Proof of concept of a robot arm controlled by light using opensource tools

Théo DEFFRENNES, Charles MOULIN, Benoît VIDOTTO
Email: Firstname.lastname@umons.ac.be
Department of Electromagnetism and Telecommunication
Mons Polytechnic Faculty
Mons, Belgium

***Abstract*—This paper presents the implementation of the VLC open source communication system OpenVLC for controlling an Arduino robot arm. The objective is to determine if this technology can be used for applications like the Industry 4.0. For this purpose, a VLC communication between a human machine interface installed on a user laptop and the targeted robot arm has been established. In order to make a statement on the efficiency of this technology and exploit it in the best possible way for this type of application, a set of tests has been performed. After analysis of the test results, it is clear that this implementation can be used in industry. Indeed, the transmission remains correct regardless of the presence of parasitic ambient light and for distances up to 5m.**

***Index Terms*—OpenVLC, VLC, Robot arm, BeagleBone, Arduino.**

I. INTRODUCTION

Currently, communication radio frequency systems like the Wi-Fi or Bluetooth are mainly used in the industry to control tools. However, RF technology has several disadvantages. One of these drawbacks is the problem of spectrum deficiency in radio frequency wireless communications. Indeed, the limited RF spectrum constrains the growing demand for ubiquitous and high capacity connectivity. Moreover, the RF spectrum is constrained by a number of regulations. Not every frequency band can be applied when using RF communication; these bands are regulated. Another important drawback is the fact that in the industrial environment, many other RF communications produce interference. One solution to solve these major problems is to use Visible Light Communication (VLC) systems in the industry. It is an optical wireless communication (OWC) link using visible light spectrum. VLC has several advantages. First, the high bandwidth of the visible light allows to resolve the problem of spectrum deficiency because it uses the spectrum of the visible light for the communication which ranges from 380 nm to 780 nm. It corresponds to a frequency spectrum from 789 THz to 384 THz, representing a bandwidth 250 times higher than the RF bandwidth. It is thus possible to spare a part of the RF spectrum. Moreover, unlike the RF band, the VLC band is not regulated because it is used for local applications and therefore cannot interfere with external applications. Secondly, VLC is robust to interference that may be produced by electromagnetic devices. It can therefore be useful for applications sensitive to electromagnetic waves. Then, the absence of RF waves could reassure people who are dubious about them. VLC communications are also more secured than RF communications because light waves do not pass across used for communications can also be used to illuminate the room in which the communication is done. With these many benefits of using VLC, this technology starts to find a growing commercial interest and researchers are targeting broadband for industry. For the moment, the IEEE 802.15.13 (Multi-Gigabit/s Optical Wireless Communications) standard about high speed in industry is only being standardised [1].

This project proposes a proof of concept (PoC) of the use of the VLC to control a robot arm using open source tools. The use of open source resources has several benefits: it is free or very cheap, accessible to all and there is a community. The resources used to realise the PoC are: the BeagleBone Black, OpenVLC and an Arduino Uno boards as well as the Braccio robot arm from Arduino. It aims to demonstrate the usefulness of VLC technology in the domain of the industry to eventually serve the industry 4.0 which is characterised by the integration of digital technologies into manufacturing processes.It aims to achieve new gains in competitiveness and optimise consumption through energy efficiency. Connected to each other, machines in a factory or several sites and sensors exchange information. This continuous and instantaneous communication between the different tools and workstations integrated in the manufacturing and supply chains allows the optimisation of manufacturing processes and the improvement of flexibility in order to adapt to demand in real-time and to better satisfy the individual needs of each customer[2]. The paper consists of a presentation of the material used in this proof of concept, an explanation of the steps realised to establish the connection from a Human Interface Machine to the control of the robot arm and the realisation of a set of tests carried out to investigate the limits of VLC transmission.

II. EQUIPMENT USED

A. Visible Light Communication

For this project, in order to exploit the VLC technology at its maximum, an open-source platform that is also quite matured is needed so that the technology does not have to be developed by ourselves. Version 1.3 of OpenVLC has been thus used. According to the manufacturer, it is capable of maintaining a good connection of 400 kb/s on a distance of 4 meters or more, all while staying relatively cheap [3]. As with any data transmission chain, there is a transmitter, a receiver and the transmission channel, the latter being the free space in the room in which the VLC is used. The exchanged data are in digital form. At the transmitter level, communication is achieved by transposing the electric signal into a light signal with the modulation of the intensity of the light of an Light

Emitting Diode (LED). The modulation used is an OOK (On-OFF Keying) modulation: a binary "1" is coded when the light is on and a "0" when it is completely off [4]. The chosen modulation frequency is high enough (>>200 Hz) so that the light flickering cannot be detected by the human eye. Indeed, depending on the type of modulation and the data transmitted, a flickering may be perceived. This is the effect of rapidly changing light intensity that is perceptible to the human eye. This effect must be taken into account and can be corrected by correctly adapting the modulation or by inserting additional bits to ensure a constant average power. At the receiver level, there is a photodiode that converts the optical signal into an electrical signal [5].

