MODULAR MULTI-COMPONENT ACS WITH BUILT-IN ACTIVITY AND MEASUREMENT DATA STORAGE

ALDIS PECKA, VITALIJS OSADCUKS

Faculty of Information Technologies, Latvia University of Agriculture, Latvia
apecka@gmail.com

Abstract: *The paper describes the prototype of a modular multi-component ACMS (Automatic Control and Monitoring System). This prototype gives a possibility of simultaneous process monitoring and control by natively allowing logging the states of measurement and controlling actions to the local memory or remote database for further analysis. The prototype is a modular system consisting of several modules with distinctive functionality. These microcontroller-based modules, that can be interconnected using industry-standard RS485 and IEC-870 communication networks, are developed: digital I/O, pulse counting and measuring, relay, temperature measuring modules and central control module with SD memory card and Ethernet interface with TCP/IP networking. Besides of built-in process logging feature the advantages of the proposed ACMS over traditional programmable logic controllers are simple logic expressions based language where user without a prior knowledge in programming can create his own control algorithms; and a possibility to directly connect ZACwire interface devices, e.g., temperature sensors with digital output. This allows building cost-effective multi-point temperature monitoring and controlling systems.*

Keywords: ACMS, PLC, microcontroller.

Introduction

Industrial PLC's (Programmable Logic Controllers) are designed to be robust and reliable control units for use in harsh environments in the applications with high processing performance and safety requirements, where in the case of a malfunction there is a risk of damage to equipment or injury to personnel. These units can directly interface to commonly used (Bayindir, 2010) control system sensing components with pulse or analog voltage/current outputs as well as resistive sensors and thermocouples, actuators, e.g. relays and frequency converters and use industry-standard communication networks (Osadčuks and Galiņš, 2009). The PLCs are designed to be programmed by electrical and automation engineers, that are familiar with relay-based automation (Galiņš, 2008). Although the PLCs are also used outside industry environment to control processes in, e.g. agricultural facilities, Smart Houses, to control processes and collect data in scientific research projects in agriculture, food, forestry etc.; in many cases they are not suited to this purpose very well. The drawbacks of PLCs in such cases are relatively high cost of base system and expansion units, problems with interfacing to various types of sensors that are not commonly used in industry, relatively high power consumption for battery applications. Most types of PLC's do not contain built-in functionality for software activity, actuator and sensor data logging, which is one of the most important aspects for scientific experiments (S. Da'na, 2007).
On the other hand task-specific embedded systems could be used. In such cases, specialists with knowledge in embedded system design and development are needed, as well as a prolonged process of equipment testing which results in comparatively greater expenses (Bucur, 2010).
The aim of this work is to design a prototype of intermediate solution, which could be used to target various sensor measurement and simple automatic control applications. Emphasis is made on data logging of the system operation.

Materials and methods

The architecture of the proposed automatic control system is designed hierarchically similar to SCADA systems. In the hierarchical structure of the industrial automation (Кругляк, 2002) the proposed system goes along with segment and manufacturing automation level. Examined solution is modular and works using master - slave principle. One central master device is connected to the slave devices using high-speed digital data exchange network. Master performs functions of the controller of the process by executing the algorithm that has been set and stored in it, but slave devices are used for reading variables of the process and performing control actions. Master sends commands for execution or data requests, and the slave responds with the approval of the execution or with the requested information. In the current implementation several groups of slave devices equipment have been considered: temperature measurement, digital inputs reading, pulse counting and control of relay outputs.
Asynchronous communication is being used to exchange data between master device and slave devices and slave devices could be read at any time. Sequence diagram for transactions with a slave device in general case is shown in Fig. 1.

