Ghent University

Internship Digipolis Ghent

Solid Pods for IoT

Flor Sanders

Under the guidance of
Hans Fraiponts
Pieter-Jan Pauwels
prof. Jeroen Hoebeke

2020-2021

Contents

Chapter 1: The Company

Digipolis[1] was founded in 2003 as an intercommunal collaboration between the cities of Ghent and Antwerp and offers ICT services and support to both cities. Their goal is to improve the cities for all their inhabitants as a place to live and work by making use of technology.

Besides offering ICT support for daily operations to the personnel of all city departments, Digipolis Ghent also occupies itself with the exploration of more advanced technologies which have the potential of improving life and work in the city. Examples of such projects are:

- Smart Cities + Open Data Reuse (SCORE)[2]
- Traffic Management as a Service (TMaaS)[3]
- City of Things (CoT) databroker[4]

From 2021 onwards, Digipolis will split as a company and the Ghent and Antwerp branches will go their own way. Digipolis Ghent, where this internship took place, will continue as an autonomous communal company (nl: autonoom gemeentebedrijf) under the name of District09[5].

Figure 1.1 shows the structure under which Digipolis Ghent operates as a company. Each of the operational branches shown in the diagram is further subdivided into operational units or cells. My internship was guided by Hans Fraiponts from the solutions group in the coordination branch as well as Pieter-Jan Pauwels from the strategy and foresight team in the management branch.

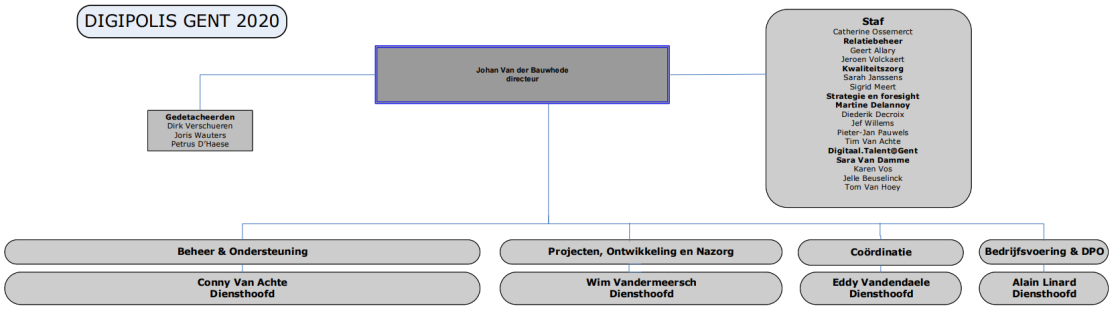


Figure 1.1: Organogram of Digipolis Ghent [nl]

[1] https://www.digipolis.be/
[2] https://northsearegion.eu/score/
[3] https://drive.tmaas.eu/
[4] https://stad.gent/sites/default/files/media/documents/Vlaio%20City%20of%20Things%20project%20DataBroker%20eindrapport_versie_12032020.pdf
[5] https://stad.gent/nl/smart-city/nieuws-evenementen/stad-gent-zet-digitale-versnelling-met-district09

Chapter 2: Internship Assignment

Solid[1], abbreviated from social linked data, is an upcoming technology that strives to change the way data is used on the web. The goal of the project is to give control of the personal data produced by using web applications back to the user. Such an evolution might have big implications for how people interact with the web and is thus a topic of interest at Digipolis.

For this internship, I was asked to help the company explore this new technology by continuing the work started during my bachelor's thesis: research how Solid might be used in IoT applications. After considering different possibilities, we decided to continue this research by building a second proof-of-concept (PoC) implementation with the goal of learning the mechanisms of saving, sharing and publishing sensor data making use of Solid Pods.

Additionally, I was instructed to investigate Solid in the current context of data protection laws and open data charters and envision some interesting use cases through which Solid could improve how the city and its citizens handle IoT data.

Aside from this report, the deliverables produced for this internship contain:

- the open-sourced code base along with documentation for installation and usage[2].
- a video with a practical demonstration of the PoC[3].
- a final presentation of the project performed for the people at Digipolis, the city of Ghent and Ghent University.

The internship took place during the six weeks between August third and September eleventh of 2020. Due to the ongoing corona pandemic, it was only possible to work at the office part time with the rest of the work performed from home.

[1] `https://solidproject.org/`
[2] `https://github.com/lab9k/Solid-Pods-For-IoT`
[3] `https://florsanders.be/vid/Solid_Pods_for_IoT_Demonstration.mp4`

Chapter 3: Technical Report

3.1 Introduction

Solid is a project founded in 2016 under the guidance of Tim Berners-Lee, the inventor of the world wide web. The technology produced in the Solid project strives to enable users to read, write and share data on the web without being dependent on closed platforms to do so. The core idea of Solid is that an individual or organisation would have a personal online data store (Pod) where personal data can be saved under the user's control, though without losing the advantages offered by the web.

The Internet of Things (IoT) describes a system where devices can communicate with each other over a network without the need for human intervention. This field has known tremendous growth since the coinage of the term by Kevin Ashton in 1999. It is generally agreed however that the truly interesting applications envisioned for the technology, such as a City of Things (CoT) or the Industrial Internet of Things (IIOT), are being held back by the fragmentation that dominates the space.

The solution for this fragmentation is being worked on from two sides. On the one hand there are the bottom-up initiatives to uniformise communication methods between devices across the industry by leveraging open standards. An example of this is the recent Connected Home over IP project[1] promoted by Amazon, Apple, Google and the Zigbee Alliance.

On the other hand there is the top-down approach taken by the semantic web community in the form of the Web Of Things (WoT)[2] and the Web of Data (WoD)[3] which proposes to use concepts from linked data (LD) and the Resource Description Framework (RDF) for the description of IoT data and systems to counter the existing fragmentation.

The work done during this internship at Digipolis Ghent builds upon my bachelor's thesis at Ghent University in which sensor data was extracted from an IoT network using the Lightweight Machine to Machine (LwM2M) protocol and saved to a Solid Pod [1]. Here we will investigate whether Solid could offer a midpoint where the two movements mentioned above could meet, which is necessary in order to make a uniform solution for interoperable IoT applications possible.

[1] `https://www.connectedhomeip.com/`
[2] `https://www.w3.org/WoT/`
[3] `https://www.w3.org/2013/data/`

3.2 Motivation

In the introduction (3.1) two movements are given that strive to solve the problem of fragmentation in the field of IoT, working from different sides. It is claimed that Solid might be a good candidate for unifying these two strategies. In this section the reasons for this claim are given.

The recent movement towards open data sparked by charters at communal[4] as well as international level[5] have caused governments and companies alike to adopt linked data and RDF as standards for publishing their data.

While semantic web technologies have long been a niche due to the complexity of the field, this trend is changing thanks to the increase in popularity mentioned above. The authors in [2] identify the application of machine learning algorithms to linked data as an emerging area of research. In short, the future of linked data looks promising.

In previous works, one strategy of incorporating IoT devices in a linked data Platform (LDP) is to run this platform right on top of them [3][4]. As concluded in [5] however, this strategy doesn't take the limited resources and constrained links into account, which are fundamental properties of wireless IoT devices. The direct implementation of a LDP onto these devices would have drastic consequences for their battery life.

Another strategy is to stick to the communication standards designed with these constrained links in mind and make the conversion to RDF at a later stage. Such conversions have been studied in literature already, e.g. [6] proposes a conversion from Sensor Measurement List (SenML) format to RDF. Usually though, this conversion is performed on large data sets as a final step before publication, rather than on a real-time stream of incoming data [7].

The premise of Solid is to offer a personal data store with full LDP capabilities, where the user stays in control. Using the conversion mechanisms mentioned above, the data produced by constrained wireless devices could be saved to a Solid Pod in real time. These data could then be processed by different applications, shared with third parties or even published publicly through the Solid platform.

