IDENTIFICATION AND EMBEDDED-SOFTWARE AUTOMATION FOR WATER DRAIN PROCESS

Monica Leba, Emil Pop

University of Petrosani, System Control,
Applied Informatics and Computer Engineering Department,
Str.Universitatii, no.20, Petrosani,
monicaleba@upet.ro, emilpop@upet.ro

Abstract: In this paper a water drain process is considered in order to design and implement an automatic control system. In the first part, the caption hydraulic process is identified and mathematical model is determined. Based on this model and on the process requirements, the simulation is done. The automatic control system is designed in two variants, first using a PLC and second by an embedded software controller. At the end of the paper are presented the experimental results.

Keywords: identification, embedded systems, controller, process automation.

1. INTRODUCTION

Hydraulic processes that contain caption, transport, treatment and evacuation of water are an important part of economical and social human activities. In these processes are used different machines, installations and electromechanical equipments and the computer systems for control and information processing. The most complex processes of the kind described above are: big cities water alimentation system, mining water evacuation, water drain, hydro-energetic systems water management etc. These complex processes are very different from each other, but the control and management of them have almost the same requirements. Further we'll take into account the hydraulic process of caption and evacuation of water from a plant having the block diagram from figure 1.

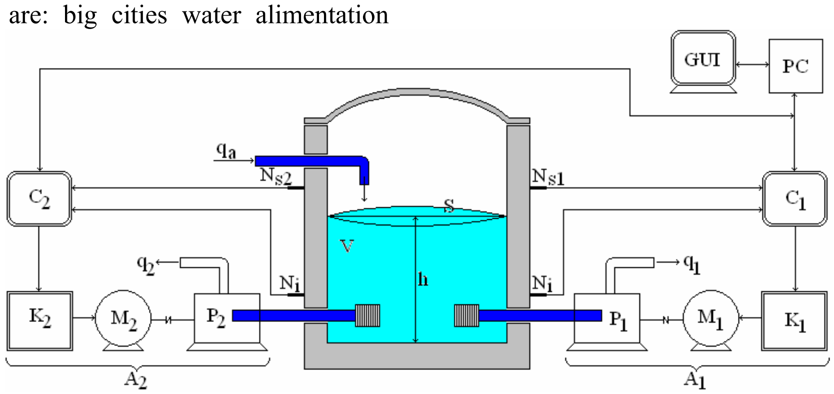

Fig.1. Process block diagram

The water accumulates with a variable q_a flow in a tank having the volume V, surface S and instant height h. The evacuation of the water can be done using any of the pumping aggregates, by q_1 and q_2

flows, formed up by pumps (P_1, P_2), motors (M_1, M_2), contactors (K_1, K_2) and controllers (C_1, C_2). The whole process is controlled and monitored with an industrial PC, having an appropriate graphical interface (GUI).

Each aggregate will be started when the water reaches a superior level (N_{s1}, N_{s2}) and stopped when the water gets below an inferior level (N_i).

There are imposed several process management conditions, as follows:

- Running can be automatic (A) and manual (M);
- Pumps must function cyclically for uniform wear and to maintain the rotor dry;
- If a pumping aggregate cannot evacuate the water by itself, then it will be started the second aggregate too;
- In order to eliminate the pump filling before starting, simplifying by this the control algorithm, the pumps are mounted under the inferior level of the tank, being filled during the water accumulation;
- Before starting an aggregate there must be emitted a preventive acoustic signal of 5-9 seconds for personnel protection purposes;
- If the two pumps cannot evacuate the water, then the alarm will be activated.

Besides the above conditions there must be monitored the emergency states of the aggregates (electrical protections, hydraulic protections, mechanical protections).

The motors have hardware electrical protections (short-circuit, overload, low voltage and grounding) by multi-port electronic relays, the control algorithm will monitor and display the protections state acting accordingly to the actual situation.

A pump flow loss represents a hydraulic emergency and the pump must be stopped.

Mechanical emergencies protection must be supervised and treated according to the client requirements.

2. IDENTIFICATION, MODELING AND SIMULATION

The above presentation can be concluded by the following three classes of conditions: management conditions, protection conditions and running conditions.

