Research Article

Implementation and performance evaluation of a library to improve the communication with Maxon Motor's

Implementación y evaluación de rendimiento de una librería para mejorar la comunicación con los motores de Maxon

Hernan Morales, Andres Cordova, Ismael Minchala, Fabian Astudillo-Salinas*

Departamento de Eléctrica, Electrónica y Telecomunicaciones, Facultad de Ingeniería, Universidad de Cuenca, Cuenca, Ecuador, 010103; hernan.morales@ucuenca.edu.ec, andresf.cordovac@ucuenca.edu.ec, ismael.minchala@ucuenca.edu.ec.

*Correspondencia: fabian.astudillos@ucuenca.edu.ec

Citación: Morales, H., Cordova, A., Minchala, I., & Astudillo-Salinas, F., (2023). Implementation and performance evaluation of a library to improve the communication with Maxon Motor's. *Novasinergia. 6*(1). 95-104.

https://doi.org/10.37135/ns.01.11.06

Recibido: 09 diciembre 2022
Aceptado: 14 enero 2023
Publicación: 16 enero 2023

Novasinergia
ISSN: 2631-2654

Abstract: The development of robotic devices like exoskeletons involves using several components, such as actuators and sensors. These components exchange information with the main device or between them using a communication system; this allows data transfer during the device's operation. In addition, a communication system with good performance is the basis for the correct operation of the control system. In this context, selecting an appropriate methodology to meet design specifications during software development is necessary. This research presents the development of ALLEX CAN software, so named because it is part of the communication system of the ALLEX-2 (ALLEX version 2) prototype and uses the Controller Area Network (CAN) protocol. The system was created using the SocketCAN library as a base, which provides a range of tools to work with CAN interfaces. C++ was the programming language selected to develop this solution for its lower execution time. The performance of this software is compared with the ready-to-use functions provided by the manufacturer Maxon Motor. Experimental tests show the superior performance of our developed software.

Keywords: CAN protocol, communication system, Maxon Motor, software.

Resumen: *El desarrollo de dispositivos robóticos como los exoesqueletos implica el uso de varios componentes como actuadores y sensores. Estos componentes intercambian información con el dispositivo principal o entre ellos mediante un sistema de comunicación, esto permite la transferencia de datos durante el funcionamiento del dispositivo. Además, un sistema de comunicación con buen desempeño es la base para el correcto funcionamiento del sistema de control. En este contexto, es necesario seleccionar una metodología apropiada para cumplir con las especificaciones de diseño durante el desarrollo del software. Esta investigación presenta el desarrollo del software ALLEX (Autonomous Lower Limb Exoskeleton) CAN (Controller Area Network) software, llamado así porque forma parte del sistema de comunicación del prototipo ALLEX-2 (ALLEX versión 2) y usa el protocolo CAN. El sistema fue creado usando la librería SocketCAN por la gran cantidad de herramientas. C++ fue el lenguaje de programación elegido para el desarrollo de esta solución debido a su bajo tiempo de ejecución. El desempeño de este software se lo compara con las funciones proporcionadas por el fabricante Maxon Motor. Las pruebas experimentales muestran el rendimiento superior del software desarrollado frente al proporcionado por el fabricante.*

Palabras claves*: Protocolo CAN, sistema de comunicación, motor Maxon, software*

1. Introduction

The CAN protocol is widely used in several applications, from industrial, medical, and robotics (Di Natale et al. 2012; Seoane et al. 2021). The CAN topology is bus-type and provides a high-speed serial interface of up to 1 Mbit/s. This protocol allows sending information under the multi-master and peer-to-peer configuration. CAN is a low-layer protocol, so it is generally used with the CANopen protocol to add higher-layer functions (CAN in Automation (CiA) e. V. 2011). The popularity of the CAN protocol is mainly because of its: simplicity, reliability, low cost, and real-time performance (Bosch 1991; Farsi, Ratcliff, and Barbosa 1999).

