applied sciences

MDPI

*Review*

# Internet of Things Platforms for Academic Research and Development: A Critical Review

**Luca De Nardis [1,*], Alireza Mohammadpour [2], Giuseppe Caso [3], Usman Ali [1] and Maria-Gabriella Di Benedetto [1]**

[1] Department of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome, Via Eudossiana 18, 00184 Rome, Italy; usman.ali@uniroma1.it (U.A.); mariagabriella.dibenedetto@uniroma1.it (M.-G.D.B.)
[2] Smart and Secure Networks (S2N) National Laboratory, National Inter-University Consortium for Telecommunications (CNIT), 16129 Genoa, Italy; alireza.mohammadpour@tnt-lab.unige.it
[3] Ericsson Research, Radio Systems and Standards, Ericsson AB, 16440 Kista, Sweden; giuseppe.caso@ericsson.com
* Correspondence: luca.denardis@uniroma1.it; Tel.: +39-06-4458-5479

**Abstract:** Tens of different IoT platforms are currently available on the market as a result of the high interest in IoT, characterized by very different characteristics in terms of utilization models, features and availability. This paper provides a review of existing platforms, both adopting a closed source and an open source access model, focusing on five evaluation criteria: communication protocols, data visualization, data processing, integration with external services and security. Afterward, the paper focuses on ten open source platforms, that are deemed more suitable for research and development activities in academia, and provides an evaluation of such platforms according to the five criteria previously defined, combined with two criteria specific to open source platforms: installation procedure and documentation. The evaluation indicates that the FIWARE platform is the best suited platform when taking into account the combination of the seven criteria; other platforms might, however, be preferred, depending on the context, thanks to specific features such as native support for a programming language, or ease and flexibility in the installation procedure.

**Keywords:** IoT; open source; research and development



**Citation:** De Nardis, L.; Mohammadpour, A.; Caso, G.; Ali, U.; Di Benedetto, M.-G. Internet of Things Platforms for Academic Research and Development: A Critical Review. *Appl. Sci.* **2022**, *12*, 2172. https://doi.org/10.3390/app12042172

Academic Editors: Gianni Pantaleo and Pierfrancesco Bellini

Received: 13 January 2022
Accepted: 15 February 2022
Published: 19 February 2022



## 1. Introduction

Internet of Things (IoT) is one of the fastest growing markets in the telecommunications industry. Recent reports estimated an IoT technology market of about 384.5 billion dollars in 2021, expected to reach a figure ranging between 600 [1] and 1800 [2] Billion dollars by 2027–2028, with over 40 Billion devices connected worldwide [3]; the market growth will encompass both consumer applications and industry business-to-business (B2B) applications, with a growing interest in the so-called Industrial Internet of Things (IIoT) [4]. The growing interest in IoT applications and services in both consumer and industrial markets led, in turn, to the introduction of a plethora of IoT hardware and software elements, spanning an extremely wide range of architectures, features, and business models. According to [5], five key building blocks can be identified in IoT: (1) applications, (2) software back-ends, (3) communications, (4) hardware, and (5) security: an IoT software platform (in the following: IoT platform) should support the functions carried out by each of these blocks, and enable their interaction. A market study in 2015 identified over 300 IoT platforms providing support for IoT to some extent [5], while more recent estimates place this number beyond 450 [6].

Given such a wide availability, the selection of the correct IoT platform, which is a key preliminary step in the design and deployment of an IoT application, can be a daunting task. As a consequence, several works were recently published that identify a set of key

characteristics defining an IoT platform and/or review existing IoT platforms, aiming at providing a tool for the selection of the platform that best suits a specific application.

In [6] a classification of software platforms based on the following five categories is proposed:

- IoT application enablement platforms, which are identified in [6] as the only proper IoT platforms;
- connectivity/machine-to-machine (M2M) platforms, mainly focusing on connectivity, rather than data storage and processing;
- infrastructure-as-a-service backends, focusing on hosting and processing services, both for general purpose applications and IoT applications;
- hardware-specific software platforms, typically developed by hardware companies in order to support their own products; and
- consumer/enterprise software extensions, introduced in software and operative systems predating the IoT era in order to integrate IoT devices.

In the same work, eight building blocks that an IoT platform should include are also identified as follows: external interfaces, analytics, data visualization, processing and action management, device management, connectivity and (data) normalization, database and additional tools (e.g., app prototyping and reporting tools). The work does not map however existing IoT platforms neither on the proposed classification nor on the proposed set of building blocks.

In [7] a review of both hardware IoT devices and IoT platforms is carried out. Nine IoT devices are compared in terms of computing capabilities, development environments and supported operative systems, and connectivity features; no ranking among the devices is however provided, suggesting that the choice should be based on the specific target IoT applications. Regarding IoT platforms, Singh and Kapoor [7] adopt, with minor modifications, the classification categories and the IoT platform building blocks proposed in [6], and considers 11 different IoT platforms; the platforms are however not classified according to the 5 categories proposed in [6], and furthermore the comparison between platforms, carried out taking into account only a subset of the 8 building blocks, does not result in a critical ranking of the platforms. The paper provides, however, an interesting comparative insight in the capabilities of the considered platforms.

A key issue in comparing different IoT systems is their inherent heterogeneity: different companies adopt different reference architectures, making a direct comparison between IoT systems difficult or even impossible. The work in [8] attempts to address this issue by proposing a reference architecture for an IoT system organized into four entities: a device, connected to sensors and actuators by means of software drivers, a gateway in charge of communications, an IoT integration middleware corresponding to the IoT platform, and an application. The paper then maps four existing IoT systems on such architecture, in order to assess whether it provides the flexibility to describe different IoT systems. Although interesting, the approach does not yet provide an objective way to compare different platforms.

In the above framework, this work provides two main contributions, starting from the broad context of IoT platforms and then progressively narrowing its focus, as follows:

1. It extends previous work on the review of IoT platforms by analyzing over 20 platforms according to to a set of five evaluation criteria, organizing them in closed source vs. open source ones.
2. It goes beyond the mere list of features for each platform typically provided in previous works, by focusing on academic research and development (R and D) activities, and providing a guide to the selection of the best open source IoT platform in this specific context. To this aim, a ranking of open source platforms is introduced, for each of the five criteria adopted in the review and for two additional criteria specific to open source platforms, in order to support researchers in determining which platform is best suited for a given research activity, based on the relevance of each criterion.

The goal of this work is not to determine the best open source platform, as this will require specific knowledge on the key elements of the considered research problem. Rather, its goal is to support researchers in the time-consuming and often frustrating exploration of available platforms in order to identify the platform best suited to address the research problem under consideration.

The paper is organized as follows. Section 2 introduces a set of evaluation criteria, and reviews closed source vs. open source IoT platforms according to such criteria. Next Section 3 compares open source platforms platform from the point of view of academic research, while Section 4 draws conclusions.