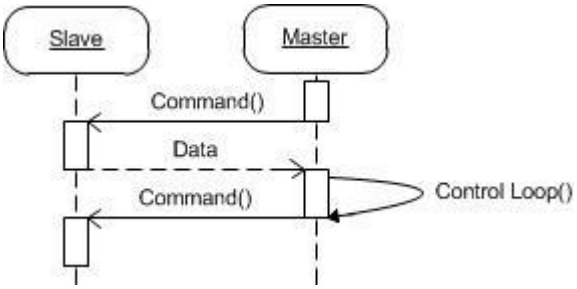


Fig. 1. **Data exchange between master and slave devices in general case**

For all types of devices in the general case the master-slave data exchange network initialization and data exchange procedure consists of the following steps.

1. The master contains a list of equipment of all known slaves. All physical devices within the network are identified by their unique MAC (Media Access Control) addresses. At the start of network initialization the master device successively dispatches test packets for each slave, if the result is a response it is considered that the machine is connected to a common network and is running.
2. The slave device default settings are sent after network initialization, such as a temperature sensor calibration values, the digital port mode or relay starting position, notice that the slave machine configuration can also be changed at any other moment later.
3. Then the slave devices are being pooled to get sensor reading data on the state of the process for use as the master control algorithm inputs.
4. Master device calculates a step of control loop and generates control action signals; the master logs actual measurements and the control signals data to a DB.
5. Control signals are then sent to the slave devices in a form of the commands and the cycle continues with the third point.
6. If slave does not provide answers to the master request in 300ms (in contrast to IEC-870 standard, where it depends on the communication baud rate) then the packet is sent again.

Discrete input block (pulse unit) is designed for reading of digital logic state and for counting status changes (pulses) at its inputs. The current prototype has eight inputs, which can be configured separately for either for logical level reading or pulse counting mode. The pulse counter at each input is 16 bit wide. This block can be used to count data form energy meters, flow meters, anemometers etc.
After master has configured operation mode of discrete input block port, data is exchanged in the following steps (see Fig. 2):

1. Digital input block reads the states of all the inputs and records them into transmission buffer, at the same time the state transition from the high logical level to the low at the specified input ports is fixed and corresponding port pulse count register is incremented.
2. When the master device transmits the reading request command, digital input block prepares a data packet and sends it to the master.
3. If the master has received the answer, it sends confirmation that the data is successfully received to discrete input block. After receipt of this package digital input block restores digital input status buffer and accumulated pulse value, if such a packet is not received, the discrete input unit continues to count pulse and at the next master request it sends a total count of pulses. Consequently, it provides integrity of transfer data and the digital input block will not count high to low transitions continuously, but only until the next master-slave transaction, thereby eliminating the need of back-up power supply or non-volatile memory and simplifying the system.
4. When the master device receives data from the digital input block, it can use the received data as input for control algorithm and also stores the data in a database.
5. If the master cannot read data from slave device until the pulse counters are overflown, then in the next data packet there is error code encapsulated in it. When master device reads the packet, it is notified about data overflow.
6. The master can request new data.

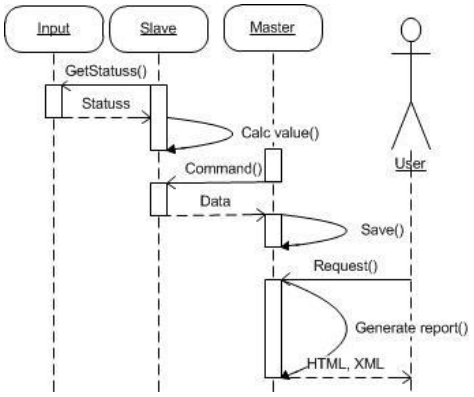


Fig. 2. **Data exchange between master and digital input blocks**

Temperature measurement block is intended to retrieve temperature data. In the current implementation to this unit can be connected up to 20 integral temperature sensors with digital output (ZACwire interface). All sensors can be read within 2 sec. After the master has configured which sensors to be read the temperature measurement in the slave is being carried out using following algorithm (see Fig. 3):

1. Cyclic reading of all temperature sensors.
2. When the request command is received from master device, temperature measurement block prepares data packet and sends it to the master.
3. When the master receives data from the temperature sensor unit, and stores them in a database and sends acknowledgement to the slave.
4. The master ser can request new data.

