Novel IoT Development Kit for Personalized Smart Ecosystems: Aliot

Jihene Rezgui, Enric Soldevila
Laboratoire **R**echerche **I**nformatique **M**aisonneuve (LRIMa)

Montreal, Canada
jrezgui@cmaisonneuve.qc.ca

Abstract – **The Internet of Things (IoT) is a growing field in computer science that needs more experts than ever before. In fact, there are currently 12.2 billion active connections of things, and it is predicted that this number will go up to 27 billion by 2025 [1]. Creating a complete IoT ecosystem is a time-consuming task and most of the time represents unnecessary trouble for a newcomer in this field. In this context, this paper proposes Aliot, an advanced development kit designed to help researchers and learners in their IoT projects. Aliot offers flexible tools available in two of the most popular programming languages in IoT, Python, and C++. It handles most of the required layers in a connected ecosystem while offering a lot of freedom to the end-user. It provides a reliable and secure connection for multiple connected devices with many unique features such as real-time monitoring and data management. Aliot was tested and used in many research projects, such as a smart city and a connected greenhouse. Preliminary results show that Aliot outperforms the traditional approach of developing an IoT ecosystem by reducing the amount of code written by more than 80% and 10 times faster to develop.**

Keywords: Aliot, Internet of Things, Connected Ecosystem, Smart City, ALIVEcode.

I. INTRODUCTION

The number of things connected on the Internet increases by 18% [1] each year. Along with the almost exponential growth of IoT connections, creating IoT solutions is a complex task that only a small amount of people can do. Building a complete, reliable, and secure IoT ecosystem is a time-consuming task, and it is a major challenge to manage. Furthermore, IoT solutions are becoming a must in the industry as well as in scientific research. A comprehensive study [2] of representative works on IoT network management highlighted and compared existing IoT solutions. Many of these solutions [3-7] have already offered a way for companies to develop and launch their personalized IoT ecosystems but at a financial cost. Moreover, some of these solutions were specialized in only one field. This can be ideal for some specific projects, but too restrictive for generalized projects. We would like to thank the previous projects [3-7] for their contribution to the advancement of research in IoT. Nevertheless, these previous tools lacked flexibility and were not accessible to the average user. For this reason, we proposed Aliot, which is an open-source [8-10] multipurpose free set of tools. Aliot can do simple things, such as monitoring the temperature in a room in real time, to more sophisticated things, such as creating a fully connected smart city to discover modern solutions to traffic. Thanks to its simplicity, we believe Aliot could be used as a set of tools for researchers in IoT and in AI. It can also facilitate data collection in numerous and various applications, such as healthcare, intelligent transport, etc. Aliot could especially be useful in AI research projects that need a large data feed in real time. For example, Aliot was used to visualize brain waves in real time and remotely control a motorized vehicle based on the driver's drowsiness using Machine Learning algorithms. It is worth noting that a preliminary version of Aliot was introduced during an ACFAS congress [11].

Our contributions in this paper can be summarized as follows: **(1)** We created a custom communication protocol to fully use Aliot's services; **(2)** We added customizable visualization components to display the ecosystem's data in real time; **(3)** We implemented a simple and easy-to-use private NoSQL database available in each ecosystem with unique features, such as real-time monitoring; **(4)** We designed Aliot libraries in Python and C++ to facilitate the use of our communication protocol; **(5)** We conducted a research on IoT, and AI-powered smart cities with Aliot; **(6)** We compared an IoT ecosystem's creation with Aliot's approach against the traditional approach, in which each layer has to be developed; **(7)** We introduced Aliot to college students in a specialized IoT course and we let them create their own ecosystem to get their feedback and improve Aliot consequently.

Section II gives a brief overview of similar tools and compares them to Aliot. Section III presents Aliot's architecture and its components. Section IV describes all the layers of Aliot. Section V shows our results. Finally, section VI concludes the paper.