## 2. Internet of Things Platforms

When addressing a problem within the framework of IoT requiring the introduction of a platform, the first choice to be faced is whether to use and/or adapt an existing platform or to build a new platform from scratch. The latter choice can be preferred, in particular, for problems with very specific requirements, not addressable by existing platforms, or when the goal is to introduce a new functionality not available in any existing platform. Several examples of platforms designed from scratch for a specific target application can be found in the literature. In [9] a platform specifically designed for railways logistics is introduced and compared with existing logistics software. An IoT platform model specifically designed to take advantage of edge/fog computing is proposed in [10], as the basis for the definition of a new IoT platform, named SAT-IoT. In [11] a platform taking advantage of blockchain technology for efficient decentralized access to edge nodes, called Edgence, is proposed. In most cases, however, building a new platform is an excessive and unnecessary effort, typically when the research goal is to develop an application with requirements that are estimated to be within the scope of existing platforms. If this is the case, the most natural choice is to use an existing platform, posing, thus, the issue of how to select the best platform for the problem at hand.

A plethora of IoT platforms with very different business and utilization models are in fact currently available, ranging from paid access to service with no source code access on one extreme, to free service and full access to the source code [7] on the other. The distinction of open source vs. closed source platforms is particularly relevant from the point of view of academic R and D, and was thus adopted as the first discerning feature in organizing the review carried out in the following.

Regardless of the availability of the source code, a common set of criteria was adopted to analyze IoT platforms. Based on several previous reviews [6,7,12], the following criteria were adopted:

- communication protocols, related to to the set of protocols made available to communicate with devices and external systems and, correspondingly, to the number of supported hardware platforms;
- data processing capabilities, related to tools for data processing, transformation and analysis;
- data visualization, related to availability of tools for presenting data collected from devices and analyzed in the platform to the external world;
- integration with external services, related to support for interconnection with external storage and processing services, such as databases and content management systems;
- security, related to the security features made available by the platform, aiming at limiting the number of data breaches and mitigating the impact of each breach in ecosystems that can include millions of connected devices.

The list above does not exhaust the possible comparison criteria for IoT platforms. Features such as processing power and corresponding speed, reaction time, storage capacity and reliability are all potentially valuable metrics in comparing IoT platforms; unfortunately it is extremely hard to obtain information on these metrics from platforms' documentation and websites. Furthermore, these metrics mostly depend on the amount of computing, networking and storage resources that are made available in the deployment of an IoT plat-

form. In turn, using these metrics when comparing the closed source platforms analyzed in Section 2.1, typically deployed in the cloud, would most often translate in a comparison between pricing tiers (see for example the case for available storage) rather than between inherent properties of the platforms.

An assessment based on these metrics would be even harder to perform for open source platforms that can be installed locally. Even when differences might exist (e.g., shorter response time due to a better software implementation of APIs) they would be very hard to quantitatively measure. For this reason, these criteria were not included in the set adopted in this work.

In the following, the most relevant platforms falling in both closed source and open source categories are reviewed, and for each category a table analyzing a wide range of platforms according to the criteria defined above is provided.

### 2.1. Closed Source Platforms

Most major players in IT services developed their own IoT platform, in most cases as an extension or integration of their preexisting cloud services, leading to solutions that fall in general under the umbrella known as software-as-a-service (SaaS) or platform-as-a-service (PaaS).

The PaaS approach is adopted by Ericsson for its IoT accelerator platform [13], that targets mostly business partners by proposing a platform that takes care of connectivity (leveraging the expertise of the company in mobile communications), security aspects, application programming interfaces (APIs) for business management support and user authentication/subscription/billing. The platform can integrate and manage all devices that support any of representational state transfer (REST), advanced message queuing protocol (AMQP), constrained application protocol (CoAP), and are compliant to LightWeightM2M (LWM2M) specifications. In terms of integration, the platform supports enterprise service buses for integration with enterprise IT back-end systems.

Autodesk Fusion Connect [14] adopted a SaaS approach, providing a cloud-based internet of things (IoT) platform that enables bidirectional connection with field devices, regardless of whether they were designed for IP connectivity or not. It provided the capability to monitor, analyze, and manage these devices allowing a dynamic duty cycle scheduling, aiming at uptime maximization and optimal asset usage. The platform is, however, not listed anymore in the Autodesk product list, suggesting it has been discontinued.

The IBM Watson IoT Platform [15] is another platform proposed by a major IT player. In this case as well, the platform adopts a PaaS approach, supporting devices that adopt REST, hyper text transfer protocol (HTTP) and message queuing telemetry transport (MQTT) protocols, among others, and providing powerful analytics, aggregation and visualization tools. The platform also makes available an editor for IoT apps, Node-RED, that allows the creation of IoT apps connecting hardware and cloud services.

The Zoho IoT platform [16], formerly known as WebNMS [17], provides a cloud-based platform aiming at businesses in fields like energy management and remote asset tracking. Zoho promises device management and monitoring, data processing, support for legacy devices and cross enterprise operations.

The Losant Enterprise IoT Platform [18] offers the possibility of building IoT solutions with minimum coding, offering public cloud access under a free pricing scheme, and private cloud and on-premises installations for enterprise customers. The platform targets ease of use and deployment of IoT applications, by means of application templates that can be customized to fit specific scenarios. The platform also supports edge-computing to prefilter data to be sent to the cloud for processing and visualization.

The Ubidots Platform [19] is an IoT cloud platform supporting HTTP, MQTT, TCP and UDP protocols, and a wide range of IoT hardware platforms and cellular technologies by means of dedicated SDKs. The platform is available under a paid PaaS scheme, although a non-commercial, free tier with support of up to three devices is available.

MindSphere [20] is also a PaaS proposal by Siemens, providing a set of APIs for interconnection of devices supporting HTTP and the JavaScript object notation (JSON) data exchange format; a wider range of protocols, including REST and MQTT, are supported through the MindConnect IoT Extension, allowing a wide range of devices to connect to the platform. Similarly to IoT Accelerator and the IBM Watson IoT Platform, the targets of MindSphere are mainly business customers, and the platform provides indeed support for integration with industrial back-ends such as supervisory control and data acquisition (SCADA) systems.

General Electric also proposed its IoT platform, named Predix [21], that operates by combining two technologies for cloud and edge computing and communication, each with its own software stack. Cloud and Edge stacks cooperate in optimizing the platform for each specific application. In terms of communication with IoT devices, Predix supports the open platform communications-unified architecture (OPC-UA), MQTT and Modbus transport control protocol (TCP), among others. The platform is particularly well suited for data analytics, thanks to the availability of machine learning models, and to the possibility of deploying analytics algorithms both at the edge or in the cloud of the platform.

Cisco is another major IT player with a strong interest in IoT, as witnessed by their Edge Intelligence platform, formerly known as Kinetic [22], that also supports edge and fog computing for distributed networks, as well a traditional device-to-cloud communication scenario. Devices send data using messages, in formats including text, binary or JSON, ensuring thus a wide device support. The platform also supports integration with external databases for data storage and processing.

Finally, Google also have their proposal for IoT, named Google Cloud IoT [23], that is organized in three basic components: device, gateway, and cloud. The device component includes every hardware or software that directly interacts with the world. A gateway is a component that allows devices that do not have internet access to reach the cloud, and which may also perform data processing on the data collected from connected devices. Finally, the cloud component is the centralized software platform where data received from gateways or devices is collected, combined and analyzed, and transferred for storage or publishing. MQTT and HTPP communication protocols are supported for data uploading by devices, and end-to-end security is provided by means of the transport layer security (TLS) security protocol. The review above shows that the above closed source platforms from major IT players share a common approach, focused on business customers interested in a SaaS/PaaS solution, rather than in doing in-house development of new features, although such development is sometimes possible in the platforms. Many other platforms not described above share the SaaS/PaaS approach; a comprehensive list is provided in Table 1, analyzing the platforms described above and several others in terms of the set of criteria defined in Section 2.

