Internet of Things Hardware and Software

Iulia-Antonia BÎRLOG, Dumitru-Marius BORCAN, George-Manuel COVRIG
Bucharest Unviersity of Economic Studies, Romania
antonia.birlog@gmail.com, borcandumitrumarius@gmail.com,
covrig.george.manuel@gmail.com

This paper is meant to describe the interaction between the main components of an IoT system. As the Internet of Things is gaining increasing attention, the overall purpose of the study is to analyze the procedure of transforming signals into processible data. To offer a better understanding of the complexity this modern concept involves, the research guides the readers towards its wide applicability, depicting the main technologies included in both hardware and software components. While these elements are connected and integrated together, a detailed decomposition of the factors involved needs to be illustrated in order to fully demonstrate the significant potential this technology of the future possesses.

***Keywords*:** Internet, hardware, software, physical world, digital world, signals, sensors, smart devices, communication protocol, wireless sensor network, IoT platform, interconnection
DOI: 10.24818/issn14531305/24.2.2020.05

1 Introduction

The Internet of Things generally known as IoT is a wide-spread concept which refers to billions of smart devices existing worldwide. These devices are connected to the Internet and they interchange large amounts of data with the outside world. Lately, several interconnecting solutions have been developed, which resulted in smart cities, smart homes, digital health and other similar automated innovations.

The main technologies involved which are at the root of IoT solutions are: cloud computing, wireless sensors networks, big data analytics, communication protocols, devices with embedded microprocessors. The functionalities provided by IoT systems can be grouped in the following categories:

- gathering the required data;
- connectivity, communication protocols;
- monitoring, control and device discovery services;
- authentication, authorization and data security;
- data analysis and processing, providing the user interface for accessing system functionalities

As the number of interconnected smart devices is exponentially increasing, their variety tends to follow the same trend: from the already common mobile phones, tablets, TVs and other household devices to health monitoring wearables and water or air quality monitors.

The IoT systems developed lately have taken into consideration this kind of expansion in number and diversity of intelligent devices. They followed the standardization of the communication solutions and the interaction with them and also tried to use the provided data by applying the advantages of Big Data, cloud computing and analytics. This involved acquiring, transferring and storing this flux of data in dedicated centralized or distributed (cloud) infrastructures and applying advanced analysis methods on them, using specialized services to identify, extract, synthesize and use the relevant information within.

This emerging paradigm would not have been possible without the evolution of nanotechnology. In the research world, there have been elaborated thousands of studies highlighting the applicability and effectiveness of IoT.

A drastic transformation occurred also on day to day routine, determining people to become adepts of the benefits that the Internet of Things brings along with the unlimited number of possibilities to make their lives easier.

Given the complexity of the subject at hand, there comes the need to split the concept into its main directions of development: hardware and software.

2 Key components of IoT systems

Internet of Things covers a wide variety of innovative technologies, which, along with the modules integrated in hardware components, enable intelligent solutions. An examination of these components designing the paradigm will be discussed in this section.

2.1 The hardware side of IoT

The hardware used in Internet of Things systems ranges from devices for control, dashboards, routers or bridges to servers and sensors. These devices deal with recurring tasks such as threat detection, system activation, security checks and support-specific actions.

The building blocks of any IoT hardware are the following:

- The asset to be controlled or monitored
- Data acquisition module
- Data processing module
- Communication module

The asset to be controlled or monitored can be standalone, or incorporated in the smart device.

The data acquisition module is the one that focuses on acquiring signals from the external world and transforming them into digital signals. It is the hardware block which contains all the sensors, thus helping acquiring real world signals, such as light, temperature, vibration or pressure. This hardware component also includes conversion of the sensor signal into digital information.

The data processing module is the third building block of an IoT device. It represents the unit that processes the data and performs operations on it, as well as storing it. It also requires storage capability. This happens because some data devices process the acquired data themselves, instead of transmitting them upstream. On the other hand, there are IoT devices which do not possess this capability and need intermediary entities in order to store and process the data. These entities are either gateway devices or cloud applications used for further aggregation.

The communication module is the part which enables communication with both third party components and cloud specific platforms

2.2 Bottom-up framework layers and components

From another perspective, an IoT framework can be divided into four main components, which consist of both physical and virtual layers. Each segment of the framework performs a well-defined role in the process of interconnecting the physical and the digital world. These layers are: end nodes, network, services and applications.

