Development of a Novel Approach to Search Resources in IoT

Nisar Hussain[1], Dr. Tayyaba Anees[2], Muhammad Rehan Faheem[3], Momina Shaheen[4]
School of System and Technologies
University of Management and Technology
Lahore, Pakistan

Muhammad Imran Manzoor[5]
Department of CS and IT University of Lahore, Pakpattan Campus
Pakpattan, Pakistan

Aimen Anum[6]
Department of Computer Science
Comsats University Sahiwal Campus
Sahiwal, Pakistan

***Abstract*—Internet of Things (IoT) referred to interconnected the world of things like physical devices, cars, sensors, home appliances, actuators and machines embedded with software at any time, any location. The increasing number of IoT devices facing challenges which are registration, integration, describing sensor, interoperability, semantics, security, discovery and searching. The current systems are suitable for limited number of devices. Our ecosystem change day by day which means we have billions and trillions of devices connecting to the Internet in future. One major challenge in current system is searching of suitable Smart Things from a millions or even billions number of devices in IoT. For the purpose of searching and indexing, some discovery methods and techniques are discussed and compared. Those techniques and methods are studied and find out the limitations and issued of the current system. Another challenge to searching the Smart Things is a variety of description models for describing the Smart Things. In this piece of work, a novel search engine is proposed to search the Smart Things with variety of description models. A web interface is implemented in this research with HTML, JSON and XML formats. The description models of Smart Things SensorML, SensorThings API and W3C JSON-LD are implemented in the current proposed system.**

***Keywords*—IoT; IoT resources; search engine for IoT; SensorM**

I. INTRODUCTION

The physical objects have the ability to connect with Internet and share data in respect of Internet of Things. In 2020 the connected devices that are able to communicate with Internet are more than 50.2 billion according to CISCO prediction [1]. These physical objects are called Smart Things. IoT provide a range of connectivity protocols, applications and mechanisms to interoperability with existing infrastructure.

In this regard, it‘s essential to propose a search engine to search the devices with the context of the user requests. There are some standardized sensors description technologies which are currently used in many systems. These technologies are SensorML [2], SensorThings API[3] and W3C JSON-LD [4]. These technologies used different languages and formats. In this research, we use all these technologies and parse into a common format for supporting our system. Indexer helps us to store and retrieve the data quickly than rational database systems. Indexer also have the abilities to store structure and non-structure data like simple text, relation Database data, PDF format, JSON format, XML format and other structured data.

Due to increasing significantly number of devices is a problem of find out a proper device which is fulfills the requirements of the user as well as machine transferable code. Only searching is not enough but also supports reliability and robustness with top ranking resources. It is also difficult as compare to searching traditional web documents because of traditional search on search limited attributes of web documents. Another problem is that there is not a signal standard of describing the devices meta-data. This problem creates another challenge which is lack of describing standard for describing functionality of the devices and resource.

A. Problem Statements

IoT platform is a scalable network that have huge amount of resources which is needed to be fast search and store. Traditional technique to store the resource description and meta-data using database system has some limitations like storing documents, non-structured data and rank up the search resources. These limitations can be solved by the indexer which can store resource description and provide facility to ranking the using scoring technique. In this research we use different sensor description formats to index descriptions of resources and also ranking up the search results for high performance and reliability. As compare to database systems indexer is more flexible to storing and searching of large data. Only indexer is not enough to search the resources another requirement is user interface which can help user to search resources.

B. Objectives

In this research different sensor description models are studied and find common descriptions. Different models have different properties and different technologies so we need to parse each model into a shared schema. The parsed description will used further process for a common model to store into indexer. Indexer store description of thing and supply interface to query and sorted its description. After storing the description we need an interface to query the stored information and apply different algorithms to rank the sensors.

- This research compares different sensor description models.
- Find similar fields between different models and descriptions of Smart Things.
- On similarity base set fields for storing in indexer under single core.

C. Contribution

This research work has developed a search engine for IoT with following functionalities:

- Parsing Smart Things descriptions: Every description model uses different languages for describing the Smart Things so there is need to parse the description from a related parser.
- Indexing of variety of description models: OGC SensorML, SensorThings API and W3C JSON-LD description models are tested in this research.
- Collecting required meta-data from descriptions: Common fields are fetched from description models and bind into a common format without losing the data.
- Searching of Smart Things: This research can help to search the Smart Things using keywords, types of Smart Things, name of location and Geo-Spatial searching within defined range.
- Open API: RESTful API service is provided for indexing and searching the Smart Things.

II. State of the Art

A. Internet of Things

The term Internet of Things (IoT) was not officially named until 1999. The pioneer in this term is a British Kevin Ashton that describes a device in the physical world that can communicate with Internet using sensors [5]. After some time Ashton shows the connectivity of Radio-Frequency identity [6] tags which are used in industry to calculate and track items without human interference. Today, the IoT has emerge as a modern term for representing sensors Things connected to Internet with description of computing capability cover objects, devices, sensors, and other daily used items.

A variety of groups studies the wide range of forecasts approximately the potential impact of Internet of Things on the Internet and the economic system throughout the following five to ten years. Cisco, predicted that the number of IoT objects connected to the Internet is greater than 50.2 billion by means of 2020 [7]; Morgan Stanley, however, predicted 75 billion networked objects in 2020. Huawei predictions 100 billion IoT connected devices in 2025 [8]. McKinsey Global Institute indicates that the economic impact of IoT on the international economic system may be as tons as 3.9to 11.1 trillion with the aid of 2025 [9]. The large number of IoT object is grouped and define a term Smart which leads to smart home, smart transportation, smart grid station, smart vehicle etc.

A simple approach is discussed in [10] to discover IoT resources. Discovery is a technique to acquire the data and resources without the knowledge of the source of the data using some discovery applications.

The main characteristic of IoT is "heterogeneity" which means the devices, sensors, and actuators are diverse in nature. The IoT devices used different protocols, different hardware, different data rates etc. The seconds challenge is storage capabilities, energy capabilities and lastly, the format of data ("audio", "video", "streams", "numeric", "textual") producing are also diverse in nature and the standards also. This diversity poses the challenge to discover things. IoT is an important source of producing data that is termed as big-data. The diversity in IoT data creates a challenge to discover the required data for the specific organizations also has the challenge of store heterogeneous data. We need a common framework which covers machine readable representation of data from different formats and stores. There is also need an interoperable mechanism to interoperate devices. These all challenge are needed to be solved for discovery.

The current technique to discover resources is directly connected with the resources using some applications that are restricted only to publisher resources not to discover publicly available resources. The application only discovers the publisher resources which is limited area because of publisher implement only own resources technologies to discovery but in the real world, the discovery mechanism covers all devices and resources which are available publicly and also support to the diversity of the protocols, formats, standards etc. To solve the issue of discovery there is need to describe the resources and devices in a standard format and need to subscribe itself with the discovery engine. The description models solve the diversity problem and discovery engine can easily parse the description of the sensors and resources. So the Discovery Engine provides the interfaces to machines and also for humans to discover the resources. Discovery phase has multiple paths to discover the real world objects using some structured models and apply some kind of knowledge driven queries for context understanding which is possible using discovery mechanism.

B. Resource Discovery Techniques

In [11] comparison analysis for different resource discovery mechanism that are currently available as well as for future aspects is discussed. Author also proposed a search engine based solution for discover resources. Following are the techniques that author compare.

