Mozilla WebThings: An open source implementation of the Web of Things

Benjamin T. Francis
Emerging Technologies, Mozilla Corporation
bfrancis@mozilla.com

Abstract

Mozilla have implemented the WebThings Gateway (a software distribution for smart home gateways focused on privacy, security and interoperability) and the WebThings Framework (a collection of re-usable software components for building web things).

These implementations follow Mozilla's proposed Web Thing API which closely follows the latest Editor's Drafts from the W3C WoT Working Group, but offers some simplifications over the current working group deliverables.

Through these implementations Mozilla have demonstrated the power of the Web of Things to provide interoperabilty between otherwise incompatible smart home systems, and have identified some gaps in current web standards which could help move the Web of Things forward.

Keywords: Web of Things, Internet of Things, REST, HTTP, WebSockets, Node.js, smart home, gateway

1 Introduction

For the last two years Mozilla, the not-for-profit organisation known for the Firefox web browser, have been working on an open source implementation of the Web of Things now known as Mozilla WebThings.

This includes:

- **WebThings Gateway** - a software distribution for smart home gateways which allows users to directly monitor and control their home over the web without a middleman
- **WebThings Framework** – a collection of re-usable software components to help developers build their own web things which directly expose the Web Thing API

Mozilla's unofficial Web Thing API [Francis, 2019] specification documents the API implemented by Mozilla WebThings, which closely follows the W3C WoT Working Group's latest Editor's drafts, but with some differences.

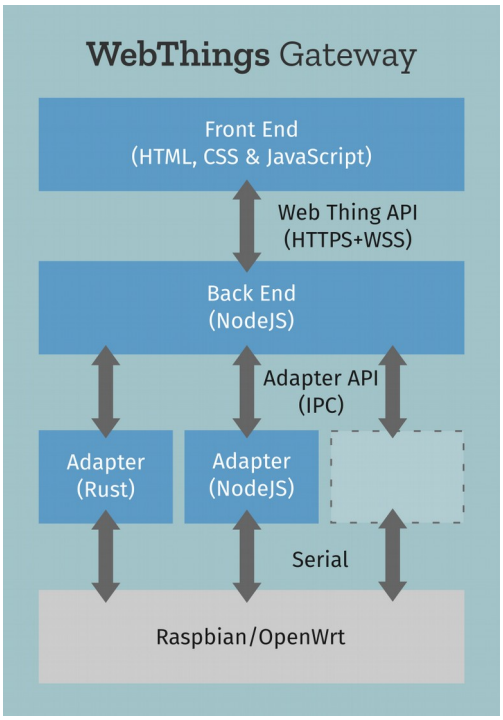
2 WebThings Gateway

The WebThings Gateway bridges a wide variety of popular smart home protocols (e.g. Zigbee, Z-Wave, Bluetooth, HomeKit, ONVIF) to a common Web Thing API (using JSON, HTTP & WebSockets) via an extensible adapter add-ons system, and hosts a unified web interface (an installable web application using a W3C web app manifest and Service Workers) which allows users to monitor, control and automate a variety of existing off-the-shelf smart home devices.

The gateway can be used on the local home network via a .local domain advertised by mDNS, or accessed remotely over the public Internet using a secure tunneling service provided by Mozilla. The tunneling service provides a dynamic, tunneled reverse proxy using PageKite and issues unique subdomains with automatically generated SSL certificates using LetsEncrypt. This allows the gateway's web interface to be safely accessed from outside the home using an end-to-end encrypted connection, without requiring users to open ports on their home firewall and without Mozilla having any access to private smart home data.

The gateway can either act as a proxy for web things which locally expose the Web Thing API (making them accessible via the public Internet over HTTPS) or bridge another smart home protocol to the Web Thing API with the use of an adapter add-on.

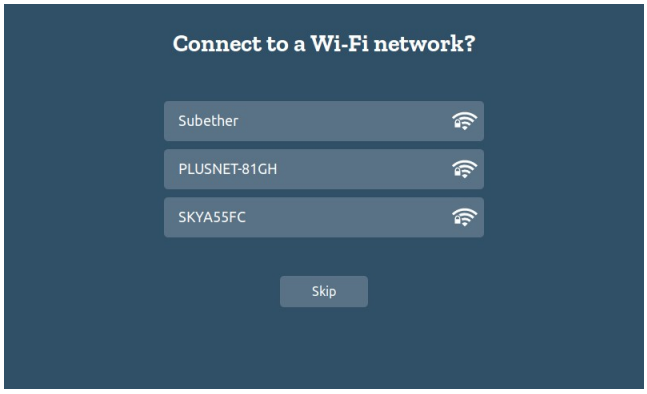
Adapter add-ons can be written in any programming language, run in their own system process and communicate with the main gateway process (written in Node.js) over IPC (interprocess communication) using Nanomsg. This means that if an individual adapter crashes, it won't bring down the main gateway process. The add-ons are packaged as npm packages and are installable via the gateway's web interface.