Data is the central part in the IoT paradigm, therefore the process of collecting and transforming it is of high importance.

The **end nodes** of an IoT system contain only hardware components which are mostly represented by sensors and actuators. As their name suggest, sensors consist of sensing modules, along with power, energy, radio frequency and management modules. Communication is done using the radio frequency module, through its signal processing feature. The sensor got its name from its most important module, the sensing module. It is used to manage sensing using several active or passive measurement devices, such as: accelerometers, magnetometers, image sensors, light sensors, humidity sensors, pressure sensors, acoustic sensors, gyroscopes, and proximity sensors.

Actuators are components of the physical layer designed to be the counterpart of sensors. These devices intervene in the physical reality, when receiving a signal of control. If they have a source of energy, they will convert it into mechanical motion, according to the signal received.

The **network layer** is responsible for the reliable transmission of information from/to end nodes. The technologies used in this include the Internet, mobile communication networks, wireless sensor networks, network infrastructures, and communication protocols. [1]

Mobile communication networks represent a type of networks which consist of a collection of interconnected terminals, nodes and people using portable mobile devices.

Wireless sensors networks are a group of spatially dispersed sensors used to monitor environmental conditions, such as pressure, temperature, motion or humidity. Typically a

wireless sensor network contains hundreds of thousands of sensor nodes. The sensor nodes can communicate among themselves using radio signals. A wireless sensor node is equipped with sensing and computing devices, radio transceivers and power components. The individual nodes in a wireless sensor network (WSN) are inherently resource constrained: they have limited processing speed, storage capacity, and communication bandwidth. [2]

The IoT communication protocols are the software components of the network layer. They have different capabilities based on the complexity of the IoT solution implemented. The most well-known protocols are: Bluetooth Low Energy, Wifi, Near Field Communications, LoRaWAN, Z-Wave and ZigBee.

Bluetooth Low Energy is designed to keep the power comsumption at a minimum, by keeping the device in sleep mode most of the time. Whenever an event occurs, the device restores from its sleeping state and a short message will be sent to a gateway. This protocol is used for home automation, wireless medical devices or exercise sensors.

WiFi is the most powerful protocol the IoT devices use in order to transfer data to each other, although it consumes high power for the operations it performs. It is preferred over Bluetooth Low Energy due to its wider range for communication, which reaches fifty meters.

Near Field Communication uses electromagnetic induction in order to ease the digital content interchange between two electronic devices. Basically, it extends the capability of contactless card technology and enables IoT devices to share information at a distance of approximately less than four centimeters. [3]

LoRaWAN is a radio transmission protocol through which networks of intelligent objects are created. The established network uses a star-of-stars topology, with gateways serving as transparent bridges, which transmit messages between sensors and the central server. Gateways connect to the network through traditional IP connections, and sensor devices use single-hop wireless communication to one or more gateways. The structure is similar to a mobile network, but instead of having a single interconnected network, LoRaWAN allows the implementation of several independent networks over the same infrastructure. Smart street lighting is a practical example of the system using LoRaWAN protocol, where the street lights are connected with the LoRa gateway that uses LoRaWAN protocol. The gateway, in turn, is connected to a cloud application that completely controls the brightness of light bulbs based on the natural lighting present in the environment. [3]

ZigBee defines a set of high level communication protocols which use small size radio emitters, with a low power consumption. It is based on IEEE 802.15.4 standard which defines Wireless Personal Area Networks. This protocol is generally used in applications which require a small transmission rate of data, thus offering high battery life. The transfer rate for ZigBee is 150 kbps, the most suitable rate for data that is acquired periodically or intermittent, or just even for one signal transmission from an input device or sensor. Due to these factors, ZigBee implies a lower total cost than Bluetooth or WiFi.

Z-Wave is a wireless communication protocol between equipments, projected for automating buildings and homes. It uses a low power consumption wireless communication protocol especially created for remote controlled applications. It is optimized for secure and fast communication of small sized packets, with speed rates of up to 100 kbps. Unlike WiFi and other LAN wireless systems, whose main purpose is high bandwidth flux of data. Z-Wave operates in frequencies of 900 MHz, avoiding interference with WiFi, Bluetooth and other 2.4 GHz systems. Z-Wave equipments are conceived to be easily incorporable in electronic devices and home appliences and most of them are powered using batteries.