Sensor reading of the temperature-measuring unit is represented as a signed integer, but the master device converts it into a float type before saving into the database. Experimenting with the prototype equipment, the need of the sensor state determination arose, if measurement is not received due to some reason. Error codes that are applied on an individual digital output ZACwire sensor were introduced: no connection to the sensor (sensor response timeout), short between leads of a sensor, parity does not match and framing error (wrong bit type or count).

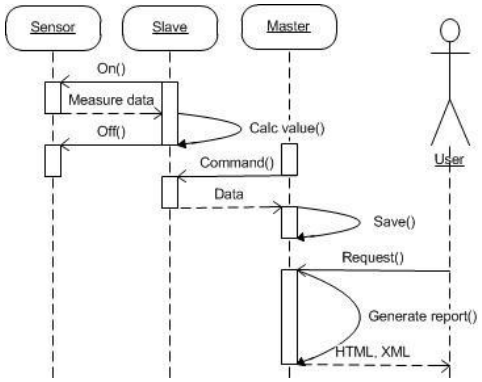


Fig. 3. **Data exchange between the master, the temperature measuring unit and integrated temperature sensors**

Relay block is provided as output device or actuator Fig. 4. The current implementation of a single block has 4 relays. Like other slave devices relay unit receives from the master the initial configuration (initial state of the relay, watchdog timer value and the default state of the relay) and operates using the following algorithm.

1. Waiting for the master commands which relay should be either turned on or turned off.
2. Receiving the master command (new relay state) setting the appropriate relays and transmitting current conditions back to the master.
3. If relay position command request is received, slave device returns current positions of all relays.
4. User can request new data.

Relay block has built-in safety mechanism, if there is a failure to receive data from the master device. In this case, user can set a watchdog timer and when no data received, the relay is set to default conditions so that control system would not produce emergency situations.

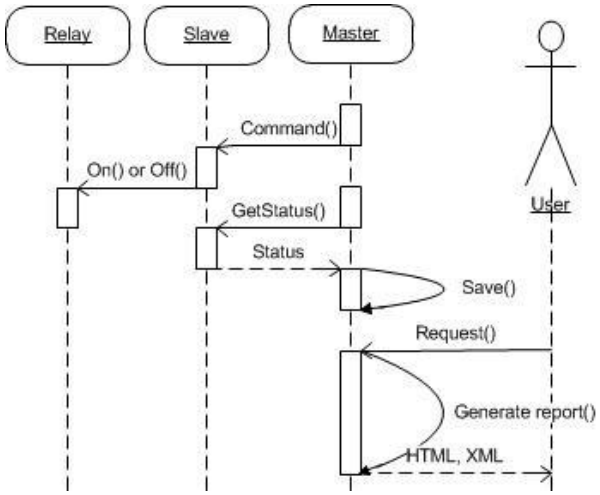


Fig. 4. **Data exchange between master and relay block**

Results and discussion

Based on the discussed theoretical control and monitoring aspects of the modular system, a prototype was developed. Equipment of the prototype, architecture and technological solutions that where used are shown schematically in Fig. 5.