Currently, some manufacturers provide ready-to-use communications protocols for diverse applications; for example, Maxon Motor produces high-precision motors, gear heads, and controllers for different purposes (Maxon 2022) and provides software with different functions for CAN communication. The ready-to-use software is great for speeding up the development process and reducing implementation time. However, some applications demand specific requirements from the control and communications system. In this case, a better approach is to develop the software according to the application's needs.

Several up-to-date research papers focus on exoskeleton development and use hardware from Maxon Motors and CAN protocol for communications. The authors of (Lu et al. 2014) developed a single-leg exoskeleton with four DOFs (Degrees of Freedom). Its joints implement brushless Maxon DC motors (EC45), while its software uses CAN and CANopen protocols. The authors of (Pan et al. 2018) present an exoskeleton for lower limbs with four joints: hip and knee. This device uses Maxon motors and, through CAN, communicates the master controller with the slave controllers. Furthermore, the work reported in (Yuan et al. 2019) presents an exoskeleton for lower extremities with eight DOFs: four actives (using Maxon motors) and four passives. Its communication system allows the monitoring, the analysis of data, and the adjustment of operating parameters in real-time.

This research compares two versions of a communication system based on the CAN protocol for a lower limb exoskeleton called ALLEX-2, which results from several research projects at the University of Cuenca. The first version of the communication system used only the CAN functions provided by Maxon Motor, resulting in a system with acceptable performance, although not sufficient for the requirements of the exoskeleton. On the other hand, the second version involved the development of proprietary software to achieve performance as close to real-time as possible. With this latest version, the communication system showed a substantial improvement in the times for reading and writing tasks. This it was possible to meet the needs of the robotic device.

This document is organized as follows: Section 2 presents the methodology of this study. Section 3 presents experimental results and a discussion of the main findings. And finally, Section 4 presents the conclusions.

2. Methodology

This work presents the development of the base functions for the software for the communication system of the ALLEX-2 prototype. Figure 1 shows the CAN architecture of the ALLEX-2 prototype. Seven nodes are connected to the CAN bus, one master node with a Raspberry Pi 4, and six secondary nodes with the EPOS4 50/8 (Easy Positioning System) devices. The initial node of the bus is node 3, and the final one is node 6. This architecture allows the communication of a distributed control system, where the master node generates the movement trajectories of each joint and sends the commands to the secondary nodes. The main objective of the CAN communication system is to guarantee that the processing and transmission times of the commands are as short as possible.

The transmission time is reduced by implementing a bus with less than 25 m of length, which allows a maximum transfer rate of 1 Mbps (CIA 2005). It is also possible to decrease the processing time by using functions that accomplish their goal with as few internal processes as possible. This work focuses on the writing and reading processes of CAN frames. The former ones are present in the last part of the processes implemented in the master node. These processes are responsible for taking the resulting data from the control calculations, assembling the CAN frames, and sending the information to the channel. The latter refers to the processing of the measurements sent by the secondary nodes.

