Collaboration in the Framework of the HUB4.0MNUVAL DIH for Innovation in Embedded OPC-UA IoT Systems

Alberto Delgado [1], Francisco Blanes [1] and José Simó [1]

[1] *Instituto Universitario de Automática e Informática Industrial, Universitat Politècnica de València, Camino de Vera S/N, Valencia, 46022, Spain*

Abstract

In the literature on DIH, there is a consensus around the services normally provided by a competence center. All of them identified in the EC EDIH call, are not normally dealt with as independent points, because they are part of a complex process where the Small and Medium Enterprises (SME) and the competence center collaborate to produce an innovation. This paper presents the solution adopted as a synergy of two entities, one industrial, the engineering company Dismuntel S.A. and one academic, the Institute of Automation and Industrial Informatics, as an improvement in interoperability and future projection of developments for the connectivity of devices normally based on very specific protocols. The solution includes using a communication bridge (ESP32) to implement an Open Platform Communications Unified Architecture (OPC-UA) communication protocol by designing a client/server program based on this distribution, improving data accessibility. This solution is the result of close relations in the HUB4.0MANUVAL (DIH participant in SMART4ALL project) ecosystem between SMEs and competence centers.

Keywords

Communication protocols, Internet of Things, OPC-UA, embedded systems.

1. Introduction

Dismuntel S.A. is a company in the engineering sector, in the field of electronics, an expert in remote management and energy efficiency. Within this framework, it usually implements specific project developments with very clearly defined communication protocols.

In a quest to add a greater future projection to the developments, this company promotes synergy with the Institute of Automation and Industrial Informatics. Considering the current industry trends, it is possible to understand the current trends influenced by the emergence of low-cost, low-power hardware modules. With these new microcontrollers, System on a Chip (SoC), the developed solutions will offer greater competitiveness and versatility within the market, thus expanding its scope and projection.

The evolution of electronics towards smaller and smaller equipment, with more powerful features and improvements in connectivity, gives new meaning to interoperability between devices, expanding the possibilities of the Internet of Things (IoT) systems and environments. The interoperability and interconnection of devices favor the flow of data and information between equipment that normally does not have common communication protocols. In this paper, an attempt is made to solve this problem by implementing the OPC-UA communication protocol in a hardware device used as a bridge (Figure 1).

More specifically, it will address the solutions implemented to provide greater interoperability to a generator set with its own data switchboard that has only one data output via the CAN communication protocol [1].

Proceedings of the Workshop of I-ESA'22, March 23–24, 2022, Valencia, Spain
EMAIL: aldelro@ai2.upv.es (A. Delgado); pblanes@ai2.upv.es (F. Blanes); jsimo@disca.upv.es (J. Simó)
ORCID: 0000-0001-7314-5743 (A. Delgado); 0000-0002-9234-5377 (F. Blanes); 0000-0003-4677-7627 (J. Simó)

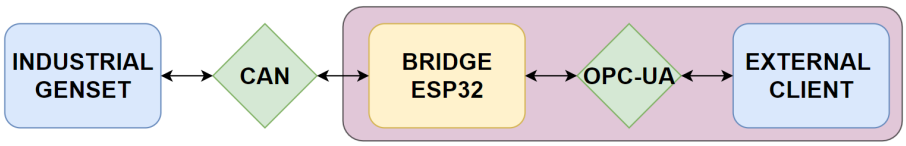

Figure 1: Scheme of interoperability structure between equipment.

The interconnection bridge chosen was a low-cost, low-power microcontroller with high versatility in terms of wireless connectivity, ESP32 manufactured by Espressif Systems [2]. In the specific case of this work, the final-stage solution implemented (bridge-end client) based on the OPC-UA communication protocol will be explained, offering the data to any client based on this type of protocol.

2. Methods and protocols

The major advances in the fourth industrial revolution (i4.0) favor the development of new possibilities for cooperation, particularly at the technical level. Communication between devices and systems is essential to facilitate data exchange and transfer between equipment. Improvements in communication in this type of environment do not require proximity between the devices to be interconnected, which broadens the range of solutions to be implemented. However, this would not be possible if there were no communication mechanisms with guaranteed security and interoperability that would allow i4.0 active devices to operate beyond the physical limits of a plant.