1) Hardware: Research and development of OpenVLC is coordinated by the Pervasive Wireless Systems group of Dr. Giustiniano at IMDEA Networks Institute (Madrid, Spain), but anyone is very welcome to contribute [4]. [4] argues that apart from being used for research and teaching like its predecessors, OpenVLC1.3 can enable real-world applications which makes this proof of concept more convenient. The different versions of OpenVLC mainly consists of three parts: the BeagleBone Black board [6], a cape, and a driver (for the latter, see the software section II-A2).

BeagleBone Black (BBB): a low-cost open-source board targeted at developers and hobbyists that runs the Linux operating System. It features an AM335x 1GHz CPU, with 512 MB of RAM and 4GB on-board flash storage. We will mostly use it through Debian (Linux) through its USB port as a server or client [6].

OpenVLC cape: based on the 1.3 version, it includes a high power LED, a low power LED and a photodiode [7]. The high power LED uses 2.8 W and has a luminous flux of 400 lm [4].

2) Software: OpenVLC1.3 is the first version of OpenVLC to make use of the programmable real time units (PRU). This technology included in the BeagleBones, allows transfer of up to 400 kb/s through the light. As such, to be able to use this technology to its full potential, the HDMI port of the BBB had to be disabled. The Debian Linux operating system had to be installed on the boards. To access the BeagleBone command prompt, the best software to use is PuTTY, an SSH (Secure Shell) and telnet client for the Windows platform [8]. Then to access the files easily, we use the free file transfer protocol solution Filezilla (client version) [9].

To drive the OpenVLC cape, the OpenVLC driver were used: it is based on the 1.3 version. It allows to use the cape as a VLC transmitter or receiver by using MAC and PHY layers and controlling the programmable real time unit of the BBB [3]. The driver is available on the GitHub provided by [3] and had to be installed on the two BeagleBones.

B. Robot arm

For the robot arm (available on figure 1), another open-source platform is also best to be able to modify it. Arduino, the open-source electronic prototyping platform, offers such kind of robot arm : the Tinkerkit Braccio robot [10].

Fig. 1. Picture of the robot arm Tinkerkit Braccio

1) Hardware: The Arduino board most compatible with the Braccio arm is the Arduino Uno, which is why this board has been chosen (revision 3). This board is also one of the most basic and consists in a microcontroller board based on the ATmega328P CPU with all the necessary analog and digital pins to drive the Braccio, a USB connection, a power jack, a reset button and other useful features. A LED screen has also been used to debug the Arduino Uno as well as one RGB LED that will be used as a visual feedback to monitor if the information has correctly reached the robot arm.

Braccio is a robot arm composed of 6 servo-motors, and is analogous with a human arm : shoulder, elbow, wrist but is ended by a pinch. It interacts with Arduino via the provided shield. This shield conveniently assigns each servo motor with the PWM (Pulse With Modulation) pins of the board. This information allows to correctly connect a LED screen or a RGB LED to the assembly. The shield also feeds power to the six servo-motors.

2) Software: In order to work with the Arduino Uno and upload files to it, Arduino's own Integrated Development Environment (1.8.13) is used. The robot arm is controlled through the Arduino board with the help of the associated library [11].

As programming language for the codes created to control and establish our different connections, Python has been chosen as it offers a plethora of resourceful libraries and for its ease of use. Python 3.7 will be used through the integrated development environment Spyder 3.3.6, an extension of the Anaconda suite [12].

III. Architecture

The architecture of the whole assembly is comprised of different parts : the Human Machine Interface, the VLC transmitter, the VLC receiver and the Arduino board. A diagram of the whole architecture is provided on figure 2. To properly identify the different parts of the set up architecture, letters have been incorporated into the end-to-end communication diagram 2. Figure 9 in the appendix A is a photo showing all the equipment used to make the transmission. It also shows the different physical connections and where power supplies are used.

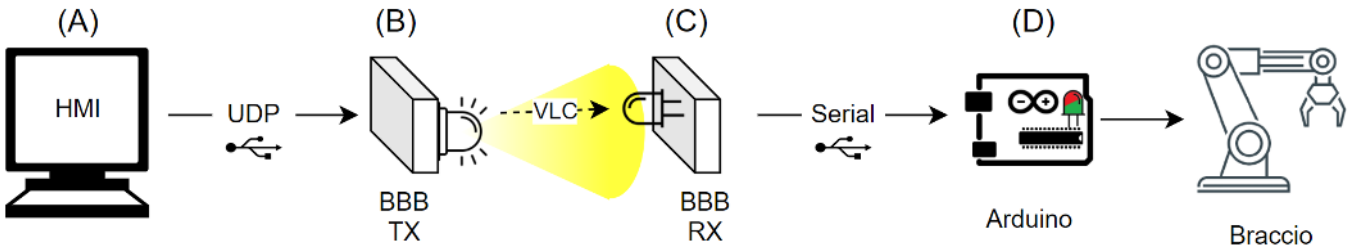

Fig. 2. End-to-end communication diagram of the complete architecture

A. Human Machine Interface

The Human Machine Interface (figure 8, in appendix A) constitutes the first part of the transmission architecture. The purpose of the interface is to provide a remote control of the Braccio Arduino by transmitting information to the BeagleBone TX. The interface must therefore provide to carry out the various operations that enable the control of the robot. Moreover, the interface aims to be as user-friendly as possible to allow each user to manipulate it.

