Creative Coding Within the Internet of Things

Lasse Steenbock Vestergaard
Alexandra Instituttet A/S
lasse.vestergaard@alexandra.dk

João Fernandes
Alexandra Instituttet A/S
joao.fernandes@alexandra.dk

Mirko Presser
Aarhus University
mirko.presser@btech.au.dk

Abstract—In this paper, we investigate how far IoT usability has come, and present a field trial on how Creative Coders – digital artists who are self-taught programmers - interact with the world of IoT through a simplified HTTP oriented publish/subscribe service. When elaborating our findings, we discuss the implications of reducing technological complexity in production oriented ecosystems like IoT platforms.

Index Terms—IoT; Creative Coders; Makers; Platforms, Development tools.

I. Introduction

In recent years, we have been witnessing a considerable increase in the number of IoT platforms made available to the broad public [43]. This platform availability enable users to share, and get access to data produced by millions of connected IoT devices, and have the possibility of developing and using applications and services that can range from home automation, energy management, healthcare, transportation, consumer applications, smart cities, etc. In parallel to the increase IoT platforms and enablers, we have also seen an increase in entrepreneurship with many start-up companies building simple but very successful products, that have extremely huge impact in society (e.g. Twitter, Snapchat, Instagram, 46Elks [1], etc.).

As the IoT platforms and their tools have evolved, it seems that usage complexity is reducing (due to usability features in exposed services and interfaces). We are interested in investigating how far IoT usability has come, and as a result this paper focuses on communities of creative people developing IoT products, and using different tools during that process.

Throughout the paper, we will present an excerpt of existing IoT platforms, and simplified services, and we discuss IoT platform characteristics. Furthermore, we carve out the Creative Coder, and how they interact with software. Finally, we present a field trial, and discuss implications of providing access to Creative Coders within the IoT.

II. IoT Platforms and Tools

IoT platforms are complex and intertwined ecosystems of hard- and software, which facilitate both development, deployment, maintenance and analytics as well as intelligent decision making capabilities. In the following, we will elaborate a few of the existing platforms, and tool that are currently prominent on the IoT market, and briefly touch on their capabilities:

Xively [15] formerly known as Pachube and later Cosm, was created in 2007 and is now a division of LogMeIn Inc. It provides a set of tools that allow users to build and manage connected products and applications. Through the provision of an extensive list of API wrappers that support different programming languages and platforms the user can easily connect their IoT devices and applications to the Xively service. Many different tools developed around Xively have been released as open source allowing the engagement of the developer community in the further use and development of these tools.

ThingWorx [6] created in 2009 focuses on the integration, transformation and presentation of data. The platform aims at providing simple, and easy to use data management functionalities, that enable people with non-technical expertise to build their applications. ThingWorx provides different tools such as the Codeless Mashup Builder, that allows the easy creation of applications by simple "drag and drop", and social networks integration like Facebook, Twitter and Google+ that allow the collaboration of both users and developers. ThingWorx also integrates a variety of protocols such as REST, MQTT, or traditional sockets and offers interfacing with business systems like SAP, Oracle and Salesforce.com.

Amazon Web Services [2] allows Internet of Things through the provision of a suite of cloud computing services including computing, storage, networking, mobile, deployment, etc. Among the different provided tools, the Amazon Kinesis tool allows the collection of high throughput data from devices, data analysis and storage over the cloud. This allows applications to consume the data and enables quick decision making. AWS provides flexibility for IoT applications in terms of tools, programming languages, data management and other infrastructure resources.

BlueMix [14] is a cloud platform as a service (PaaS) developed by IBM. Supporting different programming languages such as Java, PHP, Python, Node.js as well as others and integrating DevOps that allows the build, run, deployment and management over the cloud. It provides powerful application access to IoT devices and data and supports the easy development of analytics, visualisation and mobile IoT applications. Through the provision of simple and secure REST APIs the user is able to connect the data with the applications.