1) P2P and distributed resource discovery: A layered architecture approach is applied in [12] using hash table data

structure for distributed resource discovery with aim of 3 features: range query, p2p routing, and multiple attributes indexing. According to the author, most present techniques do not have the capability of multi-attribute and range queries. A distributed resource discovery mechanisms is discussed.

This solution support large number of heterogeneous devices. This technique is applied for discovery and registration of resources. Author also implements the technique and verified with sampling data and then evaluate on time and response base. An automated service and resource discovery mechanisms are discussed in [13]. This solution is fully automatic and not need to human intervention for configuration.

2) Centralized architecture for resource discovery: Jara et al provided a technique for the global resources discovery of devices and sensors across numerous conditions. The technique is called discovery advanced which allows sensors to be registered into a centralized registry. Another service oriented framework is suggested based on RESTful [14] API and JSON. The proposed framework integrated with centralized registry that is responsible for every activity. The resources categorized base on domains and the indexing of the resources also based on the domains. This framework also has a problem because resources are only register on base of domain.

3) CoAP-based resource discovery: A service discovery technique is used in [15] by using a RESTfull web server "./wellknown/core" that is responsible for any client request for service discovery [16]. When a client request to the server the server reply with a list of available resources with the attribute that specifies the type of the meta- data of the resource. This may use in numerous areas of a network but have also some limitations such as first-time registration. CoAP cannot implement the first time registration or announced itself the resource [14].

4) Semantic-based resource discovery: In [17] proposed a framework for discover resources with the help of Service advertisement. This research provides a semantic enhanced service proxy framework for Internet of Things to service control, service creation, and service discovery and service invocation mechanism in IoT domain. For this purpose author proposed a framework using SOA [18][19] design an ontology for semantics in the resources and a query based service reroute mechanism, micro-formats for describe the resources and a service advertisement mechanism for easy registration and discovery.

In [20] a technique is purposed to discover resources by using micro formats and micro data for WoT named DiscoWoT. Author user the XML, JSON schema and RDF technology to describe the resources and discover the resources using HTTP protocol for query the resources using GET and POST method and the response of query return by Json.

In [21] proposed a computational method for semantic similarity based on ontology and concepts, which is used to discover the resource according to user requirements. This method calculated the semantic distance between two concepts and length between two concepts.

C. Overview of Search Engines in IoT

The traditional search engine uses the crawlers to discovery and searches the documents on the Internet. The major problem with IoT search engine is the sensors, actuators, and other resources are mostly battery powered which is not available all the time. So the crawlers [22] have not collected the information for the search engine. Another problem with this search engine security, most of the sensors are deployed by the owner of a house, shop, industry etc. so they are not allowed to access the most resources like camera, door lock etc.

Following are the some proposed search engine for IoT from the literature.

1) Keyword-based search engine: A keyword base for physical object search engine [23] "Snoogle" is proposed. As the physical object has the ability to communicate that‘s mean they make IoT. The physical object is like sensors that have some description. The author claimed that this is the first such kind of Information Retrieval system for the physical objects. Secondly, the author compares his results with existing techniques for reduced data transmission overhead. The response system with bloom filter combine with the search engine is the new concept that is included in this research. Thirdly introduce an algorithm which is used for reducing the cost of the query and response time of the client. Fourthly privacy and security is resolved with cryptography is used. Fifth, develop a simulator to simulate the prototype and validate the proposed solution.

2) Location-based search engine: A SenseWeb infrastructure is introduced in [24] in which the private sensors shared their data on a public web. Owners of the sensor upload the data of the Sensor using a GUI on a web. The uploading data only for the private account is not enough if we shared our data with other to use by other applications and human for gain information. With sharing the data help to take actions before an incident as author use example of thunderstorm hit a cab and can automatically share data with other department or other cabs to prepare in advance. Another example of soil sensor for research for students about soil problems and features using the shared data set. SenseWeb is providing a mechanism to shared data among different applications. SenseWeb provides the facility for different applications for shared data but in a uniform way. This architecture is based on Coordinates and Data Transformers.

3) Real-time search engine: In [26] search engine Dyser is proposed that have the ability of scalability of Things and also support rapidly change contents or in another word real-time search engine using two approaches, A) Proactive which called Push approach in which sensor update the index itself and query response by the search engine. B) In Pull approach

when a query is initiated then request sends to the sensor for required data. We know that in future there are more sensors than the queries. So the Pull approach is better solution other the PUSH approach in which a huge amount of data produced at real-time. The main assumption of this research is there are many sensors that produced data periodically. So the sensors and entities are assigned a URL to access. In this research, the language for the query is not specifically used a predefined language like SPARQL [25] or SQL. Implementation of this research is using Java and PHP technologies. Sensor Gateways are implanted using SOAP [26] and Java technologies for fetching the information. Gateways also generate automatically pages for testing purposes using REST interface. This feature minimizes the cost of the reading and indexing of the pages instead of gateways generates its self-page and required only current states of the sensors. After that prediction model is mapped based on states.

4) Hybrid search engine: A hybrid search framework [27] proposed for searching the IoT elements in three different ways. Here this research will offer to solve this problem by combining the three different techniques in a search engine. The user can search the resources using key-word: user can search the resources using keywords, location base: user can search the resources using specific location and state-base: help the user to search resources using state or value based. This system also called real-time searching because of sampling of data from resources continuously updates [28].

5) Things description models: This section describes the details about the different description models for describing of Smart Things. The models which are discussed in this section are SensorML, SensorThings API and W3C JSON-LD.

a) SensorML: SensorML is OCG standard designed for the Sensors and actuators. The main features of this standard are interoperability between devices and also with web nodes. Sensor Web Enablement which is controlled by the OGC. In SWE provided the encoding and interface for Sensor Web which is used by applications and services for accessing sensors. SWE defines the following prototypes language which provides SensorML, O&M, SOS, SPS, and SWE-CDS [29]. The current version of SensorML is 2.0 which is defined by the XML language. This provided structure for the process and processing which is related to measurements and post-measurements of the observation. SensorML supports two types of the Physical objects, Physical System and the Physical Component. A SensorML provides the following descriptions for defining the Meta data and details description of the sensor or system.

Fig. 1. Sensorml Namespaces and Schema Tag.

- definition
- type Of
- configuration,
- features Of Interest
- inputs
- outputs
- parameters
- modes

XML heading or starting tag of SensorML with the following namespaces: SensorML (Sensor Model Language) "xmlns:sml" used to define the physical objects with a model called SensorML [30]. SensorML provides two types of models for defining the complete physical system and single component.

In figure 1 the GML (Geography Markup language) "xmlns:gml" name space is used in SensorML to provides the XML based grammar for defining the location of the Physical Components. SWE (Sensor Web Enablement) "xmlns:swe" provide facilities to developers for defining the physical components e.g. sensors, actuators, transducers.

b) OGC Sensor Things API: The OGC SensorThings API offers the open, geospatial-enabled and unique system to interconnect the Internet of Things (IoT) [31]. SensorThings has the capability of manage sensor data, devices, and applications associated with IoT. SensorThings API implements two main elements and each element is managed by a component. a) Sensing part provides the metadata and the observation of sensors with the capability of heterogeneous devices [32]. The second Tasking part is not implemented yet and this part is used for sensors and actuators parameterizing. SensorThing API used REST web services, JSON language for manipulation and MQTT protocol for communication [33].