Under these characteristics, OPC-UA appears as a fundamental pillar of the communication standards within the i4.0 framework. This communication protocol, mainly used in automation fields, is built on a client/server structure based on TCP/IP. Each OPC-UA client accesses the OPC-UA server data through point-to-point communication with this one-to-one communication mechanism. The OPC-UA client sends a request to the OPC-UA server and receives a response. The communication between the two ends provides a secure, reliable, and encrypted data exchange without data loss even in situations with unreliable net-work qualities. It is also vendor-independent and open to any platform. This feature, together with its operation based on a simple Ethernet network, makes this protocol an accessible and easy-to-implement option for any device.

About the development of this project, the final stage of communication establishes two extremes, the hardware bridge as receiver of the data sent by the Gen-set and in charge of translating this data to communication nomenclature of the OPC-UA protocol and acting as the **OPC-UA server**, and on the other hand an **OPC-UA end client** in charge of monitoring and controlling the operation of the Genset.

For the implementation of this protocol in the chosen hardware bridge, ESP32, open distribution of this protocol, written in C99, open62541 [3], has been used, more specifically, an adaptation of this implementation developed to be fully dedicated to this microcontroller. This library is available on GitHub where it is periodically reviewed and updated about the updates incorporated in the general implementation (open62541) [4].

3. Implementation

3.1. Hardware configuration

Before starting with the software implementation and design, it is necessary to define and configure the hardware parameters of the bridge. As previously mentioned, the hardware chosen is an ESP32 microcontroller (Figure 2), more specifically the ESP32-Wroover, with an 8MB flash memory to which an Ethernet module will be attached to provide much greater reliability in terms of connectivity. Also, the board has a CAN driver to establish the communication between the Generator and the bridge.

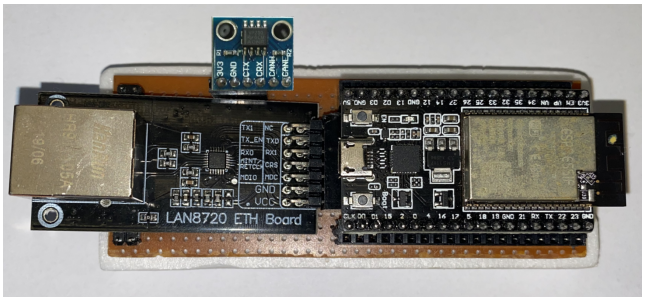

Figure 2: Hardware assembly for the implementation of the solution.

Considering the OPC-UA protocol implementation that will be used for the program design and that it is not specifically designed to be executed in this type of microcontrollers, it will be necessary to enable the external RAM memory option, equipped in the 4MB Wroover version of this SoC.

3.1.1. Software development

Focused on the needs that the company would cover with this development, we began to the establish parameters and methodology of the software solution.

First of all, is the consideration of all data and its organizational structure. As discussed in section 1, the data comes from an electrical generator. A total of 439 variables of different types of data grouped in 7 different blocks need to be processed. Based on these characteristics, two different clustering tests were performed:

A. Generation of one object for each group (7 in total) with array sub-nodes of variable length according to type and amount of data

This first data grouping structure allows it to correspond as much as possible to the original structure established by the genset, but the fact that the arrays are not complete (size of 255) prevents the memory management from being fully optimal since the empty memory gaps accumulate and the memory release is not as efficient as it should be, resulting in small losses that in the long term could col-lapse the SoC memory. This structure is easier to understand looking at the schema shown in Figure 3:

