



## Automat programabil Mini PLC4404

### Ghid de programare in Tiger BASIC

**Nota:** Acest ghid deservește și [noua generație](#) de automate programabile Mini PLC4404 dotate cu porturi USB, conexiune WiFi și configurare dinamică DHCP.

Acest tutorial se folosește împreună cu manualele de utilizare ale automatelor programabile Mini PLC4404, configurare Ethernet și GPRS:

#### [Mini PLC4404 cu porturi USB](#)



#### [Mini PLC4404 cu porturi RS232](#)



Materialul prezintă programe Tiger BASIC de inițiere, realizate pentru Mini PLC4404:

- [Mini\\_plc\\_01.tig](#) → Comanda ieșirilor digitale
- [Mini\\_plc\\_02.tig](#) → Acționarea intrărilor digitale și transmiterea mesajelor de alarmă
- [Mini\\_plc\\_03.tig](#) → Măsurarea pe 12 biți a temperaturii ambiante
- [Mini\\_plc\\_04.tig](#) → Programarea limitelor pentru marimile analogice măsurate
- [Mini\\_plc\\_05.tig](#) → Achiziția în SRAM a valorilor analogice și prezentarea graficelor
- [Mini\\_plc\\_06.tig](#) → Achiziția în SRAM a valorilor analogice, digitale și prezentarea graficelor
- [Mini\\_plc\\_07.tig](#) → Controlul în buclă închisă al temperaturii, prezentarea graficelor
- [Air\\_Quality.tig](#) → Utilizarea senzorilor AQ-1 pentru Calitatea aerului, Modbus RTU
- [Mini\\_plc\\_08.tig](#) → Folosirea modulelor de extensie pentru intrări și ieșiri digitale

Programele elaborate multitasking sunt accesibile utilizatorilor începători sau avansați, dornici să obțină rezultate practice în timp foarte scurt.

Programele au puține linii de cod utilizator și sunt intens comentate. Mare parte din activități sunt realizate de taskurile și subrutinele sistem.

Exemplele prezentate pornesc de la rezolvarea sarcinilor simple. Sunt adăugate treptat noi sarcini soluționate cu un număr redus de taskuri și subrutine utilizator.

Utilizatorii experimenteaza programele de mai sus si dobandesc astfel experienta necesara elaborarii propriilor programe.

Arhiva [Programe](#) contine fisiere in cod sursa Tiger BASIC: fisiere program avand extensie [.tig](#) si fisiere sistem (librarii) cu extensie [.inc](#).

Accesul la codul sursa se face cu mediul de dezvoltare [Tiger BASIC IDE](#) pus la dispozitie de Wilke-Technology GmbH. Urmati [manualul de utilizare](#), capitol 2.

Materialul prezinta:

- Implementarea sarcinilor cu ajutorul taskurilor si subrutinelor
- Experimentarea aplicatiilor ~ executia comenzilor la distanta si indeplinirea sarcinilor

Tiger BASIC IDE este usor de folosit. Structura multitasking asigura o constructie organizata a programelor.

## Primele indrumari

1. Secvente de linii cod pot fi copiate si mutate cu atentie in alt program.

Se verifica [Start\Compile](#) urmat de [Start\Run](#).

Dezactivarea secventelor de linii cod se face:

```
#CM          'comment  
'secventa linii cod  
#endCM      'end comment
```

2. La experimentarea si dezvoltarea aplicatiilor sunt necesare informatii de verificare. Folositi COM portul Ser1 - RS232 la rata de baud 38400, pe un [Terminal Windows](#), ex:

```
PRINT #SER, #1, CRLF; "Variabila X = "; X; " Variabila Y = "; Y      'variabilele pot fi de orice natura
```

3. Aveti nevoie de 3 adaptoare USB-RS232 pentru ca lucrati serial - 38400 baud cu 3 aplicatii:

- Mediul de dezvoltare Tiger BASIC
- Dispatcher server PC
- Terminal Windows

4. Aplicatiile sunt configurabile prin selectarea definitiilor si modificarea valorilor atribuite definitiilor. Este util conditionarea unei secvente de cod de existenta sau nu a definitiei respective.

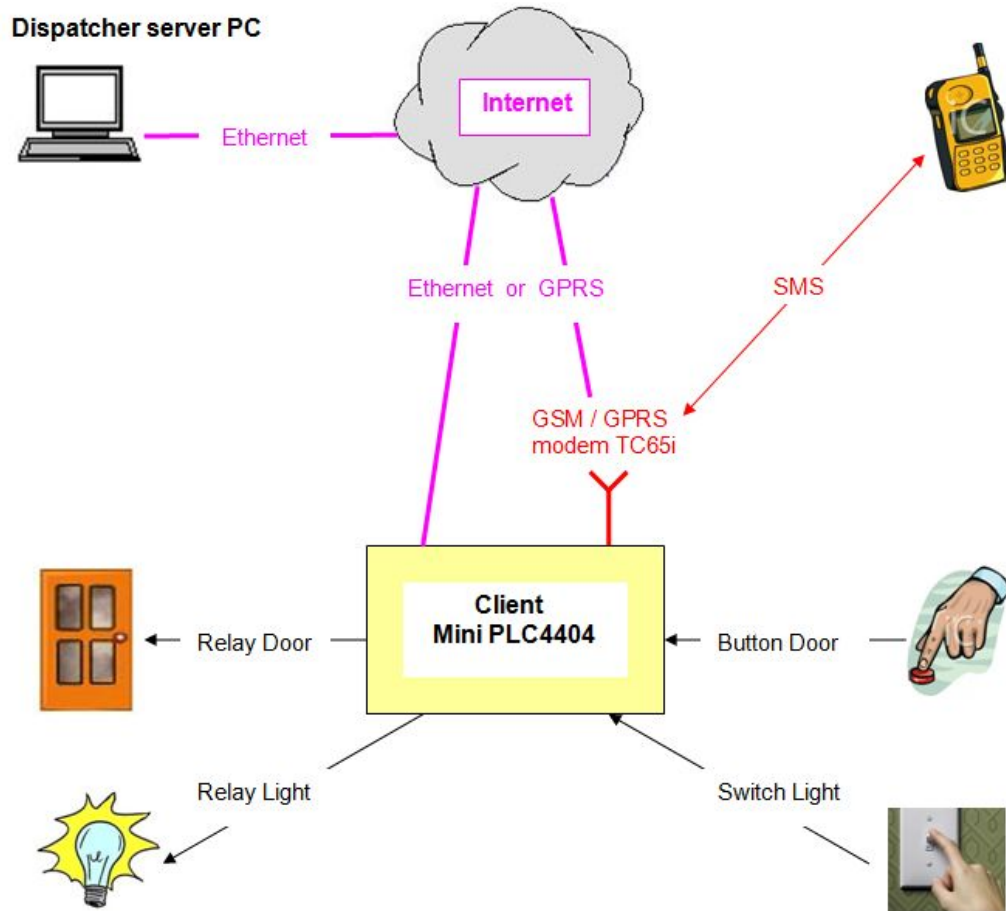
```
#ifdef FIRST_MES_USE_LIMITS  
'furnizeaza starea activa sau inactiva a iesirii digitale Relay_Heating  
CALL GetDigOutputStatus(Host, Relay_Heating, MessageToSend$)  
#endif
```

5. La experimentarea aplicatiilor e util dezactivarea parolei pentru economie de timp.

6. Programele si fisierele sistem sunt frecvent optimizate, extinse, sunt adaugate noi aplicatii in cod sursa. **Descarcati cele mai recente fisiere.**

In material sunt inserate exemple cod generale sprijinite de exemple cod particulare. Delimitarile intre general si particular se fac: [In prezenta aplicatie](#), [in acest exemplu](#), etc.