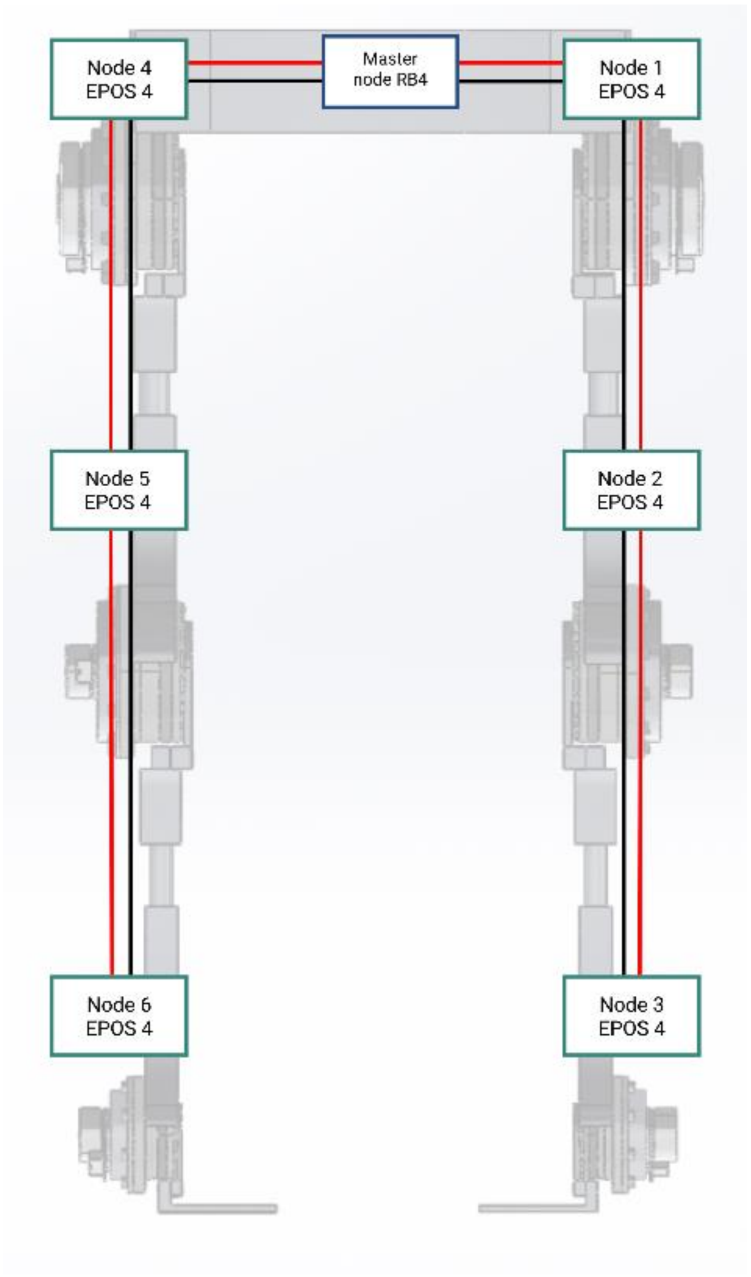

Figure 1: Layout of the CAN topology in the exoskeleton (Cordova et al. 2023).

The need to create new CAN write and read functions arises from the existing functions' poor performance. As mentioned, the prototype uses devices from the manufacturer Maxon, for which it was decided, in the first instance, to use the software provided by the same manufacturer. The following subsection shows the Maxon function description.

2.1 *Maxon Motor functions for CAN*

The EposCmd library implemented by Maxon Motors allows communication management in a master node, where the user and the EPOS4 applications are executed. This library has functions for both network management and CAN communication. The administration functions perform the necessary configurations to open a port to send and receive CAN frames and to prepare the EPOS4 for movement. In this research, these functions are not replaced. For CAN communication, this library has low-layer functions for writing and reading called VCS_SendCANFrame (MAXON_W) and VCS_ReadCANFrame (MAXON_R), respectively. The functions of Maxon Motors are not open source, so their internal operation is a black-box. Algorithm 1 presents a summary of the described functions based on an inference of the required parameters. It can be seen that there is a system for error handling and data conversion to the CAN frame format. In addition, the reading function includes a timeout or maximum waiting time. The EposCmd library implemented by Maxon Motors allows communication management in a master node, where the user and the EPOS4 applications are executed. This library has functions for both network management and CAN communication. The administration functions perform the configurations to open a port to send and receive CAN frames and to prepare the EPOS4 for movement. In this research, these functions are not replaced. For CAN communication, this library has low-layer functions for writing and reading called: VCS_SendCANFrame (MAXON_W) and VCS_ReadCANFrame (MAXON_R), respectively. The functions of Maxon Motors are not open source, so their internal operation is a black-box. Algorithm 1 presents a summary of the described functions based on an inference of the required parameters. It can be seen that there is a system for error handling and data conversion to the CAN frame format. In addition, the reading function includes a timeout or maximum waiting time.

