Multimodal Technologies and Interaction

MDPI

Article

Multimodal Interaction for Cobot Using MQTT

José Rouillard * and Jean-Marc Vannobel

CNRS, Centrale Lille, University of Lille, UMR 9189 CRIStAL, F-59000 Lille, France
* Correspondence: jose.rouillard@univ-lille.fr

Abstract: For greater efficiency, human–machine and human–robot interactions must be designed with the idea of multimodality in mind. To allow the use of several interaction modalities, such as the use of voice, touch, gaze tracking, on several different devices (computer, smartphone, tablets, etc.) and to integrate possible connected objects, it is necessary to have an effective and secure means of communication between the different parts of the system. This is even more important with the use of a collaborative robot (cobot) sharing the same space and very close to the human during their tasks. This study present research work in the field of multimodal interaction for a cobot using the MQTT protocol, in virtual (Webots) and real worlds (ESP microcontrollers, Arduino, IOT2040). We show how MQTT can be used efficiently, with a common publish/subscribe mechanism for several entities of the system, in order to interact with connected objects (like LEDs and conveyor belts), robotic arms (like the Ned Niryo), or mobile robots. We compare the use of MQTT with that of the Firebase Realtime Database used in several of our previous research works. We show how a "pick–wait–choose–and place" task can be carried out jointly by a cobot and a human, and what this implies in terms of communication and ergonomic rules, via health or industrial concerns (people with disabilities, and teleoperation).

Keywords: multimodality; human–robot interaction; MQTT protocol; cobot; open-source; industrial IoT; virtual worlds; assistive technologies

Citation: Rouillard, J.; Vannobel, J.-M. Multimodal Interaction for Cobot Using MQTT. *Multimodal Technol. Interact.* **2023**, *7*, 78. https://doi.org/10.3390/mti7080078

Academic Editor: Cristina Portalés Ricart

Received: 29 June 2023
Revised: 19 July 2023
Accepted: 24 July 2023
Published: 3 August 2023

1. Introduction

In the domain of multimodality, health informatics, biomedical engineering, and human interaction at large, we have to design and implement multimodal solutions adaptable to various users (patients, carers, health professionals) and contexts. Since Richard Bolt and his famous "Put that there" [1], it has been well known that multimodal interaction can provide more natural and easy-to-use interfaces. In our research, we focus on the use of multimodal interfaces to interact with a connected environment, especially for users with special needs. But, designing and developing such applications for people with specific needs can be difficult, time consuming and expensive, for multiple reasons:

- The particular needs, according to cognitive and/or physical disabilities of the users;
- The calibration and sharp synchronization of sensors and effectors needed;
- The requirement to test various input and output solutions;
- The health situation and mental state changing from one session to another.

In previous works, we have introduced a Multimodal Interaction Framework based on Firebase Realtime Database [2] and a Biomedical Framework for Enhanced Experimentation—BIOFEE [3]. The interest of such intelligent systems built on a multimodal basis lies in the fact that a decision is made through several independent information channels, with the subsequent aggregation of these decisions [4].

Worldwide, the number of people with significant disabilities is approximately 1.3 billion, or 16% of the world's population. Among them, one in five people has a so-called severe disability [5]. Thanks to years of research, today, assistive technologies can compensate the user disabilities regarding physical, functional, cognitive or mental disabilities. But,

unfortunately, they are often expensive, designed for a specific disability (for paraplegic people, blind people, deaf people, autistic people, elderly people...) and must be configured according to each patient. However, assistive technologies offering multimodal interfaces can accommodate a greater number of users than those offering traditional interfaces [6]. A user deficient in one sense (voice, touch, etc.) can then compensate for his handicap by alternative methods, without limiting the functionality of the system. The simultaneous availability of several communication channels makes it possible to choose the most practical, fast and intuitive channel according to the disability, the state of health and the context.

From an engineering and developer point of view, those complex multimodal systems have to be as flexible as possible. In software engineering, the terms frontend and backend often refer to the separation of concerns between the presentation layer (in front of the user), and the data access layer (backend) of a piece of software, or the physical infrastructure or hardware. Nowadays, with the needs of communication for smartphones and other connected objects (see IoT: Internet of Things), the notion of BaaS (Backend as a Service) is emerging. BaaS is a service for providing web and mobile applications with a way to easily build a backend solution. Features available include user management, scalable databases, cloud storage, push notifications, authentication, and integration with social networking services, thanks to custom software development kits (SDKs) and application programming interfaces (APIs).

We have shown in previous work how to conceive, deploy and test such multimodal and multichannel applications where push notifications are mainly used [7,8]. This also means that multimodal applications based on this technology must be constantly connected to the Internet, with a good quality network and bandwidth. This is not always possible in medical or industrial environments, for example (see the case of limited, insecure, low-capacity network). For instance, by choosing the Firebase database, developers may face problems in terms of cost (the free plan is only sufficient for small applications), hosting (Google Cloud platform) and database (NoSQL).

In this paper, we present our ongoing work on the notion of multimodal interaction in the context of human–computer Interaction to improve the quality of life. Some recent papers have been exploring the use of multimodality, with mobile robots or the Brain–Computer Interface (BCI), for instance, by addressing the MQTT (Message Queuing Telemetry Transport) protocol [9,10]. In the following, we present more detailed research works that we have carried out in the field of multimodal interaction for a cobot using the MQTT protocol. A cobot, or collaborative robot, is a robot intended for direct human–robot interaction within a shared space, or where humans and robots are in close proximity. Cobot applications contrast with traditional industrial robot applications in which robots are isolated from human contact.

The article is organized in the following way: Section 2 presents multimodal communication needs, with a focus on Firebase and MQTT tools. Section 3 describes case studies using MQTT in virtual and real worlds, and Section 4 investigates the usage of MQTT coupled to Webots in the context of industrial and multimodal cobots, before the Discussion, Conclusion and Perspectives section.

2. Multimodal Communication Needs

Our present study focuses on two means of communication (firebase and mqtt), increasingly used, in various fields (education, industry, health, home automation, robotics, transport...) allowing different modalities of interaction with the user, which can be exploited according to the difficulty and cost of implementation, communication capabilities, diversity of handicaps encountered, etc. This approach, particularly implemented during interaction with robots, allows the setting up of remote systems or even teleoperation, for the most sensitive cases (physical danger for humans, hostile environment...).

2.1. Firebase Realtime Database

We proposed in a previous work the study of BIOFEE (Biomedical Framework for Enhanced Experimentation), based on the Firebase Realtime Database, in order to design web and mobile applications for patients suffering from progressive diseases. It allows interaction with objects and robots to test various multimodal solutions, such as touch, voice, gesture, eyes gaze, etc., to perform tasks according to the patient's pathology.

Figure 1 describes the architecture of our multimodal BIOFEE framework. The user's devices (computers, smartphones...) are connected to a Firebase Realtime Database. This allows each component of the system to be notified instantly when a modification occurs in the database. The various input signals are then treated by the multimodal engine to synthesize one user command. Then, the system reacts accordingly. A Wizard of Oz module [11] is also available, in order to let a human (the wizard) inject commands when the machines are not able to do it by themselves (a complex or ambiguous multimodal request, for example).

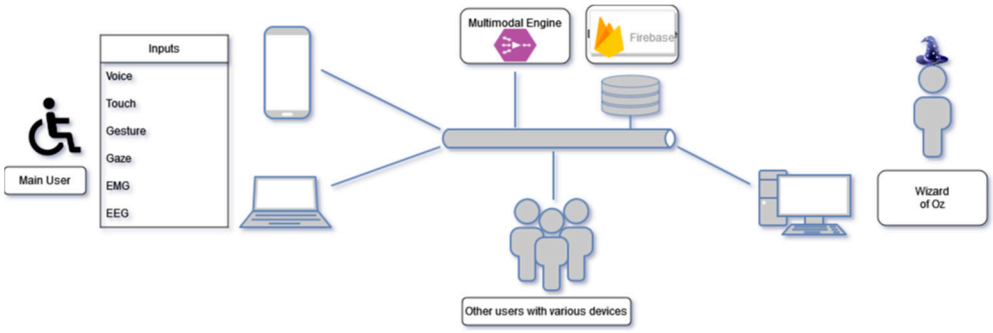
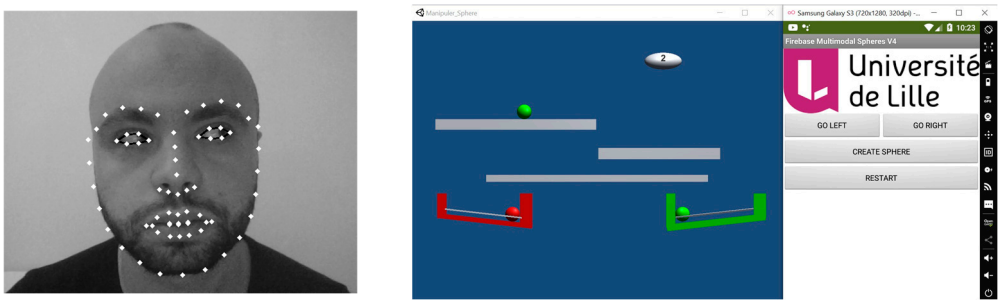


Figure 1. Architecture of our multimodal BIOFEE framework based on Firebase Realtime Database.

