WoTify: A platform to bring Web of Things to your devices

Ege Korkan
ege.korkan@tum.de
Technical University of Munich
Germany

Hassib Belhaj Hassine
hassib.belhaj@tum.de
Technical University of Munich
Germany

Verena Eileen Schlott
verena.schlott@campus.lmu.de
Ludwig Maximilian University of Munich
Germany

Sebastian Käbisch
sebastian.kaebisch@siemens.com
Siemens AG, Germany

Sebastian Steinhorst
sebastian.steinhorst@tum.de
Technical University of Munich
Germany

ABSTRACT

The Internet of Things (IoT) has already taken off, together with many Web of Things (WoT) off-the-shelf devices, such as Philips Hue lights and platforms such as Azure IoT. These devices and platforms define their own way of describing the interactions with the devices and do not support the recently published WoT standards by World Wide Web Consortium (W3C). On the other hand, many hardware components that are popular in developer and maker communities lack a programming language independent platform to integrate these components into the WoT, similar to *npm* and *pip* for software packages. To solve these problems and nurture the adoption of the W3C WoT, in this paper, we propose a platform to WoTify either existing hardware by downloading new software in them or already existing IoT and WoT devices by describing them with a Thing Description.

CCS Concepts: • **Software and its engineering** → **Software libraries and repositories**; *Abstraction, modeling and modularity*;

Keywords: IoT, WoT, Marketplaces

1 INTRODUCTION

In the recent years, the Internet of Things (IoT) was the focus of many analyses with billions of devices expected to be connected to the Internet. Some of these billion devices are already out there, connected to the Internet, providing sensing and actuation capabilities. More are waiting to be connected to the Internet and thus be part of the IoT.

When we look at the Internet, most of the applications run on the Web layer. This has resulted on the Web having one of the biggest developer communities behind [1] and we can foresee the same happening for the Web layer of IoT that was named the Web of Things (WoT). The WoT in itself is a concept of having Web technologies for IoT devices and offer the same ease of application development composed of IoT devices. WoT differentiates itself from the Web by being related to underlying hardware such as sensors, controllers, robots, actuators and more, something that was foreign to the Web community.

As with all the Web standards, the Web of Things is also being standardized by the World Wide Web Consortium (W3C) by means of different building blocks such as Thing Description (TD)[2], Scripting API[3] and more[1]. This standardization effort has started on 2016 [4] and the Thing Description standard will be released by the end of 2019. Being this recent, there has not been enough time to build a community around the W3C WoT, which is important for the acceptance and dissemination of the standard.

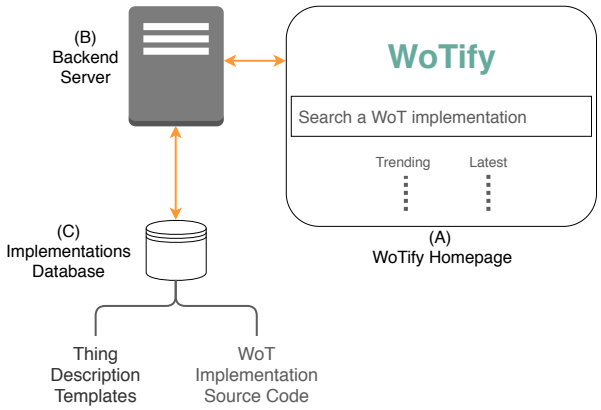


Figure 1: WoTify concept illustrating different components of the WoTify system

When we look at the wider scope of the IoT, there are multiple proposals of IoT marketplaces [5][6]. These marketplaces offer a way to obtain applications for a wide variety of IoT devices and contribute to the IoT landscape with a convenient and standardized way to obtain services and applications from the IoT devices. However, these marketplace concepts are not specifically aimed at the W3C WoT applications and devices.