978-1-5090-5873-0/17/$31.00 ©2017 IEEE

978-1-5090-5249-0/17/$31.00 ©2017 IEEE

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on September 20,2020 at 23:35:41 UTC from IEEE Xplore. Restrictions apply.

Microsoft Azure Cloud [23] is a cloud computing platform that allows build, deployment and management of IoT applications and services. Providing both PaaS and IaaS services the platform supports multiple programming languages, tools and frameworks. The platform provides connectivity of millions of devices and sensors with IoT applications and provides remote access, monitoring, and content distribution and configuration management for the connected devices. It provides big data analytics and integration with other business tools.

In addition to these platforms a big effort has also been put into creating simple tools that allow users to build their IoT applications. Some examples of these tools include:

Particle.io [31] a prototype-to-production platform for developing IoT products. Previously known as Spark.io the platform allows users to connect a number of IoT devices. Users can acquire a Development Kit that includes a number of sensors/devices and in an easy way set-up these and connect them to the platform.

Dweet.io [5] is a simple and lightweight messaging service for devices. Through a simple REST API users are able to connect devices and the data they provide - triggering alerts, real-time data streams access, etc.

IFTTT [18] is a web-based service that allows its users to create chains of simple conditional statements - called recipes - which can be triggered based on changes to other web services - such as Gmail, Facebook, Instagram, LinkedIn, OneDrive, etc. Overall the chains can connect to over 300 different channel services. This simple tool allows the users to build simple applications that connect different services and do not require complex technical skills to be used.

III. Characteristics of IoT Platforms

The existing IoT platforms have many similarities in terms of characteristics - architecture, functionalities and service provisioning. The typical IoT platform provides a simple wrapper API through which devices and applications connect to, and share or get access to data. The process of device registration usually includes security mechanisms for authentication (typically OAuth [28]), authorisation/access to the resources (typically OpenID [30]). Applications get access to data via publish/subscribe or queueing mechanisms that allow the decoupling from the involved entities (consumer and producer), as well as allows the near real-time up-to-date information at the consumer's side. One of the protocols for data exchange used in IoT platforms is MQTT [26] - Message Queueing Telemetry Transport which provides a lightweight publish-subscribe messaging pattern, where brokers distribute the messages to the interested clients based on the topic of the message. Another example of supported protocols is WebSockets [7], which provides a full-duplex communication channel over a TCP connection, allowing the same publish-subscribe pattern to be used, and reducing the overhead of HTTP as the connection is always open.

A number of IoT platforms are supporting only specific devices and therefore specialised, and providing better support for these products - firmware, detailed tutorials and general support, making it an easier experience for the customers to setup their devices and develop their solutions. Nevertheless, a strong effort has been put into standardisation and device interoperability in IoT, where devices, applications and platforms should be able to understand each other with near-zero-configuration and follow standards for communication.

Architecture-wise IoT platforms follow a monolith or micro-services pattern [25], which satisfy scalability and availability requirements, and take advantage of different technologies, programming languages and frameworks. Through the provision of a publish-subscribe pattern for data exchange and data storage services, the platforms allow simple, reliable and near real-time data propagation. These services also allow to minimise the processing, and consequently the use of power from devices and applications, which is vital as these are resource-constrained.

Despite many IoT platforms being developed for an enterprise/professional developer customer, we investigate another type of user - the Maker community, with an entrepreneurial mind-set that tap into the IoT platforms to make ideas turn into prototypes or products.

IV. Carving Out the Creative Coder

As described earlier, this paper focuses on how to embrace people tinkering with software, within the world of IoT. We are particularly interested in these communities, since they are emerging all over the technological landscape and because quite a few commercial initiatives have started to put emphasis on this group of developers. A few examples are Philips Hue [33], Particle.io, Ninja Blocks [27], freeboard.io [9] and Dweet.io. It seems that companies have realised the potential in making hobbyists tinker with their products, thereby facilitating the developments of creative interactions and prototypes (as well as getting free exposure to the general public). Inspired by this approach, our pivot point is creative digital tinkerers, which can be termed Creative Coders (CC). In order to define the CC, we first need to elaborate the term Maker, since CCs can be perceived as a subset of Makers.

