Improving ModBus Extension performance using PRU unit from Sitara AM335x

Cornel Ventuneac, Vasile Gheorghita Gaitan, Catalin Lupu
Faculty of Electrical Engineering and Computer Science
Stefan cel Mare University
Suceava, Romania
corventu@yahoo.com, vgaigtan@usm.ro, lupucata@yahoo.com

***Abstract*— Local industrial networks represent a model with only three levels physical, data link, and application for use in the domain of the industrial processes using computing elements with low resources of type microcontroller. Modbus is a simple and easy to implement protocol but partially defined from the point of view of devices determinism and functionality described in the network. Modbus Extension completes the Modbus by defining some Modus compatible messages of an acquisition cycle and a device description mode. To implement the acquisition cycle the best possible implementation in using a communication channel was pursued, in conditions that the serial communication speeds of modern microcontrollers have reached up to 27Mb/s. At this speed, any over-control given by real-time operating systems or interrupt handling routines creates significant times in the structure of an acquisition cycle. The paper presents an implementation solution using the PRUs from Sitara AM335x, which at a speed of 12 Mb/s, a 75.53% percentage of channel usage for payload data was obtained.**

Keywords*— *Local industrial networks; ModbusE; Acquisition cycle; PRU;

I. INTRODUCTION

The appearance and use of local industrial networks have resulted in the development of automation in industrial processes. Local Industrial Networks (LINW) provide physical and logical support for the integration of various sensors and transducers in IoT (Internet of Things). Local industrial networks include sensor and actuator networks, networks that connect the sensors with the execution elements (AS-I, Modbus RTU, CANOpen), device-level networks (Modbus, Interbus-S, Profibus, CANOpen), and networks at the control level that connects PCs, PLCs, automation units. Local industrial networks can be oriented to domains or applications so they can be used in cars, smart homes, motion control, the military, aviation, or in domains where reliability restrictions are imposed. Due to a large number of specifications and communications protocols, the efforts are directed toward standardization and reducing the big number of standards. The progress of industrial Ethernet will affect the development of local industrial networks. Of course, that industrial Ethernet will not make the local industrial networks on the lower levels completely to be outdated. Not all communication protocols have fully defined specifications, as a result, problems may occur with temporary coherence, description of network devices, MAC (Medium Access Control) schemes.

The Modbus protocol is an example of a communication protocol that does not have fully defined specifications. The Modbus protocol was created by Modicon in 1979 for data communication using a serial line. The Modbus protocol is positioned on level 1, level 2, and level 7 of the OSI stack. Our days the Modbus protocol is unique, very appreciated, and used protocols between industrial devices. There are several ways to implement the Modbus protocol, namely [1]: RTU mode - asynchronous serial transmission (RS232, RS422, RS485); ASCII mode - asynchronous serial transmission; MODBUS PLUS mode; TCP/IP mode. The Modbus RTU protocol is used at the devices and sensors level and the Modbus TCP/IP it is used at the application level.

The main features of the Modbus communication protocol are [2]: easy to implement; small footprint; scalable in complexity; easy to manage and improve; free specifications; is an open protocol; cheap to implement; is a standardized protocol. Starting from the Modbus communication protocol, an extension has been proposed that improves the Modbus RTU protocol. This extension is called Modbus Extension (ModbusE) and addresses level 2 (data link level) of the OSI model. According to the ModbusE protocol, a master station called BSG (Base Station Gateway) is required to introduce the time variable. It will be used further term client instead of the master term and the term server instead of the slave term. This BSG station will allow access to the local industrial network using the internet [2]. At the level of this BSG station, the acquisition cycle will be implemented, thus being introduced the time variable. In paper [3] was briefly defined the ModbusE protocol and have been presented: the BSG (Base Station Gateway) station that manages the acquisition cycle; acquisition cycle structure; messages structure; SDO (Service Data Object) and PDO (Process Data Object) objects; multiprocessor work for UART (ModbusE devices only); interrupts; DMA transfer; as well as specific RS485 facilities for switching the direction of communication; the time interval of 3.5 and/or the interval time of 1.5 between two characters; commands from 100 to 102 used for reading and slot address mapping operations; an algorithm for determining the duration of slots. In this paper is presented a solution to get the optimal

978-1-6654-8162-5/22/$31.00 ©2022 IEEE

data flow as close as possible to the bandwidth for messages communication. The solution will be presented for the maximum serial communication speed of 12Mb/s. The implementation was performed on a Sitara AM335x microcontroller, equipped with a 32-bit ARM Cortex A8 RISC core and two real-time PRU (Programmable Real-Time Unit) cores.