1.1 Problem Statement

For many communities, a dedicated platform is where new members of the community would come to. Even though code repositories like *GitHub*[2] can host projects, tutorials and more, they remain generic. Thus, specific platforms gain importance within communities, with package managers playing a similar role for software specific ones. It is possible to see platforms such as *Thingiverse*[3] for 3D Printer community, *Make:*[4] and *Instructables*[5] for the maker community, *hackster.io*[6] for hardware design community. Another example is the *Node-RED Library*[7] which supports over 3000 projects that are built specifically for the *Node-RED* programming tool. Moreover, package repositories and their associated command line interfaces as package managers became the de facto standard for installing, publishing and maintaining software

[1]The entire work of the working group can be found at https://www.w3.org/WoT/WG/

[2]https://github.com/
[3]http://thingiverse.com
[4]https://makezine.com/projects
[5]https://www.instructables.com
[6]https://www.hackster.io

projects, such as *npm*[7] for JavaScript, *pip*[8] for Python or *Maven*[9] for Java.

In the case of the W3C WoT, unfortunately there is no such platform that would enable people to find projects or anyone to publish projects in a W3C WoT compatible way, making it tedious to integrate various IoT devices into the WoT. After the publication of the Thing Description standard, there will be a need to welcome new members with a platform that consists of projects and the community interaction that is found in other platforms.

The Eclipse Thingweb project [8] aims to address some of these issues but there is currently no repository to group all the existing projects of the WoT community or to allow new projects to be published. There are numerous platforms and repositories but they are mostly package managers for a specific programming language and they are not based on WoT technologies like the TD. This makes it more difficult to convince how diverse the applications of the WoT are, hinders the further acceptance of the WoT and can result in existing IoT device owners to reinvent the wheel, which is the main reason why we often see siloed IoT solutions.

1.2 Contributions

We propose a platform called WoTify that is able to host W3C WoT projects provided by the community which can be used by anyone to turn their Internet connected devices into W3C WoT compatible devices, i.e. to WoTify them[10]. WoTify hosts project pages that can be composed of source code of any programming language or TD templates that can be used to describe already existing closed source and brownfield IoT devices with a TD. In the end, our platform allows anyone to search for W3C WoT projects or to contribute to the W3C WoT by sharing new implementations.

In particular, we propose:

(1) a currently running platform with already existing implementations, ready for the WoT community,
(2) a method to bring WoT functionality to any Internet connected device,
(3) a command line interface idea for the W3C WoT that is independent from a programming language.

We imagine WoTify to be the major WoT platform where even manufacturers can find a TD for their device that is provided by the community.

2 W3C WEB OF THINGS

The Web of Things is a set of design and programming norms that enable real world internet connected devices to be part of the World Wide Web. It started as an academic initiative to with the main goal of enabling device interoperability. It is based on the use of existing and widespread Web concepts, standards and protocols, such as REST, HTTP, CoAP and JSON, to enable inter device communication and device access.

For example, some Web of Things devices are the Philips Hue lights, even though they might not be marketed as such. They can be accessed and controlled using a RESTful API and standard communication protocols such as HTTPS. The problem with such devices is that discovering and understanding how to use their API is not a straightforward task. The Hue API is described in proprietary websites and datasheets. This requires developers to do more work

Figure 2: Photo of a Raspberry Pi with a Sense HAT attached on it[9].

to understand the proprietary API descriptions of each device they are trying to work with.

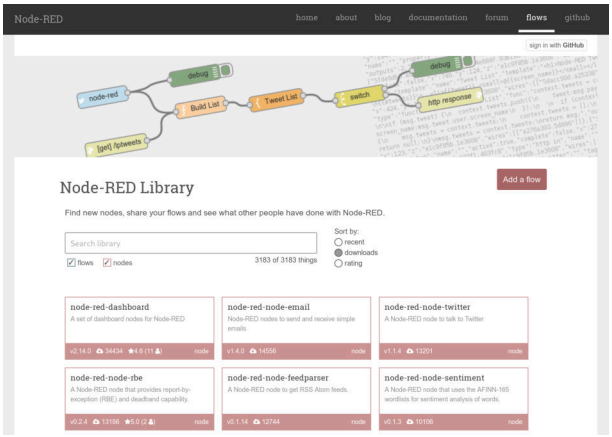
To solve this problem, the W3C started the WoT working group in 2016 to standardize a set of mechanisms for describing and working with Web of Things devices with the goal of enabling interoperability between devices, independent from the underlying framework and implementation. The main building blocks of the W3C WoT standardization effort are now being finished and include the WoT Thing Description[2] and WoT Architecture[10]. Of these two, the WoT Thing Description is the primary and most important building block, describing the public interface of a Thing. A Thing's TD provides a formal description of the functionalities of the Thing and how to use them. This removes the need to understand a multitude of non-standard, manufacturer specific API descriptions when integrating devices from different manufacturers. Another advantage provided by TDs is that they can be written for existing Web connected devices. An existing Thing can be made W3C WoT compatible simply by having a TD added to it.