II. SIMILAR TOOLS AND SOLUTIONS

Various projects also shared a similar goal to ours by creating an easy-to-use customizable IoT ecosystem. We will compare two different platforms: **a)** SYNAISTHISI: *An Enabling Platform for the Current Internet of Things Ecosystem* [3] and **b)** SEnviron: *An IoT Platform Based on Microservices and Serverless Paradigms for Smart Farming Purposes* [4].

A. Synaisthisi [3] vs aliot

SYNAISTHISI is a platform that offers the creation and management of IoT services from research to business areas. The main difference with Aliot is its accessibility. The SYNAISTHISI platform and its services are only accessible to chosen parties with money, whereas Aliot is available to everyone and completely free to use. We also widen our spectrum of targeted people to the newcomers in IoT who have never created IoT solutions before. This was possible by offering them complete courses on how to get started in IoT and create your own IoT solution.

B. Senviro [4] vs aliot

SEnviro is also a web platform made to create and manage complex IoT ecosystems. This platform is specialized in monitoring the status of the current environment, such as air quality, humidity, lighting, etc. What differentiates Aliot is its flexibility. SEnviro is specialized in farming IoT devices, whereas Aliot is purposely made to support any type of IoT device. Aliot could be convenient to make any kind of ecosystem. Its limits are only caused by the practical limitations of the hardware used.

III. INTRODUCTION TO ALIOT'S ARCHITECTURE

Aliot works alongside an already existing project named ALIVEcode [12]. ALIVEcode is an open source [8] web platform used to teach the different fields of studies in computer science. It specializes in the Internet of Things, Artificial Intelligence, and coding. The specialized branch in IoT of ALIVEcode is named ALIVEIoT, in which the visualization tools and the management tools of Aliot are accessible to the end-users. It implements a customizable Graphical User Interface (GUI), a NoSQL private database, permissions management of the connected devices, and automation scripts. The following **Fig.1** describes Aliot's architecture and its key components.

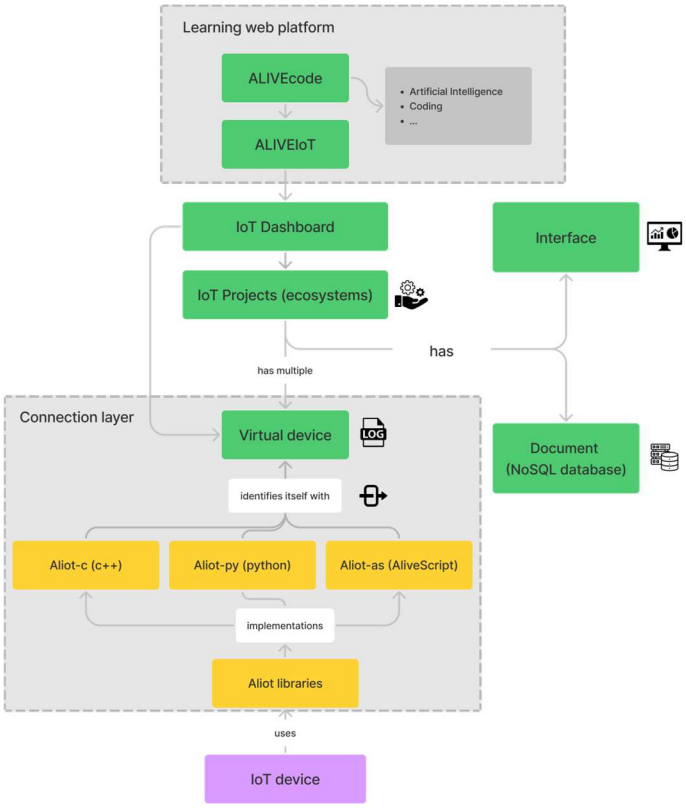


Fig.1. Aliot's architecture and its key components

Subsection A will present the IoT dashboard on ALIVEIoT. Subsection B will explain the virtual devices on ALIVEIoT, also referred as IoT objects. Finally, subsection C will define IoT projects on ALIVEIoT, also called IoT ecosystems.