The next layer in the framework is the **service layer,** which represents the linking entity between the lower network layer and the

application layer on the top. This component introduces the notion of cloud and fog services, as they provide the required computational power. At this level, all the required computational power is mostly provided as a cloud and fog services. In this layer, cyberattacks target personal and confidential information, IoT end devices, and also monitor and control functions. The impact includes people safety, money losses, and information leakage. Protection mechanisms include encryption, authentication, session identifiers, intrusion detection, selective disclosure, data distortion. [1]

Encryption is used to protect against gathering security attacks, such as skimming, where the transmitted messages are quickly read for data abuse, or eavesdropping, where raw data about the target is collected.

Authentication represents the validity of the interactive things. The most common authentication modes are based on shared secrets/keys, the entities identity and a trusted third-party. [4]

Session identifiers represent a random sequence of characters, letters and numbers that is stored on both client and server side, in order to be able to identify that session. It has to be protected with a powerful cryptographic mechanism.

Intrusion detection is used to detect malicious attacks and to enable the systems and communications in secure status.

Selective disclosure and data distortion are used to protect against group privacy disclosures, such as commercial espionage.

The **application layer** provides a mechanism which allows the integration of the services included in the bottom layer via mobile techology. This layer is responsible with data sharing along the IoT network, thus it needs to ensure security, data privacy, access control and to prevent information leaks.

2.2 IoT programming by architecture levels

Because of the very wide range of applications, IoT systems are very complex by nature and imply programming skills from a large variety of domains. A complete IoT system requires a large number of different interconnected devices, and it is precisely this variety in device types and forms that makes IoT programming a consistent effort covering a large number of areas.

IoT programming can be structured in three main areas: cloud components, gateways and end devices.

Cloud components are the central parts of an IoT system. A cloud component is a collection of cloud services, where each service is usually responsible with one or more functionalities of the entire system. With the large number of features that IoT systems cover nowadays and the increasingly large amounts of data to manage, the trend is more and more to follow a Single Responsibility principle and therefore to allow a cloud service to take care of exactly one functionality of the system.

With cloud services being the critical part of an IoT system, they need to be highly available and to run fast and with a data and memory footprint as small as possible. Therefore, for these types of services, fast, resource efficient programming languages are preferred.

One additional requirement for cloud services is that they need to be stable and these requirements translate to the choice of programming languages and technologies used to build these systems.

Good choices of programming languages used to build cloud services can be Java, C# or even C. They are stable, they have been used for years, they are well documented and there is a tremendous amount of engineers working with these programming languages.

Other options could also be Python or Node.js. Not being strongly typed, they are less stable, but the trade-off is that they are faster and they offer much more processing power. In addition to that, the community behind these programming languages is growing continuously, meaning more and more engineers are available to work on these kinds of services.

Data management is also a central part of a cloud system that lives in the cloud. Depending on the purpose of the system we may need different types of database systems. The IoT

system requirements from a data perspective cover three main areas: data storage, data retrieval and data visualization.

Data storage should be done in a consistent and efficient manner. When we talk about consistency, we usually mean that all devices connected to an IoT system should see the same data at the same time. This of course is an ideal way, but highly impossible in a large IoT system. That is why data latency comes to discussion. This refers to the amount of time it takes from the moment of a change in the database to the moment that all devices can be aware of this change. For small IoT systems this is possible in real-time, but for larger architectures near-real-time consistency is also acceptable by today's standards. In these cases, IoT systems are built in such a way that real-time consistency is achieved for critical data and near-real-time consistency is achieved for less critical data.

Storage space efficiency is also desirable for IoT systems. Because most of the data today is stored in the cloud, more and more data centers are built around the world to support these types of architectures. At a certain point in developing an IoT system, developers should take into consideration storing only data that needs to be stored and to consider on the spot processing for data that does not necessarily need to be stored.

More important is the aspect of data maintenance and data cleanup. Especially for real-time IoT systems, there comes a moment in time where some of the data can be rendered useless by the state of the system or by the system requirements. Therefore, it is important for developers to consider building services that maintain and clean up data that is no longer needed in an efficient manner, so that the space can be reused in the future.

Data retrieval is necessary so that the devices connected to an IoT system receive fresh data which may be required for a state change or for other behavior aspects. As with data storage, IoT systems developers should strive to obtain real-time accuracy, but it is acceptable for large IoT systems to pass data to end-devices in near-real-time.

