Synchronized Micro-Controllers-based Data Acquisition System for Energy Plants using Modbus Protocol

Maëva Courcelle
Institute of Technical Physics (ITEP)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany

Dustin Kottonau
Institute of Technical Physics (ITEP)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany

Giovanni De Carne
Institute of Technical Physics (ITEP)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany

***Abstract*—Real-time simulators are powerful devices able to deal in real-time with input and output signals. Among others, some extensions provide communication protocol implementation possibilities, that are often expensive and could have utilization limitations. Using a programmable micro-controller can be a solution for the price and flexibility of the communication. However, the micro-controller must be able to provide reliable output in terms of time and value. This paper proposes two synchronized Modbus communication interfaces with two real power devices. The system created enables the data acquisition of variables stored in the Modbus registers of the devices.**

***Index Terms*—Data acquisition, power plant, communication, synchronization, micro-controller, real time**

I. Introduction

Testing of new solutions is always considered a challenge, particularly for Power Hardware In the Loop (PHIL) simulations, that involve a reliable data acquisition system and work with a real power plant [1], [2]. However, needed data for the testing are sometimes not directly accessible with common data acquisition software. The device variables are measured internally by sensors, that store them in the internal memory of the system. To read the value of these hidden variables, a communication system able to communicate with the device storage is needed. Traditional SCADA systems are able to perform this task, but they are complex and expensive to build [3].

Real-time simulators provide solutions to build communication interfaces, able to use a device-specific communication protocol to read data in the power system [4]–[7]. However, each communication protocol requires the user to buy additional communication licenses and sometimes the corresponding hardware interfaces, as shown in figure 1. An extension of the data acquisition system could be expensive in terms of hardware for laboratories. Moreover, some communication synchronization limitations have been noticed, if we want several master-slave communications to be synchronized.

The concept of the synchronized micro-controllers-based data acquisition system is described in this paper. The two real power devices used for the experiments require the Modbus RS485 communication protocol. Two Raspberry micro-controllers have been programmed to communicate with these two devices, but the general idea can be applied to other devices using different communication protocols. The system is also flexible regarding the sample group of communication possibilities and is easily scalable in varying the number of slaves or changing the communication protocol. Moreover, micro-controllers allow an easy synchronization with a real-time simulator, which enables the data acquisition on different slaves to be synchronized. Each micro-controller can record the read data and correct eventual time errors during the process. It becomes also a reliable data acquisition system.

The paper is structured as follows: the first section depicts experimental set-ups and Modbus parameters initialization. The second section deals with the proposed algorithm for data acquisition synchronization, whereas the last section shows the result of a simulation.

II. Experimental setup

This section describes the experimental setup and the developed data acquisition system. The final goal of the project is to acquire data from the micro gas turbine (MGT) and the cooling system (CS) to develop a real-time model of the micro gas turbine.

A. Experimental set-up and data acquisition system

The experimental set-up depicted in figure 2 consists of a 3.2 kW electric and 15.6 kW thermal power combined-heat-and-power MGT, where gas is supplied by external fuel tanks [8], [9]. The cooling is performed by an external water-based cooling system. In a first step, the MGT and the CS are controlled via a real-time simulator, whereas the Raspberry PI micro-controllers are used as masters in the Modbus communication with real devices (slaves). Both devices have internal registers, where sensor values and other useful variables are stored, which can be read with the Modbus RS485 protocol only.

This work was supported by the Helmholtz Association under the program "Energy System Design" and within the Helmholtz Young Investigator Group "Hybrid Networks" (VH-NG-1613).

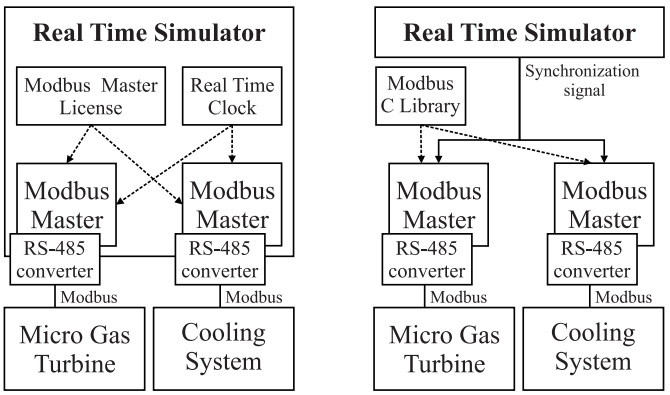


Fig. 1: Communication interface of a data acquisition system based on (a) the real time simulator or based on (b) two synchronized micro-controllers.