A. Iot dashboard

On ALIVEIoT, the user has access to a personal IoT dashboard as shown in **Fig.2**. In this dashboard, the user can manage and access all his virtual devices and ecosystems.

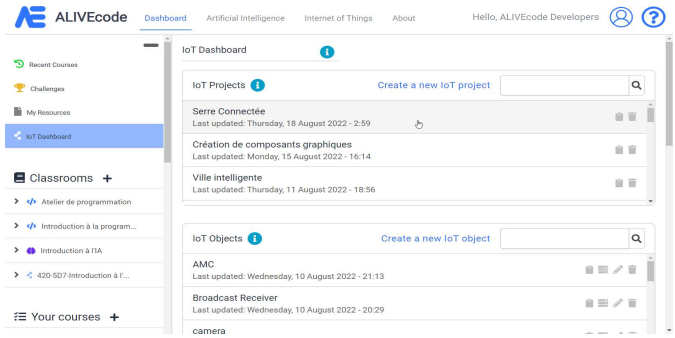


Fig.2. ALIVEIoT: IoT Dashboard

B. Virtual devices

A virtual device is the virtual representation of a user's IoT device on ALIVEIoT. Aliot handles secure and private connections by using a key-based authentication system. Each connected device needs to be registered on ALIVEIoT, in other words, it needs its virtual representation to be created. When registered, a unique identifier (uuid4) is generated. This unique identifier is used and verified by Aliot's services in each request to ensure protection against malicious third parties. This also enables permissions management for each IoT device. For example, some IoT devices might be able to interact with a specific IoT project (ecosystem) while others do not. Additionally, logs of every request are kept and made available in the IoT dashboard to the owner of the connected object.

C. Iot projects (iot ecosystems)

On ALIVEIoT, you can create your own ecosystem in which you decide the permissions of the connected devices and the visibility of your ecosystem to the public. In a project, you have access to the customizable GUI, the NoSQL database, the virtual devices, and the automation scripts.

IV. LAYERS OF ALIOT'S IOT ECOSYSTEM

Aliot is the name given to the set of services and tools we created to facilitate the accessibility of the Internet of Things for researchers and learners.

Our services include notably: **(1)** Bi-directional communication with IoT devices; **(2)** Data sharing between connected devices; **(3)** Real-time triggers and events on a private NoSQL database; **(4)** Automation scripting using AliveScript [13], a homemade language without any IDE or software.

Our tools include in particular: **(1)** Two libraries to use Aliot's services, one in C++ and one in Python, two broadly used languages in the IoT industry; **(2)** Real-time visualization of an ecosystem's data and its sensors metrics in a fully customizable GUI; **(3)** IoT ecosystem management on ALIVEIoT.

All IoT layers of Aliot and their inner workings are shown in **Fig.3**. The figure reads from bottom to top. The following subsections will talk about all the layers in an IoT ecosystem and how Aliot works and contributes to each one of them.