## 1. Aplicatie → Comanda iesirilor digitale



Programul [Mini\\_plc\\_01.tig](#) prezinta comanda temporizata si netemporizata a iesirilor digitale. Comenzile provin:

- La distanta, prin Internet / ETHERNET sau GPRS, respectiv SMS
- Local, de la intrarile digitale ale automatului programabil

Deschideti aplicatia [Mini\\_plc\\_01.tig](#) in mediul de dezvoltare Tiger BASIC. Acest prim exemplu pune in evidenta structura comuna a programelor realizate pentru automatul programabil.

Programul Tiger BASIC are trei sectiuni:

- **Sectiunea 1** DECLARA SARCINILE APLICATIEI
- **Sectiunea 2** LIBRARIILE SISTEM
- **Sectiunea 3** EXECUTA SARCINILE APLICATIEI

### 1.1. Sectiunea 1 - DECLARA SARCINILE APLICATIEI

Primele linii cod declara numele aplicatie si al automatului programabil:

```
#define APPLICATION_NAME      "Mini plc 01"  
  
#define MINI_PLC4404         "Mini PLC4404"
```

'----- Selecteaza standardul ETHERNET atunci cand este folosit Ethernet Adapter EM01 -----

```
#define ETHERNET    "Ethernet"    'conexiune Internet / Ethernet
```

'----- Selecteaza standardul GSM atunci cand este folosit modemul GSM/GPRS TC65i -----

```
#define GSM          "GSM"
#ifdef GSM
    'insemnand cel putin facilitatea SMS
```

'----- Selecteaza standardul GPRS -----

```
    #define GPRS      "GPRS"      'Internet / GPRS este conditionat de semnalul GSM
#endif 'GSM
```

ETHERNET si GPRS folosesc aceleasi resurse si nu pot fi selectate in acelasi timp.  
GPRS asigura implicit comunicatia SMS.

In continuare este declarata singura comanda a aplicatiei:

'declara comanda in forma text

```
#define OPEN_DOOR          "open door"
```

'declara comanda in forma numerica

```
#define OPEN_DOOR_nr      2          'poate fi intre 1...250
```

**Declaratiile** sunt de regula scrise cu majuscule si sunt precedate de **#define**. Declaratiile pot fi puse in corespondenta cu un text care este utilizat in diverse scopuri:

```
- #define OPEN_DOOR          " open door "
```

Declaratiile pot fi puse in corespondenta cu o valoare numerica, desemnand astfel constante:

```
- #define OPEN_DOOR_nr      2
```

**Comenzile** sunt declarate atat in forma text cat si in forma numerica.

'declara variabila de iesire digitala personalizata

```
WORD RelayOpenDoor      'conectata la Relay_1_Dig_Out, iesire dig. generalizata
```

'declara variabila auxiliara necesara temporizarii releului

```
LONG TimerOpenDoor
```

'declara limita valorii de temporizare

```
#define TIMER_OPEN_DOOR_LIMIT      5000    'in milisecunde
```

'declara variabila de intrarea digitala personalizata, folosita pentru butonul de deschidere a usii

```
WORD ButtonOpenDoor      'conectat la First_Dig_Input, intrare dig .generalizata
```

### 1.1.1. Declararea variabilelor

Tiger BASIC opereaza cu variabile **BYTE** (8 biti), **WORD** (16 biti), **LONG** (32 biti inclusiv semn), **REAL** (64 biti, virgula flotanta, semn), **STRING** si **ARRAY**.

Utilizatorul declara doua tipuri de variabile:

- **Variabilele globale** se declara inainte de **TASK MAIN** si pot fi modificate de orice task sau subrutina care le utilizeaza.
- **Variabilele locale** se declara in interiorul taskurilor sau subrutinelor si sunt utilizate doar de taskurile si subrutinele in cauza.

In exemplul de fata, variabilele [RelayOpenDoor](#) si [TimerOpenDoor](#) sunt globale.

Recomandam ca declararea constantelor si variabilelor sa fie sugestiva.  
Declaratii de genul: a, Z, x, trebuie evitate pentru inlaturarea confuziilor.

## 1.2. Sectiunea 2 - LIBRARII → TASKURI SI SUBRUTINE SISTEM

Procedurile sistem cuprinse in fisierele [.inc](#) indeplinesc 95% din activitatile oricarei aplicatii:

- Comunicatii RS232, RS485, Internet / ETHERNET sau GPRS, Email, SMS
- Sincronizarea cu serverele NTP (ceasurile atomice ale Internetului)
- Receptionarea si executia comenzilor
- Salvarea evenimentelor de comanda, transmiterea mesajelor de alarma
- Masurarea marimilor analogice, achizitii date
- Alte proceduri si protocoale ex Modbus RTU, M2M, GPS etc

**Se recomanda ca modificarea fisierelor sistem sa fie efectuata de utilizatori experimentati.**

## 1.3. Sectiunea 3 - EXECUTA SARCINILE APLICATIEI

[Sectiunea 3](#) executa sarcinile declarate in [sectiunea 1](#) cu ajutorul taskurilor si subrutinelor utilizator.

[TASK MAIN](#) este primul task utilizator si ruleaza subrutina sistem [Main\\_Task\\_Settings\(\)](#) indeplinind in principal configurarea porturilor seriale de comunicatie Ser0, Ser1 si lansarea in executie a taskurilor sistem.

```
'=====
TASK MAIN          'beginning of TASK MAIN
'-----
  CALL Main_Task_Settings ()
'-----
END                'end of TASK MAIN
'=====
```

[TASK Execute\\_Customer\\_Commands](#) ruleaza subrutina [Customer\\_Dig\\_In\\_Out\\_Exe\(\)](#) in bucla infinita.

```
'=====
TASK Execute_Customer_Commands
'-----
  WHILE 1 = 1
    CALL Customer_Dig_In_Out_Exe ()
    WAIT_DURATION 250          'milisecunde
  ENDWHILE
'-----
END          'end of TASK Execute_Customer_Commands
'=====
```

Subrutina [Customer\\_Dig\\_In\\_Out\\_Exe \(\)](#) indeplineste mai multe sarcini prezentate in exemplele urmatoare.