Next in section II are presented related work, in section III is presented the basic architecture of a BSG (Base Station Gateway), in section IV is presented the structure of Modbus Extension messages and the acquisition cycle implementation, and the performances evaluation of the acquisition cycle, section, in section V are presented comparisons with other solutions, and in section VI conclusion and future work.

II. Related Work

Because the Modbus protocol is a very popular and appreciated protocol various extensions were proposed and some of them keep their backward compatibility with most of the statements that were mentioned in section I. In paper [4] the authors proposed to expand the addressing space from 8 bits addresses to 16 bits addresses. This way, only devices that implement this extension will accept this addressing mode, and the others will ignore these messages. Also in this paper, a multi-master architecture has been proposed in which each device can become a master at some point through a master's choice protocol. The extension is very simple and does not require any additional hardware, so it is suitable for cheap embedded distributed systems. Another direction of research regarding Modbus RTU refers to the correction and detection of errors. In paper [5] a specialized repeater device is used for the detection of an error, this device is used both in the receiver and in the transmitter. This approach respects the classic Modbus RTU protocol so the bus extension can also be realized with classic Modbus RTU devices. In paper [6] it is proposed a better approach to the Modbus RTU protocol for the integration of devices in industrial networks, using wired or wireless communications Modbus RTU. This type of communication protocol increases the topological and control limits of the Modbus RTU classic, allowing a wireless/wired tree bus network topology. The proposed architecture presented a reduced rate of error communication, showing that the solution that was described in the paper can meet the tough requirements of industrial communications networks. In paper [7] is provided a case study where Protege language can be used in industrial systems that use Modbus protocol for serial port and TCP/IP. The authors of the paper [8] a Modbus slave model using Modbus RTU for serial line RS-485, and the work is focused on creating a software architecture that reduces the development period. In paper [9] it is presented a tool to measure the response time in industrial networks using Modbus protocol on the RS485 serial line. In paper [10] it is proposed an adaptation of Modbus for controller area network (CAN) using embedded devices that are not expensive. In paper [11] it is presented a simulation of the traffic in networks using Modbus protocol to evaluate security in the systems of type SCADA (Supervisory control and data acquisition).

Regarding the Modbus Extension protocol in previous works, the performances resulted from using Cortex Mx cores were 49.6% payload data at a speed on the serial port of 10.5 Mb/s with STM32F407, 36% payload data at speed on the serial port of 27Mb/s with STM32F746, and 58.9% payload data at speed on the serial port of 11.5Mb/s with LPC4300 which has 2 Cortex M0 and M4. But Cortex M7 and Cortex M4 do not have enough resources that can be used to implement OPC UA client and server of high performance [12].

III. Internet Of Things Architecture Implemented By Base Station Gateway

In the paper [13] it is presented the architecture of the IoT gateway used for improving the useful data flow in the acquisition cycle level. The solution proposed in this paper can be integrated into a monitoring and control system used in smart buildings [14]. Namely, it was proposed to use the processor Sitara AM335x developed by Texas Instruments. Thus the processor Sitara AM335x is composed of a 32 bit ARM Cortex-A8 core that operates at 1GHz and 2 Programmable Real-Time Units (PRU) cores that operates at 200 MHz. In the paper [13] it was proposed that implementation of the Modbus Extension protocol acquisition cycle to be done on one of the two PRU cores and that the OPC UA server and client be implemented on the high power ARM Cortex A8 core. It was also presented that the real-time PRU (Programmable Real-Time Unit) cores allow the building of the client/server Modbus Extension protocol with higher usage performance of communication channel than the solutions presented in previous works on Modbus Extension protocol.