To read these registers and store the values, Raspberry PI micro-controllers have been connected to the power plant through 3-meter wide serial RS485 cables, where the voltage difference of the two wires is related to a one or zero bit. Since the CS and the MGT do not have the same Modbus communication parameters, sending a synchronized broadcast message from one single master is not possible. Thus, both data acquisition systems need to be synchronized because collected data aim to be involved in the real-time modeling of the MGT, which includes data from the CS [10]. Master synchronization occurs also through two signals sent from the real-time command unit to both micro-controllers. A start signal (step signal) is sent to start the data recording. The real-time simulator sends a second signal (square wave) to the micro-controllers to enable the command synchronization of both master units, and also to set a fixed real-time step for the acquisition. To validate the results, LabVIEW measurements have been used for selected MGT variables, and they will be taken as reference values, enabling a realistic performance assessment of the Modbus data acquisition system.

B. Synchronization setup

Communication with real devices is not instantaneous and read data can not continuously be sent back to the master. A fixed measurement rate is also required for data processing. However, each device has its own communication time, and the measurement rate should be identical for both communications to enable the data acquisition synchronization.
The synchronization requires the knowledge of the communication behavior of the real devices, and particularly the maximum time needed for the master to receive an answer. If the communication time of one device is faster than the other, the request sending rate should be set according to the slowest device. Thus, the following time length information is required: request transfer time, answering delay, and answer transfer time. The communication time with the CS and the MGT is calculated by summing up these three parameters. To assess the performance of the communication with real devices, the communication time influence parameters have been compared with those of simulated devices. A Raspberry PI micro-controller simulates either the MGT or the CS, and another Raspberry-PI micro-controller is used as a master in both communication with the real and the simulated devices. In the following parts, the Modbus communication with real devices has been analyzed and compared with the communication with the simulated power plant.

1) Data length: The first parameter impacting the communication time is the length of the message. For the same transmission speed, the lengthier the message, the more time needed for transferring the message.
The Modbus message is made up of four parts: Slave ID – Function Code – Data – Error [11], [12]. Furthermore, the convention used for the devices requires one start-bit before each byte and one stop-bit after each byte. Each message part has a fixed length except the data value part. In the experimental setup, the reading request length is always 60 bits, whereas the answer length is $20 \cdot N + 50$ bits. Bits distributions in Modbus messages have been sum up in tables Ia and Ib.

ID	**FC**	**Data** Address	**Data** N	**CRC**	**Total length**
8_{+2}	8_{+2}	8_{+2}	8_{+2}	16_{+4}	**60 bits**

(a) Request

ID	**FC**	**Data** N	**Data** Value	**CRC**	**Total length**
8_{+2}	8_{+2}	8_{+2}	$16 \cdot N_{+4 \cdot N}$	16_{+4}	$\mathbf{20 \cdot N + 50}$ **bits**

(b) Answer for N data read

TABLE I: Bits distribution in Modbus messages.

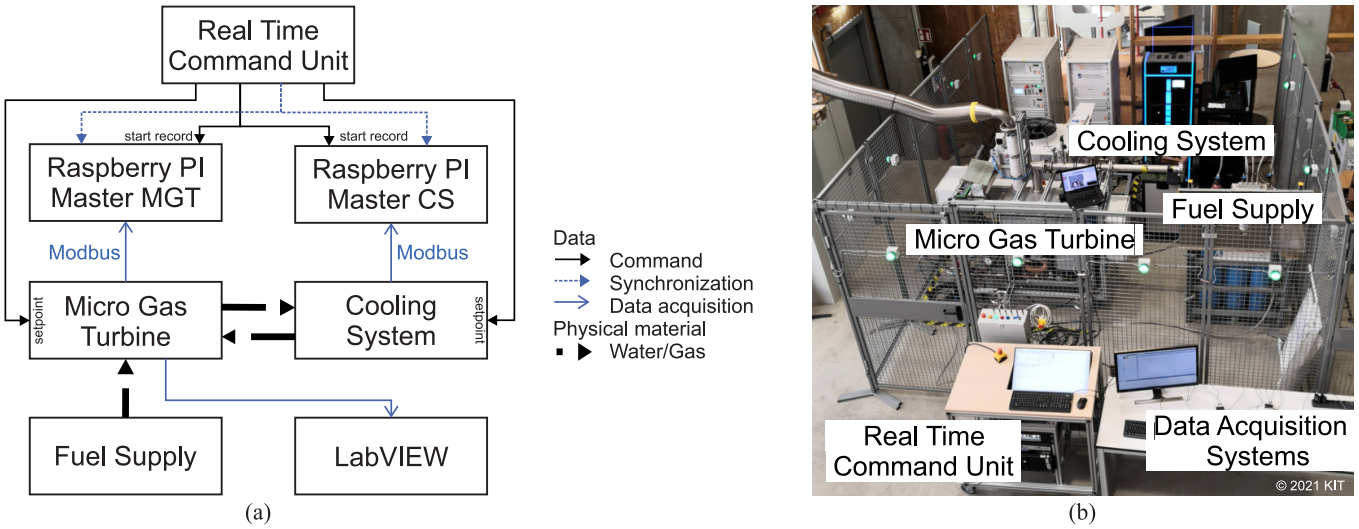
The transmission time of the messages depends on the speed of bit transmission (baud rate). The MGT uses a baud rate of 19200 bits/s [13]. Therefore, it can transmit answers containing N data in :