Sensing part of this standard also provides the GET, PUT, DELETE and POST methods to create, delete and update the applications and sensors [34]. This is part is developed based on Observation and Measurement model. This part also provides the location of the Sensor or Thing which helps the application to identify the position of the sensor deploy. The following figure 2 displays the Entities of the OGC SensorThing API which is discussed in details.

Fig. 2. OGC SensorThing API Thing Entity.

Each entity manage two types of concepts a) attributes and b) relationship with other the common attributes that are defined the id and self-link of the entity which is used for processing and locate the actual data from the implemented server. Here are the entities which are offered from SensorThing API.

Thing entity: This Entity represents the Thing.

Observed property entity this defines the observation that is covered under a Thing, This entity main feature provides the definition of the observation of the sensor or Thing.

Observation entity describes the details of the observation, DataStream, phenomenon time and feature of interest in for understanding the context of the Thing as shown in figure 2.

Feature of interest defines the context which is required by the user for observation. This entity helps the user to understand the context or observations in a particular place or location.

c) W3C JSON-LD: W3C proposed a language format for interoperability with machines using JSON extends to JSON-LD. Linked Data provides serialization of links to different documents and sites using JSON for interoperability. JSON-LD used JSON based storage because of JSON-LD follows the same format of the JSON.

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you?

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

III. Research Methodology

A. Solr Indexing Tool

For the purpose of the storing sensors description in this system, we use the Solr [29] that is working on the top of Lucene. Solr not only provides the indexing also support for analyzing and searching using RESTfull web service. The documents which are supported by the Solr are PDF, Word documents, Text Files, Rich Text Formats, XML, and JSON. Solr also support multiple databases connectivity and index data from the database.

B. Parser

SensorML uses XML technology and W3C JSON-LD uses JSON with Linked Data technology which is needed to be parsed for obtaining required data.

1) Orchestra XML Parser: Orchestra XML parser [30] version 3.0 is used which is an open source parser specially developed using PHP for web service. This Parser used for parsing the SensorML document which is complex as compared to the simple XML file.

SensorML used not only nested tags but also attribute so there is need a proper parser for avoiding the loss of the data which is complex to handle by the simple document reader.

2) JSON-LD Parser: Lanthaler [31] developed JSON-LD parser which is an open source parser that is officially used by the JSON-LD. This parser provides Expand, Compact, Frame, Flattern, RDF and String formats. By using this parser we use the Expand method to collect sensor description form the JSON-LD format. The version of Lanthaler JSON-LD 1.2 is used in this system.

C. APIs

In this system, we use two API‘s which are Google Maps and Solarium which is providing connectivity between Solr server and Web service.

1) Google Maps: Google Maps [32] is a web service which provides maps of the world, street view, 360degree view of the globe and many other services. In this system, we used the Google Maps to fetch the locations using coordinates which are offered by the sensor. Location name provides also an easy search for searching sensors for a particular location.

2) Solarium: Solarium [33] is a client library for providing services to PHP frameworks by connected with Solr. Solarium provides setting parameters, Modifying query or expanded a query, create indexing documents, Provide CURD operation, building strings, hiding all this with an easy to use API, which is actually your business logic.

D. Web Framework

Laravel Framework [34] is an open source MVC framework which provides integration of API‘s and also a RESTful Web service. The version of Laravel 5.4 is used in this system. Laravel not just providing web framework also provide such as authentication, routing, sessions, and caching.

IV. Implementation

A conceptual model of the proposed architecture is provided. The proposed architecture is implemented using several different software/hardware technologies. This chapter describes a prototype implementation of the proposed layered architecture using APACHE, JAVA, XML, JSON, PHP and MVC framework.

A. Proposed Architecture

In order to complete the proposed solution the layered approach is used to Index and Search the Smart Things in the IoT using a search engine technique.

Figure 3 shows the proposed architecture which support indexing and searching large number of Smart Things, this system provides the following functions.

- Indexing Smart Things: The description of the Smart Things can be indexed using the WEB and RESTful API interface.
- Parsing Description Model: This system automatically detect sensor model language and select parser to collect information.

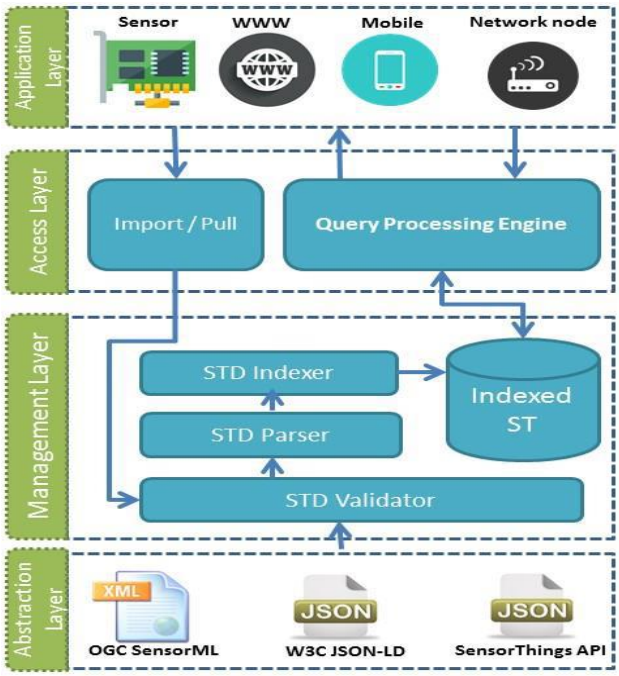

Fig. 3. Purposed Architecture.

- Analyzing Data: This system also analyze the data from the parser and convert it to the proper format under a common schema.
- Collect Structured Information: When the model is parsed this system collected selected fields which are required by the system for searching and indexing.
- Convert coordinates to location: Smart Things are deployed on different locations which are support to Geo-Spatial search. The location of sensors are in GPS coordinates latitude and longitude. The coordinates are needed to further process to fetch name of location.
- Creating Indexes: After collecting and processing the required data this system creates indexes and store them to the indexer.
- Searching Smart Things: The proposed solution provides WEB interface and RESTful API for searching Smart Things with HTML, JSON and XML format.

The proposed architecture consists on four layers Abstraction layer, Management layer, Access layer and Application layer.

1) Abstraction layer: This layer covers the sensors description that supports different models of sensors and Things. Different sensors descriptions models has different capabilities to describe the sensors. The different description technologies which are used in this research are OGC sensorML, OGC sensorThings API, and W3C JSON-LD.

2) Management layer: In this layer, the description of the sensors parses and index to indexer. First we check "Sensor Description" against the schema of the description models. If Description passes the Schema Model then the required and common attributes of the sensor mapped to our required schema attributes. Using this technique the Sensors Description is now in a common format that is matched our model next step is to store the Sensor Description to the indexer. Because of different description models converted into a common format so there is no need to indexing the Sensor Description separately instead of single indexer.

3) Access layer: This layer supports the connectivity of the Application Layer and the Parsing and Storage Layer. Query Parser, Optimizer and Things.

4) Application layer: This layer provides the interface to the user for registering their Things and also querying to search engine. This layer support multiple interfaces for web, sensor, machine, and mobile also support open API for query and register things using REST API Service with different methods.

B. Abstract Languages and Model

Description languages of the Smart Things which are used as a standard, have the variety of sections, fields, and attributes. Description models provide identification as well as observation of Smart Things but our system not required all the details from the model but only a few important fields and attributes that are common in all the description models. Follow table describes the fields and attributes with datatype which are required from all implemented Description language models.