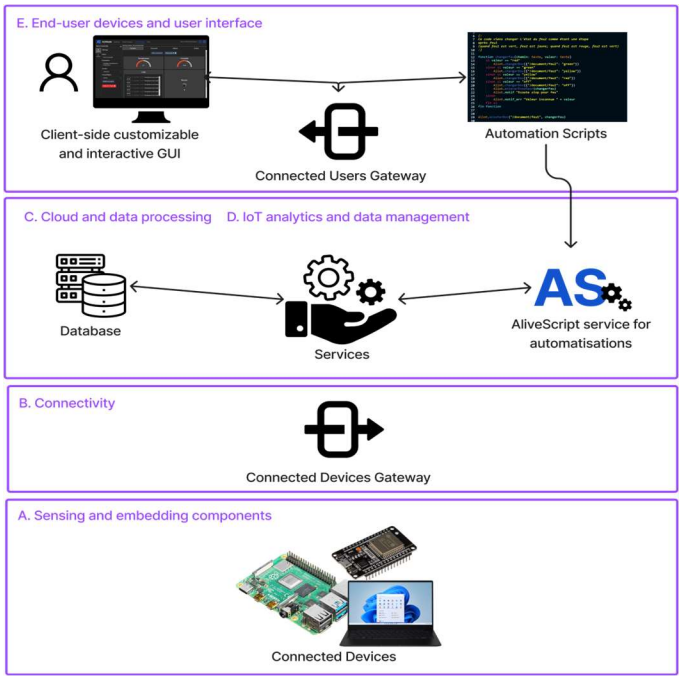


Fig.3. Layers of Aliot's IoT ecosystem

A. Sensing and embedding components

Aliot accepts every Wi-Fi-enabled device. The sensors and actuators can be anything and have no impact on Aliot's performances. Some of the components using Aliot are presented in **Fig.4** below.

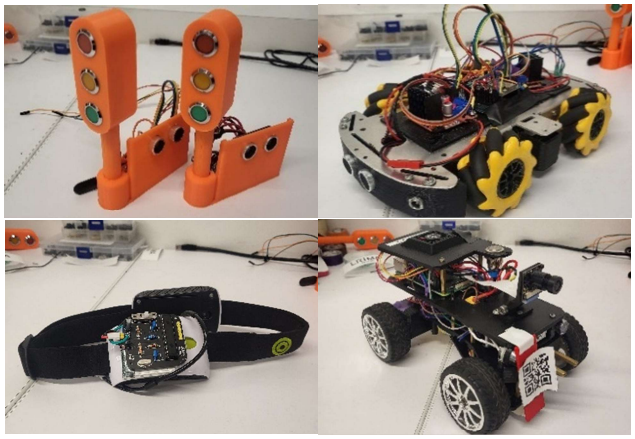


Fig.4. Some Aliot-powered IoT devices

For demonstration and research purposes, we already built many devices [14], such as a connected greenhouse, a connected light, connected cars, a connected bridge, a connected streetlight, and a connected headset that reads brain signals.

[1] This code is written in English, but it is possible to write the code in French with a French version of AliveScript.

B. Connectivity

Aliot is based on TCP packets traveling over the secured HTTPS protocol. To enable bi-directional communication, we make use of the WebSocket communication protocol [15] that creates a full-duplex communication, which is a TCP long-lasting bi-directional connection between our server and the connected device. Throughout the connection, it is ensured that the connected device is registered and allowed to perform certain actions with the unique identifier of its corresponding virtual device. This is done in the IoT gateway that ensures secure communication. We built this expendable and modular gateway with its own REpresentational State Transfer Application Programming Interface (REST API) to redirect data flow or utilize certain available services in the Cloud. Additionally, we developed two different libraries which utilize our API in both C++ and Python [9-10]. They provide an easy way to use Aliot's API and access cloud services for end-users. It is worth mentioning that there is an alternative to WebSocket named MQTT [16] which is a standard messaging protocol in IoT. At the time we did not opt for this protocol, but we plan on implementing it for a future work. Nevertheless, we integrated the publish-subscribe mechanism in our own way with the WebSocket protocol.

C. Cloud and data processing

For the cloud and processing layer, we created flexible, fast and unique services that take complicated uses and concepts of IoT and make them easier and more accessible to researchers. For example, in one of our demonstrations, the *Connected Greenhouse* [14], we could visualize data coming from the sensors in real-time on ALIVEIoT by writing one simple function call in Python, as shown in **Alg.1** below.

Alg. 1. The greenhouse updates the NoSQL database with the sensors' data [1]

```
greenhouse.update_doc({
    '/document/humidity': greenhouse_state.humidity,
    '/document/temperature': greenhouse_state.temperature
})
```

The Python call is then translated to communicate more efficiently with our API. Afterwards, the IoT gateway validates permissions and decides what service to call in the cloud processing layer. The service can then perform the requested action. In this case, it updates the fields for the temperature and humidity in the private NoSQL database and automatically reflects the changes in the end-user GUI as shown in **Fig.5**.

