THE 16TH EDITION OF THE INTERNATIONAL CONFERENCE
EUROPEAN INTEGRATION
REALITIES AND PERSPECTIVES

Implementation of an IIoT Access Gateway for the ModBusE – Modbus Extension using BeagleBone Black

Cornel Ventuneac[1], **Vasile Gheorghiță Găitan**[2]

Abstract: The implementation of this Industrial Internet of Things (IIoT) gateway for the ModBusE protocol aims to improve the data flow at the level of slots and acquisition cycle, using microcontroller's architectures. In this paper, the gateway implementation is performed using a BeagleBone Black board. BeagleBone Black uses the Sitara AM335x processor from Texas Instruments. This processor consists of a 32 bit ARM Cortex A8 processor that operates at a frequency of up to 1 GHz and two 32 bit PRU (Programmable Real-Time Units) processors that operate at a frequency of 200 MHz. The acquisition cycle of the ModBusE protocol based on a client-server (master-slave) architecture will be implemented on one of those two PRU processors. The high-power ARM Cortex A8 processor can be used to implement OPC UA server and client. The aim is also to develop a specific AM335x Sitara driver that allows fast transfer between the two processors (PRU – ModbusE client / server and Cortex A8 - OPC UA server and client). This paper proposes the general structure of an IIoT gateway that makes the connection between the physical environment using devices connected through the ModBusE protocol and the upper levels of IIoT through OPC UA.

Keywords: acquisition cycle; PRU; fieldbus systems;

1. Introduction

1.1. Modbus Protocol

Fieldbus systems, networks created for the lowest levels of the automation hierarchy, have had an enormous influence on the flexibility and performance of modern automation systems in all areas of application. A more detailed definition of fieldbus is given by the Fieldbus Foundation: Fieldbus is a digital communication link, bidirectional, multi-drop between intelligent measuring and control devices. It serves as a Local Area Network (LAN) for advanced process control, remote input / output applications and high-speed factory automation applications (Persechini & Jota, 2013).

The MODBUS protocol was published in 1979 by MODICON for communication with programmable controllers (PLCs) produced by the company. Modbus is a communication protocol located on levels 1, 2 and 7 of the OSI reference model. It has become a standard of communication in the industry and is currently one of the most widely used protocols for connecting industrial devices. The most important reasons for its widespread use are:

1. It is an open protocol, with documentation available.

[1] PhD, Faculty of Electrical Engineering and Computer Science, Stefan cel Mare University of Suceava, Romania, Address: University Street 13, Suceava 720229, Romania, Corresponding author: corventu@yahoo.com.
[2] PhD, Faculty of Electrical Engineering and Computer Science, Stefan cel Mare University of Suceava, Romania, Address: University Street 13, Suceava 720229, Romania, E-mail: vgaitan@usm.ro.

2. It can be implemented in a short time (days not months).

3. Works with bits / bytes and thus does not impose special requirements on manufacturers.

Today, the Modbus protocol is a unique protocol, one of the most popular among automation devices. Schneider Electric transferred the Modbus and Modbus / TCP specifications (Modbus over TCP / IP) to Modbus.org. There are 2 versions of the MODBUS protocol one for the serial port (RS-232 or RS-485) and one for Ethernet.

Modbus serial communication variants:

a. Modbus RTU - data is represented in binary into a compact form.

b. Modbus ASCII - This version uses ASCII characters that can be read by human operators.

The Modbus / TCP is similar to the Modbus RTU, but the data is transmitted in TCP / IP packets. The Modbus protocol defines the PDU (Protocol Data Unit), see figure 1, as the basic data unit of the protocol, regardless of the communication levels (Modbus.org-2, 2006).

Figure 1. MODBUS Protocol Data Unit (Modbus.org-2, 2006)

We can see both the MODBUS frame for serial line and MODBUS request/response over TCP/IP in figures 2 and 3.

