International Journal of Innovative Computing, Information and Control
Volume **19**, Number **1**, February **2023**

ICIC International ©2023 ISSN 1349-4198
pp. **181–196**

DEPLOYMENT OF A HIGH-SPEED COMMUNICATION NETWORK TO ENABLE REAL-TIME CONTROL OF A LOWER LIMB ROBOTIC EXOSKELETON

Andres Fabricio Cordova[1], Hernan Morales[1], Fabian Astudillo-Salinas[1]
Huiyan Zhang[2] and Luis Ismael Minchala[1,*]

[1]Department of Electrical Engineering, Electronics and Telecommunications
University of Cuenca
Ave. 12 de Abril y Agustin Cueva, Cuenca 0101168, Ecuador
{ andresf.cordovac; hernan.morales; fabian.astudillos }@ucuenca.edu.ec
*Corresponding author: ismael.minchala@ucuenca.edu.ec

[2]National Research Base of Intelligent Manufacturing Service
Chongqing Technology and Business University
No. 19, Xuefu Avenue, Nan'an District, Chongqing 400067, P. R. China
huiyanzhang@ctbu.edu.cn

Received April 2022; revised August 2022

Abstract. *This paper presents a practical approach to deploying a real-time communication network applied to a hierarchical control architecture of a lower limb robotic exoskeleton. Previous experimental results of a communication network using Controller Area Network (CAN) protocol, which uses Service Data Objects (SDO) within frames of the CAN protocol, showed some disadvantages such as non-constant sampling time and data loss. These issues are completely solved by switching SDO objects to Process Data Objects (PDO) within the CAN protocol's frame and using a non-concurrent programming methodology for deploying the control system. Experimental results show high accuracy in the repetitiveness of the sampling time, data transmission, and high precision of open-kinematics position control of the lower limb exoskeleton prototype.*
Keywords: CAN network, Lower limb robotic exoskeleton, High-speed communications, PDO, SDO

1. **Introduction.** Modern medicine is highly influenced by robotic-assisted procedures as innovative tools to improve physical and cognitive treatments. For example, according to [1] in the United States, there are about 20.8 million people with limitations for walking or climbing stairs. In addition, the World Health Organization (WHO) estimates international new cases of Spinal Cord Injury (SCI) between 250,000 to 500,000 each year [2], which severely affects the life expectancy of patients [2,3]. Furthermore, forecast data on older adults' growth rate predicts that by 2050 there will be 1.5 billion elderly worldwide. Due to their age, these people could have some restrictions in their locomotor system [4]. Therefore, there is an increasing need for the development of robotic devices to treat pathologies like SCI and devices to facilitate the daily life of this vulnerable group of people. Among the possible robotic devices, there is a current trend of research and development of wearable robotics applied to medicine, such as robotic exoskeletons.

The employment of exoskeletons implies several clinical and physiological advantages. Firstly, using these devices in rehabilitation allows for replacing repetitive procedures with more exact processes. Additionally, the possibility of free locomotion contrasts with the

DOI: 10.24507/ijicic.19.01.181

pathologies produced by a sedentary lifestyle [5]. For instance, in [6] the authors have developed a walking training robot for physical rehabilitation; this device also has a fall detection feature that helps to avoid secondary damage to the users during physical therapy. Furthermore, certain robotic applications require a real-time operation. For example, the authors of [7] present a mobile robot that requires a system with real-time decision-making to perform various tasks. This involves that the characteristics of its control and communication systems must be designed for this purpose.

This paper presents improvements over the findings reported in [8] where the first version of a lower limb exoskeleton prototype is introduced. The primary motivation to enhance this initial version is to have a fully functional device that, in the future, can be used in rehabilitation sessions. To achieve this, the development of the exoskeleton must continue in both hardware and software, taking advantage of the experience and opportunities for improvement mentioned in [8]. For instance, one of the significant challenges was the loss of synchronization during data transmission due to an unstable and sometimes excessively high transmission time. This research tackles these difficulties by updating the Service Data Objects (SDO) with Process Data Objects (PDO). The use of PDO reduces the processing time in the central node and the channel since PDO avoids some processes such as confirmation to the central node and object identification within the CAN frame. This research showed a lack of methods or libraries for PDO transmission capable of fitting the requirements of a lower limb robotic exoskeleton; therefore, it was necessary to develop specific PDO communication methodologies. Experimental results showed a 5x improvement in processing time. In addition, it was possible to guarantee a constant transmission rate and an increase in the number of captured samples compared to the previous system.

The organization of this document is as follows. Section 2 presents a literature review of CAN communication networks applied to mechatronics systems with similar throughput requirements as the prototype tested in this research. Section 3 describes the hardware used in the exoskeleton. Section 4 focuses on details of the deployment of the communication system. Section 5 presents results obtained from experimental tests. Section 6 presents the conclusions of this work.

2. **Related Work.**

Table 1 presents the main findings of related works concerning communication systems that use CAN and CANopen protocols. Results reported in these articles show that the CAN and CANopen are suitable for several applications where high-speed and real-time communication is required. However, under the best judgment of the authors of this research, up to date, there was not a fully dedicated paper to the communications systems of a robotic exoskeleton, which is the main purpose of this work.

2.1. **Service Data Objects (SDO).**

SDO are communication objects of the CANopen protocol that allow communication between nodes in a CAN network. These objects allow access to a node's object dictionary to read or write information. They use a client/server type architecture, where the node whose dictionary is accessed acts as a server while the node requesting access is the client. Through SDO, it is possible to establish point-to-point communication. In addition, SDO use acknowledgement of data receipts by the client [17,18].

2.2. **Process Data Objects (PDO).**

These communication objects allow data transfer in real time through a producer/consumer model. In contrast with SDO objects, there is no need for acknowledgment. There are two types of PDO objects: transmission (TxPDO) used by producers and reception (RxPDO) used by consumers. There are no predefined

