Development and validation of distributed communication protocol from finite-state machines

Desarrollo y validación de protocolo de comunicación distribuida a partir de máquinas de estados finitos

RODRÍGUEZ-FRANCO, Martín Eduardo†*´, MALDONADO-RUELAS, Víctor Arturo´, VILLALOBOS-PIÑA, Francisco Javier´´ and ORTIZ-MEDINA, Raúl Arturo´

´ Universidad Politécnica de Aguascalientes, Research Lab, Postgraduate and Research Direction, Mexico.
´´ Instituto Tecnológico de Aguascalientes, Tecnológico Nacional de México, Postgraduate Studies and Research Division, Mexico.

ID 1st Author: *Martín Eduardo, Rodríguez-Franco* / **ORC ID:** 0000-0002-6804-4777, **Researcher ID Thomson:** T-1539-2018, **CVU CONACYT ID:** 660892

ID 1st Co-author: *Víctor Arturo, Maldonado-Ruelas* / **ORC ID:** 0000-0003-2125-1557, **Researcher ID Thomson:** I-2774-2018, **CVU CONACYT ID:** 209649

ID 2nd Co-author: *Francisco Javier, Villalobos-Piña* / **ORC ID:** 0000-0003-1053-5642

ID 3rd Co-author: *Raúl Arturo, Ortiz-Medina* / **ORC ID:** 0000-0002-3790-3807, **Researcher ID Thomson:** H-9584-2018, **CVU CONACYT ID:** 878025

DOI: 10.35429/JOCT.2022.17.6.1.9 Received: January 10, 2022; Accepted: June 30, 2022

Abstract

This work documents the use of communication finite state machines for the representation of interactions in a network of devices. Likewise, a hierarchical structure between elements is adopted, under an organization of master-slave nodes; from which, the implemented communication protocol assumes the particular functions of each device, while ensuring the exchange of information between them. In order to validate the adequate constitution of the proposed protocol, through its respective programming, Arduino and Raspberry free hardware and software are used, as well as a basic interface created from the Thonny development environment, for data entry by the user and the feedback of information to this. The results obtained demonstrate the functionality of the developed protocol, which was implemented from the serial communication standard, supported by the hardware used. However, it is essential to clarify that it is possible to implement this type of application from the use of other standards, as available.

Communication finite-state machines, Communication protocol, Device network

Resumen

Este trabajo documenta la utilización de máquinas de estados finitos de comunicación para la representación de las interacciones en una red de dispositivos. Asimismo, se adopta una estructura jerárquica entre elementos, bajo una organización de nodos maestro-esclavo; a partir de la cual, el protocolo de comunicación implementado asume las funciones particulares de cada dispositivo, al tiempo que asegura el intercambio de información entre éstos. Con el fin de validar la adecuada constitución del protocolo propuesto, a través de su respectiva programación, se emplean hardware y software libre Arduino y Raspberry, así como una interfaz básica creada a partir del entorno de desarrollo Thonny, para el ingreso de datos por parte del usuario y la retroalimentación de información hacia éste. Los resultados obtenidos demuestran la funcionalidad del protocolo desarrollado, mismo que fuera implementado a partir del estándar de comunicación serial, soportado por el hardware empleado. Sin embargo, es indispensable aclarar que es posible implementar este tipo de aplicación a partir del uso de otros estándares, según se disponga.

Máquinas de estados finitos de comunicación, Protocolo de comunicación, Red de dispositivos

Citation: RODRÍGUEZ-FRANCO, Martín Eduardo, MALDONADO-RUELAS, Víctor Arturo, VILLALOBOS-PIÑA, Francisco Javier and ORTIZ-MEDINA, Raúl Arturo. Development and validation of distributed communication protocol from finite-state machines. Journal of Computational Technologies. 2022. 6-17:1-9.

* Correspondence to Author: (E-mail: mc190002@alumnos.upa.edu.mx)
† Researcher contributed as first author.