1) OGC SensorML: SensorML describe the Smart Things using XML language.

ID: Tag <sml:PhysicalComponent gml:id= MY_SENSOR‖> provides the attribute of "gml:id" for ID of the sensor.

Name: Tag <gml:identifier> describe the id of the sensor.

TABLE I. COMMON REQUIRED FIELDS FOR SEARCH ENGINE

#	Fields	Type	Example
1	id	string	Davis7817
2	name	string	urn:davisweather:7817
3	description	string	a simplet thermistor sensor
4	Coordinates	double	47.8 , 88.56
5	uri	url	http://www.opengis.net/def/crs/EPSG/0/4326
6	type	string	weather station

Description: SensorML use <gml:description> to provide the short description of sensor. Coordinates: Tag <gml:coordinates>47.8 88.56</gml:coordinates> in

<sml:position> provides the latitude and longitude for the sensor.

Uri: Document itself.

Type: Tag <swe:elementType name=temprature> in <swe:DataStream> provide the type of the sensor.

2) OGC sensorthings API

```
{
   "Thing":{
      "@iot.id": 252560,
      "@iot.selfLink": "http://sensors.example.com/v1.0/Things(252560),
            "name": "Humidity Monitoring In a Mall",
            "description": "Collected the Humidity values of different
     shops in a mall"
   }
   "Location": {
      "location": {
         "type": "Point",
                  "coordinates": [30.133, 49.08]}}
```

ID: in Thing, object "@iot.id" provides the id of the thing.

Name: The name attribute also provided by the Thing object with name attribute. Description: Thing attribute also contains the description attribute for the short description of the sensor.

Coordinates: Location object describes the location object with the array of latitude and longitude under the coordinates array.

Uri: Under the Thing object the attribute @iot.selflink provides the URI of the thing. Type: The type of the Thing is described under the object of DataStream with name attribute. We tokenized the name to get the original type of the sensor.

3) W3C JSON-L

```
{ "iot": "https://iotdb.org/pub/iot#",
"iot-attribute": "https://iotdb.org/pub/iot-attribute#",
"iot-unit": "https://iotdb.org/pub/iot-unit#",
"vocab": "http://www.example.org/vocab#",
"sensorID": "iot:uuid",
"sensorName": "iot:name",
"sensorDescription": "iot:description",
"sensorValue": {
      "@id": "iot-attribute:sensor.chemical",
"sensorUnit": "iot:unit",
"timestamp": "iot:datetime",
"sensorType": "iot:type",
"sensorUrl": "iot:uri",
"sensorGeo": "http://schema.org/geo",
"latitude": {
      "@id": "http://schema.org/latitude",
```

ID: sensorID describe the ID of the sensor as "https://iotdb.org/pub/iot#uuid": 5484 Name: "sensorName" describe the name of the sensor as "https://iotdb.org/pub/iot#name":

"Flam Sensor with Servo Motor" Type: sensorType describe the Type of the sensor as "https://iotdb.org/pub/iot#type":

"Flam"

URI: sensorUrl describe the URI of the sensor as "https://iotdb.org/pub/iot#uri": http://www.sensors.

Coordinates: http://schema.org/geo": latitude and longitude attributes defines.

C. Management Layer

In this Layer there is discussion about how an Description Model is dispatched with the Schema and pass the Schema for successful parsing.

1) Things modeling languages validator: A schema is a language for describing the information about the tags and attributes which are used in the XML or other languages. There are different schema languages for XML like DTD and XSD for defining the schema of the XML. The XML its self not enable to provide meanings so we need to use the schema for defining the meanings to the tags and attributes, associate the attributes or tags to data types, control the appearance of the tags or positions of the tags and attributes, provide documentation for machine readable and for human readable and proving the formal definition to one or more documents.

As like XML schema validator we have JSON schema validator also which validate the JSON against JSON Schema. Before processing further to parse or obtain data from the file or from the server we need to verify the attributes against the schema. JSON-LD also has the capability of multiple and complex data types which are described under the @context object to validate each attribute with data types.

Schema validator also helps us to verify the uniqueness of the data from different resources. This mechanism helps to produce a compatible system with all other IoT resources which are wishing to connect with our system. If the validator passes the data then process further otherwise discard the processing and throw an exception of error a code for response to the requester.

2) Smart things description language parser: This module parses the Things modeling; Languages after validating the schema of the related Model. When the schema validates the document of Things then the next procedure is started which is parsing. The main purpose of this module is parsing. In simple this module collected the related information from the document which is required. This module collects all the attributes step by step procedure which is defined by internal logic. We need three different model parser because of we implanted three different Molding Languages.

- SensorML used XML language to describe the sensor meta-data and description. Our proposed work works on WEB, so that we use a PHP language to parse the

model. XMLReader is used to read the document. Because of XML has the capability of nesting tags. So we have a lot of processing to collect required attributes. As mention early in Background Study chapter under the section of SensorML, The XML document has multiple namespaces so we need to examine all the namespaces for collected related data, not unrelated data.

- SensorThings API used JSON for describing the Sensors and Things. This Modeling Language has also other related objects like observations, locations, data streams etc. We need only required attributes which are in the entities of Things, Sensors and Location. So we need recursive processing to retrieve the all required attributes because of the entities have different URL's.
- W3C JSON-LD is based on JSON language for describing the Smart Things. All required attributes are defined in same file. After collecting the required attributes from these description models the next step is to convert into same schema called Thing Schema for removing and refining the attributes and for further processing.

3) Smart things indexer: In this module, the collected attributes are mapped to the internal schema and refined. The refinement of the collected attributes is necessary because of the use of the different Modeling languages. In this module, the collected attributes are formatted in common data-types because of the internal indexing system. If the format system is not used then we have another problem which is the heterogeneity of the Modeling Languages. To solve the heterogeneity problem there is need of different indexer and when we query the data we need to combine the indexer on the query. So one problem creates another problem to solve this problem the best way is to convert the data types into same data types without losing the data. One attribute which is URI is used for getting original data from the sensor.

Another requirement for location-based query needs because of human search with an area. But in the Modeling Language cannot provide the location names but only coordinates latitude and longitude. We have another scenario which is query base on location. To solve this issue we have Google Map API which is helping us to convert coordinates to location. Google API returns the Country, State, City, Postal code and street address. This information is enough for searching sensors based on specific locations. The attributes which are required from modeling language are mentioned in 4.1 tables. Required processing done thing information sends to indexer to store.

4) Indexing descriptions of smart things: This module is the major part of the search engine which holds meta-data of the Things. For index meta-data, we use lucene Solr which provides RESTfull API for selecting, updating and inserting structure data. Solr provide the customized schema to store structure documents. For meeting the system requirements, we customize the schema.

Solr work as a cloud service and use JAVA language. So we need an API to interact with Solr because of this research deployed on HTTP server. A plug-in named Solarium PHP is used for connectivity with the Solr and with our server.

Solarium PHP provides connectivity and also the mechanism to query and updating the index, customized schema and other facilities. Following are the steps to index the sensor.

- After parsing the sensor to the schema, the converted schema transforms as Solr document (Thing) for storing.
- Initialized a client and open connection with Solr using Solarium PHP API and send the data to the Sorl.
- Solr check income data with the ID. If the ID already exists then update the data and if ID does not exist then new document tokenized and store in Solr Indexer.
- After updating the Things Solr response back to the client with Code and Query Time.
- If the Thing is not updating within a given time period a service delete the document from the indexer.