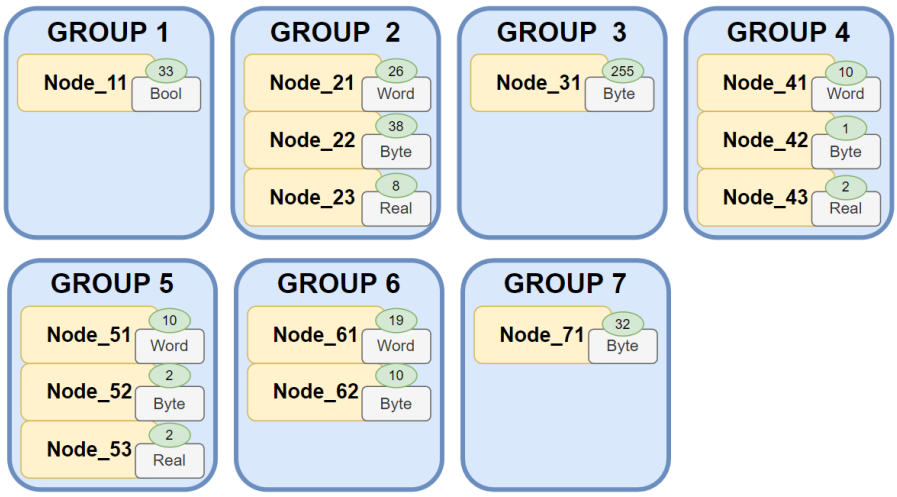

Figure 3: The first option for the data structure scheme in the OPC-UA Server.

B. Generation of a single object with one sub-node for each group (7 in total), with arrays of maximum length (255)

Of the 7 sub-nodes defined within the single object, six of them have the same characteristics, UInt32 type array with a size of 255 positions, in order not to leave memory spaces without variable allocation.

The last sub-node will be defined as a Boolean array with a size of 255 positions, since this block will have read/write permissions, thus allowing the direct control of the electric generator variables. This data structure can be seen more graphically below in Figure 4.

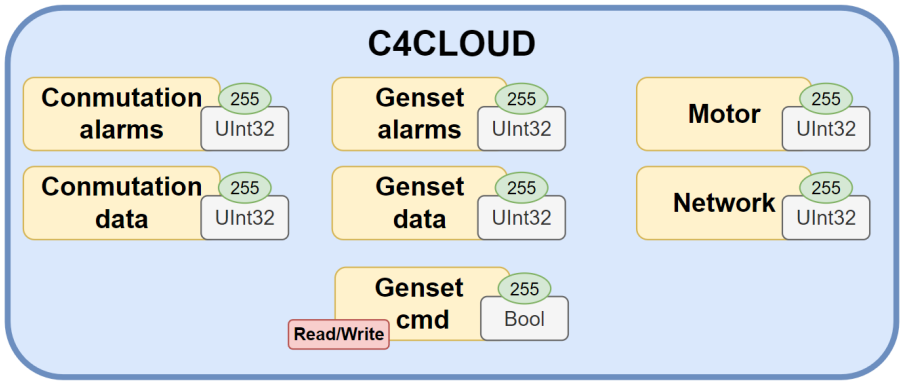

Figure 4: The second option is for the data structure scheme in the OPC-UA Server.

The advantages of this type of structural organization of the data in the OPC-UA server allow a remarkable improvement in the memory management of the SoC, as it has been possible to verify with memory load tests which will be explained in section 4.

4. Results

The development of two different code solutions for data grouping has made it possible to study which of the two solutions is optimum in terms of performance and memory consumption to be implemented as the final solution in a microcontroller in which memory management is key for correct operation over time. For this purpose, several functional tests have been performed in which information about the memory status of the microcontroller has been extracted.

To test both developments, an OPC-UA client (UA Expert [5]) has been used to make random data requests to the server, as well as to modify parameters in the write sub-node for the modification of the Genset control parameters. For both tests, the same operating conditions were considered.

The first distribution of data. Seven different objects with sub-nodes of variable length.

In Figure 5, we can observe the available RAM versus the program cycles, directly related to the running time of the application. At the beginning of the test, there is a sharp drop in the available memory, which then smooths out until 20 hours of operation, at which point there is a recovery and subsequent smooth drop again. This recovery cycle will be repeated approximately every 20 hours of operation ensuring certain stability in the running of the application.