Most of the time, the used sensors are available in a smartphone (touch screen, voice recognition, accelerometer, light detection, etc.), but it can also involve third-party applications connected to the Firebase database. For instance, Figure 2 shows an example of eye gaze detection to perform a command in a Unity3D application [12]. The user can choose various ways to interact with the system: the keyboard or mouse on a classic computer, voice or buttons on a mobile device, tilting the smartphone one way or the other, in a natural way, to move the ball on the PC screen, or a combination of those modalities, sequentially.



(**a**) (**b**)

Figure 2. (**a**) Eye gaze detection (left direction); (**b**) performing a ball movement in a Unity 3D application.

Although functional, this approach is not trivial, and requires multiple steps to be implemented. Developers have to create a project in Firebase, to define the access rights for the database, and add an "application" to the project (here, the "Unity" option was chosen from among iOS, Android, Unity, Web, and Flutter). Then, it is necessary to download the configuration file obtained, as well as the Firebase SDK, to add into the Unity project. Then,

the developers have to code, in C# for example, the necessary functions to write an element in the NoSQL database and to be notified when an element is updated by others.

As stated previously, some researchers and developers prefer non-vendor lock-in solutions (as Firebase is held by Google), and choose open-source projects, to have a better control over the entire application development. Under these conditions, MQTT is a serious alternative that deserves to be studied.

2.2. MQTT

Message Queuing Telemetry Transport, better known as MQTT, is a lightweight and reliable machine-to-machine (M2M) messaging protocol. Being an open OASIS standard [13] and an ISO recommendation (ISO/IEC 20922), MQTT is also declared by the MQTT organization [14] to be "The standard for IoT Messaging". This reputation has not been usurped. MQTT is widely chosen for IoT because the protocol is very easy to implement. MQTT can be used locally or remotely, and simultaneously on different platforms, while consuming minimal bandwidth.

MQTT's success is first due to its ease of installation, which does not require implementing client/server architecture, as such. Based on social network-type communication (subscription to a discussion topic or publication), MQTT facilitates the communication between microprocessor-based machines, making each independent and autonomous. As illustrated in Figure 3, each machine carries out its work according to the information it needs to do so and receives in real time, while also being able to inform the other participants in the system.

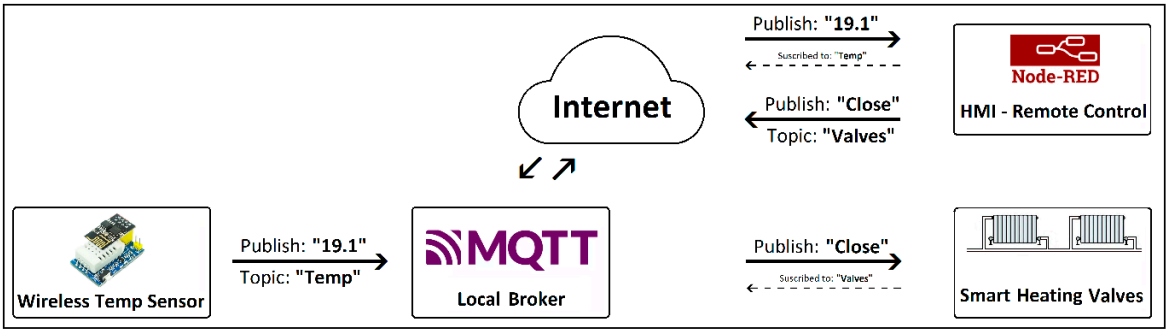


Figure 3. Typical MQTT communication between sensors and actuators through an HMI.

MQTT's success is also due to its large community of developers and users. MQTT brokers, the entities that enable MQTT clients to communicate, as well as many clients, can be found for any type of operating systems, and over dedicated clouds. The same can be said for software stacks and libraries that exist for any development environment.

MQTT reaches a wide audience interested in DIY (do it yourself) applications and domotics (home automation), but the protocol is due to IBM, which used it first to monitor oil pipeline sensors linked to the existing Supervisory Control And Data Acquisition (SCADA) system [15]. That said, a MQTT broker (or server) cannot be directly compared to the OPC-UA data servers commonly used for industrial data sharing. Although both are secured communication protocols, the main difference is that OPC UA induces a semantic data representation model [16] and encourages system interoperability [17], while MQTT mostly shares strings.

3. Case Studies Using MQTT

Below is a classic "switch on/off" example, demonstrable both in the virtual and real worlds. The virtual world is implemented with the Webots environment tool [18]. The same messages can also be used to communicate with physical sensors and effectors (such as push buttons and LEDs), and virtual and real worlds can obviously be mixed in advanced and multimodal scenarios.

3.1. Webots and MQTT

Webots is an open-source robotics simulator. It is used in industry, in research, and in education. The Webots project began in 1996, originally developed by Dr. Olivier Michel at the Ecole Polytechnique Fédérale de Lausanne, Switzerland [19].

Figure 4 shows two states (on and off) of a floor light available in Webots. The «pointLightIntensity» value of a floor light can be changed programmatically. The scripts (controller, supervisor) can be written in C, C++, Java, Python and Matlab. We used Python, which provides an easy way to integrate the MQTT protocol, in order to allow our virtual world to be manipulated both from inside (by other scripts) and outside Webots.

(**a**) (**b**)

Figure 4. A Webots virtual world presenting a floor light that can be remotely controlled by MQTT messages. (**a**) light off; (**b**) light on.

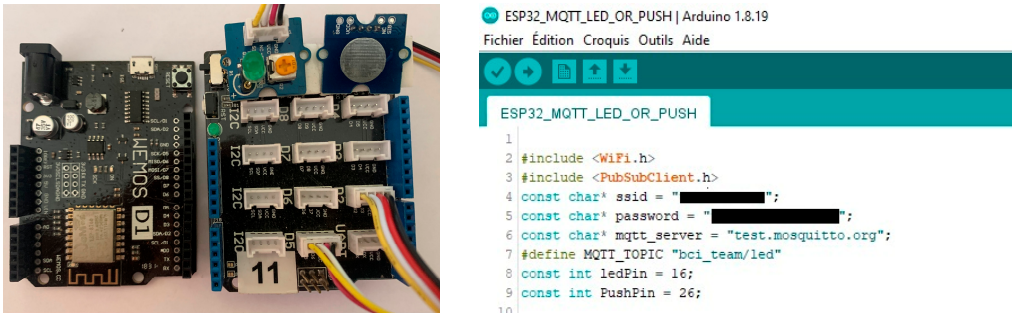
Thus, thanks to a publish/subscribe mechanism, this world can be driven by third-party applications. For instance, in a Windows 11 operating system, on which Mosquitto has been installed, the DOS commands below allow, respectively, to publish an «ON», and publish an «OFF» message, and subscribe to a topic named «bci_team/led», via the broker «test.mosquitto.org»:

```
mosquitto_pub -h test.mosquitto.org -t bci_team/led -m “ON”
mosquitto_pub -h test.mosquitto.org -t bci_team/led -m “OFF”
mosquitto_sub -h test.mosquitto.org -t bci_team/led
```

This software solution is interesting for testing the projects, locally and remotely. But sometimes it is also necessary to provide users with various real effectors (push buttons, sliders, potentiometer, etc.). The next section discusses this topic.

3.2. MQTT, ESP8266 or ESP32 and Arduino

It is very easy to perform the “switch on/off” task using ESP8266 and ESP32 ESPRESSIF chips [20]. These powerful Wi-Fi communication chips are very practical when soldered onto Arduino boards, as we have done (WEMOS D1 R2—ESP8266 and WEMOS D1 R32—WROOM32). They can be programmed directly using the Arduino IDE, and are compatible with Arduino daughter boards, especially the D1 R32, which has more pins than the D1 R2. Figure 5 (left) shows the WEMOS D1 R2 board and the grove daughter board with LED and push button, and (right) shows, in the case of the use of the WEMOS D1 R32 board, how simple it is to declare the Wi-Fi access point, the MQTT parameters, the pins used and the inclusion of the MQTT PubSubClient library. Subscribing to a topic, and publishing or decoding the received strings is also really simple, and well-documented on the internet [21,22].