$$T_{data19200}(N) = \frac{20 \cdot N + 50\,bits}{19200\,bits/s} = 1.04 \cdot N + 2.60\,ms \quad (1)$$

As for the CS, that uses a 9600bits/s baud rate [14], a N-data message will be transmitted within :

$$T_{data9600}(N) = \frac{20 \cdot N + 50\,bits}{9600\,bits/s} = 2.08 \cdot N + 5.21\,ms \quad (2)$$

These theoretical functions describing the transmission time of a Modbus message containing N data, considering



(a) (b)

Fig. 2: Description of (a) the communication system and (b) energy plant overview.

the transmission rate of the real plant, have been validated with measurements using an oscilloscope. Figure 3 shows these results since black lines (theory) and colored lines (real systems) are merged.

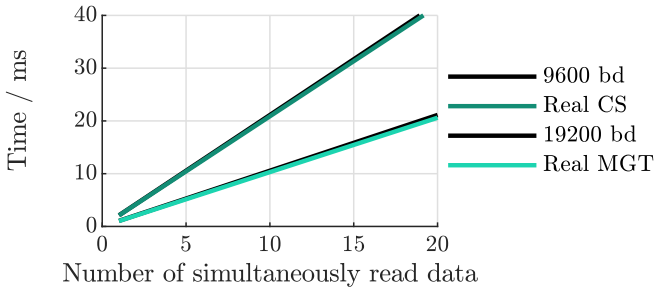


Fig. 3: Message transmission for the cooling system (9600 bits/s) and for the micro gas turbine (19200 bits/s).

It turns out that both devices have a maximum number of data N_{max}, that they can send simultaneously in one single message. For the cooling system, internal registers allow a maximum of $N_{max} = 20$ data (documentation), whereas the MGT can afford to read and send up to 80 data in the same message. $N_{max} = 80$ has been found experimentally. It is not a limit of the internal register due to the constructor's implementation, but above his limit, the communication is not reliable anymore, because the MGT sometimes does not reply to the request.

2) Delay: To know the complete communication time between a master and his slave, not only the baud rate and the message length have to be considered, but also the time needed by the slave to answer: the answering delay. The time required is actually very small ($\backsim$ 400 μs) and unpredictable in the studied case with the two micro-controllers communication. The delay between the request and the answer from the master point of view has been measured, using its transmission (Tx) and receiving (Rx) pins. Results can be seen in table II. Then, the delay in the communication with the physical plant has been measured and compared with the delay in a Raspberry-to-Raspberry communication. It has been observed that the delay in a Raspberry-to-Real system communication is much larger than the delay in a Raspberry-to-Raspberry communication (factor 100). Figures 4 and 5 depict the delay difference between the real systems and the simulated one, regarding the number of data requested. Dash lines represent the measurement ranges.

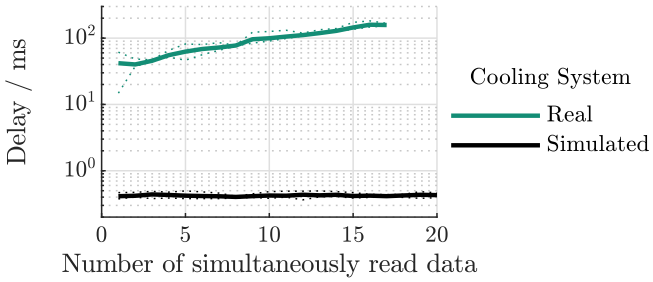


Fig. 4: Communication delay with the cooling system. Comparison between the real system (green) and the simulated one (black).