Table 1. Literature review

Authors	Title	Main objective	Main contribution	Results
Zhang et al. [9]	Implementation of CANopen Distributed Control Network Based on ARM in Automatic Production Line	Performance measurement of a CANopen network for industrial purposes	Experimentation to obtain the processing time of CANopen communication objects on slave node	The processing time of communications objects meets the requirements of automatic production lines.
Werewka and Jan [10]	Response-Time Analysis of a CAN Network Used for Supervisory Control and Diagnostic Systems	Implementation of a SCDS based on CAN for the Swiss Light Source accelerator control system to meet real time constraints	Introduction to the design and principles of SCDS construction together with the bases of time analysis for this type of systems	Theoretical and experimental data of the response-time obtained for different number of nodes in a CAN bus in the Swiss Light Source
Fan et al. [11]	Communication of the Wind Turbine Testing System Based on CANopen Protocol	Development of data communication of a wind turbine testing system with CANopen protocol	Presentation of the design criteria and development of a CAN-based communication algorithm for a wind turbine test system that allows high-speed and real-time communication	The CAN communication network operates properly, since the system provides a characteristic curve of the wind turbine measured in real-time that adjusts to the theoretical curve.
Xu and Dong [12]	The Design and Implementation of a CANopen Slave Stack for Powertrain Controller in Hybrid Electric Vehicle	Development of a slave stack based on CANopen to provide real-time communication for hybrid electric vehicles (HEV)	Consistency testing of the proposed stack to verify if it meets the requirements of the CiA 301 standard and the needs of HEV	The proposed slave stack complies with the major functions of CiA 301 standard and demonstrates that it has a potential application for hybrid electric vehicles.
Hung et al. [13]	Multi-Motor Synchronous Control with CANopen	Development of motors synchronous control based on specification CiA 301 and the motion device specification CiA 402	Theoretical comparison between SDO and PDO for synchronized control, calculation of the maximum number of nodes for SDO and PDO	The experimental results showed that the proposed CANopen synchronous control is workable.
Fu and Tong [14]	CANopen Message Real-Time Optimization Based on Hybrid Scheduling Method	Design of a new hybrid scheduling method (static-dynamic) for the CANopen protocol using PDO	This optimization method aims to reduce the probability of message collisions and optimize real-time performance.	Experiments carried out with and without this method indicated an evident reduction in the average response times for transmission PDO frames (TPDO).
Seoane et al. [15]	CAN Implementation and Performance for Raman Laser Spectrometer (RLS) Instrument on Exomars 2020 Mission	Development and testing of a CAN-based network together with CANopen to provide a unique data interface for Telecommands (TCs) and Telemetries (TMs)	Implementation of the CAN standard for space use together with the CANopen protocol to fulfill the needs of spacecraft data handling systems	The CANopen protocol has an appropriate performance for the scientific space instrument RLS. The proposed protocol uses PDO for short TCs and TMs, and SDO for long science data.
Minchala et al. [8]	Low Cost Lower Limb Exoskeleton for Assisting Gait Rehabilitation: Design and Evaluation	Design and implementation of 3 degrees of freedom (DoF) lower limb exoskeleton for rehabilitation purposes	Development of a lower limb exoskeleton (LLE) together with the control and communication algorithms for its operation	The LLE achieves a correct tracking of gait biomechanics trajectory; the overall performance of the LLE is good. However, there are mechanical & software issues to be solved.
Quan et al. [16]	Design and implementation of Multi-Axis Synchronous Motion Control System Based on CANopen	Development of a communication CAN network to achieve synchronous motion control for a hexapod robot	Implementation and evaluation of real-time performance of a communication network with one master and six distributed nodes, based on CANopen	The use of CAN together with CANopen in robotic systems reduces response time and meets real-time requirements for synchronous distributed control.

frame communication formats, and it depends on the application's needs under development. For this purpose, it is necessary to perform a procedure called PDO mapping [17,18], which consists of indicating which communication object is to be located at a specific address of the CAN frame. Thus, by placing data in that address, the network nodes will interpret this data as the values of the mapped object. Figure 1 shows how data from SDO frames is mapped into PDO frames. The data initially sent in the first SDO

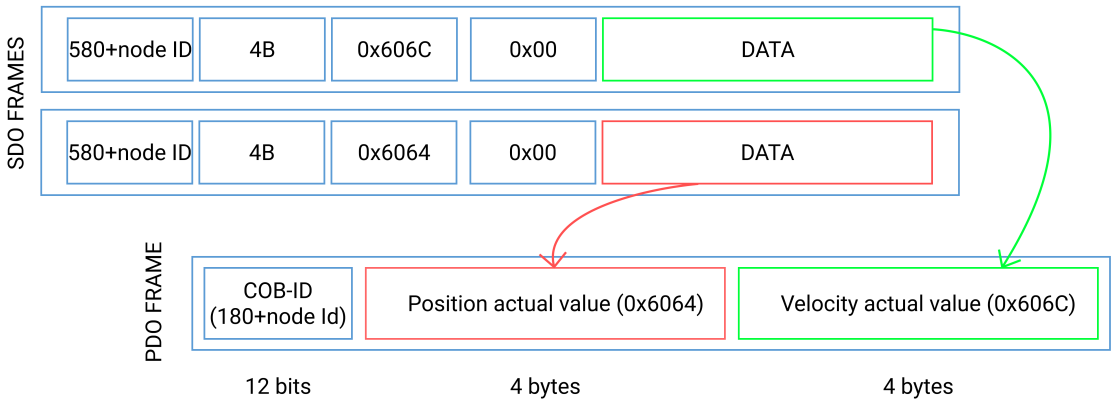


FIGURE 1. PDO mapping process

frame data section (velocity actual value) is taken and placed in the last 4 bytes of the new PDO frame. A similar situation happens for the second SDO frame data (position actual value), which becomes the first 4 bytes of the PDO frame. After this process, devices receiving or transmitting data in the PDO frame will interpret the first 4 bytes as position and the last 4 bytes as velocity.

PDO objects perform synchronous and asynchronous communications. The type of communication can be configured according to the user's needs, but generally, the choice is made based on operation modes allowed by the primary device. Synchronous communication requires the transmission of an SYNC object.

3. **Lower Limb Exoskeleton Prototype ALLEX-2.** The Autonomous Lower Limb Exoskeleton version 2 (ALLEX-2) employs an Easy Positioning System (EPOS4) 50/8 distributed controller (manufactured by Maxon Motor) for each of its six joints. Each EPOS4 attaches to a CAN bus for the communication system. These components are the same as those used in ALLEX-1 [8]. The central controller responsible for the control and communication algorithms is a Raspberry Pi 4B (RB-4) microcomputer with better features than the version used in ALLEX-1. A mechanical update has also been deployed, implying more robust limbs and harmonic drives in the joints. Figure 2 shows the communication architecture of ALLEX-2.

Figure 3 shows the gait biomechanics trajectories used as a reference for motion control. These are based on the signals presented in [19]. For example, right leg trajectories from Figure 3(b) are obtained by shifting the left leg signals a 50% of the cycle.

3.1. **Master node/central controller architecture.** The controller node of the prototype is a Raspberry Pi 4B, which has a 64-bit quad-core processor at 1.5 GHz and 4 GB of RAM. Raspberry Pi 4B operates with open-source software and works with Raspbian Buster as an operating system [20]. This device is responsible for processing the control and communication algorithms. For this purpose, the RB-4 handles the data sent to the distributed controllers (secondary nodes) and the measurements obtained during their operation. The data sent to each distributed node corresponds to the angular biomechanics of the articulations, and the received data in the central node is each distributed node's position, speed, torque, and current.

For the physical implementation of the CAN bus, the central node uses a PiCAN2 module that establishes the interface for communication with each distributed controller. This interface is compatible with CAN2.0 and supports communications up to 1 Mb/s [21].