Coming back to the Philips Hue light example discussed above, we can make it W3C WoT compatible simply be creating a TD to describe its API. This can be a simple translation of the API description provided on the Philips website to the standard TD format, making the Hue's API understandable to any developer familiar with the WoT TD standard. This can enable much quicker development times when integrating it with other WoT enabled devices from other manufacturers, as developers no longer need to navigate multiple manufacturer websites to fetch the API information for each device independently.

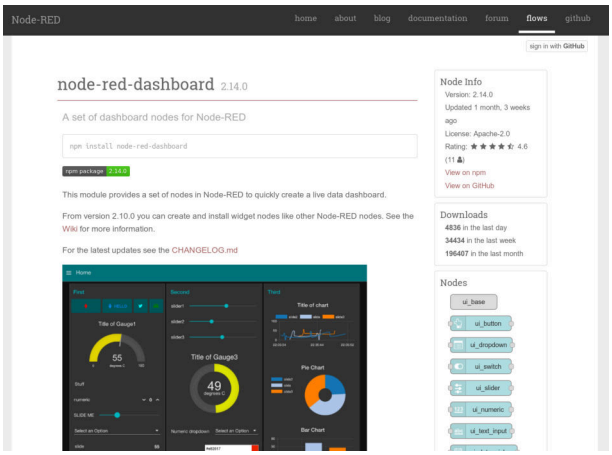
Similarly, any Internet connected device can be WoTified. WoTifying a device can be as trivial as simply writing a TD to describe an existing device's API or as complicated as extending a non-Internet-connected device with network interface to be remotely accessed (using an ESP8266 or a Raspberry Pi for example) and describing the resulting interface in a TD. Any device that can have a Web server with any Web protocol can thus be turned into a W3C WoT enabled device.

An example of a device that may be WoTified this way is the Raspberry Pi Sense HAT. It is a suite of sensors and an LED display that can be attached to a Raspberry Pi like in Figure 2. Out of the box, the Sense HAT is not an IoT device. However, it is possible to write a Web server that runs on the underlying Raspberry Pi and enable the functionality of Sense HAT as a Web service. Once

[7] http://npmjs.com
[8] http://pypi.org
[9] https://maven.apache.org/
[10] By WoTify, we mean two things: the process of integrating a device into WoT and the name of the platform that will help with that.



(a) The homepage of Node-RED Library, showing different contributions by the Node-RED community[7].



(b) A project page of Node-RED Library, containing information to integrate the project into the Node-RED programming tool[11].

Figure 3: Two screenshots of the Node-RED Library platform that can host different projects from contributors

this work has been done, and a TD describing the resulting API is created, anyone can install this Web server on their Sense HAT attached Raspberry Pi and have it automatically turned into a W3C WoT device.

3 CONCEPT

The idea behind WoTify is to have a platform available for the entire WoT community, composed of expert and new members alike. The community would contribute to WoTify by sharing their WoT implementations or by downloading what the other members of the community have already shared. Since the word *platform* is used in many different contexts, we want to explain what exactly WoTify can do in this section.

We think that Node-RED library is a very good example of a similar platform that has also influenced design decisions of WoTify. As seen in Figure 3a, Node-RED library allows anyone to contribute and to search for contributions. After choosing a contribution, i.e. a node, one would be met with a project page like in Figure 3b, that contains all the relevant information and statistics about the project.