The interface on Python has been achieved with the PySimpleGUI library ([13]).

As a reminder, this interface will have to perform many functions: send a fixed position, the real time control of the robot arm, return to a rest position, save demonstration positions and record movements composed by a sequence of several positions. The real-time control of the robot consists in sliders and spins (text input) for each motor of the robot as well as for the speed of these motors. These two objects are linked to each other through software. At the opening of the code, sliders are initialized to a preset rest position to place the robot arm motors in a safety position before starting any sequence of movements. To control the robot in real time, it is sufficient to move the slider to the desired value or to manually enter the desired value directly into the spin. Each time the value is changed, the information is sent and the robot moves. To send a fixed position, a ”Send” button has been added and can be used to retrieve the current value of the sliders/spins in case the communication has had a malfunction previously during real-time communication. A ”Rest” button has been also added, allowing the user to return the robot to its safety resting position directly and updating the values of the different sliders/spins to the values of the resting position. Regarding the saving of a demo position, there is a ”Save Position” button to retrieve the current values of the different sliders/spins. It is thus possible to visualise the position before saving it and save them. Afterwards, a simple click on the button corresponding to the saved position is needed to send it directly. Finally, we can also record a sequence of movements by clicking on the ”Save movement” button, and then choosing one of the three possible movement register places. A page opens permitting the user to choose how many positions will constitute the movement. Then the user defines the different spins for the different motors and speed. Then, to send this sequence of movements, click on ”Movement X” (X = 1, 2 or 3) and the BeagleBone TX will send the different positions in turn, respecting the delay in between.

A client-server connection has been chosen to establish the connection between the human machine interface and the BBB with the VLC transmitter cape. The client-server connection consists in a client Python code set in the interface and a server Python code imported into the BeagleBone TX. In this connection, the interface acts as a client and the transmitter as a server. Note that here, the user’s laptop with the graphical interface is connected via a USB cable to the BBB TX. To send the data, a function is placed in our interface whose purpose is to create a socket with the IP address and the port of the server. Note that the IP address of OpenVLC is programmable and the port generally used is 10001. The User Datagram Protocol was chosen: no response is expected from the servers and it offers a better response time for our real time application. The function later sends to the server the different angles of the six engines plus the speed (rotation speed of the servo-motors) in the form of a string. The real-time sending is done by sending all the data at each angle value change. For the one-time sending of data with the ”Send” or ”Rest” button, the information is sent 5 times consecutively for redundancy purpose to make sure that it will reach the robot at least once, in order to deal with problems of a link that might be defective. However, the robot will only perform the action the first time it receives the packet as the robot does not move if there is no difference between two packets received. These will be covered in the Arduino section III-D.

Before sending the data, a sending condition using the different values of the motors has been added in the ”Sent” function. In its rawest form, the robot arm does not process any data sent to it. Thankfully, the Braccio’s library comes with limit angles in order to prevent excess uses of the servo-motors. However, despite these safety features, it can occur that the arm sets itself in an unworkable position due to the different angles requested by the users which cannot be matched. It is therefore possible to have a set of angles that would place the robot arm below the level of the work surface on which it is placed. To prevent this, a formula which takes into account the length of the different parts of the robot arm has been established. It adds up the different vertical components of the arm depending on the angles required and ensures that the whole arm is above the level of the work surface. If this condition is fulfilled, the Human-Machine interface sends the data otherwise an error message is deployed telling the user that the selected movement is not allowed. The formula obtained to check the sending condition is available at the equation 1, with lengths l_1, l_2, l_3, l_4 and angles a, b, c

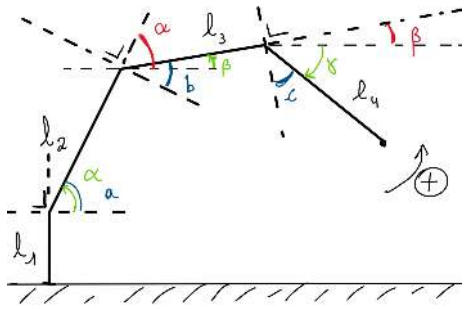

Fig. 3. Diagram of the Braccio arm

can be identified at figures 1 and 3. This formula ensures that the point at the end of the pinch (at the end of l_4) does not go beneath the ground 3.

$$l_1+l_2\sin(a)+l_3\sin(a+b-90)+l_4\sin(a+b+c-180)\geq 0 \quad (1)$$

B. VLC transmitter

The VLC transmitter is composed of the BeagleBone with its software configured as transmitter and an OpenVLC cape (BBB TX). The simple 5V USB connection to the laptop provides it power. The information received from the HMI is converted in the form of bytes and the electrical signals received in the BBB is transformed by the OpenVLC into a light signal.