### 1.3.1. Subsectiunea 3 - SUBRUTINE AUXILIARE UTILIZATOR

Subsectiunea 3.1. Subrutinele auxiliare utilizator sunt folosite in toate programele:

- DeliverCustomerCommands (...)
- ConvertCustomerCommands (...)
- Assign\_Name\_To\_Dig\_I\_O (...)
- ExecuteCustomerCommands (...)
- GetCustomerDig\_IO\_Status (...)

Nota: Subrutinele auxiliare utilizator sunt apelate de taskuri si subrutine sistem.  
In unele programe subrutinele auxiliare au continut nul.

Subrutina DeliverCustomerCommands (...) livreaza lista de comenzi in format text ale aplicatiei. In exemplul de fata exista comanda: OPEN\_DOOR

Subrutina ConvertCustomerCommands (...) cuprinde codul pentru realizarea conversiei comenzilor din format text in format numeric si viceversa. Capul de tabel este:

```
SUB ConvertCustomerCommands (BYTE CmdConvSense; VAR STRING commandName$;  
                             VAR WORD commandNumber)
```

Campurile precedate de notatia VAR desemneaza variabile care pot fi modificate in interiorul subrutinei, devenind astfel variabile de iesire.

Subrutina Assign\_Name\_ToDigInput (...) ataseaza intrarii digitale o denumire care poate fi un text informativ sau numele comenzii ce urmeaza executata atunci cand intrarea digitala comuta pe nivelul logic activ.

Subrutina Assign\_Name\_ToDigOut (...) ataseaza iesirii digitale o denumire care poate fi un text informativ sau numele comenzii care comuta nivelul logic al iesirii digitale.

## 1.4. Executia comenzilor, prezentare generala

Comenzile provin din doua surse:

- A. Tranzitii logice ale intrarilor digitale ale automatului programabil
- B. Explicit in format text, pe caile de comunicatie:
  - o KEY ~ keyboard-ul statiei Dispatcher server PC prin Internet sau serial RS232
  - o SMS ~ telefonul mobil al utilizatorului

### Cazul A ~ comenzi provenite de la intrarile digitale

Intrarile digitale au impedanta de intrare 30 kΩ, sunt protejate la supratensiuni accidentale ±32Vcc si sunt marcate vizibil pe placa de circuit imprimat.

Starea normala in vant este HIGH\_LEVEL putand fi trase la masa LOW\_LEVEL. Intrarile digitale sunt conectate la contacte NC sau NO ale unor butoane, comutatoare, rele, senzori PIR etc.



Starea logica activa a unei intrari digitale este declarata in functie de particularitatile dispozitivului de intrare, putand fi [LOW\\_LEVEL](#) sau [HIGH\\_LEVEL](#).

Un buton este activ in stare apasat [LOW\\_LEVEL](#) pe cand in general senzorii de alarma sunt activi in stare [HIGH\\_LEVEL](#). In acest fel sunt puse in evidenta cazuri suplimentare de alarma privind distrugerea carcasei si intreruperea conductoarelor electrice ale sensorului.

Celor trei intrari digitale le sunt atasate variabile generalizate conform manual utilizare.

- [First\\_Dig\\_Input](#)
- [Second\\_Dig\\_Input](#)
- [Third\\_Dig\\_Input](#)

Intrarile digitale sunt configurate si investigate in subrutina [Customer\\_Dig\\_In\\_Out\\_Exe \(\)](#) care indeplineste mai multe sarcini prezentate in continuare.

### **Scanarea intrarilor digitale adresate generalizat**

Subrutina [Read\\_Host\\_Dig\\_Inputs \(\)](#) identifica starile logice ale intrarilor digitale generalizate.

### **Utilizarea intrarilor digitale adresate personalizat**

Elaborarea si interpretarea programelor sunt simplificate daca este utilizata adresarea personalizata, sugestiva a intrarilor, iesirilor digitale. Variabilele digitale personalizate iau valorile determinate ale variabilelor digitale generalizate, conform [exemplului de fata](#):

[ButtonOpenDoor = First\\_Dig\\_Input](#)

### **Configurarea nivelului logic activ**

Configurarea intrarilor digitale adresate generalizat sau personalizat se face in subrutina:

[SUB Validate\\_Dig\\_In \(WORD DigInput; BYTE ActiveLevel; BYTE OneOrTwoActiveLevels; BYTE EventAlarm; VAR LONG TimerInhib; LONG LimitInhib; BYTE CmdNumber\)](#)

Campul [ActiveLevel](#) se configureaza [LOW\\_LEVEL](#) sau [HIGH\\_LEVEL](#). Tranzitia pe nivelul activ lanseaza diverse proceduri executabile: actioneaza o iesire digitala, transmite o alarma etc.

Intrarile digitale pot avea un singur nivel logic activ sau ambele nivele active.

Campul [OneOrTwoActiveLevels](#) poate fi configurat [ONE\\_LEVEL](#) (ex. cazul unui buton apasat) sau [TWO\\_LEVELS](#) (ex. cazul unui comutator).

### **Configurarea campului [EventAlarm](#)**

Intrarile digitale pot fi conectate la:

- Butoane actionate manual de utilizator, lansand astfel in executie comenzi locale. In acest caz campul [EventAlarm](#) se configureaza [EVENT](#) pentru a salva evenimentul de comanda.
- Senzori de alarma. Tranzitia pe nivelul logic activ produce salvarea evenimentului de alarma si transmiterea alarmei prin Email, SMS, notificare in panoul Dispatcher server PC. In acest caz campul [EventAlarm](#) se configureaza [ALARM](#).
- Alte dispozitive de intrare. Campul [EventAlarm](#) poate fi configurat [NO\\_EVENT](#).

## Configurarea duratei de inhibitie intrarilor digitale

Tranzitia pe nivelul activ al intrarii digitale poate fi validata de fiecare data sau poate fi inhibata o durata limitata de timp:

- Fara durata de inhibitie, `TimerInhib = NO_TIMER` si `LimitInhib = NO_TIMER_LIMIT`
- Cand exista durata de inhibitie, in **Sectiunea 1** se declara o variabila LONG pentru campul `TimerInhib` si o constanta pentru campul `LimitInhib`. Pe de o parte, inhibarea elimina oscilatiile parazite ale intrarilor digitale. Pe de alta parte, atunci cand intrarile digitale sunt conectate la senzori de alarma, aceste campuri stabilesc intervalul de timp la care pot fi transmise alarme succesive identice. Transmiterea alarmelor identice la intervale scurte de timp este nepractica pentru utilizator. Valorile introduse in campul `LimitInhib` sunt de ordinul 10...15 minute sau mai mult, vezi programul `mini_plc4404_02.tig`.