### 2.2. Open Source Platforms

Compared to the extremely wide range of closed source platforms, the number of open source ones is definitely smaller; several interesting options can however be identified. The most relevant ones are described below. Note that in order to provide support for the in-depth comparison of open source platforms presented in Section 3, the analysis of open source platforms was extended beyond the information available from platforms websites and documentation and in the reviews and surveys introduced in Section 1, by performing both an analysis of the source code and a local installation of each platform on a Linux machine. This allowed to assess two additional aspects specific to open source platforms beyond those already introduced in Section 1, that are installation procedure and documentation.

**Table 1.** Comparison of closed source platforms.

| Platform | Comm. Protocols | Data Processing | Data Visualization | Integration | Security |
|---|---|---|---|---|---|
| ALTAIR SmartWorks [24] | API, MQTT | Real-time visualization, stream processing | Dashboard | API, MQTT | Enforce oversight using fine-grained access control. Easily integrate with existing enterprise user management system |
| Appcelerator [25] (to be discontinued in 2022 and released under an open source license) | MQTT, HTTP | real-time analytics (Titanium) | yes (Titanium UI Dashboard) | REST API | link encryption (SSL, IPsec, AES-256) |
| Autodesk Fusion Connect [14] | API | drag and drop analytic engine | user interface creation tools | M2M/IoT protocol and data format adapters: CoAP, UDP, HTTP, Modbus, XMPP, DDS, MQTT and vendor-specific M2M technologies | support all possible device authorization, authentication, device data encoding, data transfer securing, such as HTTPS |
| AWS IoT [26] | MQTT, HTTP, LoRaWAN | real-time analytics (Rules Engine, Amazon Kinesis, AWS Lambda) | yes (AWS IoT Dashboard) | REST API | link encryption (TLS), authentication (SigV4, X.509) |
| Bosch IoT Suite—MDM IoT Platform [27] | MQTT, CoAP, AMQP, streaming text-oriented messaging protocol (STOMP) | unknown | yes (user interface integrator) | REST API | HTTPS |
| C3 Platform [28] | HTTPS, MQTT, SFTP, AWS Kinesis | real-time analytics (stream processing, batch processing, iterative processing), data science platform and tools | connectivity to relational databases, applications | pre-built and extensible industry canonical models, RESTful federated in an image, AWS, Oracle and Apache DBs, Azure, JSON, application connectors | single sign-on via SAML 2.0, two-factor authentication support, OAuth 2.0 support, hosting model (virtual private clouds, implements a rigorous cyber security program), role-based access control security framework |
| EVRYTHING—IoT Smart Products Platform [29] | MQTT, CoAP, WebSockets | real-time analytics (rules engine) | yes (EVRYTHING IoT Dashboard) | REST API | link encryption (SSL) |

**Table 1.** *Cont.*

| Platform | Comm. Protocols | Data Processing | Data Visualization | Integration | Security |
|---|---|---|---|---|---|
| ThingWorx—MDM IoT Platform [30] | MQTT, AMQP, XMPP, CoAP, DDS, WebSockets | predictive analytics (ThingWorx machine learning), real-time analytics (ParStream DB) | ThingWorx SQUEAL | REST API | standards (ISO 27001), identity management (LDAP) |
| IoT Accelerator (Ericsson IoT) [13] | AMQP, CoAP, compliant to LWM2M specifications, 3PP device and data management systems | real-time, orchestration and integration and automation framework, monetization engine | third-party applications, developer portal, composer-rapid application developer | REST API | policy-based security orchestration, proactive auditing, monitoring, and data integrity secured by industrialized blockchain; link encryption (SSL/TSL), authentication (SIM based) |
| Wyliodrin [31] | HTTP, MQTT | real-time analytics in dashboard and supports R programming | dashboard, mobile app, Google Chrome app | Wyliodrin API, JSON | secure deployment and update infrastructure |
| IBM Watson IoT Platform [15] | Device-WISE, Watson IoT SLA (cloud services), MQTT v5, HTTPS | real-time, analytics service add-on; AI and machine learning models, edge analytics | cloud analytics services (web portal) | API, Node-RED | blockchain, private cloud-ready, link encryption (TLS), authentication (IBM Cloud SSO), identity management (LDAP) |
| LOSANT Platform [18] | MQTT, API | Jupyter notebooks for batch processing of historical data and deep analytics | Losant notebook.Visualize information using Losant graphs, maps, and logs | MQTT, API | TLS encryption protocols, fully revocable access keys for each device |
| MindSphere (Siemens) [20] | MindConnect APIs natively, MQTT and other protocols through MindConnect IoT Extension | real-time (using MindAccess IoT Value Plan and MindConnect applications) | MindConnect Nano and Fleet Manager and mindSphere applications | historian databases, enterprise resourcing planning (ERP), manufacturing execution system (MES), SCADA systems, Amazon Web Services (AWS) and Microsoft Azure public cloud | unknown |
| GE PREDIX [21] | OPC-UA, Modbus TCP and MQTT protocols and can be extended using an SDK to support proprietary ones + external (location, weather) systems | Predix Edge Manager, low-latency applications at the edge + analytics in the cloud | industrial applications, e.g., Predix APM Services to build browser-based and mobile device user interfaces | SCADA, Historian, IT (CRM, ERP) | Embedded cybersecurity; identity and access management, data security, application security, network security, audit/reporting, logging |

**Table 1.** *Cont.*

| Platform | Comm. Protocols | Data Processing | Data Visualization | Integration | Security |
|---|---|---|---|---|---|
| Cisco Edge Intelligence [22] | TCP, HTTP, WebSockets, Unix Domain Sockets | Real time GPS, apps, move data and compute data (DCM) modules, ad-hoc and embedded analytics | dataflow editor, third-party applications | APIs , binary, text, JSON, MsgPack, JSON, Kafka, RDBMS, ParStream, and Hadoop-based repositories | encrypted IPSEC tunnel between gateways and the Cisco Kinetic GMM cloud, certificate-based authentication, Flex VPN to establish IPSEC tunnels over IKEv2, support AnyConnect VPN and port forwarding |
| Google Cloud IoT [23] | secure MQTT, HTTP | pipeline processing, rule processing, Google cloud functions, BigQuery and Cloud Data lab, Google machine learning engine | Google Cloud Dataflow Cloud Bigtable | API, Cloud Pub/Sub | secure MQTT |
| Microsoft Azure IoT Suite [32] | HTTP, HTTPS, MQTT | Azure Stream Analytics manager microservice | Azure Stream Analytics, Amazon QuickSight, Jupyter notebooks, and applications (IoT Things Graph, etc.) | device simulation microservice (RESTful), Amazon MQ, SNS, SQS, SWF, JSON | authentication and authorization microservices, blockchain |
| Ubidots Platform [19] | HTTP, MQTT, TCP, UDP, API | analytic engine | dashboard | HTTP, MQTT, TCP, UDP, API | authorization event engine |
| Zoho IoT platform [16] | Described as multi-device/multi-protocol, no additional details provided | cloud-based real-time data processing | cloud-based data visualization with widgets and dashboards | Information not available | encrypted HTTPS and MQTT for data transfer, access management |

The Eclipse Kapua platform [33] is a modular platform focused on the integration of IoT devices available on the Eclipse IoT community [34]. The platform offers device connectivity by means of a multi-protocol message broker supporting the MQTT protocol toward the IoT devices and the AMQP and WebSockets protocols toward applications. The platform also provides device management by means of an open application protocol running over MQTT, that allows to control, configure, start and stop devices remotely from applications; support for additional protocols such as LWM2M is currently under development. Data management is a third feature offered by the Kapua platform: persistent storage is provided, allowing storing data waiting for retrieval by applications; the platform does not provide, however, any data processing or visualization capability. Finally, security is guaranteed by supporting both access control, through user identities that can be defined in the platform, and SSL authentication.