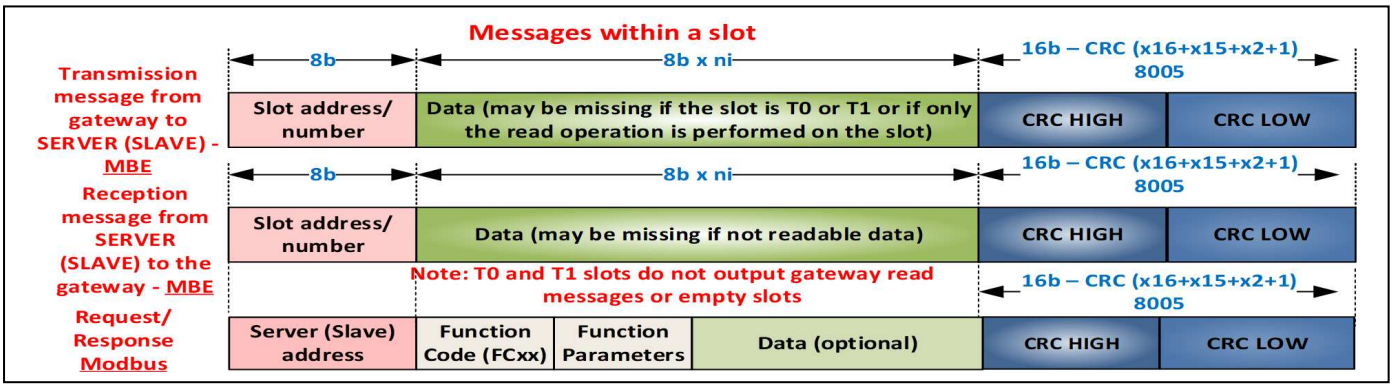

Fig. 1. ModBus Extension, Modbus RTU messages [12].

IV. PERFORMANCE ANALYSIS OF THE DATA ACQUISITION CYCLE

Three types of messages can be used for the Modbus protocol [3]: send data with recognition (Send Data with Acknowledgment - SDA) - request-response messages; send data without recognition (Send Data with No Acknowledgment - SDN) - request messages; send and request data (SDR) - request-response. In Fig. 1 it is presented the structure of ModbusE and Modbus RTU protocol messages (MBE is short for Modbus Extension).

As can be seen in Fig. 1, Modbus Extension protocol messages have no function code and no function parameter fields either in the request messages or response messages so the bandwidth is lower than of Modbus RTU case, so for an identical quantity of useful data the Modbus Extension message is shorter than the Modbus RTU message. A Modbus Extension message can be easily converted into a Modbus RTU classic message if the address of the slot becomes the address of the server (slave), and the beginning part from the data area becomes the function code and the function parameters. In the ModbusE protocol, addresses larger than 0x80 are treated as Modbus RTU addresses. In Fig. 2 it is presented the acquisition cycle (AC) structure where slot S0 is SYNC and slot Sn-1 is SEND.

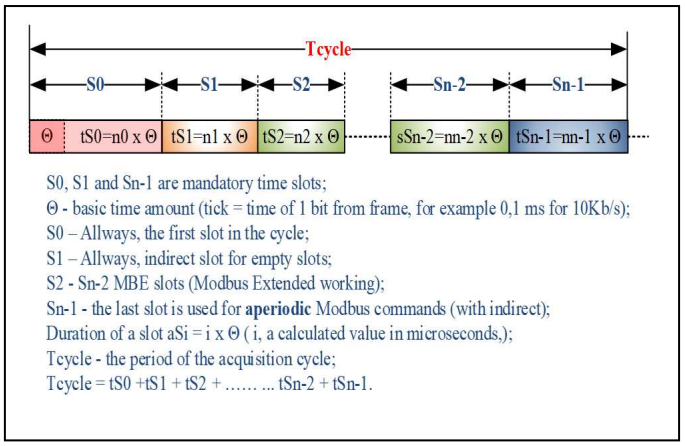

Fig. 2. Acquisition cycle structure [12].

For the acquisition cycle are defined [3]: the acquisition cycle may have at least 3 slots and cannot have more than 246 slots (the 247 number being kept for local commands on the gateway); each slot supports n_0, n_1,....,n_{n-1} characters sent and received (slots 0,1 and those indirected to the slot 1 do not receive characters); the slots have a data structure attached with status information, control, and data; each slot consists of ni x θ ticks (it is better that the tick to not generate interrupts but to be input clock for the timer); a Modbus Extension protocol device can subscribe to multiple slots using configuration; slot 0 (n_0) is a slot with special functions that marks the beginning of the cycle and can be used, for example, to propagate the command "start scanning" the inputs or send data to the outputs; slot number 1 (n1) is a slot with special functions, it is used to indirect those slots that are empty (whether they are not defined or server stations working with these kind of slots are no working anymore or defective); the slot (nn-1) is a slot with special functions and generally used for aperiodic commands and commonly indirected to slot 1, if no aperiodic commands exists , or to a slot that has the number greater than the last slot from the acquisition cycle with the indirect command. The number of bytes of the message issued for this command is less than or equal to that allowed for the last slot.