Management conditions:

- A dynamic graphic user interface (GUI) presents the process running;
- The user choose the way: automat or manual;
- Aggregates cyclic running and the number of consecutive starting for each aggregate is established by the user;
- In normal conditions the water evacuation will be done in a period of the day when the power system load is minimal.

Protection conditions:

- Before the aggregates starting, it will be signaled by a preventive acoustic signal;
- When the water gets over the superior levels, it will be signaled by an emergency signal;
- The electrical, hydraulic and mechanical protections are monitored and treaded according to the requirements.

Running conditions:

- When the water reaches the N_{si} superior level the aggregate A_i (i=1,2) will start;
- When the water gets under the inferior level N_i all the aggregates will stop;
- Will be counted each aggregate A_i running times;
- When an aggregate is started it is used a watchdog for the pump flow monitoring;
- Each running aggregate will be helped by the other one to evacuate the water in case of big accumulation flow.

Based on the elements from fig.1 block diagram and assuming the pumps nominal flows q_{n1}, q_{n2} and KC_1, KC_2 being the K_1, K_2 contactors control functions, we can determine the tank water volume variation.

$$\frac{dV}{dt} = q_a(t) - KC_1 \cdot q_{1n} - KC_2 \cdot q_{2n}$$

From this results the tank level h variation law:

$$h = \frac{1}{S} \cdot \int_0^t [q_a(t) - KC_1 \cdot q_{1n} - KC_2 \cdot q_{2n}] \cdot dt \qquad (1)$$

Because equation (1) is a relation with continuous time variables and the next ones are logic variables with {0,1} values, for compatibility reasons, we'll represent the logic operation AND by "$\wedge$" symbol, OR by "$\vee$" symbol and NOT by "–" symbol, as follows:

$$x \wedge y = x \cdot y$$

$$x \vee y = x + y - x \cdot y$$

$$\bar{x} = 1 - x$$

The commutation values of electrical protection PE_i, grounding PT_i and hydraulic PH_i relays are of logic type having a serial action through logic operation AND.

These values are combined with the automat/manual control option (AM_i) that selects the manual commands stop/start with memorizing (O_i, P_i, KC_i) or automatic command from the controller (C_i) of KC_i contactor (i=1,2).

$$KC_i = PE_i \cdot PT_i \cdot PH_i \cdot [AM_i \cdot C_i + (1-AM_i) \cdot (1-O_i) \cdot (P_i + KC_i - P_i \cdot KC_i)] \quad (2)$$

The cyclic functioning equations of the pumps allow the real levels N_{s1}, N_{s2} connection directly or crossed to N_{s1}^P, N_{s2}^P according to the signal $C \in \{0,1\}$ given by a modulo n counter whose value is done by the human operator (e.g. n=8).

$$N_{s1}^P = N_{s1} \cdot (1-C) + N_{s2} \cdot C \quad (3)$$

$$N_{s2}^P = N_{s2} \cdot (1-C) + N_{s1} \cdot C \quad (4)$$

The automatic controllers equations C_i (i=1,2) depend on N_{s1}^P, N_{s2}^P, N_i levels and on floating level h. When $h > N_{s1}^P$ it will be started the A_i aggregate and if h is over N_{s2}^P than it will be started A_2 too. The aggregates will function until $h < N_i$. The C_i commands are transformed from analogical to logical and memorized by RS flip-flops.

$$C_i = [1-(h < N_{s1}^P)] \cdot [(h < N_{s1}^P) + C_i - (h > N_{s1}^P) \cdot C_i] \quad (5)$$

Equations (1), (2), (3), (4), (5) allow the achievement of process simulation using the MatLab-Simulink platform (fig.2.a).

Simulation results are presented in fig.2.b and fig.2.c for real situations if only one pump is running and for both pumps running respectively. A/M commands and protections are simulated by switches.