Introduction

Communication systems are increasingly concurrent and distributed, attributes that are especially distinguishable in Internet of Things (IoT) applications (Duhart, Sauvage, & Bertelle, 2016) (Blanc, Bayo-Montón, Palanca-Barrio, & Arreaga-Alvarado, 2021). Thus, the effective coordination between the different components implies a challenge that must be assumed from the planning (Basu, Bultan, & Ouederni, Deciding choreography realizability, 2012). And, since the contemplated devices must ensure their own executions, while interacting with each other, the term "choreography" is used to distinguish the set of valid sequences for the exchange of messages between them (Basu & Bultan, Automated choreography repair, 2016) (Cruz-Filipe, Montesi, & Peressotti, 2021).

It is worth mentioning that the model proposed by a choreography can be interpreted from finite-state machines (FMS) properly defined. (Barbanera, Lanese, & Tuosto, Choreography automata, 2020) (Orlando, Di Pasquale, Barbanera, Lanese, & Tuosto, 2021). It is a set of states, and another set of transitions; the latter establish an action or conditions that determine a change or achievement between two different states (Ben-Ari & Mondada, 2017). From finite-state machines, it is possible to represent the interaction between elements of an analyzed communication system, through the asynchronous exchange of messages, through one or several channels disposed (Lange, Tuosto, & Yoshida, 2015).

For its part, a communication protocol can be conceived as a set of finite-state machines, which establishes roles, while the messages exchanged are created with a specific vocabulary (Barbanera, de'Liguoro, & Hennicker, Connecting open systems of communicating finite state, 2019). Therefore, the modeling of a communication protocol is formalized, to represent it as a transition of configurations for the transport of information through channels (Fragal, Simao, & Mausavi, 2016). In this way, both a concrete definition of communication processes and its low propensity for deadlocks or mismatches are promoted (Yoshida, Zhou, & Ferreira, 2021); with which, the tests of conformity are automated and speed up the detection and location of faults, once the protocol has been implemented (Yang & Kim, 2019).

Originally, choreographies were used to characterize peer-to-peer communication processes (Muscholl, 2010). However, recently it is possible to represent distributed processes, through master-slave interactions (Popovic, Marinkovic, Djukic, & Popovic, 2021), interest of this work. In such a hierarchical relationship of linked devices, one of these leads the communication process, managing in parallel, the information generated in, or guided towards, one or several alternate devices (Rodríguez Penín, 2012). Moreover, the representation of the relationships established by the communication protocol used must be independent of any programming language that may be used for its implementation (Walkinshaw, Taylor, & Derrick, 2016).

Motivation

The use of formal methods for establishing operation sequences has proven its effectiveness in a large number of technological fields. Important applications have been glimpsed mainly in the industrial environment, through the implementation of highly coordinated processes, involving a large number of physical systems. However, the field of information has not been the exception, constituting complex systems that, through the execution of algorithms, exercise data processing to guide their results to the performance of some task. In both cases, the constant has been the proposal of some methodology whose expression of the required process has gone beyond both the implementation itself, as well as the means used for this.

A way to combine the development of computer applications to ensure proper handling of hardware has been exploited since the proposal of the first cyber-physical systems. Therefore, if the adequate understanding between the real and virtual counterparts is adapted to a mode of operation that ensures functional interaction between them, it is possible to guarantee the fulfillment not only of a specific task, but a set of these, according to the scope of implemented system. Such coordination between the functions of a system, based on the information acquired, processed and issued, is especially enhanced through the use of formal methods, such as finite state-machines.

From the study of finite-state machines, as well as their application to the development of communication systems, promoted by this study, it is intended to establish a model that clearly defines the interactions between several interconnected devices, for data exchange purposes. In this way, data coming from input pins, after being processed and the pertinent decisions taken, will determine specific results, which will be destined to output pins, within a distributed architecture. Through which, not only is a physical organization of devices proposed, but also, from the constituted algorithms, a total adherence to the particular function that each one must perform can be ensured, to guarantee said process.

Device network and communication protocol operation

A hierarchical network of devices is proposed, made up of three classes of elements: end device, router and coordinator, commonly associated with IoT applications (Shrestha & Shakya, 2020). The end device is the means for the acquisition of data from the environment, at the same time that it is capable of generating signals that may affect physical conditions. For its part, the router serves as the communication interface between the end devices and the one used for process management. Finally, the coordinator operation allows interaction with the system user; from the performance of the functions of monitoring and control of the variables acquired, or manipulated, through the final devices. The structure of the proposed interaction is shown in figure 1.