The Maker term describes the process of tinkering with a material, where learning-by-doing is the main approach. It is all about building sketches and prototypes, exploring material qualities, externalising ideas (not necessarily considering the business model) and following the urge of "just doing stuff". A major driver is the explorative nature, where everything goes and sketchy prototypes are the primary outcome [10]. Making, as a term, has emerged and expanded rapidly, over the last decade. What started out as a Do-It-Yourself (DIY) community focusing on technologies like 3D printing, laser cutting and tinkering with Arduino [3] has now expanded to also apply in other communities such as knitting, cooking, carpentry, physical art installations, architecture etc. The term has broadened and is turning into a methodology or approach that apply to more or less all crafting oriented communities [32]. The dynamics and processes playing out within Maker communities are quite close to how CC communities work, and they can therefore be understood as a specific type of Makers,

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on September 20,2020 at 23:35:41 UTC from IEEE Xplore. Restrictions apply.

where the overarching approach is the same - learning-by-doing through exploring material qualities.

Creative coding has roots back to the 1960's [11], but the term has crystallized over the last decade [21]. What differentiates CCs from being merely Makers is that their main tool for expression is code. The pivot point is the creative processes that unfolds around externalising thoughts, construction and the explorative conversation with the code [38]. Comparing CCs to the professional enterprise programmers, the latter is primarily focused on convergence, and solving tasks in order to develop a fully working system that is robust and production-ready. On the other hand, the CC is more interested in exploring problems through parallel and divergent creative processes [20]. One could state that the enterprise programmer is programming from a specification, while the CC is programming as a conversation with the code [16].

CCs are usually portrayed as focusing on visual expressions on the computer screen, often leveraging virtual 3D environments. In our interpretation of CC, we broaden the definition to also encompass microcontroller developments, e.g. leveraging the Arduino platform and hardware, both since hardware often is programmed and configured through code, and because the Arduino approach definitely invites CCs to unfold their creativity through hardware. Additionally, we argue that developing APIs and libraries is an art form in itself, which can be perceived as a creative process of externalising thoughts. This argument is supported when looking at existing CC platforms like Arduino, Processing [35], OpenFrameworks [29] etc. They all support and encourage the communities to develop new extensions, both to enrich the platforms, and as a way of supporting other CCs. This particular aspect is our main focus in this article, and we investigate how APIs can lower the entry barrier for CC's to interact with professional IoT platforms as well as how these APIs can strengthen the creative processes.

As a final remark, the CC term is constantly evolving much like the Maker term. It is therefore necessary to establish what type of CC we are investigating. In general, there seem to be a consensus about software as being the main tool for constructing artefacts, but there are two overarching approaches towards usage of software. One is to use visual programming environments, where the drag and drop paradigm is prevalent. This counts platforms like Max/MSP [22], Pure data [36], VVVV [41], and Scratch-like [39] applications. The other approach is to write actual code through e.g. Arduino (the Wiring language [42]), Processing (Java), openFrameworks (C++), and d3.js (JavaScript) [4]. Since we are interested in how CCs get access to professional IoT platforms, and we have focus on API usage, it makes sense to have the latter CC approach as our pivot point. When focusing on actual code, CCs are able to tap into IoT platforms on a low level, and we can analyse existing APIs directly without taking graphical user interfaces into account, and we can focus on the crafting of code.

V. Creative Coders Interacting with Software

Most current CC tools are following the same approach, where lowering the entry barrier and reducing complexity are the main aspects. This can be seen in e.g. Processing, which is both a simplified programming abstraction on top of Java, and an integrated development environment (IDE) that enables CCs to immediately get up and running. The Arduino platform is similar to Processing in approach, and its IDE is based on Processing. Arduino is leveraging the Wiring programming language, thereby adding a simplified abstraction layer on top of C++. The Particle.io platform is yet another platform heavily inspired by Arduino, and has taken the initial setup phase a step further. It is not necessary to download and install an IDE, because it is readily available in the browser, and the Particle.io hardware can be flashed directly through the website. Looking at the visual programming platforms, the same simplification approach applies here as well. VVVV, IFTTT, Pure data, MAX/MSP, TinkerSpace [40] all utilise a graphical drag and drop abstraction on top of the actual code.