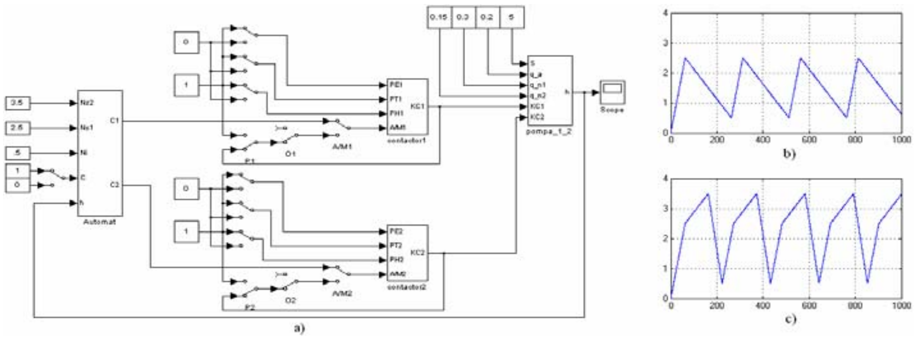

Fig.2. Process simulation: a) MatLab-Simulink diagram; b) One pump running; c) Two pumps running

3. SOFTWARE CONTROL IMPLEMENTATION

Based on this model and on simulation results there was achieved the logic diagram that contain the programming principle (fig.3).

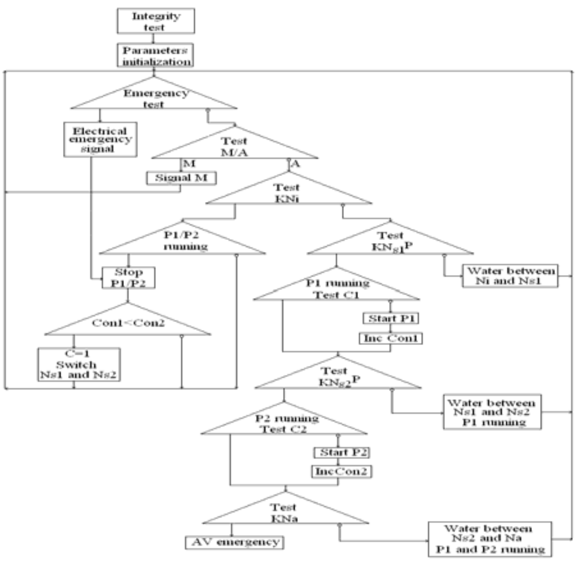

Fig.3. Algorithm diagram

The algorithm from fig.3 can be implemented in two ways: using a PLC (minimal variant) and with embedded-software controller (maximal variant).

Minimal variant

This variant uses a Klockner-Moeller PLC, programmed by ladder diagrams method. The PLC has 8 inputs (I_1 … I_8) and 4 outputs (Q_1 … Q_4). There were used 5 inputs and 2 outputs. The mathematical model is presented bellow:

$I_1=N_{s1}=M_1$ the first superior water level

$I_2=N_{s2}=M_2$ the second superior water level

$I_3=N_i=CC_1=CC_2=M_3$ the inferior water level

$I_4=M_4$ the first aggregate protection switch

$I_5=M_5$ the second aggregate protection switch

$RC_1=C_2$ resets the first counter C_1

$RC_2=/C_1$ resets the second counter C_2

$C_1=M_8$ pumps switching cycle

$M_1 \cdot /M_8 + M_2 \cdot M_8 = M_9$ superior levels commutation for the first aggregate

$M_2 \cdot /M_8 + M_1 \cdot M_8 = M_{10}$ superior levels commutation for the second aggregate

$M_3 \cdot (M_9+M_6) \cdot /M_4 = Q_1 = M_6$ starts the first aggregate

$M_3 \cdot (M_{10}+M_7) \cdot /M_5 = Q_2 = M_7$ starts the second aggregate

In fig.4 is presented the ladder diagram for the minimal variant.