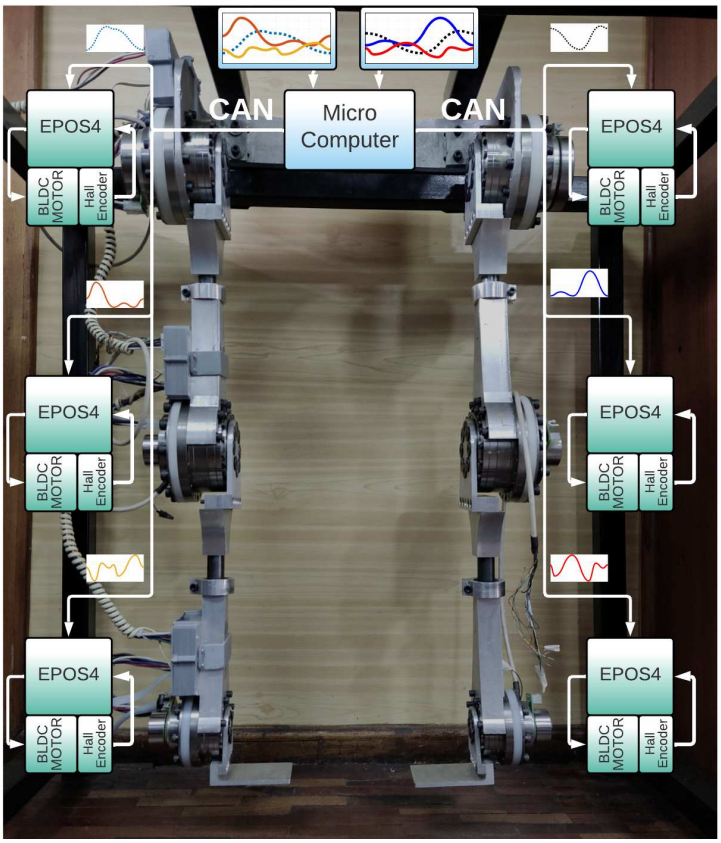
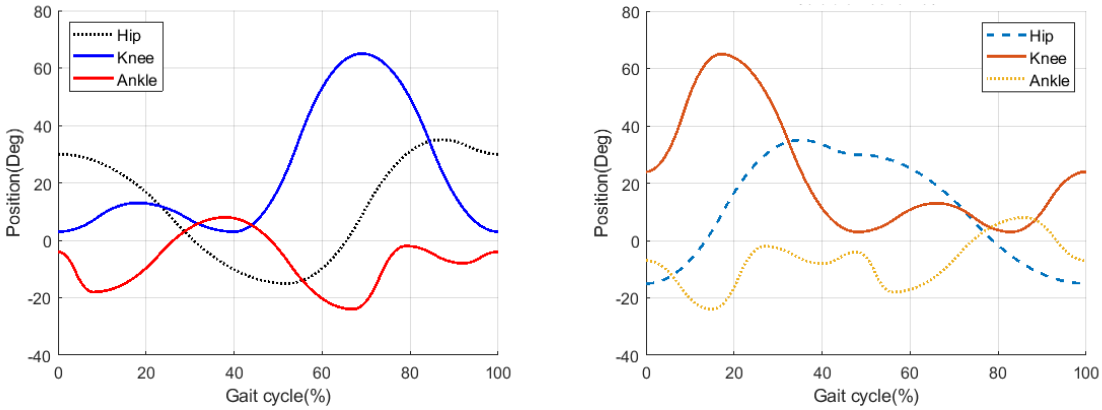


FIGURE 2. ALLEX-2 front view, control architecture



(a) Left lower limb biomechanics

(b) Right lower limb biomechanics

FIGURE 3. Non-pathologic gait biomechanics

3.2. **Distributed controller architecture.** The distributed nodes are the EPOS4 50/8 position controllers, which allow working with direct current motors of up to 400 W both with brushes (brushed DC motors) and without brushes (brushless EC motors, BLDC) [22]. In addition, these drivers have, among others, the following features:

- CAN and EtherCAT communication interfaces;
- Five operating modes, which involves speed, position, and torque control [22];
- Three control loops, namely: current (proportional-integral, PI), position (proportional-integral-derivative, PID), and speed (proportional-integral, PI).

EPOS4 devices allow executing joint movements in different modes, such as Profile Position Mode (PPM) and Cyclic Synchronous Velocity (CSV). Each EPOS4 calculates the necessary path to reach a reference in PPM mode. Unlike the PPM mode, the CSV mode requires a prior trajectory calculation. This operating mode requires the trajectory samples to be used as a reference; these samples must be sent periodically (the period between sample and sample is the interpolation time). The PPM mode works with SDO or PDO communication, while CSV works only with PDO. Table 2 presents the communication parameters or objects needed for PPM and CSV mode [22].

Table 2. PPM and CSV mode command parameters

PPM	CSV
Controlword	Target velocity
Target position	Velocity offset
Profile velocity	
Profile acceleration	
Motion profile type	

4. **Deployment of the Communications System.**

4.1. **Network topology.** The communication protocol is CAN, so the topology used is bus type. Seven nodes are attached to this bus: one master node and six slave nodes. Figure 4 shows the topology with nodes six and three at both ends of the channel. The system operates at the highest allowed rate, 1 Mbps, possibly because the bus length is less than 25 m [23].

The design of the communication system implies switching among two architectures:

1) Client-server, for network management;
2) Master-slave, for the joint moving during the gait cycle.

The primary node RB-4 performs fundamental processing tasks, which implies network management and trajectory tracking of the joints. This node operates as a master and server. The EPOS4 50/8 drivers are configured with numbers 1 to 6, corresponding to secondary nodes' identifiers.

4.2. **Software implemented in the exoskeleton, comparison between SDO and PDO approaches.** In previous research [8], the authors presented a solution for CAN communication using PPM mode through SDO (PPM-SDO) applied to a three-joint exoskeleton. The PPM-SDO configuration employed a concurrent approach for the movement and reading processes, i.e., the four processes occurred in parallel. This operation implied sending three SDO frames in the reading frame of each joint. These frames allowed capturing the output parameters (position, velocity, and current). As the authors of [8] mention, this approach presented synchronization problems in addition to a variable sampling rate. Figure 5 shows a comparison of the process flowcharts, where Figure 5(a) corresponds to the described solution.