(**a**) (**b**)

Figure 5. (**a**) ESP board; (**b**) MQTT Arduino-based programming.

3.3. MQTT, IOT2040 and Arduino

The IOT2040 device is a gateway provided by SIEMENS, widely used in the field of IoT and IIoT (Industrial Internet of Things). This equipment can be used autonomously, as is the case here, or in an industrial context, as we will see below. As illustrated in Figure 6, using the IOT2040's on-board Arduino board is as easy as using a UNO board [23]. It is just necessary to install the Intel Galileo package using the Arduino board's manager, as illustrated in Figure 7. The only difference in programming with the ESP32 boards concerns the pins declaration that correspond to the UNO ones and the use of the Ethernet library instead of the Wi-Fi one. Finally, MQTT communication is carried out in the same way as with the ESP32 board, using the PubSubClient library.

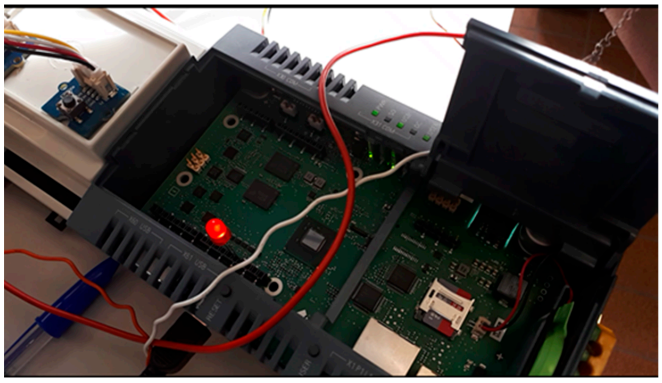
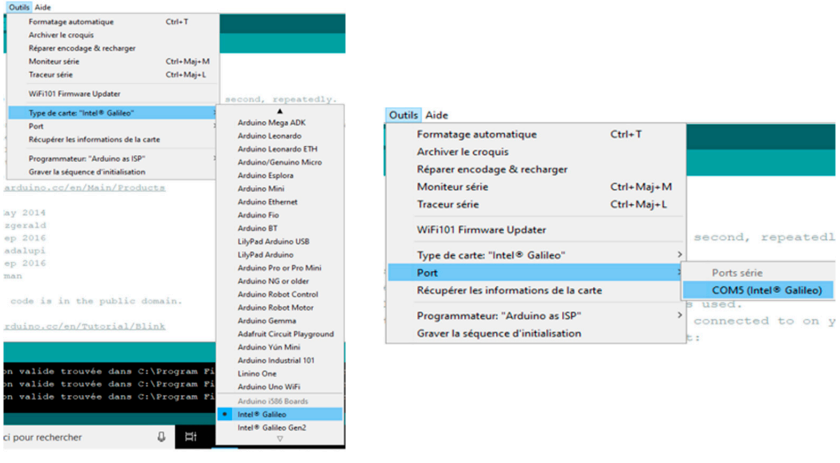


Figure 6. IOT2040 on-board Arduino connector with LED driven using MQTT.



(**a**) (**b**)

Figure 7. IOT2040 MQTT Arduino-based programming. (**a**) Type of card; (**b**) Port COM.

3.4. MQTT: IOT2040 Industrial Approach Case Study

An industrial approach case study is illustrated in Figure 8. This describes a robot cooperation task developed within the INCASE Interreg project [24]. The heart of our system is based on a SIEMENS IOT2040 that can be directly connected to an industrial PROFINET network, making it easy to communicate with an Industrial Human–Machine Interface (HMI) as the KTP600 used and an S7 1200 SIEMENS PLC (Programmable Logic Controller) (CPU 1212C). The IOT2040 runs a Yocto Linux operating system, and gives access to convenient software (MQTT broker, Node-Red WEB HMI design, Arduino programming compatibility...) as well as Internet connectivity [25]. The purpose of this case study was to show that the robots and the conveyor could work together to accomplish a task asked by an end user consisting of:

1. Cube color selection by the end user via KTP600. MQTT message published on a topic to which robot #1 (on the right in Figure 8) and #2 (on the left in Figure 8) have subscribed;
2. Robot #1 (on the right) takes a colored cube in the entry warehouse. It is supposed to be the right one;
3. Robot #1 places the cube on the conveyor with the MQTT message published on a topic to which robot #2 and the conveyor have subscribed;
4. The conveyor belt starts up and continues until the presence sensor detects the cube. The conveyor sends an MQTT message to robot #2;
5. Robot #2 takes the cube and checks its color, using a sensor located in the middle, between robots #1 and #2. If it corresponds to the expected one, the cube is stored in the output warehouse and a message is published for the attention of the HMI, to let it know that the task is achieved; if not, a message is published for robot #1, asking it to evacuate the cube before placing another one on the conveyor.

Figure 8. Industrial case study. Robot cooperation by MQTT messaging.

An interesting programming possibility given by the IOT2040 is illustrated in Figure 8. The demonstrator uses two DOBOT Magician robots [26]. These robots must be initially calibrated for the experiment to work, and the coordinates of the positions to reach in the input and output warehouses need to be recorded in a database or calculated from a reference point. This can quickly become a constraint if you proceed in a supervised way, using the S7 PLC. For example, we need HMI in the KTP600 other than the one intended for the end user, or a joystick to be connected to the IOT2040, knowing that the PLC can communicate with this device or with the robots using MQTT.

We can also use the Node-Red flow programming tool already installed in our IOT2040. This is illustrated in Figure 9. This is an interesting solution from three points of view. First and foremost, it requires no programming skills for the PLC, and therefore for the TIA Portal. Secondly, the Node-Red designed HMI can be accessed from a simple browser on the local network, or from anywhere in the world if the IOT2040 is connected to the Internet, of course securely. Finally, the Node is an open-source solution used in the industry within the context of IIoT and Industry 4.0 for instance data acquisition, storage tasks and monitoring [27], design and implementation of a SCADA architecture [28], or as a useful software tool to store IoT device data onto a blockchain [29].

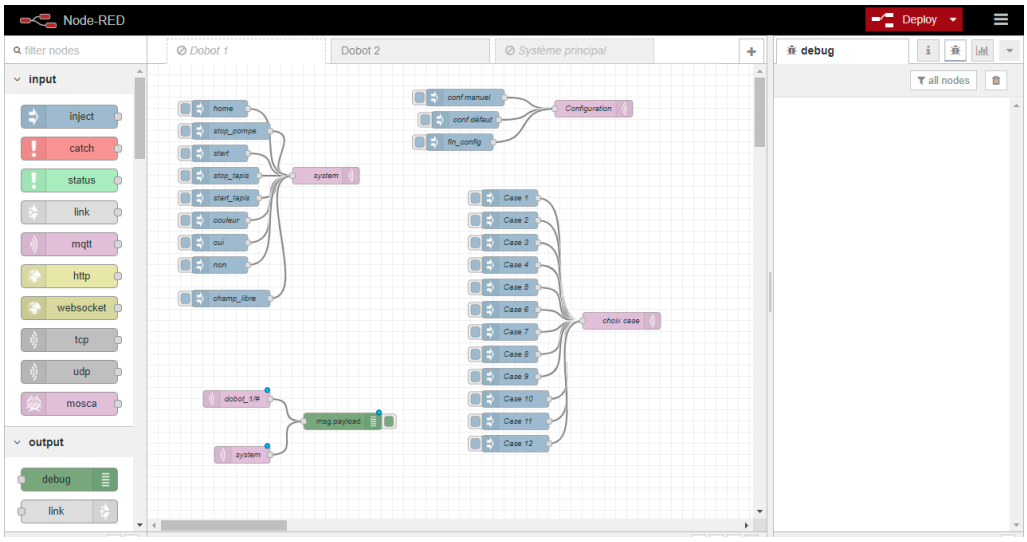


Figure 9. Node-red interaction with robot #1.

Communication with the DODOT robots becomes then very simple, knowing that we operate using the MQTT messaging protocol for the process control.

This solution has been used several times without any malfunctions being observed when restarting. However, we cannot guarantee the reliability of the solution in the long term. That said, as these hardware and software tools are available on an industrial platform, we can rely on SIEMENS' reputation in considering long-term deployment.

4. Webots and MQTT for Cobot Solutions

We have seen that MQTT is powerful for machine-to-machine communication. But it is also interesting for human–machine interaction. Indeed, one particularly interesting thing that the use of the MQTT protocol allows is the chaining of tasks, following notifications that can come from multiple sources. For example, imagine a scenario in which a robotic arm interacts with a human in an industrial context. The robot's task is to send a product (e.g., a packet of biscuits) onto one of the two conveyor belts available, to its right or to its left, as illustrated in Figure 10.