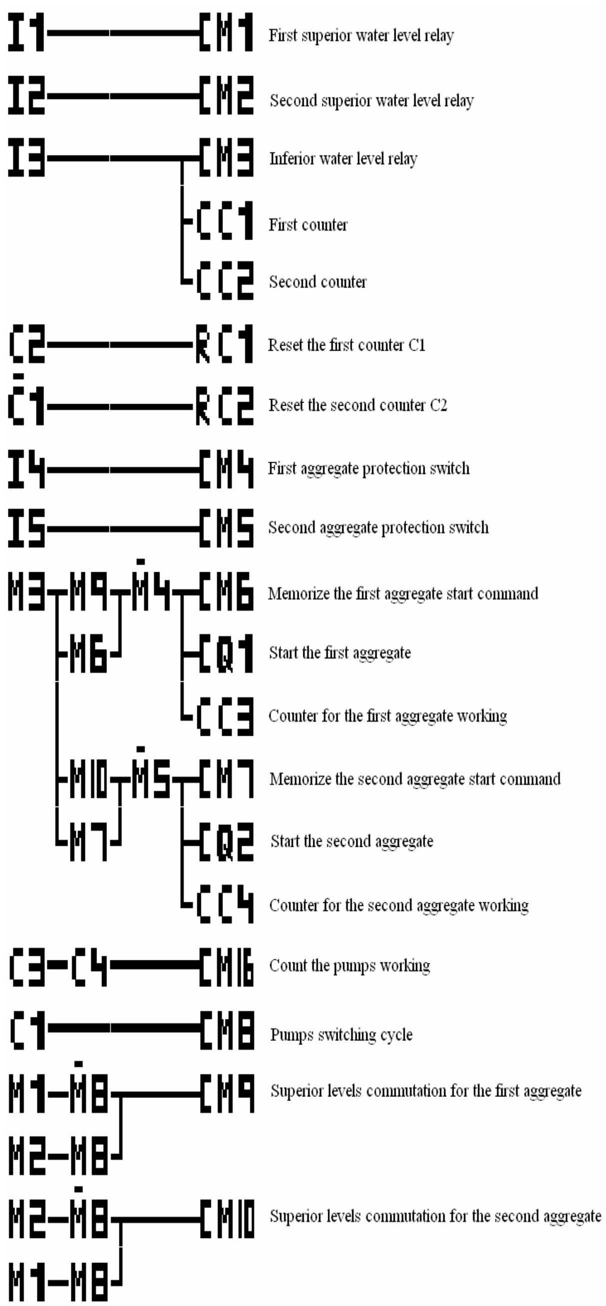

Fig.4. Ladder diagram

Maximal variant

This variant uses an industrial PC and an acquisition card. The embedded software is written in assembly language, for real-time working reasons.

In the first part, the control software tests the system integrity. If defects or abnormal situations are detected, the functioning will be interrupted, waiting for their remediation. If everything is OK, follows the parameters initialization, like: inferior level N_i, superior levels N_{s1} and N_{s2}, emergency, accumulation flow q_a, evacuation flows q_1 and q_2 etc.

In the running state the emergency transducers are tested, and if an emergency is detected the entire system is stopped.

There are implemented two running ways: manual and automat. In manual running, the software achieves only monitoring; the control is done by the human operator. In automat running, the level transducers are read. If the water is below the inferior level N_i the pumps are stopped, waiting for the water accumulation. If the water is over the superior level N_{s1} the P_1 pump will be started. If it cannot evacuate the water by itself and the water is over the superior level N_{s2} than the P_2 pump will be started too.

In the above logic, the P_1 pump is the main pump, and the P_2 pump is the auxiliary one. In order to prevent the P_1 pump over wear, because P_1 will function more than P_2, there was introduced a pumps cycling algorithm P_1-P_2-P_1-… .

The program, written in I80X86 assembly language, implements the algorithm described above. There was designed a friendly dynamical graphic user interface, that presents the animated functioning and monitoring of the entire process.