WebThings Gateway Architecture

WebThings Gateway is distributed as a pre-built software image designed to be used with a Raspberry Pi single board computer, or can optionally be built from source on Windows, MacOS or Linux. The Mozilla IoT team is also working an a WebThings Gateway software distribution based on OpenWrt, targeting consumer wireless routers.

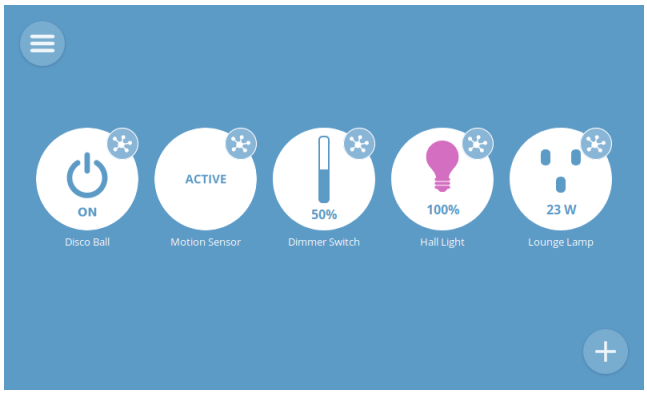
When first booted the gateway software acts a Wi-Fi access point which can be connected to from any desktop/laptop computer, tablet or smartphone and a web interface advertised via a captive portal guides the user through a first time setup process.



First time setup UI

First time setup includes connecting the gateway to an existing Wi-Fi network (or configuring a new one in the case of OpenWrt-based builds for routers), optionally registering a subdomain with Mozilla's tunneling service, and creating a first user account on the gateway. The user is then re-directed to the gateway's web interface via their unique subdomain or a .local domain advertised on the network via mDNS.

The gateway's web interface allows the user to connect smart home devices with the gateway using push-button commissioning (e.g. in the case of Zigbee and Z-Wave) or by scanning the local IP network (e.g. for mDNS broadcasts).

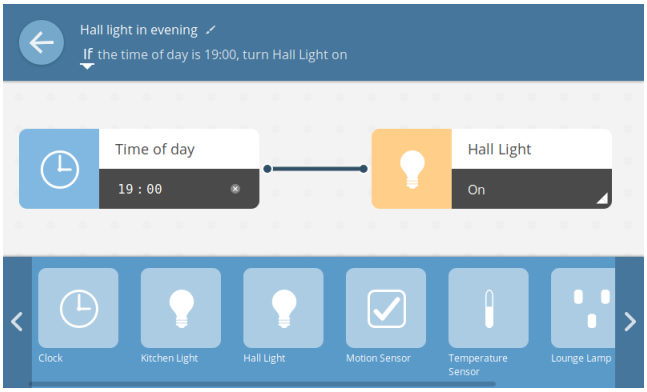


Things UI

Smart home devices are classified using a predefined but extensible set of "capability schemas" which define the kinds of properties, actions and events that devices might expose. These schemas are referenced using JSON-LD style semantic annotations in the JSON-based Web Thing Description, as an entry point the to Web Thing API.

The device and its properties and actions are then represented in a graphical web interface written in HTML, CSS and JavaScript. This representation is updated in real time and acts as a front end for the Web Thing API (using HTTP & WebSockets).

Other features of the gateway web interface include a drag-and-drop rules engine for defining "if this then that" style rules to automate the home, a floorplan view where the user can lay out their devices spatially on an interactive visual map of their home, a logging feature which can log and visualise data from smart home devices with interactive graphs and a smart assistant which accepts natural language commands using text or speech to control the home.



Rules Engine UI

The gateway also provides a mechanism for third party apps and services to request access to smart home devices using OAuth, where the user has full control over read or write access to individual devices. This creates the potential for an ecosystem of smart home apps and services designed to work with Web of Things gateways, where users remain in full control over their private smart home data.