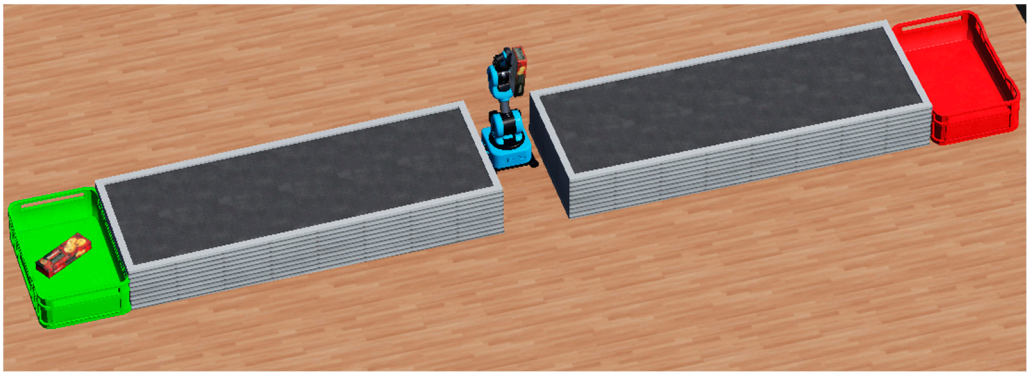


Figure 10. In Webots, the cobot Ned Niryo picks up a packet of biscuits and waits for a human decision to perform the next task (move left or right).

The user is in charge of the final decision, and has to collaborate with the robotic arm, by sending it information. This can be done in various ways, according to the context and profile of the user. The cobot (here a Ned Niryo, provided by Webots) can perform a «Pick and Wait» movement when it receives a specific MQTT message (See «PICK_WAIT» in the Table 1).

Table 1. MQTT topics and messages available to perform actions.

Topic	Message	Action
bci_team/box	SPAWN	Generate a new instance of a packet of biscuits in front of the cobot.
bci_team/box	PICK_WAIT	Perform various joint movements of the cobot to pick up a packet of biscuits, and raise it.
bci_team/arm	LEFT	Perform a left rotation of the cobot, and open the gripper.
bci_team/arm	RIGHT	Perform a right rotation of the cobot, and open the gripper.

So, after picking up a packet of biscuits, the cobot waits for a human decision. It can be achieved by a voice command, a touch on a dedicated button in a graphical interface, an eye-gaze, and so on. When a message is sent to a MQTT subject to which the cobot has subscribed, this triggers the execution of a function (see callback mechanism). In our case, a movement of the robotic arm to the right or to the left is activated. Then the opening of the robot's gripper releases the package of biscuits, which is then transited on one of the two belts, as illustrated on Figure 11.

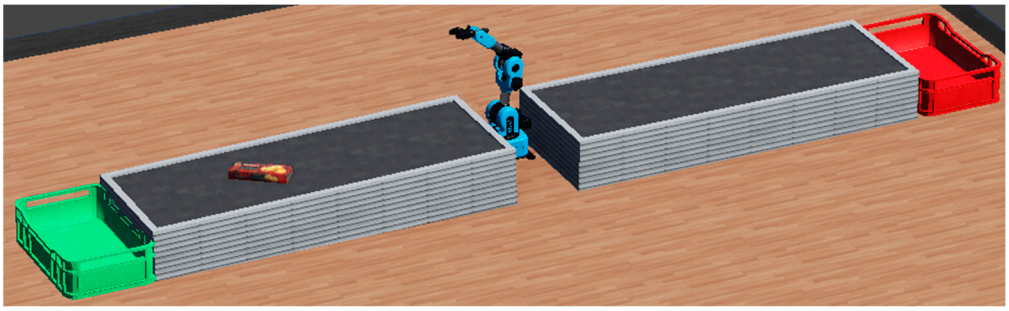


Figure 11. The cobot received the MQTT message «RIGHT» for the topic «bci_team/arm». Accordingly, the packet is send to the conveyor belt situated on its right.

The developer can decide to code the published and subscribed functions in order to chain the tasks, with more or less automation, and more or less exchange with humans, depending on the level of complexity or security required. For example, when the Webots supervisor receives a "SPAWN" message to generate a new packet of biscuits, it is possible,

following this, to request the cobot to execute a "PICK_WAIT" task, triggered (published) by the supervisor. The Niryo controller will receive this message (thanks to a MQTT subscribe mechanism), and will move the various joints of the robot in order to perform this task. Another message, coming directly from the user will command a left or right move.

As illustrated in Figure 12, we used the Ned from the Niryo robot provided by Webots to conduct tests on MQTT communication for a multimodal purpose. MQTT, which is well known for machine–machine communication, is here also used for communication between the human operator and the cobot that can help him/her to achieve a desired task.

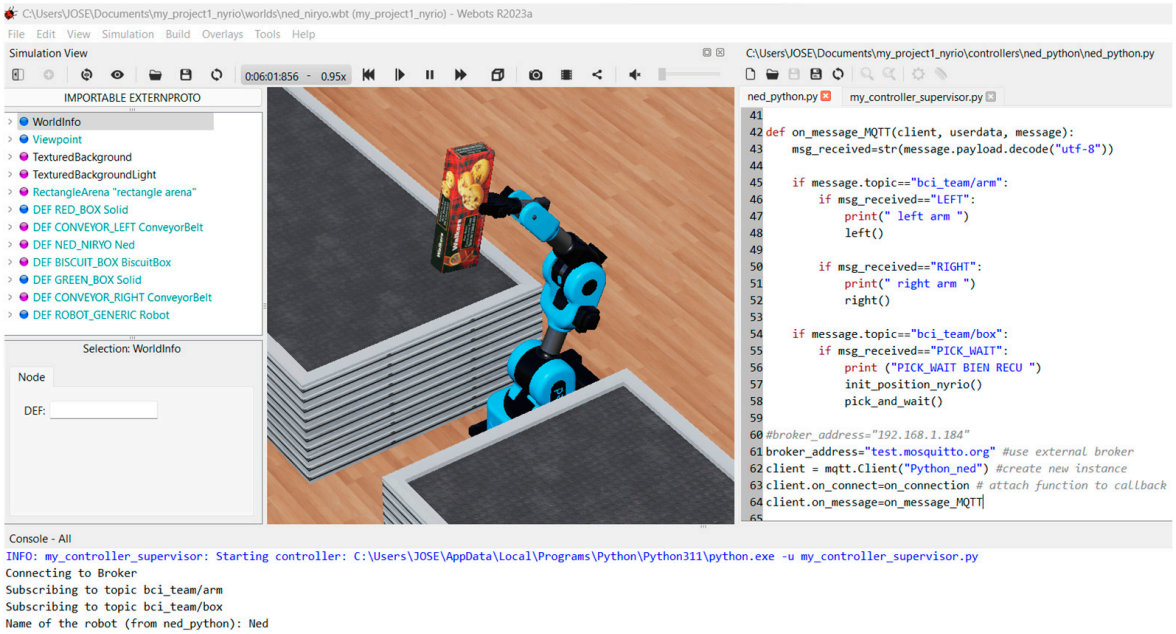


Figure 12. A close capture of the Ned Niryo robot in Webots, holding a packet of biscuits and waiting for human instructions. On the right part of the screen, the editable Python scripts are used to program interaction with other robots or humans (MQTT).

Specifically, a robot controller coded in Python is used to drive the Ned robot. A supervisor, also coded in Python, is able to generate elements in the virtual world (see left in Figure 12). In our case, the supervisor generates a packet of biscuits to respond to a «SPAWN» MQTT message. These two scripts are communicating across MQTT messages. Thus, a Wizard of Oz (human agent) (see Figure 1) can inject specific messages into the system, to unblock a situation or intervene in place of a human or robot agent that is momentarily failing (because of fatigue, a network problem, misunderstanding of the command, the required level of security not being authorized, etc.).

Figure 13 represents a snippet of our Python program (ned_python.py) in charge of the Ned robot movements, according to the MQTT messages received.

