FOSS IoT Frameworks review

Benoit RENAULT

2017/08/10

Contents

1 Introduction

The context for this document is the need for an IoT server Gateway component for the OCCIware LinkedData Use Case. As the OCCIware project is FOSS[1], the component **must** be open-source itself. Therefore, what you will find here is a short state of the art of such frameworks, and no consideration shall be given to proprietary solutions.

Please note that, while the evaluation criteria may be considered timeless, the analyses presented below only reflect the state of the different projects as of this document's last update date, written on the first page.

2 Evaluation criteria

Each of the criteria is presented below with at least one argument as to how it allows to measure a given characteristic. The criteria are also chosen so that each solution needs not be analyzed more than one hour. For qualitative criteria, a little argumentation will be given in the detailed analysis (noted with excellent/good/average/poor/non-existent). Rational critics are, of course, welcome.

Some of them come from , and the ones about security have been inspired by Craig SMITH's "IoT Framework Assessment" OWASP page.

2.1 General criteria

Here are some general criteria that allow us to assess the quality of any FOSS project.

Project Health Determines how "mature" the project is, and its potential to last on the long term.

- First Release Date: The older, the better, might clue a position of pionneer, or a certain stability.
- Latest Release Date: The newer, the better, might clue that the project's release cycle is still active.
- Latest Commit Date: The newer, the better, if the project's release cycle is slow, might clue that the project is still ongoing active development.
- Number of main contributors: The higher, the better. If equal to one, exercise caution. Main contributors have proportional contributions to the code repository.
- Open issues ratio: The lower, the better. Is equal to the number of open issues over the total number of issues. Might clue at a good responsivity to user input, be it for bugs or improvements.

Company backing Partly shows how reliable the software can be: if there is a company ready to back it up and provide support to customers, and if there are other companies using the software, it might mean that it is trustworthy.

- Company Support: The better the company support is, the better the appreciation.
- Company Adoption: The more the software is adopted by other actors, the better the appreciation.

Documentation A well-documented project is a must.

- Currentness: The documentation must be up to date with the code. The more it respects this principle, the better the appreciation.
- Adaptability: Documentation must be adapted to the different users. The more it respects this principle, the better the appreciation.

[1] Free or Open-Source Software

UI The software must provide appropriate tools at all levels, be it CLIs or Graphical UIs, depending on the context.

- Server Management UI: the more powerful and concise, the better the appreciation.
- Sample applications: the more there are and the more complete, the better the appreciation.

2.2 IoT-specific criteria

Here are some general criteria that allow us to assess the quality of IoT projects in particular.

Security A key characteristic for IoT systems. As we cannot conduct a security assessment ourselves, we will rather look for a commitment to security and whether audits have been made or not. Security must be ensured at all levels: connectivity, storage, update, authentication, appropriate management of devices...

- Statements: the more is explained on the security measures taken in the documentation, the better the appreciation. Clues at a concern and understanding of the IoT problematics.
- Audits: the more positive reviews about the security are, the better the appreciation. Clues at a good applications of best security practices.

Connectivity/Flexibility

- Number of compatible hardware platforms: The higher, the better. Might clue to a shorter time-to-market.
- Number of supported protocols: The higher, the better. Might clue at a will not to lock the user in the system.
- Number of SDK implementations: The higher, the better. Might clue at an easier time for in-house developpers to incorporate the solution with their own prefered language.
- Modularity: the more the components of the solution can easily be changed (for instance, the DB), the better the appreciation.

2.3 Application-specific criteria

Here are some specific criteria that are related to the peculiar needs of the OCCIware project.

Arduino/NodeMCU compatibility Since it is the most popular platform for hardware experimentation, it is an important criteria that the service musts provide good support for it.

- Library: The more complete and recognized the library, the better the appreciation.
- Boilerplate code: The smaller the boilerplate code, the better the appreciation.

Java The framework must be based on Java for easy editing of the sources and better maintainability, since it is the main language of the author.

- Server percentage: The higher the quantity of Java code for the server, the better.