The strategy proposed here offers the advantage of using the principles of linked data in IoT applications close to the data source, while not having to cannibalise the constrained nature of the devices used in the process.

In conclusion, the use of Solid in IoT applications strikes a good balance between the advantages and disadvantages that come with the two movements, implementing a semantic Web of Things and uniformising the communication standards across the industry, which work to alleviate the problem of fragmentation in IoT.

[4] `https://smart.flanders.be/open-data-charter/`
[5] `https://opendatacharter.net/`

3.3 Architecture

In what follows an overview of the PoC built to investigate how well the reality of the situation corresponds with the expectations outlined in the previous section (3.2) is given. The code base for the implementation itself is made publicly available under an AGPL-3.0 license[6].

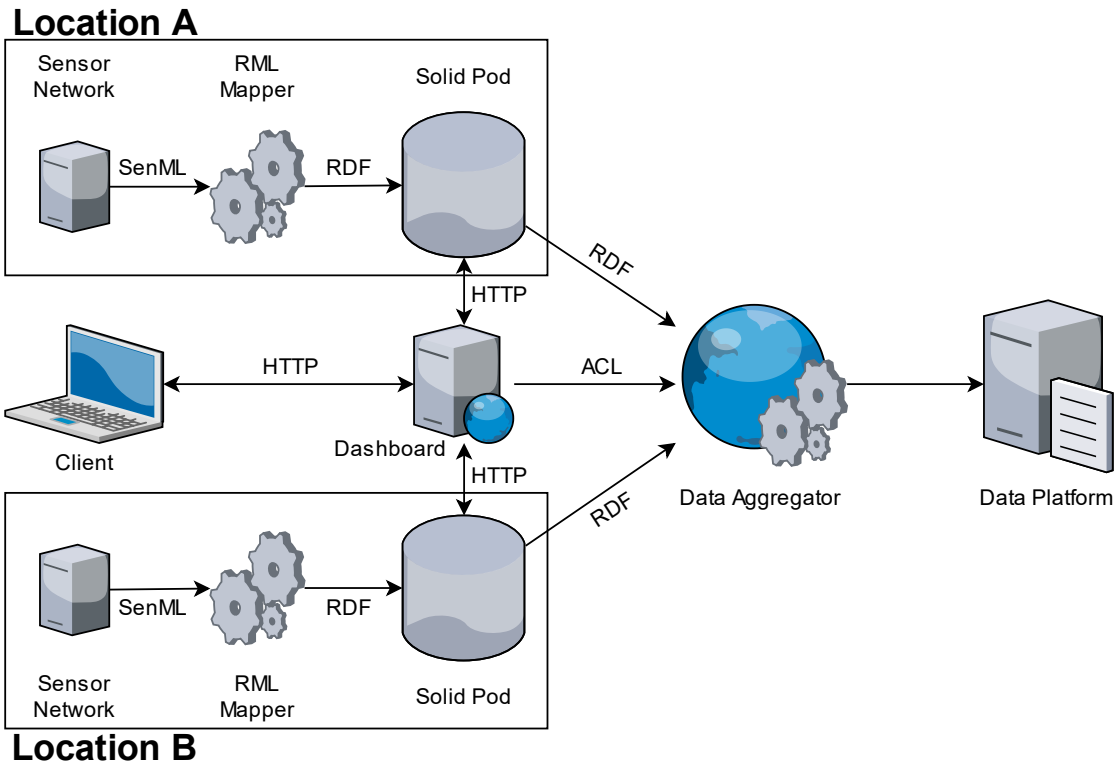


Figure 3.1: Diagram overview of the implemented system

Figure 3.1 shows a schematic overview of the PoC. In the following sections the function and design of the components in this diagram are discussed.

[6] https://github.com/lab9k/Solid-Pods-For-IoT

3.3.1 Sensor Network

The schematic in figure 3.1 contains two sensor networks at two different locations, the office and a home environment. These sensor networks act as the data producers in the system and each contain two distinct sensors.

Both sensors are developed using NodeMCU 1.0 development boards, based on the ESP8266 microchip which contains a 32-bit micro-controller, a full Transfer Control Protocol/Internet Protocol (TCP/IP) stack as well as Wi-Fi 2.4GHz capabilities, standardised by the Institute of Electrical and Electronics Engineers (IEEE) in the IEEE 802.11 b/g/n specifications [8]. They can be programmed using the Arduino IDE[7] to realise the desired functionality.

One sensor uses a DHT11 device to measure the temperature (range: 0°C to 50°C, precision: 2°C) and relative humidity (range: 20%RH to 95%RH, precision: 5%RH), connected to the NodeMCU as described in [9].

The other sensor employs a light dependent resistor (LDR) in a voltage divider configuration as shown in figure 3.2. The output voltage is given by:

$$V_{out} = \frac{R}{R_{LDR} + R} V_{CC} \tag{3.1}$$

Since, for the LDR, the logarithm of the conductance is inversely proportional to the logarithm of the illuminance on its surface, this circuit will result in an output voltage which rises for increasing light intensity [10], hence it can be used as a device to measure this property. Since no calibration has been done however, we will simply report the measurement as a fraction with respect to its maximum value V_{CC}.

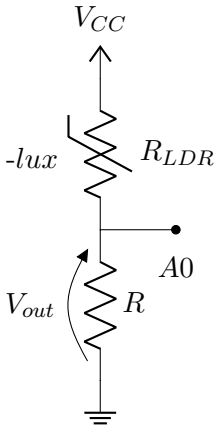


Figure 3.2: LDR in voltage divider

The sensor devices publish their measurements over the Message Queuing Telemetry Transport (MQTT) protocol in the Sensor Measurement List (SenML) format using its JSON serialisation.

[7] `https://www.arduino.cc/en/Main/Software`

MQTT is a lightweight bi-directional protocol for machine-to-machine (M2M) communication over TCP/IP. Thanks to its publish-subscribe architecture, only one agent in the system (the broker) needs to be available at all times, while the other agents (the clients) can be online only intermittently. This makes the protocol ideal for constrained low-power wireless IoT devices that communicate over IP [11]. For the MQTT broker, we opted for an instance of the open source Eclipse Mosquitto[8] program.

SenML is a proposed standard from the Internet Engineering Task Force (IETF) for representing sensor measurements in an efficient manner with support for JSON, CBOR, XML and EXI as serialisation formats. It is meant to be used as a media format to exchange messages between constrained devices [12].

Listings 1 and 2 give examples of messages sent by the two sensors in SenML's JSON serialisation.

```
[
  {
    'bn': 'urn:dev:mac:807d3afffe367a58_',
    'bt': '1599488174',
    'n': 'temperature',
    't': '30',
    'u': 'Cel',
    'v': '24'
  },
  {
    'n': 'humidity',
    't': '30',
    'u': '\%RH',
    'v': '42'
  }
]
```

Listing 1: Temperature and humidity measurement in SenML's JSON serialisation

```
[
  {
    'bn': 'urn:dev:mac:b4e62dfffe703f4d_',
    'bt': '1599488159',
    'n': 'light',
    't': '30',
    'v': '79'
  }
]
```

Listing 2: Light intensity measurement in SenML's JSON serialisation

Figure 3.3 shows the circuits described above built on a breadboard.