The next part of the article describes how a mobile application allows the use of various interaction modalities (voice, touch, accelerometer, light sensor, etc.) very easily, via App Inventor connected to MQTT.

```
import paho.mqtt.client as mqtt
import time
def on_connection(client, userdata, flags, rc):
if rc==0:
print ("Connected : OK")
client.publish("bci_team","MSG MQTT FROM WEBOTS: NED READY")
else:
print ("Bad Connection. Returned code :",rc)
def on_message_MQTT(client, userdata, message):
msg_received=str(message.payload.decode("utf-8"))
if message.topic=="bci_team/arm":
if msg_received=="LEFT":
print(" left arm ")
left()
if msg_received=="RIGHT":
print(" right arm ")
right()
```

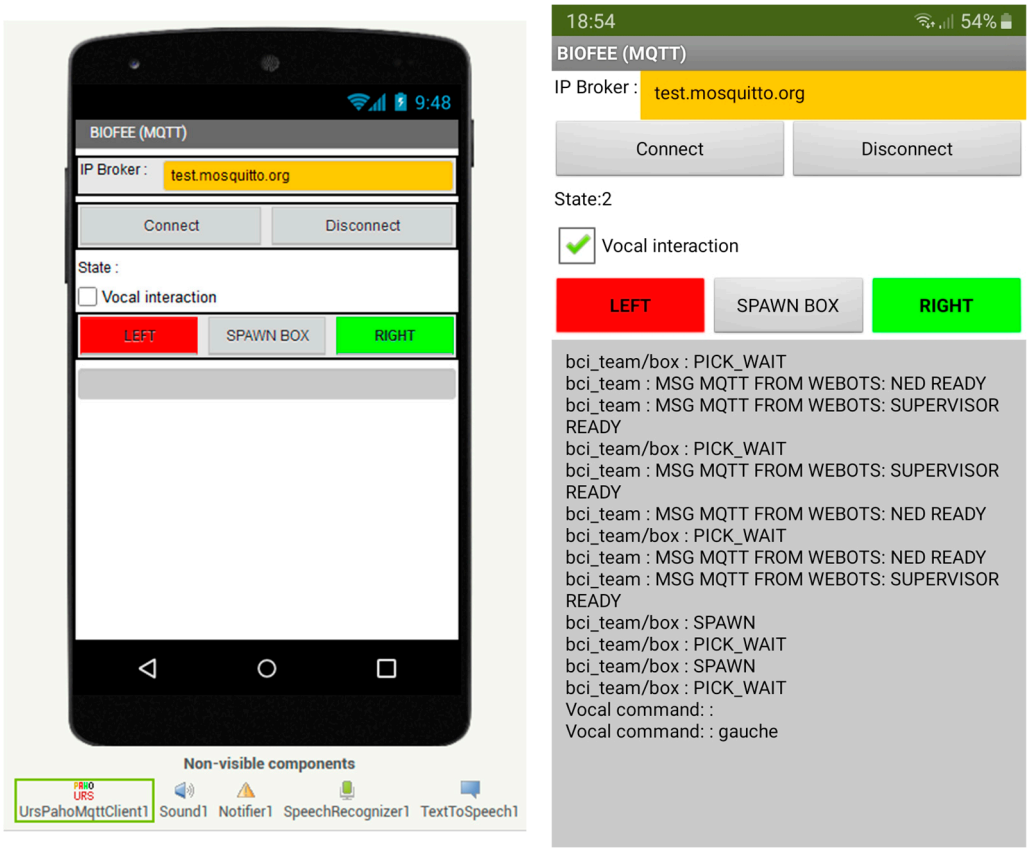
Figure 13. Snippet of the Python program (ned_python.py) in charge of the robot movements, according to the MQTT messages received.

4.1. Multimodality with App Inventor and MQTT

App Inventor for Android is an application development software created by Google, currently maintained by the Massachusetts Institute of Technology [30]. We successfully used App Inventor and an experimental component named "Firebase DB", in previous work [2] in order to create quickly and easily some mobile applications to deploy and test multimodal prototypes.

For comparison, within the framework of our current work on MQTT, we used an extension [31] allowing the easy integration of this protocol, within a mobile application developed for Android. Figure 14 shows the App Inventor designer using this MQTT extension and the Android APK executable version obtained in order to interact with the Webots Ned cobot. The «LEFT» and «RIGHT» buttons are colored the same way as the colors of the baskets positioned at the end of the two conveyor belts in the Webots virtual world. Moreover, if necessary, a vocal checkbox allows (or not) the use of a speech recognizer and a text-to-speech component, which do not require physical (touch or gesture) interaction with the mobile device.

As explained with our App Inventor blocks of Figure 15, when a «PICK_WAIT» message is received for the «bci_team/box» topic, if the checkbox is checked, then a speech synthesis is triggered, inviting the user to choose a side (left or right), and speech recognition is launched to obtain a vocal response.



(**a**) (**b**)

Figure 14. App Inventor mobile application connected to MQTT to interact with the Webots Ned cobot. (**a**) Designer view; (**b**) Android APK executable version.

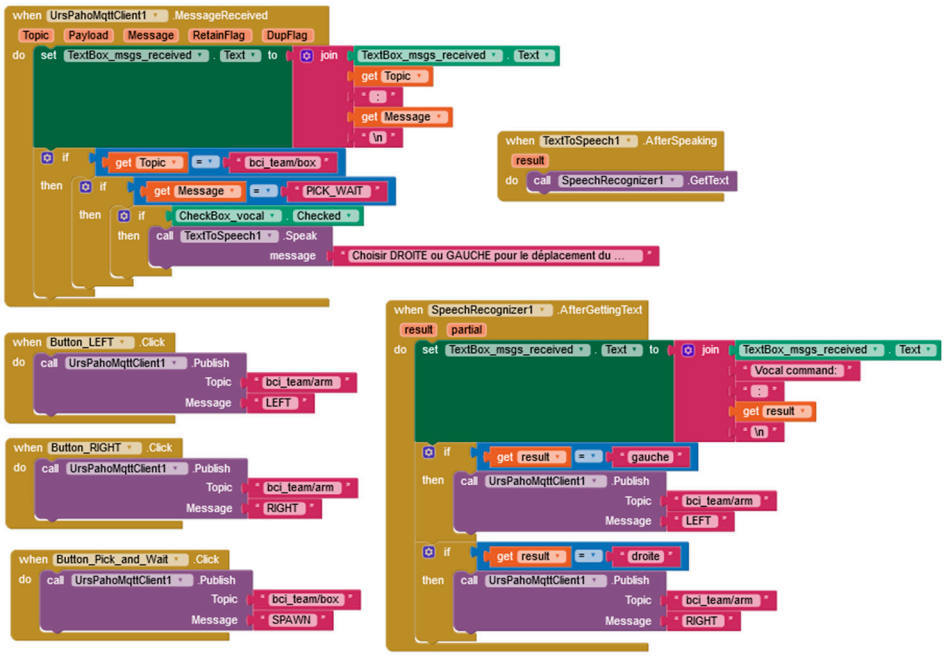


Figure 15. App Inventor blocks snippet used to interact (touch or voice) with the Webots Ned cobot, thanks to MQTT.

4.2. Leap Motion, EMG and EEG

Now, let us imagine that, unfortunately, this user is no more able to click on a smartphone using a finger. Which other modalities could be used in this specific context? We present briefly three other possibilities: a leap motion, an EMG and an EEG. We would like to test if a residual muscular activity is detectable with an optical hand-tracking module, such as a LeapMotion [32] for instance. A Leap Motion Controller (LMC) is used when it becomes difficult for a person to press a push button, or for hygiene issues. The LMC is a device that uses infrared cameras and hand-tracking technology to allow users to interact with computers and other devices through gestures made with their hands. The controller is often used in virtual reality (VR) and augmented reality (AR) applications, but it can also be used with other types of software, especially with disabled people. Although different studies and projects have been developed to use hand-tracking technology with disabled people, such as a hand rehabilitation system to assist in developing muscle tone and increase precision in gestures [33,34] for sign languages recognition [35], and in music therapy sessions [36], it is not a common or a widely used technology.

Some limitations of the device include:

- A limited tracking range (10 to 70 cm) and a restricted angular view (120 to 140°) [32];
- A possibility of bad tracking and lack of accuracy when hands or fingers are obscured by other objects. It then needs a clear line of sight [37,38];
- Aa limited number of supported gestures. It may not be able to recognize complex hand and finger movements like those used in sign languages [39];
- An interior use. The controller may not work properly in bright sunlight or other harsh lighting conditions [40];
- A use limited to desktop computers. The LMC is not compatible with mobile devices [32];
- A high cost comparative to a mouse, a keyboard or a joystick. It may not be accessible to all users.

On the other hand, benefits include:

- A high accuracy, especially for hand palms, thumb and index finger tips, making the device well suited for use with individuals with limited movement abilities;
- A natural user interface, because the users can interact with their computers using hand and finger gestures, as in the physical world;
- A portable use, since the controller is small, lightweight and easy to set up;
- Compatibility with a wide range of operating systems and programming languages, making it easy to integrate into existing software systems or as a pointing device [41];
- A large developer community, especially for VR and AR applications;
- An alternative to traditional input devices such as mouse, keyboard and joystick, which can be beneficial for users with certain types of physical or cognitive impairments.

In our case, the LMC is used as a complement or replacement for touch screens.