## Asocierea unei comenzi intrarii digitale

O intrare digitala este supusa tranzitiilor logice venite din exterior. O intrare digitala nu poate receptiona o comanda in format text sau numeric. Atunci cand sarcina tranzitiei in stare activa este executia unei comenzi declarate in **sectiunea 1** este necesar **asocierea** intre intrarea digitala si comanda ce urmeaza executata. Sunt doua metode de asociere.

### Asociere metoda 1

In subrutina `Assign_Name_ToDigInput (...)`, in campul `AssignName$` se atribuie intrarii digitale forma text a comenzii ce urmeaza executata.

```
SUB Assign_Name_ToDigInput (WORD DigIn; STRING AssignName$;  
                           WORD Dig_I_O; VAR STRING DigIn_Name$)
```

In acelasi timp, campul `CmdNumber` al subrutinei `Validate_Dig_In (...)` este configurat `IS_COMMAND`, subliniind astfel ca este cunoscuta comanda ce urmeaza executata.

### Asociere metoda 2

In subrutina `Assign_Name_ToDigInput (...)`, in campul `AssignName` se atribuie intrarii digitale un text descriptiv, informativ.

In acelasi timp, campul `CmdNumber` al subrutinei `Validate_Dig_In (...)` este configurat cu valoarea numerica a comenzii ce urmeaza executata. In acest caz asocierea comenzii se face direct.

## Validarea comenzii asociate si salvarea evenimentului de comanda atunci cand intrarea digitala comuta pe nivelul activ

Subrutina `Validate_Dig_In (...)` valideaza comanda asociata atribuind variabilei globale `Exe_Command_Nr` valoarea numerica a comenzii asociate.

Evenimentul de comanda este salvat in memoria FLASH a automatului programabil.

## Comanda iesirilor digitale

### Iesiri digitale generalizate si iesiri digitale personalizate

In mod similar intrarilor digitale, celor doua iesiri digitale le sunt atasate variabile generalizate:

- Relay\_1\_Dig\_Out
- Relay\_2\_Dig\_Out

Iesirile digitale personalizate declarate in **sectiunea 1** iau valoarea iesirilor digitale generalizate identificate de taskurile sistem:

Comanda iesirilor digitale este de regula indeplinita in subrutina `Customer_Dig_In_Out_Exec` () in sa poate fi realizata si in alte taskuri sau subrutine.

Atunci cand sarcina comenzii validate `Exec_Command_Nr` (indiferent de **provenienta A sau B**) este comutarea nivelului logic al unei iesiri digitale, se parcurg pasii urmatari.

In **subsectiunea 3.1** se atribuie iesirilor digitale un nume care poate fi un text descriptiv sau forma text a comenzilor care comuta iesirea digitala.

```
SUB Assign_Name_ToDigOut (WORD PersonalDigOut; WORD GeneralDigOut;  
                        STRING CmdOn$; STRING CmdOff$; VAR STRING DigOut_Name$)
```

Campul `CmdOn$` cuprinde comanda in forma text care aduce iesirea digitala in stare **ON** iar campul `CmdOff$` cuprinde comanda in forma text care aduce iesirea digitala in stare **OFF**.

Executia comenzii validate este realizata de subrutina:

```
SUB Switch_Relay_Out (VAR WORD DigOut; BYTE ExecCommandNr; BYTE ActiveLevel;  
                    VAR LONG Timer; LONG TimerLimit; BYTE EventOrNot)
```

Campurile se configureaza astfel:

- `DigOut` ia valoarea iesirii digitale definita personalizat sau generalizat
- `ExecCommandNr` ia valoarea numerica a comenzii
- `ActiveLevel` poate fi **ON** sau **OFF**
- `Timer` si `TimerLimit` stabilesc temporizarea la revenire in stare inactiva. In cazul comenzii netemporizate, `Timer` este configurat **NO\_TIMER** iar `TimerLimit` este configurat **0** sau **NO\_TIMER\_LIMIT**
- `EventOrNot` poate fi **EVENT** sau **NO\_EVENT**, reflectand salvarea sau nu a evenimentului de iesire

### Cazul B ~ comenzi receptionate explicit in forma text

Taskurile sistem specializate in comunicatii receptioneaza comanda in format text provenita pe calea `CmdWay`: **KEY**, **SMS**, **M2M** sau **RS485**.

Taskul sistem `TASK Receive_Command` ruleaza subrutina `ExecuteCustomerCommands (...)` care la randul sau apeleaza subrutina `ValidateReceivedCmd (...)` avand capul de tabel:

```
SUB ValidateReceivedCmd (BYTE CmdWay; BYTE CmdConfirm; STRING CmdName$;  
                       VAR BYTE cmdDetected)
```

Campul `CmdName$` este configurat cu denumirea text a comenzii. Atunci cand comanda receptionata coincide cu comanda introdusa in campul `CmdName$`, subrutina realizeaza:

- Valideaza comanda ~variabila globala `Exe_Command_Nr` ia valoarea numerica a comenzii
- Atribuie variabilei de iesire `cmdDetected` valoarea `TRUE`
- Salveaza evenimentul de comanda cu exceptia cazului `NO_EVENT`

Campul `CmdConfirm` poate fi configurat:

- `IS_ANSWER`; executia comenzii consta in transmiterea unui mesaj elaborat de raspuns. Confirmarea comenzii este asigurata implicit.
- `CONFIRM`; se transmite un raspuns standard de confirmare a comenzii.
- `NO_EVENT`

Atunci cand sarcina comenzii receptionate este modificarea starii logice a unei iesiri digitale, modificarea starii logice se desfasoara in subrutina `Customer_Dig_In_Out_Exe ()`.

### 1.5. Subrutina `ExecuteCustomerCommands (...)`, validarea comenzilor provenite pe calea B

In exemplul de fata subrutina `ConvertCustomerCommands (...)` cuprinde codul pentru realizarea conversiei comenzii din format text in format numeric si viceversa.

Iesirii digitale `RelayOpenDoor` ii este atribuit denumirea comenzii `OPEN_DOOR` in ambele stari logice:

```
CALL Assign_Name_ToDigOut (RelayOpenDoor, Dig_I_O, OPEN_DOOR, OPEN_DOOR, cmdName$)
```

Codul subrutinei `ExecuteCustomerCommands (...)` este edificator:

'calea de provenienta `CmdWay` a comenzii poate fi:

- ' - KEY ~ keyboard-ul statiilor Dispatcher server PC sau Client PC prin Internet sau serial RS232
- ' - SMS ~ telefonul mobil al utilizatorului

```
CALL ValidateReceivedCmd (CmdWay, CONFIRM, OPEN_DOOR, cmdDetected)
```

'variabila globala `Exe_Command_Nr` ia valoarea numerica `OPEN_DOOR_nr`. Comanda este confirmata imediat utilizatorului folosind aceasi cale de provenienta `CmdWay`: KEY sau SMS

**Observatie:** Exista o particularitate a comenzilor receptionate SMS. Un SMS poate cuprinde mai multe comenzi care urmeaza a fi executate individual si confirmate utilizatorului printr-un singur SMS.