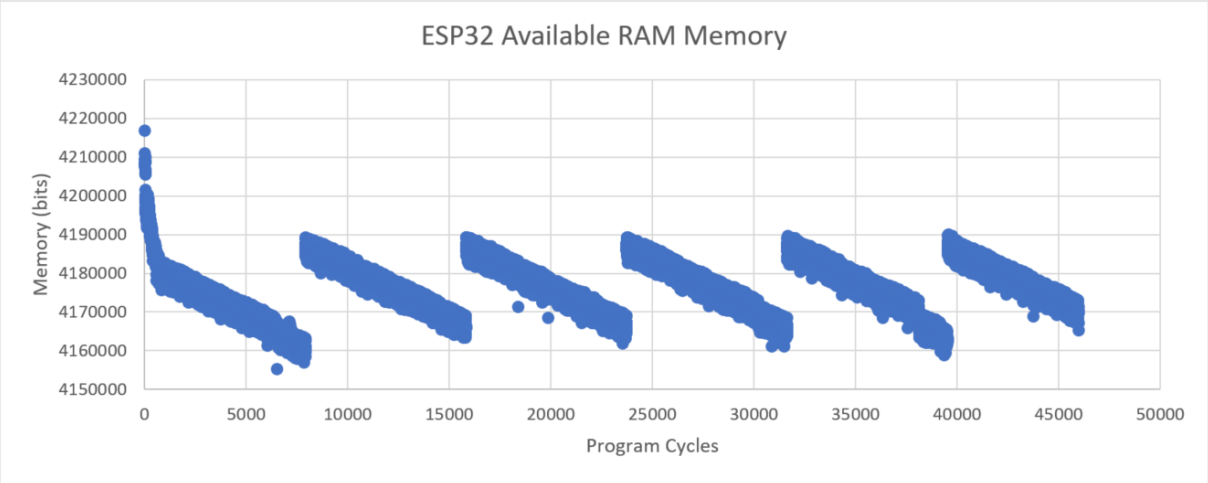

Figure 5: Graph with the representation of available RAM of the SoC for the first option of data structuring.

The second distribution of data. Just one object with seven different sub-nodes of regular length (255).

In the second test performed, the graph represents the same data as in the first test (Figure 6). From the beginning, a stabilization in the available RAM memory line can be observed, which remains constant during the entire test. There are some irregularities and decreases in memory every 5000 program cycles or so, but there is also an immediate recovery of this memory.

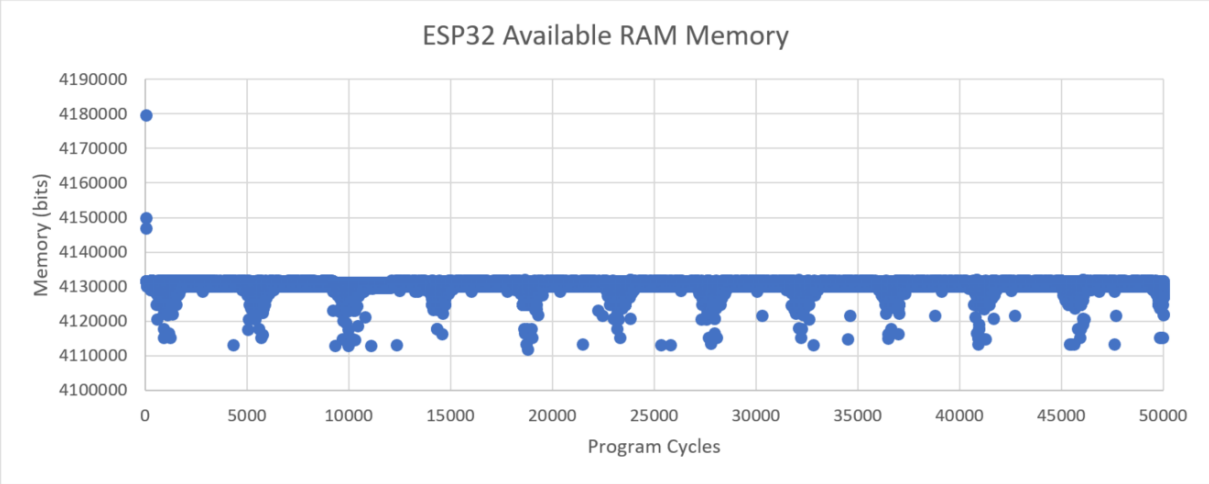

Figure 6: Graph with the representation of available RAM of the SoC for the second option of data structuring.