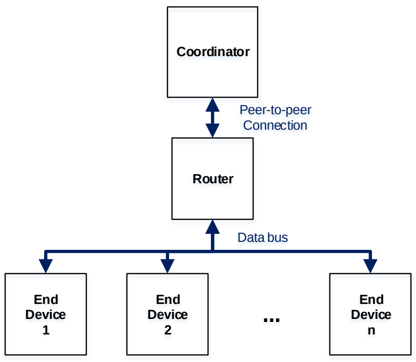


Figure 1 Proposed arrangement for device communication
Source: Own elaboration, 2022

As can be seen in the previous figure, a full-duplex interaction is promoted between the devices that conform the proposed communication network. Likewise, a peer-to-peer connection is adopted between the coordinator and the router, while a data bus is established for communication between the latter and the multiple end devices used. A data bus enables interaction between the router and the end devices by enabling communication channels, through which input or output data associated with each participating device is transported, duly addressed (Tocci, Widmer, & Moss, 2017).

Regarding the communication protocol, a structure based on finite-state machines was developed; which allows the management of certain input and output pins, of the final devices, from a remote interface. It is worth mentioning that there is no direct link, or any physical connection, between end devices and interface; for which, the router is incorporated to associate their functions. In this way, an interaction model is established under the structure of a supervision system, which could be later implemented, without the need to use a particular communication standard.

End device function

Physical input and output pins would be manipulated only in the final devices. The use of analog pins was chosen to provide a variable range of values that the device itself is capable of reading, or can generate. In this way, and if required, the management of physical processes would be facilitated through sensors and actuators that operate under the specifications described. For purposes of this analysis, the mentioned operation is contemplated for each final device incorporated into the suggested communication model. The dynamics of a final device is represented through the finite state machine of figure 2.

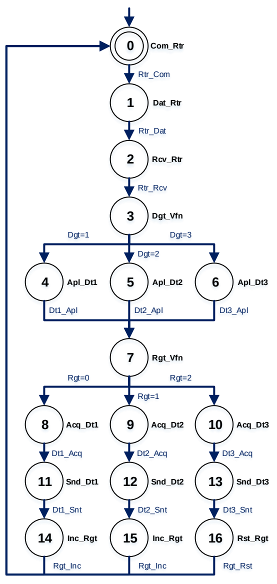


Figure 2 Operation sequence for an end device
Source: Own elaboration, 2022

The performance of the function of an end device would be triggered from the confirmation of communication with the router and the transfer of information from it; seeking immediately, the reception of the transmitted message. Three different types of data to be acquired are established, each one associated with a specific output to be manipulated; therefore, these would be differentiated from the first registered digit. Thus, regardless of the data received, it would be separated from its first digit, and the rest destined for the respective output pin.

Subsequently, the value assigned to a register in the algorithm itself would be evaluated, which would be directly associated with the signal to be acquired, coming from an input pin. Three possible values are proposed in the register, each one corresponding to the reading of a different input pin. It is worth mentioning that, regardless of the source pin, each acquired data would be sent to the router device. Whereas, once the transmission is over, the value of the register would increase by one; except that the previous value was two, in which case it would be reset to zero.

Router function

The information acquired in each end device would arrive at the router, suitably identified, to later be sent to the coordinator. However, such a management device could also send data, in this case, related to the condition of the output pins of the final devices. In this way, the router would receive the information from the coordinator, identify the final destination device and manage the transfer to it. The finite-state machine of figure 3 exposes the interaction performed by the router.