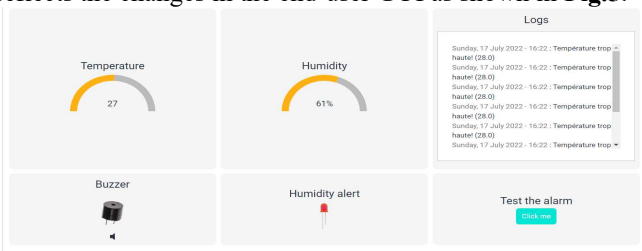


Fig.5. Connected Greenhouse customized GUI on ALIVEIoT

D. Iot analytics and data management

One of the most crucial parts of an IoT ecosystem is collecting and monitoring incoming data. When using the suitable tools, it is possible to prevent issues or detect irregularities during data collection. Aliot is equipped with tools that allow monitoring of the incoming data and creating triggers for them. These monitoring services can be used by both Aliot libraries or directly on ALIVEIoT without any IDE or software using AliveScript, a homemade language with IoT functionalities. For example, in the *Connected Greenhouse* [14], we automated the ecosystem by monitoring the temperature and humidity in the room to optimize the growth of crops. An alarm was programmed to be set off if the humidity or temperature was too high. This was achieved simply by using only a few lines of code in AliveScript as shown in **Alg.2**.

Alg. 2. Monitoring the Greenhouse humidity

```
1.  use Aliot
2.  notif "The script just started"
3.  function humidityChanges(key, val)
    3.1  notif  "Humidity changed: " + val + "%"
    3.2  if val < 10
                # Play frequency of 500Hz
                Aliot.updateComponent("buzzer", 500)
    3.3  end if
4.  end function
5.  Aliot.listenDoc("/document/humidity",
    humiditeChanges) # Start monitoring
```

E. End-user devices and user interface

As previously stated in Section III, it is on ALIVEIoT that an interface can be used to monitor sensors data in real-time and interact with the IoT devices. Aliot offers some special views for the end-user inside an ecosystem such as the customizable GUI, the document where users can view and edit the entries inside the NoSQL database, the permission for the IoT devices, and finally a page to manage all automation scripts written in AliveScript. The document view is shown below in **Fig.6**.

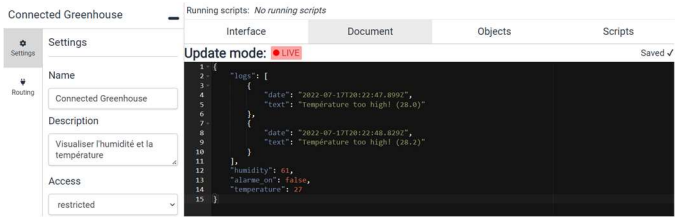


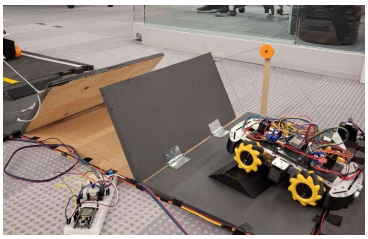
Fig.6. ALIVEIoT: Interface of an IoT ecosystem

V. RESULTS ANALYSIS

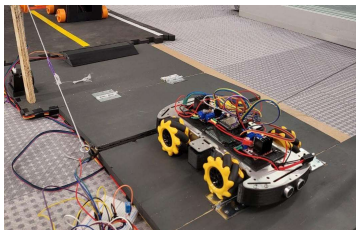
The following case studies were conducted by using Python 3.7 or higher for the Raspberry Pi IoT devices and C/C++ for the ESP32 IoT devices, along with some AliveScript available directly on ALIVEIoT.