After studying these load test results and looking for a higher scalability of the solution, the second configuration of the data structure is chosen. With this configuration, the solution is validated in real conditions. As can be seen in Figure 7, the control module of the generator set in charge of generating all the data (print-ed circuits at the top of the image) is extracted. At the bottom of the image is the assembly of the communication bridge with the ESP32, with the Ethernet module for its connection to the network and a controller module for receiving the, CAN frames from the generator control module.

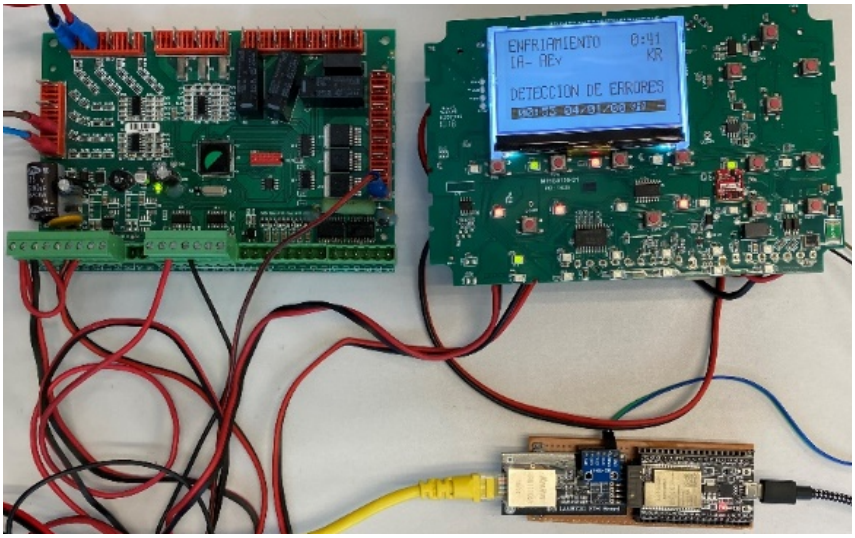

Figure 7: Image of the model used for testing in a real environment.

The bridge assembly itself will act as an OPC-UA Server and, using a computer with the indicated software to make the OPC-UA client requests, it will be possible to monitor the data and control some of them. With this assembly, it is possible to perform a battery of tests to check the correct global operation during a longer period as well as tests to check the correct operation of the disconnection protocols and their corresponding reconnections.

5. Discussion

The cooperation between centers of different natures such as a business entity and an academic organization in the search for a solution to the problem of connectivity between different equipment has generated a beneficial synergy for both parties.

On the one hand, the academic organization has been able to access a real problem existing nowadays in the plants, such as the diversity in the communication protocols and the problems in facilitating the data flow that this generates.

On the other hand, the business entity has benefited from obtaining a versatile solution with many more capabilities in the future-focused on the universality of communications, which will allow this solution to be scaled in the future and adapted to the different needs that arise in real industrial environments.

6. Acknowledgements

SMART4ALL [6] has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No 872614. We thank the reviewers for their detailed and insightful feedback to improve this document.

7. References

[1] CAN communication Protocol, 2022. URL: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf
[2] ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems, 2022. URL: https://www.espressif.com/en/products/socs/esp32.
[3] Open62541: an open source implementation of OPC UA, 2022. URL: https://open62541.org/.
[4] S. Profanter, OPC UA on a ESP32 Microcontroller, 2021. URL: https://github.com/Pro/open62541-esp32.
[5] UaExpert, UA Reference Client' - Unified Automation, 2022. URL: https://www.unified-automation.com/products/development-tools/uaexpert.html.

[6] SMART4ALL, Home, 2022. URL: https://smart4all-project.eu/.