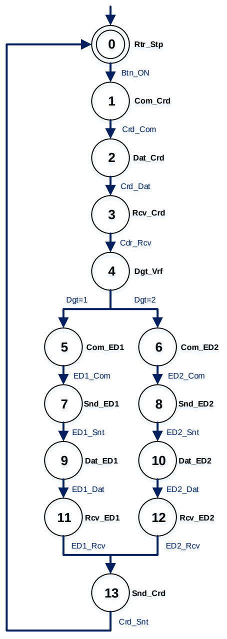


Figure 3 Operation sequence for the router
Source: Own elaboration, 2022

As proposed, the operation of the router would be established from the activation of a start button and inhibited from the activation of a stop button. Therefore, once activated, the router would verify the existence of information coming from the management device. In case of causing the transfer of a message, it would be received; while the contained data would be distinguished in reference to the first digit read. It is worth mentioning that such digit establishes the end device to which the acquired information must be destined.

However, prior to sending data to any end device, the existence of communication with it would be verified. Therefore, once such a connection is confirmed, the data transfer would proceed, excluding the first number originally registered. Additionally, and according to the end device with which communication has been established, it would be verified if it had sent any data to the router. If the transfer of any data proceeds, it would be identified, registered and sent to the coordinator.

Coordinator function

The proposed communication model would consider the coordinator operation as associated with the system user. In this way, two tasks would be assigned to this device: the registration of the data entered by the user and the presentation of information to this. It is worth mentioning that the data provided by the user would be identified, registered and then conducted, via the concentrator, to the corresponding end device, for the manipulation of one of its output pins. Meanwhile, the information exposed to the user, properly identified, would come from the input pins of each end device contemplated. The coordinator operation is exposed through the finite-state machine of figure 4.

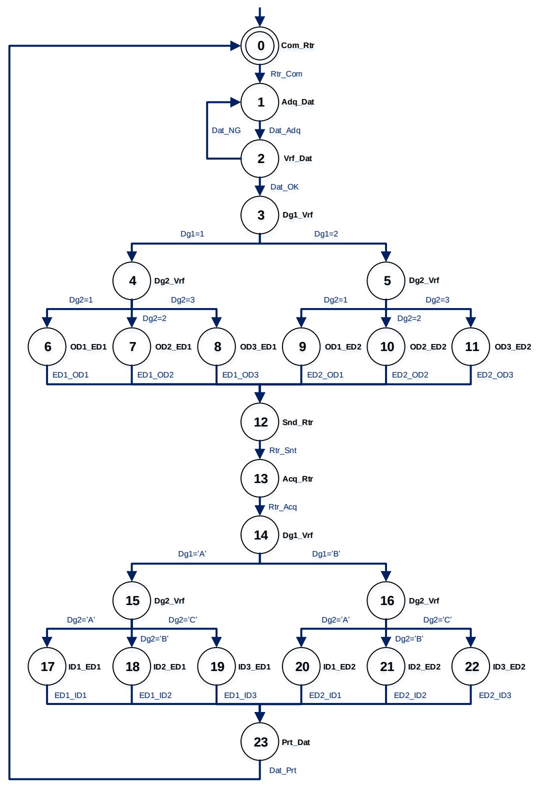


Figure 4 Operation sequence of the coordinator
Source: Own elaboration, 2022

The coordinator function would start by confirming the existence of communication with the router. Subsequently, the entry of some data by the system user would be verified, through the corresponding interface, in order to modify the status of a certain output pin in one of the end devices managed. If there is any data available, and if it complies with the format admissible by the system, it would be registered, according to the final destination device, from the first digit read. At the same time, the physical output pin to be manipulated in the end device at issue would be established, based on the second registered number, and the value assigned to said pin would be identified.

Once the registration of the data entered by the user has been validated, these would be sent to the router, and then be destined for the corresponding end device; at the same time that any data issued in the opposite communication channel would be acquired. In this case, from the identification of the first character in the received message, the final source device would be determined; while, based on the second character read, the physical input pin read would be established. Next, the registered information would be extracted by the input pin itself of the final device at issue; to conclude with the exposure to the user of both types of data processed in the system.

Implementation of the communication system

In order to validate the operation of the proposed communication protocol, the algorithms that would concentrate the functionality of each constituent device were developed. In this way, the abstraction established for each component, represented by the corresponding state machine, would be interpreted through the use of a specific programming language, while the transfer of information would be managed through some communication standard. Physically, the communication system for testing would be made up of a Raspberry Pi 4 computer as coordinator, an Arduino MEGA board as router, and two Arduino UNO boards as end devices, respectively, as shown in figure 5.