[8] https://mosquitto.org/

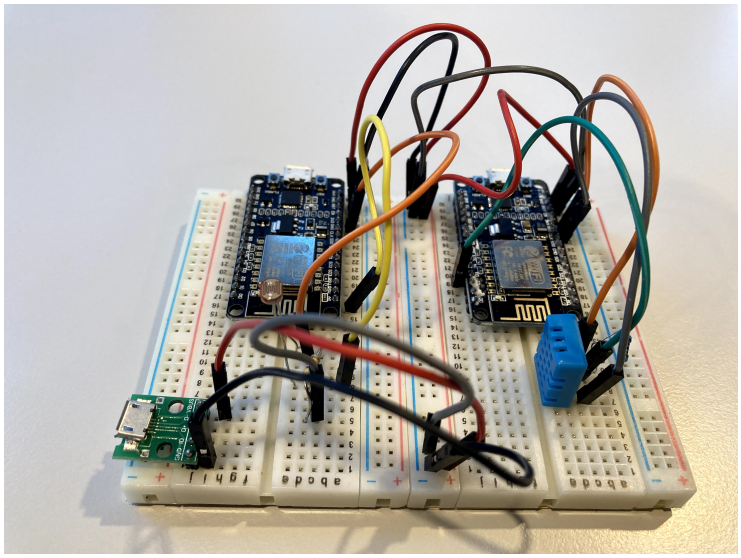


Figure 3.3: Sensor hardware based on NodeMCU development boards

3.3.2 RML Mapper

The RML mapper developed for this project is based on the one used in previous work [1][5], though some important improvements and changes were made, the latter mainly due to the use of a different protocol stack: MQTT and SenML rather than LwM2M. It consists of 3 parts: The receiver, the translator and the saver.

The receiver extracts the data from the sensor network. It comprises an MQTT client which subscribes at the MQTT broker to the topic used by the sensors for message publication. It parses the incoming SenML messages as JSON, reconstructs the data contained in the records and passes it down to the translator.

The translator accepts the incoming data from the receiver, applies a pre-processing setp and translates the data in RDF format. Appendix A gives an introduction to linked data and its description in RDF.

Since LwM2M uses its own specific data model, the previously built translator made use of a corresponding specialised ontology to make mapping as straight-forward as possible. The use of SenML as a message format however prevents us from reusing this previous implementation. Instead RML files for translation of SenML data to two different ontologies were developed:

- the Semantic Sensor Network (SSN) ontology[9], a World Wide Web Consortium (W3C) recommendation.
- the Smart appliances reference (Saref) ontology[10], developed by the Netherlands Organisation for Applied Scientific Research (TNO).

The core structure for describing a sensor and its measurements in both the SSN and Saref ontologies are shown in figures 3.4 and 3.5 respectively. Both ontologies share a lot of the same characteristics. They are both built around a Device or Sensor which makes Observations or Measurements, they both use or support the ontology of units measure (om2)[11] to describe units and measured quantities.

[9] `https://www.w3.org/TR/vocab-ssn/`
[10] `https://ontology.tno.nl/saref/`
[11] `http://www.ontology-of-units-of-measure.org/resource/om-2/`

These similarities are not a coincidence however. Saref is the result of research meant to improve integration of and interoperability between different existing standards [13]. Efforts have even been made in developing (partial) semantic translations between both ontologies [14].

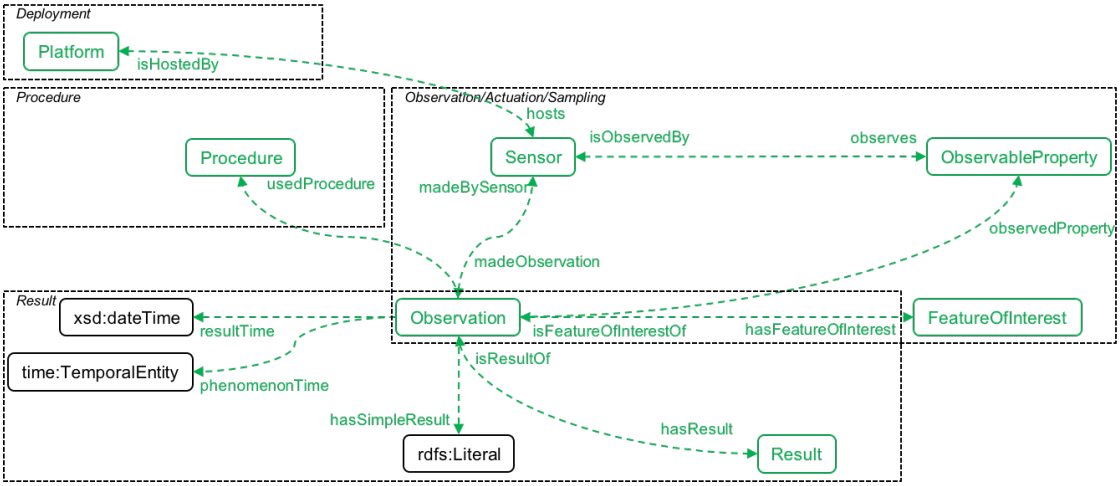


Figure 3.4: SSN ontology structure, source: [15, 4.3.1]

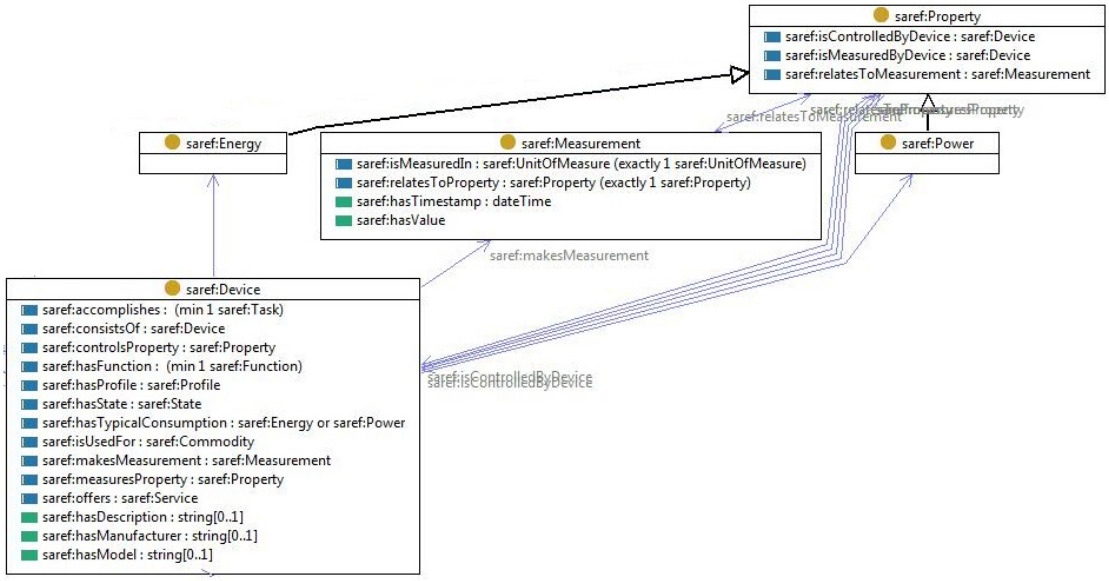


Figure 3.5: Saref ontology structure, adapted from: [16]

The mapping from SenML data to RDF and more specifically SSN has been covered in literature [6][7]. In this project a similar, though not identical approach is taken. In essence, it comes down to:

1. Converting the time data from unix time (e.g. 1599488159) to the DateTime format of the XML Schema Definition (XSD) (e.g. 2020-09-07T14:16:29Z) [17].
2. Map the list of supported units in SenML [12, 12.1] to equivalents found in the om2 ontology. A table containing such a mapping is given in appendix B.
3. Describe the sensor and its data using the pattern defined by the ontology.