3 WebThings Framework

The WebThings Framework provides web thing server implementations in a range of popular programming languages and frameworks, including Node.js, Python, Java, Rust, Arduino and MicroPython. These libraries are intended as example implementations of how to build a "native web thing", a device which directly exposes the Web Thing API using a built-in HTTP/WebSockets server.

Those devices can then be discovered by a Web of Things gateway or client, which can automatically detect the device's capabilities and monitor and control it over the web.

4 Web Thing API

The Web Thing API is an unofficial specification maintained by the Mozilla IoT team which documents the Web Thing Description format, Web Thing REST API and Web Thing WebSocket API used by Mozilla's WebThings Gateway and WebThings Framework.

The Web Thing Description section of the specification closely follows the W3C WoT Working Group's latest Editor's Draft of the Web of Things (WoT) Thing Description specification [Kaebisch et al., 2019] and has converged significantly with that specification over time. There are some remaining differences

(e.g. “links” are used instead of “forms” in several places).

The Web Thing REST API & Web Thing WebSocket API sections of the specification provide concrete Web of Things protocol bindings for HTTP and WebSockets respectively, as an alternative to the decarative approach proposed by the W3C WoT Working Group’s Web of Things (WoT) Protocol Binding Templates specification [Koster, 2019].

The specification does not include a scripting API equivalent to the W3C WoT Working Group’s proposed Web of Things (WoT) Scripting API specification [Kis et al., 2019]. It instead relies on existing client-side DOM API specifications including XMLHttpRequest [Kesteren et al., 2016], fetch [Kesteren et al., 2019] and WebSocket [Hickson, 2012] to communicate with the Web Thing API from web clients.

5 Lessons Learned

The WebThings Gateway implementation has demonstrated that it is possible to map a range of existing smart home application protocols such as Zigbee and Z-Wave to a common data model (defined by a Web Thing Description specification and an extensible system of capability schemas, serialised in JSON) and common REST & WebSockets API (using HTTP and WebSockets).

This has made it possible to integrate a range of existing smart home systems and off-the-the-shelf products from different manufacturers which were never designed to work together, by giving those devices URLs on the World Wide Web and using a standard data model and API to link them together.

The WebThings Framework implementation demonstrates how web thing servers can be written in a range of different programming languages, ranging from a more heavyweight Java implementation for platforms like Android Things, to more lightweight options like MicroPython and C++ for microcontroller platforms like Arduino.

Significant remaining challenges include:

1. HTTPS on local networks
2. Authentication/authorisation for web things
3. A more lightweight alternative to the HTTP protocol

SSL encryption on the web assumes an active Internet connection used to verify the authenticity of an SSL/TLS certificate. On a local network servers must rely on self-signed certificates which are not as secure and trigger warnings in web browsers. This makes it difficult to communicate securely with devices on a local network when a connection to the Internet is temporarily or permanently unavailable. There is now a W3C Community group looking at this issue.

The second challenge is the lack of standards for authenticating and authorising a Web of Things gateway or client to access a web thing.

The third challenge is that HTTP & WebSockets are a little too heavyweight for some low-powered microcontrollers common on the Internet of Things.

6 Future Work

In future the Mozilla IoT team will be exploring solutions for using HTTPS on local networks, a potential CoAP binding of the Web Thing API for resource constrained devices, and integrating our WebThings Gateway software into consumer wireless routers to address certain technical challenges and provide additional user value as a trusted agent for the whole home network.

7 References

Francis, B., 2019. *Web Thing API*. Mozilla. https://iot.mozilla.org/wot/

Kaebisch, S., Kamiya, T., McCool, M., Charpenay, V., 2019. *Web of Things (WoT) Thing Description*. W3C. https://w3c.github.io/wot-thing-description/

Koster, M., 2019. *Web of Things (WoT) Protocol Binding Templates*. W3C. https://w3c.github.io/wot-binding-templates/

Kis, Z., Nimura, K., Peintner, D., Hund, J., 2019. *Web of Things (WoT) Scripting API*. W3C. https://w3c.github.io/wot-scripting-api/

Kesteren, A., Aubourg, J., Song, J., 2016. XMLHttpRequest Level 1. W3C. https://www.w3.org/TR/XMLHttpRequest/

Kesteren, A., 2019. *Fetch – Living Standard*. WHATWG. https://fetch.spec.whatwg.org/

Hickson, I., 2012. *The WebSocket API*. W3C. https://www.w3.org/TR/websockets/