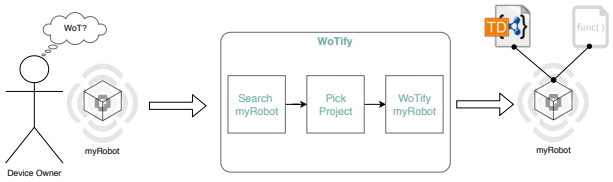


Figure 4: How to search for a project to turn an Internet connected device into a Web of Things device

Our aim with WoTify is to provide the same benefits but in the context of W3C WoT. This means, being able to search for WoT projects, download them and add new projects. However, the WoT is not a programming tool like Node-RED. The WoT is a programming language, tool or environment independent way to connect devices on the application layer and relies on standards from the W3C.

Since it relies on a set of standards, in order for a device to count as a W3C WoT device, it should have a TD representation. A W3C WoT device could be programmed in any programming language but its capabilities and services must be described with a TD. This means that WoTify must support TDs and be able to host projects programmed in any programming language. Furthermore, one can retrofit already existing IoT devices with a TD and turn them into W3C WoT devices as explained in Section 2. This means that WoTify should also support distribution of TDs of devices without the software that would run on the devices since already existing IoT devices may not allow a brand new software downloaded through WoTify to run on themselves.

In the end, WoTify aims to facilitate the two following aspects:

Using a WoT Project: Once one has a device to integrate into the WoT, may it be a barebone computer like a Raspberry Pi or an ESP8266 with some sensors or an off-the-shelf Philips Hue device, WoTify is the next step on integrating this device into the WoT. As shown in Figure 4, the device owner:

(1) searches for a WoT device project by providing the device name,
(2) chooses one of the results presented by WoTify based on programming language preference, complexity etc.
(3) if the project is a TD: Downloads the TD and start interacting with the device according to the WoT interaction patterns
(4) if the project is a software implementation: Reads the project page for installation instructions, install the project on the device and start interacting with the device according to the WoT interaction patterns
(5) gives feedback on the project.

Contributing to WoTify library: If one has a W3C WoT compatible device and its source code to share with the community, it can be done through WoTify. As shown in Figure 5, a W3C WoT device owner:

(1) goes to WoTify homepage to add a new project
(2) fills in the details of the project, such as a name, a TD template, platform such Raspberry Pi, Arduino, etc., topics such as sensor, lighting, robotics and source code repository link if the device can run 3rd party software.
(3) the project goes online, available for the community.

In the WoT community, there are already contributors with W3C WoT compatible devices as seen in the Plugfests at Face-to-Face meetings of the working group. These devices are built with the knowledge and expertise gained from the standardization process and would be considered as contributions to the community in general. With the WoTify becoming online, these could be some

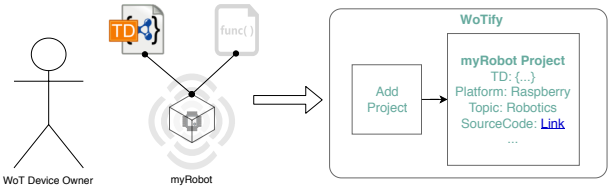


Figure 5: How to share the implementation of a W3C WoT device on WoTify

of the first contributions that further demonstrate the use case of WoTify.

3.1 WoTify Command Line Interface

Since the beginning of 2010s, package managers and command line interfaces (CLI) for programming languages have become the de-facto standard for installing different libraries. For example, we see *npm* for JavaScript, *pip* for Python, *Maven* for Java and even more when we consider other programming languages. Each of these tools can browse repositories, each containing more than 100.000 projects. Similarly, Linux distributions support package managers to install packages from different programming languages. In all the cases, these package managers and command line interfaces, together with the vast array of different packages and projects hosted in a repository become a very strong selling point, or even sometimes referred as the *killer app*[12].