The JavaScript snippet presented in Figure 16 is used to analyze the data (numbers and positions of the hands, fingers, extension moves, rotations, etc.) provided by a Leap Motion device connected to the USB port of the local user machine. The developer has selected what kind of gesture interaction will trigger a MQTT message. For example, a slight movement of the index finger of each hand can be the trigger for a message published on the «bci_team/arm» (the robot arm) topic, with the “right” or “left” value indicating the choice of the user, and therefore the side to which the cobot will rotate before releasing the gripper.

```
<script src="http://js.leapmotion.com/leap-0.6.3.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.min.js"
type="text/javascript"></script>
var MQTT_client;
var MQTT_reconnectTimeout = 2000;
var MQTT_host = "test.mosquitto.org";
var MQTT_port = 8080;
var MQTT_userID ="CRISTAL_LAB";
###(...)
  if (pointable.extended != previous_pointable_extended_right)
            {
                if (pointable.extended)
                { MQTT_message = new Paho.MQTT.Message("NO RIGHT"); }
                else
                { MQTT_message = new Paho.MQTT.Message("RIGHT");}
                MQTT_message.destinationName = "bci_team/arm";
                MQTT_client.send(MQTT_message);
                previous_pointable_extended_right=pointable.extended;
            }
```

Figure 16. Snippet of the JavaScript program in charge of publishing MQTT messages when the leap motion is used.

In order to optimize the flow of data transmitted on the network via the MQTT protocol, we only publish new data for the Ned robot simulated in Webots if they are different from the previous state (true or false) recorded by the leap motion (extended position of the index finger) during successive frames which follow one another at a very high speed.

During our tests with users handling the leap motion, we realized that an ambiguity is possible, for some people, concerning the choice to be made (right or left), depending on the angle of view proposed by the Webots simulator.

An ergonomic way to reduce this ambiguity and cognitive overload is to rotate the view, so that the user finds himself in the place of the cobot. The right-hand side of the user will be the right-hand side of the robot, and vice versa, as illustrated in Figure 17.

In previous works [42,43], we have studied other interaction modalities, particularly regarding electromyography (EMG) and electroencephalography (EEG). Figure 18 shows signals used in our hybrid system: right joystick values, EMG from the right hand and EEG from the C3 electrode. They were recorded during a right-hand movement which lasted 5 s. This movement is well detected, thanks to joystick values and the EMG signal. Indeed, EMG amplitude is greater during muscular contraction. Nevertheless, the EEG signal needs more processing in order to highlight relevant information and detect a hand movement.

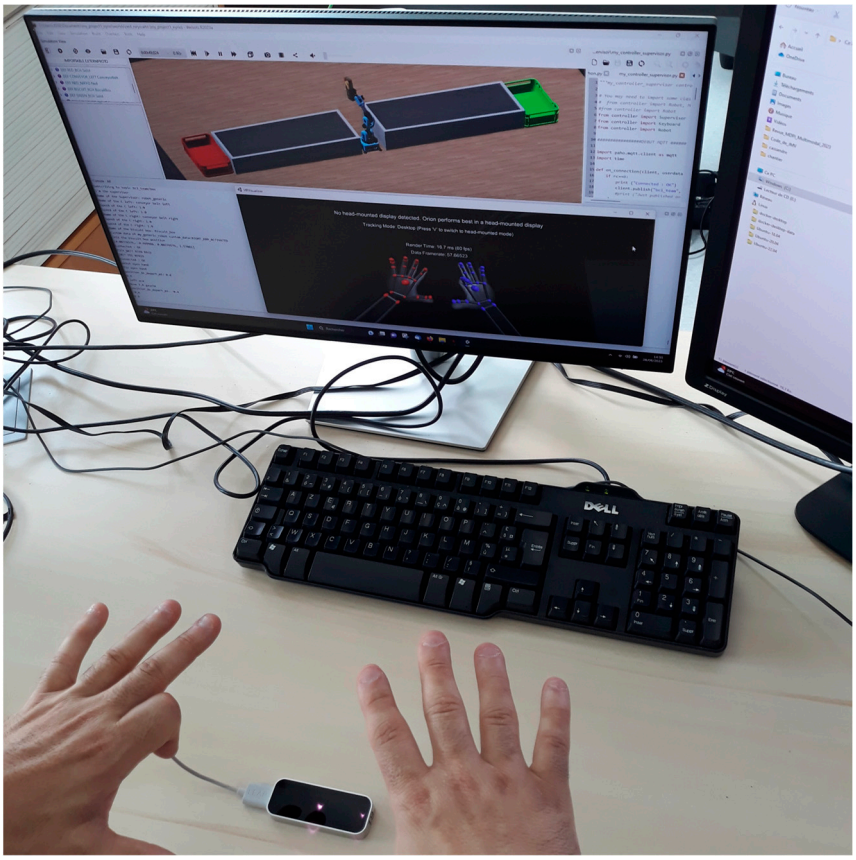
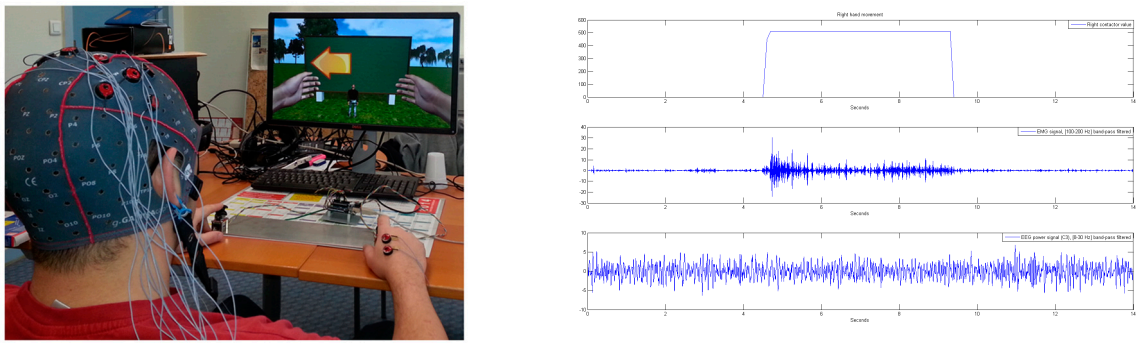


Figure 17. A leap motion is used to interact with a virtual robot. The point of view is user oriented, and the left-hand movement will send a LEFT MQTT message.



(**a**) (**b**)

Figure 18. (**a**) A user can interact with a virtual world thanks to joysticks, hand movements or brain activity; (**b**) corresponding signals from the joystick, EMG and EEG.

Figure 19 explains how we detect brain signal activity on a Laplacian around the CZ electrode when the user activates his/her foot. In the same way, a movement of the left hand (or the right hand) can be detected by focusing on the C4 (or the C3) electrode (the contralateral side). The OpenViBE software can transmit this information across LSL (LabStreaming Layer protocol) or MQTT protocols, for instance.

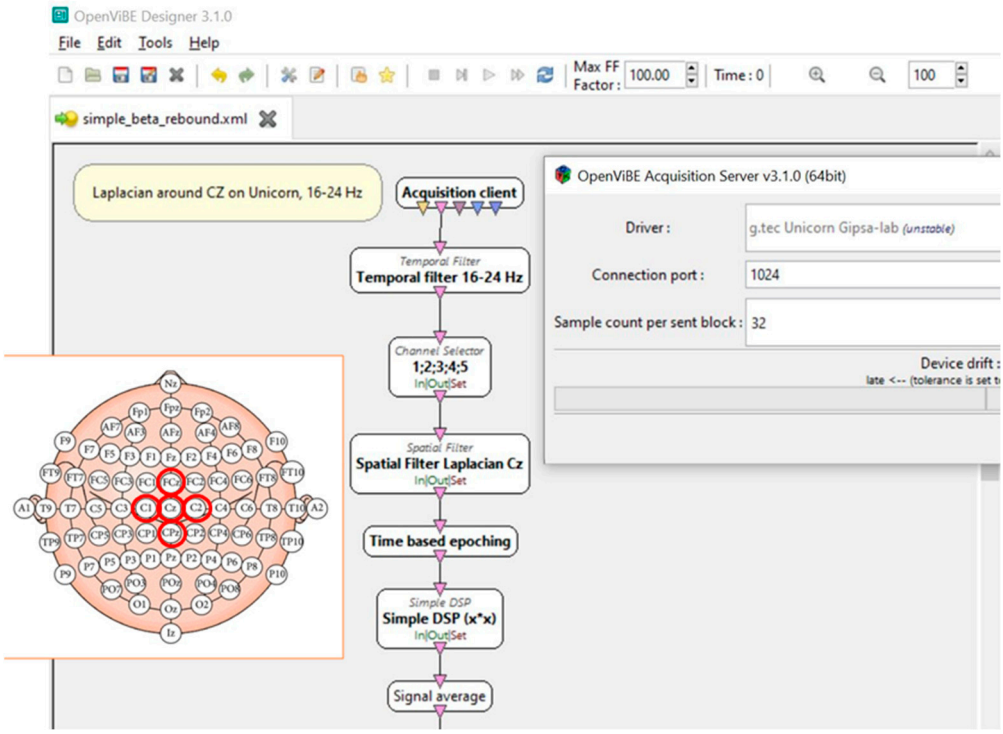


Figure 19. The OpenViBE software used to detect brain activities, with or without residual muscle activity.