As explained in the HMI part, the TX BeagleBone plays the role of server in the client-server connection with the HMI. For the second transmission between the two OpenVLC shields a client-server connection was also chosen. In this case, the TX BeagleBone is now the client and the RX BeagleBone the server. The Python code placed in the BeagleBone TX therefore has two parts: a server part that retrieves what has been sent by the HMI and a client part that sends this data received to the BeagleBone RX by creating a new socket composed of the IP address of the RX OpenVLC and the port used.

C. VLC receiver

As the VLC transmitter, the VLC receiver comprises the software of the BeagleBone configured as receiver and an OpenVLC shield (BBB RX). This time, the BBB must be connected to a 5V power supply because it is not connected to a PC that could be used as a power source. The information received from the emitter by the VLC connection is converted from light into an electrical signal.

As both the BeagleBone and the Arduino have a USB port, a serial connection via a USB cable has been created to transmit the data received from the receiving BeagleBone to the Arduino. At the BeagleBone level, a Python code includes the creation of the server for the client-server connection between the two BeagleBones and a part which takes care of writing on the serial port the message received from the VLC transmission after having formatted it to pass it into string. Note that at the end of each send, a delay of 0.1 s is applied to allow the Arduino to process the information it receives more efficiently.

D. Arduino

The data received from the BBB RX at the Arduino has to be treated to pass the different information to the arm. The treat the information, this one has to be encoded in a certain way which includes all the different parameters (angles of every motor and speed of the movement), separators between all the different parameters and a final separator to distinguish one set of data from the other. Here follows an example of such encoding where comas separate the angles and 'b' (arbitrary) separates different set of data: "10,90,90b".

The way Arduino works is through loops. If the serial connection is available, the loop will collect every byte of data received. Once the incoming byte is the separator 'b', Arduino is certain that the batch of data is complete and the micro-controller can change the angles of every servo-motor in the robot.

In order to give feedback to the user if the robot is processing data, the RGB LED will light red. On the other hand, if the robot is available to process data, the RGB LED will be colored green.

The Arduino Uno board is powered via the USB port, by connection with the BeagleBone RX. The shield however, which associates the pins of the servo-motors to the PWN pins of the board, needs an external power supply: the power from Arduino (powered via power supply or USB) is not sufficient to power the six servo-motors.

IV. Tests performed

In order to optimise the communication between the two BeagleBones, both cards were tested in different scenarios to find the best configuration between the two. To do so, the different parameters analysed were the influence of the relative position of the cards between them (horizontal/vertical), the distance between them, the presence of artificial light or not, the time taken to establish the link. For the purpose of determining these parameters, the tests have been divided into two categories: tests to determine the influence of (artificial) interfering light and tests to determine the optimal communication distance between the BeagleBones.

A. Setup

In order to test the boards in a scientific manner, we need to find a tool to test the connection in due form. Thankfully, the Debian operating system incorporates such tool under the form of iPerf, which OpenVLC recommends for testing the platform [3]. This software allows for the measurements of the maximum achievable bandwidth of an IP network and includes many features and parameters at the user's will like timing, buffers, throughput, protocols, etc. [14]. For the tests conducted in this research, the parameters used in the transmitting BeagleBone are the address of the server, the protocol, which in this case corresponds with the protocol UDP used to control the robot, the throughput of 400 kb/s,

the size of the buffer of 800 bytes, the selected port and the length of the transmission in seconds, which will heavily depend on the test, and the function of the board, client. As for the receiving BeagleBone, the parameters define the board as server, sets an interval between every burst of data, defines the protocol, its address for the client, the size of the buffer and the port. iPerf can be launched via command line (PuTTY) or script. When a BeagleBone powers up, a script will start the OpenVLC driver and iPerf, of which the result will be logged into a file that will be reviewed afterwards. The results of iPerf include the bytes transferred, throughput, jitter, ratio of lost datagram by the total, in absolute and in percentage values. To analyse the results of the different tests the main parameter taken into account has been the rate of packets lost in the transmission.

Our first tests showed some variability that was not expected. In order to check if this variability was due to the LED and/or photodiode heating up, a long-term test of 12 hours has been carried out. The two cards were placed vertically with a separation of 76 cm, keeping the BeagleBones aligned, in the dark with the least ambient light interference as possible. The test aims to observe the time taken to establish and stabilise the link, possibly because of the heat. As illustrated on figure 4, the connection is very stable over time, and settles around a mean of 12%. Since the purpose of this test is to analyse the heating the equipment, a simple look at the graph suggests that the heating won't be a problem in future tests. This test also defines the acceptable duration on which the connection is established and stays stable to perform the future tests. As such, by taking into account a margin of safety, all the future test of this report will last 20 minutes as the connection would be stable and established and waiting any longer would not provide any supplementary information. Those 20 minutes would allow us to observe any phenomenon in the connection. Finally, for parameter-specific tests, the studied parameter will be modified every 20 minutes.