Indiferent pe ce cale de provenienta `CmdWay` (KEY sau SMS) a fost receptionata comanda `OPEN_DOOR`, subrutina `CALL ValidateReceivedCmd (...)` valideaza comanda: variabila globala `Exe_Command_Nr` ia valoarea numerica `OPEN_DOOR_nr`.

In exemplul de fata sarcinile comenzii validate sunt:

- Comutarea temporizata a releului de iesire `RelayOpenDoor`
- Salvarea evenimentului de comanda provenit prin Internet sau SMS
- Transmiterea rapoartelor de executie a comenzii pe aceleasi cai de provenienta

## 1.6. Sarcinile subrutinei Customer\_Dig\_In\_Out\_Exe ()

In exemplul de fata subrutina Customer\_Dig\_In\_Out\_Exe () indeplineste sarcinile prezentate in continuare:

### 1.6.1. Scanarea intrarilor digitale generalizate

Este indeplinita de subrutina Read\_Host\_Dig\_Inputs ().

### 1.6.2. Utilizarea intrarilor, iesirilor digitale adresate personalizat

Exista butonul si iesirea digitala personalizata declarate in sectiunea 1:

'echivaleaza valoarea variabilei personalizate cu valoarea variabilei generalizate  
ButtonOpenDoor = First\_Dig\_Input

'variabila personalizata iesire digitala RelayOpenDoor ia valoarea variabilei generalizate  
Relay\_1\_Dig\_Out identificata de taskurile sistem

RelayOpenDoor = Relay\_1\_Dig\_Out

### 1.6.3. Configurarea duratei de inhibitie a intrarilor digitale

Inhibitia intrarilor digitale impiedica reintrarea in stare activa a intrarilor digitale un interval de timp specificat. In acest fel sunt evitate oscilatiile intrarilor digitale.

In exemplul de fata, configurarea campurilor in subrutina

```
CALL Validate_Dig_In (ButtonOpenDoor, LOW_LEVEL, ONE_LEVEL, NO_EVENT, NO_TIMER,  
NO_TIMER_LIMIT, IS_COMMAND)
```

arata ca nu exista timer si durata de inhibitie al timerului. Tranzitia in stare activa a intrarii digitale este validata neconditionat.

### 1.6.4. Modificarea starii logice a iesirilor digitale

#### 1.6.4.1. Iesiri digitale temporizate

Comanda OPEN\_DOOR provine de la distanta (caz B) prin Internet de la statia Dispatcher PC sau SMS, respectiv local de la intrarea digitala ButtonOpenDoor (caz A) si actioneaza temporizat iesirea digitala RelayOpenDoor.

In cazul validarii comenzii (variabila globala Exe\_Command\_Nr a luat valoarea numerica OPEN\_DOOR\_nr in subrutina ExecuteCustomerCommands (...) urmare receptionarii comenzii OPEN\_DOOR prin Internet sau SMS), subrutina Switch\_Relay\_Out (...), mentine in stare activa ON releul RelayOpenDoor pe durata TIMER\_OPEN\_DOOR\_LIMIT si nu salveaza evenimentul de iesire.

```
CALL Switch_Relay_Out (RelayOpenDoor, OPEN_DOOR_nr, ON, TimerOpenDoor,  
TIMER_OPEN_DOOR_LIMIT, NO_EVENT)
```

Subrutina readuce iesirea digitala in stare inactiva OFF de indata ce valoarea variabilei de cronometrare TimerReset depaseste limita temporizarii TIMER\_RESET\_LIMIT. Nu este salvat evenimentul de iesire.

### 1.6.4.2. Iesiri digitale netemporizate

Comenzile [LIGHT\\_ON](#) si [LIGHT\\_OFF](#) provin de la distanta ([caz B](#)) prin Internet de la statia Dispatcher PC sau SMS, respectiv local de la intrarea digitala [SwitchLight](#) ([caz A](#)) si actioneaza fara temporizare iesirea digitala [RelayLight](#).

### 1.6.5. Salvarea evenimentelor independente de factorul uman

Subrutina [Save\\_Extra\\_Events \(Host\)](#) salveaza evenimente independente de factorul uman: conectare la alimentare retea, sincronizare la ceasul atomic al Internetului, sincronizare GSM etc.

Un rol aparte il au evenimentele Internet ale automatelor programabile: conectare, pierderea conexiunii, reconectare automata. Aceste evenimente reflecta calitatea serviciilor Internet in locatia teritoriala a aplicatiei si starea conectivitatii cu statia Dispatcher server PC.

**Observatie:** In programe este intrebuintata variabila [Host](#) (gazda, locatie proprie) desemnand numarul de ordine al Mini PLC4404 in retea RS485. Un Mini PLC4404 care opereaza singur intr-o locatie teritoriala este un Master avand numarul de ordine 0, deci [Host](#) = 0. Variabila [Host](#) este foarte utila in cazul aplicatiilor cu mai multe automate programabile. Se scrie un singur program pentru toate automatele programabile si se folosesc optiunile de selectie.

### 1.6.6. Monitorizarea sincronizarii GSM, conexiunii Internet si conectivitatii cu statia Dispatcher server PC

Atunci cand sunt utilizate comunicatii la distanta este util monitorizarea sincronizarii cu reseaua GSM, conexiunii Internet si conectivitatii cu statia Dispatcher server PC. In cazul automatului programabil Mini PLC4404, sarcina este indeplinita de taskurile si subrutinele sistem.

## 1.7. Subrutina GetCustomerDig\_IO\_Status (...)

In [exemplul de fata](#) subrutina [GetCustomerDig\\_IO\\_Status \(VAR STRING MessageToSend\\$\)](#) livreaza starile logice active sau inactive ale variabilelor [ButtonOpenDoor](#), [RelayOpenDoor](#), [SwitchLight](#) si [RelayLight](#) sub forma unui mesaj de raspuns la comanda generala [Get PLC status](#).

## 1.8. Transmiterea comenzilor Internet de la Dispatcher server PC

Intervine momentul in care dorim sa transmitem o comanda, deschidem Dispatcher server PC si in max 15 secunde, panoul de notificari Dispatcher server PC prezinta:



Conexiunea Internet intre automatul programabil Mini PLC4404 si Dispatcher server PC este realizata. Mesajul este receptionat atunci cand:

- Automatul programabil a realizat propria conexiune Internet
- Automatul programabil a reusit sincronizarea cu ceasul atomic NTP al Internetului. In cazul conexiunii GPRS, sincronizarea NTP poate dura pana la doua minute.