A. First case study – *smart city*

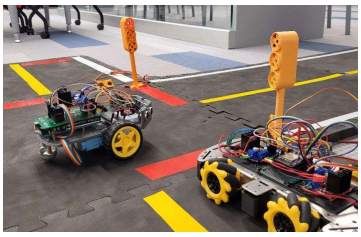
In the past, we conducted another project called the *Smart City* [14] using ALIVEIoT and Aliot. This project is the evidence of what Aliot is capable of. The goal of the *Smart City* was to replicate in a miniaturized and controlled city environment in order to find and test modern and innovative solutions to traffic. This city is composed of 5 IoT devices powered by Aliot: two cars, one using a Raspberry Pi 4 and the other one using an ESP32 microcontroller, connected streetlights (Raspberry Pi Zero), a connected bridge (esp32), and a connected parking (Raspberry Pi 4). We wrote a script in AliveScript that automated the itinerary of the car from point A to point B using many of Aliot's services. The following itinerary is shown in **Fig.7**. Additionally, a part of the script used is presented below the figure, alongside some in-depth explanations of every step of the itinerary. A video of the *Smart City* in action is also available for viewing [17].



a) The car starts near an open bridge

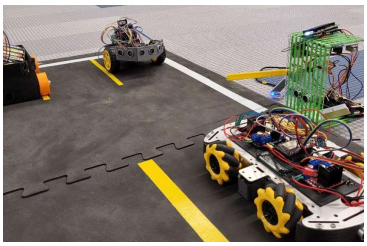


b) The car goes forward when the bridge closes

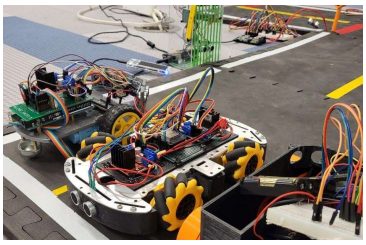


c) The car arrives at the intersection

d) The car turns left when the light becomes green



e) The car arrives at the parking and its license plate is detected



f) The car parks when the gate of the parking opens

Fig.7. Itinerary of the motorized vehicle in the *Smart City*

1. The car starts its course in front of an open bridge (**Fig.7a**).

2. The car is detected by the connected bridge and the bridge closes itself. The bridge then updates the database with its new closed state. The car which was listening for this state to change (**Alg.3a**) proceeds forward to the intersection (**Alg.3b** and **Fig.7b**).

3. The streetlight is red, so the car makes a stop (**Fig.7c**).

4. The streetlight detects the car using an ultrasonic distance sensor and the light automatically changes to green. The streetlight updates the database with its new state. The car which was listening for this state then makes a left if it's green (**Fig.7d**).

5. Once arrived in front of the connected parking, the license plate of the car is scanned and recognized using an AI algorithm (**Fig.7e**). If the license plate is owned by a person with valid access from the database, the barrier opens, and the name of the person is shown on the Liquid Crystal Display (LCD) screen. Once the barrier is fully opened, the parking updates the database with its new state. The car, which was listening for this state, parks itself (**Fig.7f**).

Alg. 3. The car is at the bridge [2]

a) The car listens for the change of the open state of the bridge

```
1. function bridgeOpensOrCloses(path, open)
   1.1 if open
           notif "The bridge just opened"
   1.2 else
           notif "The bridge just closed"
           Aliot.removeListener(bridgeOpensOrCloses)
           bridgeCloses()
   1.3 end if
2. end function
3. Aliot.listenDoc("/document/bridge/open", bridgeOpensOrCloses)
```

b) The bridge just closed

```
1. function bridgeCloses()
   1.1 Aliot.sendAction("forward", { "time": tile * 3 })
   1.2 Aliot.sendAction("left", { "deg": 90})
   1.3 Aliot.sendAction("forward", { "time": tile * 1.5 })
   1.4 Aliot.listenDoc("/document/lights/N/state", lightChanges)
2. end function
```

[2] This code is written in English, but it is possible to write the code in French with a French version of AliveScript.