The data retrieval process should be precise, focused on exactly what is needed and not to leave a considerable amount of processing power print on the overall system. End clients should be programmed in such a way that when they request data, they request only the data they need so that they do not clutter the data pipeline and the system. Moreover, in order to obtain two-way efficiency, cloud services should also ensure and enforce the fact that clients request and receive data on a need to know basis.

Data visualization helps the users of an IoT system to observe the state and the events of an IoT system. Here, the same principles with regard to data latency as for data storage and data retrieval apply. Data visualization components should obtain read-only access to the data of a system and by no means be able to modify it. Moreover, data visualization should not hinder the entire process and workflow of an IoT system and that means efficient data visualization components have only extractive roles. For example, if an event is happening in the system and we need to visualize that event, neither the event trigger nor the event observer should waste time and processing power with notifying data visualization components. Instead, data visualization tools should connect to the event logs database and extract only what's needed in order to perform visualization.

Popular among data management solutions for cloud and IoT devices are solutions like Apache Lucene, Elastic Search (which is built on top of Apache Lucene), Hadoop or Apache Hive.

Virtualization is also an important part of an IoT system architecture. This is achieved by using special technologies for virtualization and is needed for quickly and gracefully deploying new components (or new component clones) of various parts of the architecture. Data virtualization is also used for scalability purposes and for load-balancing. In the past, virtualization used to be done by using virtual machines. But that technology is now rendered outdated by the fact that real time IoT architectures require faster processing and more dynamic structures.

As mentioned earlier, the gold standard nowadays is to build services which are highly specialized. This is useful from two perspectives. First is that this helps developers achieve simple, loosely coupled, fault and change tolerant architectures. The second is that this standard helps with virtualization. By building decoupled services, we allow them to be encapsulated in virtual containers which can then be deployed in flexible and dynamic manners.

Flexible deployment and loose coupling help IoT systems in the following situations:

- A new component needs to be built without altering or stopping other existing components
- A component that is no longer needed needs to be torn down, scraped or discontinued
- There are large fluctuations of traffic in the system. In this case, we can have multiple copies of the same component (so multiple virtual containers) for the components that are constantly affected by traffic fluctuations. In this way, the number of containers can be dynamically scaled up or down with the respect to the current level of traffic.
- Only applicable for large, global IoT systems: there is much traffic in some specific geographic area and less traffic in other areas. In this case, we can multiply our containers and balance the load of the system to containers located physically closer to the regions where we register higher traffic (the principle behind CDN - Content Delivery Network - systems.

The most popular virtualization technology nowadays is Docker, which is mostly used in tandem with Kubernetes, a container orchestration platform.

Docker is a Platform-as-a-Service (PaaS) product which offers efficient, operating system-level virtualization for cloud services.

Cloud services can be isolated in containers which are reliable, secure and efficient in terms of memory and speed, because they bundle their own services and components with those of the host operating system. Even so, containers are perfectly isolated from the host, offering high security while not giving up on flexibility features.

Individual service containers are called "Workers", and as all containers in the Docker ecosystem, they can communicate with eachother because they keep ports open for this purpose. Docker workers can be clustered together with a service called Docker Swarm, and in this way, the workers will be able to communicate only with workers in the same clusters, but not with workers from other clusters or with external services. This is necessary for security purposes.

Docker workers can also be scaled up and down in a flexible way and they can be maintained with simple commands because of an architectural feature called "masters". A master is responsible for handling all the requests for its workers, and this means handling all workers in a cluster is as straightforward as handling the master of that cluster.

The masters in a Docker network can communicate with each other by the same principle as with workers, based on a distributed internal state store which is accessed by all the masters in a network. In this way, all masters can contribute to the store and all masters can be notified at the same time.

System monitoring components are responsible for watching data, events, traffic and flows throughout an IoT system. If any fault happens in the system, then the system monitoring component is responsible for sending out alerts with detailed information about the fault, the cause and ideally possible fixes. For simple, predictable faults, a smart system monitoring component can also be programmed to automate fixing and restarting the system, restoring it to a valid state.