Selectati optiunea de comunicatie **Ethernet** in fereastra principala Dispatcher server PC si transmiteti comanda **open door**. Automatul programabil raspunde cu mesajele:



## 1.9. Salvarea evenimentelor de comanda

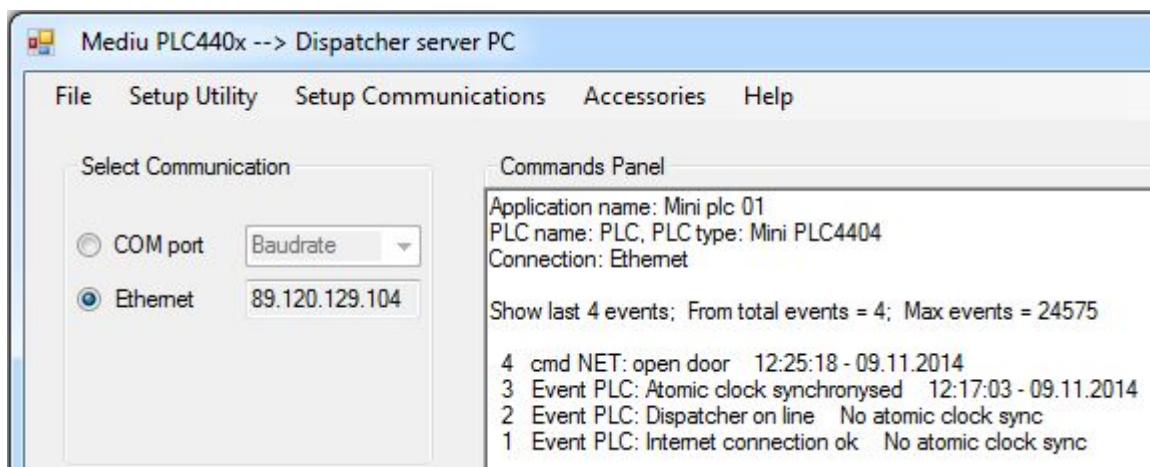
Orice comanda care modifica starea aplicatiei este salvata in FLASH ca eveniment de comanda cuprinzand numele comenzii, variabila modificata, mijlocul de comunicatie utilizat si momentul producerii.

Evenimentele nu pot fi sterse.

Utilizatorul are toata libertatea cand lanseaza o comanda, insa din diverse motive poate gresi. O comanda proasta lansata astazi poate avea efecte nedorite maine.

In acest fel utilizatorul este responsabil pentru comenzile proaste si nu poate nega acest aspect atunci cand este pus in fata evenimentelor din arhiva.

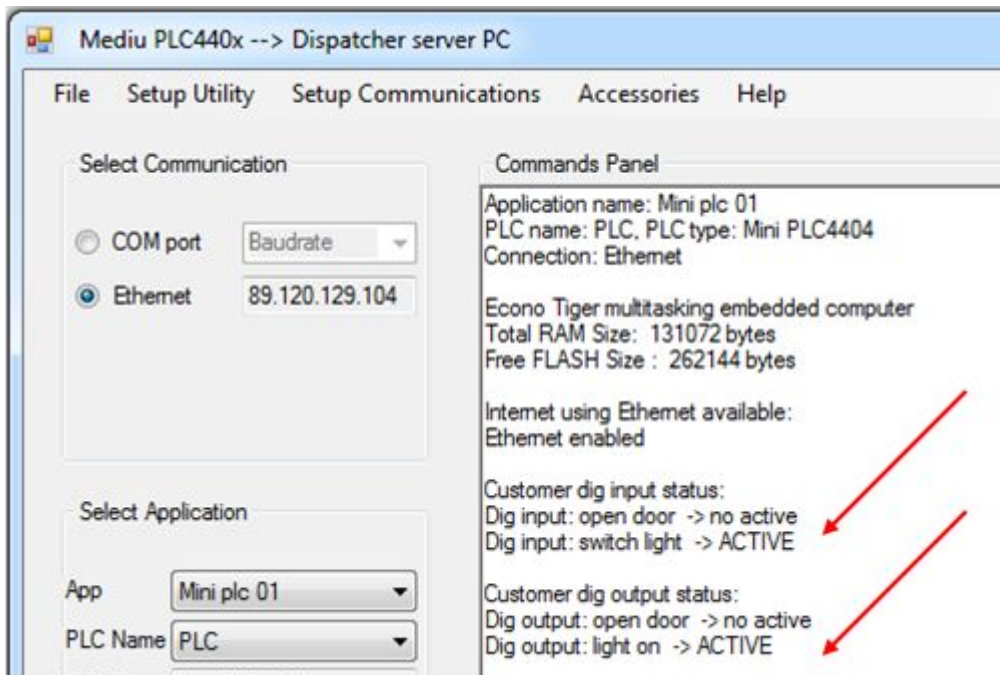
Evenimentele sunt vizualizate cu ajutorul comenzii **Get last events**.



Evenimente de ordinul zecilor de mii sunt salvate incepand de la punerea in functiune a aplicatiei.

## 1.10. Transmiterea comenzilor de la intrarile digitale

Aduceti intrarea digitala [SwitchLight](#) in stare activa (LOW) si transmiteti comanda generala [Get PLC status](#):



Comenzile de aprindere, stingere a luminii pot fi lansate si prin Internet sau SMS. Comanda cea mai recenta are prioritate indiferent de starea comutatorului local [Switch Light](#).

## 2. Aplicatie → Actionarea intrarilor digitale si transmiterea alarmelor

Aplicatia este implementata in [Mini\\_plc\\_02.tig](#) si initiaza utilizatorul in privinta intrarilor digitale ale automatului programabil.

Intrarile digitale au impedanta de intrare 30 k $\Omega$ , sunt protejate la supratensiuni accidentale  $\pm 32V_{cc}$  si sunt marcate vizibil pe placa de circuit imprimat.

Starea normala in vant este [HIGH\\_LEVEL](#) putand fi trase la masa [LOW\\_LEVEL](#). Intrarile digitale sunt conectate la contacte NC sau NO ale unor butoane, comutatoare, relee, senzori PIR etc.