On the other hand, this research uses a methodology with the CSV mode, PDO frames, and a structured software approach to tackle the issues reported in [8]. Figure 5(b) shows the flowchart generated with these changes. The reading and writing processes run sequentially and within a single loop. This operation implies sending a single PDO frame with target velocity and offset to move a joint. In the reading process, the system sends the SYNC object, and then each EPOS4 responds with two frames: position, speed, current, and torque. As shown in the results section, this approach has the following advantages:

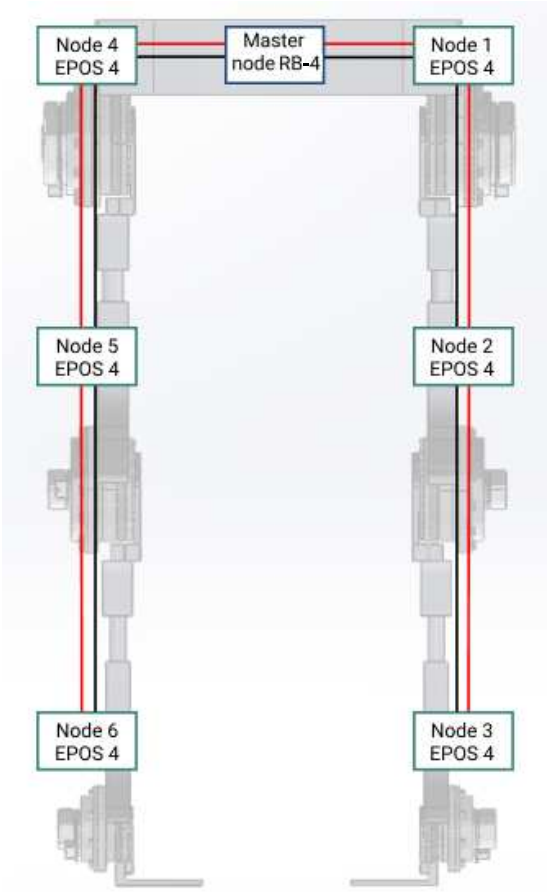


FIGURE 4. CAN network topology

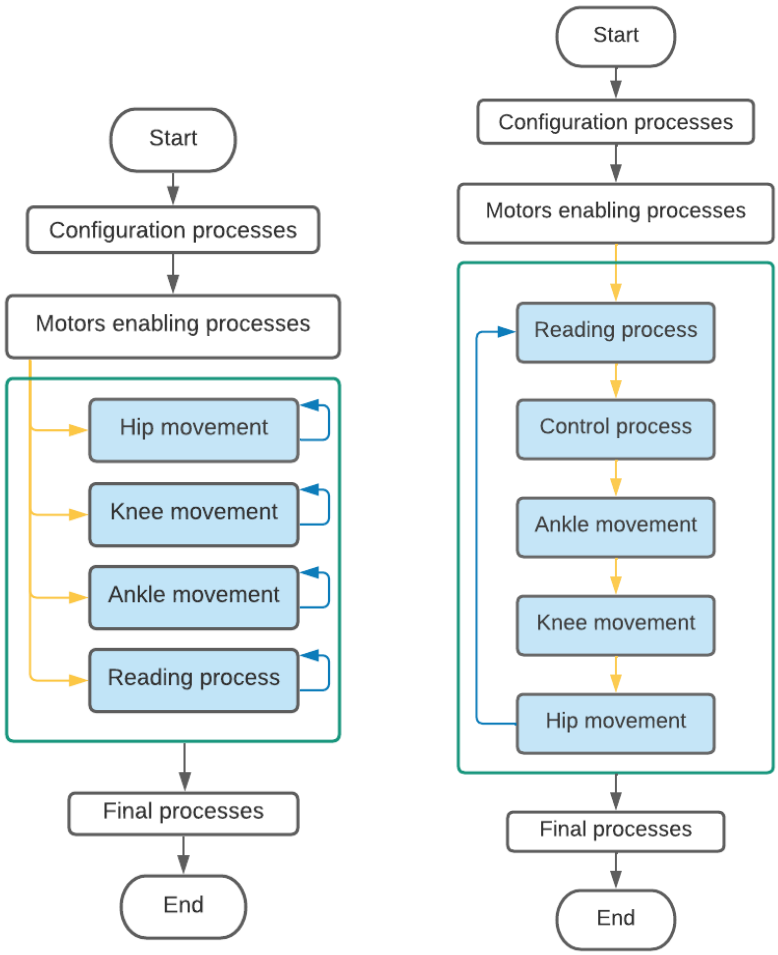
the possibility of using the readings for control processes, providing a fixed sampling rate, reducing the reading and writing processing time, and decreasing the channel time.

4.3. **PDO communication setup for CSV mode.** Figure 6 shows the PDO mapping established in the distributed controllers. The first RxPDO allows commanding the CSV movement, and the second carries the interpolation time. These objects work asynchronously. The system states or output parameters (velocity, position, torque, and current) can be read through the TxPDOs configured. Unlike the RxPDO, TxPDOs work synchronously.

4.4. **Software implementation details, CSV mode and PDO approach.** Figure 7 shows the algorithm implemented. The tasks can be grouped as follows.

4.4.1. *Writing or CAN frames sending processes.* This first task contains all the sending processes; the primary node sends instructions to the network. These instructions configure the interpolation time, commands, or PDO motion frames. When the secondary nodes receive the interpolation time (20 ms), they await PDO commands. If the interpolation time equals zero, the nodes no longer wait for a command. After the math process for control, the system sends the motion or command frames. As mentioned in the mapping, these frames contain the velocity and the velocity offset. In Figure 7 these processes are shown in red.

4.4.2. *CAN frame reading processes.* The reading processes (highlighted in orange in Figure 7) allow capturing the frames sent by the distributed nodes. The first step in this



(a) General view of the software used in former research [8]

(b) General view of the software implemented in this work

FIGURE 5. Software implemented in ALLEX-1 [8] and ALLEX-2 (this work)