The platform can be deployed using docker images with a script provided as part of the download package on the platform website; installation from the source code can be performed using the Maven build tool [35]. The documentation includes both the Github repository doc files and the developer and user guides on the project website, with links to detailed description of platform APIs.

FIWARE [36] is a European Union-funded effort to develop an open-source solution for IoT. FIWARE provides a set of open-source platform components that can be combined to build an IoT platform matching a specific application context. The components, referred to as "Generic Enablers", can be either directly developed within FIWARE or result from the incubation of software projects developed externally from the project. "Generic Enablers" fall under several categories, that can be divided in five groups.

The fist group is the "Core Context Management" group, and includes the core category: the central element in a FIWARE-compliant platform, the context broker generic enabler, falls into this category. The context broker provides an implementation of the FIWARE API, that enables the interactions with all the other components in an IoT platform: the execution of queries from sensors, the issue of commands to actuators, and more in general the exchange and management of IoT context information between devices and applications, are made possible by the context broker. The context broker originally developed within FIWARE is the Orion context broker, that provides the next-generation service interfaces (NGSI) v2 API. Several additional context brokers, available by incubation, introduced a new API defined according to the ETSI next-generation service interfaces—linked data (NGSI-LD) standard [37].

The second group is the "Context, Processing, Analysis, Visualization" group, and includes generic enablers falling under three categories: processing and analysis, visualization and media streams. The processing and analysis category currently includes the FogFlow generic enabler for distributed processing over both edge and cloud architectures, and several additional incubated enablers. The visualization category comprises the WireCloud tool, providing an easy to use interface to create widgets, dashboards and cockpits to visualize data. Finally, the media streams category includes media servers and clients to analyze and enhance media streams, such as the Kurento generic enabler.

The third group is the "Interface with IoT, robots and third-party systems" group, and includes categories of generic enablers related to the interface between a FIWARE platform and hardware devices, such as sensors and actuators. The group includes two categories of generic enablers: IoT agents and robotics. IoT-agents generic enablers provide bridges between the NGSI and the communication protocols used by IoT devices, such as MQTT, LightWeight M2M, SigFox, OPC-UA and LoRaWAN. Robotics generic enablers play a similar role with respect to robotic systems based on popular operative systems, such as the Robot Operating System 2 (ROS2).

The fourth group, "Context Data and API management, publication and monetization", comprises generic enablers divided in the following categories: API Management, data publication, data monetization and security. Generic enablers in this group focus thus on providing secure access to data collected and generated in a FIWARE platform. Security in particular is guaranteed by OAuth2-based authentication schemas provided by the Keyrock Identity Management generic enabler and the Wilma proxy generic enabler, and by access control compliant to the extensible access control markup language (XACML) standard, supplied by the AuthZForce PDP/PAP generic enabler.

The last group, "Deployment tools", includes a single category of generic enablers, also named deployment tools, providing tools to distribute and deploy FIWARE components, supporting, among others, Docker and Kubernetes. FIWARE can be installed in fact using Docker or Kubernetes images, as suggested in the documentation as a starting point to get a demo running in a short time. A full installation from source code can be performed by cloning the Github repository of the project, and compiling the generic enablers one by one. Supported host platforms vary from enabler to enabler, although they typically include popular Linux distributions. Depending on the specific generic enabler, the installation may require the download and manual installation of additional packages.

FIWARE is characterized by a rich documentation, both in the Github code repository and on the project website. The website provides in particular a large set of tutorials to explore the different groups of generic enablers.

The distinctive feature of FIWARE is its openness: any developer can in fact contribute to the FIWARE generic enabler catalogue, providing new features or alternative implementations of existing ones; this led to the development of a vibrant ecosystem.

The Kaa 0.X platform [38] is the first generation of the Kaa platform, released under an open source license. Kaa 0.X can be installed locally on a dedicated server, but also supports remote installation on the Amazon Web Services (AWS) platform for a fee, typically depending on the number of devices to be supported. The Kaa 0.X architecture is composed of three key elements: (1) the Kaa server, that has a role similar to the context broker in the FIWARE platforms: it provides the back-end support required to connect and integrate users, applications and devices; (2) Kaa extensions, that provide additional features not available in the server; (3) endpoint software development kits (SDKs), that is software libraries that enable communication and data exchange between the server and hardware platforms. The latest Kaa 0.x version currently available, 0.10, supports overall thirteen hardware platforms using one or more programming languages among C, C++, Objective-C and Java. The supported platforms include both desktop and mobile operating systems, such as Linux, Windows, Android and iOS, and development boards such as Intel Edison, EXP8266 and Texas Instruments CC3200. Kaa supports different communication protocols, including HTTP, extensible messaging and presence protocol (XMPP) and MQTT protocol; although not all protocols are available for all SDKs, Kaa provides a common architecture to transfer data between end-points and server, based on data logs that are collected at each device and then transferred to the server using one of the supported protocols. A major feature in Kaa is the support of a large number of external back-ends and databases: MongoDB, Oracle NoSQL, Kafka, Cassandra are among the supported databases. Logs collected from endpoints are sent to a database by means of software appenders provided in Kaa, and can be later accessed for processing. On the other hand, no built-in data processing capability is available on Kaa, with the sole exception of supporting the transformation of raw data into time series. The Kaa 0.X platform also does not provide any built-in visualization tools.

Kaa provides a wide range of options for installation. The quickest solution is to use a prepackaged sandbox made available in a virtual machine file, that can be downloaded and imported in a virtual engine like Virtual Box, providing a full installation without need to compile or install individual packages. Kaa modules can also be installed from binary packages using the Linux packet manager; finally, a full installation from sources using Maven can be performed with a single shell command.

The documentation consists in the Github code repository readme file and in the documentation pages on the platform website. The website provides installation instructions, a tutorial showing how to build a toy app, and succinct information on platform features and components. It should be noted that the development of Kaa 0.X was arrested in 2017, when the second generation Kaa platform was released, under the name Kaa Enterprise. Kaa Enterprise was however not released under an open source license, although a free pricing tier supporting up to five devices/endpoints using a PaaS approach is available. Although the Kaa 0.X is still available for download [39], the lack of future development should be taken into account when evaluating whether to adopt this platform for academic R and D activities.