3) Total communication time and synchronization: The total communication time is defined as the time between the start of the sent request and the end of the received answer from the master point of view ($request + delay + answer$). Figures 6 and 7 show in green the total communication time for both real slaves.

Results of sections II-B1 and II-B2 enable to calculate the total communication time, drawn in previous figures. A summary can be found in the table II. Firstly, the last row of the table (communication time) highlights the discrepancy between the time that real devices need

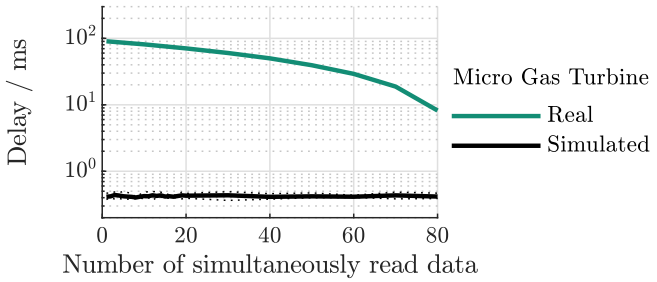


Fig. 5: Communication delay with the micro gas turbine. Comparison between the real system (green) and the simulated one (black).

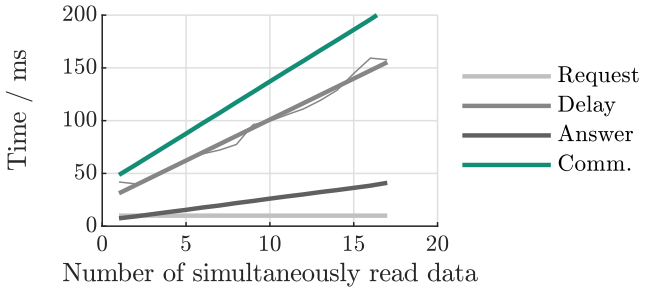


Fig. 6: Total communication time (green) with the cooling system, as sum of the request (light gray), approximation (thick gray) of the delay (thin grey) and answer (dark grey).

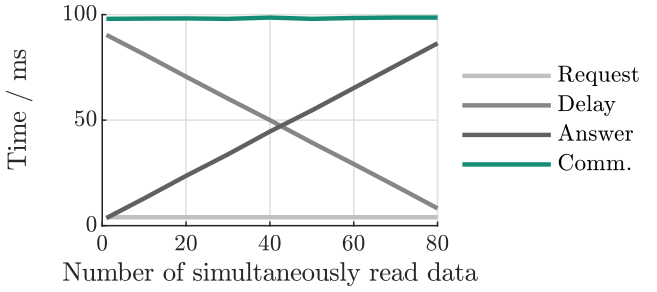


Fig. 7: Total communication time (green) with the micro gas turbine, as sum of the request (light gray), delay (gray) and answer (dark gray).

to communicate through Modbus protocol and the time needed by Raspberry-to-Raspberry systems to communicate. Communication with real plants takes much longer because of the delay. Secondly, the difference in the communication time between both real systems (highlighted in gray in the table II) can be pointed out. The total communication time of the MGT does not depends on N anymore, contrary to the CS.

As it can be observed, the final communication time of real devices are various, and especially the behaviors of measured delay for the MGT and CS are pretty different. Figures 8a and 8b summarize the difference between the Modbus communication of these two devices. R_N (resp. A_N)

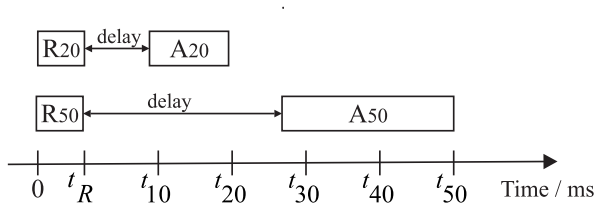
Cooling System	**Raspberry**	**Real**
Message length (time)	$2.08 \cdot N + 5.21\,ms$	
Delay	$\backsim 418\,\mu s$	$7.76 \cdot N + 23.42\,ms$
Communication time	$2.08 \cdot N + 12.92\,ms$	$9.84 \cdot N + 33.84\,ms$

(a)

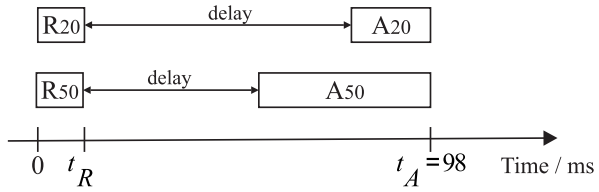
Micro Gas Turbine	**Raspberry**	**Real**
Message length (time)	$1.04 \cdot N + 2.60$ ms	
Delay	$\backsim 422\,\mu s$	$91.48 - 1.04 \cdot N\,ms$
Communication time	$1.04 \cdot N + 3.02\,ms$	$97.72\,ms$

(b)

TABLE II: Summary of communication time parameters of (a) the cooling system and (b) the micro gas turbine.