Solr has the capability of handle multiple clients and queries at a time by distributed mechanism.

D. Access layer

This layer provides query, optimization, results, ranking, importing and pulling things from servers and our system. This layer also helps us to connect with the user interface using HTTP requests.

1) Importing things: In this module importing mechanism is implemented in which user can import things and sensors using Modeling Languages which are used in our system. Using RESTful Web Service ./import/ we can import Things and Sensors.

- SensorML ./import/sensorML/ used to import sensorML document which is written in XML language.
- SensorThings ./import/sensorThings/ used to import sensorThing document which is written in JSON language.
- W3C JSON-LD ./import/W3CJson/ used to import W3C statandard JSON-LD document which is written in JSON-LD language. These documents are sent to validator Module in Parsing and Storage Layer. After completing the Parsing process the document store in indexer and response back to the user using back chain process. After successfully storing the document to indexer the document its self "file" is deleted from the server.

2) Pulling smart things from sandboxes: In this module, we can pull Sensors or Things from Sandboxes. SensorThings API implemented by other service providers so we have the capability of pulling data from that service provider which publicly available. E.g sensorup.com implements the

sensorThing server and also provides the open access to the server for working with the sensorThing server. We need only the URL of Things our system provides the facility to pull all data from the server in the recursive fashion.

Here is the implemented service http://toronto-bike-snapshot.sensorup.com/v1. 0/Things by using this URL in our system all Things that are available for public fetched and other attributes like "Location" are also fetched by using internal logic.

3) Query processing engine: Query parser accepts the query from the application layer and sends it to Optimizer for optimization for query for better results. Query parser following the step to complete the process these are following:

- The Application Layer has multiple techniques implemented so there is need to identify the query first then send to query parser.
- All application use HTTP protocol so we can easily understand the query because of defining a RESTful service.
- All search requests are created using GET method and send to /search/.
- Our system supports two types of queries keyword and Geo-Spatial.
- When query receives it evaluate the types of the query and dispatch the query to the optimizer to optimize the query for better results.

Here are the implanted query types in this system. This search engine support two types of query a) Keyword Based Query and b) Geo-Spatial Based Query.

4) Keyword query: Keyword Query support full text based query for search Things. The user can query with the keyword and this system fully supports all the attributes for the specific keyword. RESTful URL "./search/keyword" accepted the request of the keyword query. "Keyword" is the text which is requested to search. The user initiated the query using HTTP protocol. If the keyword based on more than one word then the words tokenized based on space and make the search for each token.

5) Location/geo-spatial query: The second approach is Geo-Spatial Query which is a complex query. The user can search for thing using a specific location and also within a range of the specific location. Geospatial Query needs three attributes latitude, longitude, and distance.

- "Latitude" which define the angular distance of the earth from north or South Pole.
- "Longitude" defines the distance of the earth from the east and west.
- "Distance Attribute" defines the distance from latitude and longitude in KMs.
- "Format" attribute defines the results format current supported "JSON and XML". So the user replaces format with JSON or XML.

Using all these three attributes query for Geo-Spatial Query use the RESTful URL as

/search/lat/?/long/?/dist/?/format. lat defines the latitude and first "?" is latitude value, long defines longitude and second "?" is the value of the longitude and finally, dist. defines distance in km third "?" is the value of the distance. If the user wants distance in miles then use dist.*0.621371.

6) Query and results optimizer: This module helps to optimize the query and search results and control data between Indexer in Processing and Storage Layer and with Core Layer.

7) Query optimizer: The raw query may use the more resources that not suitable for the huge amount of data. so there must be a mechanism to sort out this problem. We need an optimizer minimized the cost of the resources because of in future the data is generated more and more that cause the system speed down. The query optimizer also supports for multiple indexers. Optimizer select different query planner for the same query by the internal logic and select best query to minimize the overhead and resources.

8) Results optimizer: While the results against the query are retrieved from the Indexer there is need to optimize the results. All the results may not require by the user there is need a ranked results base on frequent results or most search Things. So the optimizer sorts out the results and rank based on the score of the Things. After ranking and sorting the results send to Search Results module.

9) Search results: Optimizer sends the results to this module. This module first converts the index documents which is return against the query to the required format. As mentioned earlier in the Query Parser format tells the system to in which language output is generated. The following attributes are defined in each of the formats. Id: defines the unique id for identifying the sensor. Name: defines the name of the according to Modeling Languages. Description: provides the short description of the sensor. lat-long: provide the latitude and longitude in "31.56, 65.66" format separated by ",". Type: defines the sensor type e.g "temperature, motion, vibration" etc.

E. Application Layer

Currently, Application Layer support only HTTP protocol for displaying results and query interfaces using HTML and CSS technologies to interact with the system. The Layer also provides features to another system to directly interact with the system by using open REST API with an authentication token. WEB interface used to directly interact with the system by the human. Other network devices also interact with the system using RESTful Web Services. This layer also provides the service to other systems to creating, updating, querying and resulting with the system. This layer also supports multiple languages responses back to the requester. This system helps the user to search and store their sensors and things using multiple Modeling Language in the web browser.

Commonly IoT devices connected to the centralized hub for accessing and storing data. In this system, there is no need of centralized hub system we use the web browser to handle the things. The web browser connects to the cloud using REST APIs so there is no need of getting access to the server or hub.

New Thing and Sensor also added to the system using API which is provided in this system for achieving the desired operation can be solved under the following modules.

1) URL resolver: This module translates the URL and solves the protocols issues. Currently, support HTTP protocol only so the protocol parameter set defaults to HTTP. This module also understands the request and map over the correct model for processing further.

2) User interfaces: This module provides the UI for interaction with the system using HTML, CSS, and Javascript language. This module works under two different conditions one for guest and one for the system user.

a) Guest: This UI for available for the public for searching the sensors, get the description in different formats. The formats that are used fo display HTML and CSS used, the user also has the ability to get the description in XML and JSON format.

*b) System User***:** This user need the login to the system if he uses the WEB, otherwise the user can also send the request to the system using authentication token. Authentication token provides to the user after registration with the system. System user has also the ability to register and update Thing directly from any application or language using POST and PUT method.

3) Inserting thing: System user has the ability to insert the Things using the authentication token. For inserting new Things with the system user can use POST method to the REST URL /thing/insert. Parameters that are required to create the new Thing are defined in Table 4.1. An extra parameter that is required for authenticating the request for security purpose is "Token". "Token" is string type and replaces the value of authentication token which provides the system on registration. When request successfully accepted a response will get in JSON format with the id of the Thing. This id uses next time to update the Thing.

4) Updating thing: System user has also the ability to update the Things after creating the Thing by using the authentication token. For updating the Things with the system user can use PUT method to the REST URL http://www.example.com/thing/update/id. Parameters that are required to update the Thing are defined in Table 4.1 table. An extra parameter that is required for authenticating the request for security purpose is "Token". "Token" is string type and replaces the value of authentication token which provides the system on registration.

5) Query interface: The UI also helps to query the system with Query Interface. Two types of queries are supported currently keyword and Geo-Spatial. The user can switch from keyword to Geospatial based query by selecting radio button on the page of Search Things.

6) Result interface: After the query to search engine results are returned from the system which is displayed on the results page on the system.