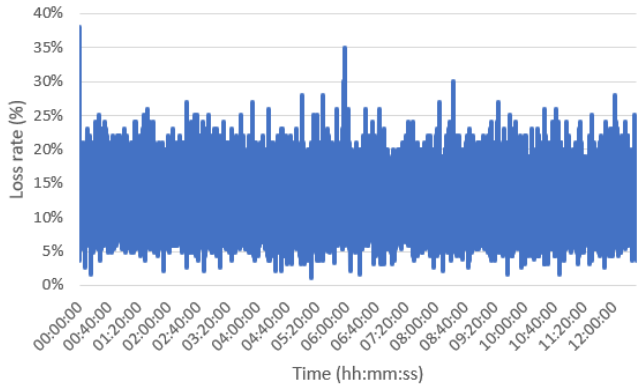

Fig. 4. Result of the 12-hour test in the dark

B. Interference of the light

The following test is intended to show the influence of light on the communication between the BeagleBones. Indeed, in the industry point of view, other lights in the building could influence the quality of the VLC communication. In this test, the two BeagleBones have been placed in a dark room, vertically and 1.86 meter apart (see appendix C for a representation of this setup). To begin, the room will be lit by an artificial light. One hour and a half later, the parasitic light will be turned off and the test will continue for another hour and a half. The results of this test can be seen on figure 5. No major difference can be observed before and after the 1h30 mark (in red), when the parasitic light has been turned off: the communication seems to adapt perfectly to this interference. The average packet loss rate remains constant in both cases and hovers around a value of 1.2%. A peak is present at the beginning of the communication over a short period of time during which the receiver is blinded.

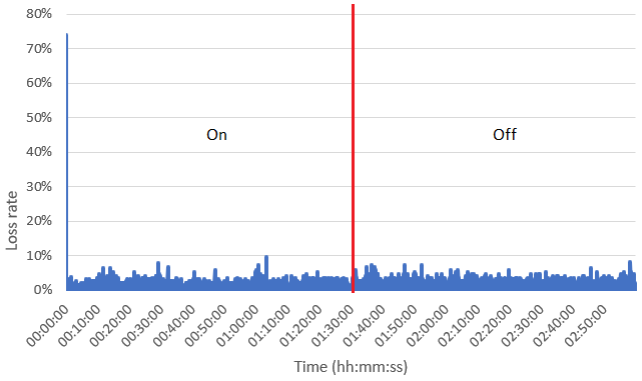

Fig. 5. Communication in the dark with lights on then off after 1 hour 30

However, in order to ensure that artificial light did not impact communication, we carried out a second test under the same physical conditions (BeagleBones placed horizontally at with 1m86 distance), while turning on and off an artificial light every 20 minutes during 2 hours (initially, the light will be on). The purpose of this test would be to see the impact of frequent changes in external light on the rate of packet loss and to possibly observe other effects on communication as the environment parameters are changed. As shown in Figure 6, there is no change when the light switches from on to off or vice-versa. The VLC communication between the two BeagleBones does not seem to be impacted by the external light. The average lost packet rate this time was 0.92% and a peak is again seen in the establishment of the communication.

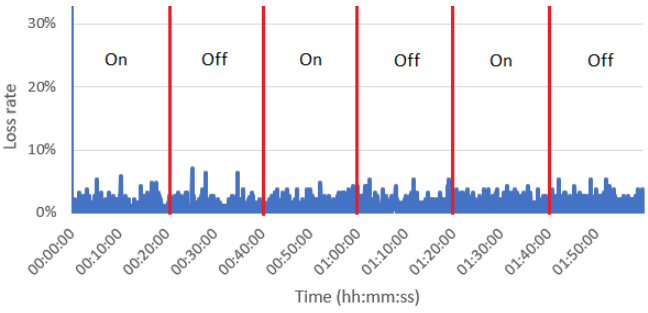

Fig. 6. Communication in the dark with artificial on/off every 20 minutes

C. Influence of the distance

To show the influence of the distance, a large-scale test has to be executed to cover a large distance. We chose to cover a distance ranging from 7 m to 38 cm, keeping the BeagleBones aligned. The setup is therefore divided into two parts to make it easier to reach large distances. The results of both experiments are concatenated on figure 7.

First, the BeagleBones were placed vertically at a distance of 1.90 m, gradually decreasing the distance between them of 19 centimeters height every 20 minutes. At the lower distances on figure 7, the number of lost packets is negligible above 1 meter but it increases below 1 meter. This may be due to receiver being blinded by the incoming light.