The first two steps are implemented as a pre-processing action after which the data is passed on to an RML mapper which performs the third step: to actually translate the data. Listings 3 and 4 show the result of the mapping of the data from listings 1 and 2 for SSN and Saref respectively, serialised in the Terse RDF Triple Language (Turtle) [18].

```
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix om2: <http://www.ontology-of-units-of-measure.org/resource/om-2/> .
@prefix xml: <http://www.w3.org/2001/XMLSchema#> .

<dev:mac:807d3afffe367a58_temperature> a sosa:Sensor ;
        sosa:madeObservation <uuid:94ebf374-ba40-4d5c-9743-d742f61d9260> ;
        sosa:observes om2:Temperature .
<uuid:94ebf374-ba40-4d5c-9743-d742f61d9260> a sosa:Observation ;
        sosa:hasResult <uuid:94ebf374-ba40-4d5c-9743-d742f61d9260_result> ;
        sosa:resultTime "2020-09-07T14:16:44Z"^^xml:dateTime .
<uuid:94ebf374-ba40-4d5c-9743-d742f61d9260_result> om2:hasNumericalValue "24"^^xml:float ;
        om2:hasUnit om2:degreeCelsius ;
        a om2:Measure .

<dev:mac:807d3afffe367a58_humidity> a sosa:Sensor ;
        sosa:madeObservation <uuid:d4d03b48-03be-4f88-a39a-4507318047e9> ;
        sosa:observers om2:RelativeHumidity .
<uuid:d4d03b48-03be-4f88-a39a-4507318047e9> a sosa:Observation ;
        sosa:hasResult <uuid:d4d03b48-03be-4f88-a39a-4507318047e9_result> ;
        sosa:resultTime "2020-09-07T14:16:44Z"^^xml:dateTime .
<uuid:d4d03b48-03be-4f88-a39a-4507318047e9_result> om2:hasNumericalValue "42"^^xml:float ;
        om2:hasUnit om2:percent ;
        a om2:Measure .

<dev:mac:b4e62dfffe703f4d_light> a sosa:Sensor ;
        sosa:madeObservation <uuid:4b44fbbe-b3f8-4504-8eb0-bc1f3473a56e> .
<uuid:4b44fbbe-b3f8-4504-8eb0-bc1f3473a56e> a sosa:Observation ;
        sosa:hasResult <uuid:4b44fbbe-b3f8-4504-8eb0-bc1f3473a56e_result> ;
        sosa:resultTime "2020-09-07T14:16:29Z"^^xml:dateTime .
<uuid:4b44fbbe-b3f8-4504-8eb0-bc1f3473a56e_result> om2:hasNumericalValue "79"^^xml:float ;
        a om2:Measure .
```

Listing 3: RDF graph describing the data in listings 1 and 2 using the SSN ontology.

```
@prefix saref: <https://w3id.org/saref#> .
@prefix om2: <http://www.ontology-of-units-of-measure.org/resource/om-2/> .
@prefix xml: <http://www.w3.org/2001/XMLSchema#> .

<dev:mac:807d3afffe367a58_temperature> a saref:Device ;
        saref:makesMeasurement <uuid:94ebf374-ba40-4d5c-9743-d742f61d9260> ;
        saref:measuresProperty om2:Temperature .
<uuid:94ebf374-ba40-4d5c-9743-d742f61d9260> a saref:Measurement ;
        saref:hasTimeStamp "2020-09-07T14:16:44Z"^^xml:dateTime ;
        saref:hasValue "24"^^xml:float ;
        saref:isMeasuredIn om2:degreeCelsius .

<dev:mac:807d3afffe367a58_humidity> a saref:Device ;
        saref:makesMeasurement <uuid:d4d03b48-03be-4f88-a39a-4507318047e9> ;
        saref:measuresProperty om2:RelativeHumidity .
<uuid:d4d03b48-03be-4f88-a39a-4507318047e9> a saref:Measurement ;
        saref:hasTimeStamp "2020-09-07T14:16:44Z"^^xml:dateTime ;
        saref:hasValue "42"^^xml:float ;
        saref:isMeasuredIn om2:percent .

<dev:mac:b4e62dfffe703f4d_light> a saref:Device ;
        saref:makesMeasurement <uuid:4b44fbbe-b3f8-4504-8eb0-bc1f3473a56e> .
<uuid:4b44fbbe-b3f8-4504-8eb0-bc1f3473a56e> a saref:Measurement ;
        saref:hasTimeStamp "2020-09-07T14:16:29Z"^^xml:dateTime ;
        saref:hasValue "79"^^xml:float .
```

Listing 4: RDF graph describing the data in listings 1 and 2 using the Saref ontology.

In practice the messages for each sensor get translated separately and saved to local RDF graphs. These updates get batched locally and sent out to update the data in the Pod at fixed intervals to avoid overloading the Solid server with PATCH requests.

3.3.3 Solid Pod

Each of the sensor networks has its own Solid Pod to which the corresponding RML Mapper saves the measurements coming from the sensor network.

At the time of writing only one implementation of a Solid server is publicly available, Node Solid Server (NSS)[12], though a new open-source implementation, Community Solid Server (CSS)[13], is in development and Inrupt has an Enterprise-grade Solid Server (ESS)[14] in beta.

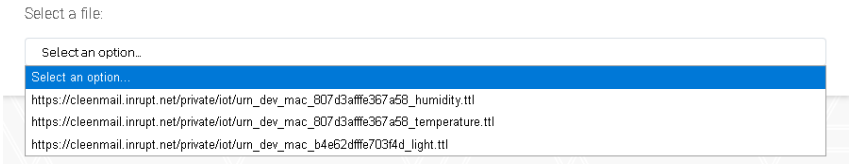
Due to time constraints we used Pods from a provider rather than a self-hosted system. For the sensor network in the home environment solid.community was used and in the office we opted for inrupt.net.

For illustrative purposes, we configured the RML mapper for the office environment to save data using the SSN ontology and the one for the home environment to use the Saref ontology.

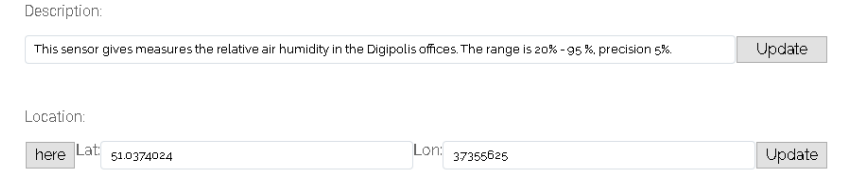
[12] https://github.com/solid/node-solid-server
[13] https://github.com/solid/community-server
[14] https://inrupt.com/products/enterprise-solid-server

3.3.4 Dashboard

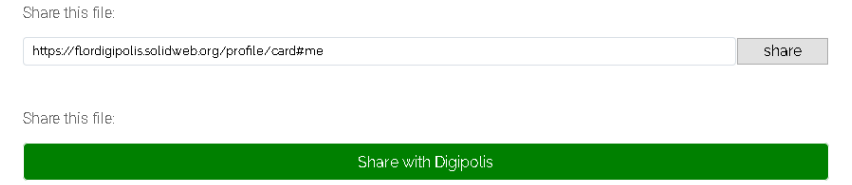
The web server shown in the centre of figure 3.1 hosts a dashboard application for visualising the IoT data saved in the Pods and managing which parties have access to them. Figure 3.6 shows screenshots of the dashboard's components.



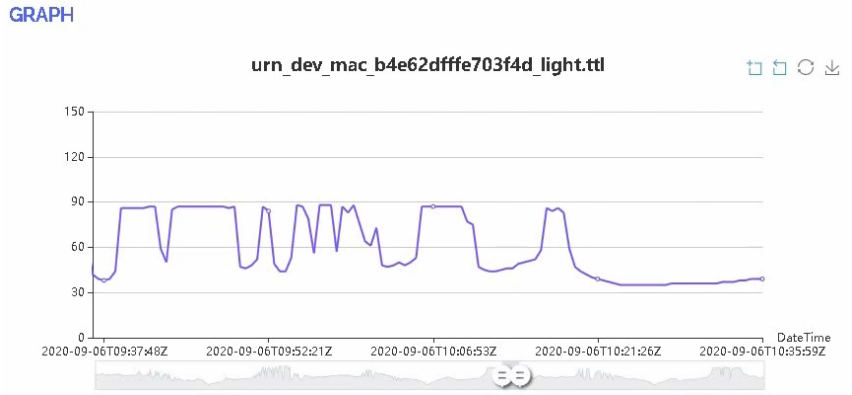
(a) Drop-down menu to select file



(b) Description and location fields



(c) Sharing fields



(d) Interactive graph showing sensor data

Figure 3.6: Screenshots of the dashboard

The different components of the dashboard provide the following functionality:

(a) Since the IoT data is published in different files for different sensor, a drop-down list is available to select which one the dashboard should show for inspection, sharing and editing.