Below is presented the main loop subroutine, and in fig.5 the application main screen.

```
MainLoop proc near
         mov ax,nivelInf
agbcl4: mov LEDcul,40 ;40=red, 48=green
         call LedTradNi
         mov t,0  ; system initialization
         mov P1,0
         mov P2,0
         mov LEDcul,27 ;40=red,
                  48=green, 27=no color
         call LedTradP1
         mov LEDcul,27 ;40=red,
                  48=green, 27=no color
         call LedTradP2
         call OpresteP1
         call OpresteP2
         mov LEDcul,48 ;40=red, 48=green
         call LedTradNi
         mov nivel,ax
agbcl: call tasta
         jnc contML1
         jmp exitML
contML1: inc t
         call calcNiv
         mov ax,nivel
         add ax,nivelCalc
         mov x1,500
         sub x1,ax
         mov x2,500
         sub x2,ax
         mov y1,321
         mov y2,479
         mov culin,51
         call nilinie
         call RedoSenzorNi
         call RedoSenzorNs1
         cmp ax,NivelSup1
```

```
          jne agbcl
          mov LEDcul,48 ;40=red, 48=green
          call LedTradNs1 ; Ns1 was reached
          mov P1,1            ; P1 starts
          call PornesteP1
          mov LEDcul,48 ;40=red,
                    48=green, 27=no color
          call LedTradP1
          mov nivel,ax        ; system re-initialization
          mov t,0
agbcl1: call tasta
          jnc contML2
          jmp exitML
contML2: inc t
          call calcNiv
          mov ax,nivel
          add ax,nivelCalc
          inc ax
          cmp ax,nivel
          jne creste0
          mov LEDcul,48
          call LEDtradNs1
creste0: dec ax
          mov x1,500
          sub x1,ax
          mov x2,500
          sub x2,ax
          mov y1,321
          mov y2,479
          cmp ax,nivel
          jae creste
          mov culin,27
          dec x1
          dec x2
          mov y2,450
          call nilinie
          mov y1,470
          mov y2,479
          call nilinie
          mov y1,321
          jmp des2
creste: mov culin,51
          call nilinie
des2:     call RedoSenzorNi
          call RedoSenzorNs1
          call RedoSenzorNs2
          cmp ax,nivelSup2
          jae agbcl2
          cmp ax,nivelInf
          ja agbcl11
          jmp agbcl4
agbcl11: jmp agbcl1
agbcl2: mov LEDcul,48 ;40=red, 48=green
          call LedTradNs2 ; Ns2 was reached
          mov P2,1            ; P2 starts
          call PornesteP2
          mov LEDcul,48
          call LedTradP2
          mov LEDcul,40
          call LedTradNs2 ; Ns2 was reached
          mov nivel,ax        ; system re-initialization
          mov t,0
agbcl3: call tasta
          jnc contML3
          jmp exitML
contML3: inc t
          call calcNiv
          mov ax,nivel
          add ax,nivelCalc
          inc ax
          cmp ax,nivelSup1
          jne scade0
          mov LEDcul,40
          call LEDtradNs1
scade0: dec ax
          mov x1,500
          sub x1,ax
          mov x2,500
          sub x2,ax
          mov y1,321
          mov y2,479
          mov culin,27
          mov y2,450
          call nilinie
          mov y1,470
          mov y2,479
          call nilinie
          call RedoSenzorNi
          call RedoSenzorNs1
          call RedoSenzorNs2
          cmp ax,nivelInf
          ja agbcl3
          jmp agbcl4
exitML: ret
MainLoop endp
```

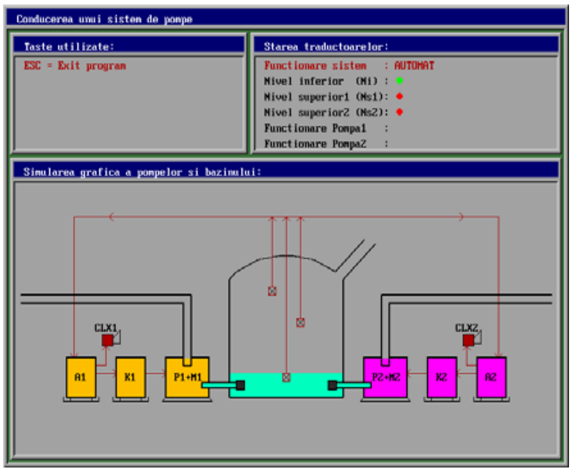

Fig.5. Application main screen

4. EXPERIMENTAL RESULTS

In fig.6 are presented several program running examples for normal accumulation flow when only one pump is running, for big accumulation flow when both pumps are running and an emergency case.

