UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

Máster Universitario en Ingeniería de Telecomunicación

ILLINOIS INSTITUTE OF TECHNOLOGY

SCHOOL OF APPLIED TECHNOLOGY

ITMT 597 SPECIAL PROBLEMS IN INFORMATION TECHNOLOGY

Information Technology and Management Master's Degree

DISEÑO Y DESARROLLO DE UNA APLICACIÓN BLOCKCHAIN PARA INTERNET OF THINGS

DESIGN AND DEVELOPMENT OF A BLOCKCHAIN APPLICATION FOR INTERNET OF THINGS

Francisco de Paula García de la Corte

2018

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

Máster Universitario en Ingeniería de Telecomunicación

ILLINOIS INSTITUTE OF TECHNOLOGY

SCHOOL OF APPLIED TECHNOLOGY

ITMT 597 SPECIAL PROBLEMS IN INFORMATION TECHNOLOGY

Information Technology and Management Master's Degree

DISEÑO Y DESARROLLO DE UNA APLICACIÓN BLOCKCHAIN PARA INTERNET OF THINGS

DESIGN AND DEVELOPMENT OF A BLOCKCHAIN APPLICATION FOR INTERNET OF THINGS

Francisco de Paula García de la Corte

2018

Mayo 2018

TRABAJO FIN DE MÁSTER

TITLE:	Design and development of a blockchain application for Internet of Things
AUTHOR:	Francisco de Paula García de la Corte
TUTOR:	Jeremy Hajek

TRIBUNAL:

SAT Dean:	Dr. Robert C. Calson
ITM Associate Chair:	Ray Trygstad
Research Faculty Member:	Jeremy Hajek
Academic Adviser:	Amber Chatellier

FECHA DE LECTURA: _____

CALIFICACIÓN: _____

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

Máster Universitario en Ingeniería de Telecomunicación

ILLINOIS INSTITUTE OF TECHNOLOGY

SCHOOL OF APPLIED TECHNOLOGY

ITMT 597 SPECIAL PROBLEMS IN INFORMATION TECHNOLOGY

Information Technology and Management Master's Degree

DISEÑO Y DESARROLLO DE UNA APLICACIÓN BLOCKCHAIN PARA INTERNET OF THINGS

DESIGN AND DEVELOPMENT OF A BLOCKCHAIN APPLICATION FOR INTERNET OF THINGS

Francisco de Paula García de la Corte

2018

Resumen

A medida que IoT, una tecnología emergente, comienza a hacerse hueco en el mercado, empieza a tener problemas de escalabilidad y seguridad que no pueden solucionarse con la tecnología y protocolos actuales.

Sin embargo, gracias a las últimas tendencias tenemos la solución: *Blockchain*. ¿Y si cada dispositivo pudiera leer o registrar datos en un libro de cuentas constantemente compartido por todos, y casi imposible de manipular? Hay muchas aplicaciones para esta nueva tecnología, siendo la gestión de casas inteligentes una de ellas: el cielo es el límite.

Por lo tanto, queremos comprender y dar vida a una idea de un caso de uso en el que *blockchain* puede resolver con éxito los problemas de IoT, con los recursos disponibles y herramientas de código abierto. Queremos demostrar que *blockchain* es la solución, y de paso proporcionar arquitecturas más baratas para sistemas IoT. Ese caso de uso incluirá los sensores y actuadores típicos que se pueden encontrar en una casa inteligente.

En este documento se tratan las decisiones más importantes, en diversos capítulos, sobre el diseño e implementación del sistema general que incluye módulos *blockchain*, *cloud computing* e IoT. Previamente a ese desarrollo, el documento contiene algunos capítulos teóricos sobre esas tres tendencias. El documento termina con un capítulo de conclusiones y otro último sobre líneas futuras.

Palabras clave: *Blockchain*, bitcoin, criptomoneda, descentralizado, *cloud computing*, *Internet of Things*, sensores, actuadores, web, escalabilidad, modular, *open source*, aplicación.

Abstract

As IoT, an emerging technology, starts to build a place for itself in the market, it is starting to face scalability and security problems that cannot be helped with our current technologies and protocols.

However, we can thank recent trends for providing us with the solution: Blockchain. What if every device could read/write on a ledger that is constantly shared by everyone, and almost impossible to tamper with? There is a lot of applications for this new technology, being smart home management just one of them: the sky is the limit.

Therefore, we want to understand and bring an idea of a use case scenario to life, in which blockchain can successfully solve IoT's problems, with the available resources and open source tools. We want to prove that blockchain is the solution, and while we are at it, propose cheaper architecture solutions for IoT systems. That use case scenario will include the typical sensors and actuators that can be found in a smart home.

In this document the most important decisions are dealt with, in diverse chapters, about the design and implementation of the overall system that includes blockchain, cloud computing and IoT modules. Previously to that development, the document goes over some theoretical chapters on those three trends. The document ends with followed by a conclusions chapter and another one on future lines as an end.

Keywords: Blockchain, bitcoin, cryptocurrency, decentralized, cloud computing, Internet of Things, sensors, actuators, web, scalability, modular, open source, application.

Index

List of figures

Acronyms

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
ACK	Acknowledgement frame
AMQP	Advanced Message Queuing Protocol
AP	Access Point
API	Application Programming Interface
AWS	Amazon Web Services
BLE	Bluetooth Low Energy
CoAP	Constrained Application Protocol.
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
D&C	Divide and Conquer
DHCP	Dynamic Host Configuration Protocol
DIOT	Decentralized IoT
DNS	Domain Name System
EC2	Elastic Compute Cloud
ELB	Elastic Load Balacing
GPIO	General Purpose Input/Output
GUI	Graphic User Interface

HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
LED	Light Emitting Diode
LoRa	Long Range
LPWAN	Low-Power Wide Area Network
LTE	Long-Term Evolution
M2M	Machine to Machine
MQTT	Message Queue Telemetry Transport
MVC	Model, View and Controller
NFC	Near Field Communication
OS	Operative System
PaaS	Platform as a Service
PAN	Personal Area Network
PAN	Personal Area Network
POJO	Plain Old Java Object
PoS	Proof of Stake

PoW	Proof of Work
REST	Representational State Transfer
RF	Radio Frequency
RFID	Radio-Frequency Identification
RGB	Red, Green, Blue
RPL	Routing Protocol for Low-power and Lossy Networks
RX	Reception
SaaS	Software as a Service
TCP	Transmission Control Protocol
TFM	Trabajo de Fin de Máster (End of Master's Degree Project)
TPS	Transactions Per Second
TX	Transmission
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
UI	User Interface
USB	Universal Serial Bus
UUID	Universal Unique Identifier
WAN	Wide Area Network
WSN	Wireless Sensor Network
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

1. Introduction

This first chapter will deal with state-of-the-art issues and the basic ideas around the scope of the project. We will go over the context of the project, its objectives and how the document is structured, in order for a better understanding.

1.1 Context

This project deals with a lot of different technologies, so there will be chapters in each one of them for more detailed contexts and basic concepts. That is one of the main features of the project, the excitement and benefits of dealing with different aspects of modern technologies. We have to cover the fundamentals of a complete software environment, from low to high level and from non-technical to technical aspects.

The first big concept, and arguably the most innovative of all of the ones we will go over is Blockchain. It allows for decentralized applications in which no authority can control or manage them, as they are community-driven by applying consensus algorithms that force a single source of truth of the data in every node that participates in a blockchain network. It could be called a decentralized database, but it has features that go beyond the database definition, it is actually a much more complex technology. Blockchain's most innovative feature is the fact that it eliminates the middle-man so that the cost of the service is much cheaper: imagine international money transfers for cheaper rates (no bank would ask for money, maybe only a network fee), AirBnB without fees so that 100% of the rent would go to the home owner, Poker and online gambling in which no casino keeps part of the winning prices, and the list goes on as new unthinkable applications are coming up.

After that, as a second theme of this project, we will talk about the Internet of Things (IoT). Ever since the appearance of Internet the world has become an interconnected society of information. Human kind have always had a strong need to communicate, and when we managed to do so we came up with better ways to do it in terms of efficiency and speed. From smoke signals to phone calls, we have experienced a lot of new ways of communicating, being the last one Internet, but now it is the turn of IoT. The moment in which machines need to communicate with each other without human interaction has arrived, and the applications are indeed endless. Any device that has wireless or just Internet connection has the ability to transmit and/or receive data, making that device "smart" and this allows for applications just as smart homes, wearable devices, smart cities in which traffic is automatically regulated and all sorts of data can be taken from the physical environment, smart grids that register energy consumption, supply chains that automatically register shipments and state of items, machines that figure out whether crops need watering or any kind of need… But current protocols that are used for IoT have clear security and scalability issues. Integrity is compromised as IoT devices are usually low-power oriented and do not have enough computing power for encryption and it is usually too much data for servers to handle, therefore compromising scalability as well. [1]

Finally, the third and last main theme of the project is brought by cloud computing. It is more known than IoT, and certainly more than blockchain. This technology has existed for a few years already, but that does not mean it cannot have a revolution for itself. The Internet first started with a few computers that were able to communicate with each other, therefore building a decentralized network, but as web services started to appear, the need for computing power and other non-functional requirements such as scalability, resilience, availability and traceability started to be a must-have for these services. Big companies like Google or Amazon started to build powerful infrastructures of servers that offer IaaS, PaaS and SaaS (Infrastructure, Platform and Software as a Service, respectively) so that the necessary requirements for web

applications could be fulfilled. Users can just use these ubiquitous services with Internet access, and developers can host their services in the cloud for everyone to use.

What if we combined blockchain, IoT and cloud computing? We will dive in these concepts on more detail throughout the document.

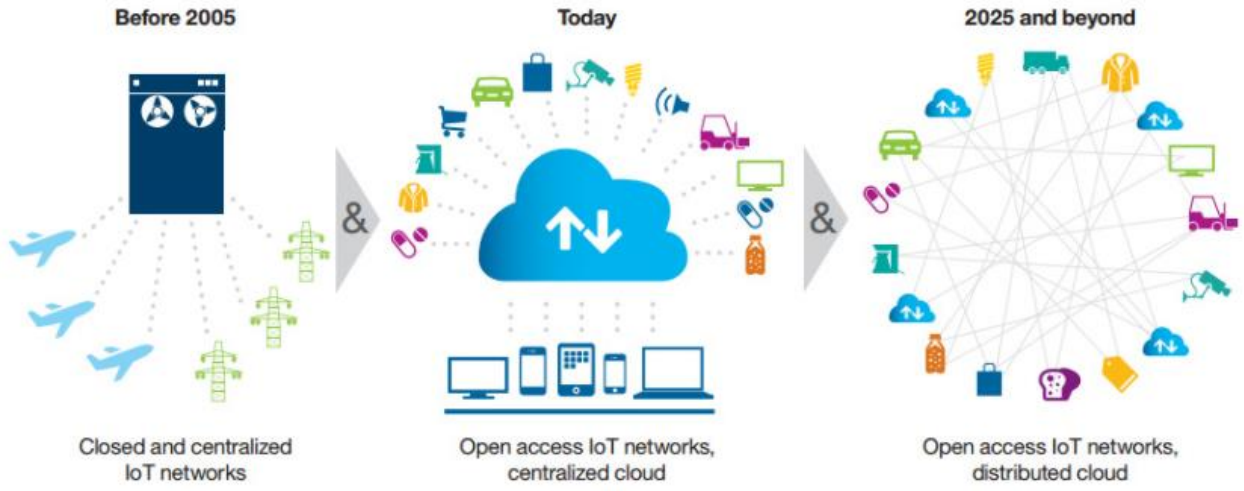


Figure 1: IoT evolution over time

1.2 Project objectives

This project, in a nutshell, aims to provide a deep conceptual study of the technologies mentioned in the previous section as well as develop the idea of solving IoT's security and scalability issues with a blockchain-based solution, also with the use of cloud computing techniques. After these concepts are laid out, a second part of the project follows the first one with the design and implementation of a real blockchain platform, built with cloud services that manages real sensors and actuators.

More in detail, we intend in the present document to give some in depth knowledge on how blockchain, IoT and cloud computing work, as well as listing and describing their systems, protocols and most known providers. We will go over the advantages and disadvantages of the

available technologies in the space, and historical data that can show the growth or other interesting facts on them.

After that, as the second section of the project, we will expose the problems that IoT faces and propose a blockchain-based solution that has been designed and developed for this project. We will therefore demonstrate in that practical scenario a proof-of-concept solution in the form of a smart home manager application that could potentially solve IoT's problems. This application controls sensors and actuators via a Graphical User Interface (scaled to be highly available) that interacts with a blockchain, which keeps record of data in a fixed and secured distributed ledger and also works as the backend logic of the service.

1.3 Document structure

The present document divides the information in a friendly manner so that the reader can easily understand, in the right order, the technologies that are being used, many of their aspects and later the design and development of the smart home manager that utilizes these technologies. Thus far we are ending Chapter 1, which covered the introduction of this project.

Chapter 2 will start off by introducing blockchain, what it is and how it works. That will be explanations about the consensus algorithms, decentralized nature and the most known blockchains, as well as going over the one that we will use in the practical use case. This chapter also defines a skeleton for the following two chapters.

After that, Chapter 3 will do the same for IoT. It will go over what it is, how it works and the most common protocols that exist in the space. It will also feature a subsection that goes more into detail about the problems it is facing.

Furthermore, Chapter 4 will deal with cloud computing. Similarly, we will see what it is and how it works, and later the most common providers will be laid out. This will be a shorter

chapter as this is an already known technology, whereas extremely important in the scope of the project.

After that, the last big chapter is Chapter 5, which will be the longest one as it deeply deals with the practical use case that involves many of the concepts that were laid out in the previous chapters. It will be broken down into subsections that explain the motivation, system overview and high-level architecture modules: blockchain, cloud computing and IoT systems. Each one of them will go over design and implementation with help of diagrams and arguments that back up any design decisions.

Finally, the document ends with two chapters: conclusions and improvement suggestions. These will review the whole project, what was actually accomplished and some good ideas on future developments that could use this project as a reference.

2. Blockchain

In this theoretical chapter, a general view of what blockchain is about is going to be covered. As we will do with the rest of the theoretical chapters, we will talk about what this technology is, its basics on its functionality. After that, we will talk about concrete blockchain applications like Bitcoin and Ethereum, and then discuss some other blockchain platforms and use cases.

2.1 Introduction

A lot of software developers have tried to implement electronic cash ever since the 1990's, as in a digital currency that has no physical backup and could be used for electronic payments, but they could not find a way to solve the double-spending problem. It is defined by a scenario in which the same amount of money can be spent twice (or more) by just duplicating the digital data that represents that money, or somehow maliciously tamper the digital records that represent money balance. [2]

We have ways of make payments via Internet with currencies that are physically backed and avoid the double-spending problem. It is accomplished by third parties, such as banks, that verify transactions themselves and therefore avoid malicious transactions. This is the centralized approach, which involves trusting a third party that represents a single point of failure.

On the other hand, in 2009 **Bitcoin** made its appearance and solved the double-spending problem with a **decentralized** approach [2]. By being decentralized, Bitcoin eliminates the need to trust a third party and is not a single point of failure, as there are many nodes that

participate in the validation process of every transaction. It uses cryptography in order to create transactions and a special protocol to make every node in the network reach consensus and share a single source of truth in the form of a public ledger. Nowadays, this technology is known as **blockchain** because all of the transactions are registered in the ledger packed together in blocks that reference themselves in a chain fashion.

Many other **cryptocurrencies** (digital currencies that exist in distributed ledger technologies) have turned up since Bitcoin's first appearance. While some of them just copied Bitcoin's source code or improved some of its features, most of the rest of the cryptocurrencies serve as tokens to be used in a certain ecosystem powered by blockchain. Blockchain is not meant just for digital currencies anymore, because its capability of storing data in general, the possibilities are endless. Some examples of use cases are decentralizing the Internet (no need for big Internet service providers such as Amazon), decentralizing cloud storage (eliminating intermediaries like Dropbox or Google drive, and achieving much cheaper rates for storage), poker (no fees for casinos), and the list goes on and on. A very special mention is necessary for smart contracts, introduced by **Ethereum**, as they store code in the blockchain along with data and therefore provide the blockchain with intelligence and logic that can be used for building decentralized applications.

2.2 What is blockchain?

While a lot of different blockchains have appeared over the years, the majority follow a common high-level logic that is displayed in the following figure:

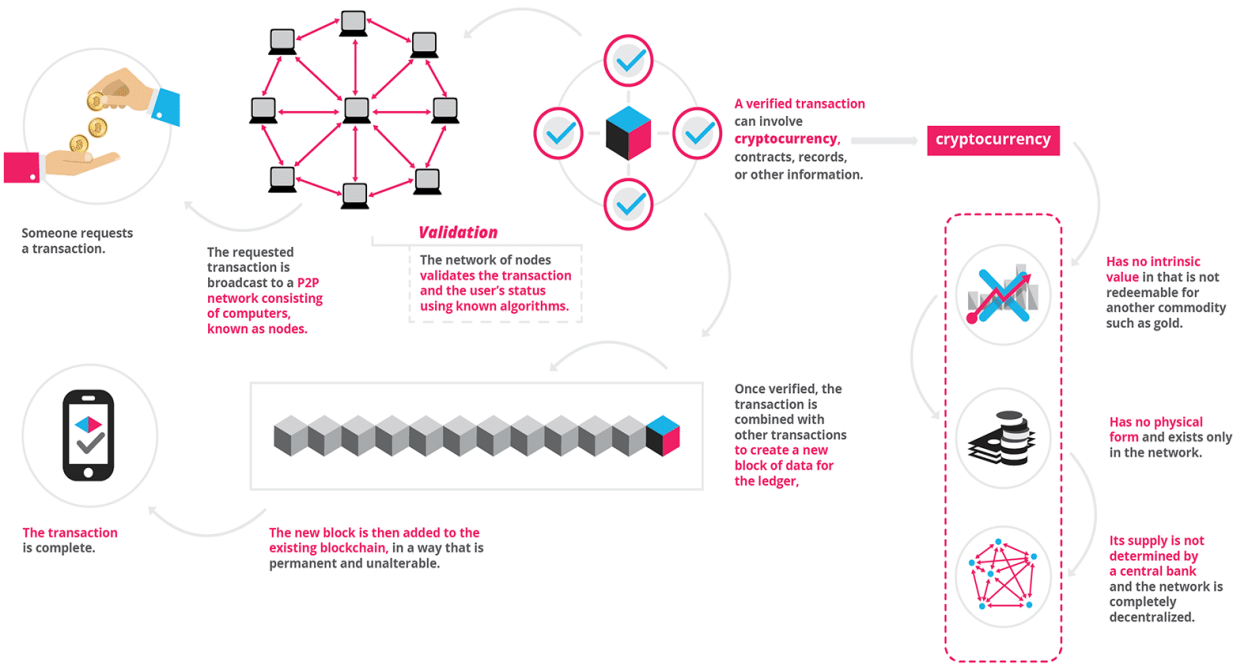


Figure 2: High level functional diagram of blockchain [3]

Taking into account that a blockchain is a distributed ledger, any digital data can be stored or traded in it through transactions. When someone requests a transaction (does so with a private key that enables changes in a specific wallet), it is sent to the nodes that are part of the blockchain, a peer-to-peer network. All of the nodes validate the transaction using algorithms that vary depending on the blockchain (Bitcoin uses Proof of Work, others use Proof of Stake or Delegated Proof of X, etc.), and this transaction is combined with many others and packed together to form a block of data. This block is then added to the blockchain along with a hash number that identifies the previous block.

Since the blockchain is a public ledger, once the transaction is complete it is then published in the blockchain and accepted as valid. Whatever the asset that was traded was, it will stay in the blockchain forever and cannot be tampered with, as it would be necessary to change all of the following blocks and currently it is not possible to hack as many nodes as to make the entire network believe that the blockchain is different than the current one.

One could ask where these nodes are and why they would contribute to the network. Depending on the blockchain, nodes that contribute get something in return, which is called mining fee, a small amount of cryptocurrency or other asset that has been traded and goes to the node that validated that transaction first because it reached that conclusion after solving an algorithm. That incentives people to run nodes and contribute to the blockchain.

2.3 First generation: Bitcoin

"A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone."

- Bitcoin Whitepaper [4]

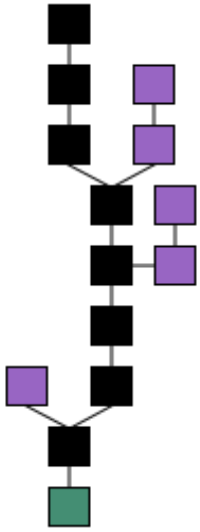
Bitcoin is the first blockchain's use case to ever come out. It made a huge impact on economics, as it came out as the first completely virtual currency that solved the double-spending problem and did not need any regulation in order for it to be secure. In a nutshell, holders of Bitcoin just need a private key in order to access their virtual wallets and send money, also to receive it they just need to share their public address: a hexadecimal number of 26-32 characters.

As an interesting fact, nobody knows who created Bitcoin, and many have tried to uncover his or her identity. What we know is that the person or group of developers that created it go by "Satoshi Nakamoto", so one could think that it was a Japanese invention (he claimed to be a 37-year-old man living in Japan [5]), but the use of perfect English in their whitepaper (published in bitcoin.org) makes that assumption less credible. Famous persons have been said to be Satoshi Nakamoto, like Elon Musk (CEO of Tesla) or Gavin Andresen (a famous developer), but they all denied it.

Moving on how Bitcoin's protocol works, it all starts with the need to build a secure electronic cash system that cannot be tampered with in any way. This is achieved by making a whole network of nodes validate transactions, incentivized to do so by receiving Bitcoin in return. When a certain number of transactions are requested, the nodes pack them together in a block (therefore, **mining** that block) and have to create a complex hash (SHA-256) to seal that block along with a timestamp, and it has to comply with a set difficulty and reference the previous block in the blockchain. This algorithm is called Proof of Work, because it requires computing power and once the hash is achieved, it is easily checked to determine whether it is a valid hash (has to reference the previous block's hash and have a certain length/difficulty) and then the winner node receives the mining fees of the validated transactions. Anyone can set up nodes in their computers to mine Bitcoin and contribute to the blockchain.

This is how the blockchain solves the Byzantine problem, defined by a hypothetical scenario in which every single soldier in an army has to agree to attack at the same time with a signal. The Proof of Work algorithm does not entirely make every node validate transactions at the same time, but it almost solves the problem and it is accepted even if blocks take 10 minutes to be mined and slow down the transaction rate of the network.

If two nodes achieve a valid hash at almost the same time, there is a chance that the blockchain branches out, but all of the nodes in the network will always choose the longest chain as the single source of truth. Any blocks mined in the branch will not be will not be part of the main branch and therefore their transactions have not actually happened. As these transactions would not be published in the public ledger (only the longest blockchain is published), the initiators of the transactions will keep on publishing them so that they can be included in the main branch.



In order to hack the blockchain, more than half of the nodes in the network would have to go cooperate in the attack, and it is currently impossible to have enough computing power to take control over that number of computers. If a malicious transaction takes place (spending money that has already been spent or a bad signature) the nodes will reject it.