As previously mentioned, one of the real-time PRU processors is used to implement the acquisition cycle (AC). The acquisition cycle will be implemented on the client station (master) which will send messages to the server station (slave). The server station retrieves data from the sensors or sends data to the execution elements. Reception and transmission of messages from slots is done with speeds of up to 12 Mb/s, the speed of 12Mb/s requires a very good response for the PRU (the duration of a 10-bit character to be processed is 0.83 microseconds).

The software for the client station (master) is based on the indicators of the peripheral devices used UART and the timer IEP (Industrial Ethernet Peripheral), the IEP timer with two comparators used to determine the end of a message at reception and to signal the end of the slot. When the end of the slot is signaled, the comparison register with the duration of the next slot is loaded and the RS485 driver is switched to transmission, after which the transmission of the message starts using the indicator for the empty transmission register.

During this time, the overrun indicator for slot duration is also tested. If an overrun occurred, a transmission error is signaled, the RS485 driver is switched to reception and the next slot is moved. If the entire message has been sent, it is switched to reception and the reception of the first character or overrun is expected for the duration of the slot, in which case a transmission error is signaled, the RS485 driver is switched to reception and move to the next slot. If a character has been received, it stays on reception until the end of the message is indicated. In this loop the indicator for overtaking the slot is tested and the same actions mentioned above are taken. It then switches to broadcast and waits for the slot completion indicator to be exceeded, then the main loop resumes with the next slot (Next_Slot = (Current_Slot + 1) % NumberMaxSlot).

The software for the server station (slave) similarly takes place with the difference that at the level of the server (slave) the operation of moving the message from the reception buffer to the buffer of the application appears. Further we will present the results obtained for an acquisition cycle consisting of 10 slots. Only serial communication between client and server was taken into account and no external action was taken through the ARM Cortex A8 processor.

As can be seen from Fig. 3 the period of a 10 slot acquisition cycle is 1.085ms. Theoretically in continuous emission conditions, 1307 10 bit characters can be transmitted on a 10 slots cycle. In reality 1234 characters (8 useful data bits without the start bit and the stop bit) were transmitted over a 10 slot acquisition cycle. So it results in an upload percentage of characters of 96.79% and a payload (useful data) of 75.53%.

