 2021. [Online]. Available: https://thethings.io/.

[21] "Orchestra Cities", Orchestra Cities, 2021. [Online]. Available: https://www.orchestracities.com/.

[22] "OpenIoT – Open Source cloud solution for the Internet of Things", Openiot.eu, 2021. [Online]. Available: http://www.openiot.eu/.

[23] A. Bustamante, "Home", Thinger.io, 2021. [Online]. Available: https://thinger.io/.

[24] "LinkSmart® - Free, Open Source IoT Platform", Linksmart.eu, 2021. [Online]. Available: https://linksmart.eu/.

[25] "DistributedFog: LinkSmart", Distributedfog.com, 2021. [Online]. Available: http://distributedfog.com/middleware/linksmart/.

[26] C. o. i. partners, "Networked Embedded System Middleware for Heterogeneous Physical Devices in a Distributed Architecture," The Hydra consortium, 2011.

[27] "Getting Started with ThingsBoard", ThingsBoard, 2021. [Online]. Available: https://thingsboard.io/docs/getting-started-guides/helloworld/.

[28] "ThingsBoard architecture", ThingsBoard, 2021. [Online]. Available: https://thingsboard.io/docs/reference/.

[29] ThingsBoard Documentation", ThingsBoard, 2021. [Online]. Available: https://thingsboard.io/docs/.

[30] Thingsboard, "ThingsBoard IoT platform deployment scenarios," ThingsBoard. [Online]. Available: https://thingsboard.io/docs/reference/iot-platform-deployment-scenarios/.

[31] "Beach Water Quality- Automated Sensors", DataHub, 2021. [Online]. Available: https://datahub.io/JohnSnowLabs/beach-water-quality---automated-sensors.