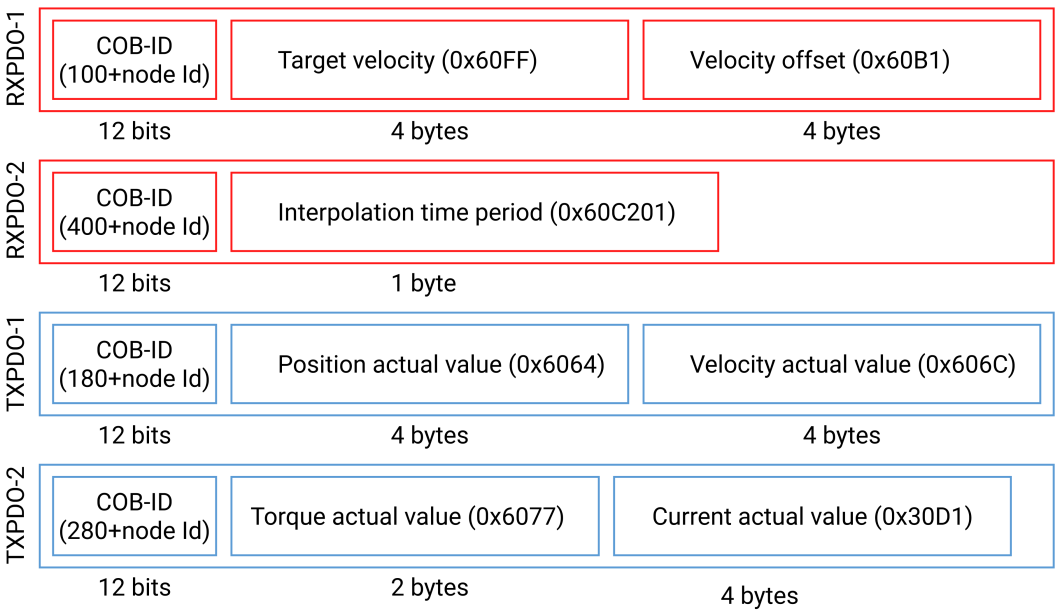


FIGURE 6. PDO mapping for CSV mode configured in each EPOS4

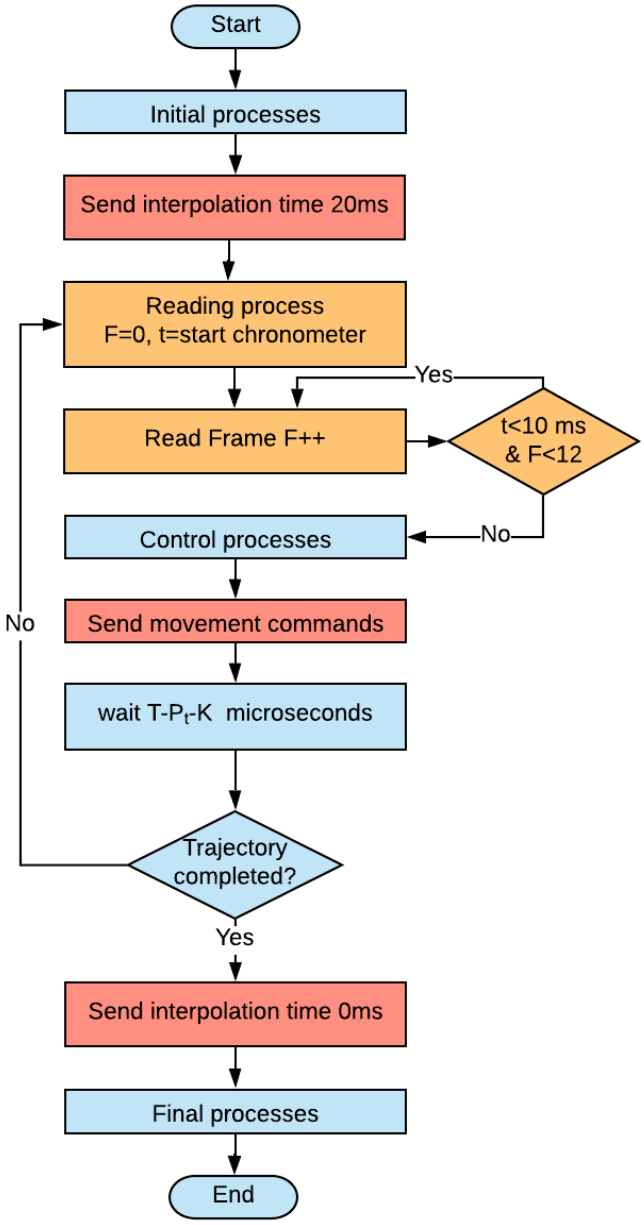


FIGURE 7. (color online) Communication algorithm flowchart

process is to request all nodes in the network to send TxPDO frames; for this purpose, the system uses the SYNC object. Next, the primary node waits until collecting all the 12 frames or until the time slot for reading (10 ms) runs out; after this, the algorithm continues.

Both the reading and writing processes run until completing one gait cycle. The sampling time for the gait trajectories is 20 ms. In this work, the gait cycle lasts 8 s; thus, there are 400 samples, the number of the software's loop iterations.

4.4.3. *Secondary processes.* These include control processes, dynamic waiting, initial movement, the final movement, and data storage. In the context of this report, control processes are secondary. The dynamic waiting process ensures the system sends samples every $T = 20$ ms. This process measures the processing time of the previous tasks (P_t) and calculates the time the algorithm must wait to complete the 20 ms. Equation (1) shows

the calculation performed, where K is an experimentally determined tuning constant, $K = 132$ ms. It compensates for the processing times that cannot be measured (such as the execution of the equation itself). The constant was obtained through the trial and error method. A greater or lower value of $K = 132$ ms produces a more considerable loop error even when K increases in a unit.

$$waiting\_time = T - P_t - K \tag{1}$$

The initial movement refers to the series of actions necessary to bring the exoskeleton to the starting position. This process is similar to the gait cycle but applied to a short path without a reading process. Here, trajectories from zero to Θ_0 are used for each joint with a duration of two seconds. The final movement employs inverted trajectories (from Θ_0 to 0) to move the joints to the resting position.

The storage process and the last movement are part of the final secondary processes; volatile memory stores data from the reading process until the end of the cycle. After concluding the gait cycle, the system saves the data in a comma-separated file. Furthermore, this process stores the execution time measurements for the reading and writing process (one at a time). These measurements do not affect the flow of the algorithm and are used to obtain the data analyzed in the results section.

5. **Results.**

5.1. **Performance comparison between PDO and SDO communication approach.**

The experimental results presented throughout this section imply reproducing the experimentation protocol used by authors of [8]. The primary purpose is to compare the performance of the communication systems when using either SDO or PDO.

Table 3 shows two data sets representing writing times during communications using SDO and PDO, respectively. The channel time is measured by using the software *Wireshark*. The channel time corresponding to PDO frames is an estimation since these packages are unidirectional. The estimation is made through *sent by us* and *broadcast* packets captured by *Wireshark* that contained the same information, but different sources. Therefore, the master node generates the *sent by us* packet, and receives the *broadcast* packet. The acknowledgment option accurately measures channel time for SDO frames [8]. As can be seen, the channel time for PDO is reduced to less than half of the time for SDO.

Table 3. Comparative table for writing times

	SDO	PDO
Channel time (ms)	0.6365	0.2597
Processing time (ms)	32.1848	0.3275

Another experimental result corresponds to the processing time measured by an embedded timer in the code. The processing time for PDO frames is calculated from 9000 writing processes. The most significant improvement of the communication system with the approach presented in this work in comparison with [8] is the processing time, which is for the PDO frames, approximately ten times less than the time it takes for the processor to execute the same task using SDO frames. This is because SDO uses more frames to send motion commands to the exoskeleton while PDO employs a single RxPDO frame.

It is worth mentioning that the implemented stopwatch aims to be minimally invasive and not affect the measurement. This is achieved using a structure with constructor and destructor methods from C++ [24]. In this way, it is only necessary to initialize the "timer"

object, and when it is destroyed, the object gives the measurement of the execution time of the interest process.

Table 4 shows both processing and channel times for reading data from a single joint. SDO times are the averages presented in [8]. PDO reading times are obtained using *Wireshark* for measuring and averaging 2000 samples of the channel's time and a timer within the program to obtain the processing time. Once the system sends the request frame (SYNC for PDO and request frames for SDO), the measurement starts and ends with the reception of the response from the node (TxPDO 1 and frame with the data for SDO). These times show that PDO is processed 4.72 times faster than SDO reading time. Additionally, the time on the channel decreases by 0.3126 ms.