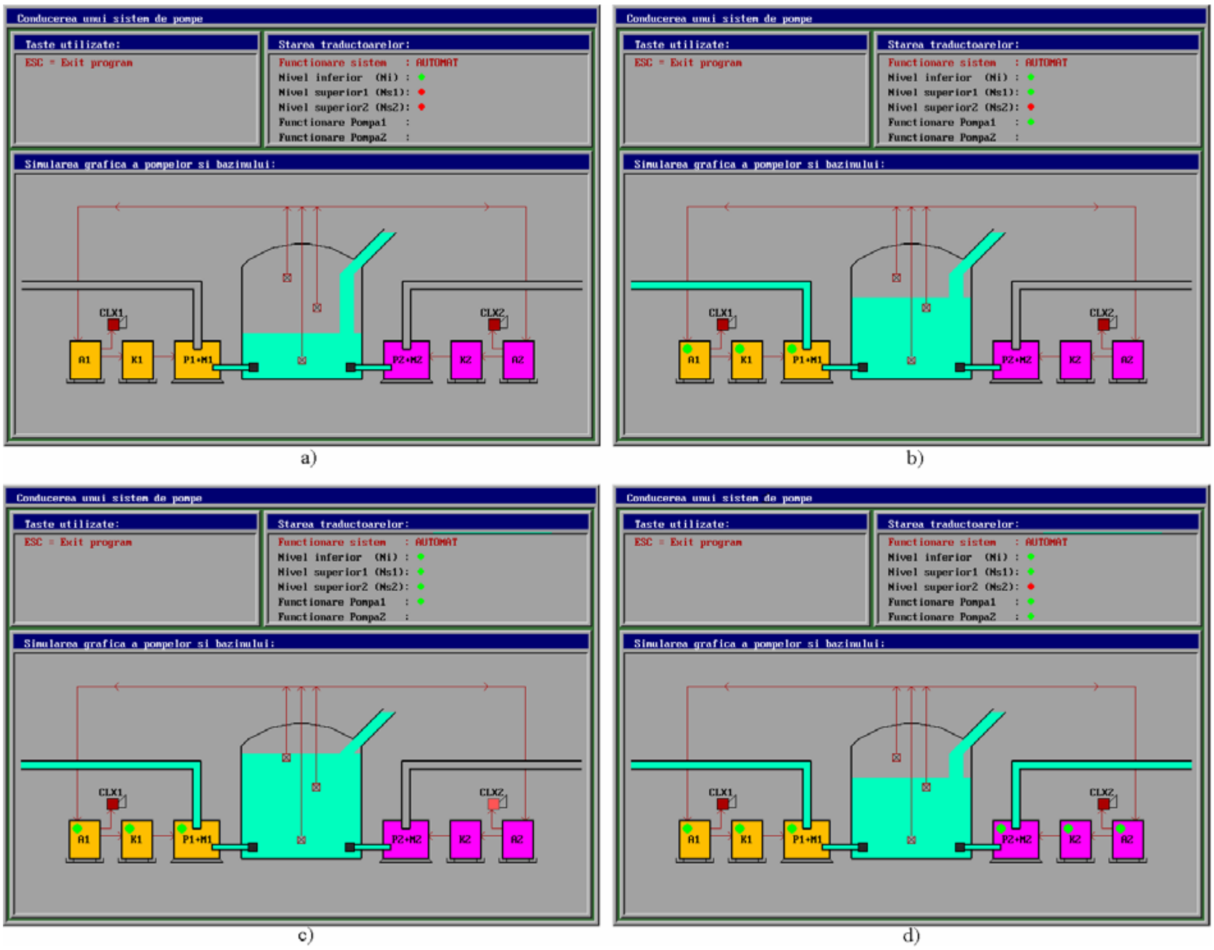

Fig.6. System running: a) Water caption; b) One pump running; c) Second pump start; d) Both pumps running

5. CONCLUSIONS

- Control algorithm implementation can be done in two ways, depending on the process complexity. In case of simple processes, the PLC can be used. In case of complex processes the embedded software controller must be used.
- These solutions reduce at least 30% of the present electro-mechanical equipments.
- Automatic control leads to minimizing the power load peak.

REFERENCES

Pop, E. (1983). *Automatizări în industria minieră*. Editura Didactică şi Pedagogică, Bucureşti.

Pop, E. and Leba, M. (2003). *Microcontrollere si automate programabile*. Editura Didactică şi Pedagogică, Bucureşti.

Pop, E. and Pop, M. (1999). *Programare în limbaj de asamblare I80X86*. Editura Didactică şi Pedagogică, Bucureşti.

*** (2003). *MatLab Simulink Users Guide*. MathWorks.