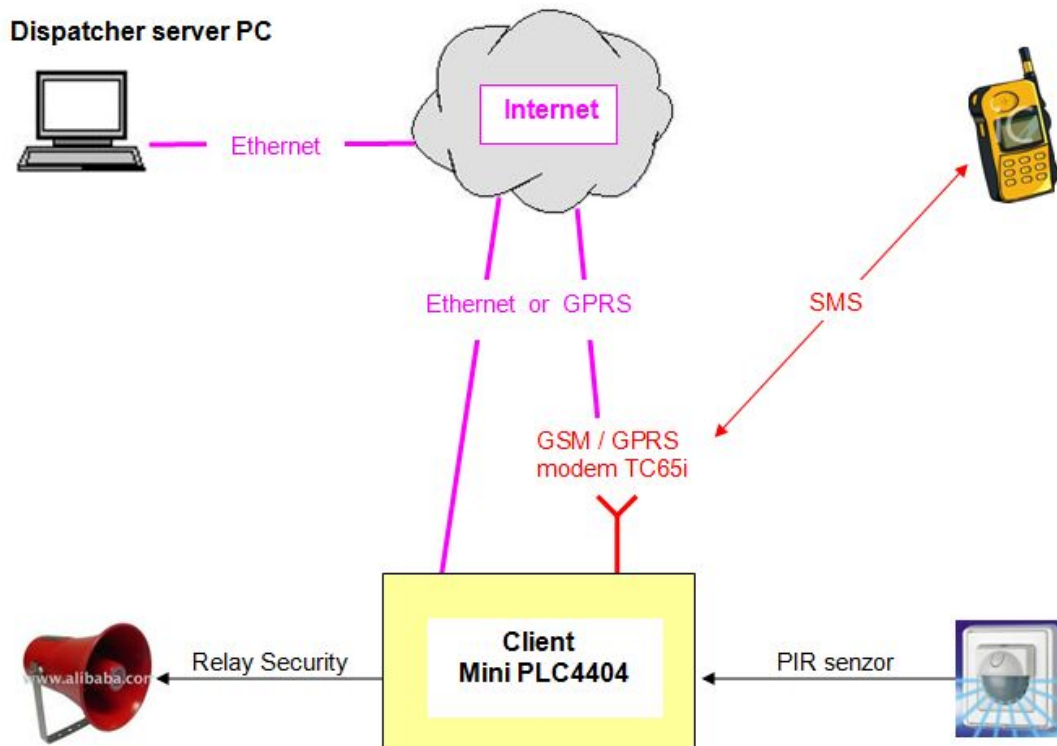


Starea logica activa a unei intrari digitale este declarata in functie de particularitatile dispozitivului de intrare, putand fi [LOW\\_LEVEL](#) sau [HIGH\\_LEVEL](#).

Un buton este activ in stare apasat [LOW\\_LEVEL](#) pe cand in general senzorii de alarma sunt activi in stare [HIGH\\_LEVEL](#). In acest fel sunt puse in evidenta cazuri suplimentare de alarma privind distrugerea carcasei si intreruperea conductoarelor electrice ale senzorului.

Aplicatia este implementata in [Mini\\_plc\\_02.tig](#) si prezinta:

- Un sistem de alarma cu senzor PIR de miscare conectat la o intrare digitala.
- Un releu de iesire este folosit pentru alarma locala generand semnale sonore sau aprinderea unei lumini, un timp limitat.
- Transmiterea mesajelor dealarma prin notificare in Dispatcher server PC, Email si SMS. Alarmerile sunt provocate de comutarea pe nivelul activ al senzorului PIR.
- Evenimentele de comanda si alarma sunt salvate in memoria FLASH si pot fi vizualizate cu ajutorul comenzii [Get last events](#).



**Sectiunea 1** este similara aplicatiei anterioare.

## 2.1. Declararea procedurii SECURITY

Automatul programabil asigura optional securitatea propriei locatii teritoriale in care isi desfasoara activitatea, in mod independent de alte sarcini. Codul program este bine comentat:

```

#define SECURITY
#ifdef SECURITY      'declara senzorul de alarma PIR, variabila personalizata intrare digitala
WORD PIR_Sensor      'conectat la First_Dig_Input, intrare dig. generalizata

'declara releul local de alarma, variabila personalizata iesire digitala
WORD Relay_Security  'conectat la Relay_1_Dig_Out, iesire dig. generalizata

#define TIMER_SECURITY_LIMIT      60000  '60 secunde -> sunete alarma sau
                                       'aprinderea unui reflector pentru
                                       'intimidarea persoanei intruse

LONG TimerSecurity    'variabila numerica, contorizeaza in milisecunde starea ON
                     'pt. Relay_Security limitata de constanta TIMER_SECURITY_LIMIT
  
```

```
'declara comanda pentru releul Relay_Security in forma text si numerica
#define RELAY_SECURITY_ON          "relay security"
#define RELAY_SECURITY_ON_nr      10    'poate fi intre 2...250

'declara cat de des poate fi retransmisa aceeaasi alarma folosind Email si SMS
LONG TimerAlarmInhib_1
```

'se selecteaza doar atunci cand se utilizeaza cu adevarat senzori PIR !!!

```
#define FUNCTIONARE_REALA_PIR
#ifdef FUNCTIONARE_REALA_PIR
#define LIMIT_ALARM_INHIB      1200000  '20 minute in milisekunde
#define LIMIT_ALARM_INHIB      1800000  '30 minute in milisekunde
#else    'pentru experimentari
#define LIMIT_ALARM_INHIB      10000    '10 secunde pentru testari
#define LIMIT_ALARM_INHIB      120000   '2 minute pentru testari
#define LIMIT_ALARM_INHIB      1200000  '20 minute in milisekunde
#endif  'FUNCTIONARE_REALA_PIR
#endif 'SECURITY
```

Variabila `TimerAlarmInhib_1` si constanta `LIMIT_ALARM_INHIB` asigura inhibarea (dezactivarea) intrarii digitale un interval de timp pentru a nu transmite aceeaasi situatie de alarma ori de cate ori persoana intrusa actioneaza senzorul PIR.

In acest fel se evita transmiterea repetata a mesajelor de alarma Emal si SMS.

Toate sarcinile privind identificarea situatiei de alarma, inhibarea intrarii, salvarea in FLASH a evenimentului de alarma si transmiterea mesajelor de alarma sunt indeplinite de subrutina sistem:

```
CALL Validate_Dig_In (PIR_Sensor, HIGH_LEVEL, ONE_LEVEL, ALARM, TimerAlarmInhib,
LIMIT_ALARM_INHIB, RELAY_SECURITY_ON_nr)
```

## 2.2. Proceduri executate de fisierele sistem

Procedura `SECURITY` este indeplinita in mare parte de fisierele sistem si foloseste doua comenzi declarate in fisierele sistem. Aceste comenzi sunt destinate activarii-dezactivarii sistemului de alarma:

```
#define SECURITY_ENABLED          "alarmon"          'valabil si pentru SMS
#define SECURITY_DISABLED        "alarmoff"        'valabil si pentru SMS

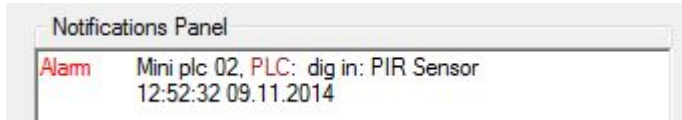
#define SECURITY_ENABLED_nr      253
#define SECURITY_DISABLED_nr     254
```

Procedura `SECURITY` atribuie valorile `YES_` sau `NO_` variabilei globale `SecurityEnabled$`. Valoarea initiala este `YES_`.