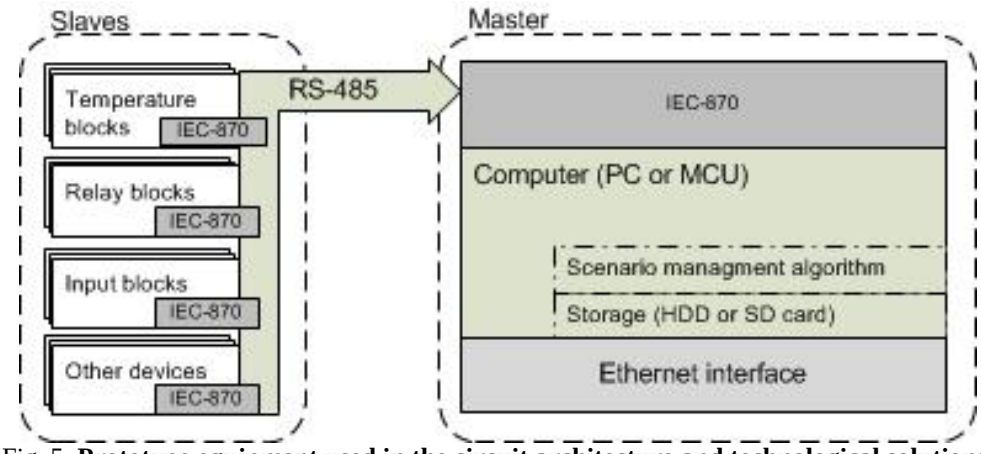


Fig. 5. **Prototype equipment used in the circuit architecture and technological solutions**

Communication is carried out via RS-485 physical interface, which can provide up to ~ 1 km long line of data at a rate of 9.6 kbod. IEC-870 protocol is used as OSI (Open Systems Interconnection model) model data link layer. The prototype currently has three device types: temperature measurement unit, relay unit and both digital input and pulse counting blocks in a single unit. Temperature unit works with TSIC506 integrated sensors with the ZACwire digital output. Up to 20 sensors can be connected to the unit and all sensors are pooled within 2 seconds. Relay block contains relays with single pole double throw 3A, 250V AC and 30V DC contact configuration. Pulse unit is able to count digital pulses and read the current input logical states (optically decoupled, max input pulse frequency is 10kHz). Arrangement of the prototype is shown in Fig. 6.

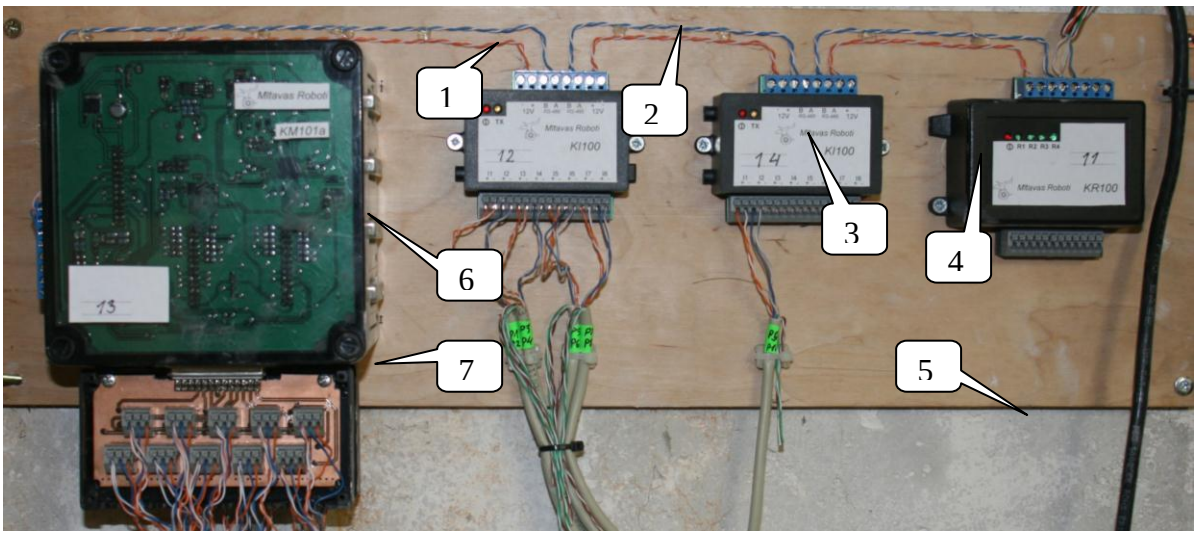


Fig. 6. **Management system prototype:**
1 - power supply (12V), 2 – RS485 interface, 3 - input block, 4 - relay block, 5 - RS485 to PC, 6 – temperature block, 7 - connection from TSIC506 sensors