Lelylan [40] is an open platform with an architecture based on multiple micro-services that provide key functionalities. These include communications with devices, security and exposing devices to the external world. Lelylan provides in fact a middleware between users and applications

running in the cloud or on the internet and hardware platforms by providing so-called Lelylan Nodes; the current version of the platform only provides MQTT as a Node, limiting the support to devices that use this communication protocol. The core goal of Lelylan is thus to collect user/application requests sent from a Web App over HTTP and forward them to the interested hardware devices, decoupling thus applications and hardware. As such, Lelylan has no built-in data processing or visualization capabilities: it only provides a notification service to applications/users of the outcome of the actions, allowing them to collect the corresponding data for visualization or processing on external platforms. Security in Lelylan is provided by means of the People API, enabling OAuth 2.0-compliant user authorization and authentication.

Lelylan can be installed using docker containers with a single script; local installation of sources requires however the separate download and build of each micro-service from the Github repository. Documentation is limited to the the Github repository documentation, since the project webpage is no longer available.

The Lelylan platform, although still currently available for download, has not seen significant updates in the last few years. It can be thus assumed that it is not being developed any longer, and thus no updates will be released in the foreseeable future. As in the case of Kaa 0.X, the lack of development should be taken into account when evaluating the adoption of the platform for R and D activities.

The Macchina.io EDGE platform [41] allows to develop applications for IoT devices running Linux. The platform provides access to a wide range of devices, exposing hardware-independent APIs to developers, and thus allowing to write code for multiple platforms using JavaScript and C++ languages. The Macchina.io EDGE exposes in fact sensors and actuators made available by each device through a standard web interface, that can be accessed by the application using the platform APIs. This significantly reduces the effort required to build an application for supported devices. Support for devices is provided in Macchina.io by means of the open service platform, that adopts a plugin-based approach to support a wide range of IoT sensors and actuators. The current version supports the MQTT protocol, but support for other protocols such as SOAP, CoAP and Modbus is under development. Security in Macchina.io EDGE is mostly demanded to an external component, Macchina.io REMOTE, that provides secure access to IoT devices by creating a secure tunnel. External clients will connect to Macchina.io REMOTE rather than directly to Macchina.io EDGE. This feature is, however, limited to five devices in the open source version of the platform.

The platform can be installed by downloading sources and compiling them with the make tool with a single command. Following the compilation the platform can be launched as an executable, and accessed through a web interface. Documentation is available both in the Github repository and on the platform webpage. The webpage, in particular, offers a rich documentation including installation instructions and several tutorials on the use of the platform.

The platform is available for free under a GPLv3 license, but paid access is also available for companies interested in releasing proprietary products based on this platform without the open source license restrictions.

OpenMTC [42] is an implementation of the oneM2M standard for M2M communications [43]. The platform consists of two main components: the OpenMTC Backend, that consists of a middleware aggregator that takes care of connectivity with devices, data management and authentication/authorization aspects, and a set of OpenMTC Gateways. Third party users and applications interact with the OpenMTC Backend by means of oneM2M-compliant APIs, that allow them to place data queries and issue commands without specific knowledge of the communication protocol supported by sensors and actuators. The translation between the hardware agnostic APIs and the actual device is carried out by the backend and by the Gateways, that convert hardware-specific data formats to and from the data format compliant to the oneM2M standard. The platform support HTTP/HTTPs and REST APIs as communication protocols with external applications, and the MQTT protocol with devices. The OpenMTC does not provide any built in processing capabilities, deferring them to the external applications that collect data from devices. Visualization features are also limited, since only a basic dashboard is provided. It should be noted, however, that an OpenMTC FIWARE connector was recently made available. The connector provides a bridge between the OpenMTC middleware and the FIWARE Orion context broker, and allows to process and visualize data made

available by devices connected through OpenMTC using all the Generic Enablers defined in FIWARE. Security is guaranteed by supporting secure HTTP connections and certificate-based TLS authentication, and by implementing the concept of dynamic authorization server (DAS), part of the OneM2M standard [44].

The platform can be installed by downloading sources and building both the software development kit and the components into docker containers, that can be then accessed over HTTP. Documentation is available both in the Github repository and on the platform webpage. The webpage includes a short tutorial for getting started and a few examples of code samples to access the platform.

The SiteWhere [45] platform adopts a micro-services-based architecture, similarly to Lelylan. Micro-services provide connectivity with devices, integration with external services, and command delivery to devices. While the first generation of the platform was available in both Community (free) and Enterprise (paid) editions, the SiteWhere 2.0 is currently available only in its open source Community Edition. SiteWhere supports a wide range of protocols including MQTT, AMQP and STOMP protocols for communications with device. SDKs are available supporting integration of Sitewhere with several platforms, including Android, iOS, Arduino and Bluvision. Similarly to Lelylan and OpenMTC, Sitewhere does not offer built-in data processing and visualization; the platform offers instead outbound connectors that allow to asynchronously read data associated to and event source (e.g., a device performing measurements) and forward it for processing to external databases such as Azure and Apache SoIr. Security-related aspects are not addressed in the platform documentation.

The platform can be installed on a kubernetes cluster, either deployed on a local machine or on a cloud installation. Installation from sources can be performed using the gradle building tool with a single shell command. Documentation is available on the platform webpage, and includes deployment and installation guides, although instructions do not make reference to any specific architecture, and include links to external pages for installing tools required for installation. The webpage also provides examples on how to connect to the platform using JSON as well as from Android apps and from the Kura IoT framework [46].

ThingsBoard [47] is a platform available under multiple licenses, ranging from a PaaS version running in the cloud, to an open source community edition, to a paid professional edition. The open source community edition allows both cloud-based and local deployments, and supports several protocols through dedicated APIs, including MQTT, HTTP, constrained application protocol (CoAP), LwM2M and SNMP. In addition, several additional protocols, including BLE, Modbus, OOPC-UA are supported through the IoT Gateway component, that uses converters to translate such protocols in a format compatible with ThingsBoard. Additional converters can be defined taking advantage of the open source nature of the platform. However, support for some technologies, including SigFox, NB-IoT and LoraWAN as well as major enterprise IoT platforms such as AWS, IBM Watson and Microsoft Azure, are only available using the platform integration feature in the professional edition.

ThingsBoard also provides a powerful mechanism for data visualization. Dashboards can be created and used to visualize data using widgets; widgets can be also used to send commands to devices using the RPC protocol.

ThingsBoard enables analytics by means of Kafka Streams that can be generated based on device data and sent to external analytics plaforms. On board advanced analytics tools are available in the professional edition through the Trendz Analytics component.

Finally, security in ThingsBoard is guaranteed by the support of the HTTPS protocol, as well as the use of access tokens compliant to the OAuth 2.0 protocol over all supported communication protocols.

ThingsBoard offers a wide range of installation and deployment options. The platform can be deployed using either docker containers, or installation packages for all major desktop operative systems. Installation from sources can be performed by cloning the project Github repository and using the Maven building tool. The documentation is provided on the platform website, and includes detailed step-by-step deployment and installation guides, as well as a rich set of tutorials and examples covering all key platform features.