From the above, it is evident that CCs are mostly interested in getting up and running fast, and developing rapid prototypes. One technique used for simplification is the convention over configuration approach [37], where the platform or the tool has made initial assumptions about certain settings. This specific technique is quite clear in a power reduction library written for Arduino [19]. When programming power optimizations in microcontrollers, one would usually write code on bit shifting specific registers in a specific order, but the Arduino library has massively reduced the complexity of such a task. It is simply possible to write a single line of code stating that the microcontroller should go to sleep for x seconds. Such an abstraction reduces complexity immensely, but it also makes quite a few assumptions behind the back of the developer. In relation to the CC, this complexity reduction is of great value since they otherwise would not be able to utilise power optimization features of microcontrollers.

In addition to the implicit complexity reduction paradigm in existing CC tools and platforms, field studies have been conducted on how programming is performed in practice, both from a general api usability and CC perspective [12][34][24]. Especially, Mitchell and Bown have reported six CC needs when interacting with code. In order to support CCs in their coding practices it is necessary to have 1) a visible system state that explicates immediate errors and issues. 2) Since CC's are quite explorative in their coding practices, they need the ability to mix and match different libraries directly in their code. There is a need for modularity, in the same way as the prevalent practices around professional coding where the import statement is heavily leveraged in most modern languages like Java, Python, Node.js, C++ etc. 3) CCs need to get up and running fast, meaning that they should be able to draw e.g. shapes on the screen immediately after starting the IDE. This way feedback is provided fast and continuously, and thereby the conversation between CC and code becomes a loop of tweaking the code, running the program, and observing consequences. 4) A prominent aspect when CCs work with code is the distance between the CC's mental mapping of the

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on September 20,2020 at 23:35:41 UTC from IEEE Xplore. Restrictions apply.

problem and the outcome of tweaking code. If the outcome is too far from what was expected, it might impose a breakdown [13] in the creative process. It is therefore relevant that the programming environment is capable of providing support in these situations. This could e.g. be through suggestions, code examples or having an active forum where CCs can ask questions. 5) Another important aspect is that CCs often use parameter tweaking as a code exploration technique. This aspect relates to the need of getting direct and fast feedback when writing code. This is much like the renowned WYSIWYG [8] paradigm, where visual feedback is provided directly as a consequence of tweaking specific parameters. Additionally, this aspect helps CCs to test and explore potentials and boundaries for their code. 6) Finally, CCs want to have a good overview of what is happening in their code. If an external library or code snippet is too complex to comprehend they might reject it, thereby trying to develop the functionality themselves or simply discarding it all together. As a concluding remark, CCs need tools that are easy to use and learn, and which are relevant for their immediate problem space [24].

VI. Creative Coders and IoT

As established in the previous section CCs are primarily working with code in a rapid prototyping fashion, where ease of use and immediate feedback is favoured. Code qualities like performance, security and code purity is not as important as the ability to develop creative expressions and interactions. From a professional enterprise developer perspective, this approach might seem laissez-faire, and professional IoT platforms are developed with performance and security as the absolutely most critical elements. In addition, IoT platforms often conform to established standards like REST, MQTT, TCP, OAuth etc. These standards can be quite hard to comprehend, especially for self-taught CCs. As a consequence, there is quite a huge gap between IoT platforms and CCs, and it is therefore relevant to explore how the entry barrier can be lowered so that CCs can start to understand, and utilise these kinds of platforms. By providing CCs access to IoT platforms, they gain new abilities, and they can start using real-time data streams, and real-world data in their artistic expressions.