2.4 Second generation: Ethereum

"Satoshi Nakamoto's development of Bitcoin in 2009 has often been hailed as a radical development in money and currency, being the first example of a digital asset, which simultaneously has no backing or "intrinsic value" and no centralized issuer or controller. However, another - arguably more important - part of the Bitcoin

experiment is the underlying blockchain technology as a tool of distributed consensus, and attention is rapidly starting to shift to this other aspect of Bitcoin. Commonly cited alternative applications of blockchain technology include using on-blockchain digital assets to represent custom currencies and financial instruments ("colored coins"), the ownership of an underlying physical device ("smart property"), non-fungible assets such as domain names ("Namecoin"), as well as more complex applications involving having digital assets being directly controlled by a piece of code implementing arbitrary rules ("smart contracts") or even blockchain-based "decentralized autonomous organizations" (DAOs). What Ethereum intends to provide is a blockchain with a built-in fully-fledged Turing-complete programming language that can be used to create "contracts" that can be used to encode arbitrary state transition functions, allowing users to create any of the systems described above, as well as many others that we have not yet imagined, simply by writing up the logic in a few lines of code."

- Ethereum Whitepaper [6]

Proposed and co-created by Vitalik Buterin, he thought of the idea of using Bitcoin's blockchain to build decentralized applications and published a whitepaper describing Ethereum back in 2013. The development was funded in the summer of 2014 and the platform went online in the summer of 2015 [7].

Ethereum successfully brought a lot more attention to the blockchain spaced by enabling it to store code aside from assets in the form of cryptocurrency. The pieces of code that are stored in the **Ethereum Virtual Machine** (the decentralized platform for applications and exchange of **Ethers** that is made up of all the nodes contributing to the blockchain) are called **smart contracts**. In a nutshell, it is possible to define "if this happens, then something else

happens” as executable logic like any other platform and programming language. It is possible to develop smart contracts with a new Turing-complete programming language (can potentially solve any computation problem) called Solidity.

These smart contracts have an address, just like any Ether or Bitcoin holder, and to start their functionality it is necessary to power them with “gas”. The gas concept in Ethereum comes from a fuel-like model of powering smart contracts, or literally paying nodes in the blockchain to execute the contracts with their computing power. In order to execute a contract or start a transaction of Ether, the starter of that transaction has to set a gas limit and the price of that gas in “GWei” (one GWei is 10^-9 Ether). The more gas a transaction has, the further it can go as in how many contracts it has to execute, and the more GWei the more expensive the transaction is but the faster it goes as the miners get more Ethers in return.

On the other hand, how Ethereum implements blockchain technology is pretty similar to how Bitcoin does it, with some notable differences aside from smart contracts integration and the way transactions work as explained above: [8]

- Block time: Bitcoin takes 10 minutes for every new blocked to be mined, whereas Ethereum takes around 15 seconds, because Ethereum sometimes allows for branches of its blockchain to be valid and eventually add them to the main chain.
- Dynamic block size. Bitcoin is limited to 1 MB per block, but Ethereum depends on gas-limit and number of transactions.
- Two types of accounts: holder of Ether or a smart contract.
- Different hashing algorithm (KECCAK-256) that allows for more decentralization as it cannot have specialized hardware like Bitcoin’s SHA-256.
- Bitcoin’s network is not as regularly updated as Ethereum. Ethereum is planning to change its consensus algorithm from Proof of Work to Proof of Stake, in which miners

tend to win at mining blocks based on the amount of Ethers instead of computing power. Also, “sharding” is supposed to improve scalability by breaking the blockchain down into many interconnected sub-blockchains.

The most interesting feature of Ethereum is the smart contracts, as they allow for developers to create decentralized applications or “dApps”. The use cases are endless, as long as it makes sense to eliminate the middle man in digital services. We will dive more into detail of some interesting use cases, and we have already mentioned some of them in the Introduction section of this chapter.

2.5 Third generation

As the time of writing this document (April, 2018) there is no obvious third generation blockchain as there are a lot of new projects that seek to revolutionize blockchain protocols. These projects are known as infrastructure coins because they try to implement new blockchain platforms and their development teams usually create tokens with the help of Ethereum smart contracts and sell them in an Initial Coin Offering (ICO) in order to fund the projects. People are usually interested in buying these tokens for speculative purposes and invest real money in exchange for them.

As a measure of how important these projects are, we can take a look at coinmarketcap.com and see the most valued cryptocurrencies right now:

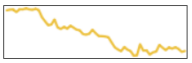
#	Name	Market Cap	Price	Volume (24h)	Circulating Supply	Change (24h)	Price Graph (7d)
1	Bitcoin	$118.429.190.195	$6.986,70	$4.330.160.000	16.950.662 BTC	-1,17%	
2	Ethereum	$39.139.038.528	$397,19	$1.183.020.000	98.540.335 ETH	-3,29%	
3	Ripple	$19.949.425.645	$0,510287	$246.398.000	39.094.520.623 XRP *	-2,42%	
4	Bitcoin Cash	$11.834.337.446	$694,15	$298.172.000	17.048.650 BCH	-4,14%	
5	Litecoin	$6.532.211.785	$116,90	$272.156.000	55.881.020 LTC	-4,60%	
6	EOS	$4.581.750.338	$6,00	$202.111.000	763.256.149 EOS *	-2,67%	
7	Cardano	$4.095.569.698	$0,157965	$112.283.000	25.927.070.538 ADA *	4,97%	
8	Stellar	$3.889.354.666	$0,209662	$85.560.500	18.550.594.129 XLM *	8,03%	
9	NEO	$3.256.721.000	$50,10	$62.197.700	65.000.000 NEO *	-3,90%	
10	IOTA	$3.023.072.726	$1,09	$26.950.800	2.779.530.283 MIOTA *	-4,60%	

Figure 3: Top 10 cryptocurrencies by market capitalization on April 1st, 2018 [9]

We could consider that the next generation of blockchains are the infrastructure-blockchain projects that are in the Top 10 right now (aside from Ethereum):

- EOS: promises to be more scalable, modular, easier to work with for developers and free transactions.
- Cardano: wants to improve scalability (transaction speed, bandwidth, block size), interoperability (work with different blockchains) and sustainability.
- Stellar: faster transaction speed, interoperable, unique consensus protocol, simple to use.
- Neo: Delegated Byzantine Fault Tolerance as consensus algorithm, a thousand transactions per second, possible to reach even more speed.

2.6 Interesting use cases

In this section we will go over some very interesting use cases of blockchain. These will be projects that are already out in the market or in development process. Most of them are built in Ethereum, creating a cryptocurrency with the help of a smart contract, so as to collect money for the project and also help promote it as trading with cryptocurrencies is one of the best marketing strategies.

As there are more than 1500 cryptocurrencies at the moment [9], and the projects that use blockchain but have not launched a cryptocurrency, it is impossible to go over all of them. We are going to cover some projects that have been in development for a while now or that already have a working product.

2.6.1 Substratum, the decentralized Internet

"Substratum is creating an open-source foundation for a decentralized web which will provide unrestricted access to content and sharing of information for users across the globe. Our mission is to bring forth the free and fair internet of the future by combining proven technological building blocks with emergent technologies in an innovative and holistic way to help solve many of the problems that plague the modern internet. Substratum will gain mass adoption by revolutionizing the hosting industry with per request billing via micro-transactions and incentivizing users to run nodes to create the network by paying node hosts in Substrate. This is all being managed by blockchain technology and machine learning."

- Substratum Whitepaper [10]

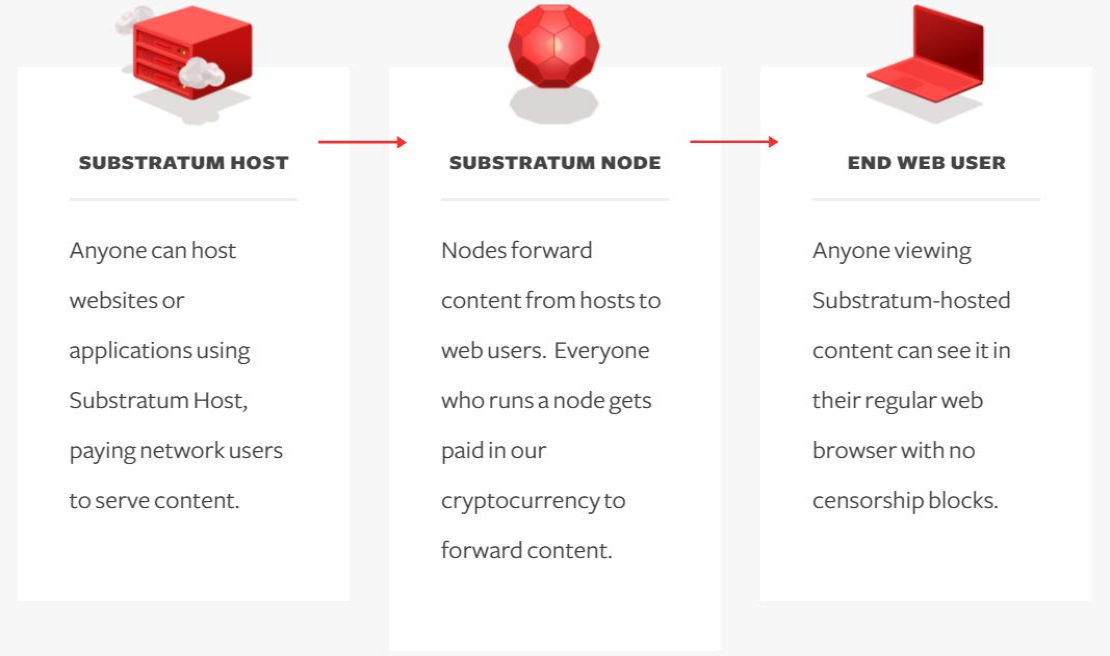


Figure 4: Substratum tiers [11]

As can be read in their whitepaper, the Substratum team tackles some big problems that the Internet has nowadays. It is restricted in many countries by their governments, censored or even prohibited. As well as banks, the current Internet model is centralized in big companies like Google or Amazon, which almost make a monopoly and control most of the servers worldwide.

How does eliminating the middle man help the Internet? Substratum promises much cheaper hosting prices for websites (that is computing power, storage, bandwidth, etc.), no third party would be able to censor anything or prohibit the use of Internet as all the traffic would be highly encrypted and it would probably enhance the loading speed of websites as there is a machine learning algorithm that would load the content from the closest hosting nodes.

The Substratum network uses blockchain in order to build a peer-to-peer Internet. The hosting nodes will be incentivized by receiving Substratum tokens (their cryptocurrency built in Ethereum) in return for computing resources, thereby “renting” computers. These nodes will deliver encrypted content similarly to VPNs and will be paid according to the number of

transactions that they dispatch. The Substratum team is currently developing the system and will release an application that is as simple as just running it and setting how much of each resource it is desired to give to the network. The same software also works for the user to access the Substratum network.

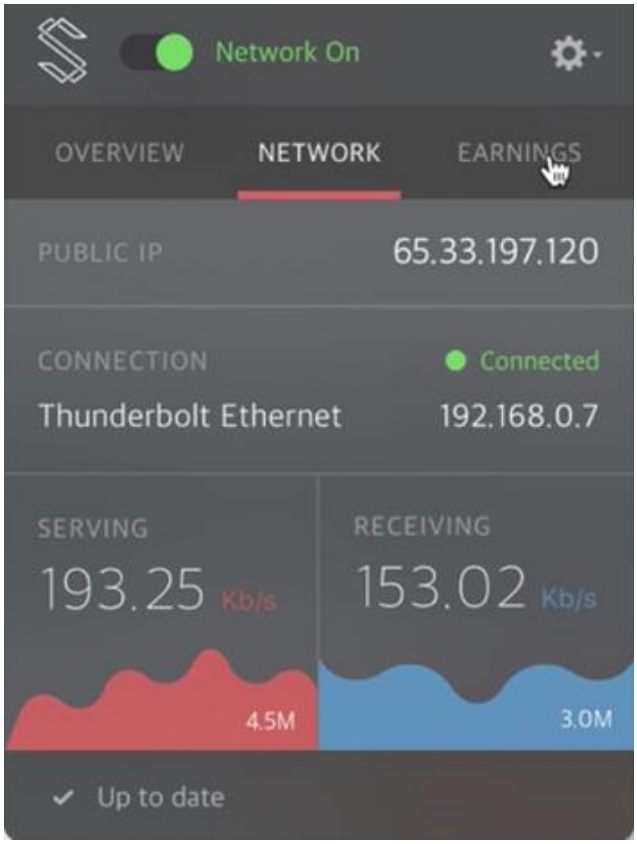


Figure 5: Substratum desktop application [11]

2.6.2 Sia, decentralized cloud storage

"The authors introduce Sia, a platform for decentralized storage. Sia enables the formation of storage contracts between peers. Contracts are agreements between a storage provider and their client, defining what data will be stored and at what price. They require the storage provider to prove, at regular intervals, that they are still storing

their client's data. Contracts are stored in a blockchain, making them publicly auditable. In this respect, Sia can be viewed as a Bitcoin derivative that includes support for such contracts. Sia will initially be implemented as an altcoin, and later financially connected to Bitcoin via a two-way peg."

- Sia Whitepaper [12]

Developed by Nebulous Inc., Sia promises a decentralized cloud storage with many advantages over the current solutions. It actually came out to the public before Ethereum, so they are only based on Bitcoin's blockchain but also implement smart contracts in order for the business to work.

The way Sia intends to work (it is actually working at the moment but not fully implemented yet) is encrypting and replicating whatever files are desired to be store throughout a peer to peer network of nodes that will host those files (encrypted, replicated and broken down into many pieces) in return of Siacoins, their own token that powers the platform. These nodes accept a smart contract that states a periodical proof of storage from them, and they will receive Siacoins if they comply with that. On the other hand, the users that want to upload files will do so by paying Siacoins to the network and will hold a private key that can manage those files. When a user wants to download a file, he or she will use the private key and the file's chunks will be pulled from the nodes so that they can build it.

The advantages of this technology, among others, are the price, availability and resilience. It is claimed that it costs 2$ to store 1 TB of data in Sia, whereas it costs 23$ to store the same amount of data in Amazon S3. On the other hand, the fact that the service is decentralized and the data is replicated, there is no single point of failure that could make the service fail, so it is both resilient (robust to network or node failures) and available (could potentially have a much higher up-time than any centralized cloud storage service).

2.6.3 Bee, decentralized home sharing

"Beenest is the first decentralized home-sharing network built on top of a set of Bee Protocols that connects hosts with guests without taking any commissions. […] *These protocols provide the Beenest network with three essential systems:*

> *1. A secure payment system that allows two authenticated P2P entities to send and receive money that gets held in Bee Tokens until after a successful exchange of services between the two entities.*
>
> *2. A decentralized arbitration system that resolves user disputes, providing positive incentives to grow a network of genuine arbiters and negative incentives to deter scammers.*
>
> *3. A reputation system that couples a valid identity, which is obtained by a trusted digital fingerprint protocol on the Ethereum blockchain, with a rating determined by transparent, immutable review and scoring interchange between P2P entities (such as guests and hosts).*

- Bee Whitepaper [13]

As their whitepaper says, Bee is a decentralized home-sharing network, which is built as a compound of Ethereum smart contracts powered by the Bee Token. It plans to be the Airbnb killer as it cuts the middle man in the home-sharing process.

They have a very complex method of achieving a home-sharing service that can work without a middle-man and at the same time solve any problems that tenants and landlords may have. It is divided up in three protocols: first, the payments are 100% done via blockchain, in a

peer-to-peer network and with the use of Bee Tokens (cryptocurrency built in a Ethereum smart contract). Second in case of any problems between clients of the service, there is a arbitration system that holds stakes from the participants, and a decision is made with the help of a vote in the community. This vote takes us to the third part of the Bee protocol, which is a rating system that gives each user a rating based on disputes and/or past experiences without disputes.

The Bee team have a well-defined roadmap in which they start their service in San Francisco, and plan to expand in the upcoming years in hopes of reaching Airbnb's market.

2.7 Hyperledger

As a final subsection to this blockchain chapter, we have to cover Hyperledger, as it is one of the technologies we will be implementing in this project's use case (the reasons for this choice will be explained in the use case's development chapter). In a nutshell, Hyperledger is a compound/umbrella of open source projects around building a blockchain environment. There are provisioning systems, frameworks, platforms and many more under the name Hyperledger. [14]

It is a global collaboration hosted by the Linux Foundation, which considered that the blockchain technology is the biggest revolution ever since the Web itself. Blockchain's distributed ledger nature, thanks to the peer-to-peer network that reaches consensus with smart contracts, makes it possible to build transactional applications that do not need a third party or bigger authority to gain trust, accountability and transparency.

Hyperledger brings an open source approach that promises all those characteristics in blockchain-based applications that use their products, along with a long-awaited mass adoption

of the technology. Despite most blockchain implementations in the space, Hyperledger does not introduce a cryptocurrency that powers the ecosystem as it found its way around that incentive for computing contribution from the member nodes of the network. Another big difference with most blockchains is the fact that Hyperledger introduces a permissioned blockchain, so as to respect transactions' privacy and enable roles in the network.

The compound known as Hyperledger started in 2016 [14], with 30 founding corporate members and a well-established governance structure. The first projects to became part of it were Hyperledger Fabric and Hyperledger Sawtooth. Over 2016 and 2017, the Hyperledger project grew up to 200 members and 70+ open source organizations.

As their most important framework projects, we have: [15]

Figure 6: Hyperledger Sawtooth

Figure 7: Hyperledger Iroha

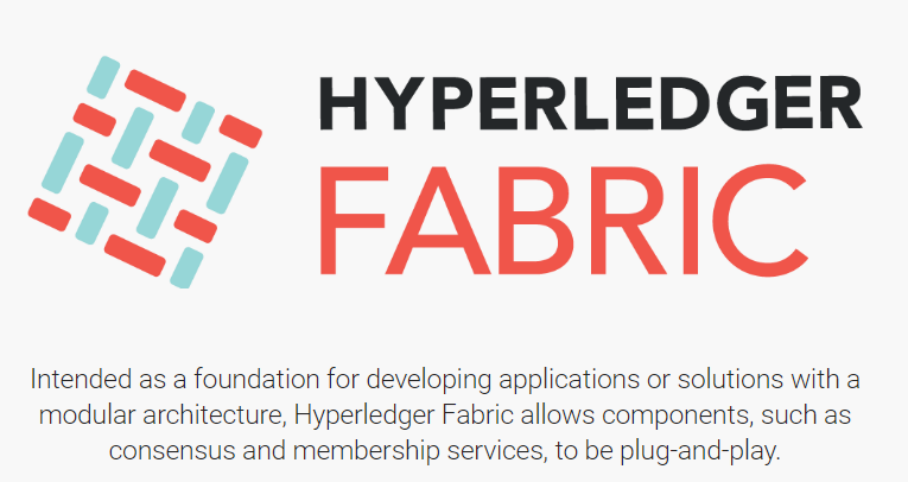


Figure 8: Hyperledger Fabric

Figure 9: Hyperledger Burrow

Figure 10: Hyperledger Indy

On the other hand, there are also tool projects that do not provide an infrastructure or blockchain system but provide help in order to manage or develop in it: [16]

- Hyperledger Caliper: serves as a health checker for a deployed blockchain. Able to generate reports on metrics like transactions per second, resource utilization, number of nodes, etc. It is contributed by Huawei, Hyperchain, Oracle, Bitwise, Soramitsu, IBM and the Budapest University of Technology and Economics.
- Hyperledger Cello: provisioning tool that automates deployment of blockchain, pretty much like Amazon Web Services does for cloud computing. Also lets operators check the status of blockchains and update parameters. Initially contributed by IBM, with sponsors from Soramitsu, Huawei and Intel.
- Hyperledger Composer: set of tools that allows developer an easier approach to development of smart contracts on top of Hyperledger Fabric in Javascript. It will be the tool that we will be using in the practical use case of this project, as it provides a very friendly environment and less steep learning curve in blockchain applications development. It also automates the creation of a REST API that interacts with the

deployed smart contract using HTTP requests, which makes it much easier for web applications to interact with it. Contributed by IBM.

- Hyperledger Explorer: lets users interact with a blockchain with queries through a web application. Initially contributed by IBM, Intel and DTCC.
- Hyperledger Quilt: offers interoperability between distributed ledgers and non-distributed ledgers, so that transactions of assets can happen between any type of ledgers in an atomic way. Originally contributed by NTT Data and Ripple.

3. Internet of Things

In this theoretical chapter, we are going to cover the essentials of the Internet of Things (IoT), an emerging technology that is gaining a lot of media attention lately.

3.1 Introduction

As many more devices are able to communicate with the Internet, society can no longer ignore this tendency and realized what its use cases are, just like it happened with blockchain. We will go over what IoT is, talk about those uses cases and which are the most common protocols. In the end, we will address the scalability and security problems that IoT is facing.

These devices that are able to communicate with the world wide web increased 31% from 2016 to 2017, and it is expected to consist of 30 billion devices by 2020. [17] IoT networks allows for sensing physical environment and actuating so that it changes. The applications for this are endless: automation of daily tasks in smart homes, supply chains, traffic surveillance and adjusting (smart cities), data collecting for analytics purposes, etc.

3.2 What is IoT?

IoT stands for Internet of Things, as it addresses the ability of “things” in general to access the Internet to transmit data or receive it so that the “thing” can do something as a consequence. IoT consists of a network of billions of devices embedded with electronics and software capable of communicating, usually **low-energy** devices that are limited to sensing and

being controlled remotely via network infrastructure, therefore reducing human intervention as much as possible with the automation of daily tasks or new applications. Examples of IoT devices are live cameras, biochips, vehicles with sensors, thermostats, voice assistants, smart meters, wearable devices, door locks, light bulbs, smoke detectors, ovens, toasters, refrigerators, automatic windows, and the list goes on. Basically, any device that is able to connect to the Internet and does not have computing resources comparable to a computer, can be called an IoT device. For example, a smartphone is not an IoT device because it has too much computing power and its battery cannot last for longer than a day or a few days, and a smoke detector in a forest can stay on for years (especially if it has solar panels), but the smartphone can participate in an IoT environment because it can access an IoT network and control it or just browse its data. Typically, devices that are part of an IoT application are called **smart** devices.