(a) Cooling System



(b) Micro Gas Turbine

Fig. 8: Modbus communication differences between (a) the cooling system and (b) the micro gas turbine.

is the request for N-data reading (resp. answer containing N data). For the cooling system, the bigger the number of simultaneously read data, the bigger the delay. The times t_N, at which the answer message containing the N data is received, are proportionally distributed, as previously shown in figure 6. However, the MGT total communication time is insensitive to the number of transmitted data. The answer is always received by the master at the same time $t_A = 98\,ms$, no matter the number of data requested, as detailed in figure 7.

C. Choice of the synchronization period

For the experiment, only one single CS data needs to be read at a time. That means each answer will be received within $50\,ms$. Regarding the MGT, less than 50 data will be transferred in the answer for the system needs. The communication time required to get these data is fixed and equal to $98\,ms$. To synchronize the sending of the requests to the MGT and the CS, the command unit sends a square wave of $10\,Hz$ to

enable data of the MGT and the CS to be picked up every $100\,ms$.

III. Synchronized micro-controllers-based data acquisition system

The data acquisition systems should be synchronized to handle the collected data from the MGT and from the CS in the same way.

A. Synchronization requirements

The proposed synchronized micro-controllers-based data acquisition system uses two C-programmed Raspberry PI micro-controllers, relying on a general Modbus protocol implementation library in C to communicate with the slaves [15]. The communication requests are sent from the masters on a fixed time-step defined by the square wave, which is transmitted by the real-time control unit.
This pulse signal has a $100\,ms$-period chosen in the previous section, and its duty cycle has been set to $30\,\%$, which enables the master to send one and only one request during each signal period. For each new period, a new request is sent, meaning that one period corresponds to one read data set. The read values are also stored in a file with its corresponding measurement number index. When the chosen number of measurements has been reached, the record stops, and data can be downloaded by the user.
In case of a larger communication delay (e.g., greater than $100\,ms$) due to the slave answering time, the micro-controller will miss some pulses sent by the real-time simulator. This leads to a wrong index of measurement, and thus to a wrong time assignment in the data record. To enable reliable data acquisition, we need to manage these errors by detecting when the delay happens and assigning the right measurement index to the missed values. Figure 9 shows an error-less communication (a) and a communication with larger delay and time correction (b). Rn (resp. An) corresponds to the n^{th} request (resp. answer). If the answer exceeds the deadline (as shown in green), the next request can not be sent and data will be missed. The algorithm should be able to keep the correct time assignment for the next records. This process is the purpose of the next section.

B. Proposed data acquisition system

To avoid wrong time assignment, the algorithm checks the internal clock of the micro-controller and compares the communication time with the communication rate limit of the synchronization ($10\,Hz$). To do so, the clock time in milliseconds at the start of the record and the clock time after receiving the answer from the slave are read. Because of the synchronization with the real-time simulator, each answer must be received before the next pulsation, i.e. one data reading should be done every $100\,ms$. The difference in milliseconds between the time after the receipt of the answer and the time at the beginning of the record should be equal to a hundred times the index number of the read data. If it is not the case, each missed step is replaced by the previous reliable

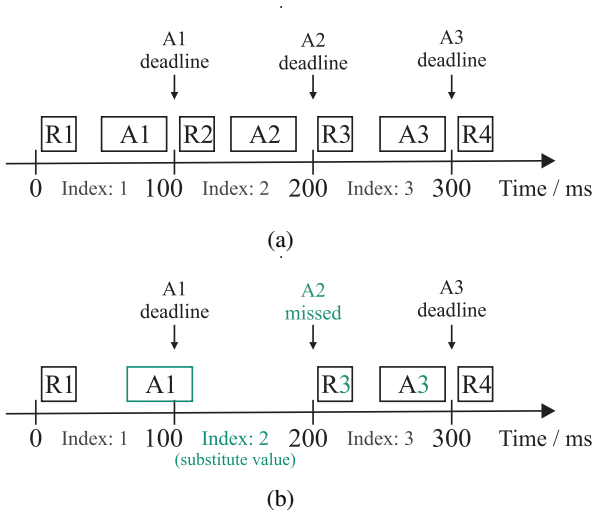


Fig. 9: Communication with (a) respected and (b) exceeded deadlines.

value recorded, and the measurement index is set to the time difference, divided by the factor 100. Figure 10 explains the process, where the error correction part has been highlighted in green.