In order to investigate how CCs and IoT mix, we have conducted a field trial, where we exposed CCs to a specific lightweight publish/subscribe oriented IoT service. A description of and findings from the trial will be elaborated in the following section.

A. Dweet.io as Case

As mentioned under the section on characteristics of IoT platforms the publish/subscribe pattern is a central part of most modern IoT platforms. Consequently, we have chosen to focus on the publish/subscribe pattern as specific case for exploring how CCs can interact with IoT platforms. More specifically, we have conducted a field study on the IoT tool Dweet.io. The tool is an extremely simplified implementation of the publish/subscribe pattern leveraging the HTTP protocol, REST and JSON. From a practical perspective, we created a three-hour workshop with five participants, where three were professional developers (two of which are the authors of this paper), and two were CCs. All participants got a Particle.io Photon, and was instructed to publish sensor data using Dweet.io (the two CCs did not know about the tool beforehand). Additionally, everyone should develop a simple graphical user interface which subscribed to a data stream (again using Dweet.io). During the workshop, all participants were instructed to tweet about their learnings, and we conducted a 45-minute focus group interview [17] at the very end.

Our primary findings, from the field trial, showed that Dweet.io has managed to simplify messaging to a level so that it is easy to get up and running, and easy to use. One of the tweets states "I was confused. It was too easy"[1]. And another stated "This is one of the best CLIs [command line interfaces] I have ever seen". It turned out that the participants felt Dweet.io to be quite intuitive, but they initially struggled with getting the Photon up and running. In general, everyone managed to start pushing sensor data to Dweet.io within half an hour. A few of the participants did struggle with the API due to its use of HTTPS. Participants experienced that the data would not be sent from the sensor to the Dweet.io platform. This problem was quickly solved by using some of the client libraries Dweet.io have developed. As a result of the practical part of the workshop, all participants found the tool intuitive, they did not experience any severe issues, and managed to keep up the pace of their creative processes while developing simple sensor prototypes.

An interesting finding from the following interview was that none of the participating CCs knew about the publish/subscribe pattern, and did not have any previous knowledge or experience with IoT platforms. They were surprised when the workshop facilitator told them that they had been utilizing a standard IoT platform feature. Until this moment, the CCs had been trying to avoid IoT platforms because they felt that they were too complex for them to use.

At the same time, they stated that they had now been motivated to leverage this kind of technology in their future work. One of the participating professional developers was also quite positive about the tool, but did not like that Dweet.io had chosen to use the GET request method wrongly. In the API, it is possible (and encouraged) to use GET for publishing data to Dweet.io. Additionally, he stated that the tool was so simple that he would probably develop his own version if he would need this feature in a future project.

Findings from the empirical study suggests that Dweet.io has managed to reduce complexity to a level so that CCs are able to comprehend the ability of the tool, it is easy to get up and running, and it is fast to develop prototypes. Dweet.io can therefore be seen as a successful tool for narrowing the gap between CCs and IoT platforms. From these results, it is relevant to discuss the relationship between CCs and IoT platforms - what is gained from enabling CCs to tap into the world of IoT?

[1] We have extracted the tweets so that they are available in an offline spreadsheet

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on September 20,2020 at 23:35:41 UTC from IEEE Xplore. Restrictions apply.