After cloud components, **gateways** are the next main node of an IoT system. Gateways still live in the cloud, but their sole responsibility is to route flows throughout the system. Requests from clients (end devices) go first through a gateway, which does an initial validation of the data and the request. Here, the request is also authenticated and authorized. Then the data is passed on to the correct cloud service. The response follows the same path back through the gateway and on to the end

device.
Because of their role as an intermediary between all requests in the system, gateways should be extremely fast and should have as little footprint as possible on the overall flow. This is why, for gateways, programming languages with non-blocking I/O and with multi-threading features are preferred. Node.js, Go and Java are common examples of programming languages used for building gateway components.
With API (Application Programming Interface) Gateways, all requests need to be authenticated and authorized before being accepted and processed. So the first time a client (be it a browser, a user or another application or device) can use the gateway, it needs to be authenticated and authorized.
Usually, for IoT networks where we have a large number of inter-connected services and gateways, we can use an authentication scheme called Single sign on, which will allow the client to use the same credentials for several related services.
With single sign on, the request is first authenticated ("Is the client who pretends he is?") and then authorized ("Is the client allowed to do the action he wants to do? Is he allowed to access the resource he is trying to access?"). If both validations pass, the client is offered a token that he can use the next time he tries to do the same operation. The token can be used with an expiration time, then single sign on will have to be performed again.
The gateway has information about the validity of a token, so when the client will offer a token, this will be validated. Only if this additional validation passes, then the gateway will fetch the resource from the Resource server or will perform a certain action.
End devices or edge devices are the physical devices which connect to an IoT system. They are responsible for detecting data, sending data to the cloud components and also reacting to commands sent back from the cloud. They usually have limited processing power so they need to be programmed and configured with very fast, low level programming languages, like C or C++. For more advanced devices(like Arduino boards for example), more advanced, high level programming languages can be used, and popular choices are usually Python and Java.

2.3 Hardware and Software interaction flow

As **Fig. 5** illustrates, there is a complex data flow describing the interaction between the hardware components and the software solutions available.
The normal data flow through an IoT network starts from the main source of data available in the architecture that is the end devices or edge devices. These devices collect various forms of data from the users and pass it on through special gateways which will redirect the data to a main Cloud gateway.
From here on, all the data is processed and streamed, with several control applications deciding when and where this data should be passed on. After the decisions in these flow control nodes, the data can already be visualized using Web and Mobile applications.
The storing of the data happens in and is controlled by data warehouses, which will maintain the data in a fast and efficient way. Also it is normal that data processing happens at this stage. Various operations can be applied to the data so that efficiency is increased or new data/metadata is obtained from existing information. With the data being stored, it can now be used for more advanced data visualization techniques or it can become training, testing and production data for Machine Learning models.

2.4 Machine learning in IoT

On a very basic sense, machine learning in technology today is the process of elimination of human intervention wherever possible. It is allowing the data to learn patterns by itself and take autonomous decisions without a coder having to write a new set of codes. [5]
Generally, machine learning is valuable when a large amount of data exists, but its potential is not capitalized. This happens because of the lack of knowledge regarding the importance of each decision variable in making the decision.
The philosophy behind machine learning is to

automate the creation of analytical models in order to enable algorithms to learn continuously with the help of available data. [6]

With the big influence Big Data started to have in IoT, there comes the need for machine learning algorithms to begin transforming the data stored into valuable information, eventually used for reducing cost, increasing efficiency or even making predictions. This way, data sets resulted from IoT hardware components, which in most of the cases are sensors, are processed and the information extracted from algorithm is then interpreted and used for future decisions.

Depending on the learning style, ML algorithms can be grouped into four categories:

- Unsupervised learning: Algorithms in this category try to identify patterns on testing data and cluster the data or predict future values. [7]
- Supervised learning: Supervised learning deals with problems involving regression such as weather forecasting, estimating life experience, and population growth prediction, by using algorithms like Linear Regression or Random Forest. Additionally, supervised learning addresses classification problems such as digit recognition, speech recognition, diagnostics, and identity fraud detection, by using algorithms such as Support Vector Machines, Nearest Neighbor, Random Forest, and others. There are two phases in supervised learning. The training phase and testing phase. [8]
- Semi-supervised learning: This is a combination of the previous two categories. Both labeled data and unlabeled data are used. It works mostly like the unsupervised learning with the improvements that a portion of labeled data can bring. [7]
- Reinforcement Learning: In this learning style, the algorithms try to predict the output for a problem based on a set of tuning parameters. Then, the calculated output becomes an input parameter and new output is calculated until the optimal output is found. Artificial Neural Networks (ANN) and Deep Learning, which will be presented later, use this learning style. Reinforcement learning is mainly used for applications like AI gaming, skill acquisition, robot navigation, and real-time decisions [7]

Machine learning algorithms have proven their importance during the IoT era. They became key components in decision making processes, starting from garage companies up to IT behemoths.