With WoTify, we have also considered having such a CLI to enable easy installation of WoTify projects. This CLI would enable to run a command like `wotify install myRobot-flask` which would install the project called `myRobot-flask`, found at WoTify platform, on a Raspberry Pi computer. There are two main differences compared to existing package managers and CLIs when installing a project on a device with WoTify:

(1) The programming language can be different for each project. This results in the need to have a wrapper for the language of the project.
(2) The project might need to be installed on an external device. This results in requiring a wrapper for copying/flashing the source code to the target device. (Cross Compiling)

```
1 {
2   "name": "wot-mearmpi",
3   "version": "1.0.0",
4   "description": "W3C WoT interface for the MeArm Pi
        Robotic Arm",
5   "scripts": {
6     "install": "pip install -r requirements.txt"
7   }
8 }
```

Listing 1: package.json file used by *npm* to install a Python package

Given that most of the languages used by the current WoT community already have CLIs, package managers and repositories, we aim to leverage them and only create a wrapper around them. However, all the different languages require a prerequisite set of tools for the CLIs to work. For example, to install a package with *npm*, the *npm* software has to be installed, along with node.js development environment. This means that a command like `wotify install myRobot-flask` should check and install build environment, which depends on the underlying operating system.

For WoTify CLI, we are being inspired by the way *npm* handles different configurations. This way, WoTify would be similar to a tool that a significant number of the current WoT implementations are built with. As seen in line 6 of Listing 1, *npm* is able to rely on a set of predefined terms that can run any command. In this

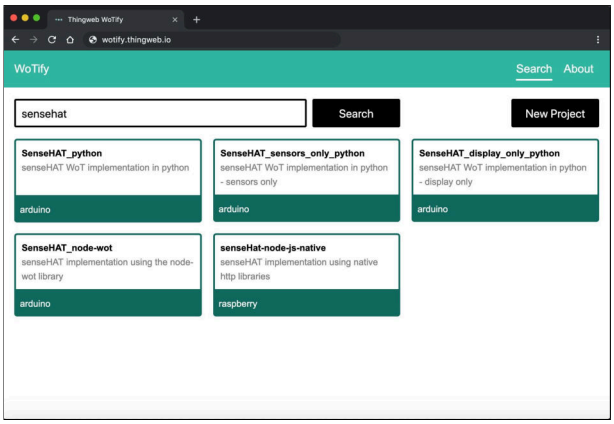


Figure 6: WoTify main page showing search results for Sense HAT related projects

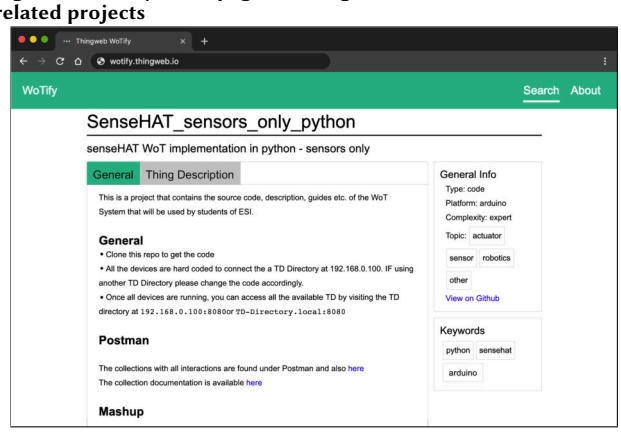


Figure 7: A project page on WoTify showing general information on how to use the project.

case, even though we are using *npm*, which is for the JavaScript language, we are installing a project in the Python language by overriding the `install` command on line 6. This is not the recommended use of *npm* but we want to use a similar approach in WoTify.

4 WOTIFY IMPLEMENTATION

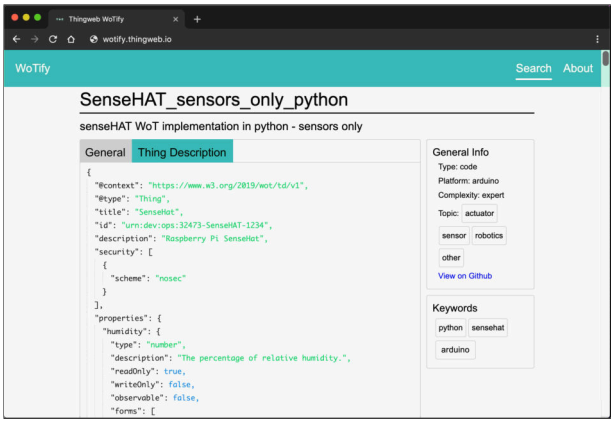


Figure 8: Project page on WoTify showing a Thing Description Template. This Thing Description can be downloaded in case of already existing non-W3C WoT devices and to enable interoperability with other W3C WoT devices.