B. Second case study – approach comparison

To further demonstrate the potential of Aliot, we developed a small ecosystem using two different approaches. The ecosystem had two ESP32 microcontrollers with one LED on each device and one photoresistor on one of them. The goal of the ecosystem was to have a real-time representation of the light level and a small button on a web page that could turn on both LEDs. We compared the traditional approach where you must build each layer of an IoT ecosystem, to our approach with Aliot. The traditional approach included a web server coded in Python hosted with Django on a dedicated server with manually set port forwarding from WAN, with an Asynchronous Server Gateway Interface (ASGI) using WebSockets to establish the connection from the C++ code of the ESP32 to the web server and ensure the communication between the two instances, and finally, a JavaScript code to update the website in real-time with another WebSocket connection. Preliminary results show the effectiveness of Aliot with a **reduction in lines of code written of more than 80%** and a **reduction of time spent developing of more than 10 times** Results are shown below in **Table 1**.

Criteria	Aliot's approach	Traditional approach
Time spent	0:22:04	3:50:12
# Of lines of code written	42	217
# Of programming languages needed	1	3
Complexity	*	****
Secured connection	X	-

Table 1. Traditional approach versus Aliot's approach. The number of * indicates in a range of 1 to 5 the degree of the criterion's appliance, while ' X' signifies the presence of the criterion and '–' means not present.

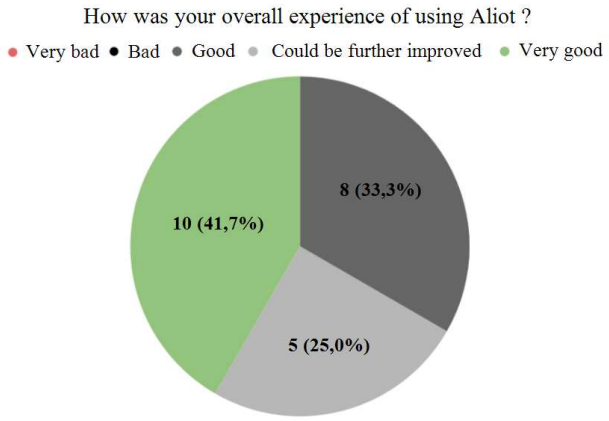
C. Third case study – college course

In Autumn 2021, Aliot was used as a teaching tool at Maisonneuve's College, Montréal, Canada, in a new IoT course composed of 28 students. Most of the students were completely new to IoT. They used Aliot to monitor the incoming data from many different IoT devices. As a final project, they were asked to program and build a connected parking similar to the one showcased in the *Smart City* project. They were also asked to build a connected musical keyboard that played musical notes in real time on ALIVEIoT.

After the course, the students were asked to fill out of form about their experience of using Aliot. 24 out of the 28 students

filled out the form and rated how easy-to-use Aliot was and their overall appreciation. At that time, the students used an earlier version of Aliot with fewer debugging tools, some remaining bugs and did not include most of the services available now, such as the automation, the database, and the real-time triggers and events. Nevertheless, the feedback received, illustrated in **Fig.8** below, was mostly positive.

a) Overall experience of using Aliot



b) How easy it was to integrate Aliot into existing projects.

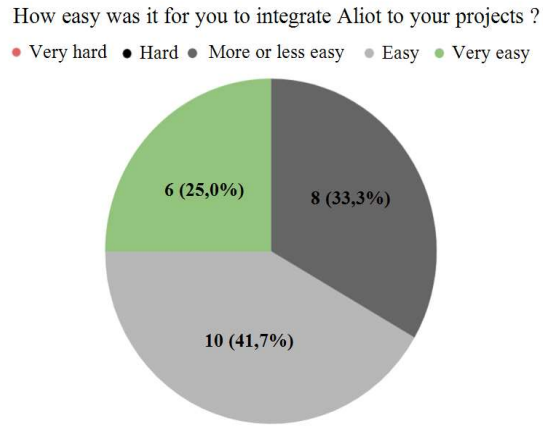


Fig. 8. Feedback received after using Aliot in the college IoT course

The feedback received confirm the effectiveness and especially the potential that our experimental set of tool has. As the table shows, the level of difficulty experienced by the students was relatively low. It is important to note that this was Aliot in its early stage of development, with the lack of major features, error handling, and reliability which have all been addressed in the newer versions of Aliot. Also, no students were dissatisfied with Aliot. We will soon be able to further confirm the effectiveness and flexibility of Aliot in an upcoming second iteration of the IoT course this Autumn 2022 with two classes resulting in a total of 56 students.