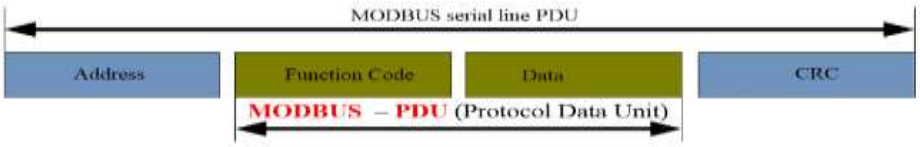

Figure 2. MODBUS Frame for Serial Line (Modbus.org-2, 2006)

For the MODBUS serial protocol, the address field often contains only slave addresses and the function code indicates to the server what actions to perform. The function code can be followed by the data field, which contains the request and response parameters. Error detection is performed depending on the content of the message. There are two calculation methods which depend by the chosen transmission mode (RTU or ASCII) (GĂITAN & ZAGAN, 2019).

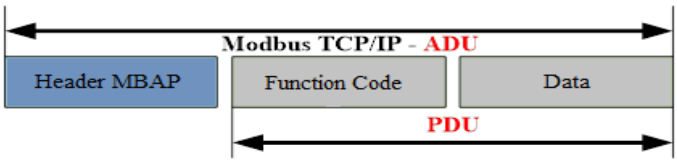

Figure 3. MODBUS Request/Response over TCP/IP (Modbus.org-1, 2006)

Header MBAP field contain transaction identifier, protocol identifier, length and unit identifier, the Function Code field represents the requested service to the server when it is a request and represents the success or failure of the requested service in a response, excluding the unanswered broadcast case. The size of the code field is 1 byte. Exceptions are easy to detect because the value of the code field will have the most significant bit enabled. The Data field contains additional functions, when it is a request, and the results of processing functions when it is a response. For successful processing, the content will

be any date that has been requested by the function. For unsuccessful processing, the content will be an exception code. The size of the data field is from 0 to 252 bytes, depending on the value of the code field (Găitan & Zagan, 2019).

1.2. Modbus Extension

ModBusE is addressing level 2 of OSI model. Because the protocols that are incompletely defined do not include specifications about the time variable, this time variable is not used explicitly at the level of the server station. In this case a client station is required to enter the time variable.

This client station will be called Base Station Gateway (BSG). This BSG allows access to the fieldbus through the internet via direct connection or via a host computer. The client station can add a time stamp to a unit of information or a set of information (Găitan & Zagan, 2019). This time interval depends on the one hand by the definition in the protocol of a broadcast command (such as the start of data acquisition) and on the other hand by the structure of the acquisition cycle (logic variables are read or written faster than analog variables). The time required for signal processing in such a network should also be included, the stations being fully synchronized. To improve the performance of the MODBUS protocol, an original extension called Modbus Extension (ModbusE) was proposed (GĂITAN & ZAGAN, 2019). A data acquisition cycle at BSG level will be implemented to satisfy the time coherence. In fieldbuses the data can be transmitted cyclically, on request, based on events or combined. The acquisition cycle (AC) consists of slots. These slots are used to allow the transmission and reception of messages. The ModBusE slot message is presented in figure 4. There are two methods that can be used regarding the access to the communication environment: non - deterministic (Ethernet) and deterministic (central master, token passing - round-robin, collision with a winner). The structure of the acquisition cycle depends by the type of industrial network used (Xiang-li, 2009).

Figure 4. MODBUS Extension Slot Message (Găitan & Zagan, 2019)

2. Related Work

Industry 4.0 refers to the growing need to unify digital technologies and the Internet with the conventional industrial devices. Automation and data exchange include: the Internet of Things (IoT), the Industrial Internet of Things (IIoT), cloud computing, cognitive computing and artificial intelligence. There are a multitude of communication protocols that can be used by IoT gateways. The (Silva & Silva, 2019) paper presents a low-cost prototype IoT gateway for connecting conventional Modbus RTU sensors / actuators to an MQTT IoT cloud. The (Cena, Cereia, Bertolotti, & Scanzio, 2010) paper shows how MODBUS can be extended, in a backward-compatible way, to address these shortcomings. The extension is very simple and does not require any additional hardware, hence it is suitable for inexpensive, distributed embedded systems. From previous work related to ModBus Extension it has been shown that the performance obtained with Cortex Mx architectures were approximately 49.6 % useful data from the acquisition cycle with STM32F407 at 10.5 Mb / s, 36 % useful data from the acquisition cycle with STM32F746 at 27Mb / s, and 80.6 % useful data from the acquisition cycle with LPC4300 which, however, has 2 Cortex M0 and M4. However, Cortex M4 and M7 does not have

sufficient resources to implement high - performance OPC UA server and client. (Gaitan & Zagan, 2021).