Table 4. Comparative reading times table

	SDO	PDO
Channel time (ms)	0.8471	0.5345
Processing time (ms)	2.9486	0.6242

5.2. **Sampling by PDO.** An essential difference between the SDO approach used in [8] and the PDO communication presented in this work corresponds to the captured samples. The former approach does not guarantee a fixed sampling rate, i.e., in each gait cycle, this system captures a different sample number. On the other hand, in this work, the system has a constant sample rate of 20 ms; this allows an equal number of samples in each gait cycle. This constant rate is mainly due to the algorithm's structured implementation approach. Figure 8 shows that the number of samples captured with PDO frames is greater than the system with SDO and also is proportional to the walking time.

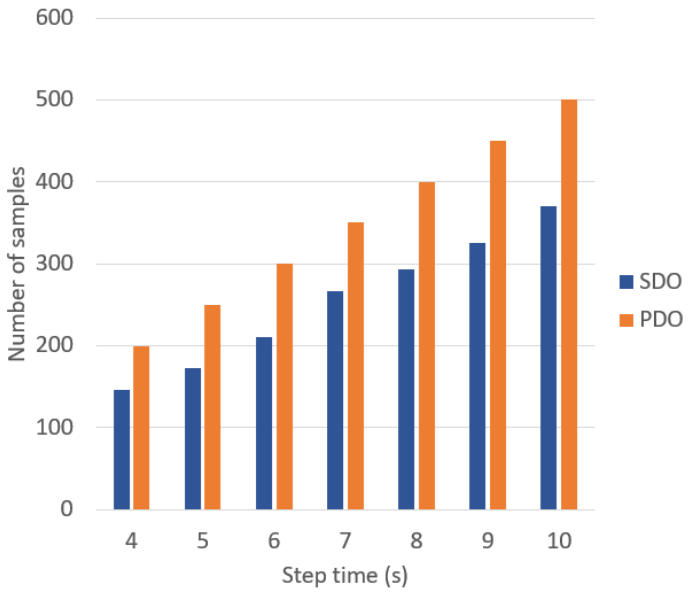


Figure 8. Samples number for different walking time with SDO or PDO

Although sampling using PDO and a deterministic programming approach gives excellent results regarding the sampling time, the non-ideal characteristics of the central node processor cause a variation in the sampling rate. This error occurs in the dynamic waiting process, where an exact trade-off to complete one loop iteration in 20 ms is not always achieved. Two hundred and twenty experiments were carried out to analyze this

behavior. The experiment's objective was to measure the gait cycle execution time; this should be 8 s for each cycle. The results show that the average cycle error is 1.266 ms with a standard deviation of 1.3345 ms.

Figure 9 shows the histogram of the collected data along with the Normal Inverse Gaussian distribution NIG (1.4281,1.3075, 0.2674,0.4389), which is the best fit. This distribution has a confidence interval of 95% between 0.01634 ms and 5.2386 ms. Therefore, in the worst case, each sample in the cycle will be sent in 20,013 ms instead of 20 ms.

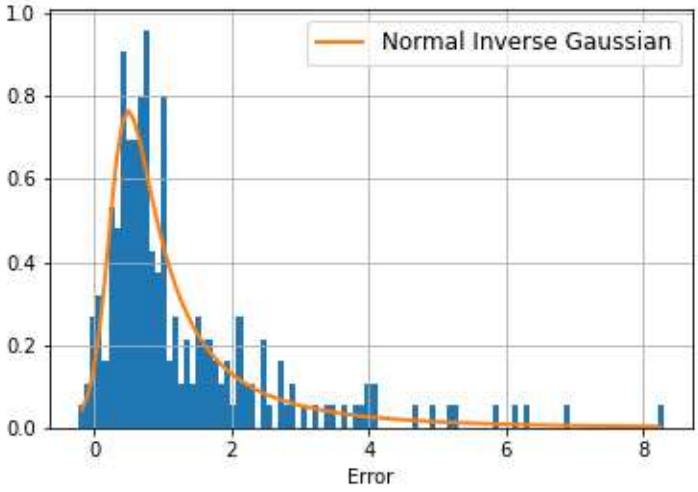


FIGURE 9. Dynamic waiting process error probability distribution function

5.3. **Performance of the PDO communication system.** Table 5 shows results obtained by measuring the processing time of the PDO writing and reading methods. The reading process experiment implied capturing six frames and measuring 1600 reading events. The processing times of 9000 CAN frame transmissions were measured for the writing process.

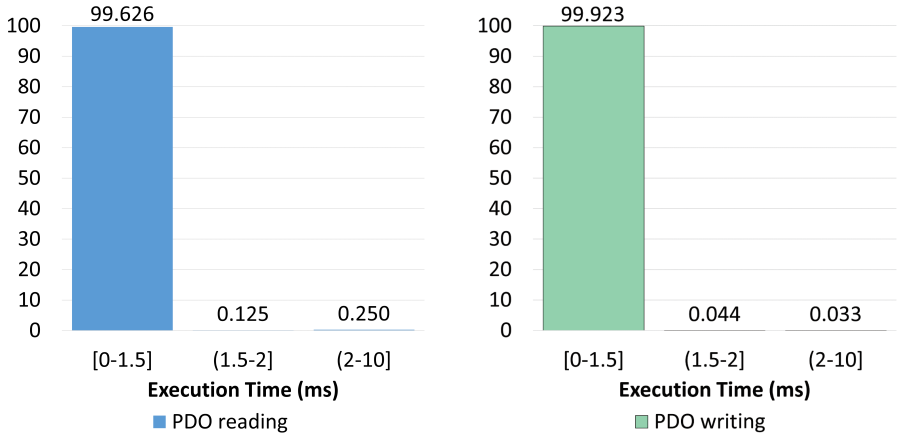
TABLE 5. Statistics of the processing time of the CAN PDO reading/writing process

	Reading (ms)	Writing (ms)
Mean	1.1711	0.3275
Standard deviation	0.0912	0.1492
Max	2.68297	8.5573
Min	1.08147	0.2258

The maximum values presented in Table 5 hardly occur. Figure 10(a) shows that of all the captured reading process times, only 0.375% exceeds 1.5 ms. Furthermore, Figure 10(b) shows that only 0.077% of writing is between 1.5 and 10 ms. There were no abnormal data greater than 10 ms.

5.4. **Trajectory tracking.** The benefits of the distributed control system are shown through two additional experiments:

1) Prototype movement execution of a complete gait cycle without a load;
2) Prototype movement execution of a complete gait cycle with a 12 kg load, of which the hip joint moves all the load, the knee joint 3.5 kg, and the ankle joint 1 kg. These weights are based on the anatomical percentages presented in [25].