V. Results and Discussion

Few experiments were conducted to validate the Indexing and Searching of the prototype. Indexing technique implemented using Solr by providing the WEB interface and also searching using the same interface. A REST API is also tested and validated by Indexing and Searching of Smart Things for machines. The proposed solution also support for indexing Smart Things from different sandboxes.

A. Results

This research helps the sensors description model to parse, index and searched. Following are the results which are performed in this system.

1) Indexing: Figure 4 provide the GUI to index the description of the Smart Things. This interface has the ability to select the variety of description models that are selected more than one at a time.

The selected models are parsed using different parsers and then converting into the shared schema that is shown in the figure 5 the common fields that are same in models are fetched and converting the fields into same data types.

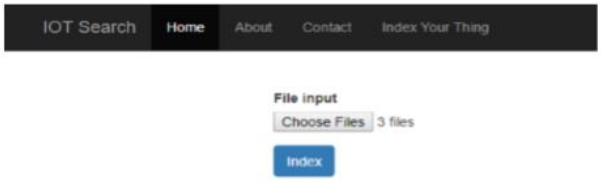

Fig. 4. Indexing Smart Things by Uploading Descriptions.

Fig. 5. Smart Things Description Parsed into Shared Schema.

```
[
  {
    "id":"sensorThing:3971",
    "description":"Ita nemo beato beatior....",
    "name":"Flam Sensor with Servo Motor",
    "latlong":"30.828,72.038",
    "location":"address:Gharh Mahraja Road,city:Shorkot,district:Jhang District,state:Punjab,country:Pakistan'
    "uri":"http://toronto-bike-snapshot.sensorup.com/v1.0/Things(3971)",
    "type":"Flam",
    "_version_":1582571829369765888
  },
  {
    "id":"urn:tr:21449",
    "description":"Primum quid tu dicis breve...",
    "name":"IR Sensor on High way raod",
    "latlong":"30.431,71.744",
    "location":"address:Unnamed Road,district:Khanewal District,state:Punjab,country:Pakistan",
    "uri":"http://www.opengis.net/def/crs/21449",
    "type":"IR",
    "_version_":1582571829444214784
  },
  {
    "id":"JsonLD:30474",
    "description":"Ita nemo beato beatior.....",
    "name":"Gas Sensor with available actuators",
    "latlong":"31.141,71.261",
    "location":"address:Unnamed Road,district:Layyah District,state:Punjab,country:Pakistan",
    "uri":"http://www.sensors.org/sensor/30474",
    "type":"Chemical",
    "_version_":1582571829449457664
  }
]
```


Fig. 6. Smart Things Descriptions Indexed in Solr.

Indexing query executed
Number of Smart Things Index : 30
Query status: 0
Query time: 0.791 seconds

Fig. 7. Smart Things Description Index Response from Solr.

After parsing the models into shared schema the collection of schema convert the model into the Solr document. Solarium provided the connectivity between Solr and web framework.

Using Solarium the documents transferred to Solr from the web figure 6.

When the documents are indexing into the Solr the response of the indexed files with the status of indexing and time is the return which is shown in the figure 7.

The figure 8 compared the response time of indexing the description of the Smart Things. The figure shows that the indexing of description from 5 to 500 with time in seconds. All the description models have the same time to indexing with a little difference.

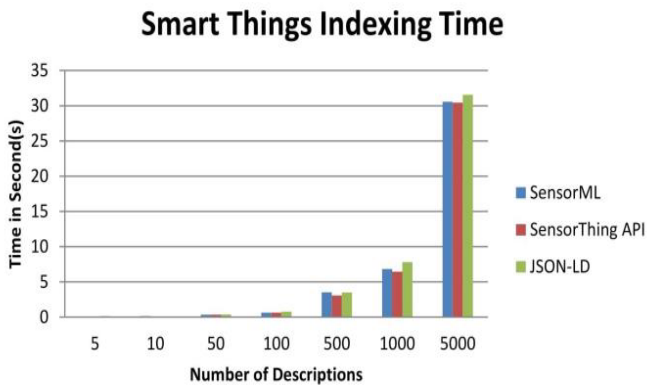

Fig. 8. Response time for Indexing of Descriptions.

2) Searching: After indexing the description of Smart Things the next step is to search. Our system provides two types of the query for searching Smart Things first Keyword-based searching and second one is Geo-Spatial using coordinates of location with range. In figure 8 keyword-based searching interface is shown in which user type the keywords like vibration, temperature, flam, fire etc.

Another keyword technique is implemented using types of the sensor. As shown in figure 9 all sensors which are register with this system dynamically load all the types of the Smart Things with a number of register Things.

Under the keyword-based search, the second functionality which is implemented is the location as a keyword. As shown in figure 10 users can also type the name of the location and search Smart Things which are deployed on given location.

The second approach for searching Smart Things is Geo-Spatial which used coordinated along with latitude and longitude and distance which is depicted in figure 11. The user can search the Smart Things for a specific location with distance kilometers.

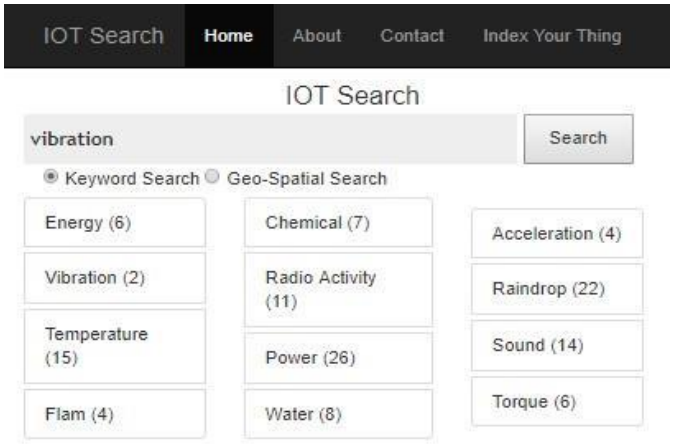

Fig. 9. Keyword based Search WEB Interface.

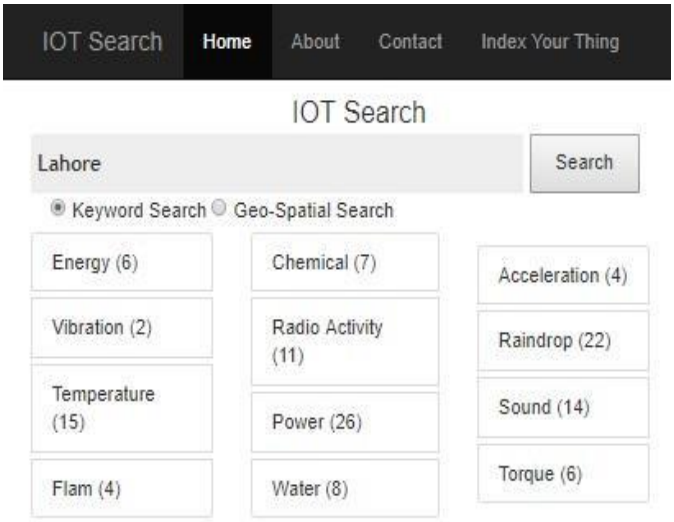

Fig. 10. Searching Smart Things with Name of Location.