Algorithm 1: MAXON's CAN writing and reading functions
```
1:  function MAXON_W(handle,
2:        COBID, data lenght,
3:        data,  error pointer)
4:      process data
5:      send can frame
6:      if everthing ok return True
7:      else return False
8:
9:  function MAXON_R(handle,
10:       COBID, data length,
11:       timeout, errorCode)
12:     Wait for  CAN frame
13:     if everything ok
14:        return true and CAN frame
15:     else
16:        return false and error code
```

2.2 *Implemented writing and reading functions*

The purpose of these functions (ALLEX_W and ALLEX_R) is to improve the performance of the communication system by reducing processing time. To achieve this, functions with more efficient control over the subprocesses necessary to send or receive CAN frames are implemented. These new tools use the SocketCAN (Community. n.d.) library as a base. This library is designed to be similar to the TCP/IP protocols. Such an approach makes it easier for programmers to develop CAN applications.

Algorithm 2 shows the structure of the writing function called ALLEX_W. This function takes the data to send and transforms it into a frame object. This object serves as an input to the writing method of the SocketCAN library. The input parameters of this function are communication object identifier (COBID (CAN in Automation (CiA) e. V. 2011)), data, data length, and a socket. The latter is the link to the CAN interface and contains the necessary information for communication.

Algorithm 2: ALLEX's CAN writing function.

```
1:  function ALLEX_W(socket,
2:  COBid, data length, data)
3:  for byte in data:
4:  frame[i]= byte
5:  nbytes = write(socket,frame)
6:  if nbytes == data length:
7:  return true
8:  else
9:  return false
```

The reading function called ALLEX_R uses another socket object. A socket must be used for writing and a different one for reading to avoid communication conflicts (Community. n.d.). The function calls the SocketCAN reading method to listen to the channel waiting for a CAN frame. If the process is successful, the function returns the received frame. Unlike the MAXON_R function, the reading function does not have an internal timeout; the time control is executed in a higher process where the read function is called 12 times (2 times for each joint). Algorithm 3 presents the structure of ALLEX_R.

Algorithm 3: ALLEX CAN reading function

```
1:  struct can_frame ALLEX_R(socket)
2:      define nbytes, frame
3:      call read method
4:      if nbytes < 0
5:          return error
6:      else if nbytes < size of frame
7:          return error, incomplete frame
8:      else
9:          return frame
```

Maxon and ALLEX functions to write and read data on the CAN channel are compared using the same software flow. Figure 2 indicates, in a general way, the mentioned process flow of the exoskeleton. In this diagram, it is possible to see the reading and writing processes together with the device's other processes. The MAXON_W and ALLEX_W functions were used in the writing process, while MAXON_R and ALLEX_R were used in the reading process. The algorithm is executed using one of the writes and reads options to measure the execution time, achieved through the methodology explained in the following subsection.

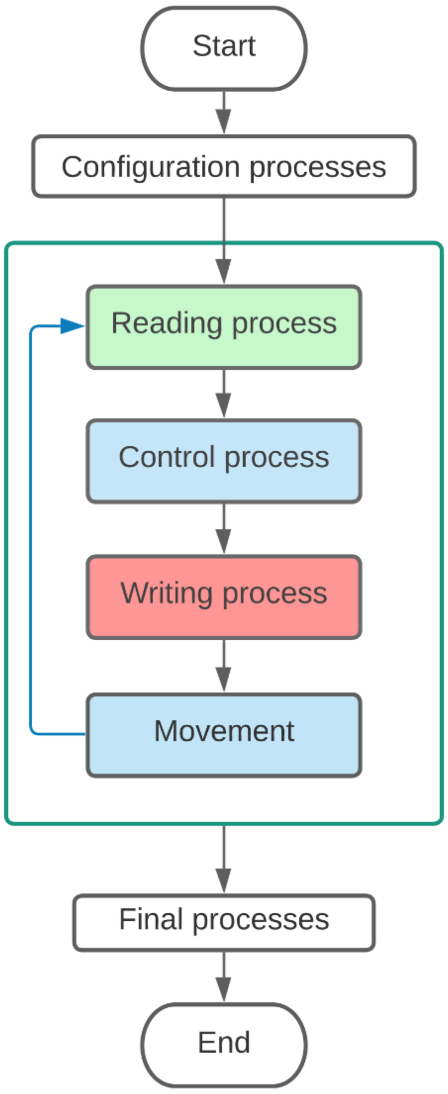

Figure 2: General flow chart.