(a) Reading process (b) Writing process

FIGURE 10. Execution time confidence intervals for reading and writing processes

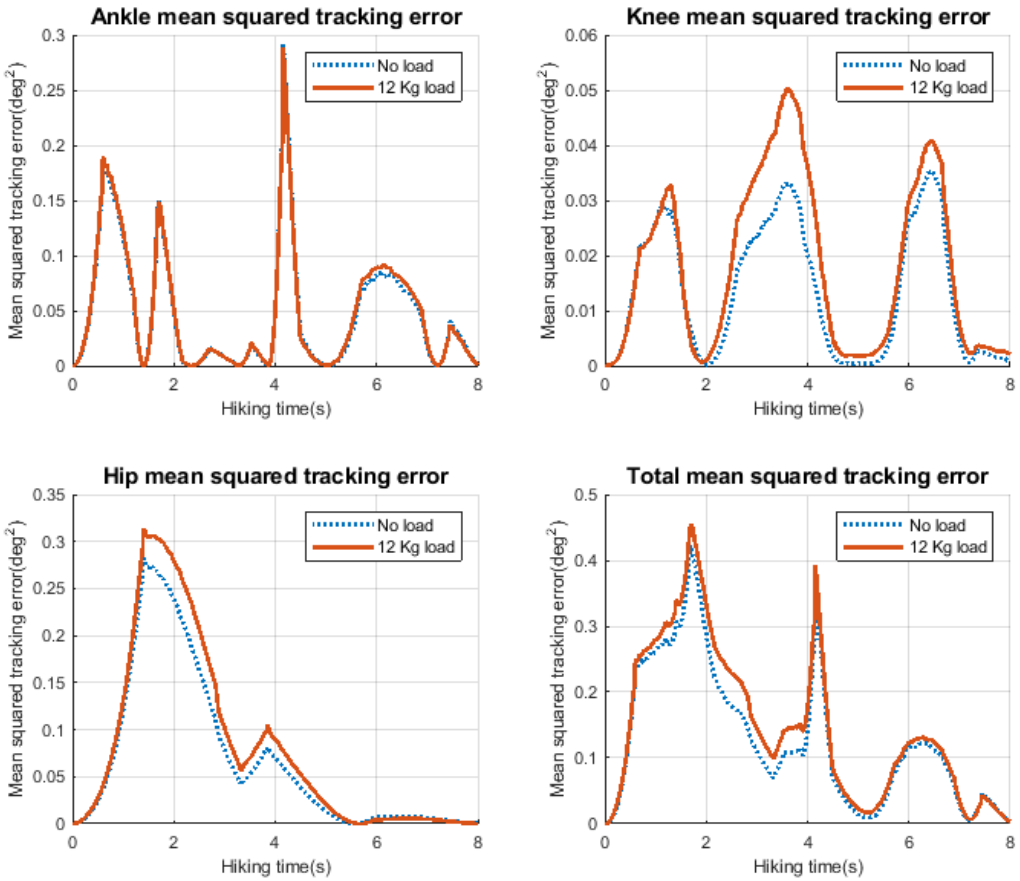


FIGURE 11. Joints tracking error during a walk cycle, right lower limb

Figure 11 shows the mean squared tracking error for one gait cycle with and without load. These results show the correct trajectory tracking by the distributed nodes. It can be seen that the error increases when the prototype carries a load; however, this increase is minimal. Moreover, this figure shows that the root-mean-square tracking error is low. In the context of the communication system, this result proves that the data transmission

is carried out properly, and there are no problems that lead to a malfunction of the control system due to poor data transmission between nodes.

Table 6 shows a statistical description of the tracking result. It can be seen that the greatest growth of the error is in the knee, where on average, the mean squared error (MSE) grows by 33% in comparison with the tracking error of the unloaded experiment, unlike the 3% and 18% of the ankle and hip, respectively. This behavior is also observed in the maximum MSE. In the knee, it increases by 42%. The minimum error for all joints was zero. Likewise, this table indicates that the values of the mean of the MSE remain low and stable according to the standard deviation. The maximum errors do not exceed 0.3130 Deg^2. These results could not be obtained without correct synchronization between the primary and distributed nodes.

TABLE 6. Mean squared error descriptive statistics, with and without load

Joint	Mean		Std		Max	
	No load	12 kg load	No load	12 kg load	No load	12 kg load
Ankle	0.0470	0.0487	0.0540	0.0549	0.2923	0.2895
Knee	0.0134	0.0179	0.0119	0.0155	0.0352	0.0503
Hip	0.0685	0.0814	0.0839	0.0974	0.2824	0.3130

6. **Conclusions.** Prior results of the communications system of [8] showed a non-constant sampling rate, which was solved through a structured programming approach and a dynamic waiting time, leaving aside the concurrent programming methodology. This solution increases the number of captured samples and a stable sampling rate. These improvements correspond mainly to the benefits of switching the SDO to PDO in the CAN communication protocol.

The performance of the PDO communication systems presented almost negligible errors due to the tolerance of the distributed controllers regarding the variation of the interpolation time. Therefore, the employed processor has the appropriate features for this application.

It is worth mentioning that the results shown in this work regarding the programming approach do not imply a complete exclusion of concurrent programming, and it is only necessary that the reading and writing process of all the articulations are within the same process. This way, threads can be used to schedule child processes, such as a graphical interface.

Acknowledgment. The authors are grateful for the support provided in this research by the Research Direction of the University of Cuenca, under the financing of the project: "Robotic exoskeleton for functional assistance in walking patients with incomplete spinal cord injuries: design and initial application".

REFERENCES

[1] *Homepage | Annual Disability Statistics Compendium*, https://disabilitycompendium.org/, Accessed in January 2022.

[2] WHO, *International Perspectives on Spinal Cord Injury*, J. Bickenbach (ed.), WHO Press, Malta, 2013.

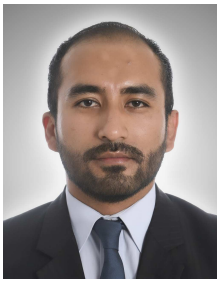
[3] A. Ferrari, M. G. Benedetti, E. Pavan, C. Frigo, D. Bettinelli, M. Rabuffetti, P. Crenna and A. Leardini, Quantitative comparison of five current protocols in gait analysis, *Gait and Posture*, vol.28, no.2, pp.207-216, 2008.

[4] C. Cooper, G. Campion and L. J. Melton, Hip fractures in the elderly: A world-wide projection, *Osteoporosis International*, vol.2, no.6, pp.285-289, 1992.