(b) As listings 3 and 4 indicate, after the mapping there is little data available yet about the properties of the sensor: its brand, location, measurement range, precision, etc. are missing from the graph. Such missing information could easily be provided by the application that manages these devices. As a proxy to this solution we added two fields through which the sensor's location and description can be changed manually.

(c) Two ways of sharing the file were added to the dashboard as well. In the upper field one can enter a WebId with whom they want to share the data. The lower button grants read access to the file to a preconfigured agent representing the data aggregator. It also makes a call to an Application Programming Interface (API) to let the aggregator know it has access to a new resource.

(d) Finally, an interactive graph visualising the data saved in the Pod is also included. It offers compatibility for reading data described using either the SSN or Saref ontology.

3.3.5 Data Aggregator

The data aggregator contains two distinct components:

- A Representational State Transfer (REST) API which acts as a compatibility layer for Solid interactions.
- A pipeline for data fetching, merging, processing and saving.

3.3.5.1 REST API

Since currently the capability of interacting with Solid Pods is nonexistent in most data processing platforms a REST API, which acts as a compatibility layer, was developed.

It keeps track of the resources it has access to and exposes these files through GET requests to clients. The API supports the following requests:

- GET /v1/localfiles or /v1/solidfiles: Returns a list of files (hosted locally or on a collection of Solid Pods) the data aggregator has access to.
- GET /v1/localfiles/filename.ttl or /v1/solidfiles/filename.ttl: Returns the contents of the file (hosted locally or on a Solid Pod) with name filename.ttl in Turtle format.
- PUT /v1/solidfiles/filename.ttl: Adds the file to the list of resources the aggregator has access to. The body should contain the URL where the file is stored.
- DELETE /v1/solidfiles/filename.ttl: Removes the file from the list of resources the aggregator has access to. The body should contain the URL where the file is stored.

3.3.5.2 Pipeline

The data aggregator's pipeline was built using LinkedPipes ETL[15]. This platform offers a graphical interface to build pipelines for the processing of linked data. The pipeline built in this project is shown in figure 3.7.

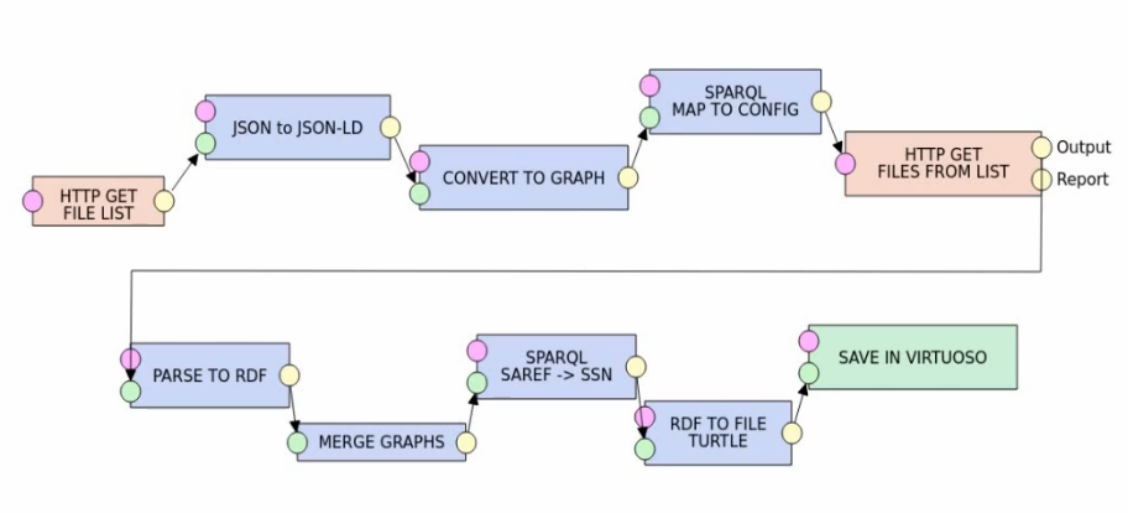


Figure 3.7: Pipeline for processing the IoT data stored in our Solid Pods.

The upper row of blocks is responsible for interacting with the API to fetch the list of files available to the data aggregator and subsequently fetching the data in those files. The blocks in the lower row serve the purpose of merging, processing and saving the data. More specifically:

- PARSE TO RDF: Takes the contents from the fetched files and interprets them as Turtle to form actual RDF graphs which are ready for processing.
- MERGE GRAPHS: Merges the RDF graphs into a single one.
- SPARQL SAREF ->SSN: Uses a SPARQL Query Language for RDF (SPARQL) CONSTRUCT query to translate the data described using the Saref ontology to SSN. This way the final graph can be queried without having to take the presence of multiple ontologies into account.
- RDF TO FILE TURTLE: Serialise the merged and converted RDF graph to a Turtle file.
- SAVE IN VIRTUOSO: Sends the graph to our data platform of choice, Virtuoso.

[15] `https://etl.linkedpipes.com/`

3.3.6 Data platform

For the data platform we opted for Virtuoso[16], developed by Openlink Software. This platform is dominant in applications concerning the Linked Open Data Cloud thanks to its support for handling linked data graphs in a scaleable manner.

Virtuoso exposes a SPARQL endpoint through which the data in its triple store can be queried and delivered in different formats. This endpoint offers a way of publishing the data for analysis or use in applications. Listings 5 and 6 give examples of SPARQL queries that can be used to retrieve data from the graph.

```
prefix sosa: <http://www.w3.org/ns/sosa/>
prefix om2: <http://www.ontology-of-units-of-measure.org/resource/om-2/>
prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
prefix terms: <http://purl.org/dc/terms/>

SELECT ?name ?description ?property ?location
WHERE {
    ?name a sosa:Sensor .
    OPTIONAL{?name terms:description ?description .}
    OPTIONAL{?name sosa:observes ?property .}
    OPTIONAL{?name geo:lat_long ?location .}
}
```

Listing 5: SPARQL query to retrieve a list of sensors with their general properties.

```
prefix sosa: <http://www.w3.org/ns/sosa/>
prefix om2: <http://www.ontology-of-units-of-measure.org/resource/om-2/>

SELECT ?name ?time ?value ?unit ?property
WHERE {
    BIND(om2:Temperature AS ?property)

    ?name a sosa:Sensor .
    ?name sosa:observes ?property .
    ?name sosa:madeObservation ?uuid .
    ?uuid a sosa:Observation .
    ?uuid sosa:resultTime ?time .
    ?uuid sosa:hasResult ?uuidresult .
    ?uuidresult a om2:Measure .
    OPTIONAL{?uuidresult om2:hasUnit ?unit .}
    ?uuidresult om2:hasNumericalValue ?value .
    FILTER (str(?time) > "2020-09-06T06:00:00Z")
    FILTER (str(?time) < "2020-09-06T06:01:00Z")
}
```

Listing 6: SPARQL query to retrieve a list of temperature measurements between 6:00AM and 6:01AM on September 6th 2020.

[16] https://virtuoso.openlinksw.com/

3.4 Evaluation

Section 3.2 mentions the possible advantages of using Solid Pods for IoT applications. This section takes a critical look at the developed implementation and evaluates the extent to which these advantages were realised.

3.4.1 Interoperability

One of the main premises of Solid is the interoperability between apps thanks to the use of common data structures described by the ontologies used in linked data. While our application makes use of RDF to describe the sensor data, this is not in itself enough to guarantee that other apps will know what to do with it. The main reason for this is that no description is given about what data our graph is guaranteed to contain, only which ontologies it uses to describe that data.