The ThingBox [48] is a platform based on the Node-Red visual editor [49] released by IBM to develop IoT applications. The ThingBox adopts an approach rather different from the other

open source platforms reviewed in this section. Rather than providing a middleware or a back-end running in the cloud, this platform provides device support and processing capabilities based on "nodes" made available in the Node-Red editor, each node corresponding to a specific feature (e.g., support for a specific hardware or software platform). The ThingBox is designed to be a "local" IoT implementation, with nodes typically deployed on Raspberry Pi devices, that can operate simultaneously but do not interact.

The ThingBox can be installed from sources, by cloning the project github repository. Since the platform website is no longer reachable the only available documentation is given by the package descriptions in the Github repository.

As in the case of of Lelylan, the platform did not receive any updates in the last few years, and can be assumed to be out of development.

ThingSpeak [50] shares with most of the platforms described so far an architecture based on an open source central server, that can be installed locally or run in the cloud. The server provides visualization, processing and analysis tools, and is complemented by software libraries that enable interaction with IoT devices. The key feature in ThingSpeak is the support of the Matlab IDE in the cloud deployment, including graphic output and several Matlab toolboxes, that provides access to Matlab commands for data processing and visualization. It should be noted that not all features are available in all deployments. In particular, free Matlab support is only available when using the public cloud platform but not in local installations. In addition, access to toolboxes requires a paid license.

Thingspeak supports both the HTTP and MQTT communication protocols to communicate with all major IoT hardware platforms, such as Arduino, Raspberry Pi, Electric Imp, Particle Photon and ESP8266, as well mobile and desktop OSes. ThingSpeak relies on two key concepts for data exchange and storage: channels and messages. Both devices and applications read data from and write data to channels by means of messages exchanged with the central server. In its cloud-based deployment ThingSpeak provides rich data processing and visualization capabilities by means of ThingSpeak apps. Apps can be executed once, periodically or as a reaction to a new message being written on a channel, and allow to process the data, visualize them by generating graphs, or distribute them on other channels or on the internet by posting them on web pages and social networks. The built-in support for Matlab scripts and functions within apps in the cloud is extremely appealing, given the widespread use of Matlab in the academic environment, that leads in fact to an easier porting of new data processing and analysis algorithms from simulation environments to experimental testbeds. This peculiar characteristic of ThingSpeak was taken advantage of in the design of an indoor positioning platform for IoT devices in [51]. However, such features are not available in local server installations, that provide basic visualization capabilities through time series charts and Google Gauges in the platform web interface, and data processing by means of Javascript-based plugins. Integration with external services is possible by exporting data in JSON, CSV and XML formats, or by adopting the ThingHTTP protocol. Finally, security in ThingSpeak is guaranteed by supporting end-to-end encryption with TLS. ThingSpeak can be installed locally by cloning the Github repository available at [52]. A rich documentation is provided on the platform website, although it is mostly focused on the cloud-based version of the platform.

The characteristics of the above platforms are summarized in Table 2, allowing a comparison-at-a-glance of the platforms.

**Table 2.** Comparison of open source platforms.

| Platform | Comm. Protocols | Data Processing | Data Visualization | Integration | Security | Installation Procedure | Documentation |
|---|---|---|---|---|---|---|---|
| Eclipse Kapua [33] | MQTT | no built in processing capabilities | no built in visualization capabilities | REST APIs, WebSockets, NoSQL persistent storage | access control, SSL authentication | demo installation using dockers; installation from sources using Maven | Github repository documentation, platform website |
| FIWARE [36] | HTTP, MQTT, LoRaWAN, OPC-UA | Generic Enablers | Different Generic Enablers in dashboards | LWM2M over CoaP, JSON or UltraLight over HTTP/MQTT or OPC-UA, API | Private OAuth2-based authentication, access control | demo installation using dockers; installation from sources | Github repository documentation, platform website, dedicated websites for generic enablers |
| Kaa 0.X [39] | HTTP, MQTT, XMPP, CoAP | no built -in processing capabilities | no built in visualization capabilities | Oracle, Apache and MongoDB databases, AWS | TLS, DTLS | Sandbox-ready for execution in a prepackaged virtual machine; local installation using binary packages; installation from sources using Maven | Github repository documentation, platform website |
| Lelylan [40] | HTTP, MQTT, APIs | no built-in processing capabilities | no built in processing capabilities | Webhooks, Websockets | OAuth 2.0 support | installation using docker containers; installation from sources | Github repository documentation |
| Macchina.io EDGE [41] | MQTT | no built-in processing capabilities | Web applications | REST API, HTTP | secure access through the external Macchina.io REMOTE component | installation from sources using make | Github repository documentation, platform website |
| OpenMTC [42] | HTTP, HTTPS, MQTT | Dashboard | no built-in processing capabilities | REST API, JSON serialization | Dynamic Authorization Server (DAS), TLS (Certificate-based), HTTPS | Installation from sources using docker containers | Platform website |
| SiteWhere [45] | MQTT, AMQP, STOMP | no built-in visualization capabilities | no built-in visualization capabilities | REST API, Siddhi, Azure EventHub, Apache Solr, Twilio | no information available | Kubernetes-based deployment, installation form sources using gradle | platform website |
| ThingsBoard [47] | MQTT, HTTP, CoAP, LwM2M, SNMP | external, through Kafka streams | dashboards and widgets | limited, platform integration feature required for some platforms | HTTPS, OAuth 2.0 | Deployment using docker containers and installation packages for multiple OSes; installation from sources using Maven | platform website |

**Table 2.** *Cont.*

| Platform | Comm. Protocols | Data Processing | Data Visualization | Integration | Security | Installation Procedure | Documentation |
|---|---|---|---|---|---|---|---|
| The ThingBox [48] | MQTT | determined by Node-RED editor nodes | determined by Node-RED editor nodes | everything supported by Linux | unknown | installation from sources | minimal documentation on the Github repository |
| ThingSpeak [50] | HTTP, MQTT | HTML JavaScript plugins in a local installation, Matlab Analysis Apps in the cloud version | time series visualization and Google Gauges in a local installation, Matlab Visualization Apps in the cloud version | JSON, XML, CSV, REST API, ThingHTTP | TLS | installation from sources | Github repository and platform website, mostly focused on cloud version |

## 3. Comparison of Platforms for R and D in Academia

The analysis carried out in the previous section highlights how the closed source platforms are typically characterized by more mature development environment and deployment options, that typically streamline and simplify the deployment of large-scale IoT apps. These benefits come, however, most often at the price of significant license/subscription fees and with restrictions to the access to the source code.

These two aspects strongly suggested that R and D activities in academic environments should specifically focus on open source platforms. Although this decision led to the exclusion of several powerful and feature-rich platforms, as shown in Section 2.1, the availability of the source code was considered a prerequisite condition for the adoption of a platform as the basis for R and D in an academic environment, where the possibility of analyzing, tweaking or extending the existing platform code base in order to introduce new functions and algorithms is in most cases fundamental.

The analysis carried out in Section 2.2 formed thus the basis for the definition of a ranking between the open source IoT platforms, carried out on the basis of seven criteria: the five criteria introduced in Section 1, common to all IoT platforms, and the criteria related to installation procedure and documentation specific to open source platforms.