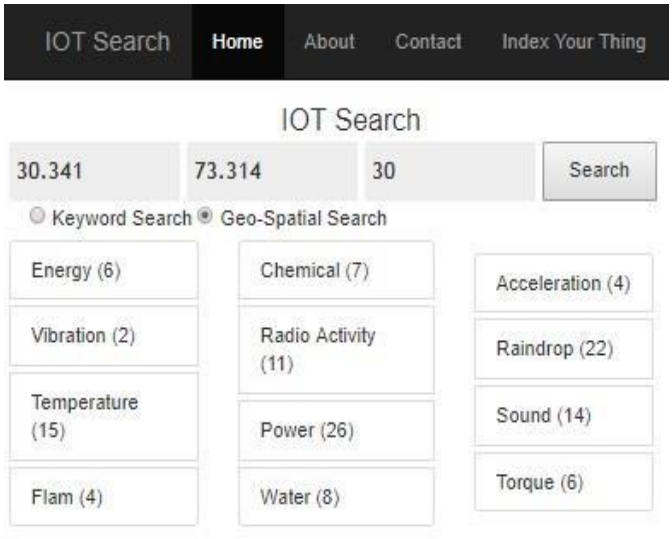

Fig. 11. Geo-Spatial Search of Smart Things with Coordinates.

3) Search results: The results of the search responses by the interface which contain the following fields id, description, name, lat-long, location, URI and type of the sensor as shown in the figure 12.

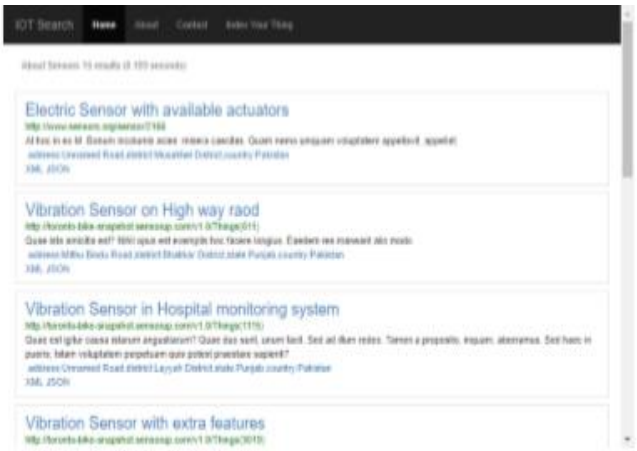

Fig. 12. Search Results for Keyword.

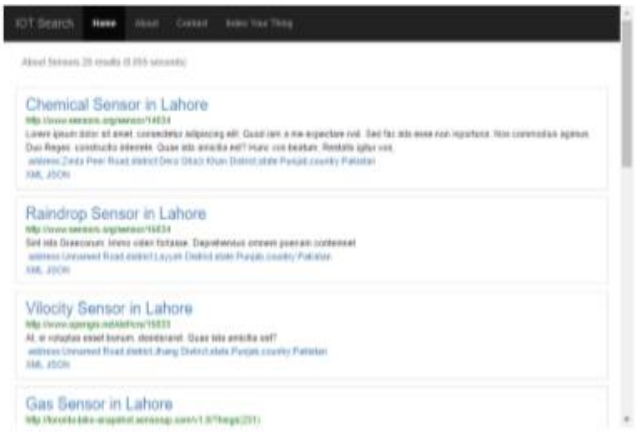

Fig. 13. Search Results for Name of Location.

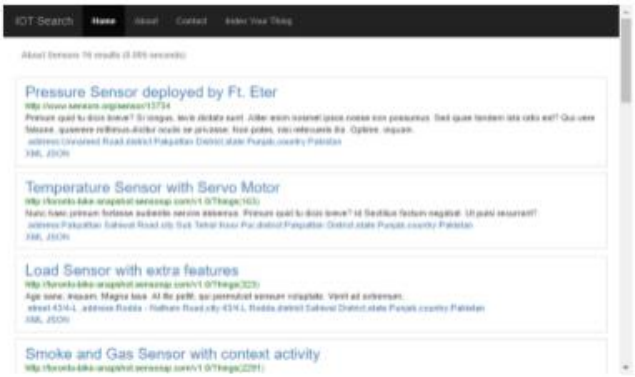

Fig. 14. Search Results for Geo-Spatial Query.


```
{
    "time": 0.002,
    "num": 1,
    "results": [
        {
            "id": "JsonLD:14034",
            "description": "Lorem ipsum dolor sit amet, c
            "name": "Chemical Sensor in Lahore",
            "latlong": "30.441,70.438",
            "location": {
                "address": "Zinda Peer Road",
                "district": "Dera Ghazi Khan District",
                "state": "Punjab",
                "country": "Pakistan"
            },
            "uri": "http://www.sensors.org/sensor/14034",
            "type": "Chemical"
        }
    ]
}
```


Fig. 15. Search Results for Machine with JSON Format.

The figure 13 depicts the results of the location-based search as a keyword. It acts like a keyword search on indexed Smart Things.

The figure 14 showed the results of the Geo-Spatial search approach with a range of distance. Using this technique user can collected data for the specific area.

This system cannot provide only the GUI for the user but also interpretable between machines. The machine can also query and index for the Smart Things over HTTP and using JSON and XML formats. In figure 15 shown the results of the search by machine in JSON format. The machine specifies the format of results at the time of the query.

VI. Conclusion

The searching of physical objects is difficult as compare to traditional searching of documents for web. The traditional systems used database to store and used crawlers to retrieve the meta-data form the websites. On the other hand description of physical objects has lack of standardization to describe itself. Traditional document search engines provides only keyword-based search which is not enough for searching of the physical objects. Databases seem not feasible to handle the exponential growth of documents. The crawlers are facing a challenge for retrieving the physical objects in IoT. These Smart Things increasing significantly and have a variety of descriptions models. These models are using different languages that pose a challenge of fetching useful meta-data required for searching. The area of searching and indexing of

Smart Things in IoT is not fully explored. The variety of the description models is a barrier to searching and indexing of Smart Things. The technique proposed in this research can helps to index and search Smart Things with a variety of description models. The major functionalities provided by the proposed system are retrieving Smart Things descriptions parsing of descriptions, analyzing meta-data, collecting structured data, converting coordinates to locations and develop index of Smart Things. Two interfaces first web interface and second for machine with XML and JSON formats. The developed solution currently supports SensorML, SensorThings API and W3C JSON-LD standards which are used to describing Smart Things. These standards are automatically analyzed by this system with internal logic and indexed them into Indexer.

The current system provides interfaces for the machine and for the human to search Smart Things using two types of queries. First keyword based and second one Geospatial. Keyword-based query not only provides the searching description of the Smart Things but also supports the location as a keyword either city, district, province or country. The second search is Geo-Spatial which helps the user to search Smart Things using coordinates with latitude, longitude, and distance. Our findings showed significant search results that can lead to an innovative search engine for Internet of Things.

VII. Future Work

For future looking for Optimizing search results on most frequent searches, Optimizing indexing process, Develop more rich queries such as values based and real time, Handling mobility of Smart Thins, Extending search results to a variety formats and Solution can be moved to cloud base.

References

[1] Tapscott D, Williams AD. Don Tapscott, Anthony D. Williams n.d.

[2] Botts M, Percivall G, Reed C, Davidson J. OGC (R) Sensor Web Enablement: Overview and High Level Architecture. Lect Notes Comput Sci 2007;4540:175–190. doi:10.1007/978-3-540-79996-2.

[3] Mitton N, Petrolo R, Loscr V, Mitton N. Towards a smart city based on cloud of things , a survey on the smart city vision and paradigms the smart city vision and paradigms 2015. doi:10.1002/ett.