3 Detailed analysis

3.1 SiteWhere

Project Health

- First Release Date: The development started in 2010 as a closed source, asset tracking platform. The first open source release happened on **2014/03/06**. With 7 years of age, it can be assumed it is quite mature and stable.
- Latest Release Date: **2017/06/19** - Still releases new versions as of today.
- Latest Commit Date: **2017/06/20** - Work is still ongoing to deliver a new version.
- Number of main contributors: **1** - It seems to be mainly a one-man project, which is a problem. Some other people have contributed little bits of code, but not to the extent the main contributor has.
- Open issues ratio: **0.0534653465** - With 505 issues opened by many various actors (and not just the main contributor as milestones) since the project's arrival on Github, only 27 remain today, which is a very, very good. Excellent reactivity.

Good project health overall, though a shame that it seems to have only one main developer behind it.

Company backing

- Company Support: **excellent** - the project is backed by a well-established company (seven years of existence), and they offer a separate enterprise version with more features. Support is available to companies through the possibility to buy block hours for assistance, and also to anyone using the community edition through a rather active Google Group instance.
- Company Adoption: **good** - the corporate page of the project gives 2 comments by supposed customers, no success story using sitewhere was found but there were a few press articles talking about it like this one. It can however be infered that if the company has existed for 7 years, it means that they have found clients that appreciate their services.

Good company backing overall, though it's a shame they don't communicate a bit more about their customers.

Documentation

- Currentness: **good** - From the short time spent in the documentation, it seems it has been updated to match the latest release (as its number is written on the documentation homepage), however, in the absence of dating on all pages, it is impossible to say if everything inside has been kept up to date.
- Adaptability: **excellent** - The documentation is well separated between user and developper use cases. Explanations are clear, with lots of screenshots and appropriate commands where needed. The README.md on the Github Repository is crystal clear. Different levels of documentation are provided, like architecture, technologies, usage, code structure, ...

Good documentation overall, does make you want to use the product.

UI

- Server Management UI: **good** - Powerful and well-documented web administration interface, though its design is a little bit old school.
- Sample applications: **good** - There are example applications for ios, android and web, however, no screenshots are provided of those, making the evaluation of their quality difficult. Let's give the benefit of doubt.

The proposed UIs seem good overall, but it would take a deeper examination to fully conclude on their capacity.

Security

- Statements: **average** - There is no open commitment to a secure system, on none of their websites. No explanation of the security measures they have taken, except that they use the Spring Frameworks integrated capabilities (which is good). No mention about the cryptographic means they use either.
- Audits: **non-existent** - No mention on the product website, neither found any through a quick search on the web.

There is no apparent concern about security on their websites, which is certainly not good. Furthermore, there is apparently an SSL configuration problem with their corporate website, which is not a good sign.

Connectivity/Flexibility

- Number of compatible hardware platforms: **4** - Android, Arduino, Raspberry Pi, and Generic Java-capable board (Cf. dedicated documentation page). Portentially highly compatible with many other platforms, but since no detail is given, we may assume with some reserves that not so many have actually been tested.
- Number of supported protocols: **8** - MQTT, AMQP, OpenWire, XMPP, HTTP REST requests, WebSocket, Hazelcast, Stomp (Cf. dedicated documentation page).
- Number of SDK implementations: **2** - Android and IOS.
- Modularity: **good** - Design built out of preexisting opensource components. DB apparently easily changeable between MongoDB and Apache HBase (Cf. dedicated documentation page)

Overall, many connectors available, though seem to be a little weak on the hardware side, and lack of a SdK for Desktop applications.

Arduino/NodeMCU compatibility

- Library: **good** - Quite a lot of work seems to have been put into arduino compatibility, and the library seems very complete (with several example sketches included). However, no particular documentation about the NodeMCU/ESP8266.
- Boilerplate code: **good** - The event/publish/subscribe structure, while being very efficient, requires here quite a lot of boilerplate code just to send a little bit of data.

Quite good Arduino compatibility, but it would really be welcome to have some documentation on its usage with NodeMCU/ESP8266 specifically.

Java

- Server percentage: 81.8% - The server is mainly just plain good old Java. Really good !

3.2 Kaa

3.3 Device Hive

3.4 Zetta JS

3.5 ThingSpeak

3.6 Parse

3.7 DSA

3.8 Blynk

3.9 Paho

3.10 Kura

3.11 Kapua

3.12 Hono

4 Summarized analysis

5 Conclusion

- Annexes -