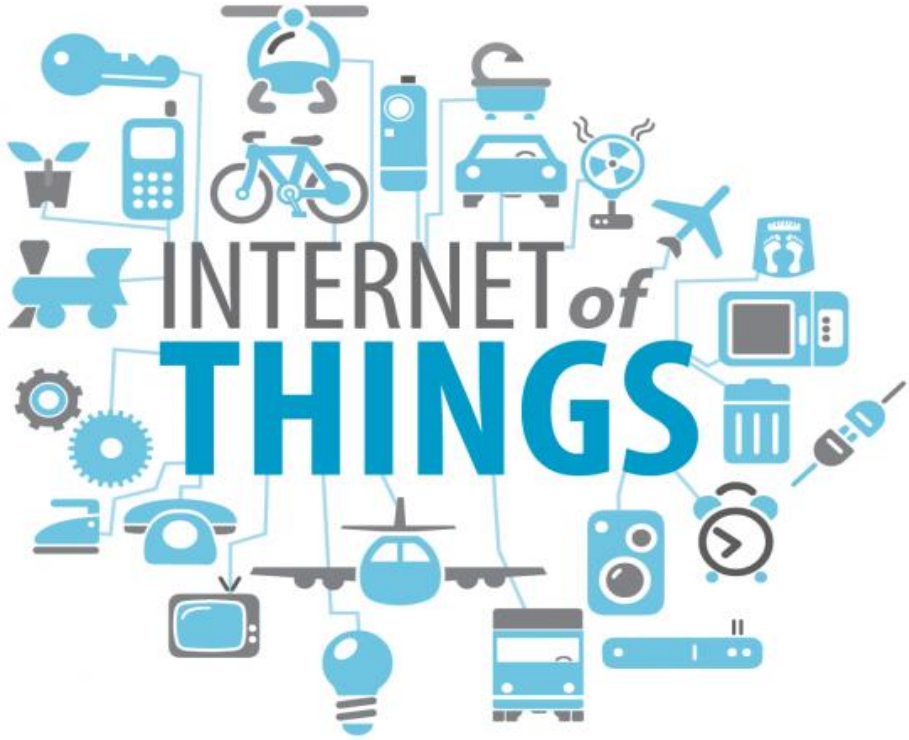


Figure 11: IoT's general idea diagram [18]

The general idea in every IoT application is collecting data for useful purposes that usually involve Big Data techniques in order to analyze the data and extract facts or patterns in it that can help businesses. Other applications or the same ones can also include control over those devices in case it is necessary to take physical actions on the controlled environment. For instance, a smart home user could set the temperature of the thermostat with a smartphone, or let it adjust itself automatically according to the time of the year or the day (higher temperatures for summer for energy saving, turn off when that person is not home…). All these cases share a sensor network as the first part of their infrastructure solutions.

There are different sensor networks (Layer 1) according to their range:

- PAN: Personal Area Network. Distances measured in meters, for scenarios like Bluetooth devices sending data to each other, or a mesh network of devices that route data to a gateway.
- LAN: Local Area Network. Short or medium range, also measured in meters. Can also be PANs that communicate with each other via a gateway or wired connections in a wider environment.
- MAN: Metropolitan Area Network. Long range, city-wide applications so that distances are now measured in kilometers. Major use case is smart cities.
- WAN: Wide Area Network. Even longer ranges, usually WANs are used in scenarios like smart agriculture or climate monitoring, which use big mesh network of sensors and actuators that need to be very far from each other.

Currently, IoT's solutions share a common infrastructure distribution, with a physical layer that involves sensors and actuators, a concentration layer that serves as a gateway between those devices and the Internet, and a final cloud layer that gathers data, performs

analytics on it and executes business logic. The following figure shows this infrastructure distribution, dividing the cloud computing layer in stages 3 and 4:

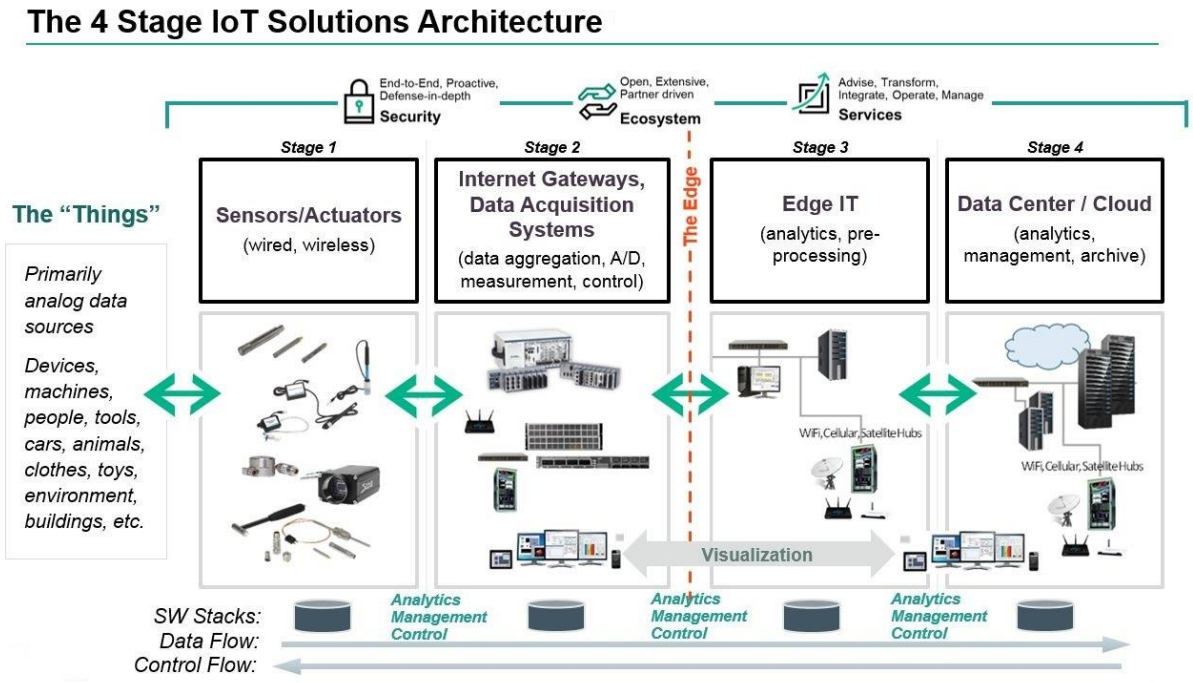


Figure 12: IoT solutions architecture [19]

Layer 1 will usually be a Wireless Sensor Network (WSN) made up of sensors and actuators that collect raw data but can also include wired devices as well. Defined as the "Things" part of IoT, send and receive data or form the Internet via the second layer that gathers the information. Then, layer 3 preprocesses the data, packs it together and may perform analytics, but will usually forward it to a fourth layer that manages and store the data (may also analyze it) in the cloud.

3.3 Use cases

IoT is a growing market that many people usually accuse of not having real and valuable use cases that justify the need. It is a tendency in the market and it is growing exponentially every year. The following figure shows us some of the most common use cases:

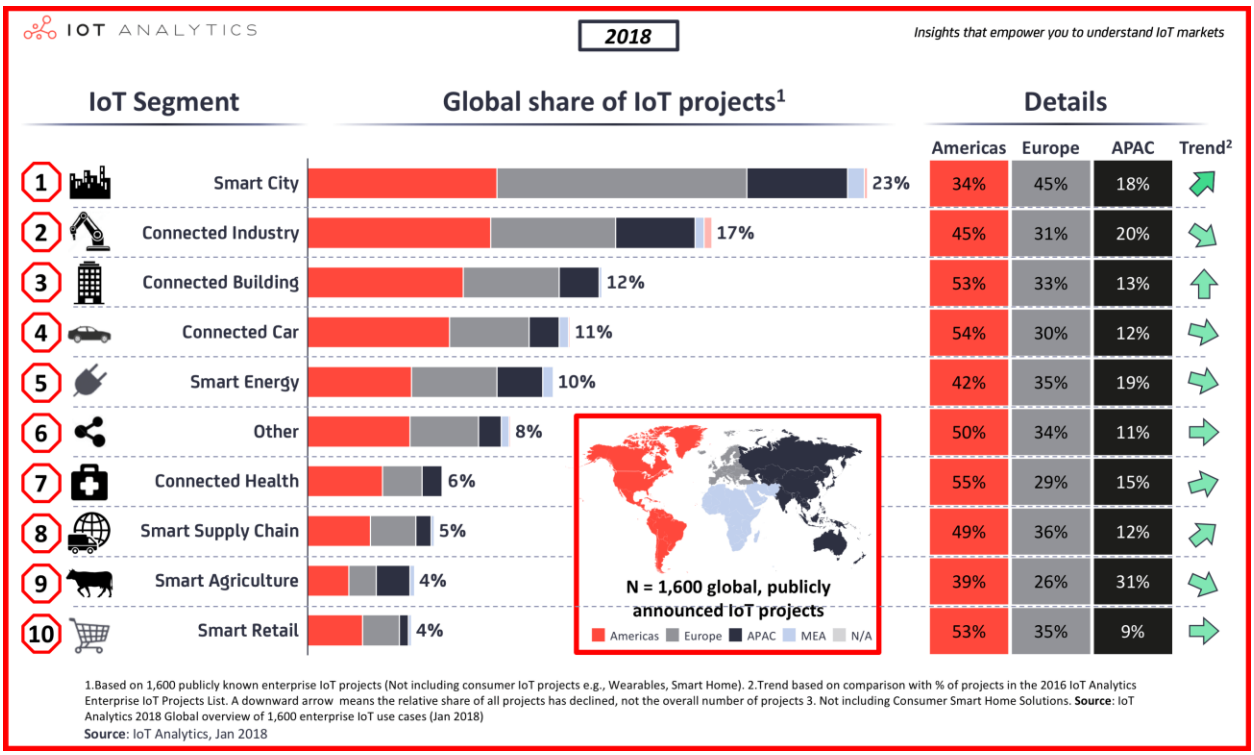


Figure 13: IoT Enterprise Projects 2018 [20]

"Connected Industry" is not really a very interesting use case, as it is relatively old. It consists of automating more tasks than usual, like remote control of machinery or workers using wearables. Also, "Connected Building" should be considered as a superset of smart home. Aside from these comments, we can proceed to give some explanations on some of the most famous use cases:

- Smart City: as the biggest segment of the IoT market, smart cities have systems that monitor traffic and adjust it accordingly, helping people to park by identifying free spots, changing traffic lights and also ways of ensuring public safety. Smart cities are very different from one another and may involve different IoT applications that together form the definition of a smart city.
- Smart Agriculture: food supply is a very serious concern for the future, as Earth's population is rapidly increasing. IoT helps increase food supply by automating tasks in agriculture, and also increasing efficiency by sensing useful data that can tell where crops will grow better or when to do it by analyzing climate parameters.
- Smart Supply Chain: one of the first thought use cases. Supply chains are easily enhanced with the help of sensors and actuators that help track, route and register assets in an inventory. IoT can provide loss prevention and ease registering with, for instance, RFID tags that hold the necessary information to identify items.
- Connected Cars: remote control of vehicles is possible with antennas in them, and a set of sensors can track and monitor the vehicle for data analysis purposes or know how an accident happened. A set of sensors and actuators is obviously required for autonomous cars too.
- Smart Grid: the electricity system is mostly the same since the 1890s. With a new IoT system, electricity is transmitted more efficiently, being faster, reducing peak demand and therefore decreasing the prices. It consists of computers, automation and a network of sensors that work together in order to respond quickly to changing electric demand.
- Smart Home: also known as home automation, smart house and domotics, it is one of the most known use cases of IoT. A smart home is an IoT environment that is full of sensors and actuators in a wired or wireless sensor network that allows its users to fully control lights, entertainment systems, thermostat, air conditioning and many appliances.

The uses are endless, as many things in a home can be automated. It is the use case we tackle in the practical part of this project.

3.4 Most common protocols

One of the biggest problems of IoT is the lack of standardization in most of its aspects. It is very notable when it comes to protocols in all layers. Some layers are more standardized than others, especially once the data is outside of the IoT environment and in the Internet, but protocols are not standardized at all in the physical part of an IoT environment: sensors and actuators communicate with each other in many different ways depending on who built the system. Moreover, those protocols can take care of more than one layer in the same network of sensors. We are going to go over some of the most used or promising protocols: [21]

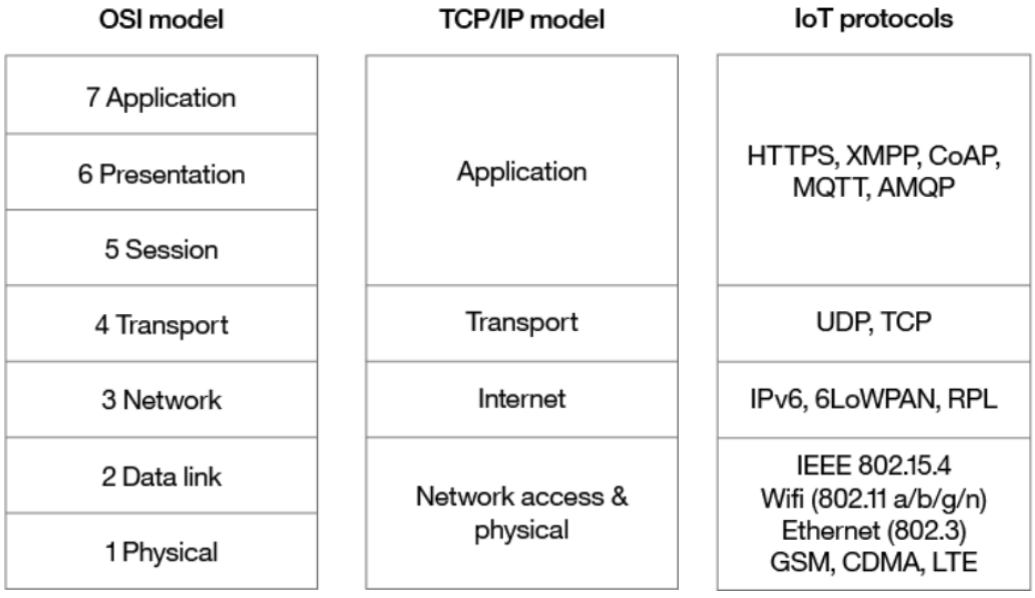


Figure 14: From OSI protocol stack to IoT's TCP/IP stack

- Application Layer: It is the highest-level layer, and it is unique to the applications that need to communicate with each other. HTTP/S are widely used protocols among internet applications, and can perfectly be used for IoT, but some new protocols are uniquely used in this scope:
    - CoAP: Constrained Application Protocol. Designed for machine-to-machine (M2M), it is based on HTTP RESTful model, with methods such as GET, PUT, POST, and DELETE, but aimed for low-power devices.
    - MQTT: Message Queue Telemetry Transport. This protocol uses the design pattern of publishers and subscribers, designed for low bandwidth or unreliable networks.
    - AMQP: Advanced Message Queuing Protocol. Open standard messaging protocol for message-oriented middleware. Supports point-to-point and publisher/subscriber models, and it is designed to have a robust reliability and security.
    - XMPP: Extensible Messaging and Presence Protocol. It was originally designed for human-computer interaction, but it has been adapted to be a lightweight communication protocol.

- Transport Layer: There is no real innovation here, as the protocols TCP and UDP are very old, widely known and used. TCP allows for more reliable communications but it is slower than UDP, which does not mind when some packets of data are lost, but it is a tradeoff for speed. There are hybrid solutions that can keep the network both fast and reliable, but it is always at the cost of one or the other.

- Internet Layer: This layer is in charge of routing packets of data throughout the internet. The most common protocols for IoT are:
    - IPv6: famous internet Layer. Uses longer IP addresses over legacy IPv4. Necessary for IoT applications as the number of devices is increasing exponentially, so they need a lot more different addresses.
    - 6LoWPAN: IPv6 Low Power Wireless PAN. This protocol allows for IPv6 in IEEE 802.15.4 wireless networks.
    - RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. A different protocol designed to route packets over low-power and unreliable networks. Calculates the optimal path to the destination taking into account many factors as predicted loss, distance or bandwidth.

- Network and physical Layer: End of the protocol stack, this layer contains all the protocols that enable physical access to the sensor networks.
    - LPWAN: Low Power WAN. Category of technologies that are designed for low-power and long-range communications. Wireless and designed for very wide area applications, LPWAN includes LoRa, Haystack, SigFox, LTE-M, and NB-IoT (Narrow-Band IoT).
    - Cellular: Mostly legacy networks. 2G, 3G and 4G are all operational, so they can be used for IoT purposes, along with LPWAN solutions. These protocols are adequate for long-distance IoT applications.
    - BLE: Bluetooth Low Energy. Low-power version of classical Bluetooth. Not suitable for file transferring as it is aimed for low-power applications (and therefore lower bandwidth). Can reach up to 100 meters, making it a PAN, and it is usually embedded in wearables.

- ZigBee: A little longer range than BLE, but lower data rate. It is a mesh network aimed for home automation and give roles to the sensors in the network depending on their location: they can be controllers or routers of the data. Based on IEEE 802.15.4
- NFC: Near Field Communication. Used for very short-range applications (4 cm), usually payment systems or smart labels
- RFID: Radio-Frequency Identification. Subset of NFC, RFID is based on tags that have no power at all, but just an electronic system that responds to electromagnetic stimulus by sending back certain information. Ideal for identifying items in supply chain applications.
- Wi-Fi: Widely known standard IEEE 802.11a/b/g/n. Very high data rates and medium-short ranges, but the low power competitors will most likely be used more for IoT applications.
- Ethernet: Wired standard for LANs, IEEE 802.3. Ideal for wired networks as the devices will usually have a permanent power supply.

3.5 Current problems

As good as IoT may sound, there are a lot of issues that many engineers are working on. In this subsection we are going to cover some of the most important problems that IoT is facing: scalability security and lack of standardization (interoperability).

3.4.1 Scalability

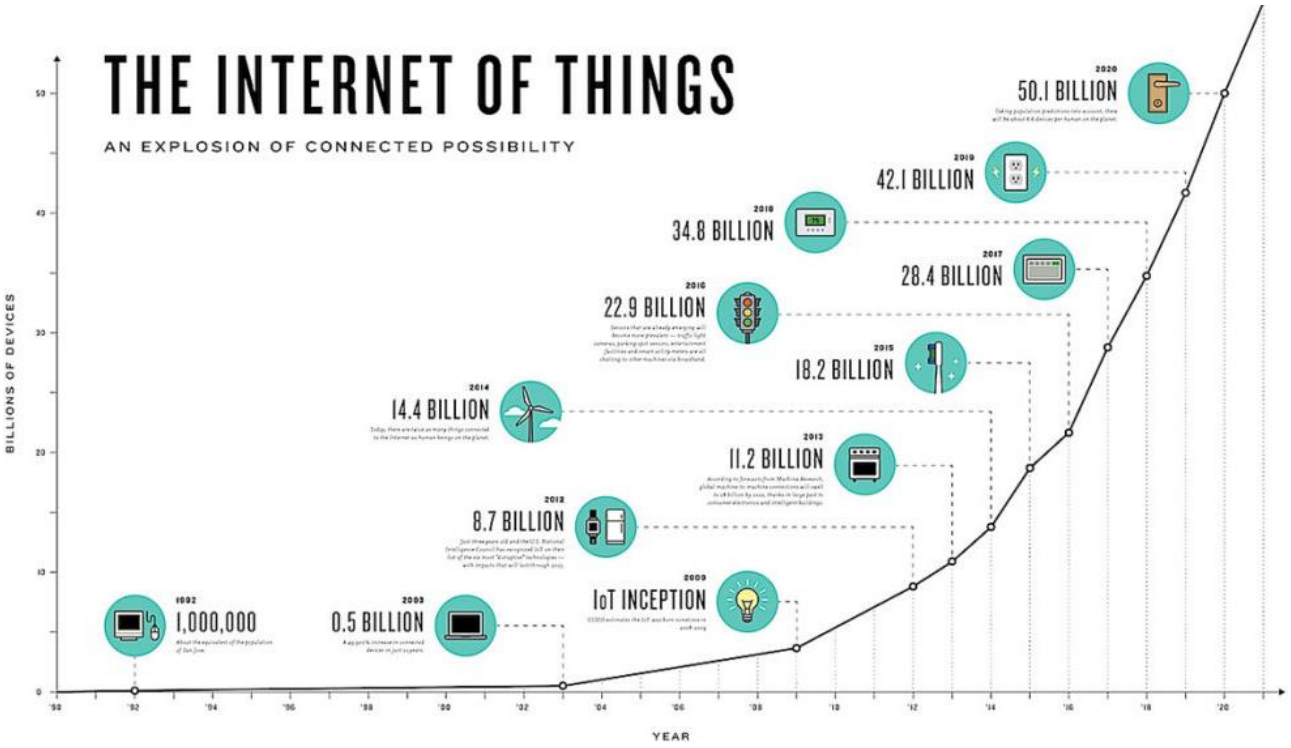


Figure 15: IoT expected growth [22]

As can be seen in Figure 15, the growth of IoT devices is absurd. It is growing exponentially and it is already at 35 billion (April, 2018). All those devices, as we well know, are able and need to communicate with the Internet and participate in whatever service they are part of, so the backend layer of their services will have to handle an exponentially growing amount of incoming traffic. So far, the cloud computing solutions that are currently available have been able to handle all those connections and data, but the growth rate of IoT devices is much higher than the scalability of current architecture solutions.

A stable IoT infrastructure needs to be capable of handling this number of devices, as well as have a scalability factor that enables the system to support the growth of devices at the same time (current scaling solutions will be covered in the next theoretical chapter). IoT's

scalability problem, besides the capability of an IoT system to scale itself, also brings a security issue that we address in the following section.

3.4.1 Security

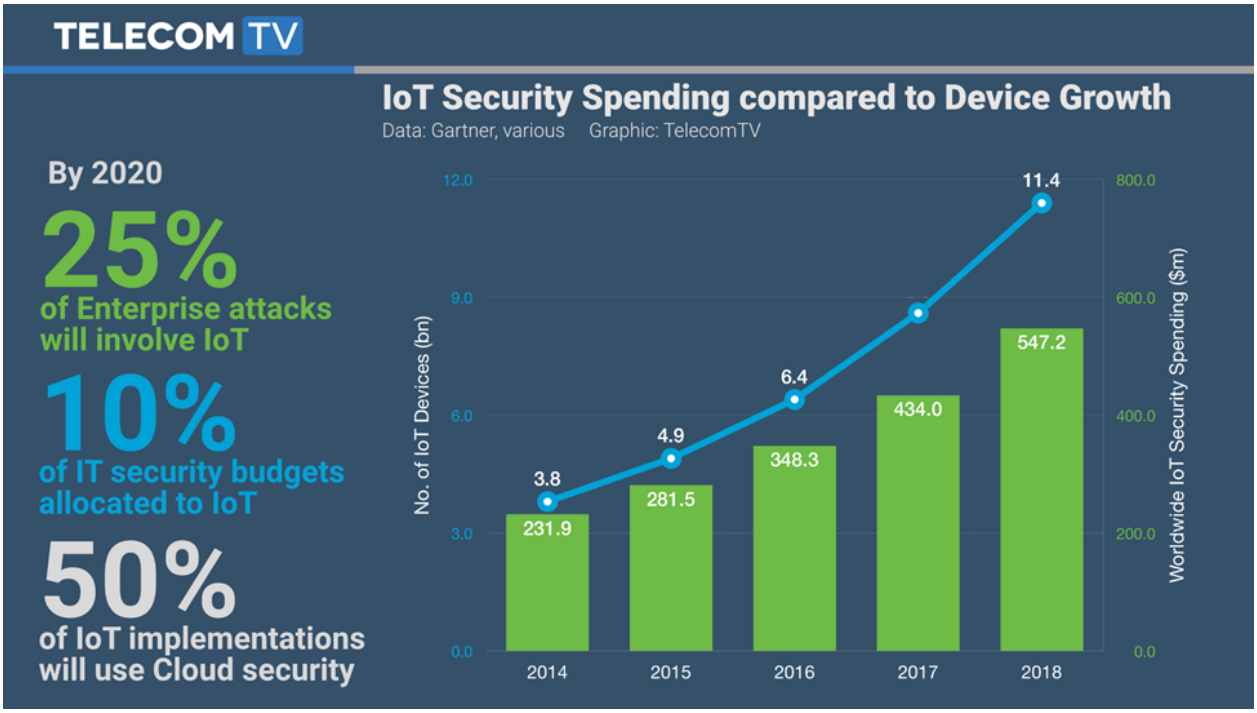


Figure 16: IoT security spending compared to device growth [23]

The need for security is a relative to the type of data a system is handling and its purpose, and involves terms like confidentiality, integrity, authentication and availability. For instance, medical data is a type of data that has always been considered the most sensible type, because if it were tampered with, a doctor could give a diagnosis that would be based on fake data, recommend different medicines or doses that could be a threat to patients' lives. On the other hand, while the fact that somebody set the AC of a smart home may not seem important, the company that provided that client with the technology may have signed a confidentiality

agreement stating that any information must not go public, so privacy is a very important requirement for the service.