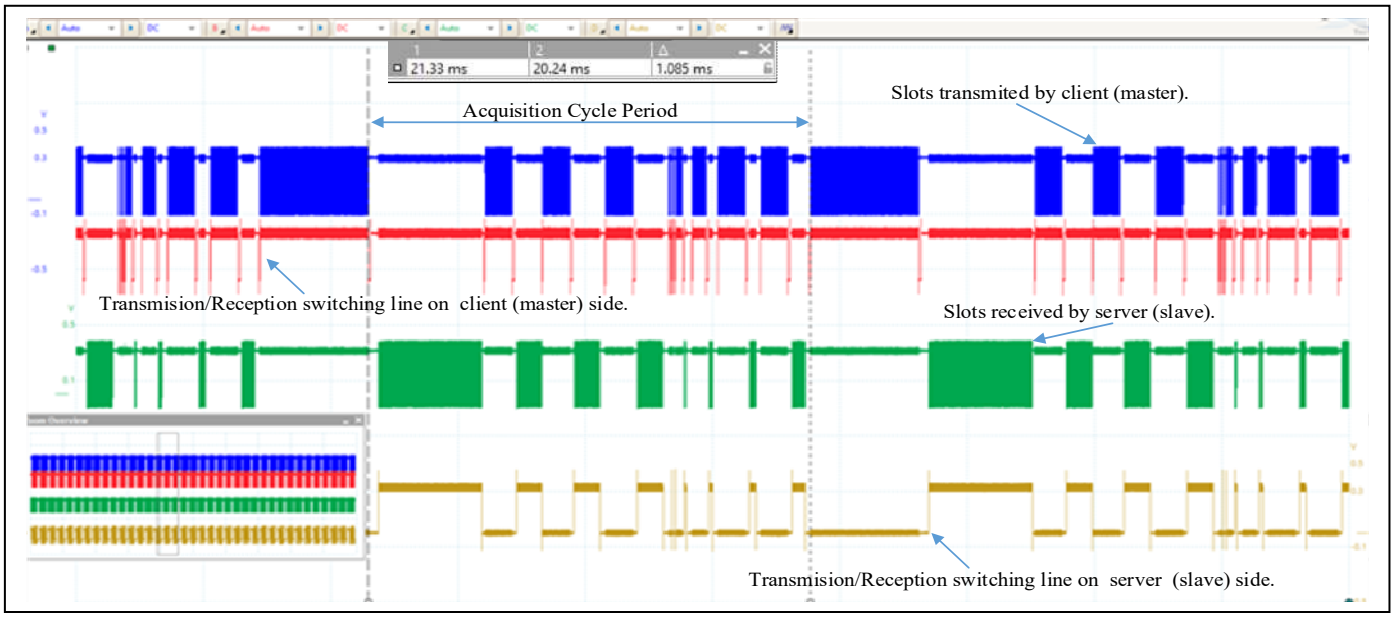

Fig. 3. Acquisition cycle.

TABLE I. COMPARISONS WITH OTHER SOLUTIONS

Hardware Support	Serial line Rate(Mb/s)	Communication Efficiency %	Efficiency useful data %	Software Implementation	Processors
STM32F407 168 MHz [12]	10.5	70.62	49.6	Real-Time Operating System, Interrupt Service Routine, Direct Memory Access	Cortex M4
STM32F746 216 MHz [12]	27	37.8	36	Real-Time Operating System, Interrupt Service Routine, Direct Memory Access	Cortex M7
MCB4300(LPC4357) 180 MHz [2]	11.5	76	58.9	Pooling	Cortex M4, M0 (used Cortex M0)
Sitara AM335x (ARM Cortex A8 1GHz, PRUs – 200MHz)	12	96.79	75.53	Pooling	ARM Cortex A8 RISC core and 2 PRU cores – used PRU0

V. COMPARISONS WITH OTHER SOLUTIONS

As we can see in Table I, from previous work the performances resulting from using Cortex Mx cores were 49.6% payload data at a speed on the serial port of 10.5 Mb/s with STM32F407, 36% payload data at speed on the serial port of 27Mb/s with STM32F746, and 58.9% payload data at speed on the serial port of 11.5Mb/s with LPC4357 which has 2 Cortex M0 and M4. The payload of 75.53% for Sitara AM335x was obtained for an acquisition cycle consisting of 10 slots. Also the Sitara AM335x microcontroller, unlike the LPC4357, has the hardware resources (ARM Cortex A8 core) to implement the server and client OPC UA required for the IoT gateway that uses Modbus Extension protocols. In Table I the data and the values for the first two rows were taken from the paper [12] and for the third row were taken from the paper [2]. In Table I, there are compared more solutions of the same Modbus Extension data source on different hardware facilities. The software implementation (that is not a simple porting, but it is a specific new algorithm for PRU architecture) is optimized on the specific hardware in order to use all facilities (UART peripherals, EDMA, etc.) that can speed up the execution and improve the data payload.

VI. CONCLUSION AND FUTURE WORK

Due to the implementation of the software without a real-time operating system and interrupts, an important over-control is eliminated, which allows the improving of the data flow on the communication channel. Thus, a percentage of 75.53% was obtained for the transfer of useful data (payload). These results were obtained in conditions when the slot duration adjustment time was the same for all slots. This time depends on the number of characters that were sent by the client station, characters that at the server station level must be moved from the data link level to the application level. The longer the message it is, the larger the adjustment time must be. So setting the adjustment time to the size of the message sent would increase the use of the channel for useful data. As future work,

it will be studied and analyze the use of DMA channels to transfer data directly from the ARM Cortex A8 processor data area and the PRU data area. A good result was also obtained by using the LPC4357 microcontroller (58.9%) but for a speed of 11.5 Mb/s. The higher the communication speed is, the more significant the time to move the message in the calculation of the percentage of channel usage because the duration of the acquisition cycle will be larger in that the duration of a bit is longer, namely 86.95 ns against 83.33ns. As the main contribution, this paper proposed an implementation for ModBus Extension that was optimized and tested on the PRU unit from Sitara AM335x.