2.3 *Processing time comparison of writing functions*

The communication process between the nodes is performed according to the writing methods described above. Before sending a CAN frame, the central node prepares the data to be sent. Once this process is complete, the writing method is called to assemble and send the CAN frame. To measure processing times, a ``timer'' object is used. Algorithm 4 describes the structure of the chronometer. It starts the measurement through its constructor method and ends when the object is destroyed to avoid undesired measurements. To measure the execution time is enough to create an object of the timekeeper class. The measured time is appended to a vector; this avoids saving data directly to disk to reduce timer processing. The storage process works at the end of the experiment.

The experimentation process comprises the execution of a gait cycle. This cycle lasts 8 seconds; a sampling and transmission period of 20 ms is used. So, there are 400 writing processes and 800 reading processes per joint (one frame is written and two are read per joint). This experiment was repeated until the data shown in the result section was obtained.

Algorithm 4: High resolution chronometer

```
1:  struct Timekeeper{
2:    Timekeeper //Constructor
3:      start = high resolution clock now
4:    ~Timekeeper //Destructor
5:      Stop = high resolution clock now
6:      duration = stop-start
7:      save duration
```

3. Results

To obtain sufficient data to compare the Maxon and ALLEX functions, the algorithm of Figure 2 was executed while the times of interest were measured. The times are measured in the read-and-write loop (circled in the green box). In the prototype's context, the experiment monitored a gait trajectory; this monitoring was carried out only with three nodes. In each execution of the algorithm, 400 frames per node were written, that is, 1200 frames per execution. When writing a frame, we have a sample of the execution time of the writing function. In total, 6000 measurements were collected for each writing method.

Table 1 shows the results of 6000 executions of the writing method, five experiments for three joints. This table indicates that the method developed in this project (ALLEX_W), on average, performs 5.76 times faster than the MAXON_W method. Also, the reduction in standard deviation means that the ALLEX_W function is more stable than MAXON_W.

Table 1: Measurement result of writing execution times.

	MAXON_W (ms)	**ALLEX_W (ms)**
Mean	1.8875	0.3275
Standard deviation	0.3135	0.1492
Max	8.6899	8.5573
Min	1.5451	0.2258

Figure 3 shows that 99.96% of the execution times for the ALLEX_W function are under 2 ms. On the other hand, 22.37% of MAXON_W execution times are between 2 and 9 ms, while only 0.033% of ALLEX_W function times are over this interval.

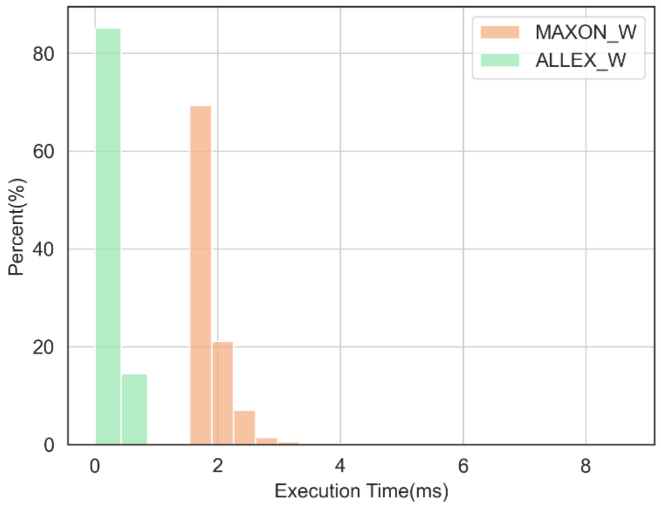

Figure 3: Execution time histogram for CAN writing methods.