1) Initialization: The communication parameters like the baud-rate or the number of starts or stop bits are set, and the micro-controller is set as a Modbus master and becomes ready to send requests. The user can choose the number N_{chosen} of value to record, where the total duration of the record will be $N_{chosen} \cdot 100\,ms$.

2) Start of the record: The micro-controller waits for the signal of the real-time simulator to start the record. While the signal is high, the data acquisition goes on.

3) Data acquisition: Data acquisition requires the data value and the record time. The time is known thanks to the $100\,ms$ period pulse synchronization with the real-time simulator. Thus, the time in milliseconds should be equal to 100 times the index number of recorded data. This information is used to check if some records have been missed.

- *Record initialization:* At the first iteration of the while loop, the record parameters are initialized. The index number N of data recorded is set to zero. Moreover, the variable t_0 is set to the current time of the internal clock of the micro-controller. The value of this variable will be used to detect an error in the data acquisition process.
- *Communication:* In the communication part, the micro-controller sends a read-request to the slave and waits for the answer. The index N of the record is then incremented. During the communication, the controller only focuses on it, and the device can not react anymore to eventual other signals coming from the command unit.
- *Error check:* When the answer has been received, a second time variable t_1 is set to the current time of the internal clock. Then, the number of measurement N is compared with the variable N_{ref}, corresponding to the

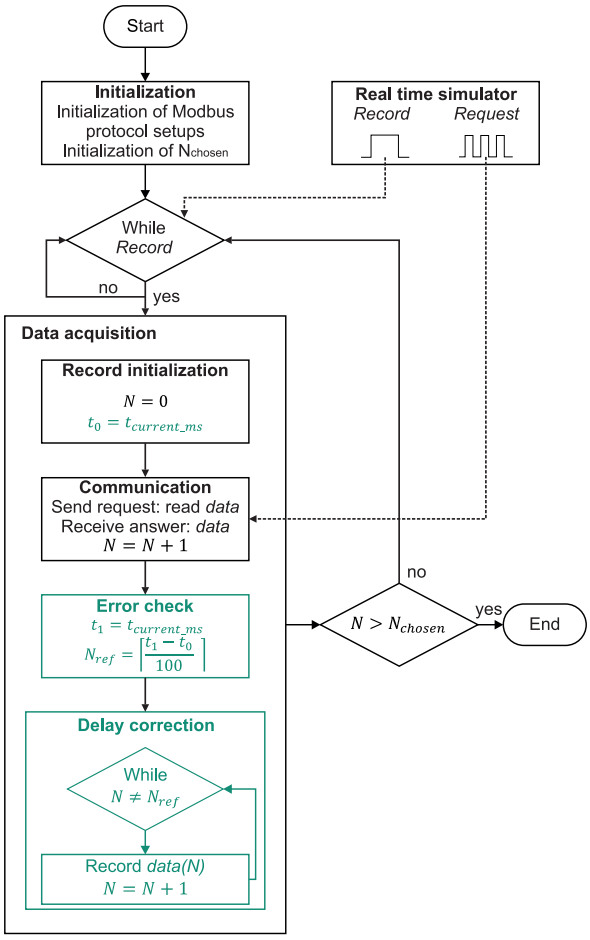


Fig. 10: Description of the data acquisition and error correction process.

ceiling integer of the time difference between t_0 and t_1 divided by hundred milliseconds. If the record index is equal to this reference number of measurement, no delay during the communication occurred, so the data value and its corresponding measurement index N can be stored in the microcontroller memory. If these numbers are different, then a correction in the record file is needed.

- *Record correction:* If an error is detected, the record file has to be corrected. As the number of measurements is not equal to the reference, the previous reliable data is stored instead of each missed data.

4) Loop continuation: If the number of recorded values has reached the number of measurements chosen, the record stops. Else, the record goes on.