Secondly, the BeagleBones were placed horizontally and the distance between them varied from 7 meters to 1.9 meter. This time, the boards were placed horizontally, as preliminary testing showed that the orientation does not impact the rate loss (the test is not reported in this document). This orientation has been chosen as setting up a 7 meters vertical test is not the most practical. On figure 7, the rate of lost packets increases considerably from 5.5 meters up to 100% at 7 meters. This test shows that the interval at which the VLC communication works best with OpenVLC 1.3 is from 1 meter up to 5.5 meters. In contrast to the packet loss rate, the throughput will be minimum when the loss rate is maximum. The maximum throughput in our experiment is 400 kb/s and is compatible with the OpenVLC datasheet.

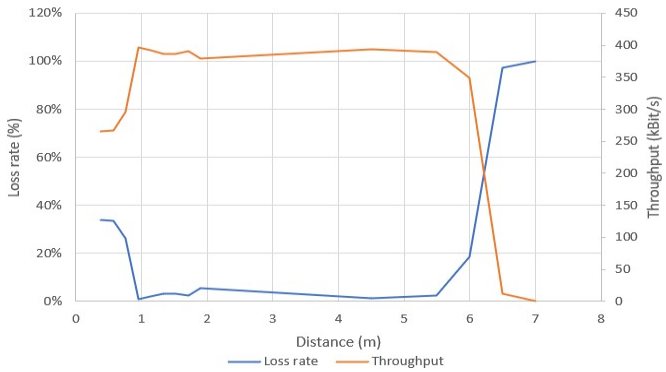

Fig. 7. Loss rate and throughput as a function of the distance

D. Conclusion of the tests

As a result of these tests, it can be stated that the VLC communication through OpenVLC1.3 is insensitive to light interference and can therefore be used along other artificial lighting. Concerning the distance, we have a wide range of uses from 1 meter to 5.5 meters with an indifference concerning the configuration (vertical or horizontal) between the BeagleBones. Finally, in comparison with the performances announced by the manufacturer (400 kb/s throughput with the possibility of communicating beyond 4 meters), the last test confirms these expected performances as we obtain a similar throughput in our range of use which goes up to 5.5 meters.

V. Conclusion

The setting up of real-time control of a robot arm from a VLC communication using the open source tools OpenVLC and Arduino works well. For this purpose, different components have been used: two black BeagleBones, two OpenVLC caps, one Arduino Uno and the Arduino robot Braccio. The architecture of the transmission includes different types of connections. There is a client-server connection via a USB cable between the Human Machine Interface and the VLC transmitter. Then, another client server connection is established with a Visible Light Communication between the VLC transmitter and the VLC receiver. Finally, a serial connection is made between the VLC receiver and the Arduino, which in turn transmits the commands to the robot arm motors.

The performance of the communication with OpenVLC has been tested too. These tests focused on the influence of the distance between the transmitter and the receiver and the impact of light interference. It was therefore found that the ambient artificial light in the room where the OpenVLC transmission is placed has no influence on its quality. Moreover, the transmission remains correct up to a distance of 5.5 meters and whether the OpenVLC transmitter and receiver are placed vertically or horizontally. The results of the actual tests confirm also that the developers of the OpenVLC open source software claimed: an average throughput of 400 kb/s and a stable connection over a distance of up to 5 meters.

The OpenVLC open source technology is therefore a promising technology that has every reason to find its place in the industry. It can therefore compete with the commonly used RF communications, which have several shortcomings such as the limited RF spectrum or the sensitivity to interference. However, the technology is not yet fully matured. Some challenges like interference between VLC devices or the integration of the VLC with existing technologies such as Wi-Fi may still need to be achieved.

The completed project could be improved by using one of the VLC applications which is LiFi. A two-way communication between the transmitter and the robot arm could be established. This would allow, for example, to replace the LED used as a visual indicator of whether the arm is receiving the information. A communication from the receiver via infrared light could indicate to the transmitter that some packets corresponding to a part of the movement to be performed by the robot arm have not been received. The transmitter could then send them back. An error-correcting code could also be added during the creation of the frames to be sent, thus making it possible to obtain a zero loss rates. Concerning the tests we have, as a prospect of improvement, to carry out a test on the throughput to see in particular if we can go above the 400 kb/s announced by the manufacturer. To finish, all along the realisation of this project, we have noticed that the BeagleBones have a rather low reliability. Several problems have been encountered like an inability to access or modify the files stored in the BeagleBone, fluctuating loss rates depending on when the tests were performed. However this open source technology did not prevent us from carrying out this project and achieving our goals.

APPENDIX A
PICTURES

On figure 8 is a capture of the Human Machine Interface made with Python and the PySimpleGUI library.