The security in IoT is seriously compromised, simply because if a certain device has a very low and know probability of being hacked, if there are 30 billion similar devices, the probability will be multiplied by 30 billion. That is an oversimplification of the mathematical problem that calculates that probability, but it is clear that the amount of IoT devices out in the open increase those probabilities, especially when the majority of those devices are designed to consume very little energy and therefore cannot implement very complex security features. In this case, scalability and security are both compromised at the same time with the huge growth rate of IoT devices.

On the other hand, IoT's infrastructure (Figure 12) present single points of failure in layers 3 and 4, as the data is gathered and managed in the cloud. Not entirely single points of failure, as these systems are usually replicated, but still a low number of servers and a centralized system that may be compromised. Could this issue be solved with blockchain?

3.4.1 Interoperability

IoT lacks standardization of platforms and protocols, as we have seen in the most common protocols section. While it makes sense that there are different protocols that may fit better in some use cases than others, there are a lot of them that basically have the same features and therefore confuse developers and manufacturers. IoT suffers from "market fragmentation", a scenario in which many competitors use very different and highly-incompatible technology stacks, so that clients can buy standalone products but struggle to put them together in an IoT environment.

These problems make IoT have almost zero interoperability, because products cannot communicate with each other. This problem has made IoT's mass adoption take longer to come (not entirely correlated to the number of devices).

4. Cloud Computing

As the last theoretical chapter, this one will go over the basics of cloud computing and how it defines internet services.

4.1 Introduction

If we previously discussed Internet of Things as a network of interconnected devices, we now have to address the Internet of Services, which is something that has appeared in the last years.

The Internet started in the 1960s in the form of a few computers that formed a packet switched network in the United States, and the idea of the initial adopters was very far from what Internet has become nowadays. It started as a decentralized network of computers, and then more and more computers joined around the world, becoming servers in the process. Servers ‘serve’ data, as opposed to general-purpose personal computers, and made the Internet more of a service marketplace instead of just a way of sending data to remote locations and receiving it.

Now, after approximately 35 years, the Internet has become the Internet of Services, a network that hosts and operates all kinds of businesses and utilities that are available worldwide. While one could host one of those services and successfully make it available for anyone to use, it would be impossible to fulfill non-functional requirements as a minimum up-time, resistance to power outages or be capable of keep the service running while thousands of people access it. This is where server farms or clusters come into action, given by big Internet providers or

companies, which introduce the concept of ‘cloud computing’ by distributing resources and ensuring their security, availability, interoperability, resilience, scalability, consistency, speed and many more non-functional requirements.

4.2 What is cloud computing?

Also known as distributed computing, cloud computing is a concept that stands for large systems that, in a well-thought hierarchy, break down and delegate different tasks to many machines that are not necessarily in the same place and are usually recommended to be distributed. It differs from the previous Internet model in the sense the number of computers and the way they are orchestrated in order to offer a service, where in traditional computing systems it used to be just one computer. Cloud computing and consist of thousands of machines that serve millions of users for a single service [24]. It is how big companies like Google, Facebook or Amazon manage to handle such number of users searching websites, businesses information on Facebook, instant messaging and a very long list of services that require enormous resources.

In other words, cloud computing refers to the set of elements and techniques that provide ubiquitous access to web services to the public. The service of cloud computing itself is usually given by big companies that possess huge IT infrastructures that they rent for whoever wants to host a service or use the cloud for any purposes: storage, authority servers, or anything that requires computing power, disk space or traffic handling.

4.3 IaaS, PaaS and SaaS

Cloud computing services can be classified in three big groups: Infrastructure, Platform and Software as a Service (IaaS, PaaS, SaaS).

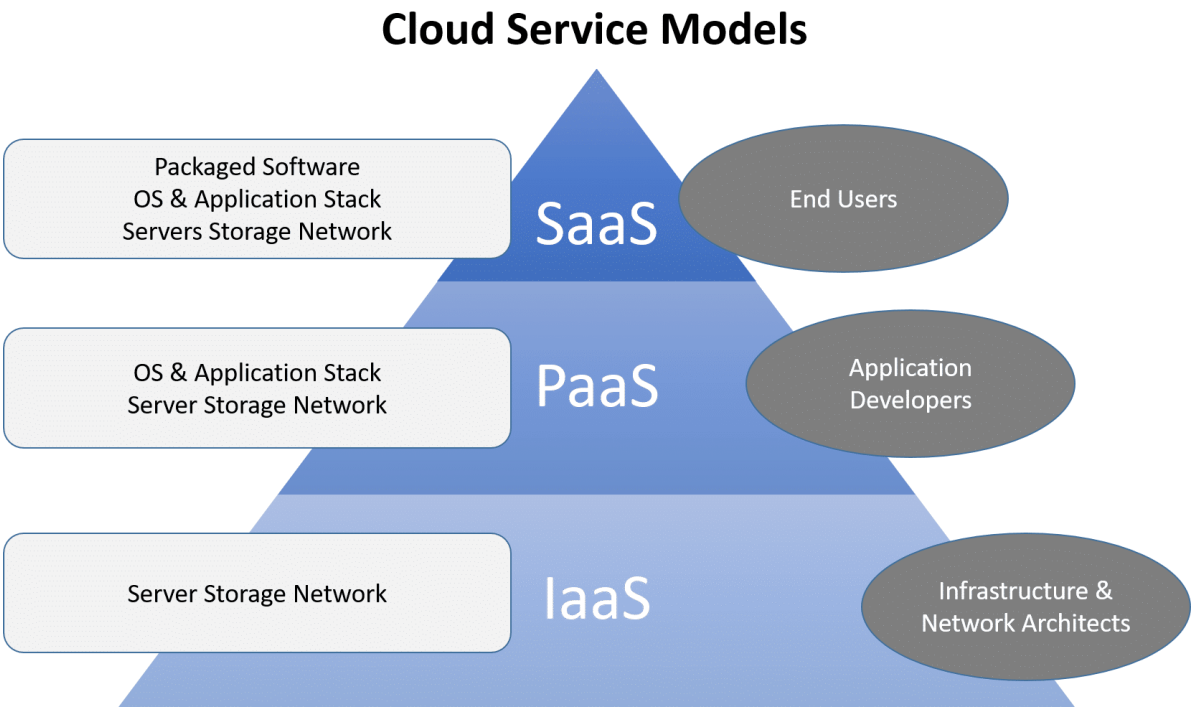


Figure 17: Cloud Service Models [25]

These cloud services are cataloged as of how far the users are kept from the details of what they are using, abstraction levels. A user can ask to own a remote computer for any purpose, computing resources for a specific purpose like installing an application or just use an online text editor.

- IaaS: raw machines that are already networked and can have any operating system installed. This is the most basic and lowest level cloud service. The provider takes care of keeping these machines available and typical server farms tasks like cooling and

internet access. IaaS can also offer remote storage. The most notable IaaS provider is Amazon with its AWS (Amazon Web Services). It is the cloud computing service that is used in the practical part of the project.

- PaaS: with the infrastructure aspect taken care of, PaaS provides an environment ready to install applications that offer scalability, security, availability and resilience. One of the most notable PaaS provider is Google AppEngine, in which a developer can upload an application and Google takes care of the infrastructure issues.
- SaaS: it is a service in which everything is implemented and it is already working as the highest possible level service. Again, one of the most commonly used SaaS is Google Apps, which provides calendar, email, online office applications and much more without the need of a user to develop nor host the services.

For a company or single user, the choice of using IaaS, PaaS or SaaS depends on how much of the service is desired to be in control of. Usually, big services will use IaaS to properly scale and operate it, and normal users will just consume applications that are already working as SaaS, which are usually websites that offer a service.

4.4 Future with blockchain

Many developers in the space think that blockchain will define the so-called Web 3.0, which is a version of the web in which services are decentralized, on top of distributed as the current web (2.0) is with cloud computing designs that distribute the network, but keep it centralized as it is governed by big authorities and located in certain points of the world.

As we have seen in the blockchain theoretical chapter, the Substratum team are working on decentralizing the web giving its users the power to run it as a whole, eliminating the need for

the current cloud computing solutions, although as of now (April, 2018), the project is not working already, but there are more competitors that try to achieve the same thing. Not only in web services but most of the digital services and businesses that exist today are being reinvented based on blockchain.

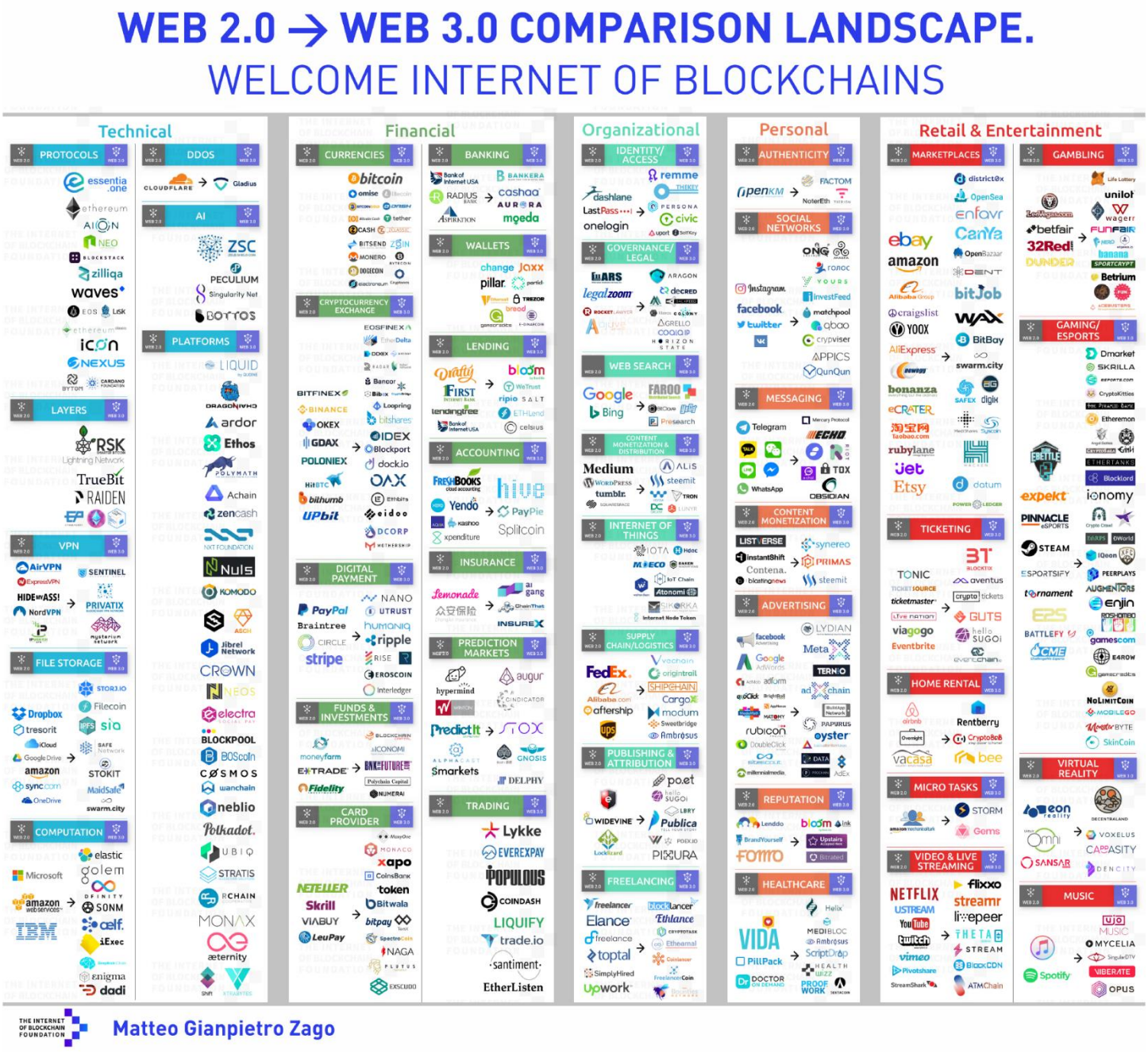


Figure 18: Most web 2.0 services and their web 3.0 competitors [30]

5. Use case development: Smart Home

This chapter will cover the main part of this project: the designed and developed smart home application powered by blockchain, cloud computing and IoT technologies that resemble the ideas and motivations expressed here. The motivation behind this project will first be laid out and then the following subsections will break down the developed system from the highest level (architecture diagrams, explanations on the biggest modules) to lower levels (implemented code).

All the code and designs of the project have been uploaded to a GitHub repository:

https://github.com/pacoard/TFM_2017-18

5.1 Motivation

As we have mentioned several times throughout the document, IoT currently faces scalability and security issues due to the ridiculous growth rate of IoT devices in the market, plus lack of interoperability. The latter has a clear solution of standardization through agreement between the biggest providers, which is a likely case scenario, but scalability and security issues currently have no other solution than replicating servers and adding layers of security on top of them (expensive), and the low-power nature of IoT devices makes them not powerful enough to implement security themselves.

We have seen how blockchain, a very new technology still making a place in the market for itself, has no problems in building a secure distributed ledger that can act as a data base and even execute code, making it a decentralized, super-cheap backend system or distributed computer. If most of the infrastructure that handles IoT applications (aside from the IoT devices

themselves), specifically the parts that are the bottlenecks for their scalability is substituted with a blockchain, the backend would not have to scale as it would be replicated in every single IoT scenario that participates in this blockchain. If the IoT devices have to communicate with an instance of the blockchain (node that participates in the consensus algorithm of the blockchain) that is right next to them, there would not be a need for security anymore for the data that comes out of the IoT environment. That would solve the security issue as it would be handled by the blockchain and not the low-power devices.

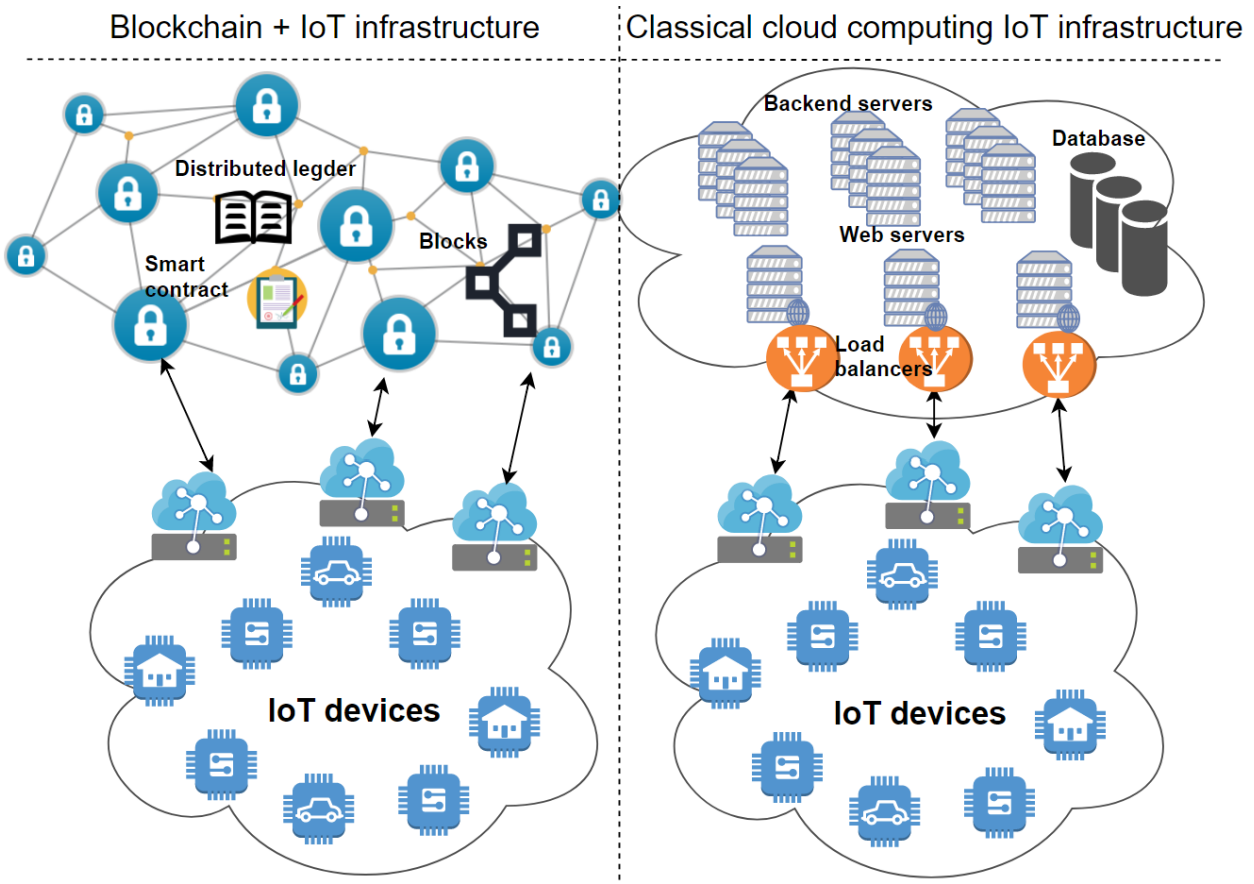


Figure 19: Blockchain VS classical IoT infrastructure

However, this solution is not as good as it sounds, and it needs specific types of blockchain. Currently, the most famous blockchains are the ones that power Bitcoin and Ethereum, which are public distributed ledgers with high transaction fees and very low

transaction speed. That does not sound as a suitable technology for an IoT environment, does it? IoT devices tend to send data very frequently and the delays and fees in a blockchain like Ethereum would make this idea unfeasible, as every device or node that is in charge of the IoT environment would constantly spend transaction fees and the data would take very long to be part of the blockchain. These transaction fees are the incentive for the nodes to spend their time and resources to validate transactions, and that is what keeps blockchain working, and the high delays in the transactions to be mined and stored in blocks (stay forever in the distributed ledger) is due to the decentralization nature of blockchains, the fact that many nodes have to validate these blocks and reach consensus.

On the other hand, there are blockchains that sacrifice decentralization and lower transaction fees to a certain degree so that higher transaction speeds are reached. For instance, Visa is a completely centralized technology (not a blockchain) that can handle 24.000 TPS (transactions per second), Ripple is a cryptocurrency that handles 1.500 TPS [26] but is very centralized compared to Bitcoin or Ethereum, having orders of magnitude less nodes participating in block mining. Ethereum is as decentralized as Bitcoin, but achieved higher TPS through dynamic block size, although it cannot be faster until new scaling solutions are implemented. Moreover, if a blockchain lowers its transactions (mining) fees or completely eliminates them, nobody will be incentivized to run a node in order to mine blocks and validate transactions and therefore decentralize more that blockchain. More miners in a blockchain is a very important necessity for a blockchain to be harder to hack and be more decentralized (more of a trustless system instead of a single third party).

To summarize, blockchains currently have a trilemma that defines the impossibility to have more than two of the following properties: scalability (high transaction speed), security (many miners, harder to hack 51% of them) and decentralization (trustless) [27]. This is true until a new blockchain (hopefully one of the ones that were mentioned in the “third generation

blockchains" section of the blockchain chapter) overcomes the trilemma and manages to have all three properties. Anyways, it depends on the application whether it is worth sacrificing one of those properties.

In our case, we need a very fast transaction speed and no fees at all as our IoT case scenario of a smart home may send data very frequently, then decentralization comes as a less important requirement, also achievable with a very simple solution. If we plan to sell our smart home service, we could force each client to run a node at their homes. That way every client would be a part of the blockchain, have a node that their smart devices can communicate with in order to access and read or update the blockchain and do not need an incentive to do so, as it would be a requirement for the service to work.

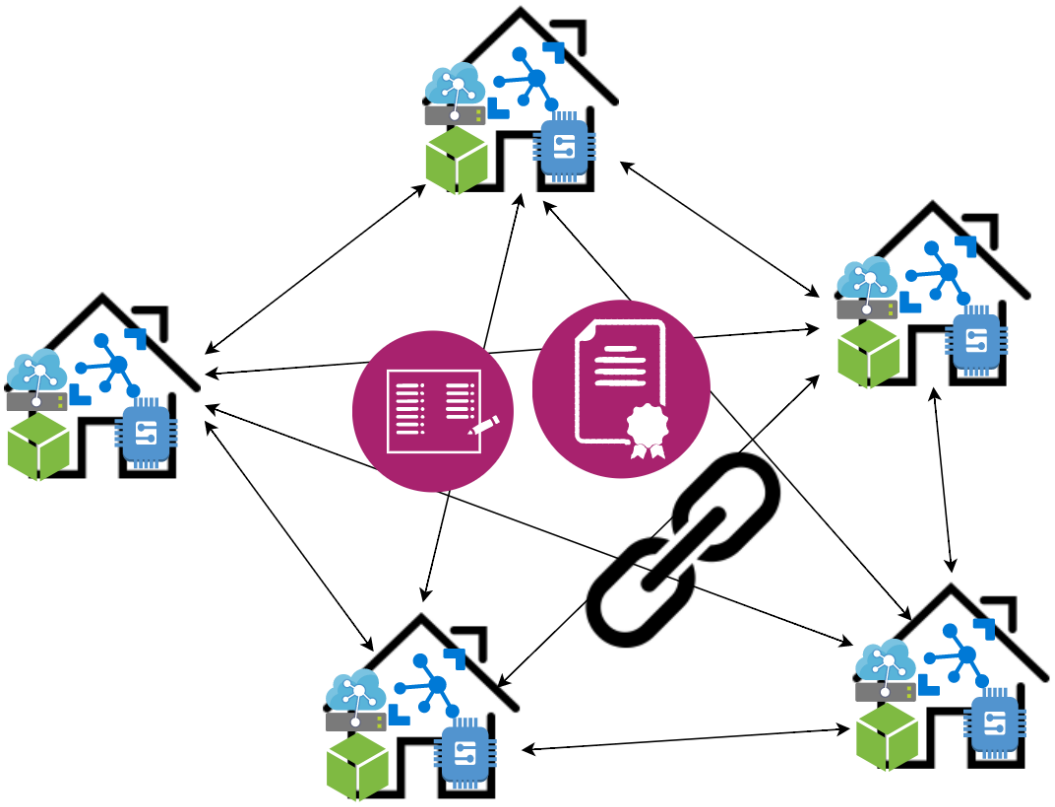


Figure 20: Ideal smart home powered by blockchain

With the proposed solution, we would avoid the cost of backend servers that performed the service logic and databases that stored the information, since the logic would be executed by smart contracts and the data would be stored in the blockchain. As the data is stored in every node, the access to it would be nearly instant, and there would be no need for scaling as the smart contracts are executed right there. The computational cost is taken out of our fictional company and put in the nodes of the clients, which would pay the electricity that the nodes consumed and the cost of the service itself, which we could lower as we would not rent backend servers or databases. On top of that, clients would not have to trust us as a third party because everything is happening within their homes. As a final point, if we designed ourselves all of the sensors and actuators of the service, sell them in packages or separately, we could contribute to the interoperability of IoT, which has proven successful for certain companies [28].