Microcontroller-based device or PC can be used as the master. In the cases where there is an immediate need for reading data, such as a dynamic experiment and monitoring with the possibility to quickly change the control algorithm and its settings, PC can be chosen as a master device. Slave device is connected to the PC using USB <-> RS-485 or RS-232 <-> RS-485 interface. PC software has been developed which is capable of providing control algorithm execution, the retention of data in the database and viewing data in real time, and a control tool for user that ensures possibility to create dynamic algorithm. With this tool the user can create various logical conditions. Readings of the given sensors are used as inputs, relay contact states are outputs. Algorithm is composed of a list of the logical conditions and is carried out with user-defined cycle time (from 1 s to 1 h). At the same time the sensor data and the taken control actions are continuously stored in the database. As for example in Fig. 7 (1), condition is verified at which logical state is the input 0 and what is the temperature setpoint, as a result relay is set to ON position, see Fig. 7 (2).

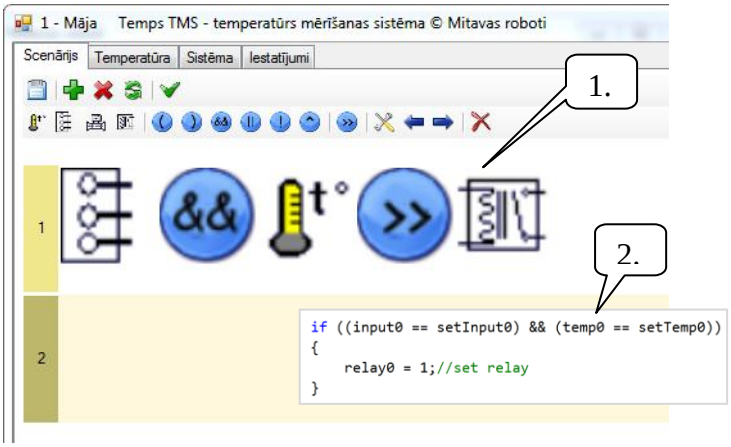
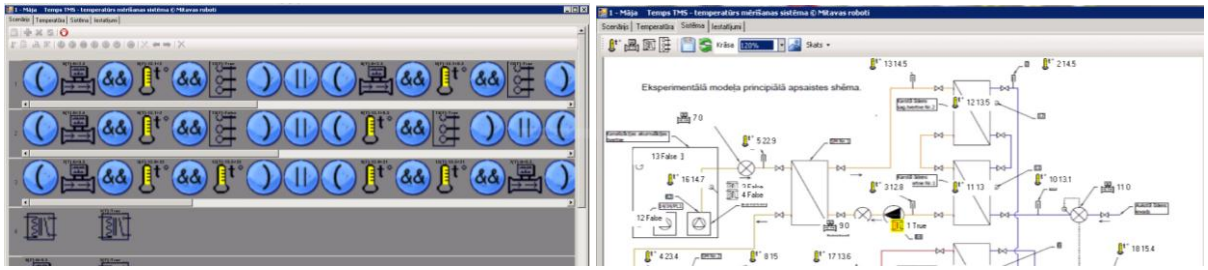


Fig. 7. **One condition:**
1 – Condition, 2 - interpretation of program code

By connecting PC to the Internet there is possibility to access the system remotely, using e.g. Remote Desktop to access the program online or read data from local data base server.
The other option as one can get sensor measurements from and send commands to slave devices is to use embedded control unit as a master, which is controlled by the MCU (Microcontroller Unit). MCU performs execution of control algorithm and communication with the rest of the equipment. SD memory cards are used for data storage; Ethernet interface provides access to the Internet. SD card also stores the logical conditions defined by the used (the control program), MCU reads them and executes cyclically. Temperature test run for 16 sensors is shown in Fig.8, 9. The control device runs an embedded WEB server and if it connected to the Internet then XML data file can be uploaded. This file consists of temperature, input, and output data.



1. 2.