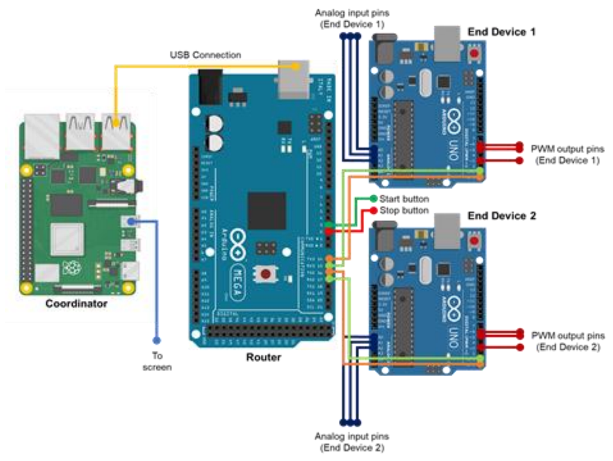
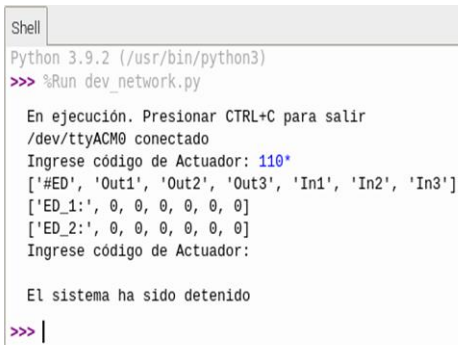

Analog input pins (End Device 1)
End Device 1
USB Connection
PWM output pins (End Device 1)
Start button
Stop button
End Device 2
Coordinator
To screen
Router
PWM output pins (End Device 2)
Analog input pins (End Device 2)

Figure 5 Distribution of devices for the communication system
Source: Own elaboration, 2022

The Python programming language would be used to ensure the operation of the coordinator; while the Arduino language would be associated with the functionality of both the router and the end devices. Likewise, the serial communication standard would be used, powered by the UART (Universal Asynchronous Receiver-Transmitter) device, which is adapted to the selected boards, for the exchange of data between them. Thus, to one of the USB ports of the Raspberry computer would be connected to the Arduino MEGA board, and in turn, to it, the Arduino UNO boards through the integrated serial communication ports.

User interface integration

In order to manage the proper exchange of information between the physical devices described, a user interface would be constituted, whose general structure can be seen in figure 6. To create the required interface, the Thonny integrated development environment was used, incorporated into the Raspberry computer's own operating system. It should be noted that this interface was completely configured in text mode, since it would not be used as a definitive means for entering and presenting information, but rather to validate the effectiveness of the interaction between the devices, based on the communication system implemented.



```
Python 3.9.2 (/usr/bin/python3)
>>> %Run dev_network.py

  En ejecución. Presionar CTRL+C para salir
  /dev/ttyACM0 conectado
  Ingrese código de Actuador: 110*
  ['#ED', 'Out1', 'Out2', 'Out3', 'In1', 'In2', 'In3']
  ['ED_1:', 0, 0, 0, 0, 0, 0]
  ['ED_2:', 0, 0, 0, 0, 0, 0]
  Ingrese código de Actuador:

  El sistema ha sido detenido

>>> |
```

Figure 6 Appearance of the user interface
Source: Own elaboration, 2022

In this way, once the proposed boards have been programmed and duly connected, the user interface will show the confirmation of communication between the coordinator and the router. However, the activation of the respective start button will be necessary for the interface to allow interaction with the user. Subsequent to the issuance of said signal, data entry by the user will be enabled. Therefore, after the user has entered a code, it will be possible to view a table with the respective values for both the status of the output pins of each of the managed end devices, as well as the signals read from of each input pin on these.