IV. Results

The two micro-controllers can acquire the data from both slaves in a synchronous way and store recorded values in their intern memory. The data can therefore be sent directly to the command unit or be collected after the end of the experiment. Sections IV-A and IV-B deal with recorded data, which have been picked up after the experiment.

A. Error correction

To validate the performance of the proposed strategy, figure 11 shows the recorded Modbus data of the MGT grid RMS current (in green), which have been compared with LabVIEW reference measurements (in grey). Tests have been done about $2h45$ ($\simeq$ $10000\,s$). Figure 11a depicts the data acquisition without the time error check explained in the previous section. A delay has been accumulated at different times during the communication with the MGT. However, at the end of the record, it becomes impossible to determine when the error occurred. Even if the micro-controller sends a request at a fixed time-step defined by the real-time simulator, the master waits some time for the MGT to answer the request. However, if the answer exceed the $100\,ms$-deadline, the master will not receive the next signal from the control unit. As a consequence, the next request will not be sent, and some data will be missed. Some very long random response delays (or sometimes communication breaks) have been observed during the communication with the MGT, but not during the communication with the CS.

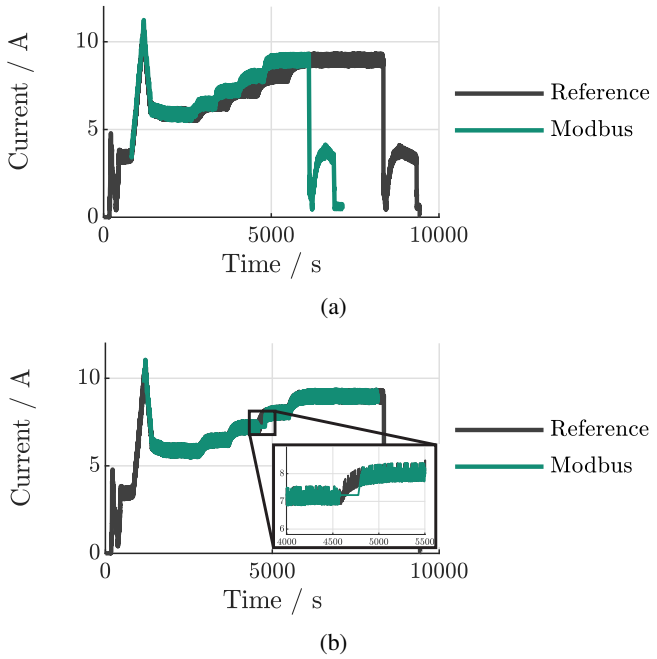


Fig. 11: Grid RMS current measurements of the MGT (a) without and (b) with error correction

The micro-controller-based synchronized data acquisition system enables to correct the time assignment. Figure 11b) shows the result of the error correction during the communication process. The Modbus stored measurements (in green) are in accordance with the reference values measured with the LabView acquisition system (in grey). Delays in the measurements do not affect the curve aspect anymore so Modbus measurements fit reference data. Moreover, the error correction can be observed at around $4500\,s$. The Modbus

measurements are constant for some minutes, because of a communication break. Some other shorter breaks have been noticed during the record, but they are only visible by zooming on the data plot. During the break, missed data are replaced by the previously recorded data value. When the communication becomes normal again, the data acquisition system can continue the record without timing error.

B. Synchronization of the data acquisition systems

The aim of the synchronization of the two data acquisition systems is to draw a parallel between measurements of the MGT and measurements of the CS. Figure 12 shows the data acquisition of the fan speed (in $\%$) of the cooling system and the grid RMS current of the MGT. The synchronization of the acquisition can be guessed, as the decrease of these two variables well occurs at the same time, while the MGT has been turned from generator operation to motor operation.

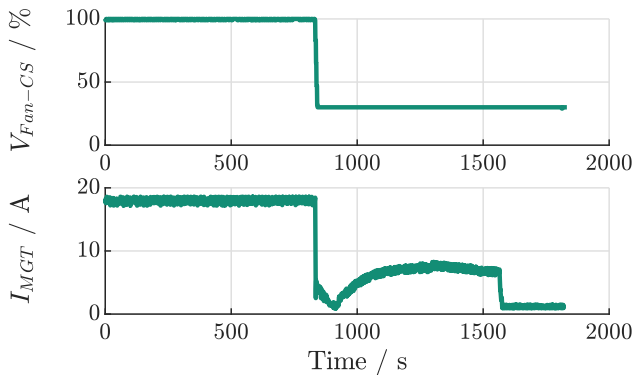


Fig. 12: Synchronized data acquisition.