Fig. 8. **PC master interface (form):**
1 - the conditions, 2 - System overview

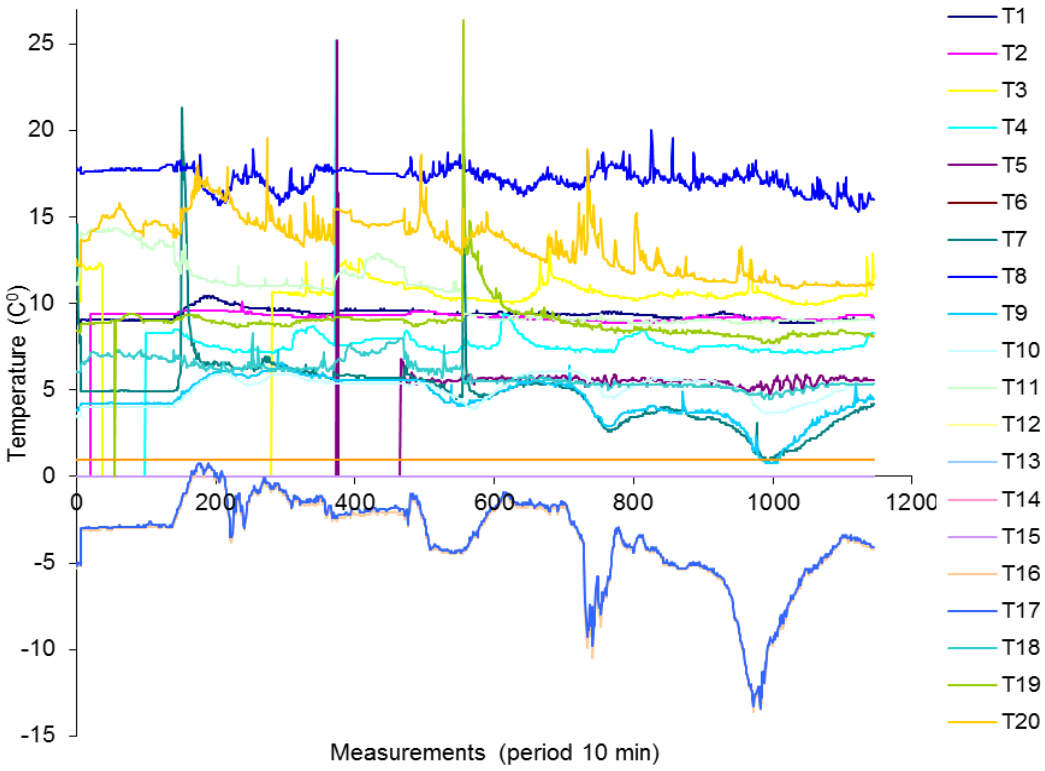


Fig. 9. **16 temperature sensor test run**

Conclusions

By designing of system, which allows for automatic control and monitoring, it can be concluded that this type of control system can be used in various scientific experiments. The control system has a relay output, temperature and pulse reading blocks, as well as the control unit, which in this case is a personal computer or embedded WEB server. These blocks can be combined in a variety of combinations, making it easy to perceive the system configuration. Control algorithm can be easily composed according to the Boolean algebra principles.

References

Bayindir R., Y. Cetinceviz Y., 2010. A water pumping control system with a programmable logic controller (PLC) and industrial wireless modules for industrial plants—An experimental setup, 8. pp

Bucur D., Kwiatkowska M., 2010. On verification software for sensor nodes, 15. pp.

Da'na S., Sagahyroon A., Elrayes A., Al-Ali A.R., Al-Aydi R., 2007. Development of a monitoring and control platform for PLC-based applications, 10. pp.

Galiņš A., Leščevics P., 2008. Programmējamie loāiskie kontrolleri: mācību līdzeklis. 135 pp. 26-31. (In Latvian)

Krugljak, K., 2002. Promishljenije seti: celji i sredctva (*Industrial networks: the aims and tools*) / / «STA» № 4, 2002.pp. 6 – 17.

Osadčuks V., Galiņš A., 2009. Review of industrial communication networks in the control of small-scale autonomous power supply systems. In: *Research for rural development*: Proceedings of Annual 15th International Scientific Conference. Latvia University of Agriculture, Jelgava 2009. pp. 332-337.