A great example is Google's application of machine learning to its data centers last year. Data centers need to remain cool, so they require vast amounts of energy for their cooling systems to function properly (or you could just dunk them in the ocean). This represents a significant cost to Google, so the goal was to increase efficiency with machine learning.

With 120 variables affecting the cooling system (i.e. fans, pumps speeds, windows, etc.), building a model with classic approaches would be a huge undertaking. Instead, Google applied machine learning and cut its overall energy consumption by 15%. That represents hundreds of millions of dollars in savings for Google in the coming years.

4. Conclusions

The Internet of Things has contributed to a revolutionary metamorphosis of people's cognition and manner of perceiving the environment. As the solutions of interconnecting intelligent devices are evolving, the adaptation to the newly outlined reality becomes more challenging. IoT has already become a way of living, and its extensive directions of development have determined both researchers and developers to work together and merge their potentials. This approach has visibly led to increased quality of delivered services. Software developers have become more attracted by the idea of adopting virtualization and modern programming paradigms, while a competitive approach has been observed among hardware producers in IoT.

The contribution and importance of cloud computing services, and machine learning cannot be denied in the evolution of this phenomenon. Not only did they become essential components of IoT systems, but they also

started using their power and complexity in order to generate accurate decisions for future development improvements.

References

[1] J. Pacheco, "Anomaly behavior analysis for IoT sensors," 2 May 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/ett.3188?casa_token=iNjeiZUqcl0AAAAA%3AZLMFxT8EoQCOmdKv4nbfCz2VkZxglAJo8r_zWKsbdMpDuxOfAtkKvtiuCkb5mSLuQfvVsvyPgJMN.

[2] M. Matin and M. Islam, "www.intechopen.com," 6 September 2012. [Online]. Available: https://www.intechopen.com/books/wireless-sensor-networks-technology-and-protocols/overview-of-wireless-sensor-network.

[3] T. Taylor, "6 Most Commonly Used IoT Communication Protocols," 2 November 2019. [Online]. Available: http://techgenix.com/iot-communication-protocols/.

[4] H. Ning, "Security and privacy" in "Unit and Ubiquitous Internet of Things", 2013.

[5] A. Sekar, "How To Apply Machine Learning Algorithms To IoT Data?" 30 November 2018. [Online]. Available: https://analyticstraining.com/how-to-apply-machine-learning-algorithms-to-iot-data/

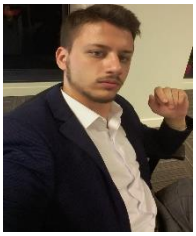
[6] M. Tanskanen, "Applying machine learning to IoT data" 2 May 2019. 30 November 2018. [Online]. Available: https://www.sas.com/ro_ro/insights/articles/big-data/machine-learning-brings-concrete-aspect-to-iot.html

[7] M. Kubat. "An Introduction to Machine Learning" Springer: Cham, Switzerland, 2017

[8] F. Zantalis, "A Review of Machine Learning and IoT in Smart Transportation" 19 March 2019.

Iulia-Antonia BÎRLOG has graduated in 2018 The Faculty of Economic Cybernetics, Statistics and Informatics, bachelor's domain Informatics. Currently she is a Java Software Engineer with a demonstrated history of working in the financial and research industries. She is passionate about science and technology, machine learning, high-level programming languages and Java related frameworks. She coordinates a small team towards innovation, applying the knowledge she acquired during the university years.

Dumitru-Marius BORCAN has graduated in 2018 The Faculty of Economic Cybernetics, Statistics and Informatics, bachelor's domain Informatics. He is currently a Java Software Engineer and has a professional working background with a large number of web technologies. Passionate about computer science, machine learning and technology in general, he is also maintaining a machine learning research blog.



George-Manuel COVRIG is a software developer. Passionate about programming and machine learning enthusiast. He works as an application development lead for a software company with various responsibilities including mentoring, tutoring and helping the team go through the most difficult times. In 2018, he graduated with a bachelor's degree at The Faculty of Economic Cybernetics, Statistics and Informatics.