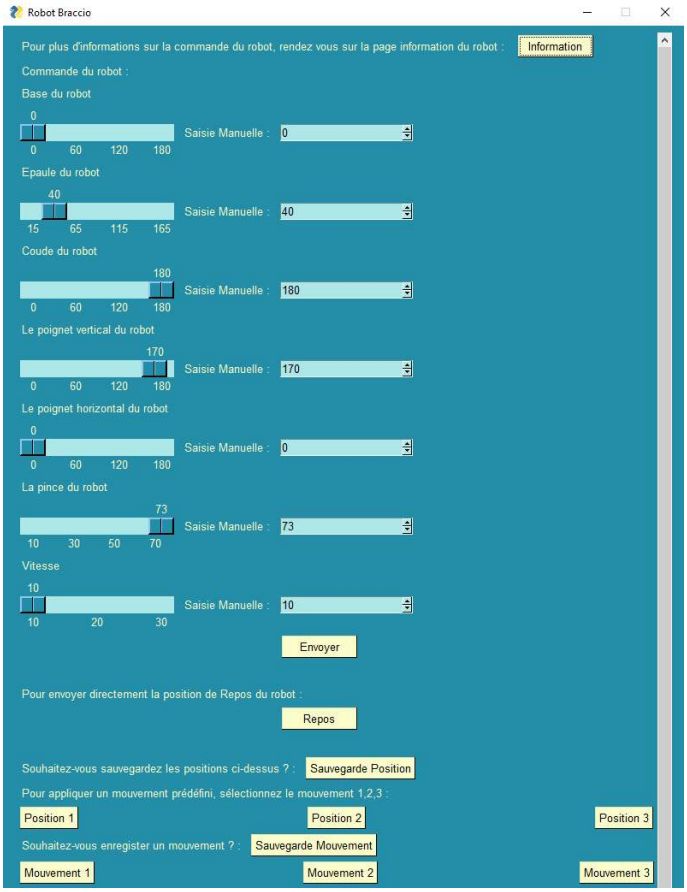

Fig. 8. Capture of the Human Machine Interface

On figure 9 is a picture showing all the equipment used to make the transmission. The different physical connections using USB cables and the place where power supplies are used can be seen.

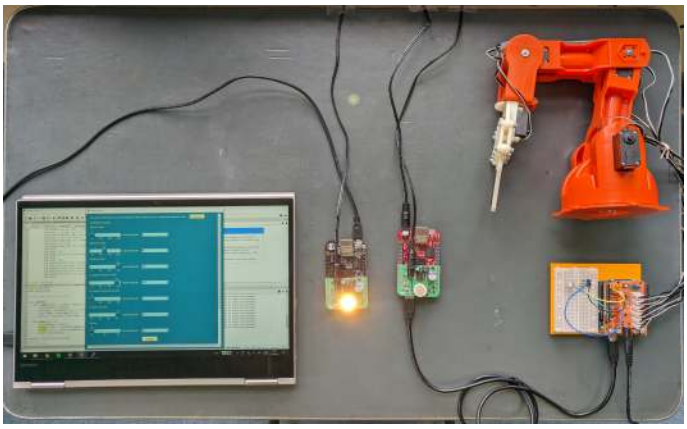

Fig. 9. Top-down picture of all the equipment

APPENDIX B
COMPLETE DEMONSTRATION OF EQUATION 1

The purpose of this calculation is to use the vertical components of each hinge, i.e. the shoulder, elbow and wrist (angles a, b and c respectively in the diagram 3) to determine how far the tip of the clamp is from the ground. Thus, when the formula is equal to zero, the tip of the clamp will be in contact with the ground and a positive value will inform that the clamp is above the ground. The angles used to calculate these vertical components are unknown (α, β and γ), so it is necessary to express them with known values: a, b, and c. Finally, the formula will be a sum of weighted sines and must be positive or equal to zero to avoid the clamp going further than possible, i.e. below the ground.

$$l_1 + l_2 \sin(\alpha) + l_3 \sin(\beta) + l_4 \sin(\gamma) \geq 0 \tag{2}$$

In this equation, the angles α, β and γ must be determined. The α angle is simple to determine: it is the angle a of the shoulder. l_1, l_2, l_3 and l_4 are the length of each contraption of the arm as seen on figures 1 and 3.

$$\alpha = a \tag{3}$$

$$\begin{aligned} 90 - \alpha &= b - \beta \\ -\beta &= 90 - \alpha - b \\ \beta &= \alpha + b - 90 \\ \beta &= a + b - 90 \end{aligned} \tag{4}$$

The calculation of angle γ is very similar to angle β except that γ is clockwise with respect to its horizontal while β is trigonometric. γ is therefore negative.

$$\begin{aligned} 90 - \beta &= c - \gamma \\ -\gamma &= 90 - c - \beta \\ \gamma &= \beta + c - 90 \\ \gamma &= a + b + c - 180 \end{aligned} \tag{5}$$

By replacing α, β and γ in equation 1 with known angles, the formula becomes:

$$l_1 + l_2 \sin(a) + l_3 \sin(a + b - 90) + l_4 \sin(a + b + c - 180) \geq 0 \tag{6}$$

with:
l_1 = 6,5 cm
l_2 = 12,6 cm
l_3 = 12,6 cm
l_4 = 19,5 cm (when the clamp is closed, i.e. the longest possible case).
a, b, c = user-defined code variables.