Even if the patient is paralyzed, for example, and no muscle activity is detectable, brain activity remains identifiable. For this reason, brain–computer interfaces (BCI) are interesting when the patients do not have enough strength to engage their muscles (hand, arm, legs, feet, etc.) while their cognitive activity remains intact.

5. Discussion, Conclusions and Perspectives

5.1. Discussion

In the literature, MQTT is seen as the standard protocol for communication in the IoT context. Unlike MQTT, the Firebase solution we previously tested cannot be deployed locally, and may pose issues with privacy and access to services that require a reliable internet connection.

During our experiments, users were all able to easily interact in a multimodal way, depending on their own context of use (bi-manual touch on two smartphones placed side by side, voice (ASR and TTS), and gesture with a leap motion), and they all appreciated being able to switch between them seamlessly during the same session.

However, it should be noted that some users have mentioned a certain fatigue when having to hold their arms up in the air in order to collaborate with the simulated robot present in the Webots scene. Similarly, an improvement in the interface was requested by users so that they could better locate themselves spatially in the virtual scene during collaborative tasks with cobots (see left/right disorientation).

The hypothesis of using MQTT, a standard and recognized communication solution, and an IOT technology from the world of industry has shown that technology is not an obstacle, and should not be an obstacle to the employment of disabled people.

The problem is less straightforward when it comes to interacting with people. We have shown that human–machine interaction technologies can be easily integrated into an IOT environment or used as a production tool. On the other hand, depending on the nature of the disability and its evolution over time, or in the case of highly sterile conditions, the problem becomes more complex. Our work prospects aim to combine several modes of interaction to improve the system's performance by sensor data fusion.

5.2. Conclusions and Perspectives

In this article, we look at various personal assistance technologies. The choices we have made concern the chain as a whole. From human–machine interaction with disabled people to the means of exchanging information via networked communications, the aim is to enable these people to interact more effectively with their domestic environment and to facilitate their integration into the workplace.

Our results show that the joint use of industrial equipment (IOT2040) and open-source hardware and software is a convincing solution in terms of robustness and interoperability. Indeed, no message was lost during several days of intensive testing, for experiments with healthy users with various types of equipment (laptops, smartphones and industrial terminals) operating through different networks.

Currently, the MQTT messages received are immediately executed, but one could consider a delay, allowing the user to send commands in multimodal ways, within a certain window of time. Thus, a multimodal data fusion agent would be in charge of receiving these intermediate data, coming from different possible sources (voice, touch, gaze tracking, etc.) and would make a final decision to send these to the robot via a single MQTT message, after having dealt with any possible conflicts [44]. This kind of solution was presented in [2] for a paint application using Firebase and App Inventor (see multimodal engine on Figure 1), and is certainly reproducible with MQTT.

Similar to MQTT, ROS [45] is based on publishing and subscribing mechanisms. Accompanied by MoveIt [46], they are interesting for controlling robots and calculating movement trajectories in a constrained environment. We have used them, across Docker, for «Pick and Place» requests with Unity connected to Firebase, and it should be also possible to use MQTT instead. However, setting them up is not as simple as advertised, as there are many compatibility issues between versions (cf. ROS1 vs. ROS2, Moveit1 vs Moveit2). For example, in Webots, a "Ned Niryo" virtual robot is available, and controllable with ROS2, but if you use a real "Niryo One" robot, connected by USB or Wi-Fi, it will only be controllable in the ROS1 version, or it is necessary to use a specific bridge between the two versions of ROS1 and ROS2.

One of our prospective short-term studies will be verifying the usability of our tools when patients will have to communicate with mobile robots like the PR2, visible in Figure 20. We will also further improve the capabilities offered by our systems by integrating automatic data recording (score, time elapsed to complete a task, etc.) in order to better compare the different modalities that can be used. Finally, in the medium and long term, we are planning to add complex fusion and fission functionalities to our framework, in order to support more natural interactions for patients.

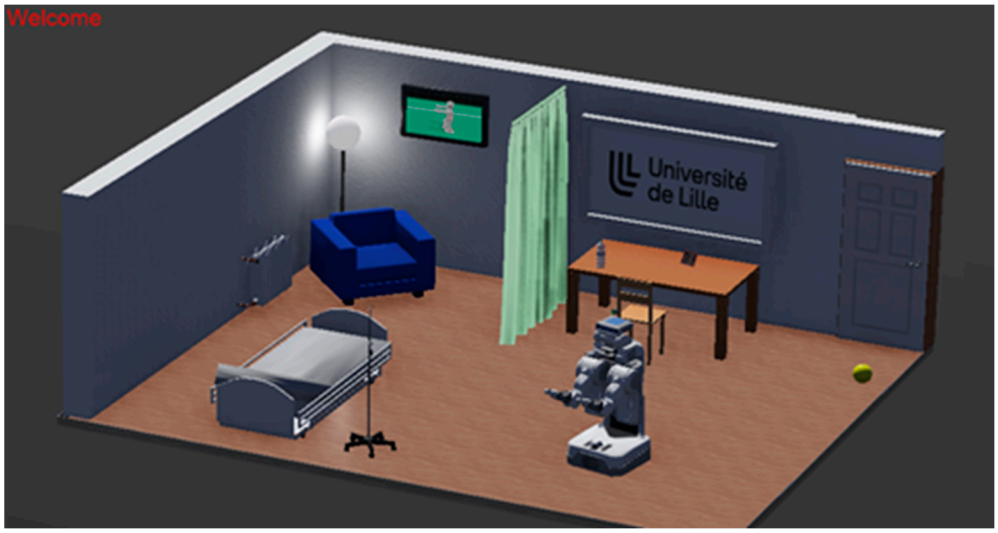


Figure 20. Webots virtual world with interactive objects (lamp, curtain, bed) and PR2 Robot.

Author Contributions: Conceptualization, J.R. and J.-M.V.; methodology, J.R. and J.-M.V.; software, J.R. (Webots and MQTT, LeapMotion) and J.-M.V. (ESP, Arduino, IOT and MQTT, LeapMotion); writing—original draft preparation, J.R.; writing—review and editing, J.R. and J.-M.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is containted within the article.