The author of [19] proposes a solution to this problem by making use of shapes, which describe specific views of data chunks. Shapes allow an application to verify whether given data conforms to a predefined data structure and could even be used for processes akin to content-negotiation in web APIs.

Another solution could be the use of RDF Data Cubes, which is a system to publish multi-dimensional data sets in RDF format and allows you to describe the structure of a data set alongside the data itself [20].

3.4.2 Access Control

Solid not only promises full control over personal data, but does so without throwing away the advantages offered by the web. This requires a well thought-out mechanism to manage who can access what data and in what way. Solid makes use of the Web Access Control (WAC) mechanism [21], which defines Access Control List (ACL) resources to describe Read, Write, Append and Control access to documents. An example of an ACL adding read permission to an agent is provided in listing 7.

```
@prefix acl: <http://www.w3.org/ns/auth/acl#>.
<#owner> a acl:Authorization;
    acl:agent </profile/card#me>;
    acl:accessTo <./urn_dev_mac_807d3afffe367a58_humidity.ttl>;
    acl:mode acl:Read, acl:Write, acl:Control.
<#Read-0> a acl:Authorization;
    acl:agent <https://flordigipolis.solidweb.org/profile/card#me>;
    acl:accessTo <./urn_dev_mac_807d3afffe367a58_humidity.ttl>;
    acl:mode acl:Read.
```

Listing 7: ACL for the humidity resource after granting read access to an agent.

At the time of writing the most granular level of access control possible on Solid is on files, not the data within them. This implies that if you only want to share a subset of data (e.g. measurements between certain dates) with a third party, this has to split of into a separate file, which is quite cumbersome.

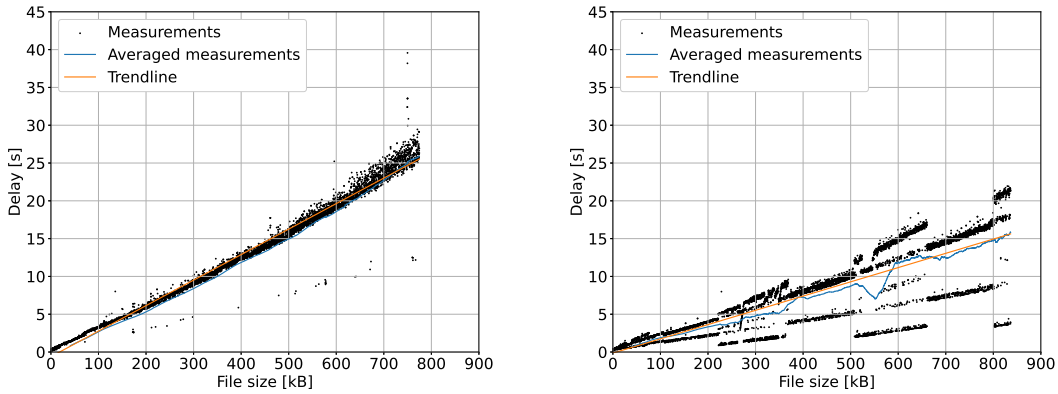
This is currently an open problem in the Solid community with proposed solutions such as content-addressable RDF [22].

A better approach might be to save only the most recent data in its raw form and keep historical records for derived data on which some processing has already been performed in line with the trend towards edge computing [23]. This pre-processed data chunks could then be shared with third parties rather than the whole raw data set.

3.4.3 Performance

It's well known that the performance aspects of web applications such as loading times are important factors of the perceived practicality of these applications.

To evaluate this property for the current Solid implementation, a log was kept of delays between the PATCH requests sent by the RML Mapper and the 200 OK response by the Solid server. Figure 3.8 shows these measurements in relation to the file size to which the new data had to be appended for both Pod providers (both running NSS v5.5.1).



(a) Solid Community Server

(b) Inrupt Server

Figure 3.8: Server delay for PATCH request on different file sizes.

The graph in figure 3.8a shows a linear relation between file size of the total database and loading time, with an increase of $33.74\frac{ms}{kB}$. The results in figure 3.8b are less consistent, but show a similar trend, with a delay trendline that increases with $18.85\frac{ms}{kB}$. These results are, mildly put, concerning, since for the Solid Community Server a file size of a mere 600 kB corresponds to a delay of almost twenty seconds.

Since we did not carry out these tests on a self-hosted instance of the NSS and no other implementations of a Solid server are publicly available as of yet, there is no telling if the cause lies in a lack of server resources, the way NSS is implemented or the mechanisms defined in the Solid specifications.

3.4.4 Storage Efficiency

Incorporating the context of the data into the data itself, as is done in RDF with the use of ontologies, has its benefits but also its costs.

After translating the SenML messages from listings 1 and 2 to RDF Turtle format, as showng in listings 3 and 4, we obtain payload sizes of 1409 bytes and 1044 bytes respectively for SSN and Saref. In comparison, the SenML messages themselves only contain 240 bytes worth of content. This corresponds to respective increases of total file sizes by factors 5.87 and 4.35. This aspect certainly has to be improved upon in order to produce a scaleable solution.

3.5 Future work

This section gives an overview of the extensions, changes and improvements that could be made to this system to improve its scalability and adoptability.

3.5.1 Sensor network and RML Mapper

In this project the sensor network and RML Mapper are considered as separate entities, but it doesn't have to be this way. Ideally, Solid would be incorporated as storage mechanism in existing IoT platforms, though in the meantime this concept could be evaluated by adding Solid as a persistance method in an open platform like OpenHAB[17].

Alternatively, for IoT platforms with traditional storage methods the mapper could be kept as a middleware service running alongside a Solid Pod, which interfaces with the platform's API and publishes the collected data to the Pod.

3.5.2 Dashboard and Personalisation

Since the IoT platforms mentioned in the previous section generally offer a user interface (UI) alongside it, a separate user interface wouldn't be strictly necessary. This doesn't mean however that this couldn't be advantageous. The separation of data and applications offered by Solid opens up the possibility of different apps using the same data. A good example of this could be the use of different dashboards depending on the user's comfort level with technology, give beginners a clean and simple UI and offer power users as much configurability as they can handle.

3.5.3 Solid User Experience

Besides the UI for the applications, the user experience (UX) when using Solid is an important aspect that could be improved upon.

Rather than treating the Solid Pod as network attached storage (NAS) device that also happens to support application data, one could imagine a situation in which it is used as a platform to enable services equivalent to or even surpassing those currently running on closed systems such as those offered by Apple, Google, Amazon, etc.

Since all the data concerning a certain individual would be saved in a central platform, it should be possible to make use of these by means of edge computing to offer context-based and personalised information and services through interaction points such as smart displays and voice assistants [23]. Many pieces still need to come together however before such a system becomes feasible.

More practically, in the short term it would be beneficial to improve the experience of setting up a local instance of a Solid Pod with the possibility of encrypted backups in the cloud for the case that something goes wrong. This would allow application developers to more easily experiment with the different functionalities Solid has to offer since they're less likely to be limited by bottlenecks such as total storage capacity and bandwidth limitations.

[17] `https://www.openhab.org/`

3.5.4 API and Data Aggregator

Between the development of the API covered in section 3.3.5.1 and the writing on this report a tool called Walder for setting up web APIs on top of decentralised knowledge graphs such as Solid Pods was published[18]. The use of this tool would allow a more flexible and thorough approach for building compatibility layers between Solid Pods and data aggregation or processing tools.

Of course support for Solid Pods immediately baked into these tools would be even better. The discovery of resources available to them could happen through accessing the public type registration of user profiles, but this level of adoption still requires a large amount of effort.

3.5.5 Data Platform

While this PoC makes a first effort in employing Solid Pods in a distributed network for IoT data publication, it hasn't been applied to an actual application yet. Assuming the upcoming Solid server implementations solve the performance issues mentioned in 3.4.3, the possibility for this is not that far of.

Good project candidates within Digipolis would be the IoT and CoT projects in which a large amount of data from different sources is already being used, such as the TMaaS project[19] or the IoT registry[20].