5.2 System overview

Having the motivations in mind, we wanted to develop a proof-of-concept project that puts those ideas into practice. Thanks to the great amount of open source technologies, it has been possible to put them all together and build a project that involves technologies related to blockchain, Internet of Things and cloud computing. From this point forward, we will call our system DIOT: Decentralized IoT. It follows the naming convention that appeared after Ethereum launched, so that all the applications that use blockchain are called “dApps”.

Moving on to a high-level view of DIOT, it is similarly structured to the theoretical chapters of this document, so it is made of three big parts. The blockchain part of DIOT has been built with Hyperledger Fabric, which is a permissioned and private blockchain

implementation. By being private, it ensures that the participants can only be clients of the service, as our fictional company would have to issue a certificate in order for a participant to access the blockchain. Then, by being permissioned, we can establish a control access so that only the owners of a smart home can interact with it, and not, for instance, turn of the lights of another client's home or access sensible data. Hyperledger Fabric also has no fees for transactions, which is a very important requirement for DIOT. Then, given that Hyperledger Fabric's features convinced us, we used it to register participants, digital assets that refer to sensors, their metrics and actuators and transactions that wrap the actions of interacting with an actuator or a sensor sending a measure. As a final point for the blockchain section, we have used Hyperledger Composer in order to ease the development on top of Fabric: the smart contract has been written in Javascript, and the blockchain could be easily deployed along with a REST API server that allows blockchain interaction with the commonly known format JSON and through HTTP requests.

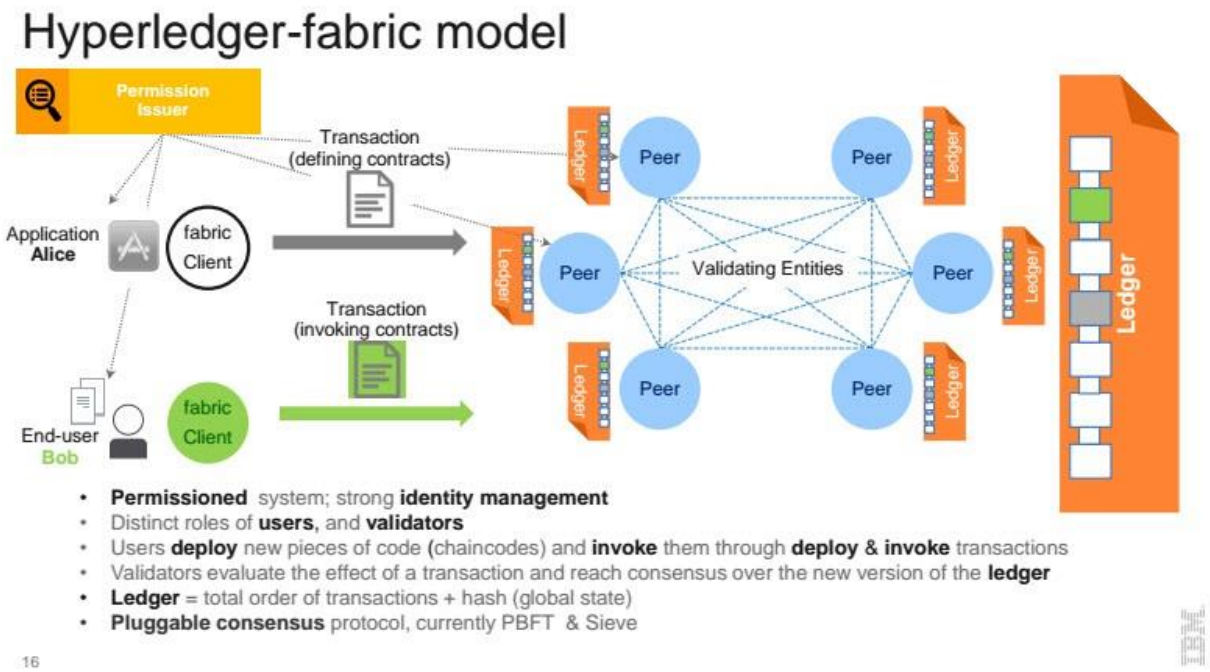


Figure 21: Hyperledger Fabric summary [1]

On the other hand, the blockchain cannot host a website or a graphical user interface (frontend) that allows users to control their homes. This maps to the cloud computing part of DIOT, in which a website that can access the blockchain has been developed and deployed with AWS (Amazon Web Services). This way it was possible to automate the deployment and configure an infrastructure with replicated servers that serve the frontend website behind a load balancer. This way the service meets the typical requirements of a cloud computing system: availability and resilience. The website has been developed with ReactJS (a Javascript library that modularizes the components of a website) and the deployment script that interacts with AWS is written in Python.

Finally, for the IoT section of DIOT, we wanted to test the service with real sensors and actuators that could simulate the behavior of an IoT environment in a smart home. As the service allows the user to register any smart devices that can communicate with HTTP messages and the design of electronic equipment was out of the scope of this project, we have used open source hardware and cheap electronic components available in the market. The are a few sensors connected to an Arduino, which gathers their measurements and sends them to the REST API of the blockchain, and then a Raspberry Pi 2 Model B is in charge of the actuators. This last decision was based on the idea that our fictional company would install a computer in every clients' homes that plays the roles of a blockchain node and a gateway between the smart home IoT environment and the blockchain, so a Raspberry Pi is a good fit for some of these tasks. Given that it is costly to install more nodes, and it is pointless as we are only simulating one smart home, we have decided to implement the blockchain as another server in AWS instead of on the smart home itself.

The following figure shows the result of this project: the DIOT system:

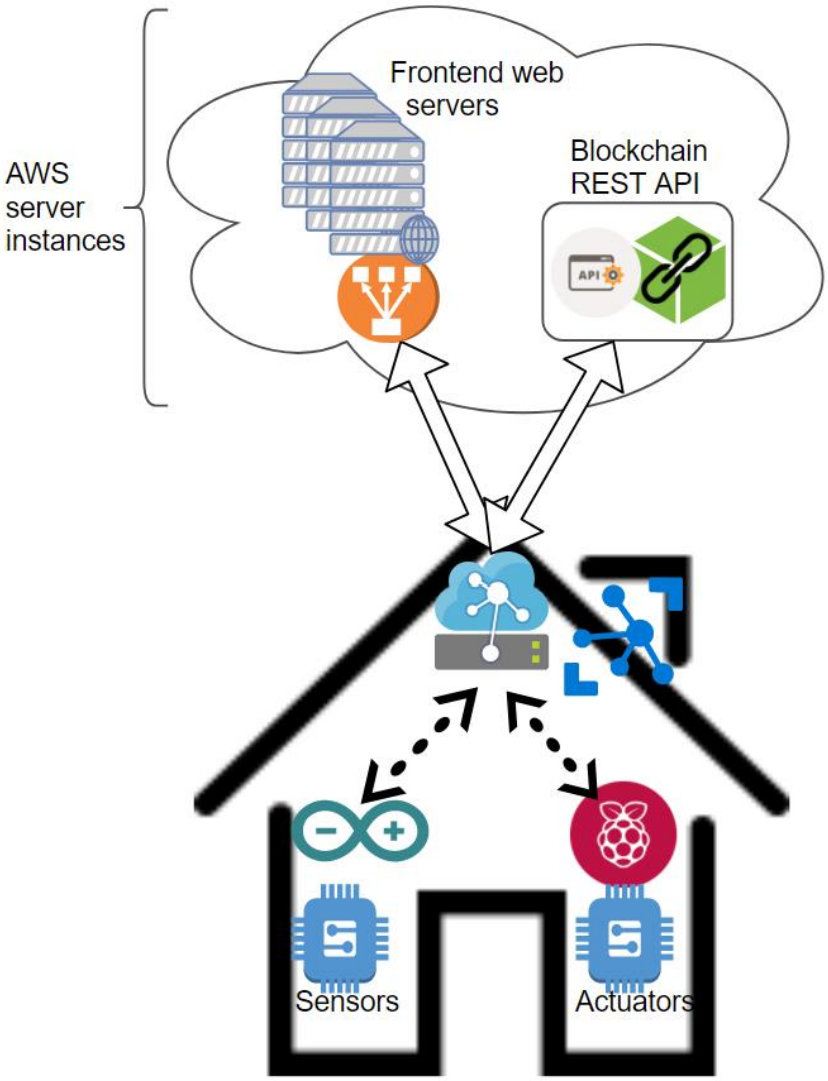


Figure 22: DIOT system overview

Now that we have a general view of what DIOT is about, we will proceed with explanations on the design and implementation of each of the sections, for a better understanding of the logic and decisions behind each module.

5.3 Blockchain

As the backend and database of DIOT, we have developed a smart contract, or chaincode as it is called by IBM (Hyperledger Fabric was developed by IBM), which has to fulfill the following requirements:

- Allow participants to access their homes only, and not anyone else.
- Store sensors and actuators as digital assets.
- Transactions that offer CRUD functions (Create, Read, Update, Delete) on sensors and actuators.
- Transactions that let sensors register measures.
- Transactions that let users interact with actuators by changing their state.
- Emit events when sensors register a measure that surpasses a certain threshold.
- Emit events when an actuator's state has been changed by the user.

A requirement of user authentication was also considered, but was not implemented due to two reasons: first, the creation of certificates and private keys was very tedious and we were not planning on testing more than one physical IoT environment, and second Hyperledger Composer did not entirely implement authentication against the REST API server that interacts with the blockchain as of February, 2018 [29]. At the time of writing, Hyperledger has just fully implemented OAUTH 2.0.

5.3.1 Design

By following Hyperledger Composer's tutorial [29], the design is fixed to the following basic structure, which we have kept for DIOT:

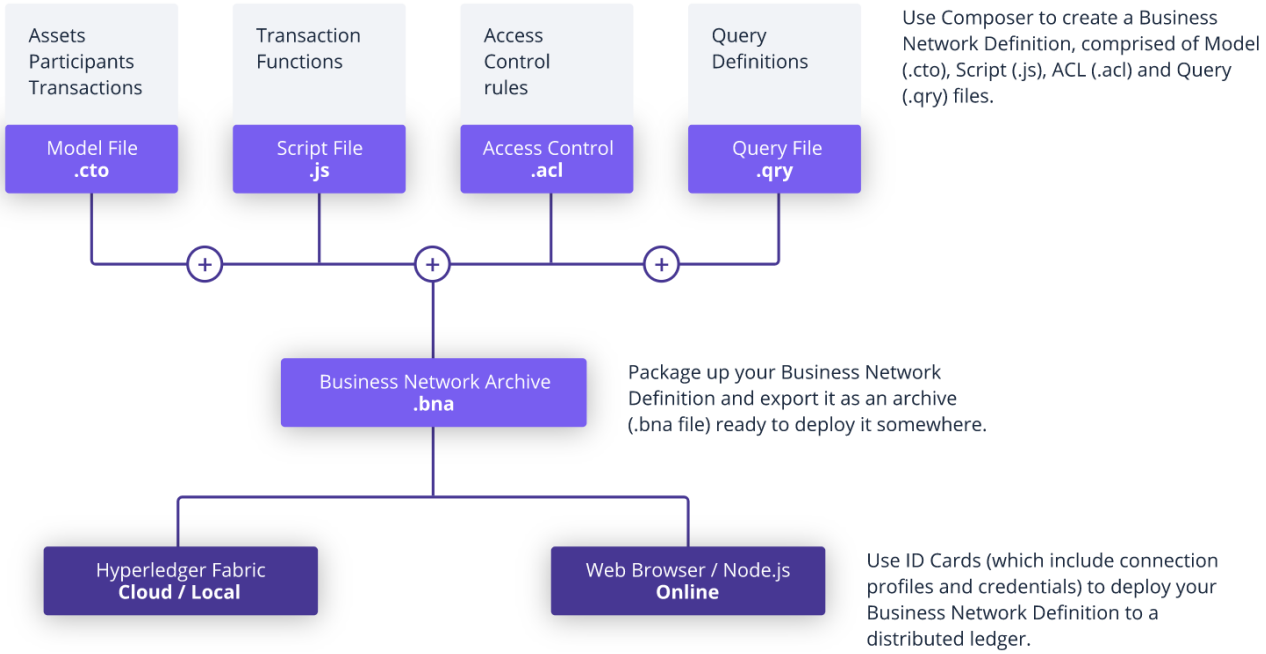


Figure 23: Structure of a Hyperledger Composer project [29]

The files in the upper level define the smart contract or chaincode, which is packaged into a BNA file that can be deployed to Hyperledger Fabric as it if were a widely known EXE file. In the third level, we use Hyperledger Fabric Local, as the Cloud version is not free, but we have deployed it in an AWS server so that the IoT devices of DIOT can access the blockchain as a remote server. The development environment of Hyperledger Composer (Local) automatically creates 4 nodes, each in a Docker container as Hyperledger Fabric does, that conform the blockchain.

As can be seen in the structure of Figure 23: Structure of a Hyperledger Composer project , the smart contract is designed in the following files:

- CTO: models participants, assets (sensors and actuators), transactions and events.

- JS: Javascript file that defines the logic of the transactions. They are developed in “Transaction Functions”.
- ACL: defines the access rules. For instance, it is defined in this file that nobody can access anything unless it’s a participant that owns devices, and then can only access those devices (defined as digital assets in the CTO file).
- QRY: implements queries for an easier way to read data in the blockchain with SQL-like prepared commands that are accessible via the REST API.

5.3.2 Implementation

Thanks to Hyperledger Composer, the implementation of the chaincode has been the easiest part of the development of DIOT. Given the design that we have just covered, we only had to implement the logic in the aforementioned files.

The CTO models file defines the participants, assets, transactions and events as follows:

```
participant DeviceOwner identified by email {
  o String email
  o String name optional
  o String phone optional
}
```

```
abstract asset Device identified by deviceId {
  o String deviceId
  --> DeviceOwner deviceOwner
}
asset Sensor extends Device {
  o Reading[] data optional
  o String unit
  o Integer eventThreshold optional
}
asset Actuator extends Device {
  o String state optional
  o Boolean enabled default=false optional
}
```

```
transaction CreateDevice {
  o String deviceId
  o DeviceType deviceType
  o String unit optional
  o Integer eventThreshold optional
  o String state optional
  o Boolean enabled optional
  --> DeviceOwner deviceOwner
}
transaction DeleteSensor {
  --> Sensor device
  --> DeviceOwner deviceOwner
}
transaction DeleteActuator {
  --> Actuator device
  --> DeviceOwner deviceOwner
}
transaction SensorReading {
  o Double value
  --> Sensor sensor
  --> DeviceOwner deviceOwner
}
transaction ActuatorWrite {
  o String newState optional
  o Boolean enabled optional
  --> Actuator actuator
  --> DeviceOwner deviceOwner
}
```

```
event SensorEvent {
  o String sensorId
  o String ownerEmail
  o String msg
}

event ActuatorEvent {
  o String actuatorId
  o String eventCode optional
  o String newState optional
  o Boolean enabled
  o String ownerEmail
  o String msg optional
}
event LogEvent {
  o String ownerEmail
  o String msg
}
```

The modifiers in this code (a model language implemented by IBM) are very self-explanatory and give constraints to the objects that make the code type-safe. These objects will later be available as Javascript objects for the transactions and perfectly suited for JSON messages that will be exchanged between the frontend website, the blockchain and the IoT devices.

As of the Transaction Functions in the Javascript file, a function for each transaction has been implemented. Each of these transactions do not need to perform many validations on the data as the CTO file strongly forces the input data of a transaction to comply with their

definitions. The Transaction Functions simply access the blockchain in order to write the data, according to which transaction is being invoked. Then, depending on the transaction, the functions may emit events that are broadcasted on the REST API via WebSocket, so that we can see alarms when a sensor surpasses a threshold or actuators can know when their state has to change.

The ACL file implements the following rules:

```
rule AdminCanDoAnything {
    description: "Allow admin to do anything"
    participant: "org.hyperledger.composer.system.NetworkAdmin#admin"
    operation: ALL
    resource: "diot.biznet.*"
    action: ALLOW
}

rule OwnerHasFullAccessToTheirAssets {
    description: "Allow all participants full access to their assets"
    participant(p): "diot.biznet.DeviceOwner"
    operation: ALL
    resource(r): "diot.biznet.Device"
    condition: (r.deviceOwner.getIdentifier() === p.getIdentifier())
    action: ALLOW
}

rule SystemACL {
  description:  "System ACL to permit all access"
  participant: "ANY"
  operation: ALL
  resource: "org.hyperledger.composer.system.**"
  action: ALLOW
}
```

The first rule would not be implemented in a production environment but helps as a backdoor in the code in case we wanted to access anything. The second rule makes sure that the digital assets of an owner can only be accessed by that owner, and the third rule lets the system itself to access anything.

Finally, the QRY file implements queries that are of the interest of the frontend web application:

```
query selectSensorsByOwner {
    description: "Select sensors by owner"
    statement:
        SELECT diot.biznet.Sensor
            WHERE (deviceOwner == _$deviceOwner)
}

query selectActuatorsByOwner {
    description: "Select actuators by owner"
    statement:
        SELECT diot.biznet.Actuator
            WHERE (deviceOwner == _$deviceOwner)
}

query selectSensorById {
    description: "Select sensor by id"
    statement:
        SELECT diot.biznet.Sensor
            WHERE (deviceId == _$deviceId)
}

query selectActuatorById {
    description: "Select actuators by id"
    statement:
        SELECT diot.biznet.Actuator
            WHERE (deviceId == _$deviceId)
}
```

We will see more in detail the results of those queries in the frontend section, but the names of the queries resemble what they are for. For instance, the query “selectActuatorsByOwner” will retrieve a list of the actuators that are under the email of the user that is making that query, and the Hyperledger Composer’s REST API will send back a JSON with the result of the query.

5.4 Cloud computing and frontend

This section goes over the cloud computing part of DIOT. It is in charge of provisioning the system with a resilient and robust web frontend to interact with the smart home, within the limits that AWS has for a free account. The requirements for this part of the project are:

- Automate the deployment of the frontend server's architecture.
- Deploy the frontend (website for DIOT) with an architecture for the service that complies with typical web-application characteristics: scalability, resilience and availability.
- Give users a nice UI experience, intuitive and easy-to-use website that manages a smart home.
- Communicate with the blockchain's REST API server with JSON messages.
- Graphs for the readings of each sensor with dates and units.
- Let the user interact with actuators by enabling them and changing their status.
- Add and/or delete new devices.

In order to fulfill those requirements, there are two main parts in this section: a script that automates the deployment, creation and destruction of AWS servers that serve the website and the website itself.

5.4.1 Design

Firstly, we will cover the cloud deployment. We have coded a script in Python that automates the deployment of a number of servers, working as a command line tool that can be executed from any environment with configured credentials to use AWS. It takes arguments such as the DNS of the blockchain's REST API, a private key to manage the server instances

(which we will call EC2 instances from now on, as they are AWS server instances) and a few more arguments that are necessary to launch EC2 instances. The blockchain, along with its REST API is hosted in a separate EC2 instances that is not automated in the script.

We have chosen Python as the programming language of the script as it is very readable, traceable and its general purpose nature makes it have libraries for pretty much everything, like JSON parsing and building and string manipulation, which was very useful for the script. Once it is executed, it takes 5-10 minutes to deploy as many servers desired in the scenario shown in the following figure:

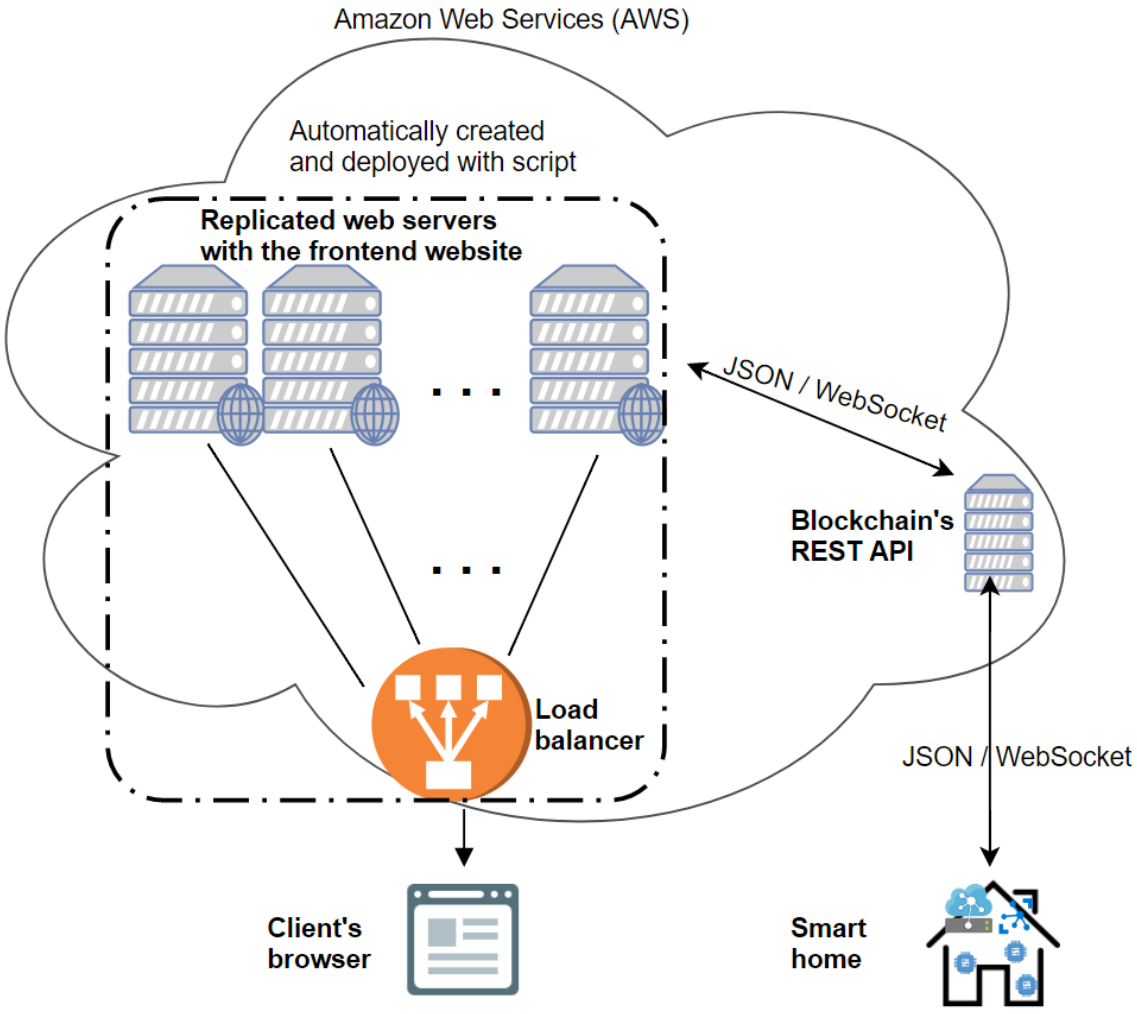


Figure 24: Cloud deployment of DIOT

On the other hand, the website has been developed with the typical stack of the latest trends in web development: ReactJS + Redux and NodeJS (Javascript library and runtime), Webpack (module bundler) and Bootstrap (toolkit for web design). These technologies have been chosen as are proven to be reliable and professional in web development environments. ReactJS is a Javascript library that wraps chunks of HTML code into reusable, dynamic components with metadata and lifecycle, so highly modular applications can be built with it. Thanks to those features, the web design process was very healthy and intuitive because items like tables, transactions with the REST API and notifications could be wrapped into React components and be reused in many parts of the application. On the other hand, Redux is a state container that bypasses the one-way data binding paradigm of React and lets every single component in the hierarchy of components access a globally managed state.