During the experimentation, exchanging the reading methods, it was found that the internal structure of the MAXON_R function was not designed for a 20 ms sampling period. This function caused the system to stop, and the reading processes were not executed. Instead, the function developed in this research performs reading processes at a time lower than the mentioned period. The results show that 99.75% of the times the reading takes less than 2 ms when measuring the processing time to read six frames (2000 measures of six frames reading time). The communications system requires reading two frames per join every 20 ms, and the results show that the implemented reading function is better than expected.

This communication system is an essential part of the development of the ALLEX-2. The trajectory references generated by the central processor through this system are transmitted to the distributed controllers. Then, proof of the efficiency of the ALLEX-2 CAN functions can be obtained by analyzing the result of the trajectory tracking in a system where the functions developed in this work are the base of the communication. Figure 4 shows that the root-mean-square tracking error is low in the device performance tests with and without load; this proves that the communication system meets the design requirements. More details about the control system result can be found (Cordova et al. 2023). The source code of the ALLEX-2 project is available in Github (https://github.com/Hiyperion/ALLEX2_SOFT.git).

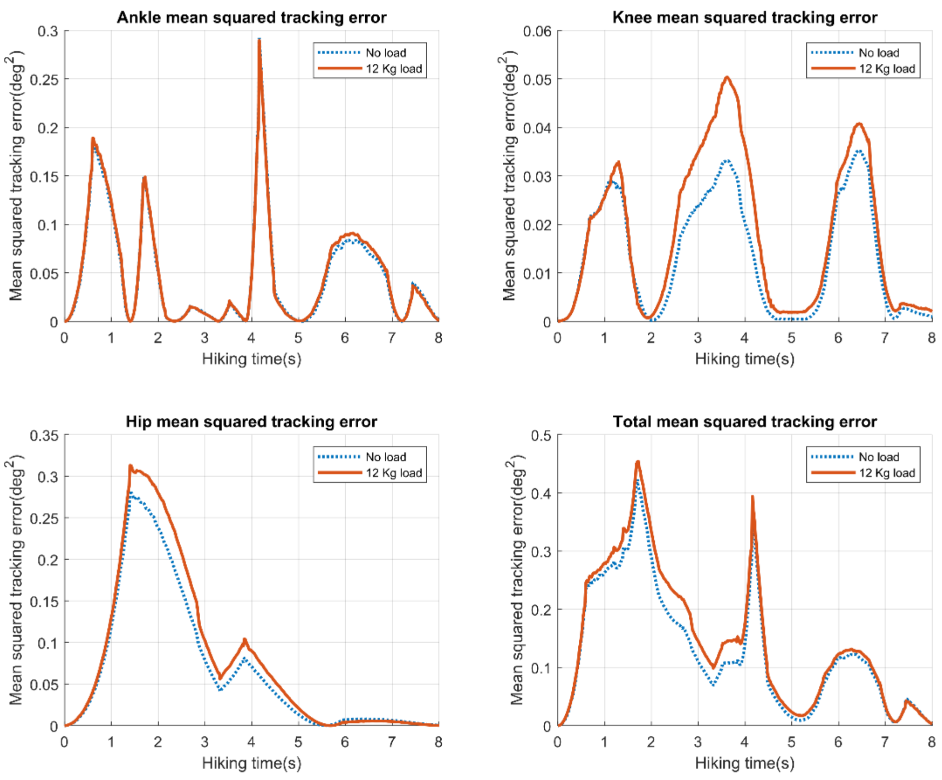

Figure 4: Tracking error for a gait cycle, right lower limb (Cordova et al. 2023).

4. Conclusions

The library developed in this research has better performance features compared to the one provided by Maxon for both writing and reading tasks. In the case of writing, the ALLEX_W function executes 5.76 times faster than the function provided by the Maxon Motor. Also, this improvement implies that the control system can send more operation commands to the exoskeleton.

In this work, the ALLEX_R reading function is significantly better than that provided by Maxon Motor. The function developed in this work can be used in similar applications that require sampling times between 2 ms and 20 ms. This is supported by the results, which show that 99.75% of the reading times for 6 frames is less than 2 ms. Even if fewer frames were used, this time could be less than 2 ms, allowing other processes to be carried out until the sampling time is completed.