The concept that we have introduced in Section 3 is implemented as a Web application that will be available on the Eclipse Thingweb Homepage[11]. It consists of a front-end Web user interface, a back end and a database, as illustrated in Figure 1. The WoTify front-end homepage allows querying and displaying available WoT projects, and allows users to contribute their own projects. The back end is connected to the front end and database and it handles queries, validates, retrieves and stores projects and organizes user authentication. The database stores all user projects with their attached data.

4.1 Searching for a WoT Project

As the main purposes of WoTify are the discovery of WoT projects and the contribution of these, the landing page offers searching existing projects and adding new projects. Users can search for a specific term, an implementation platform (e.g. Raspberry Pi), a topic or custom keywords. The results of an entered query will then be shown below the search bar of the homepage as shown in Figure 6. To better distinguish between projects that are TD templates and actual WoT implementations, the search results will be displayed in a different colors. When a queried project is selected by the user via clicking on it, a new page comprised of the project's details will be loaded as in Figure 7. Besides the standard information like name, short description and implementation type, the associated keywords, topics, level of complexity and link to Github repository can be also examined. However, the focus of the project page lies on the *General* and *Thing Description* area at the center:

- The *General* tab will display the content of the associated repository's main markdown file that is usually referred to as the *Readme* document. Contributors are strongly encouraged to describe the implementation in detail in this file and provide installation instructions and requirements. If no such file is available, the project's description is viewed instead.
- The *Thing Description* tab displays the project's TD file which will be viewed in JSON format as shown in Figure 8. If the projects is meant to WoTify an already existing IoT device, this TD will be used as the template.

4.2 Contributing to WoTify with a new Project

If a user wants to contribute to the WoTify platform, the *Add Project* button on the main page needs to be clicked. A form, with all the required inputs needed for a WoTify implementation will be displayed and needs to be filled out. This form can be seen in Figure 9. To fill out the Thing Description input, the user can either paste the raw content or upload a JSON file from the local machine. When all required fields are correctly filled and validated by the back end, a new project will be added to the system's database and can then be retrieved by other users.

The form we use to ask for the input relies on the WoTify implementation schema that can be seen in Listing 2.

Used Technologies: The back end is based on Node.js, the front end is implemented using the Vue.Js frontend framework. MongoDB was chosen as database, as it is based on the JSON format and therefore aligns with JSON based WoT technologies. These implementation choices allow our system to be highly extensible and scalable.