Adaptation of exchanged data

Regarding the format used to enter data to the interface, the formation of a text string is contemplated, whose first digit designates the final device with which the session will be established. The second digit of the string will correspond to the specific output channel to which the effective data would be sent. Finally, the remaining numbers in the string refer to the value that will be sent directly to the output channel, in the end device established. It is worth mentioning that the system will ignore any value entered other than those proposed. For output control purposes, data input to the interface was restricted to values between 0 and 255. Such admissible range is determined from the resolution of 8 bits, at which the PWM (Pulse Width Modulation) output channels of the Arduino UNO boards used operate; and it is equivalent to the application of a variable voltage between 0 and 5 volts of direct current. On the other hand, for the representation in the same interface of the data read from the analog input pins used, the printing of values between 0 and 1023 was considered. In this case, the input channels have a resolution of 10 bits to register a variable voltage signal between 0 and 5 volts, also direct current.

System operation results

In order to validate the functioning of the implemented system, the architecture proposed in figure 5 was physically developed, contemplating the connection of potentiometers to the input pins and of LEDs (light-emitting diode) in the output pins, of both Arduino UNO boards used, as shown in figure 7. In this way, it would be possible to characterize, or influence, as the case may be, some variation in the behavior of the final devices. Note that, for purposes of validating the results, some values were previously assigned from the interface to the LEDs connected to the end devices disposed, at the same time, that the position of each of the potentiometers used has been modified and acquired the corresponding data, as reported in figure 8.

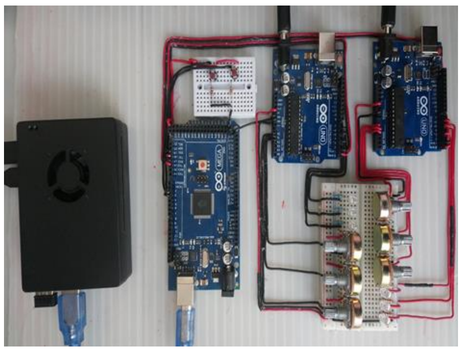
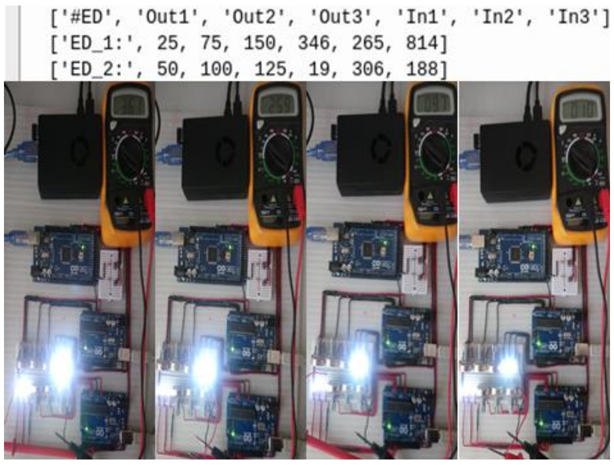


Figure 7 Communication system implemented
Source: Own elaboration, 2022

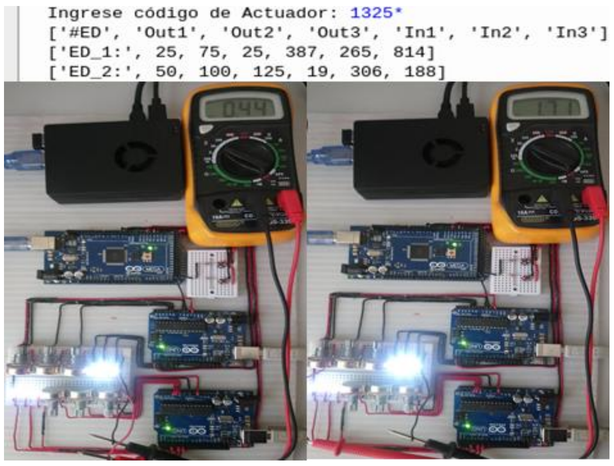


```
['#ED', 'Out1', 'Out2', 'Out3', 'In1', 'In2', 'In3']
['ED_1:', 25, 75, 150, 346, 265, 814]
['ED_2:', 50, 100, 125, 19, 306, 188]
```

a) *b)* *c)* *d)*

Figure 8 Initial state of the interface and the communication system for tests, and measured values in: *a)* the third output, *b)* the first input, both of the first end device, *c)* the first output and *d)* the first input, both of the second end device.
Source: Own elaboration, 2022