VI. CONCLUSION

To conclude, Aliot offers a variety of unique tools to help researchers develop in a faster and easier way their own IoT ecosystem based on their needs. For further improvements, we would like to widen our range of available services in the IoT analytics and data management layer to enhance the data collection and processing required in AI research. Moreover, we would like to integrate the MQTT protocol as an alternative to the WebSocket protocol we are using.

ACKNOWLEDGMENT

We would like to thank Mitacs and Énergie Scolaire for financially supporting this research. We would also like to thank all the other LRIMa's members who helped on ALIVEcode and the *Smart City* project.

REFERENCES

[1] IoT Analytics, State of IoT 2022: Number of connected IoT devices: https://iot-analytics.com/number-connected-iot-devices [last visited 29 august 2022].

[2] M. Aboubakar et al. A review of iot network management: Current status and perspectives Journal of King Saud University-Computer and Information Sciences (2021).

[3] Georgios Pierris, Dimosthenis Kothris, Evaggelos Spyrou, and Costas Spyropoulos. 2015. SYNAISTHISI: an enabling platform for the current internet of things ecosystem. In Proceedings of the 19th Panhellenic Conference on Informatics (PCI '15). Association for Computing Machinery, NY, USA, 438–444.

[4] Trilles, S. et al. (2020). An IoT Platform Based on Microservices and Serverless Paradigms for Smart Farming Purposes. Sensors (Basel, Switzerland), 20(8), 2418.

[5] Isakovic, H. et al. (2019). CPS/IoT Ecosystem: A Platform for Research and Education. In: Chamberlain, R., Taha, W., Törngren, M. (eds) Cyber Physical Systems. Model-Based Design. CyPhy WESE 2018 2018. Lecture Notes in Computer Science(), vol 11615. Springer, Cham.

[6] Jamin, A. et al. (2016). An Aggregation Plateform for IoT-Based Healthcare: Illustration for Bioimpedancemetry, Temperature and Fatigue Level Monitoring. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 187. Springer, Cham.

[7] thethings.io, 2018. The IoT platform to monitorize your devices. URL: https://thethings.io/ [last visited 29 August 2022].

[8] ALIVEcode open-source repository: https://github.com/ALIVEcode/ALIVEcode [last visited 26 August 2022].

[9] Aliot-c public repository: https://github.com/ALIVEcode/aliot-c [last visited 28 August 2022].

[10] Aliot-py public repository: https://github.com/ALIVEcode/aliot-py [last visited 28 August 2022].

[11] E. Soldevila et M. Laroche, M. L'abbee et J. Rezgui, "Aliot : un environnement numérique pour un apprentissage accessible et interactif de la programmation des objets connectés", Communications étudiantes, Congrès ACFAS 2022, CANADA.

[12] ALIVEcode: https://alivecode.ca [last visited 28 August 2022].

[13] J. Rezgui, F. Jobin, S. Beaulieu and Z. Ardekani, 'Autonomous Learning Intelligent Vehicles Engineering in a Programming Learning Application for Youth: ALIVE PLAY, accepted IEEE ISNCC 2021, Dubei.

[14] Aliot showcased projects: https://alivecode.ca/showcase/projects [last visited 29 August 2022].

[15] RFC6455, The Websocket Protocol: https://www.rfc-editor.org/rfc/rfc6455 [last visited 26 August 2022].

[16] Message Queuing Telemetry Transport (MQTT). http://mqtt.org/ [last visited 29 August 2022].

[17] The Smart City feature video: https://www.youtube.com/watch?v=a-wLMgqOz9E. [last visited 26 August 2022].