The ranking of the platforms was carried out on a per-criterion basis. For each criterion, platforms were assigned to classes based on a relative comparison between their features: the platform with the richest set of feature would automatically define the bar for the top class. Whenever possible, the comparison between platforms was based on quantitative data, e.g., on the number of supported protocols for the communication protocols criterion and on the number of supported services for the integration criterion. Since, however, it was often hard to accurately measure quantitative differences between platforms, we decided to use a small number of classes, rather than adopting a finer granularity that would have led at times to arbitrary assignments. As a result, four classes were defined: “high”, including platforms with the richest set of features, “medium”, including platforms with a good set of features, but not on par with the first group, “low”, including platforms only providing basic functionalities, and finally “absent”, including platforms that do not provide any feature relevant to the criterion under consideration, or for which no information is provided.

Note that the rankings by criteria were not combined to determine a global ranking. This would require in fact to use a multi-criteria decision making (MCDM) method [53], such as a simple weighted sum of scores or the more sophisticated analytic hierarchy process (AHP) [54]; in either case, this implies the assignment of weights associated to the importance of the different criteria. Establishing the relative importance of the criteria and the corresponding weights is however part of the research problem definition, that is beyond the scope of the present work. The rankings provided in the following can however constitute the input for a MCDM method, jointly with weights defined by the researcher based on the considered research problem.

The results of this qualitative comparison are presented in Table 3. Note that for platforms that provide different set of features between their open source, local installation vs. commercial, cloud-based versions, such as ThingsBoard and ThingSpeak, Table 3 refers to the open source installation.

**Table 3.** Qualitative comparison between open source platforms analyzed in Section 2.2.

| Platform | Communication Protocols | Data Processing | Data Visualization | Integration with External Services | Security | Installation Procedure | Documentation |
|---|---|---|---|---|---|---|---|
| Eclipse Kapua [33] | medium | absent | absent | medium | medium | low | low |
| FIWARE [36] | high | high | high | high | medium | medium | high |
| Kaa [39] | high | low | absent | high | low | high | medium |
| Lelylan [40] | medium | absent | absent | medium | medium | medium | low |
| Macchina.io EDGE [41] | medium | absent | absent | medium | low | medium | high |
| OpenMTC [42] | medium | absent | very low | high | high | medium | medium |
| SiteWhere [45] | medium | absent | absent | high | absent | medium | medium |
| ThingsBoard [47] | high | low | high | medium | medium | high | high |
| The ThingBox [48] | low | low | low | medium | absent | low | low |
| ThingSpeak [50] | medium | medium | medium | medium | medium | low | medium |

Table 3 highlights that, with the exception of the ThingBox, which can be considered to some extent a niche platform, most platforms are characterized by satisfactory features under most of the considered criteria. However, the FIWARE platform stands out for its consistently high level of features for most criteria , and emerges thus as a very good candidate as the IoT reference platform for academic R and D activities. As also suggested in [7], however, the selection of the platform to support an R and D project should also be based on the specific requirements of the project itself. As an example, if support for multiple communication protocols and integration with external services are key aspects in the project, FIWARE could be definitely considered the best candidate platform, while if ease and flexibility in the installation process is deemed the most important aspect, Kaa and ThingsBoard are the best suited platforms.

## 4. Conclusions

This paper analyzed existing IoT platforms from the perspective of R and D activities in academia, for which the possibility of adding new features and introducing new processing algorithms, by extending and tweaking the source code, can be considered the most important aspect to be taken into account in the selection of an IoT platform. The review carried out in this work analyzed over 30 platforms and identified a set of ten open source platforms that are suitable for adoption in an academic research project: Eclipse Kapua, FIWARE, Kaa, Lelylan, Machina.io EDGE, OpenMTC, SiteWhere, ThingsBoard, The ThingBox and ThingSpeak. The analysis indicates that although typically multiple platforms provide a similar set of features with respect of each criterion, the FIWARE platform has the overall best performance when considering all criteria simultaneously, and is thus a strong candidate when selecting a IoT platform for academic R and D activities, also thanks to the rich and active community of developers working to add components to the platform. Other platforms provide however specific features that can be extremely appealing for some activities. As an example, Kaa and Macchina.io EDGE provide easy and flexible installation, guaranteeing a quick, smooth start of activities.

**Author Contributions:** Software, L.D.N.; investigation, L.D.N. and A.M.; writing—original draft preparation, review and editing, L.D.N. and M.-G.D.B.; review and editing, A.M., G.C., U.A.; project supervision, M.-G.D.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by Sapienza University of Rome within research projects with Grants No. RP11715C7EFAA443, RP11715C7CA5279D and RM116155068578FB.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

**Abbreviations**

The following abbreviations are used in this manuscript:

| | |
|---|---|
| AMQP | advanced message queuing protocol |
| AP | access point |
| API | application programming interface |
| AWS | Amazon Web Services |
| CIR | channel impulse response |
| CoAP | constrained application protocol |
| DAS | dynamic authorization server |
| DOA | direction-of-arrival |
| HTTP | hyper text transfer protocol |
| ICT | information and communication technology |
| IIoT | industrial internet of things |
| IoT | internet of things |
| IT | information technology |
| JSON | JavaScript object notation |
| LWM2M | LightWeightM2M |
| M2M | machine to machine |
| MQTT | message queuing telemetry transport |
| NGSI | next-generation sensors initiative |
| OPC-UA | open platform communications-unified architecture |
| PDF | probability density function |
| RAT | radio access technology |
| REST | representational state transfer |
| R and D | research and development |
| RP | reference point |
| SCADA | supervisory control and data acquisition |
| SDK | software development kit |
| STOMP | streaming text-oriented messaging protocol |
| TCP | Transport control Protocol |
| TLS | transport layer security |
| TOA | time of arrival |
| TPP | test point |
| TS | ThingSpeak |
| UI | user interface |
| XACML | extensible access control markup language |
| XMPP | extensible messaging and presence protocol |

## References

1. Markets and Markets, IoT Technology Market with COVID-19 Impact Analysis, by Node Component (Sensor, Memory Device, Connectivity IC), Solution (Remote Monitoring, Data Management), Platform, Service, End-Use Application, Geography—Global Forecast to 2027. 2021. Available online: https://www.marketsandmarkets.com/Market-Reports/iot-application-technology-market-258239167.html (accessed on 17 February 2022).
2. Fortune Business Insights, IoT Market Size, Share, Growth, Trends, Business Opportunities, IoT Companies, Statistics, Report 2028. 2021. Available online: https://www.globenewswire.com/news-release/2021/11/10/2331267/0/en/IoT-Market-Size-Share-Growth-Trends-Business-Opportunities-IoT-Companies-Statistics-Report-2028-Internet-of-Things-Industry-Report-Fortune-Business-Insights.html (accessed on 17 February 2022).