[4] Rose K, Eldridge S, Chapin L. THE INTERNET OF THINGS: AN OVERVIEW. Understanding the Issues and Challenges of a More Connected World. Internet Soc 2015:80. doi:10.5480/1536-5026-34.1.63.

[5] Gope P, Amin R, Hafizul Islam SK, Kumar N, Bhalla VK. Lightweight and privacy-preserving RFID authentication scheme for distributed IoT infrastructure with secure localization services for smart city environment. Futur Gener Comput Syst 2018;83:629–37. doi:10.1016/j.future.2017.06.023.

[6] Kilper DC, Bergman K, Chan VWS, Monga I, Porter G, Rauschenbach K. Optical Networks come of Age. Opt Photonics News 2014:51–7. doi:10.1364/OPN.25.9.000050.

[7] Riggins FJ, Wamba SF. Research Directions on the Adoption, Usage, and Impact of the Internet of Things through the Use of Big Data Analytics. Syst Sci (HICSS), 2015 48th Hawaii Int Conf 2015:1531–40.

[8] Friess P, Vermesan O. Building the Hyperconnected Society. 2015. doi:978-87-93237-99-5.

[9] Guinard D, Trifa V, Karnouskos S, Spiess P, Savio D. Interacting with the SOA-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. IEEE Trans Serv Comput 2010;3:223–35. doi:10.1109/TSC.2010.3.

[10] Arnold AS, Wilson JS, Boshier MG. Arnold , A S and Wilson , J S and Boshier , M G (1998) A simple extended- cavity diode laser . Review of Scientific Instruments , 69 (3). pp . 1236- This version is available at https://strathprints.strath.ac.uk/33635/ A simple extended-cavity diode l 1998;69:1236–9.

[11] Paganelli F, Parlanti D. A DHT-Based Discovery Service for the Internet of Things. J Comput Networks Commun 2012;2012:1–11. doi:10.1155/2012/107041.

[12] Tuna G, Kogias DG, Gungor VC, Gezer C, Taşkın E, Ayday E. A survey on information security threats and solutions for Machine to Machine (M2M) communications. J Parallel Distrib Comput 2017;109:142–54. doi:10.1016/j.jpdc.2017.05.021.

[13] Alghamdi TA, Lasebae A, Aiash M. Security Analysis of the Constrained Application Protocol in the Internet of Things Security Analysis of the Constrained Application Protocol in the Internet of Things. Futur Gener Commun Technol (FGCT), 2013 Second Int Conf 2013:163–8. doi:10.1109/FGCT.2013.6767217.

[14] Li S, Xu L Da, Zhao S. 5G Internet of Things: A survey. J Ind Inf Integr 2018;10:1–9. doi:10.1016/j.jii.2018.01.005.

[15] Jara AJ, Lopez P, Fernandez D, Castillo JF, Zamora MA, Skarmeta AF. Mobile Digcovery: A Global Service Discovery for the Internet of Things. 2013 27th Int. Conf. Adv. Inf. Netw. Appl. Work., IEEE; 2013, p. 1325–30. doi:10.1109/WAINA.2013.261.

[16] Fielding RT, Taylor RN, Erenkrantz JR, Gorlick MM, Whitehead J, Khare R, et al. Reflections on the REST architectural style and "principled design of the modern web architecture" (impact paper award). Proc 2017 11th Jt Meet Found Softw Eng - ESEC/FSE 2017 2017:4–14. doi:10.1145/3106237.3121282.

[17] Wollschlaeger M, Sauter T, Jasperneite J. The future of industrial communication. IEEE Ind Electron Mag 2017;11:17–27. doi:10.1021/ie50124a022.

[18] Soylu A, Giese M, Jimenez-Ruiz E, Vega-Gorgojo G, Horrocks I. Experiencing OptiqueVQS: a multi-paradigm and ontology-based visual query system for end users. Univers Access Inf Soc 2016;15:129–52. doi:10.1007/s10209-015-0404-5.

[19] Ejaz W, Naeem M, Shahid A, Anpalagan A, Jo M. Efficient Energy Management for Internet of Things in Smart Cities. IEEE Commun Mag 2017:84–91. doi:10.1109/MCOM.2017.1600218CM.

[20] Zhou M, Ma Y. A web service discovery computational method for IOT system. 2012 IEEE 2nd Int. Conf. Cloud Comput. Intell. Syst., IEEE; 2012, p. 1009–12. doi:10.1109/CCIS.2012.6664533.

[21] Durumeric Z, Adrian D, Mirian A, Bailey M, Halderman JA. A Search Engine Backed by Internet-Wide Scanning. Proc 22nd ACM SIGSAC Conf Comput Commun Secur - CCS '15 2015:542–53. doi:10.1145/2810103.2813703.

[22] Yi S, Li C, Li Q. A Survey of Fog Computing: Concepts, Applications and Issues. Proc 2015 Work Mob Big Data - Mobidata '15 2015:37–42. doi:10.1145/2757384.2757397.

[23] Guo B, Wang Z, Yu Z, Wang Y, Yen NY, Huang R, et al. Mobile Crowd Sensing and Computing: The Review of an Emerging Human-Powered Sensing Paradigm. ACM Comput Surv 2015;48:1–31. doi:10.1145/2794400.

[24] Nitti M, Pilloni V, Colistra G, Atzori L. The Virtual Object as a Major Element of the Internet of Things: A Survey. IEEE Commun Surv Tutorials 2016;18:1228–40. doi:10.1109/COMST.2015.2498304.

[25] Khan MW, Abbasi E. Differentiating Parameters for Selecting Simple Object Access Protocol (SOAP) vs . Representational State Transfer (REST) Based Architecture. J Adv Comput Networks 2015;3. doi:10.7763/JACN.2015.V3.143.

[26] Datta SK, Da Costa RPF, Bonnet C. Resource discovery in Internet of Things: Current trends and future standardization aspects. IEEE World Forum Internet Things, WF-IoT 2015 - Proc 2015:542–7. doi:10.1109/WF-IoT.2015.7389112.

[27] Pydipaty R, Saha A. On Using Non-Volatile Memory in Apache Lucene n.d.

[28] Chang B-R, Tsai H-F, Hsu H-T. Secondary index to Big Data NoSQL Database – Incorporating solr to HBase approach. J Inf Hiding Multimed Signal Process 2016;7:80–9.

[29] Yu HR. Design and implementation of web based on Laravel framework. Atl Press 2015:301–4. doi:10.2991/iccset-14.2015.66.

[30] Sporny M, Bazaar D, Kellogg G, Associates K, Lanthaler M, Sporny M, et al. A JSON-based Serialization for Linked Data 2018.

[31] Kingston R. Public Participation in Geocomputation to Support Spatial Decision-Making. Geocomputation A Pract Prim 2015:301–19.

[32] Smiley D, Pugh E. Apache Solr 3 Enterprise Search Server. Search 2009:7–10.

[33] Laravel: Up and Running - O'Reilly Media n.d. http://shop.oreilly.com/product/0636920044116.do (accessed June 14, 2018).

[34] Medina - Santiago A, Cisneros - Gómez A, Melgar - Paniagua EM, Gutierrez T, B Nango - Sólis MG, Moreno - López EA, et al. Web Application Development by Applying the MVC and Table Data Gateway in the Annual Program Budget Management System. IJACSA) Int J Adv Comput Sci Appl 2017;8:250–5. doi:10.14569/IJACSA.2017.080233.