ACKNOWLEDGMENT

This work is supported by the project ANTREPRENORDOC, in the framework of Human Resources Development Operational Programme 2014-2020, financed from the European Social Fund under the contract number 36355/23.05.2019 HRD OP /380/6/13 – SMIS Code: 123847.

REFERENCES

[1] Modbus.org, "MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b", 2006. Available: from https://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. [Accessed: 14- Feb- 2022].

[2] V. GĂITAN, & I. ZAGAN, Local industrial networks – Modbus Extension /Rețele industriale locale – Modbus Extins. Suceava: Editura Universității "Ștefan cel Mare", 2019.

[3] V. G. Gaitan, N. C. Gaitan, I. Ungurean, "A flexible acquisition cycle for incompletely defined fieldbus protocols", ISA Transactions, vol. 53, no. 3, 2014. DOI: 10.1016/j.isatra.2014.02.006.

[4] G. Cena, M. Cereia, I. Cibrario Bertolotti and S. Scanzio, "A MODBUS extension for inexpensive distributed embedded systems," 2010 IEEE International Workshop on Factory Communication Systems Proceedings, Nancy, pp. 251-260, May 2010. doi: 10.1109/WFCS.2010.5548625.

[5] C. Urrea , J. Kern, C. Morales, "Error detection and correction to enhance the data rate of smart metering systems using Modbus-RTU", Electr Eng, 2020. Available: https://doi.org/10.1007/s00202-020-01067-7.

[6] G. B. M. Guarese, F. G. Sieben, T. Webber, M. R. Dillenburg, C. Marcon, "Exploiting Modbus Protocol in Wired and Wireless Multilevel Communication Architecture", 2012 Brazilian Symposium on Computing System Engineering, Natal, Brazilia, pg. 13-18, 2012. doi: 10.1109/SBESC.2012.12.

[7] Y. Wang, V. Gaspes, "A compositional implementation of Modbus in Protégé", 2011 6th IEEE International Symposium on Industrial and Embedded Systems. doi:10.1109/sies.2011.5953654.

[8] T.-S. Nguyen, T.-H. Huynh, "Design and implementation of modbus slave based on ARM platform and FreeRTOS environment", 2015 International Conference on Advanced Technologies for Communications (ATC). doi:10.1109/atc.2015.7388372.

[9] G. Künzel, M.A. Corrêa Ribeiro, C.E. Pereira, "A tool for response time and schedulability analysis in modbus serial communications". In Proceedings of the 2014 12th IEEE International Conference on Industrial Informatics (INDIN), Porto Alegre, Brazil, 27–30 July 2014; pp. 446–451. doi:10.1109/indin.2014.6945554.

[10] G. Cena, I. C. Bertolotti, T. Hu, A. Valenzano, "Design, verification, and performance of a MODBUS-CAN adaptation layer", 2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014). doi:10.1109/wfcs.2014.6837605.

[11] R. Al-Dalky, O. Abduljaleel, K. Salah, H. Otrok, M. Al-Qutayri, "A Modbus traffic generator for evaluating the security of SCADA systems", 2014 9th International Symposium on Communication Systems, Networks & Digital Sign (CSNDSP). doi:10.1109/csndsp.2014.6923938.

[12] V. G. Găitan and I. Zagan, "Experimental Implementation and Performance Evaluation of an IoT Access Gateway for the Modbus Extension," Sensors, vol. 21, no. 1, p. 246, Jan. 2021, doi: 10.3390/s21010246. [Online]. Available: http://dx.doi.org/10.3390/s21010246 .

[13] C. Ventuneac, V. G. Găitan, "Implementation of an IIoT Access Gateway for the ModBusE – Modbus Extension using BeagleBone Black", INTERNATIONAL CONFERENCE European Integration - Realities and Perspectives 16 th Edition, EIRP 2021, Danubius University, May 2021.

[14] I. Ungurean and N. C. Gaitan, "Monitoring and control system for smart buildings based on OPC UA specifications," 2016 International Conference on Development and Application Systems (DAS), 2016, pp. 82-85, doi: 10.1109/DAAS.2016.7492552.