The web application has the logos of Hyperledger, Hyperledger Composer and React+Redux in its welcome site, along with basic data on the blockchain: number of sensors, number of actuators, number of readings from sensors and a blue bell that indicates that the WebSocket events are online (sensor triggering an alarm, actuator changing its state, creation or deletion of a device).

Furthermore, the sensors page retrieves a list of sensors from the REST API and puts the data in a table with some actions available to the user: delete the device or click on its name to see a graph with all the registered readings in the blockchain. This graph is highly interactive, being possible to zoom in and out, see the values of each reading with its timestamp or select a period of time to zoom in. There is also an actuators page that works very similarly to the sensors page, but there is also an option to edit its state and enable or disable each actuator.

After that, there are two pages to add or delete devices, although they can also be deleted in the sensors or actuators pages. The “add device” page presents a form in which the

user can enter the data for a new device, and the “delete device” page fetches two tables with the list of devices (sensors and actuators) and a button next to each one for deletion.

As explained in the blockchain’s smart contract or chaincode, there are events for the creation and deletion of devices, every time an actuator’s state is changed and every time a sensor registers a reading that surpasses the set threshold for an alarm. These events are shown as notifications in the right bottom corner of the website.

After going over all the features of the website, the following figures show the pages of DIOT’s frontend:

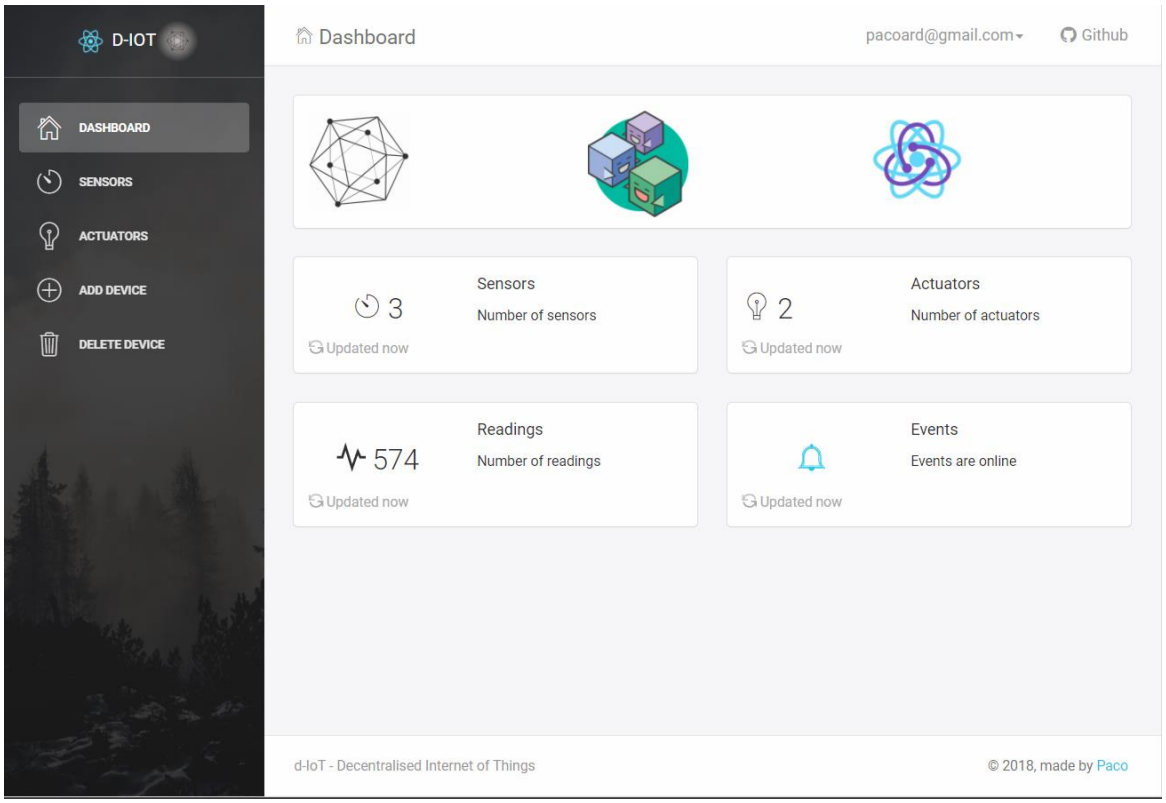


Figure 25: DIOT welcome page

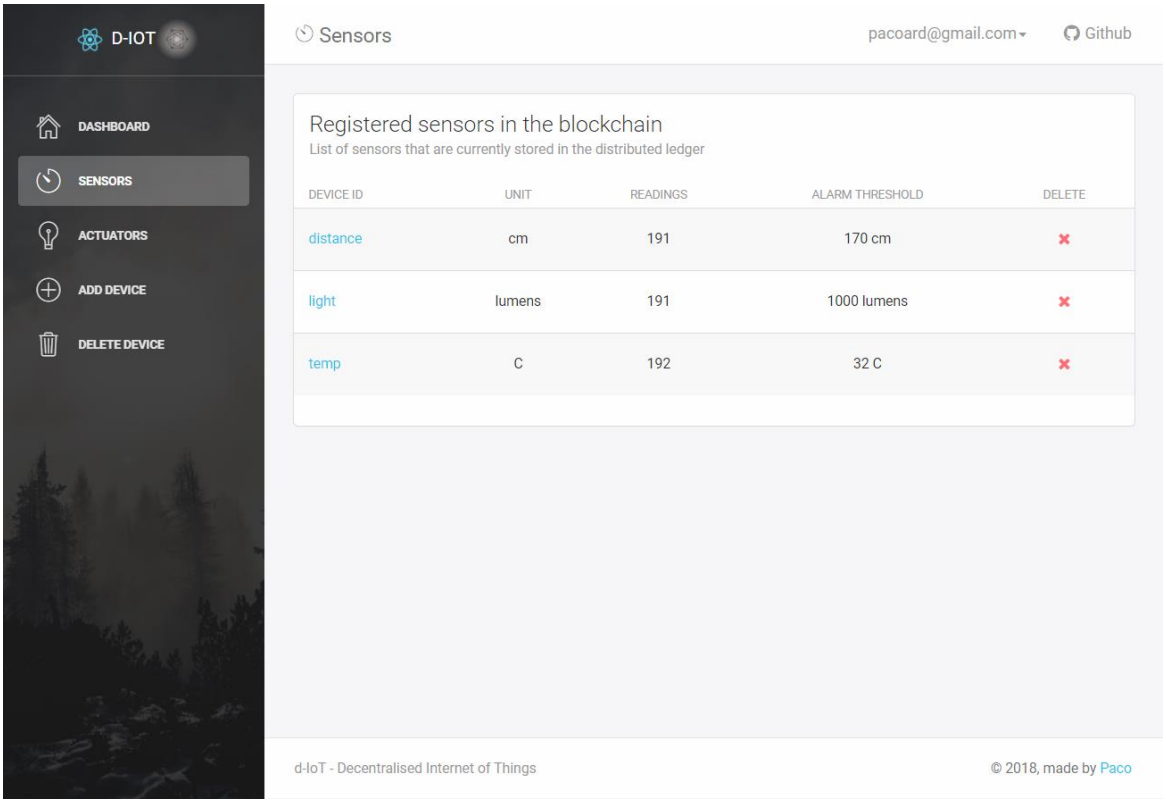


Figure 26: Sensors page

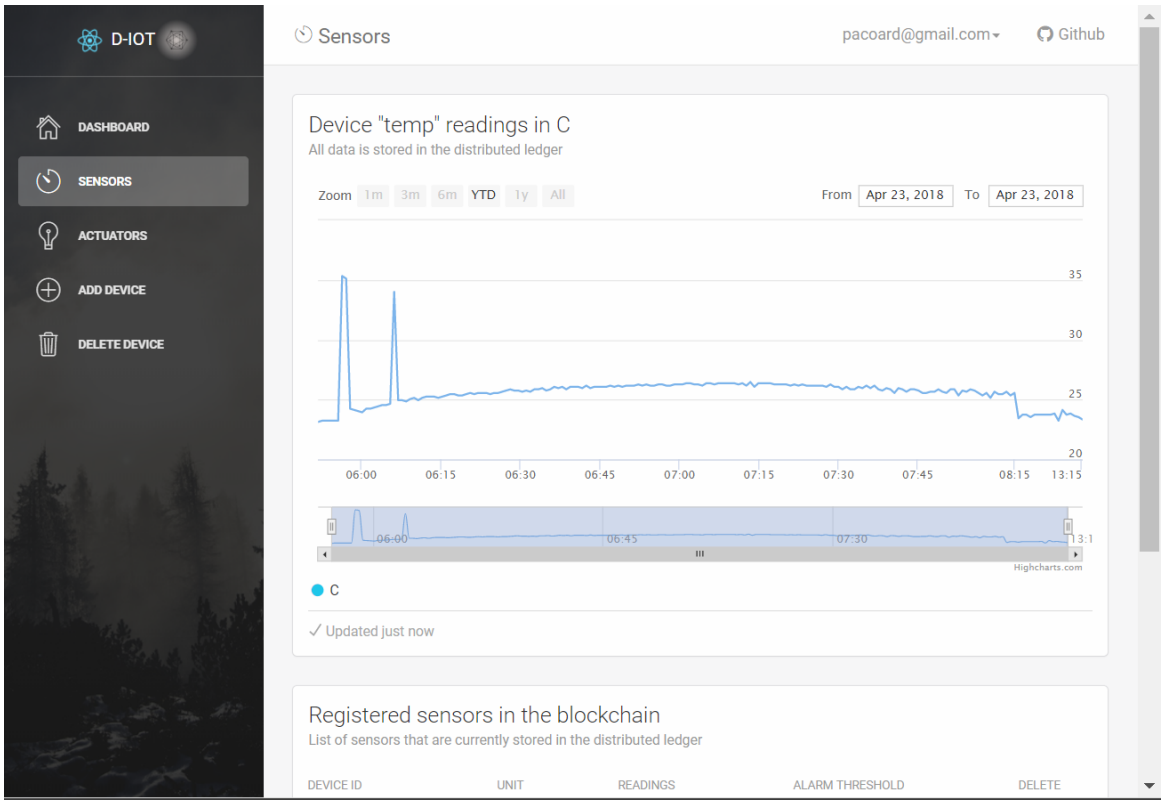


Figure 27: Temperature sensor readings graph

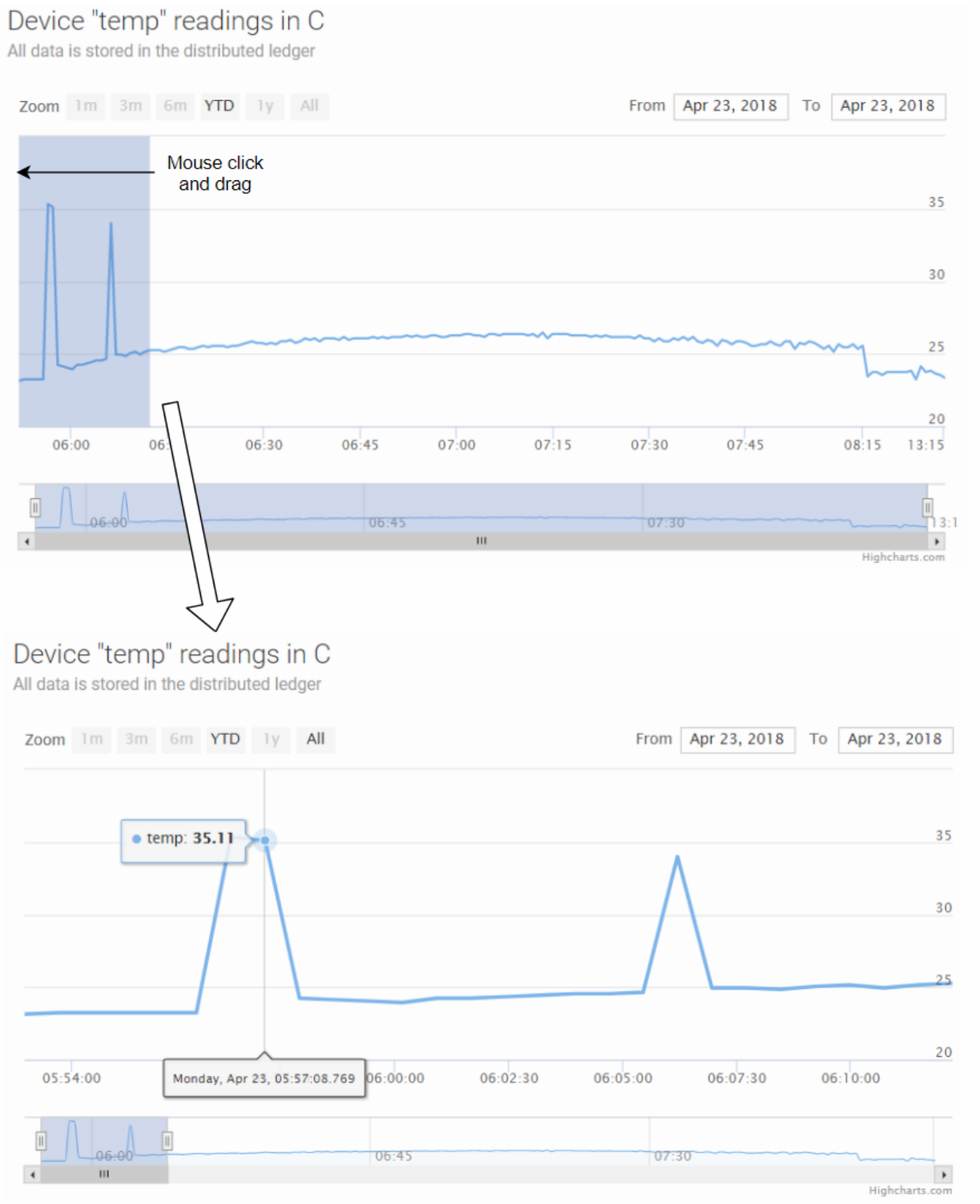


Figure 28: Example of interaction with a sensor graph

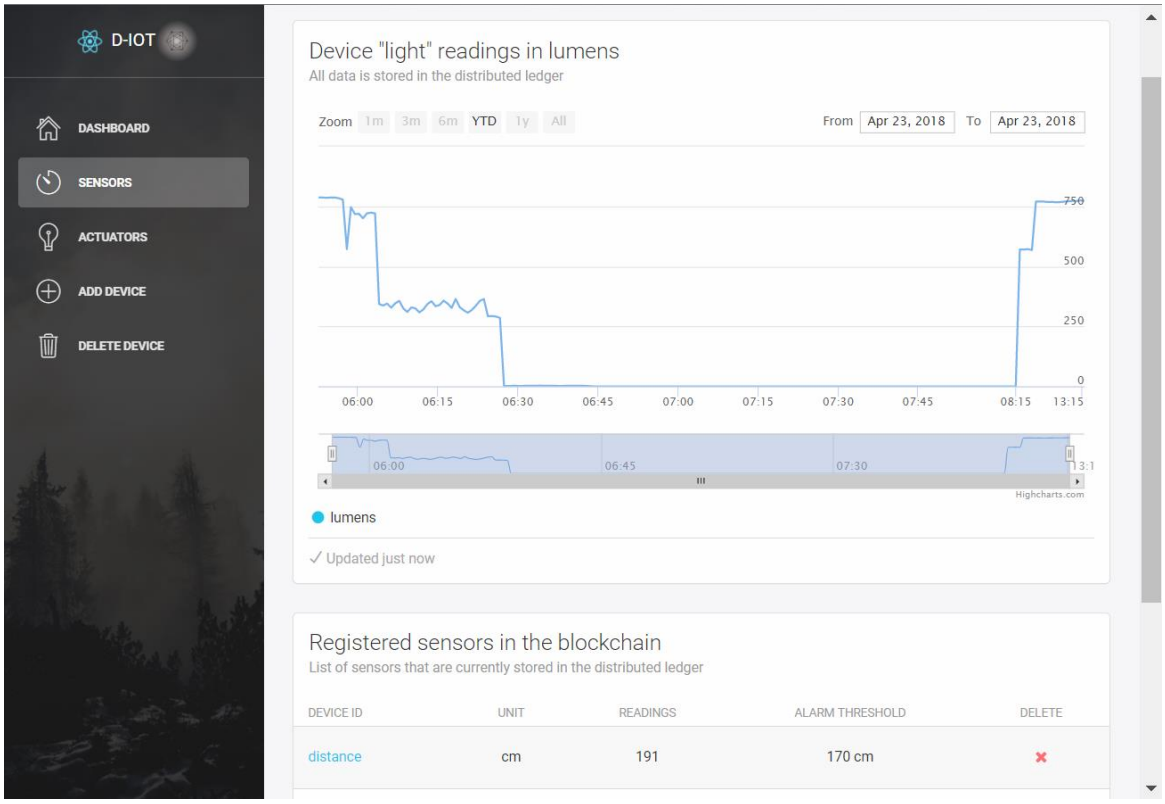


Figure 29: Light sensor readings graph

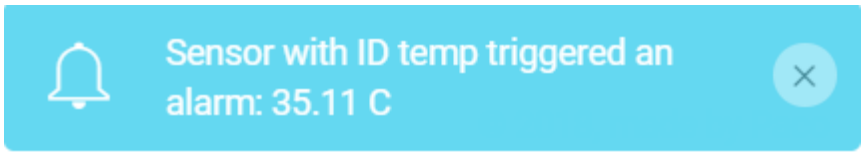


Figure 30: Notification from temperature sensor

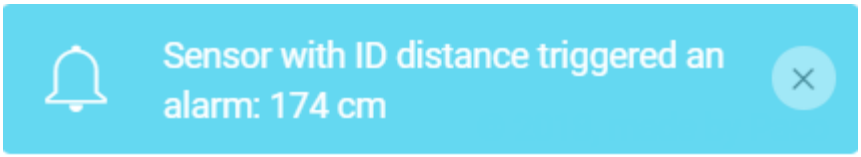


Figure 31: Notification from distance sensor

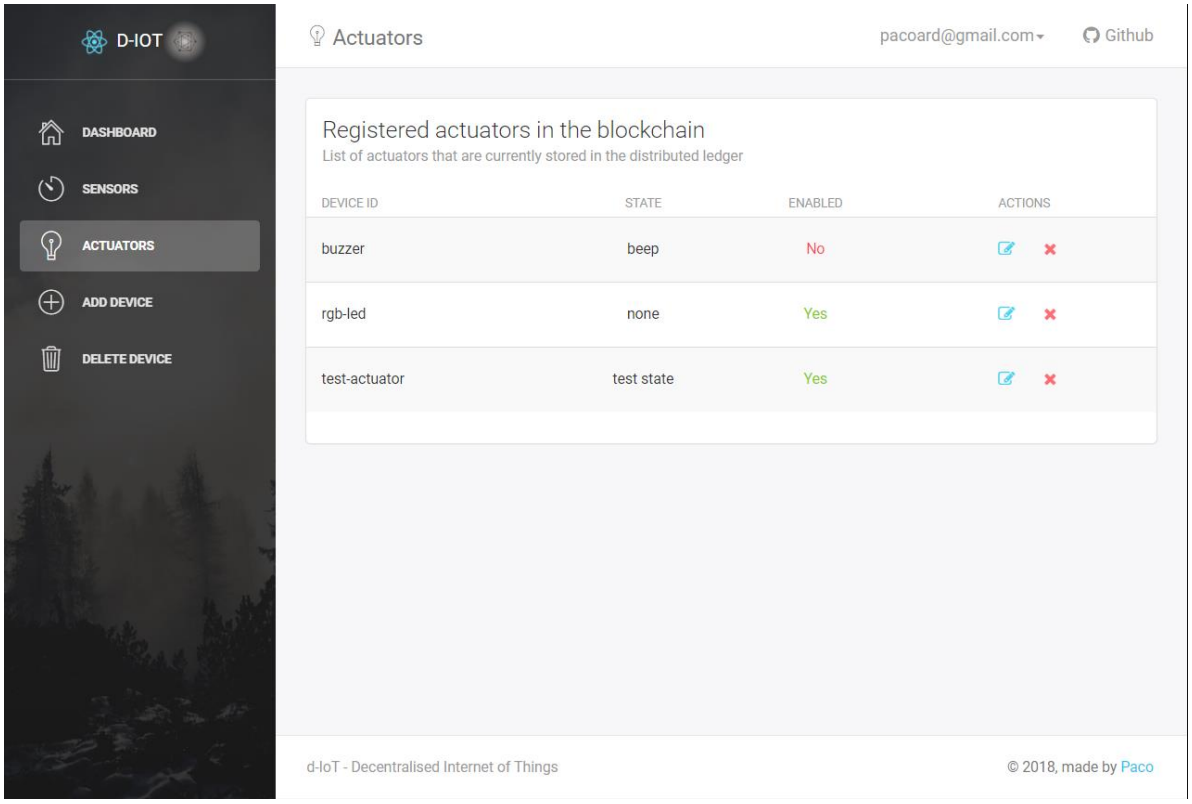


Figure 32: Actuators page

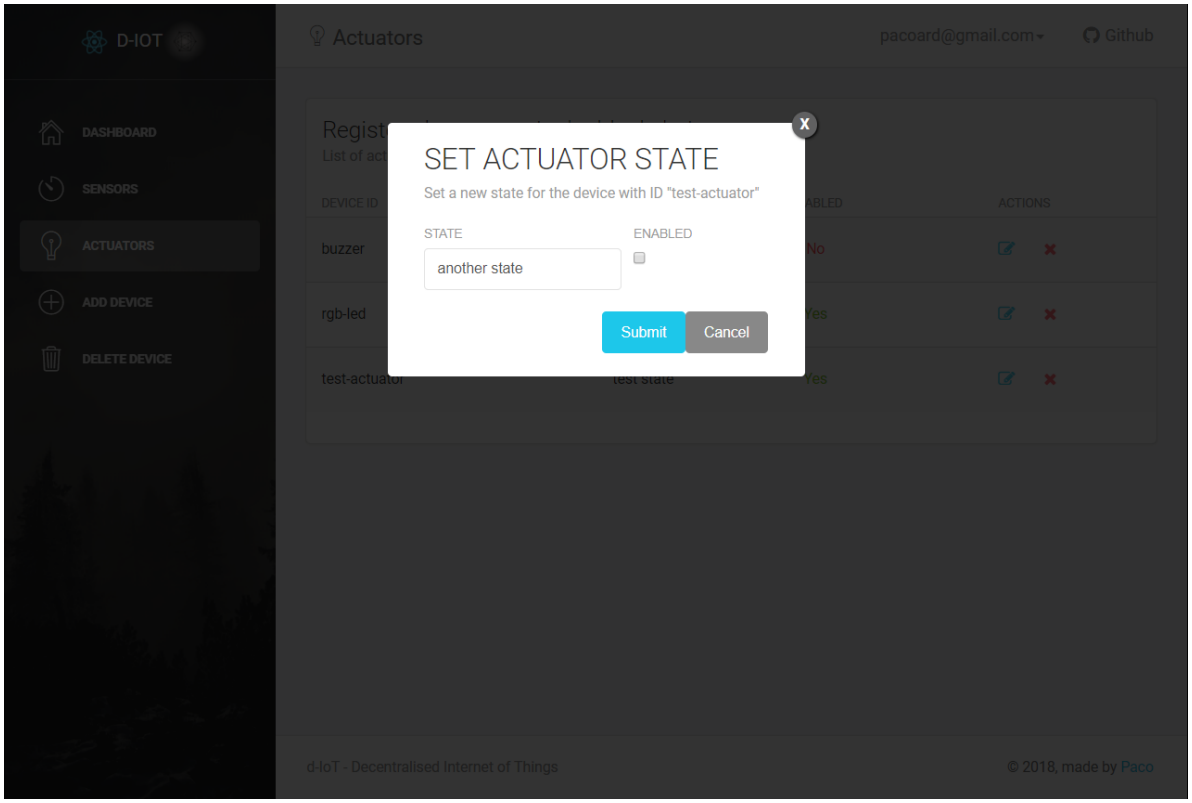


Figure 33: Actuator state change form

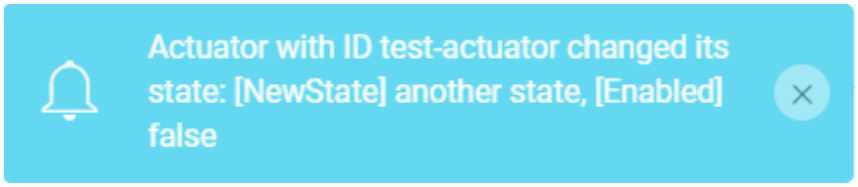


Figure 34: Notification from actuator

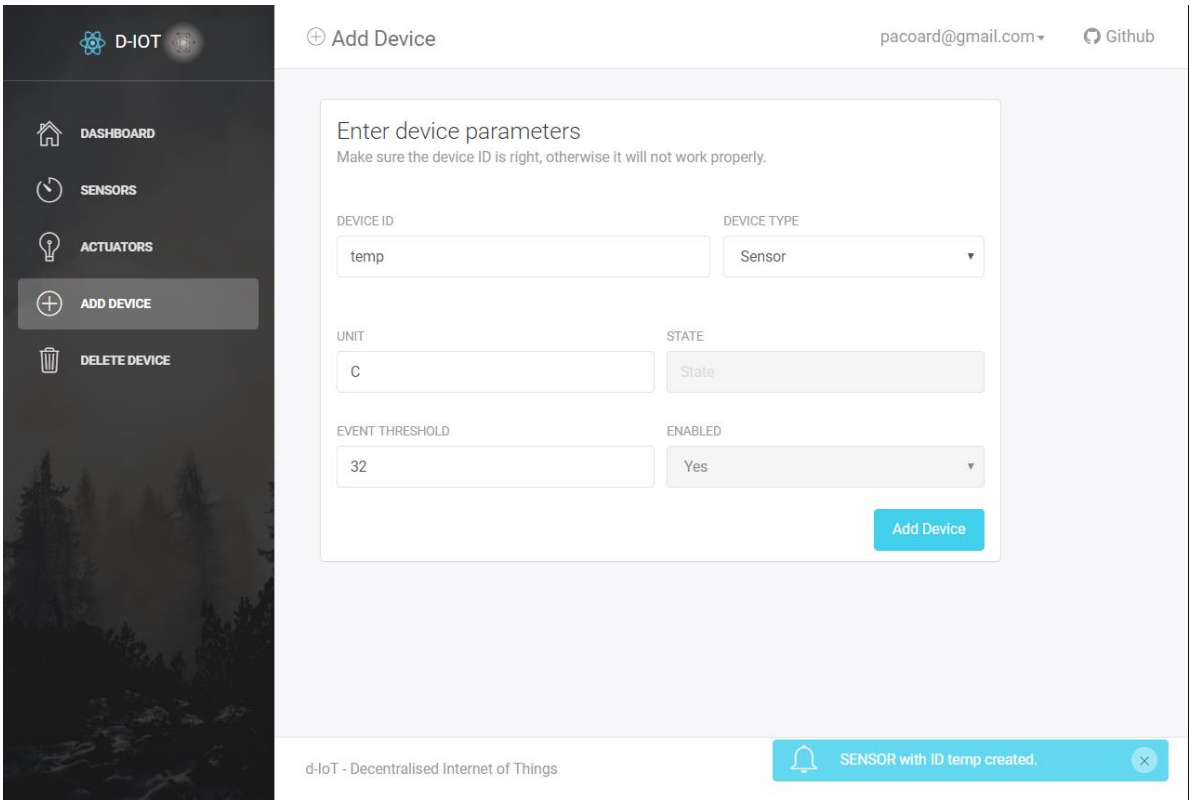


Figure 35: Add device page, adding a sensor

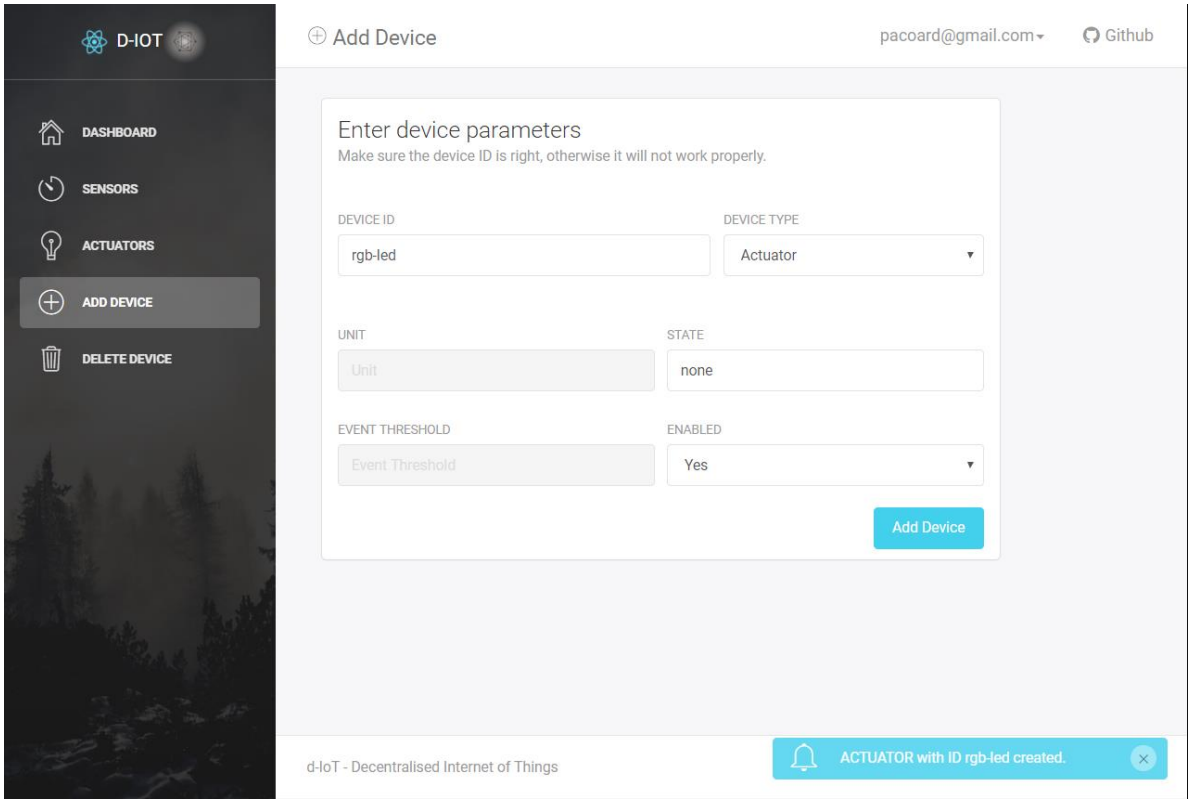


Figure 36: Add device page, adding an actuator

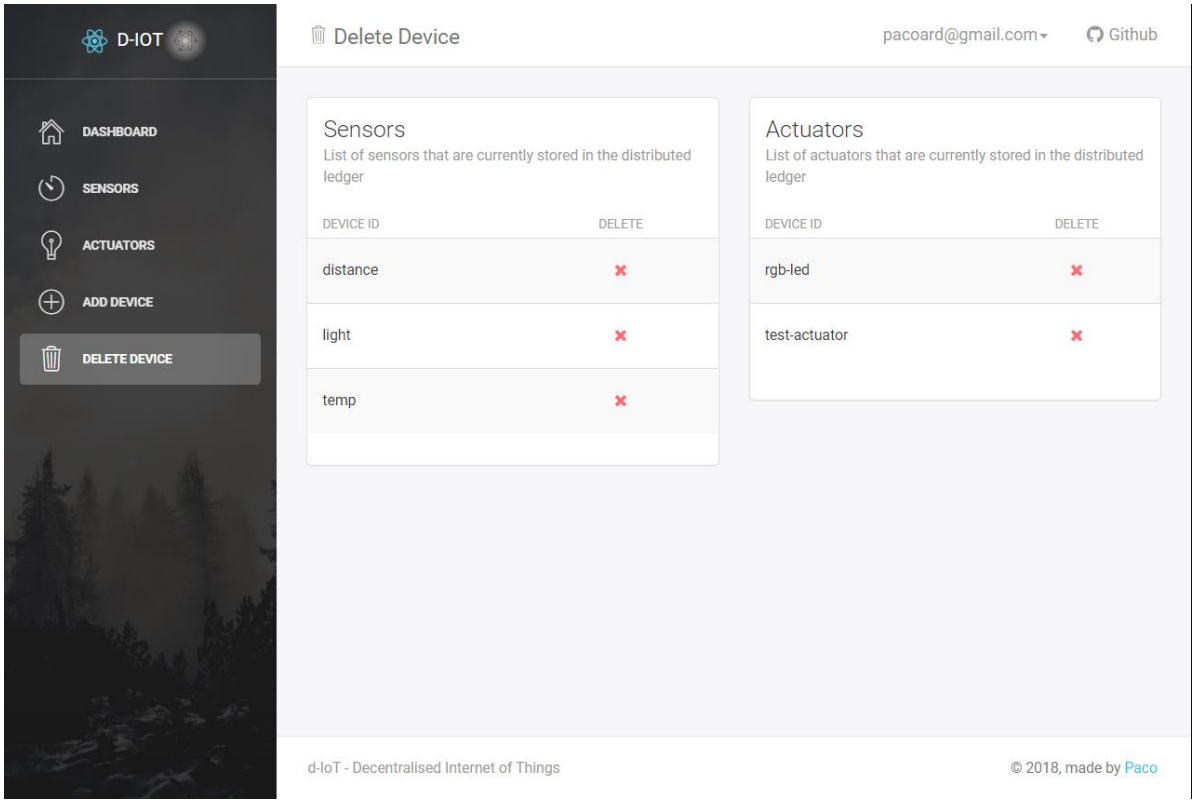


Figure 37: Delete device page

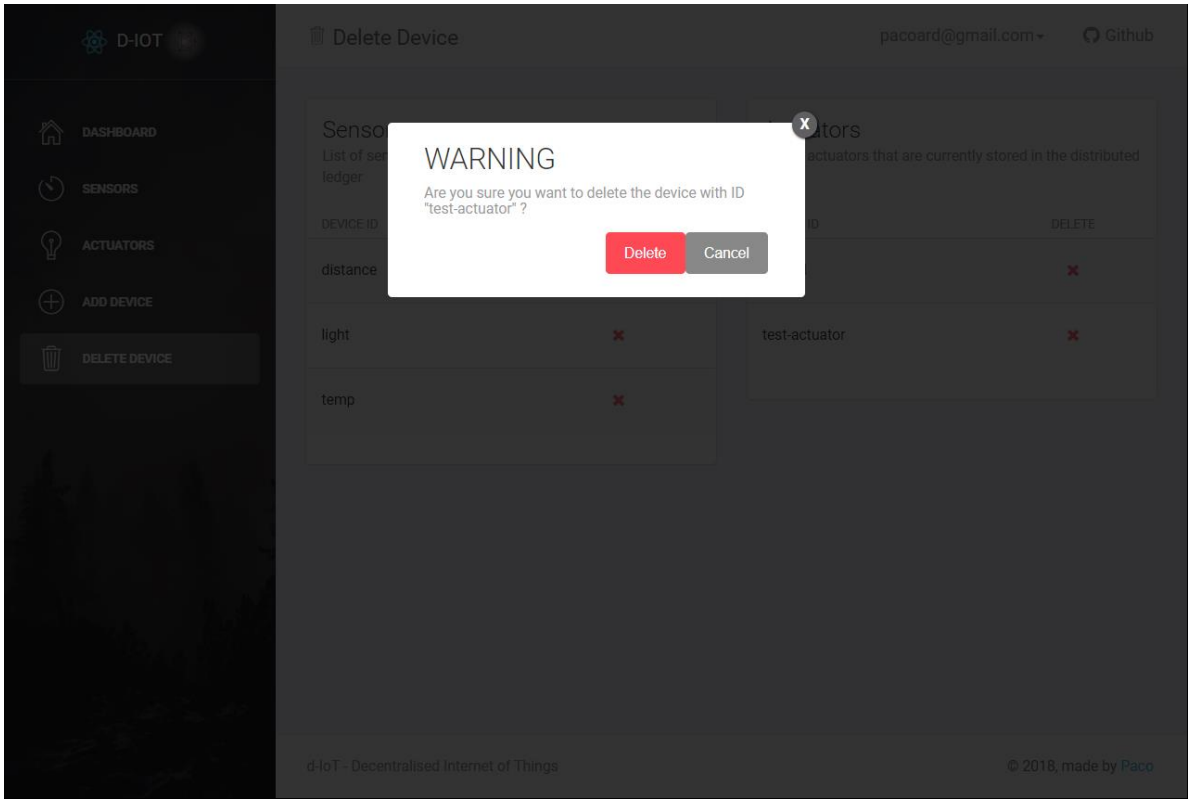


Figure 38: Delete device confirmation dialog

Figure 39: Notification about device deletion

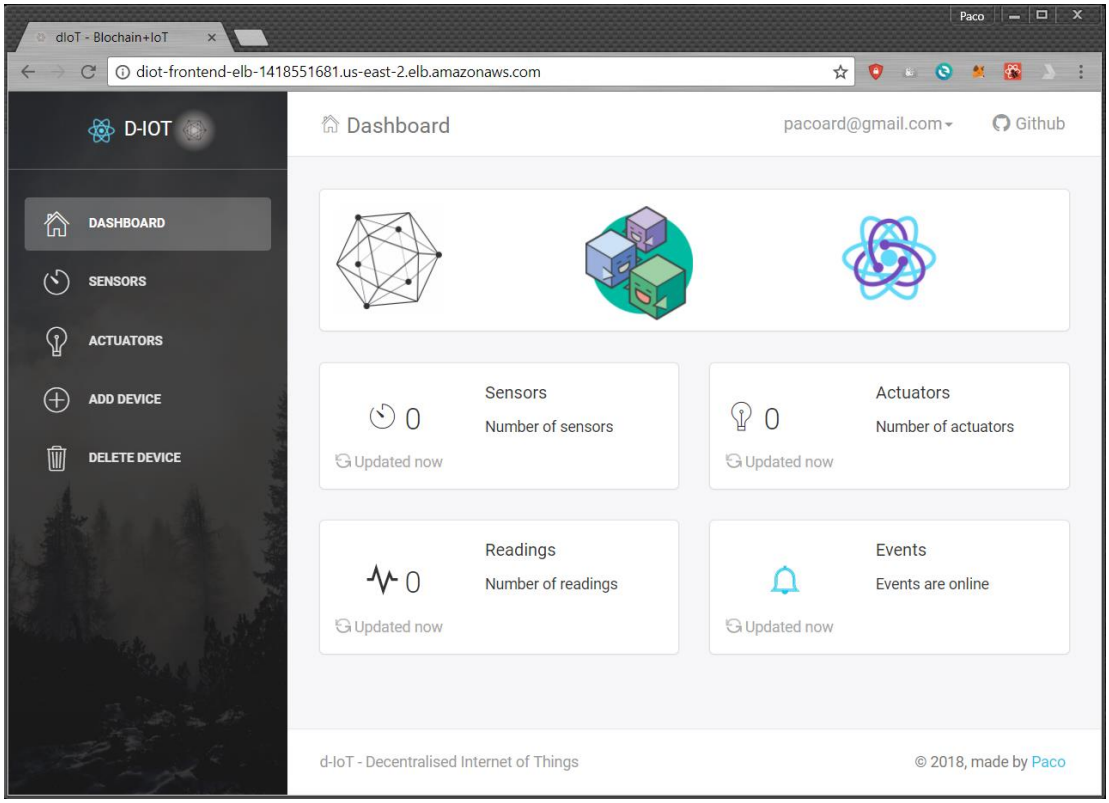
5.4.2 Implementation

The deployment script in Python has almost 200 lines of code, with many comments on each step of the process. It takes the arguments from the command line and adds them to a group of values to be used throughout the program and starts a timer to check how much time it takes to finish. Then, an autoscaling group of EC2 instances is created, configured to run a script that pulls the frontend code from a GitHub repository, installs it and runs it so that it is accessible via HTTP. After that, a load balancer or ELB as it is called in AWS is created and

attached to the autoscaling group, also creating a stickiness policy to sessions so that users and servers maintain communication. After that, the script waits for all of the instances to be up and running and then returns the DNS address:

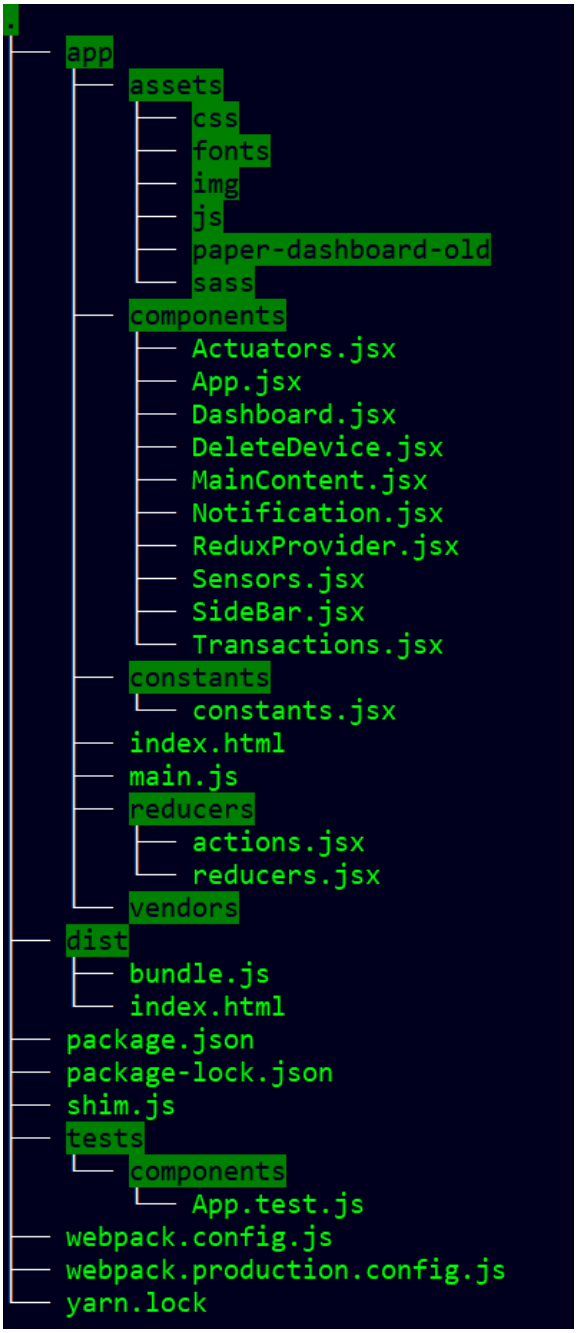
```
Waiting for the service to get up and running...
=> Running command: aws elb wait any-instance-in-service --load-balancer-name diot-frontend-elb

The service is ready to use. Go to the browser and use the ELB DNS address:

diot-frontend-elb-1418551681.us-east-2.elb.amazonaws.com

The blockchain's DNS address is:

ec2-18-220-109-156.us-east-2.compute.amazonaws.com:3000


Execution time:
0:06:25.729187
```

If we access the DNS address shown in the output of the script:



The destruction script is very similar as to its interaction with AWS, but it simply retrieves the metadata of the EC2 instances and ELB that were created in the creation scripts and sends commands to AWS to terminate them.

On the other hand, for the frontend we have the following files:



Once the application is packaged with Webpack (configured in the “webpack” files), the files that contain the whole website are placed in the folder “dist”. The file “package.json” contains all the NodeJS dependencies and scripts to run, test, debug, deploy, etc. as it is a NodeJS application. Then, the “app” folder contains the source code of the website, having a structure of a ReactJS application, being worth mentioning: the “assets” folder contains static fonts, css stylesheets and other necessary files for the web design, the “components” folder contains all the React components that build the website, the folder “reducers” contains the necessary logic of Redux to maintain a global state for the application. Finally, the files “index.html” and “main.js” are the entry point of the application.

5.5 Internet of Things

Finally, the last section of DIOT is the IoT environment that simulates a smart home. This way, we test a client as proof of concept of an ideal implementation of the DIOT system, as can be seen in Figure 20: Ideal smart home powered by blockchain We needed some sensors and actuators that could interact with the rest of the system, therefore we define the following requirements for this part:

- Implement sensors and send readings in the form of JSON messages every few seconds.
- Implement actuators and change their states upon user request by listening to the REST API's WebSocket.

We have chosen to use temperature, light and distance sensors and an RGB LED and a buzzer as actuators.

5.5.1 Design

As we have mentioned in the system overview section, we have chosen Arduino for the management of the sensors because it is easy-to-use open source hardware that can be programmed to read and write its GPIO (General Purpose Input/Output) ports, but lacks computing power to handle JSON parsing and WebSocket communication. That is where Raspberry Pi takes the role of handling the actuators with its superior processor, also somehow working as a hub that should control the whole system. The Arduino is aimed for low-power applications, so we found it suitable to organize the devices this way.