[11] http://www.thingweb.io/

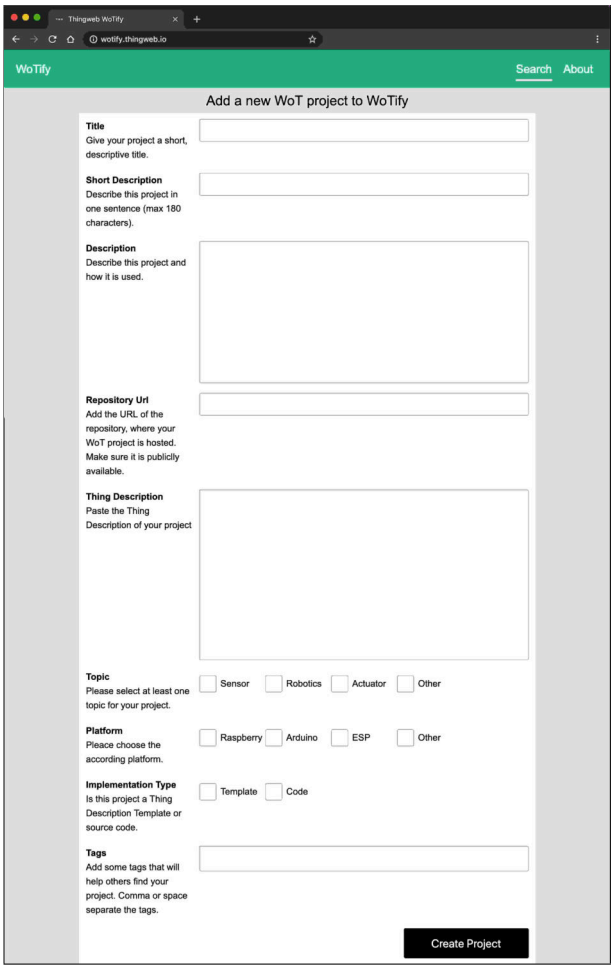


Figure 9: WoTify *Add Project* page showing what information one has to provide to share their project on WoTify

5 RELATED WORK

In recent years, various concepts have been developed to integrate different IoT offerings (e.g. services such as properties or actions from physical Things) into existing IoT ecosystems. The BIG IoT (Bridging the Interoperability Gap of the Internet of Things) project [12] developed a marketplace on which IoT offerings can be setup, discovered and used through simple integration into new IoT services or applications[5]. As a technical concept, the BIG IoT ecosystem is based on a so-called Offering Description (OD), which can be seen as a similar approach to the W3C Thing Description. WoTify can be also seen as a marketplace that offers (existing) WoT projects for the community. However, WoTify is independent of a specific IoT ecosystem such as BIG IoT and uses the upcoming standardized W3C WoT building blocks to make services or devices WoT enabled.

We also see Thingweb Directory[13] as an important work towards hosting WoT related projects. Thingweb Directory is designed to host TDs and offer semantic search to find TDs of (once) running devices. It is similar to WoTify, as WoTify can also host

[12] http://big-iot.eu/
[13] https://github.com/thingweb/thingweb-directory

```
{
 title: "WoTify JSON Schema for checking
      implementation inputs",
 type: "object","required: [...],
 properties: {
  name: {type: "string",minLength: 5},
  shortDescription: {
   type: "string",minLength: 5,maxLength: 180},
  longDescription: {
   type: "string",minLength: 5,maxLength: 500},
  github: {type: "string",format: "uri"},
  readme: {type: "string",format: "uri"},
  implementationType: {enum:["template", "code"]},
  topic": {
   type: "array",additionalItems: false,
   minItems: 1,uniqueItems: true,
   items: {
    enum: ["sensor", "actuator", "robotics", "
        lighting","other"]
   }
  },
  platform:{enum:["raspberry","arduino","ESP","other"
      ]},
  tags: {
   type:"array",additionalItems:false,minItems:1,
   uniqueItems: true,items: {type: "string"}
  },
  complexity:{enum:["simple","medium","expert"]},
  version: {
   type: "object",required: ["instance"],
   properties:{instance:{type:"string"}}
  },
  td:{type:object,properties:{...},required:[...]}
 }
}
```

Listing 2: The JSON Schema WoTify uses to describe and validate the project information

TDs, but the main difference is that WoTify is not limited to TDs and its aim is not to find TDs of other devices but to turn any device into a WoT device.

6 OUTLOOK

The task we have overtaken with WoTify is still in its infancy regarding the features it can offer based on the concept introduced in Section 3. We are planning to continue adding new features and shape it according to the feedback from the community. We have already seen the following features as the most needed in the short run:

- **WoTify CLI:** The WoTify CLI has been introduced in Section 3.1, but this part is not ready yet. Thus, in the current state of WoTify, developers need to install the WoTify projects according to the General Information page of the project.
- **TD Template Editor:** In the current state of WoTify, TD templates have to be edited manually after downloading them. This can be an error-prone process with errors resulting in a cumbersome debugging process since one would need to read the API description of the given device. An embedded TD editor that forces the developer to input only the relevant and correct information is necessary in the development of WoTify.
- **Rating System:** It is very important to stress the fact that WoTify projects that are downloaded will run on actual physical devices. Since we cannot run every WoTify project for each intended device, the community will need to be involved in a certain feedback mechanism to ensure the quality of the projects. In platforms like *npm*, this can be shown by weekly download counts or in Node-RED Library, this is done by users giving stars as rating. This feature would also help to show the support of the community, encouraging the adoption of W3C WoT.