3. IIoT Architecture Implemented by BSG

The implementation of this Industrial Internet of Things (IIoT) access gateway for the ModBusE protocol aims to improve the data flow at the level of slots and acquisition cycle, using multiprocessor architectures. In this paper, the gateway implementation approach was done using a BeagleBone Black board. BeagleBone Black uses the Sitara AM335x processor from Texas Instruments.

The AM335x Sitara consists of a 32-bit ARM Cortex A8 RISC processor that operates at a frequency of up to 1 GHz and two 32-bit real-time PRU (Programmable Real-Time Units) processors that operate at a frequency of 200 MHz. The acquisition cycle of the ModBus Extension protocol based on a master slave architecture will be implemented on one of those two real-time PRU processors. The high-power ARM Cortex A8 processor can be used to implement OPC UA server and client. The aim is also to develop a specific AM335x Sitara driver that allows fast transfer between the two processors (PRU – ModbusE master / slave and ARM Cortex A8 RISC - OPC UA server and client). However, Cortex M4 and M7 does not have sufficient resources to implement high - performance OPC UA server and client.

The solution presented in this paper allows the implementation of the ModBus Extension client/server protocol with better communication channel usage performance than those previously presented (higher than 70%), using real time PRU (Programmable Real-Time Units) processors. At the Industrial Internet of Things (IIoT) connection, the 1MHz ARM Cortex A8 processor offers superior performance to the M4, M7 cortex for implementing OPC UA server and client specifications (specifications supported by Industry 4.0), or other IIoT middleware such as MQTT, AMQP, REST, DDS, CoAP (6LowPAN), etc.

4. Conclusions

By using of the Sitara AM335x processor, it is possible to implement both the acquisition cycle and the implementation of OPC UA server and client. The PRU unit will cover the acquisition cycle and ARM Cortex A8 processor will cover the OPC UA server and client. And in this way, we aim at obtaining performance for the use of communication channels greater than 70%.

5. Future Work

The comparative studies with the other implementations mentioned earlier in the Related Work section are to be done. Also, on the next step is to implement the OPC UA server and client on the ARM Cortex A8 processor, and to study the performance of the implementation.

6. Acknowledgement

“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

Cena, G.; Cereia, M.; Bertolotti, I. C. & Scanzio, S. (2010). *A MODBUS extension for inexpensive distributed embedded systems*. pp. 251-260. http://dx.doi.org/10.1109/WFCS.2010.5548625.

Găitan, V. G. & Zagan, I. (2019). *Rețele industriale locale – Modbus Extins/Local industrial networks – Modbus Extension*. Suceava: Editura Universității "Ștefan cel Mare".

Gaitan, V. G. & Zagan, I. (2021). *Experimental Implementation and Performance Evaluation of an IoT Access Gateway for the Modbus Extension*. https://doi.org/10.3390/s21010246.

Modbus.org-1. (2006). *MODBUS Messaging on TCP/IP Implementation Guide V1.0b*. Retrieved from https://modbus.org/.

Modbus.org-2. (2006). *MODBUS over Serial Line Specification and Implementation Guide V1.02*. https://modbus.org/.

Persechini, M. A. & Jota, F. G. (2013). *Centralized and distributed control architectures under foundation fieldbus network*. 52(1), pp. 149–161. http://dx.doi.org/10.1016/j.isatra.2012.09.005.

Silva, C. R. & Silva, F. A. (2019). *An IoT Gateway for Modbus and MQTT Integration*. http://dx.doi.org/10.1109/IMOC43827.2019.9317637.

Xiang-li, Z. (2009). *Study on communication scheduling of fieldbus*. pp. 565–570. http://dx.doi.org/10.1109/CCDC.2009.5194952.