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

References

1. Bolt, R.A. "Put-that-there": Voice and gesture at the graphics interface. *ACM SIGGRAPH Comput. Graph.* **1980**, *14*, 262–270. [CrossRef]
2. Guedira, Y.; Rouillard, J. Multimodal Interaction Framework Based on Firebase Real-Time Database. In *Universal Access in Human-Computer Interaction. Access to Media, Learning and Assistive Environments, Proceedings of the 15th International Conference, UAHCI 2021, Held as Part of the 23rd HCI International Conference, HCII 2021, Virtual Event, 24–29 July 2021*; Proceedings, Part II; Springer: Berlin/Heidelberg, Germany, 2021; pp. 367–384. [CrossRef]
3. Rouillard, J.; Vannobel, J.-M.; Bekaert, M.-H. BIOFEE: Biomedical Framework for Enhanced Experimentation. In Proceedings of the 14th International Conference on Applied Human Factors and Ergonomics, San Francisco, CA, USA, 23–24 July 2023.
4. Filist, S.A.; Al-Kasasbeh, R.T.; Shatalova, O.V.; Aikeyeva, A.A.; Al-Habahbeh, O.M.; Alshamasin, M.S.; Alekseevich, K.N.; Khrisat, M.; Myasnyankin, M.B.; Ilyash, M. Classifier for the functional state of the respiratory system via descriptors determined by using multimodal technology. *Comput. Methods Biomech. Biomed. Eng.* **2022**, 1–19. [CrossRef] [PubMed]
5. World Health Organization. WHO | World Health Organization. Available online: https://www.who.int/ (accessed on 28 June 2023).
6. Oviatt, S.; Jacko, J.A.; Sears, A. (Eds.) *The Human–Computer Interaction Handbook*; Multimodal Interfaces; Lawrence Erlbaum Associates: Mahawah, NJ, USA, 2003; pp. 286–301.
7. Rouillard, J. Multimodal and Multichannel issues in pervasive and ubiquitous computing. In *Multimodality in Mobile Computing and Mobile Devices: Methods for Adaptable Usability*; Information Science Reference; Idea Group Inc.: Calgary, AB, Canada, 2009; ISBN 978-1-60566-978-6.
8. Rouillard, J. Developing a multimodal application for a scientific experiment on smartphone: Case study, tools and results. In *Tools for Mobile Multimedia Programming and Development*; Tjondronegoro, D., Ed.; Part of the Advances in Wireless Technologies and Telecommunication (AWTT) Book Series; IGI Global: Dauphin, PA, USA, 2013; ISBN 9781466640542.
9. Zavala, S.P.; Chicaiza, K.O.; López, J.L.M.; Sulca, J.; Yoo, S.G. BCI Based Home Automation using User Controlled Blinks. *J. Eng. Appl. Sci.* **2020**, *15*, 1377–1384. [CrossRef]
10. Švec, J.; Neduchal, P.; Hrúz, M. Multi-modal communication system for mobile robot. In *IFAC-PapersOnLine*; Elsevier: Amsterdam, The Netherlands, 2022; pp. 133–138, ISSN 2405-8963. [CrossRef]
11. Hoffman, G. Openwoz: A Runtime-Configurable Wizard-Of-Oz Framework for Human-Robot Interaction. In Proceedings of the 2016 AAAI Spring Symposia, Palo Alto, CA, USA, 21–23 March 2016; Stanford University AAAI Press: Washington, DC, USA, 2016; pp. 121–126.
12. Unity Technologies. Unity Real-Time Development Platform | 3D, 2D VR & AR Visualizations. Available online: https://unity.com/ (accessed on 26 April 2020).
13. OASIS Open MQTT. Available online: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt (accessed on 28 June 2023).
14. MQTT Organization. Available online: https://mqtt.org (accessed on 28 June 2023).
15. IBM MQTT. Available online: https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/ (accessed on 28 June 2023).
16. OPC USA. Available online: https://commsvr-com.github.io/Documentation/ModelDesigner/html/a2d55988-b59a-4a87-95b9-933f6bbdf5bd.htm (accessed on 28 June 2023).
17. UA Information Model Concept. Available online: https://commsvr.gitbook.io/ooi/semantic-data-processing/informationmodelconcept (accessed on 28 June 2023).
18. Webots, Webots, Open-Source Mobile Robot Simulation Software. Available online: http://www.cyberbotics.com (accessed on 28 June 2023).
19. Michel, O. Webots: Professional Mobile Robot Simulation. *J. Adv. Robot. Syst.* **2004**, *1*, 39–42.
20. ESP Rainmaker. Available online: https://www.espressif.com/ (accessed on 28 June 2023).
21. ESP32 MQTT. Available online: https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/ (accessed on 28 June 2023).

22. ESP-MQTT. Available online: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/mqtt.html (accessed on 28 June 2023).
23. Siemens Simatic IoT2000-2020-2040. Available online: https://www.14core.com/siemens-simatic-iot2000-iot2020-iot2040-arduino-ide-integration (accessed on 28 June 2023).
24. Incase Project 2014–2020. Available online: https://www.incase2seas.eu (accessed on 28 June 2023).
25. Siemens Simatic IoT2000. Available online: https://www.siemens.com/global/en/products/automation/pc-based/iot-gateways/iot2000.html (accessed on 28 June 2023).
26. Dobot Magician. Available online: https://www.dobot-robots.com/products/education/magician.html (accessed on 28 June 2023).
27. Folgado, F.J.; González, I.; Calderón, A.J. Data acquisition and monitoring system framed in Industrial Internet of Things for PEM hydrogen generators. *Internet Things* **2023**, *22*, 100795. [CrossRef]
28. Omidi, S.A.; Baig, M.J.A.; Iqbal, M.T. Design and Implementation of Node-Red Based Open-Source SCADA Architecture for a Hybrid Power System. *Energies* **2023**, *16*, 2092. [CrossRef]
29. Lopez, N.; Agbu, A.; Oloyede, A.; Essien, E.; Eze, A.; Mhambe, C. Software tool to store IoT device data onto a blockchain. *Softw. Impacts* **2023**, *16*, 100511. [CrossRef]
30. App Inventor, Google and MIT. Available online: http://ai2.appinventor.mit.edu (accessed on 28 June 2023).
31. PahoMQTT Extension for App Inventor. Available online: http://ullisroboterseite.de/android-AI2-PahoMQTT-en.html (accessed on 28 June 2023).
32. Ultraleap Inc., Ultraleap—Hand Tracking Module. Available online: https://www.ultraleap.com/ (accessed on 28 June 2023).
33. Alimanova, M.; Borambayeva, S.; Kozhamzharova, D.; Kurmangaiyeva, N.; Ospanova, D.; Tyulepberdinova, G.; Gaziz, G.; Kassenkhan, A. Gamification of Hand Rehabilitation Process Using Virtual Reality Tools: Using Leap Motion for Hand Rehabilitation. In Proceedings of the 2017 First IEEE International Conference on Robotic Computing (IRC), Taichung, Taiwan, 10–12 April 2017; pp. 336–339. [CrossRef]
34. Cortés-Pérez, I.; Zagalaz-Anula, N.; Montoro-Cárdenas, D.; Lomas-Vega, R.; Obrero-Gaitán, E.; Osuna-Pérez, M.C. Leap Motion Controller Video Game-Based Therapy for Upper Extremity Motor Recovery in Patients with Central Nervous System Diseases. A Systematic Review with Meta-Analysis. *Sensors* **2021**, *21*, 2065. [CrossRef] [PubMed]
35. Galván-Ruiz, J.; Travieso-González, C.M.; Tejera-Fettmilch, A.; Pinan-Roescher, A.; Esteban-Hernández, L.; Domínguez-Quintana, L. Perspective and Evolution of Gesture Recognition for Sign Language: A Review. *Sensors* **2020**, *20*, 3571. [CrossRef] [PubMed]
36. Baratè, A.; Elia, A.; Ludovico, L.A.; Oriolo, E. The Leap Motion Controller in Clinical Music Therapy—A Computer-based Approach to Intellectual and Motor Disabilities. In Proceedings of the 10th International Conference on Computer Supported Education—Volume 2: CSEDU, Funchal, Portugal, 15–17 March 2017; pp. 461–469, ISBN 978-989-758-291-2. [CrossRef]
37. Potter, L.E.; Araullo, J.; Carter, L. The leap motion controller: A view on sign language. In Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration, (OzCHI 2013), Adelaide, Australia, 25–29 November 2013; pp. 175–178. [CrossRef]
38. Vannobel, J.-M.; Bekaert, M.-H.; Baumann, J. Le Leap Motion Controller: De la souris gestuelle à la commande gestuelle, Deux études de faisabilité. In Proceedings of the Handicap 2022 Conference, Paris, France, 8–10 June 2022.
39. Leap Motion JavaScript SDK v2.3 Documentation. Available online: https://developer-archive.leapmotion.com/documentation/v2/javascript/devguide/Leap_Overview.html (accessed on 28 June 2023).
40. Insani, C.N.; Nurtanio, I.; Ilham, A.A. The effect of light on Leap Motion Controller in the classification of Sign Language Translator System. In Proceedings of the 2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 5–6 December 2019; pp. 296–300. [CrossRef]
41. Bachmann, D.; Weichert, F.; Rinkenauer, G. Evaluation of the Leap Motion Controller as a New Contact-Free Pointing Device. *Sensors* **2014**, *15*, 214–233. [CrossRef] [PubMed]
42. Rouillard, J.; Duprès, A.; Cabestaing, F.; Leclercq, S.; Bekaert, M.-H.; Piau, C.; Vannobel, J.-M.; Lecocq, C. Hybrid BCI Coupling EEG and EMG for Severe Motor Disabilities. *Procedia Manuf.* **2015**, *3*, 29–36. [CrossRef]
43. Duprès, A.; Cabestaing, F.; Rouillard, J.; Tiffreau, V.; Pradeau, C. Toward a hybrid brain-machine interface for palliating motor handicap with Duchenne muscular dystrophy: A case report. *Ann. Phys. Rehabilitation Med.* **2019**, *62*, 379–381. [CrossRef] [PubMed]
44. Martin, J.C. Tycoon: Six primitive types of cooperation for observing, evaluating and specifying cooperations. In Proceedings of the AAAI (Association for the Advancement of Artificial Intelligence), Orlando, FL, USA, 18–22 July 1999.
45. Robot Operating System (ROS). Open Source Robotics Foundation. Available online: https://www.ros.org/ (accessed on 28 June 2023).
46. MoveIt Motion Planning Framework. Available online: https://moveit.ros.org/ (accessed on 28 June 2023).