3. Juniper Research. The Internet of Things—Consumer, Industrial and Public Services 2016–2021. 2016. Available online: https://www.juniperresearch.com/researchstore/key-vertical-markets/internet-of-things/consumer-industrial-public-services (accessed on 17 February 2022).
4. Xu, L.D.; He, W.; Li, S. Internet of Things in Industries: A Survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243.
5. IoT Analytics, IOT PLATFORMS—The central backbone for the Internet of Things. 2015. Available online: http://iot-analytics.com/wp/wp-content/uploads/2016/01/White-paper-IoT-platforms-The-central-backbone-for-the-Internet-of-Things-Nov-2015-vfi5.pdf (accessed on 17 February 2022).
6. Scully, P. 5 Things to Know About the IoT Platforms Market. 2021. Available online: https://iot-analytics.com/5-things-to-know-about-iot-platforms-market/ (accessed on 17 February 2022).
7. Singh, K.J.; Kapoor, D.S. Create Your Own Internet of Things: A survey of IoT platforms. *IEEE Consum. Electron. Mag.* **2017**, *6*, 57–68.
8. Guth, J.; Breitenbücher, U.; Falkenthal, M.; Leymann, F.; Reinfurt, L. Comparison of IoT platform architectures: A field study based on a reference architecture. In Proceedings of the 2016 Cloudification of the Internet of Things (CIoT), Paris, France, 23–25 November 2016; pp. 1–6.
9. Lorenc, A.; Michnej, M.; Szkoda, M. Information system aiding the logistics processes of loading and securing in railway transport. *Int. J. Shipp. Transp. Logist.* **2016**, *8*, 568.
10. López Peña, M.A.; Muñoz Fernández, I. SAT-IoT: An Architectural Model for a High-Performance Fog/Edge/Cloud IoT Platform. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; pp. 633–638.
11. Xu, J.; Wang, S.; Zhou, A.; Yang, F. Edgence: A blockchain-enabled edge-computing platform for intelligent IoT-based dApps. *China Commun.* **2020**, *17*, 78–87.
12. Dayarathna, M. Comparing 11 IoT Development Platforms. 2019. Available online: https://dzone.com/articles/iot-software-platform-comparison (accessed on 17 February 2022).
13. Ericsson IoT Solutions. Available online: https://www.ericsson.com/en/internet-of-things/solutions (accessed on 17 February 2022).
14. Autodesk Fusion Connect. Available online: https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/ENU/FUSIONCONNECT-Help/What-Is-Fusion-Connect-html.html (accessed on 17 February 2022).
15. IBM IoT Resources. Available online: https://developer.ibm.com/iotplatform/resources/ (accessed on 17 February 2022).
16. Zoho IoT. Available online: https://www.zoho.com/iot/ (accessed on 17 February 2022).
17. Unified IoT (Internet of Things) Platform—WebNMS. Available online: https://www.webnms.com/iot/unified-webnms-iot-platform.html (accessed on 17 February 2022).
18. Losant Enterprise IoT Platform. Available online: https://www.losant.com/ (accessed on 17 February 2022).
19. Ubidots. Available online: http://ubidots.com/ (accessed on 17 February 2022).
20. MindSphere. Available online: https://new.siemens.com/global/en/products/software/mindsphere.html (accessed on 17 February 2022).
21. General Electric Predix IoT Platform. 2018. Available online: https://www.ge.com/digital/iiot-platform (accessed on 17 February 2022).
22. Cisco Edge Intelligence. Available online: https://www.cisco.com/c/en/us/solutions/internet-of-things/edge-intelligence.html (accessed on 17 February 2022).
23. Google Cloud IoT Solutions. Available online: https://cloud.google.com/solutions/iot (accessed on 17 February 2022).
24. Altair Community. Available online: https://www.altair.com/smartworks/ (accessed on 17 February 2022).
25. Axway Appcelerator. Available online: https://www.appcelerator.com (accessed on 17 February 2022).
26. Amazon Web Services. Available online: https://aws.amazon.com (accessed on 17 February 2022).
27. Bosch IoT Suite. Available online: https://www.bosch-si.com/iot-platform/bosch-iot-suite/homepage-bosch-iot-suite.html (accessed on 17 February 2022).
28. C3. Available online: https://c3.ai (accessed on 17 February 2022).
29. Evrything. Available online: https://evrythng.com (accessed on 17 February 2022).
30. ThingWorx. Available online: https://www.ptc.com/en/resources/iot/product-brief/thingworx-platform (accessed on 17 February 2022).
31. Wyliodrin. Available online: https://www.wyliodrin.com/ (accessed on 17 February 2022).
32. Microsoft Azure IoT Accelerators. Available online: https://azure.microsoft.com/en-gb/services/iot-central/ (accessed on 17 February 2022).
33. Eclipse Kapua Platform. Available online: https://projects.eclipse.org/projects/iot.kapua (accessed on 17 February 2022).
34. Eclipse IoT Community Website. Available online: https://iot.eclipse.org/ (accessed on 17 February 2022).
35. Apache Maven Build Automation Tool. Available online: https://maven.apache.org/ (accessed on 17 February 2022).
36. FIWARE. Available online: https://www.fiware.org (accessed on 17 February 2022).
37. ETSI White Paper No. 31: NGSI-LD API: for Context Information Management. Available online: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp31_NGSI_API.pdf (accessed on 17 February 2022).
38. Kaa Project. Available online: http://www.kaaiot.com (accessed on 17 February 2022).
39. Kaa 0.10 Open Source Platform. Available online: https://www.kaaiot.com/kaa-open-source (accessed on 17 February 2022).

40. Lelylan. Available online: https://github.com/lelylan/lelylan (accessed on 17 February 2022).
41. macchina.io—IoT Edge Device Software Development and Secure Remote Access Solutions. Available online: https://macchina.io/index.html (accessed on 17 February 2022).
42. OpenMTC. Available online: http://www.open-mtc.org (accessed on 17 February 2022).
43. OneM2M—The IoT Standard. Available online: https://www.onem2m.org/ (accessed on 17 February 2022).
44. OneM2M Technical Report—Dynamic Authorization. Available online: https://www.onem2m.org/images/files/deliverables/Release2/TR-0019-Dynamic_Authorization-V2_0_0.pdf (accessed on 17 February 2022).
45. SiteWhere. Available online: http://www.sitewhere.org/ (accessed on 17 February 2022).
46. Kura IoT Edge Framework. Available online: https://www.eclipse.org/kura/ (accessed on 17 February 2022).
47. ThingsBoard Open-Source IoT Platform. Available online: https://thingsboard.io/ (accessed on 17 February 2022).
48. Thingbox. Available online: https://www.npmjs.com/~thethingbox (accessed on 17 February 2022).
49. Node-Red Visual Editor. Available online: https://nodered.org (accessed on 17 February 2022).
50. ThingSpeak. 2018. Available online: http://www.thingspeak.com (accessed on 17 February 2022).
51. De Nardis, L.; Caso, G.; Di Benedetto, M.G. ThingsLocate: A ThingSpeak-Based Indoor Positioning Platform for Academic Research on Location-Aware Internet of Things. *Technologies* **2019**, *7*, 50.
52. ThingSpeak Github Repository. Available online: https://github.com/iobridge/thingspeak (accessed on 17 February 2022).
53. Jadhav, A.; Sonar, R. Analytic Hierarchy Process (AHP), Weighted Scoring Method (WSM), and Hybrid Knowledge Based System (HKBS) for Software Selection: A Comparative Study. In Proceedings of the 2009 Second International Conference on Emerging Trends in Engineering Technology, Nagpur, India, 16–18 December 2009; pp. 991–997.
54. Saatyì, T.L. *The Analytic Hierarchy Process*; McGraw-Hill: New York, NY, USA, 1980.