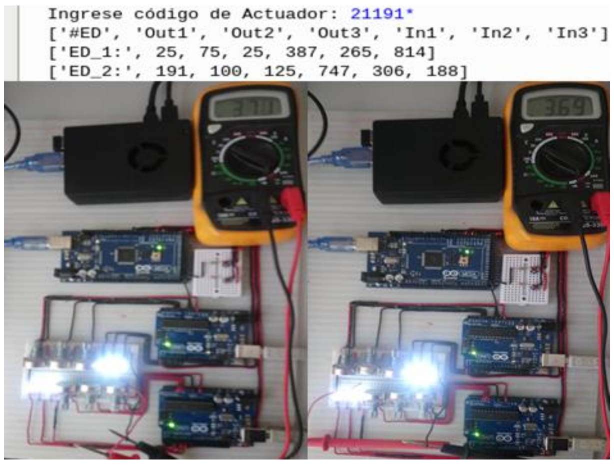
In figures 9 and 10, the designation of 25 and 191 resolution units is observed to the third output pin of the first end device, and to the first pin of the second device, respectively. Being possible to verify the equivalence between both data imposed by the user, from the interface, and those actually adopted, when measured in the corresponding output pin. Also, it is verified that, once the first data has been entered, the interface updates the value read from the first input pin of the same device to 387 units of resolution; while, for the second entered data, the value corresponding to the first input pin of the second device is the one that is updated to 747 units.



```
Ingrese código de Actuador: 1325*
['#ED', 'Out1', 'Out2', 'Out3', 'In1', 'In2', 'In3']
['ED_1:', 25, 75, 25, 387, 265, 814]
['ED_2:', 50, 100, 125, 19, 306, 188]
```

a) *b)*

Figure 9 Behavior of the interface and the communication system when: *a)* entering 25 units to the third output of the first end device, and *b)* reading 387 units from its first input
Source: Own elaboration, 2022



```
Ingrese código de Actuador: 21191*
['#ED', 'Out1', 'Out2', 'Out3', 'In1', 'In2', 'In3']
['ED_1:', 25, 75, 25, 387, 265, 814]
['ED_2:', 191, 100, 125, 747, 306, 188]
```

a) *b)*

Figure 10 Behavior of the interface and the communication system when: *a)* entering 191 units to the first output of the second end device, and *b)* reading 747 units from its first input
Source: Own elaboration, 2022

From the evidence provided, it is possible to verify that the proposed communication protocol assumes the complete mapping of the functions of each of the devices involved, regardless of the type of hardware used and its configuration. As it was validated, the use of finite-state machines as support for the planning and development of communication systems can be applicable both for peer-to-peer and distributed configurations, constituted under a hierarchical structure. Additionally, from the physical implementation, compliance with the designated operation specifications from the beginning of the development process was assessed.

Conclusions

Finite-state machines are considered powerful tools in the organization of sequential processes. However, a variant of these, used in the management of communication processes, has become very important due to the way in which it is possible to establish an interaction between devices, from abstraction to successful implementation, including planning. Emphasizing the application of finite-state machines for communication with total independence of methodologies, programming languages or established communication standards, without sacrificing the required functionality specifications.

On this occasion, in addition to exploring the adoption of this type of directed graphs in the implementation of peer-to-peer communication, its mostly widespread application, its adaptation between devices whose relationship is hierarchical, such as the master-slave interaction, was also analyzed. Therefore, it is important to highlight that the planning of a hierarchically structured information exchange process requires special care in the development of the necessary algorithms, since it is essential to ensure the proper function of each of the integrating devices, while ensuring the proper interaction between them.

Thus, the constitution of a protocol for the integration and interaction of three devices with different technical specifications is reported. And that, however, from the establishment of the effective relationships between them, by means of finite-state machines, the required exchange of data was successfully achieved.

This protocol could be validated from the use of open-source technology, such as Arduino and Raspberry, and simple electronic devices; in addition to using the serial communication standard, supported by both types of boards, to combine technologies whose potentialities and capabilities are far from each other.