VII. Discussion

When introducing CCs to the world of IoT there are quite a few aspects to consider. From the IoT platform perspective, it could be an advantage to embrace CCs because a broad segment of developers is enabled to actually understand and utilise such platforms. A consequence is that innovation might accelerate due to lower entry barriers, and ease of use. IoT platforms could be used in unconventional settings like art installations, and it would be relevant to explore how to make these platforms into creativity support tools. Finally, the CC communities are quite vibrant, and people often show of their creations, thereby providing free advertisement for the IoT platforms. On the other hand, security and performance measures might become an issue since these are now show stoppers for the CCs creative coding process. Additionally, developed applications will primarily be explorative prototypes which might not explicate the strength of the platforms. The prototypes might not be leveraging the IoT platforms optimally, they would probably stop working because of bad coding style, and the produced applications might not even have a value proposition, thereby not really showing how the platform can sustain business. From a CC perspective, IoT platforms can be seen as a new material to explore. Being able to play around with real-time and big data is a rather novel element for CCs. They become empowered to utilise the here-and-now aspect of technology, which again provides new constraints and opportunities in their prototyping processes. Finally, CCs get to know the basics of how IoT platforms are constructed, thereby giving them a better understanding of large software infrastructures. When providing CCs with a sneak peek into the engine room of large scale systems, they might also be frightened by all the complexity, and simply just try to avoid it altogether. The high degree of security might scare CCs of, since this layer adds extra complexity to an already complex system. IoT platforms usually conforms to a variety of standards and protocols, which again makes the learning curve steep as well as forcing the CC to be quite strict with their coding style and syntax. Finally, the documentation might be too esoteric or simply insufficient.

On an overall level, there are some unresolved issues like how to go about performance and security when engaging CCs. From a practical perspective, it might actually propel future developments on interfaces and services in order to make more people actively engaged in using and further develop IoT platforms. It might also accelerate companies' willingness to push the CC mind-set into their products by refining existing software tools (like Philips HUE), IoT enable conventional products, and think IoT as a basic element of future product developments.

VIII. Conclusions and Future Work

In this paper, we have presented our initial research related to Creative Coders, and their use of IoT tools. We have elaborated a few of the most prominent IoT platforms on the market, described common IoT characteristics, defined who the Creative Coders are, and how they interact with software and particular IoT tools. Additionally, we have conducted a field trial on how Creative Coders interact with a specific high level publish/subscribe IoT tool.

Our findings suggest that it is possible to reduce IoT tool complexity so that a broader segment of developers can leverage the world of IoT, and a consequence could be that more Creative Coders want to embrace IoT instead of circumventing it.

Finally, we have discussed the implications of providing access for prototyping oriented spaghetti coding Creative Coders within the world of IoT. The discussion tells us that the IoT definitely could benefit from embracing Creative Coders, but this group of developers might need to be educated on especially security.

Acknowledgment

The research reported in this paper has been supported by the H2020-ICT-2014-1 OrganiCity project – 645198.

References

[1] 46elks: *https://46elks.com/*. Accessed: 2017-02-07.

[2] Amazon Web Services (AWS) - Cloud Computing Services: *https://aws.amazon.com/*. Accessed: 2017-02-07.

[3] Arduino: *https://arduino.cc*. Accessed: 2017-01-24.

[4] D3.js - Data-Driven Documents: *http://d3js.org/*. Accessed: 2016-04-28.

[5] dweet.io - Share your thing- like it ain't no thang.: *http://dweet.io/*. Accessed: 2017-02-07.

[6] Enterprise IoT Solutions and Platform Technology: *https://www.thingworx.com/*. Accessed: 2017-02-07.

[7] Fette, I. 2011. The WebSocket Protocol. *Internet Engineering Task Force, Request for Comments*. 53, 9 (2011), 1–79.

[8] FOLDOC - Computing Dictionary: *http://foldoc.org/WYSIWYG*. Accessed: 2017-02-07.

[9] freeboard - Dashboards For the Internet Of Things: *http://freeboard.io/*. Accessed: 2017-02-07.

[10] Gauntlett, D. 2014. ON MAKING , SUSTAINABILITY AND THE IMPORTANCE OF SMALL STEPS : Corresponding Authors. 1, 1 (2014), 1–27.

[11] Greenberg, I. 2007. *Processing : creative coding and computational art*. Friends of Ed, an Apress Co.

[12] Hansen, N.B. et al. 2014. Crafting code at the demo-scene. *Proceedings of the 2014 conference on Designing interactive systems - DIS '14*. (2014), 35–38.

[13] Heidegger, M. and Macquarrie Edward, J. and R. 1962. *Being and Time*.