Additionally, the work of the W3C WoT Testing Task Force can be used in the future to certify the correctness of the projects, strengthening the trust on projects. We can foresee tools like WoT Test Bench[14] integrated into WoTify CLI to run tests before publishing the project. This verification can be further enhanced by the contributors when they provide the recently introduced path descriptions[13]. The paths would also provide the developers a more constrained but safer way of using WoT devices.

7 CONCLUSION

In this paper, we have presented a new platform for W3C Web of Things projects that is already online for the WoT community. This platform is called WoTify and it is able to host WoT projects that can be used to WoTify existing devices, including brownfield IoT devices. Additionally, we have proposed a new type of command line interface for managing WoTify projects that is independent of programming languages. We believe that thanks to WoTify, the W3C WoT can be adopted by a wide array of developers, allowing them to quickly deploy the Web layer on various Internet connected devices and turn them into W3C WoT compatible devices.

REFERENCES

[1] 2018. *Stack Overflow Developer Survey 2018.* , Stack Overflow. https://insights.stackoverflow.com/survey/2018#developer-roles

[2] S. Kaebisch, T. Kamiya, Michael McCool, and Victor Charpenay. 2019. *Web of Things (WoT) Thing Description.* Candidate Recommendation, W3C. https://www.w3.org/TR/2019/CR-wot-thing-description-20190516/.

[3] Zoltan Kis, Kazuaki Nimura, Daniel Peintner, and Johannes Hund. 2018. *Web of Things (WoT) Scripting API.* . (nov 2018). https://www.w3.org/TR/2018/WD-wot-scripting-api-20181129/

[4] Raggett Dave, Ashimura Kazuyuki, and Chen Yingying. 2016. *White Paper for the Web of Things.* , W3C. http://w3c.github.io/wot/charters/wot-white-paper-2016.html

[5] Arne Bröring, Stefan Schmid, Corina-Kim Schindhelm, and Denis Kramer. 2017. Enabling IoT Ecosystems through Platform Interoperability The Problem of Missing IoT Interoperability. *IEEE Software* 34, 1, 54–61.

[6] Bhaskar Krishnamachari, Jerry Power, Seon Ho Kim, and Cyrus Shahabi. 2018. I3: an IoT marketplace for smart communities. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 498–499.

[7] JS Foundation. 2019. Node-RED Library. https://flows.nodered.org/?sort=downloads&num_pages=1 [Online; accessed April 21, 2019].

[8] Daniel Peintner, Matthias Kovatsch, Christian Glomb, Johannes Hund, Sebastian Kaebisch, Victor Charpenay. 2018. Eclipse Thingweb Project. https://projects.eclipse.org/projects/iot.thingweb [Online; accessed April 21, 2019].

[9] Raspberry Pi Foundation. 2019. Sense HAT - Raspberry Pi. https://www.raspberrypi.org/app/uploads/2017/05/Sense-HAT-plugged-in-1-1383x1080.jpg [Online; accessed April 21, 2019].

[10] M. Kovatsch, R. Matsukura, M. Lagally, T. Kawaguchi, K. Toumura, and K. Kajimoto. 2019. *Web of Things (WoT) Architecture.* Candidate Recommendation, W3C. https://www.w3.org/TR/2019/CR-wot-architecture-20190516/

[11] JS Foundation. 2019. node-red-dashboard. https://flows.nodered.org/node/node-red-dashboard [Online; accessed April 21, 2019].

[12] marmot. 2002. Re: Killer Apps in Perl? https://www.perlmonks.org/bare/?node_id=187498 [Online; accessed April 21, 2019].

[13] Ege Korkan, Sebastian Kaebisch, Matthias Kovatsch, and Sebastian Steinhorst. 2018. Sequential Behavioral Modeling for Scalable IoT Devices and Systems. In *2018 Forum on Specification & Design Languages (FDL).* 5–16.

[14] https://github.com/tum-ei-esi/testbench