Taking into account that the sensors are connected to the Arduino and the actuators are connected to the Raspberry Pi, the general diagram of the hardware layer of the DIOT system takes the form of Figure 40: DIOT hardwareFigure 40: DIOT hardware:

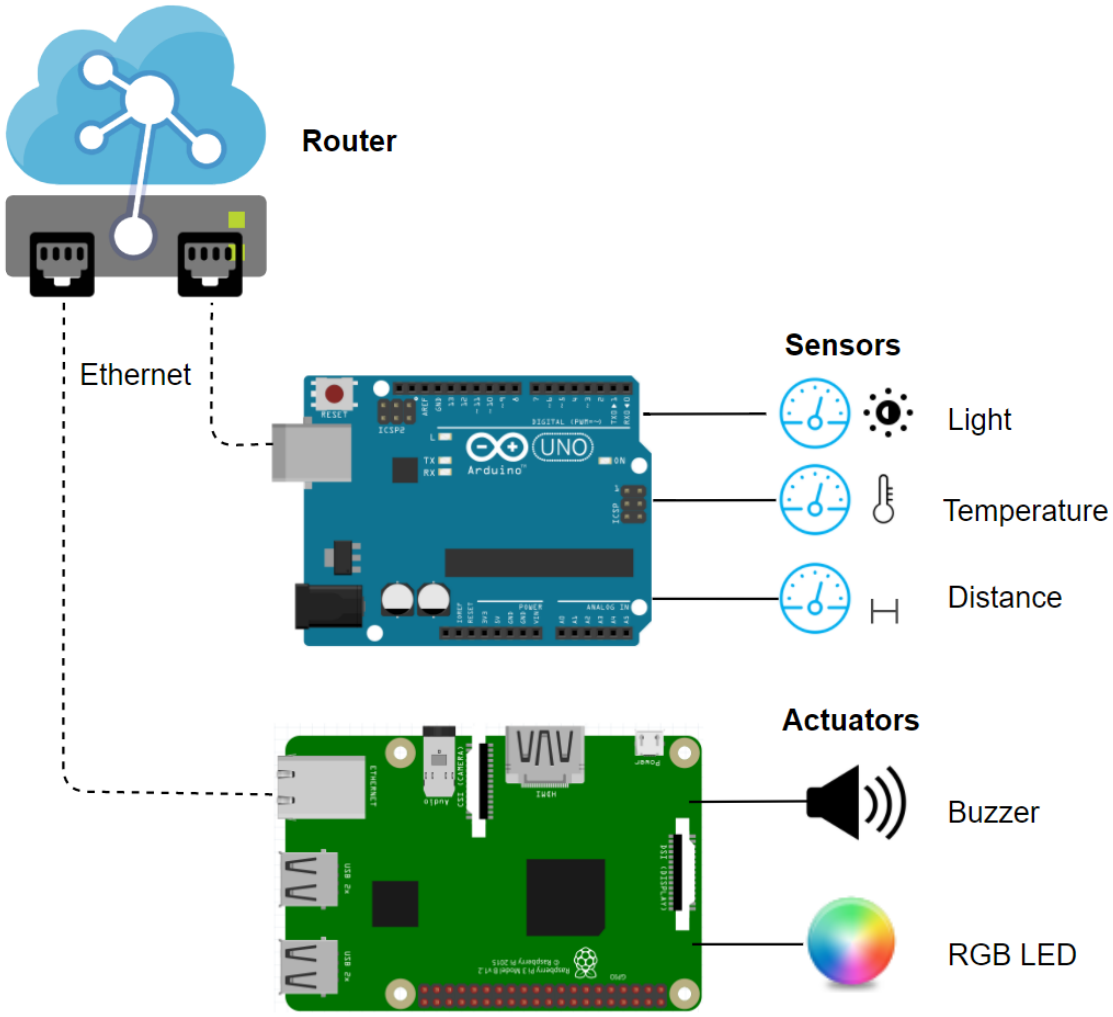


Figure 40: DIOT hardware

According to the diagram, both the Arduino and the Raspberry Pi are connected to the Internet via Ethernet cables. The Arduino needed an Ethernet Shield, which is easily attached to the Arduino. The sensors and actuators are organized in breadboards, as we will see in the implementation section.

The following figure shows a picture of the real system: the Arduino and the Raspberry Pi, the sensors and actuators and all the wiring in the breadboards:

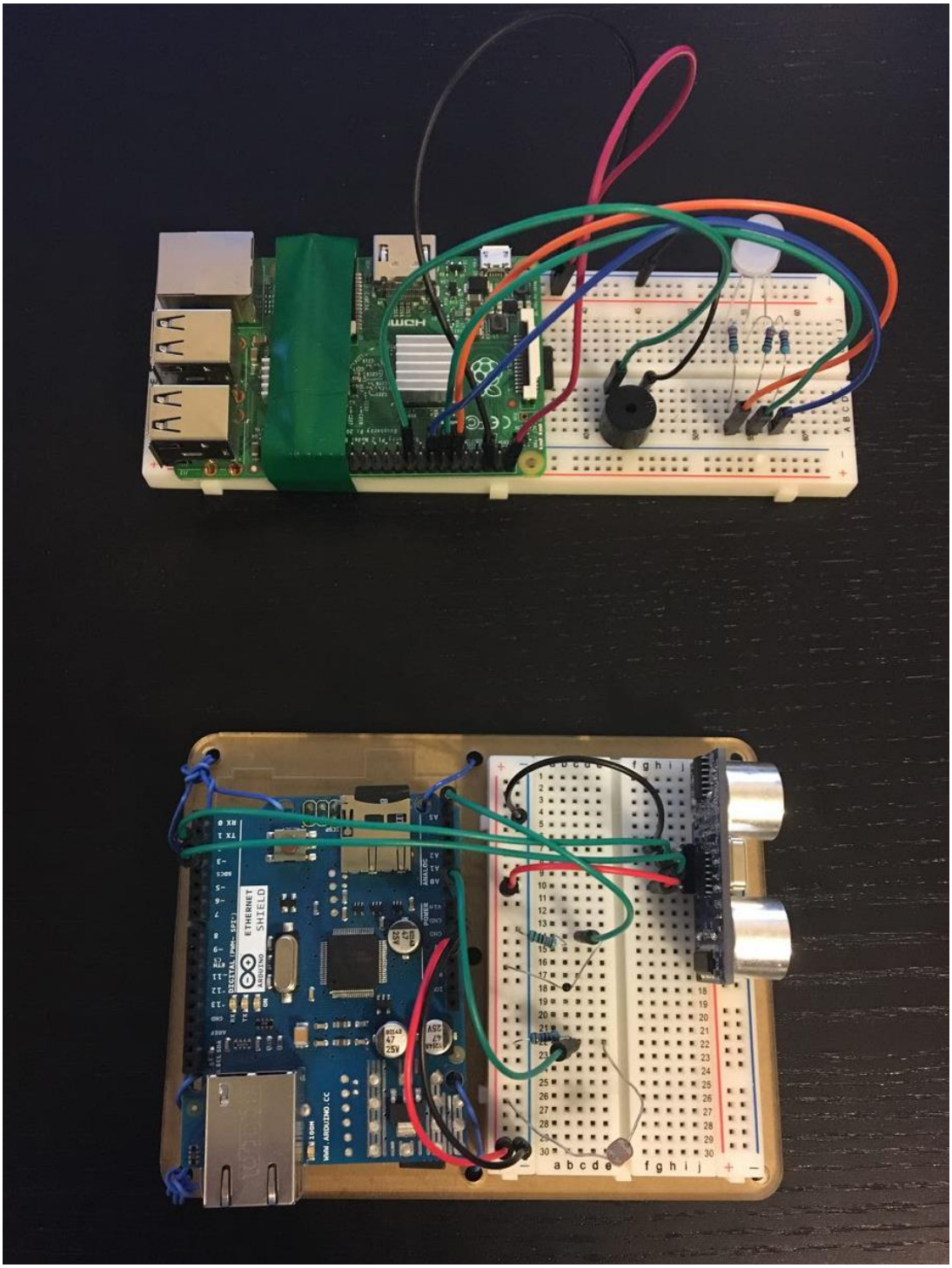


Figure 41: Picture of the real DIOT electronic setup

5.5.2 Implementation

Once the design has been covered, the implementation is limited to the wiring of the system and the code that is running in the Arduino and the Raspberry Pi.

The Arduino and sensors are organized as Figure 42 shows:

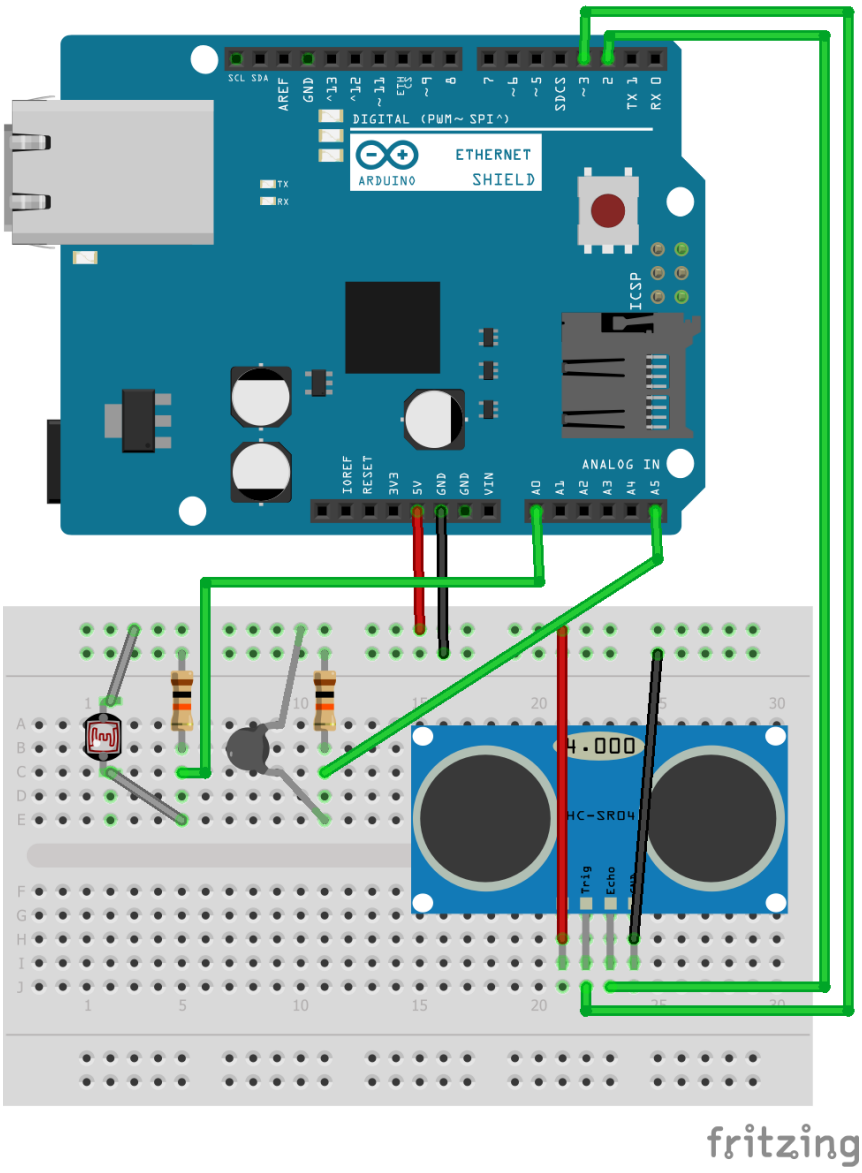


Figure 42: Arduino and sensors wiring

The program running inside the Arduino, written in C, follows a very simple scheme:

1. Import Ethernet library
2. Define and implement methods that retrieve readings from temperature, light and distance sensors.
3. Define and implement a method that builds a JSON string with the name and reading of a sensor and sends it to the blockchain's REST API
4. Open the Ethernet connection with the MAC address of the Ethernet shield and obtain an IP address with the DHCP protocol.
5. Run the loop function forever:

```
void loop() {
  // Connect to the blockchain server
  if (client.connect(server, 3000)) {
    sendJSON("temp", temperatureSensor());
    delay(1000);
    sendJSON("light", lightSensor());
    delay(1000);
    sendJSON("distance", distanceSensor());
    delay(1000);
    client.stop();
  } else {
    Serial.println("connection failed");
  }
  delay(5000);
}
```

The loop function connects to the blockchain's REST API server and sends a reading from each sensor every 5 seconds, waiting 1 second between each reading so that the server is

not overloaded. The REST API should also be replicated in the cloud computing part of the DIOT system, but it is expensive to do so and that is out of our scope.

On the other hand, the Raspberry Pi and the actuators have the wiring shown in Figure 43:

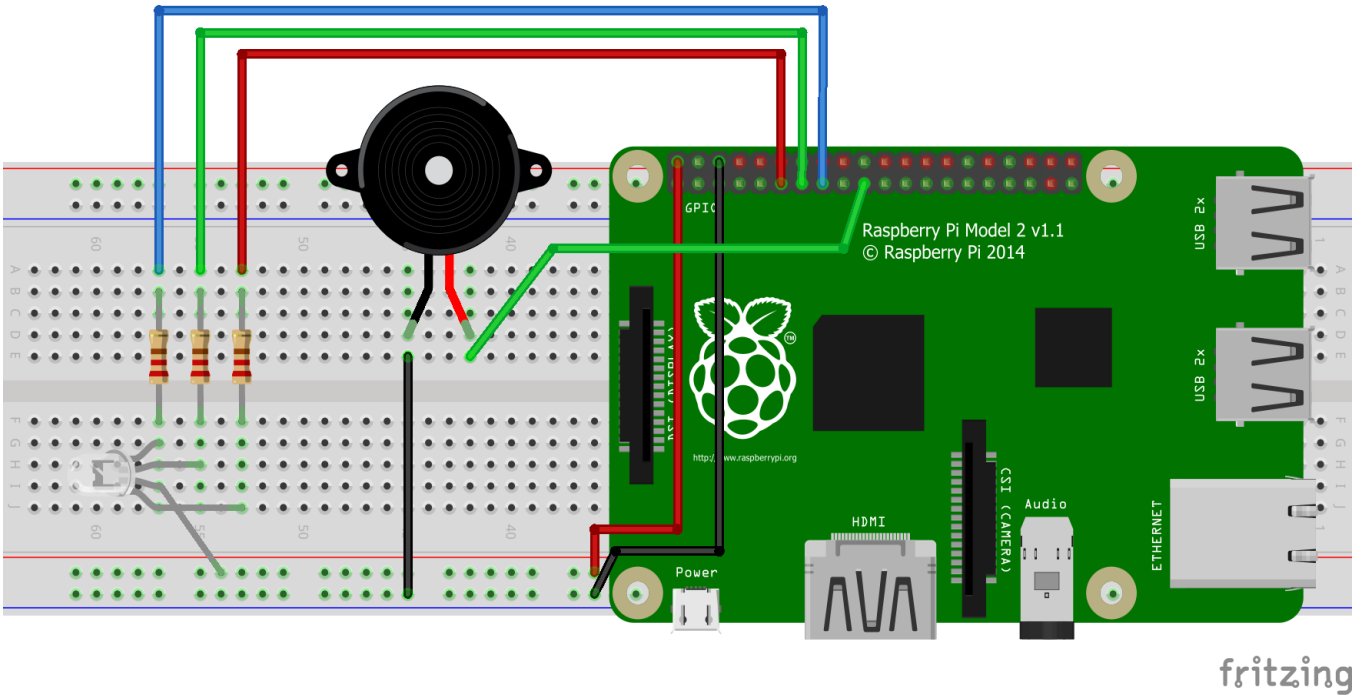


Figure 43: Raspberry Pi and actuators wiring

The Raspberry Pi runs a program in Python instead of C. Given that the Raspberry Pi has an Operative System, it can run programs just like in a computer, without the constraints of a low-power device such as an Arduino. The Python script lets us easily connect to the blockchain's REST API server via WebSocket, so that any change in an actuator's state from the website can be received and handled properly.

The program is configured to turn the RGB LED to the colors red, green, blue, white or none and the buzzer beeps for half a second when its status is changed:

```
# Filter function: delegate actuator events to actuator handlers
def actuator_handle_event(event):
    actuator = event['actuatorId']
    if actuator == 'buzzer':
        buzzer_handler(event)
    elif actuator == 'rgb-led':
        rbg_led_handler(event)
    else:
        print 'No device matches that event.'

def buzzer_handler(event):
    if not event['enabled']:
        rgb_led.color = (1, 1, 1)
        return
    buzzer.on()
    sleep(0.5)
    buzzer.off()

def rbg_led_handler(event):
    color = event['newState']

    if not event['enabled']:
        rgb_led.color = (1, 1, 1)
        return
    else:
        if color == "red":
            rgb_led.color = (0, 1, 1)
        elif color == "green":
            rgb_led.color = (1, 0, 1)
        elif color == "blue":
            rgb_led.color = (1, 1, 0)
        elif color == "white":
            rgb_led.color = (0, 0, 0)
        elif color == "none" or color == "":
            rgb_led.color = (1, 1, 1)
        else:
            print("Not a valid command")
```

6. Conclusions

This project covers many ideas, theory and practical work. The fundamentals of blockchain have been laid out in a thorough theoretical chapter that explains how this technology overcame the double-spending problem with a consensus algorithm in a peer-to-peer network. This opens up a new world of decentralized applications that do not need to put trust on a third party. We have seen some very interesting use cases, as well as the most famous blockchains to this date.

After that, another theoretical chapter on IoT lists its most common protocols and explains the great problems that IoT is facing right now: scalability, security and interoperability. Then, cloud computing is covered in a third theoretical chapter, we see the business model that it has and what its future could be with blockchain.

Finally, we express in the last chapter the motivation behind this project as the desire to solve IoT's problems with blockchain, as well as proposing a different and potentially much cheaper architecture solution for IoT applications. Then, we go over all the work behind a real application of the three technologies feature in the document: a decentralized smart home management system. We tested this system with real sensors and actuators and controlled them via a website that is deployed with a focus on user experience and high availability in a cloud computing environment. We have used and learned many different open source technologies, along most of the protocol stack.

7. Improvement suggestions

There is a lot of room for improvement in this project, if we considered to actually build and sell this product. It is proven that selling a package with all the necessary sensors and actuators for a smart home is successful as a business model, but the electronics were out of the scope of the project as the main purpose was to study the idea of solving IoT's problems with a blockchain solution. Then, some parts of the project were limited due to expenses on electronic equipment or AWS freemium limits. Once that is clear, there are some future lines that could be interesting for this project:

- Development of a full WSN (Wireless Sensor Network) with sensors and actuators spread out in a house, as well as complete design of PCBs (Printed Circuit Boards) with an economic study and detailed cost of the project.
- Smartphone and/or smartwatch applications to control the smart home.
- Once the project is finished, test the speed of transactions, as Hyperledger Fabric claims to be very fast.
- Notifications via SMS or other ways to warn about sensors extreme readings.
- Strict access control, with authentication of identity before being able to manage a smart home. This would be done by issuing certificates in Hyperledger Fabric.
- Develop the ideal system that was proposed in the document: simulate several homes with a blockchain node each so that they participate in the consensus algorithm and make the network secure and scalable.

Bibliography

[1] C. Gutierrez, "Altoros - Decentralized Blockchain as a Solution to IoT Security," [Online]. Available: https://www.altoros.com/blog/decentralized-blockchain-as-a-solution-to-iot-security/.

[2] [Online]. Available: https://www.bitcoin.com/info/what-is-bitcoin-double-spending.

[3] A. Rosic, "Blockgeeks," [Online]. Available: https://blockgeeks.com/guides/what-is-blockchain-technology/.

[4] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System (white paper)," 2008.

[5] S. Nakamoto, "P2P Foundation," [Online]. Available: https://web.archive.org/web/20120529203623/http:/p2pfoundation.ning.com/profile/SatoshiNakamoto.

[6] V. Buterin, "A Next-Generation Smart Contract and Decentralized Application Platform (Ethereum's white paper)," 2013.

[7] "Wikipedia, the free encyclopedia," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Ethereum.

[8] S. Lederer, "Medium," [Online]. Available: https://medium.com/blockmatics-blog/top-10-differences-between-bitcoin-and-ethereum-d2d3dd62101.

[9] "CoinMarketCap," [Online]. Available: https://coinmarketcap.com/. [Accessed 1 April 2018].

[10] "The Substratum Network," 2017.

[11] [Online]. Available: https://substratum.net/.

[12] L. C. David Vorick, "Sia: Simple Decentralized Storage," 2014.

[13] "The Bee Token: The Future of the Decentralized Sharing Economy," 2018.

[14] "About - Hyperledger," [Online]. Available: https://www.hyperledger.org/about.

[15] "Hyperledger Projects - Hyperledger," [Online]. Available: https://www.hyperledger.org/projects.

[16] "Hyperledger - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Hyperledger.

[17] "An empirical examination of consumer adoption of Internet of Things services: Network externalities and concern for information privacy perspectives," [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563216302990?via%3Dihub.

[18] C. Weaver, "MSPAlliance - IoT is really the security of things," [Online]. Available: https://mspalliance.com/iot-really-security-things/.

[19] "TechBeacon - How to design an IoT-ready infrastructure: The 4-stage architecture," [Online]. Available: https://techbeacon.com/4-stages-iot-architecture.

[20] P. Scully, "IoT Analytics - The Top 10 IoT Segments in 2018 – based on 1,600 real IoT projects," [Online]. Available: https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/.

[21] A. Gerber, "IBM Developer Works - Connecting all the things in the Internet of Things," IBM, [Online]. Available: https://www.ibm.com/developerworks/library/iot-lp101-connectivity-network-protocols/index.html.

[22] J. Baker, "Hackernoon - Internet of Everything: The IoT Market Is Projected to Expand 12x from 2017–2023," [Online]. Available: https://hackernoon.com/internet-of-everything-the-iot-market-is-projected-to-expand-12x-from-2017-2023-175f845c2bcf.

[23] G. Daniels, "TELECOM TV - Improving IoT security with smart edge devices," [Online]. Available: http://www.telecomtv.com/articles/iot/improving-iot-security-with-smart-edge-devices-13673/.

[24] S. R. C. C. J. H. Thomas A. Limoncelli, The Practice of Cloud System Administration, Vol II, Addison-Wesley, 2015.

[25] A. Fu, "7 Different Types of Cloud Computing Structures," UniPrint, 3 March 2017. [Online]. Available: https://www.uniprint.net/en/7-types-cloud-computing-structures/.

[26] "Sources: How Do Cryptocurrencies Stack Up To Visa or PayPal?," [Online]. Available: https://howmuch.net/sources/crypto-transaction-speeds-compared.

[27] G. Konstantopoulos, "Scalability Tradeoffs: Why “The Ethereum Killer” Hasn’t Arrived Yet," [Online]. Available: https://medium.com/loom-network/scalability-tradeoffs-why-the-ethereum-killer-hasnt-arrived-yet-8f60a88e46c0.

[28] "Why Vivint Smart Home Is One Of The Most Innovative Companies Of 2017," [Online]. Available: https://www.fastcompany.com/3067476/why-vivint-smart-home-is-one-of-the-most-innovative-companies-of-2.

[29] "Tutorials | Hyperledger Composer," [Online]. Available: https://hyperledger.github.io/composer/latest/tutorials/tutorials.html.

[30] M. G. Zago, "Why the net giants are worried about the Web 3.0," 16 March 2018. [Online]. Available: https://medium.com/@matteozago/why-the-net-giants-are-worried-about-the-web-3-0-44b2d3620da5.