C. Closed-loop for future experiments

The data acquisition systems are well synchronized and the data are reliable. The variable values are then sent back to the real-time simulator, instead of being stored in the internal memory of a micro-controller. The two PWM (Pulse-Width Modulation) channels of the Raspberry PI micro-controllers have been turned on, and their duty cycles are set proportionally to the new recorded value during the Modbus communication. A simple low-pass filter converts the PWM signal into an analog signal, which is directly sent to the real-time simulator analog input. The control loop has been closed, and the MGT and the CS can be studied without real-time simulator extension or licenses. For further application, more PWM channels are required, because two signals may not be enough to visualize the state of a real energy plant. Some micro-controllers have more analog output channels. Thus, the improvement of the data acquisition system for the use in real-time simulation consists of the choice of the micro-controllers.

V. Conclusion

It has been proved that real-time communication with physical systems is possible using a simple programmable micro-controller. The collected data can be used for further work, thanks to the synchronization of the data acquisition systems. The sampling rate of $100\,ms$ depends on the used slaves, whereas the proposed algorithm could be able to perform the same task a hundred times faster. The data acquisition system is also a low-cost and easy-to-implement solution, which can be applied to different slaves and other communication protocols.

References

[1] A. Benigni, T. Strasser, G. De Carne, M. Liserre, M. Cupelli, and A. Monti, "Real-time simulation-based testing of modern energy systems: A review and discussion," *IEEE Industrial Electronics Magazine*, vol. 14, no. 2, pp. 28–39, 2020.

[2] C. S. Edrington, M. Steurer, J. Langston, T. El-Mezyani, and K. Schoder, "Role of power hardware in the loop in modeling and simulation for experimentation in power and energy systems," *Proceedings of the IEEE*, vol. 103, no. 12, pp. 2401–2409, 2015.

[3] L. D. Cuayo, J. Kerbee Culla, J. Gualvez, S. E. Padua, and R. John Gallano, "Development of a wireless microcontroller-based scada rtu," in *TENCON 2018 - 2018 IEEE Region 10 Conference*, 2018, pp. 2566–2570.

[4] M. D. Omar Faruque, T. Strasser, G. Lauss, V. Jalili-Marandi, P. Forsyth, C. Dufour, V. Dinavahi, A. Monti, P. Kotsampopoulos, J. A. Martinez, K. Strunz, M. Saeedifard, X. Wang, D. Shearer, and M. Paolone, "Real-time simulation technologies for power systems design, testing, and analysis," *IEEE Power and Energy Technology Systems Journal*, vol. 2, no. 2, pp. 63–73, 2015.

[5] X. Guillaud, M. O. Faruque, A. Teninge, A. H. Hariri, L. Vanfretti, M. Paolone, V. Dinavahi, P. Mitra, G. Lauss, C. Dufour, P. Forsyth, A. K. Srivastava, K. Strunz, T. Strasser, and A. Davoudi, "Applications of real-time simulation technologies in power and energy systems," *IEEE Power and Energy Technology Systems Journal*, vol. 2, no. 3, pp. 103–115, 2015.

[6] P. K. N. H. Marios Maniatopoulos, Dimitris Lagos, "Combined control and power hardware in-the-loop simulation for testing smart grid control algorithms," vol. 11, no. 12, pp. 3009–3018, 2017.

[7] A. Bani-Ahmed and A. Nasiri, "Development of real-time hardware-in-the-loop platform for complex microgrids," in *2015 International Conference on Renewable Energy Research and Applications (ICRERA)*, 2015, pp. 994–998.

[8] EnerTwin. Enertwin, accelerating energy transition. [Online]. Available: https://enertwin.com/

[9] MTT. Micro turbine technology. [Online]. Available: https://www.mtt-eu.com/

[10] G. Coviello, G. Avitabile, and A. Florio, "The importance of data synchronization in multiboard acquisition systems," in *2020 IEEE 20th Mediterranean Electrotechnical Conference (MELECON)*, 2020, pp. 293–297.

[11] Modbus. Modbus application protocol specification v1.1b. [Online]. Available: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

[12] ——. Modbus organization website. [Online]. Available: https://modbus.org/

[13] M. T. Technology, "Enertwin installationshandbuch," pp. 70–84, 2018.

[14] Guentner. Interface specification modbus gmm interface specification-modbus gmm v3.2. [Online]. Available: www.guenter.de

[15] S. Raimbault. Libmodbus c-library. [Online]. Available: https://github.com/stephane/libmodbus