[5] F. Ferrati, R. Bortoletto, E. Menegatti and E. Pagello, Socio-economic impact of medical lower-limb Exoskeletons, *2013 IEEE Workshop on Advanced Robotics and Its Social Impacts*, pp.19-26, 2013.
[6] Y. Wang, W. Xiong, J. Yang and S. Wang, A new fall detection method based on fuzzy reasoning for an omni-directional walking training robot, *International Journal of Innovative Computing, Information and Control*, vol.16, no.2, pp.597-608, 2020.
[7] J. Xu, L. Wang, Q. Kou, T. Fang, D. You, L. Zhou and Y. Zhang, Real-time behavior decision of mobile robot based on the deliberate/reactive architecture, *International Journal of Innovative Computing, Information and Control*, vol.18, no.4, pp.1163-1180, 2022.
[8] L. I. Minchala, A. J. Velasco, J. M. Blandin, F. Astudillo-Salinas and A. Vazquez-Rodas, Low cost lower limb exoskeleton for assisting gait rehabilitation: Design and evaluation, *ACM International Conference Proceeding Series*, 2019.
[9] J. Zhang, B. Chen and X. Zou, Implementation of CANopen distributed control network based on ARM in automatic production line, *Advanced Materials Research*, vols.139-141, pp.2217-2220, https://www.scientific.net/AMR.139-141.2217, 2010.
[10] J. Werewka and M. Jan, Response-time analysis of a CAN network used for supervisory control and diagnostic systems, *Control and Cybernetics*, vol.39, no.4, pp.1135-1157, http://eudml.org/doc/209739, 2010.
[11] Y. Fan, R. Chen and Z. Chen, Communication of the wind turbine testing system based on CANopen protocol, *2011 IEEE Power Engineering and Automation Conference*, vol.1, pp.162-165, 2011.
[12] Z. Xu and S. Dong, The design and implementation of a CANopen slave stack for powertrain controller in hybrid electric vehicle, *2010 International Conference on Intelligent Computation Technology and Automation*, vol.3, pp.755-758, 2010.
[13] C.-W. Hung, R. CL Lee, B.-K. Huang and S.-T. Yu, Multi-motor synchronous control with CANopen, *Proc. of International Conference on Artificial Life and Robotics*, vol.24, pp.37-40, 2019.
[14] L. Fu and G. Tong, CANopen message real-time optimization based on hybrid scheduling method, *Proc. of the 2015 International Industrial Informatics and Computer Engineering Conference*, vol.12, pp.585-588, 2015.
[15] L. Seoane, C. Diaz, J. Zafra, S. Ibarmia, C. Quintana, C. Perez, A. Moral and A. Araujo, CAN implementation and performance for Raman Laser Spectrometer (RLS) instrument on ExoMars 2020 Mission, *IEEE Transactions on Emerging Topics in Computing*, vol.9, no.1, pp.67-77, 2018.
[16] Y. Quan, B. Li, W. Wang, Z. He and S. Zhang, Design and implementation of multi-axis synchronous motion control system based on CANopen, *2020 5th International Conference on Control, Robotics and Cybernetics (CRC2020)*, pp.240-244, 2020.
[17] *CANopen Application Layer and Communication Profile CAN in Automations*, CiA 301 Version 4.2.0, CAN in Automation E.V., Nuremberg, Germany, 2011.
[18] *EPOS4 Communication Guide*, Maxon Motor, Sachseln, CH, Switzerland, 2019, https://www.maxongroup.com/maxon/view/product/control/Positionierung/EPOS-4/504384, Accessed on November 9, 2021.
[19] C. Y. Ko, J. W. Ko, H. J. Kim and D. Lim, New wearable exoskeleton for gait rehabilitation assistance integrated with mobility system, *International Journal of Precision Engineering and Manufacturing*, vol.17, no.7, pp.957-964, 2016.
[20] *Raspberry Pi 4 Model B Specifications – Raspberry Pi*, https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/, Accessed in May 2022.
[21] *PiCAN2 – Controller Area Network (CAN) Bus Interface for Raspberry Pi 2*, https://copperhilltech.com/pican-2-can-bus-interface-for-raspberry-pi/, Accessed in June 2022.
[22] *EPOS4 Firmware Specification*, Maxon Motor, Sachseln, CH, Switzerland, 2019, https://www.maxongroup.com/maxon/view/product/control/Positionierung/EPOS-4/504384, Accessed on July 25, 2021.
[23] CIA, *CAN in Automation (CiA): CAN Knowledge*, https://www.can-cia.org/can-knowledge, Accessed in June 2022.
[24] S. Sarangi, *Constructor and Destructor in C++*, https://www.scaler.com/topics/constructor-and-destructor-in-cpp/, 2021.
[25] P. de Leva, Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters, *Journal of Biomechanics*, vol.29, no.9, pp.1223-1230, https://www.sciencedirect.com/science/article/pii/0021929095001786, 1996.

Author Biography

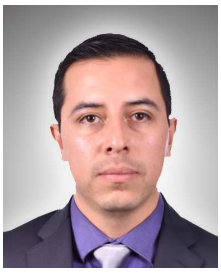
Andres Fabricio Cordova received the B.Sc. degree in Electronics and Telecommunications Engineering from University of Cuenca, Cuenca, Ecuador, in 2021. He was member of the Innovation and Technological Research Group of the University of Cuenca, Cuenca, Ecuador. He also worked as a researcher in the Department of Electrical Engineering, Electronics and Telecommunications, University of Cuenca, Cuenca, Ecuador. His research interests are robotics, automation, and computer science.



Hernan Morales received the B.Sc. degree in Electronics and Telecommunications Engineering from University of Cuenca, Cuenca, Ecuador, in 2021. He was member of the Vanguardia Honours Program (Fifth Cohort); this program was organized by the University of Cuenca together with KU Leuven to prepare the future generation of researchers. He worked as a researcher in the Department of Electrical Engineering, Electronics and Telecommunications, University of Cuenca, Cuenca, Ecuador. His main research interests include control methods applied to robotics and wireless communication.

Fabian Astudillo Salinas received the B.S.E. degree from Universidad de Cuenca, Cuenca, Ecuador, in 2007, and the M.S. and Ph.D. degrees from the Institut National Polytechnique de Toulouse, Toulouse, France, in 2009 and 2013, respectively. Since 2013, he has been a full-time researcher with the Department of Electrical Engineering, Electronics and Telecommunications, Universidad de Cuenca, Cuenca, Ecuador. His research interests include network coding, wireless sensor networks, vehicular networks, networked control systems, simulation of networks, performance of networks, cybersecurity and HPC.

Huiyan Zhang received the M.Sc. degree in Control Engineering and the Ph.D. degree in Control Theory and Control Engineering from the Harbin Institute of Technology, Harbin, in September 2014 and April 2019, respectively. From September 2015 to September 2017, she was a joint training Ph.D. student with the School of Electrical and Electronic Engineering, The University of Adelaide. She is currently an Associate Professor with Chongqing Technology and Business University. Her research interests include stochastic switched systems, event-triggered scheme, model reduction, balanced truncation, robust control, and filtering design.



Luis Ismael Minchala received a B.S.E.E. degree in Electronics from the Salesian Polytechnic University, Cuenca, Ecuador, in 2006, and the M.S. and Ph.D. degrees in Control Engineering from the Tecnológico de Monterrey, Monterrey, Mexico, in 2011 and 2014, respectively. From 2012 to 2013, he was a Visiting Scholar at Concordia University, Montreal, QC, Canada. Between 2017 and 2018, he was a Postdoctoral Fellow with the Climate Change Research Group, Tecnológico de Monterrey. Currently, he is a full-time researcher with the Department of Electrical Engineering, Electronics and Telecommunications, Universidad de Cuenca, Ecuador. He has authored and co-authored more than 60 indexed publications, including journal articles, conference proceedings, book chapters, and a book. His research interests are applied control engineering, renewable energies, wind turbine control, and robotics.