[18] `https://github.com/KNowledgeOnWebScale/walder`
[19] `https://drive.tmaas.eu/`
[20] `https://iot.lab9k.gent/`

3.6 Applications

The current online environment treats customers as data producers to the companies they use (free) services from. With the adoption of IoT devices, the amount of data that can be collected not only grows but also transfers from being purely online to revealing aspects of the physical world.

This section will discuss how the use of Solid in (IoT) applications can be beneficial to both parties involved: the data producer and the service provider.

3.6.1 Data producer perspective

As shown by the recent trends on the web, customers are willing to share their data with third parties if they get something in return. Currently the most frequent situation is the following: service providers offer their products to customers at a very low cost, often even for free, and in return these companies enjoy a monopoly on the data produced by using these services [24].

By making use of Solid this deal could be renegotiated. Since the data are saved in an interoperable fashion, multiple applications could compete for the function of data platform. Some of these might opt for a more classical payment model in which no data sharing is required, while others could keep the current model of data in exchange for services. Another possibility is even to have companies which offer no services whatsoever but set up smart contracts on the blockchain to give monetary compensation for the data provided [25]. What the end result will look like is unknown and might differ strongly between different groups of users, but Solid offers a technological framework with the flexibility to put such different models to the test.

An additional benefit of the possibility to have multiple applications using the same data is the aspect of personalisation. Different companies can create separate products that cater to different groups of users (e.g. elderly vs. youth, professional vs. personal use) while at the core still reusing the same data.

Furthermore, these access mechanisms could make it significantly easier for grassroots organisations, living labs and citizen science groups to start and manage projects in which citizens collaborate to collect data that can freely be used for scientific research, or increase the transparency about the data sharing process in more classical research setups.

3.6.2 Service provider perspective

While the renegotiation of the data sharing deal mentioned above would mean that companies will lose their monopolistic grip over the data produced by their products, they will also enjoy some advantages caused by the adoption of Solid.

First of all, the loss of this monopolistic grip would mean these companies suddenly could have access to data produced by other applications beside their own, provided the user is compelled in some way to give them access to those data.

Additionally, with the recent rise of data protection laws such as the EU's General Data Protection Regulation (GDPR)[21] and the California Consumer Privacy Act (CCPA)[22] the do's and don'ts of handling personal data have become increasingly complicated and this trend will only get worse as governments around the world develop their own regulations.

[21] `https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en`
[22] `https://oag.ca.gov/privacy/ccpa`

Since with Solid the control over the data and who has access to them lies with the user, the applications produced using this technology are almost guaranteed to be inherently compliant with above laws and regulations.

Finally, the fact that Solid users can know exactly which data gets shared with whom and at what times drastically increases the transparency of this data sharing process. As a result it is likely that users will be more trusting with their data and actually share it willingly.

3.7 Conclusion

The introduction (3.1) introduces two movements, the semantic Web of Things (WoT) and the uniformisation of communication standards, both striving to solve the problem of fragmentation in the field of IoT, and identifies Solid as a possible midpoint where these initiatives could join, which is further motivated in section 3.2.

In section 3.6 we consider the Solid technology in its socio-economic context and explore the advantages its adoption for applications might result in from the perspective of both the individual as a data producer and the companies providing services.

Section 3.3 gives an overview of the PoC built to evaluate the current state of affairs and in sections 3.4 and 3.5 the limitations of this system as well as future work that could be done to improve upon these limitations as well as to make it a more viable solution is discussed.

In conclusion, Solid is an interesting project since it has quite large implications for the organisation of the world wide web and by extension the Internet of Things. Currently however, the technology is not ready yet for prime time and it still has a long way to go before it's truly suitable for mass adoption.

Chapter 4: Personal Evaluation

As an electrical and electronics engineering student, the topics I usually occupy myself with are those of circuit analysis, hardware design and low-level software development, such as for embedded systems. In this internship however, I was able to explore these systems on a whole other level of abstraction and learn more about a range of new topics such as data models, the semantic web and the development of RESTful APIs.

On a personal level, this internship served as my first contact to the industrial sphere of engineering. I got to take a look behind the screens of a company, Digipolis Ghent, that occupies itself with a lot of varying tasks and subjects, from the maintenance and support of of-the-shelf IT solutions to high-tech project-based work in collaboration with international partners such as the SCORE project.

Through the daily scrum sessions with Hans Fraiponts, my internship mentor, as well as Jef Willems and Tim Van Achte from the foresight group, I learnt a lot about what it takes to organise a project in order to complete it successfully, while taking many aspects such as budget, timing and security into account.

For above reasons I can only conclude this internship has been an excellent learning experience for me and I am hence grateful to have had the opportunity to do this.

Bibliography

[1] A. Bomhals, K. Hens, T. Paelman, and F. Sanders, "Iot en solid: Een applicatie voorprivégegevens met lwm2m," *UGent*, 2020. [Online]. Available: https://www.florsanders.be/files/Solid_en_IoT_verslag.pdf

[2] P. Bloem and G. K. D. Vries, "Machine learning on linked data, a position paper," 09 2014.

[3] D. Le-Phuoc and M. Hauswirth, "Linked Data for Internet of Everything," in *Integration, Interconnection, and Interoperability of IoT Systems*, R. Gravina, C. E. Palau, M. Manso, A. Liotta, and G. Fortino, Eds. Cham: Springer International Publishing, 2018, pp. 129–148, series Title: Internet of Things. [Online]. Available: http://link.springer.com/10.1007/978-3-319-61300-0_7

[4] G. Loseto, S. Ieva, F. Gramegna, M. Ruta, F. Scioscia, and E. D. Sciascio, "Linked Data (in resourceless) Platforms: a mapping for Constrained Application Protocol," in *The Semantic Web - ISWC 2016*. Springer International Publishing, 2016.

[5] B. Moons, F. Sanders, T. Paelman, and J. Hoebeke, "Decentralized linked open data in constrained wireless sensor networks," *Ghent University, unpublished*, 2020.

[6] X. Su, H. Zhang, J. Riekki, A. Keränen, J. K. Nurminen, and L. Du, "Connecting iot sensors to knowledge-based systems by transforming senml to rdf," *Procedia Computer Science*, vol. 32, pp. 215 – 222, 2014, the 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050914006176

[7] V. Kothandapani and K. Sathiyamurthy, *A Framework for Semantic Annotation and Mapping of Sensor Data Streams Based on Multiple Linear Regression: Methods and Protocols*, 01 2019, pp. 211–222.

[8] *ESP8266EX Datasheet*, Espressif Systems, 2 2018, v5.8. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[9] *DHT11, DHT22 and AM2302 Sensors*, Adafruit, 9 2020. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/dht.pdf

[10] *CdS Photoconductive Cells*, Lida Optical&Electronic Co. [Online]. Available: https://pi.gate.ac.uk/pages/airpi-files/PD0001.pdf

[11] *MQTT Version 5.0*, Oasis, 3 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf

[12] C. Jennings, Z. Shelby, J. Arkko, A. Keranen, and C. Bormann. (2018) Sensor measurement lists (senml). IETF. [Online]. Available: https://tools.ietf.org/html/rfc8428

[13] L. Daniele, F. den Hartog, and J. Roes, "Study on semantic assets for smart appliances interoperability : D-s4: Final report," 2015.

[14] J. Moreira, L. Daniele, L. Ferreira Pires, K. Wasielewska, P. Szmeja, W. Pawlowski, M. Ganzha, and M. Paprzycki, "Towards iot platforms' integration: Semantic translations between w3c ssn and etsi saref," 09 2017.