APPENDIX C
SETUP OF INTERFERING LIGHT TEST

In this appendix is a representation of the setup used for the test concerning the interfering light. On figure 10, the receiving BeagleBone (BBB RX) is subjected to a maximum of ambient interfering light while being aligned with the transmitting BeagleBone (BBB TX).

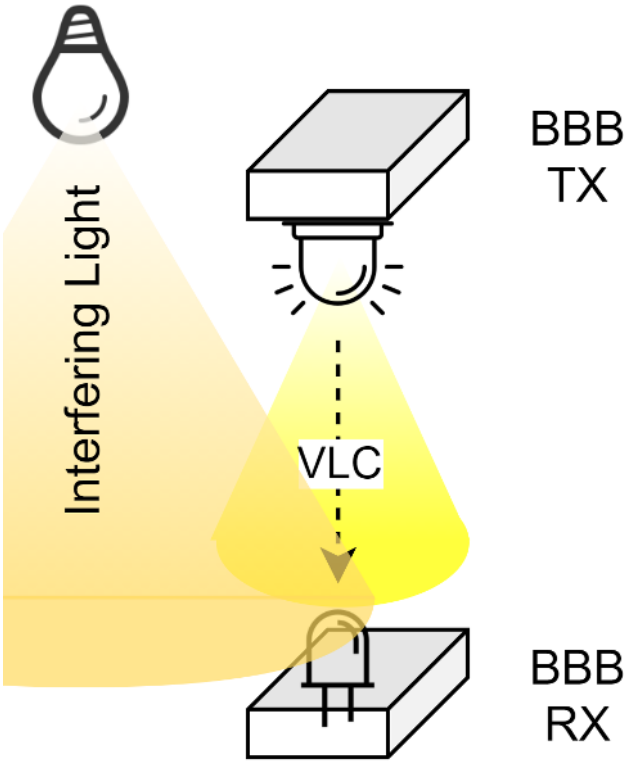

Fig. 10. Setup of interfering light test

ACKNOWLEDGMENT

The authors would like to thank Dr. Ir. professor Véronique Moeyaert, Ph.D student Ir. Véronique Georlette and Ir. Florian Piras for the help provided during the realisation of this project.

REFERENCES

[1] Latif Ullah Khan. "Visible light communication: Applications, architecture, standardization and research challenges". In: *Digital Communications and Networks* 3.2 (2017), pp. 78–88. ISSN: 2352-8648. DOI: https://doi.org/10.1016/j.dcan.2016.07.004. URL: https://www.sciencedirect.com/science/article/pii/S2352864816300335.

[2] Philippe RICHARD. "L'usine 4.0, c'est quoi ?" In: *Informatique et Numérique* (Oct. 26, 2016). URL: https://www.techniques-ingenieur.fr/base-documentaire/innovation-th10/innovations-en-electronique-et-tic-42257210/technologie-lifi-light-fidelity-te7511/ (visited on 05/27/2021).

[3] *OpenVLC*. URL: http://www.openvlc.org/home.html (visited on 05/23/2021).

[4] D. Giustiniano A. Galisteo D. Juara. "Research in Visible Light Communication Systems with OpenVLC1.3". In: *IEEE WF-IoT* (2019).

[5] Luc CHASSAGNE. "Technologie LiFi (Light Fidelity)". In: *Techniques de l'ingénieur Innovations en électronique et TIC* base documentaire : TIB257DUO.ref. article : te7511 (2017). fre. eprint: basedocumentaire : TIB257DUO.. URL: https://www.techniques-ingenieur.fr/base-documentaire/innovation-th10/innovations-en-electronique-et-tic-42257210/technologie-lifi-light-fidelity-te7511/.

[6] *BeagleBone Black*. URL: http://beagleboard.org/Products/BeagleBone+Black (visited on 05/23/2021).

[7] Q. Wang D. Giustiniano A. Galisteo D. Juara. "OpenVLC1.2: Achieving Higher Throughput in Low-End Visible Light Communication Networks". In: *WONS 2018* (2018).

[8] *Download PuTTY - a free SSH and telnet client for Windows*. URL: https://www.putty.org/ (visited on 05/23/2021).

[9] *Filezilla - The free FTP solution*. URL: https://filezilla-project.org/ (visited on 05/23/2021).

[10] *Tinkerkit Braccio robot*. URL: https://store.arduino.cc/tinkerkit-braccio-robot (visited on 05/23/2021).

[11] *Arduino Braccio Library*. URL: https://github.com/arduino-libraries/Braccio (visited on 05/23/2021).

[12] *Home - Spyder IDE*. URL: https://www.spyder-ide.org/ (visited on 05/23/2021).

[13] *PySimpleGUI*. URL: https://pysimplegui.readthedocs.io/en/latest/ (visited on 05/23/2021).

[14] *iPerf, The TCP, UDP, and SCTP network bandwidth measurement tool*. URL: https://iperf.fr/ (visited on 05/23/2021).

May 28, 2021