Although manufacturers provide tools and software for the fast development of applications. In the case of robotic devices such as exoskeletons, where high performance is required, it is better to develop software that allows the device's requirements to be satisfied. In this research, the developed functions allow the communication system of the ALLEX-2 exoskeleton to comply with the times required for its operation.

Author contributions

En concordancia con la taxonomía establecida internacionalmente para la asignación de créditos a autores de artículos científicos (https://casrai.org/credit/). Los autores declaran sus aportes en la siguiente matriz de contribuciones:

	Morales, H..	Cordova, A..	Minchala, I..	Satudillo-Salinas, F..
Conceptualización	▨	▨	▨	▨
Análisis formal	▨	▨	▨	
Investigación	▨	▨		
Metodología	▨	▨		
Recursos	▨	▨	▨	▨
Validación	▨	▨	▨	▨
Redacción – revisión y edición	▨	▨		▨

Conflicts of Interest

The authors declare that there are no conflicts of interest in any nature.

Acknowledgment

The authors are grateful for the support provided by the Research vice-rectorate 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

Bosch. (1991). "CAN Specification.". Obtenido de http://esd.cs.ucr.edu/webres/can20.pdf

CAN in Automation (CiA) e. V. 2011. "CANopen Application Layer and Communication Profile.". Obtenido de file:///mnt/429874b3-0f62-46ad-98c6-f1dd11bed9c1/Descargas/301v04020006_cor3.pdf

CIA. (2005). "CAN in Automation (CiA): CANopen." (January):1–26.

Community., Kernel development. (n.d). "SocketCAN - Controller Area Network — The Linux Kernel Documentation.". Obtenido de https://docs.kernel.org/networking/can.html

Cordova, A. F., Morales, H., Astudillo-Salinas, F., Zhang, H., & Minchala, L. I. (2023). "Deployment of a High-Speed Communication Network to Enable Real-Time Control of a Lower Limb Robotic Exoskeleton. *International Journal of Innovative Computing, Information and Control*. DOI: DOI: 10.24507/ijicic.19.01.181

Farsi, M., Ratcliff, K., & Barbosa, M. (1999). An overview of controller area network. *Computing & Control Engineering Journal*, 10(3), 113-120. DOI: 10.1049/cce:19990304

Lu, R., Li, Z., Su, C. Y., & Xue, A. (2013). Development and learning control of a human limb with a rehabilitation exoskeleton. *IEEE Transactions on Industrial Electronics*, 61(7), 3776-3785. DOI: 10.1109/TIE.2013.2275903

Maxon. 2022. "Discover the Maxon's World of Drive Technology | Maxon Group." Obtenido de https://www.maxongroup.com/maxon/view/content/index

Di Natale, M., Zeng, H., Giusto, P., & Ghosal, A. (2012). Understanding and using the controller area network communication protocol: theory and practice. *Springer Science & Business Media*. DOI: 10.1007/978-1-4614-0314-2

Pan, C. T., Chang, C. C., Sun, P. Y., Lee, C. L., Lin, T. C., Yen, C. K., & Yang, Y. S. (2019). Development of multi-axis motor control systems for lower limb robotic exoskeleton. *Journal of Medical and Biological Engineering*, 39(5), 752-763. DOI: https://doi.org/10.1007/s40846-018-0449-z

Seoane, L., Díaz, C., Zafra, J., Ibarmia, S., Quintana, C., Canora, C. P., ... & Araujo, A. (2018). CAN implementation and performance for Raman Laser Spectrometer (RLS) Instrument on Exomars 2020 Mission. *IEEE Transactions on Emerging Topics in Computing*, 9(1), 67-77. DOI: 10.1109/TETC.2018.2874643

Yuan, Y., Li, Z., Zhao, T., & Gan, D. (2019). DMP-based motion generation for a walking exoskeleton robot using reinforcement learning. *IEEE Transactions on Industrial Electronics*, 67(5), 3830-3839. DOI: 10.1109/TIE.2019.2916396