Referencias

Barbanera, F., de'Liguoro, U., & Hennicker, R. (2019). Connecting open systems of communicating finite state. *Journal of Logical and Algebraic Methods in Programming*, 100476. doi:10.1016/j.jlamp.2019.07.004

Barbanera, F., Lanese, I., & Tuosto, E. (2020). Choreography automata. *International Conference on Coordination Languages and Models*, 86-106. doi:10.1007/978-3-030-50029-0_6

Basu, S., & Bultan, T. (2016). Automated choreography repair. *International Conference on Fundamental Approaches to Software Engineering*, 13-30. doi:10.1007/978-3-662-49665-7_2

Basu, S., Bultan, T., & Ouederni, M. (2012). Deciding choreography realizability. *Acm Sigplan Notices*, 191-202. doi:10.1145/2103621.2103680

Ben-Ari, M., & Mondada, F. (2017). Finite State Machines. En M. Ben-Ari, & F. Mondada, *Elements of Robotics* (págs. 55-62). Cham: Springer. doi:10.1007/978-3-319-62533-1_4

Blanc, S., Bayo-Montón, J. L., Palanca-Barrio, S., & Arreaga-Alvarado, N. X. (2021). A service discovery solution for edge choreography-based distributed embedded systems. *Sensors*, 672-690. doi:10.3390/s21020672

Cruz-Filipe, L., Montesi, F., & Peressotti, M. (2021). Certifying choreography compilation. *International Colloquium on Theoretical Aspects of Computing*, 115-133. doi:10.1007/978-3-030-85315-0_8

Duhart, C., Sauvage, P., & Bertelle, C. (2016). A resource oriented framework for service choreography over wireless sensor and actor networks. *International Journal of Wireless Information Networks*, 173-186. doi:10.1007/s10776-016-0316-1

Fragal, V. H., Simao, A., & Mausavi, M. R. (2016). Validated test models for software product lines: Featured finite state machines. *International Workshop on Formal Aspects of Component Software*, 210-227. doi:10.1007/978-3-319-57666-4_13

Lange, J., Tuosto, E., & Yoshida, N. (2015). From communicating machines to graphical choreographies. *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 221-232. doi:10.1145/2676726.2676964

Muscholl, A. (2010). Analysis of communicating automata. *International Conference on Language and Automata Theory and Applications*, 50-57. doi:10.1007/978-3-642-13089-2_4

Orlando, S., Di Pasquale, V., Barbanera, F., Lanese, I., & Tuosto, E. (2021). Corinne, a tool for choreography automata. *International Conference on Formal Aspects of Component Software*, 82-92. doi:10.1007/978-3-030-90636-8_5

Popovic, M., Marinkovic, V., Djukic, M., & Popovic, M. (2021). Formal verification of distributed master-slave finite state machine. *2021 29th Telecommunications Forum (TELFOR)*, 1-4. doi:10.1109/TELFOR52709.2021.9653241

Rodríguez Penín, A. (2012). *Sistemas SCADA.* Barcelona: Marcombo.

Shrestha, S., & Shakya, S. (2020). Technical analysis of ZigBee wireless communication. *Journal of trends in Computer Science and Smart technology (TCSST)*, 197-203. doi:10.36548/jtcsst.2020.4.004

Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Sistemas digitales: Principios y aplicaciones.* Naucalpan de Juárez: Pearson Educación.

Walkinshaw, N., Taylor, R., & Derrick, J. (2016). Inferring extended finite state machine models from software executions. *Empirical Software Engineering*, 811-853. doi:10.1109/WCRE.2013.6671305

Yang, Q.-P., & Kim, T.-H. (2019). Fault diagnosis of a protocol implementation specified in a system of communicating finite-state machines. *Applied Mathematics & Information Sciences*, 239-252. doi:10.18576/amis/130212

Yoshida, N., Zhou, F., & Ferreira, F. (2021). Communicating finite state machines and an extensible toolchain for multiparty session types. *Fundamentals of Computation Theory*, 18-35. doi:10.1007/978-3-030-86593-1_2.