[15] Semantic sensor network ontology. W3C. [Online]. Available: https://www.w3.org/TR/vocab-ssn/#Observations-overview

[16] saref:device. TNO. [Online]. Available: https://ontology.tno.nl/saref/saref_Device.html

[17] S. Gao, C. M. Sperberg-McQueen, H. S. Thompson, N. Mendelsohn, D. Beech, and M. Maloney. (2012) W3c xml schema definition language (xsd). W3C. [Online]. Available: https://www.w3.org/TR/xmlschema11-1/

[18] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, and G. Carothers. (2014) Rdf 1.1 turtle. W3C. [Online]. Available: https://www.w3.org/TR/turtle/

[19] R. Verborgh. (2020) Shaping linked data apps. [Online]. Available: https://ruben.verborgh.org/blog/2019/06/17/shaping-linked-data-apps/

[20] D. Reynolds, R. Cyganiak, and J. Tennison. (2014) The rdf data cube vocabulary. [Online]. Available: https://www.w3.org/TR/vocab-data-cube/

[21] D. Zagidulin, M. de Jong, K. Kjernsmo, R. Verborgh *et al.* (2020) Web access control (wac). Solid. [Online]. Available: https://github.com/solid/web-access-control-spec

[22] pukkamustard. (2020) Content-addressable rdf. [Online]. Available: https://openengiadina.net/papers/content-addressable-rdf.html

[23] J. Ren, Y. Pan, A. Goscinski, and R. A. Beyah, "Edge computing for the internet of things," *IEEE Network*, vol. 32, no. 1, pp. 6–7, Jan 2018.

[24] M. Falch, A. Henten, R. Tadayoni, and I. Windekilde, "Business models in social networking," 01 2009.

[25] A. Le-Tuan, D. Hingu, M. Hauswirth, and D. Le-Phuoc, "Incorporating blockchain into rdf store at the lightweight edge devices," in *Semantic Systems. The Power of AI and Knowledge Graphs*, M. Acosta, P. Cudré-Mauroux, M. Maleshkova, T. Pellegrini, H. Sack, and Y. Sure-Vetter, Eds. Cham: Springer International Publishing, 2019, pp. 369–375.

[26] Linked data. W3C. [Online]. Available: https://www.w3.org/standards/semanticweb/data

[27] G. Klyne, J. J. Carroll, and B. McBride. Rdf 1.1 concepts and abstract syntax. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/

Appendix A: Linked data and RDF

Linked data is the data structure powering the sematic web. It makes it possible to reason about decentralised data by relating it to each other, i.e. linking it [26].

The Resource Description Framework (RDF) is a standardised format for linked data, in which all data sets are described as a directed graph. The nodes (IRIs, literals or blanks) are interrelated by means of predicates, which point from subject to object. The combination (subject, predicate, object) is called a triple [27].

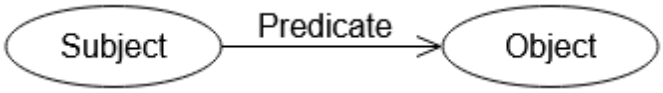


Figure A.1: Components of an RDF graph, source: [27, Fig.1].

To describe data in RDF one makes use of ontologies, which act as a lexicon describing which subjects, predicates and objects are available and how they can be combined. An example is the Friend Of A Friend (FOAF) ontology[1], which can be used to describe properties of and relations between people. Below we give an example of a valid RDF triple.

```
@base <https://www.example.com/> .
@prefix foaf: <https://xmlns.com/foaf/0.1/> .
<JohnDoe> foaf:knows <JaneDoe> .
```

In this example `<JohnDoe>` is the subject, `<JaneDoe>` the object and `foaf:knows` the predicate. The syntax used to describe above triple is the Terse RDF Triple Language (Turtle), which is a serialisation format offering a compact and easily comprehensible notation of RDF graphs.

While the concepts of linked data en RDF might seem unnecessarily complicated at first, its properties of incorporating the context within the data and allowing data structures more complicated than a hierarchical tree are highly advantageous when it comes to the ability for machines to process it.

[1] `http://xmlns.com/foaf/spec/`

Appendix B: SenML-om2 table of units

@prefix om2: <http://www.ontology-of-units-of-measure.org/resource/om-2/ >.

Symbol	Description	OM2 Unit	OM2 Quantity
m	meter	om2:metre	om2:Length
kg	kilogram	om2:kilogram	om2:Mass
g	gram	om2:gram	om2:Mass
s	second	om2:second-Time	om2:Time
A	ampere	om2:ampere	om2:ElectricCurrent
K	kelvin	om2:kelvin	om2:Temperature
cd	candela	om2:candela	om2:LuminousIntensity
mol	mole	om2:mole	om2:AmountOfSubstance
Hz	hertz	om2:hertz	om2:Frequency
rad	radian	om2:radian	om2:Angle
sr	steradian	om2:steradian	om2:SolidAngle
N	newton	om2:newton	om2:Force
Pa	pascal	om2:pascal	om2:Pressure
J	joule	om2:joule	om2:Energy
W	watt	om2:watt	om2:Power
C	coulomb	om2:coulomb	om2:ElectricCharge
V	volt	om2:volt	om2:ElectricPotential
F	farad	om2:farad	om2:Capacitance
Ohm	ohm	om2:ohm	om2:ElectricalResistance
S	siemens	om2:siemens	om2:ElectricalConductance
Wb	weber	om2:weber	om2:MagneticFlux
T	tesla	om2:tesla	om2:MagneticFluxDensity
H	henry	om2:henry	om2:Inductance
Cel	degrees Celsius	om2:degreeCelsius	om2:Temperature
lm	lumen	om2:lumen	om2:LuminousFlux
lx	lux	om2:lux	om2:Illuminance
Bq	becquerel	om2:becquerel	om2:Frequency
Gy	gray	om2:gray	om2:AbsorbedDose
Sv	sievert	om2:sievert	om2:AbsorbedDose
kat	katal	om2:katal	om2:CatalyticActivity
m2	square meter (area)	om2:squareMetre	om2:Area
m3	cubic meter (volume)	om2:cubicMetre	om2:Volume
l	liter (volume)	om2:litre	om2:Volume
m/s	meter per second (velocity)	om2:metrePerSecond-Time	om2:Velocity
m/s2	meter per square second (acceleration)	om2:metrePerSecond-TimeSquared	om2:Acceleration
m3/s	cubic meter per second (flow rate)	om2:cubicMetrePerSecond-Time	om2:VolumetricFlowRate
l/s	liter per second (flow rate)	undefined	undefined
W/m2	watt per square meter (irradiance)	om2:wattPerSquareMetre	om2:Irradiance
cd/m2	candela per square meter (luminance)	om2:candelaPerSquareMetre	om2:Luminance
bit	bit (information content)	om2:bit	om2:InformationCapacity
bit/s	bit per second (data rate)	om2:bitPerSecond-Time	om2:SymbolRate
lat	degrees latitude	undefined	undefined
lon	degrees longitude	undefined	undefined
pH	pH value (acidity; logarithmic quantity)	undefined	om2:Acidity
dB	decibel (logarithmic quantity)	undefined	om2:Magnitude
dBW	decibel relative to 1 W (power level)	undefined	undefined
Bspl	bel (sound pressure level; logarithmic quantity)	undefined	undefined
count	1 (counter value)	om2:one	om2:Number
/	1 (ratio, e.g., value of a switch)	om2:one	om2:Ratio
%	1 (ratio, e.g., value of a switch)	om2:one	om2:Ratio
%RH	percentage (relative humidity)	om2:percent	om2:RelativeHumidity
%EL	percentage (remaining battery energy level)	om2:percent	om2:Percentage
EL	seconds (remaining battery energy level)	om2:second-Time	om2:Time
1/s	1 per second (event rate)	om2:reciprocalSecond-Time	om2:Frequency
1/min	1 per minute (event rate, "rpm")	undefined	undefined
beat/min	1 per minute (heart rate in beats per minute)	undefined	undefined
beats	1 (cumulative number of heart beats)	om2:one	om2:Number
S/m	siemens per meter (conductivity)	om2:siemensPerMetre	om2:ElectricalConductivity