[14] IBM Bluemix - Cloud Infrastructure, Apps &amp; Platform Services: *https://console.ng.bluemix.net/*. Accessed: 2017-02-07.

[15] IoT Platform for Connected Devices | Xively by LogMeIn: *https://www.xively.com/*. Accessed: 2017-02-07.

[16] Kirton, T. et al. 2013. C4: a creative-coding API for

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on September 20,2020 at 23:35:41 UTC from IEEE Xplore. Restrictions apply.

media, interaction and animation. *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*. (2013), 279–286.

[17] Krueger, R.A. and Casey, M. a 2001. Designing and conducting focus group interviews. *Social Analysis Selected Tools and …*. 36, October (2001), 4–23.

[18] Learn how IFTTT works - IFTTT: *https://ifttt.com/*. Accessed: 2017-02-07.

[19] Lightweight Low Power Arduino Library – Rocket Scream: *http://www.rocketscream.com/blog/2011/07/04/lightweight-low-power-arduino-library/*. Accessed: 2017-02-07.

[20] Lindell, R. 2013. Crafting interaction: The epistemology of modern programming. *Personal and Ubiquitous Computing*. 18, 3 (2013), 613–624.

[21] Maeda, J. 2004. *Creative code*. Thames & Hudson.

[22] MAX/MSP: *https://cycling74.com/products/max/#.VyJXzaNcRHw*. Accessed: 2016-04-28.

[23] Microsoft Azure: Cloud Computing Platform &amp; Services: *https://azure.microsoft.com/en-us/*. Accessed: 2017-02-07.

[24] Mitchell, M.C. and Bown, O. 2013. Towards a Creativity Support Tool in Processing : Understanding the Needs of Creative Coders Bolstering creative engagement via computational building. (2013), 143–146.

[25] Monolithic Architecture pattern: *http://microservices.io/patterns/monolithic.html*. Accessed: 2017-02-07.

[26] MQTT: *http://mqtt.org/*. Accessed: 2017-02-07.

[27] Ninjablocks: *https://ninjablocks.com/*. Accessed: 2017-02-07.

[28] OAuth Community Site: *https://oauth.net/*. Accessed: 2017-02-07.

[29] openFrameworks: *http://openframeworks.cc/*. Accessed: 2017-01-24.

[30] OpenID Foundation website: *http://openid.net/*. Accessed: 2017-02-07.

[31] Particle: Connect your Internet of Things (IoT) devices: *https://www.particle.io/*. Accessed: 2017-02-07.

[32] Peppler, K. and Bender, S. 2013. Maker movement spreads innovation one project at a time. *Phi Delta Kappan*. 95, 3 (2013), 6.

[33] Philips HUE: *http://www2.meethue.com/*. Accessed: 2015-10-25.

[34] Piccioni, M. et al. 2013. An empirical study of API usability. *International Symposium on Empirical Software Engineering and Measurement* (2013), 5–14.

[35] Processing.org: *https://processing.org/*. Accessed: 2016-04-28.

[36] Pure Data — Pd Community Site: *https://puredata.info/*. Accessed: 2017-02-07.

[37] Ruby on Rails: Doctrine: *http://rubyonrails.org/doctrine/#convention-over-configuration*. Accessed: 2017-02-07.

[38] Schön, D.A. 1983. *The Reflective Practitioner How Professionals Think in Action*. Basic Books.

[39] Scratch - Imagine, Program, Share: *https://scratch.mit.edu/*. Accessed: 2016-04-28.

[40] Tinkerspace | Tinkerspace: *http://www.tinkerspace.se/*. Accessed: 2017-02-07.

[41] vvvv - a multipurpose toolkit | vvvv: *https://vvvv.org/*. Accessed: 2016-04-28.

[42] Wiring: *http://wiring.org.co/*. Accessed: 2017-01-24.

[43] IoT Analytics – Market Insights for the Internet Of Things Will the industrial internet disrupt future smart factories.

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on September 20,2020 at 23:35:41 UTC from IEEE Xplore. Restrictions apply.