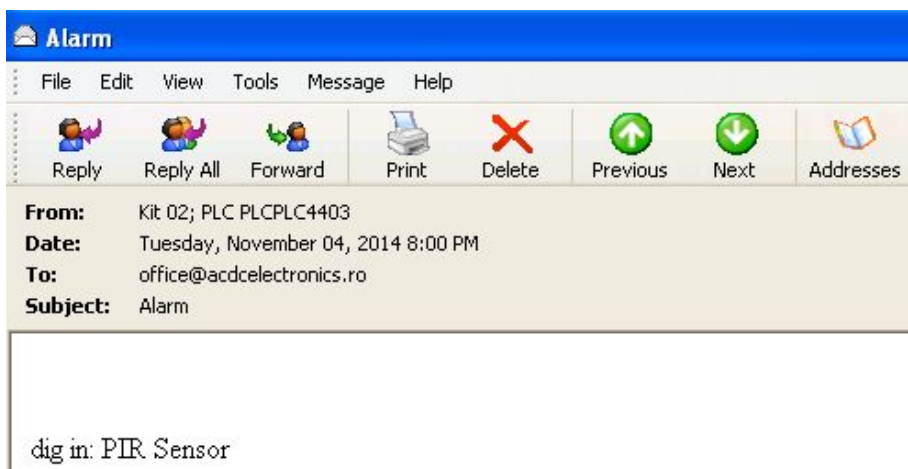
## 2.3. Receptionarea mesajelor de alarma

Mesajele de alarma sunt receptionate pe toate caile de comunicatie disponibile:

- **Notificare** in Dispatcher server PC



- **Email**



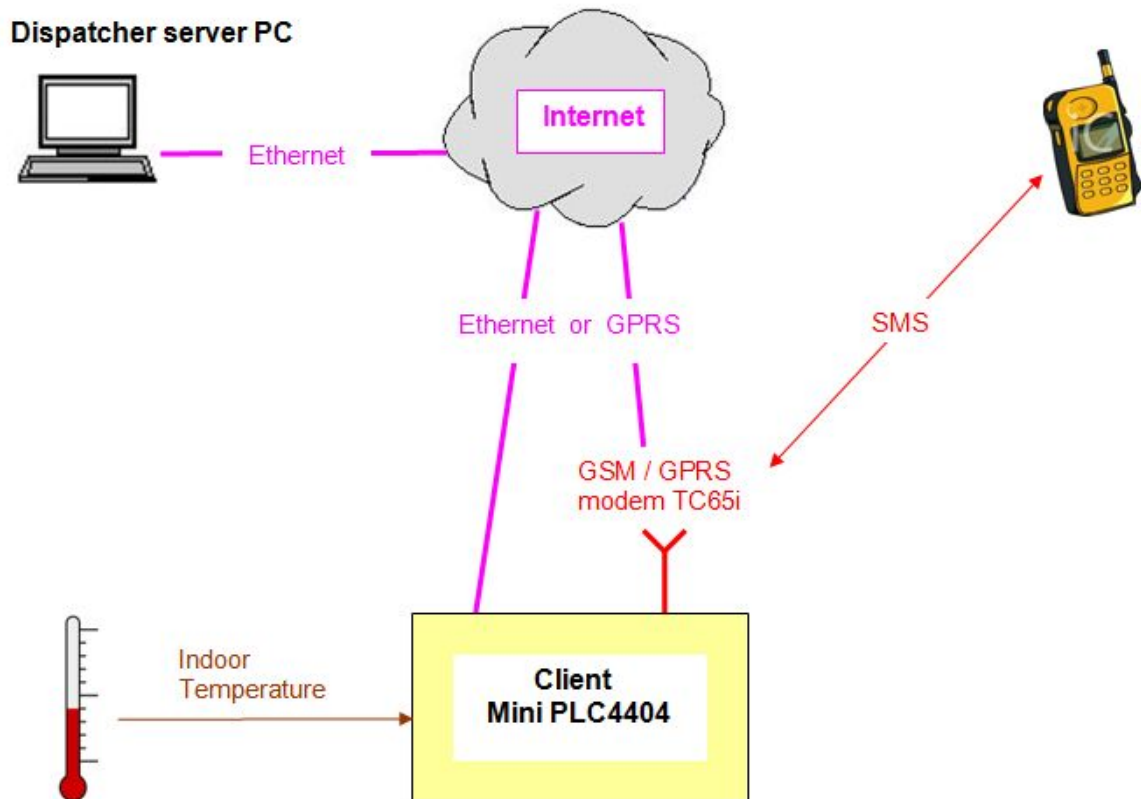
- **SMS**



### 3. Aplicatie → Masurarea pe 12 biti a temperaturii ambiante

Aplicatia este implementata in [Mini\\_plc\\_03.tig](#) si asigura:

- Masurarea pe 12 biti a temperaturii ambiante cu ajutorul traductorului de temperatura KTY81-210 inclus in automatul programabil.
- Transmiterea temperaturii masurate serial RS232, Internet in Dispatcher server PC si SMS.



Declara procedura pentru folosirea canalelor analogice proprii, cuprinsa in [System\\_Analog.inc](#).

```
#define ANALOG
```

```
'Tensiune interna de referinta: 5,00 Volts
```

```
'Rezolutia de masura: 12 bits
```

```
'50 achizitii /secunda
```

```
'----- Declara canalului analogic 0 -----
```

```
'declara numele canalului analogic 0
```

```
#define HOST_ANALOG_CH_0_NAME "Indoor temperature"
```

TASK Analog\_Data este lansat automat de fisierele sistem si indeplineste sarcinile urmatoare:

```
'===== Host analog settings =====
```

```
'stabileste modul de prezentare al marimilor analogice: denumire, unitate de masura, pozitia virgulei: 1,234;  
12,34; 123,4 sau fara virgula 1234.
```

```
CALL Set_Host_Analog_Channel (HOST_ANALOG_CH_0_NAME, GRAD_C, DECIMAL_12DOT34)
```

```
WHILE 1=1
```

```
'livreaza rezultatul conversiei analog numerice
```

```
CALL Get_Analog_12_Bit_AD_Conv_Result ()
```

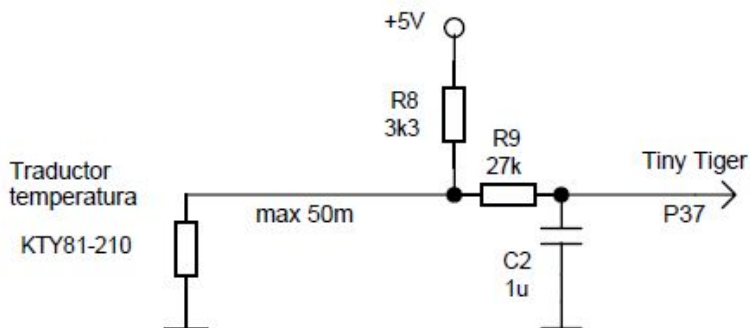
```
'furnizeaza temperatura masurata in interiorul imobilului cu traductorul KTY81-210
```

```
CALL UseTransducer (INT_KTY81_210, HOST_ANALOG_CH_0_NAME)
```

```
WAIT_DURATION 500
```

```
ENDWHILE
```

Schema electronica pentru traductorul KTY81-210 intern sau extern este:



Montajul electronic asigura legea de variatie:

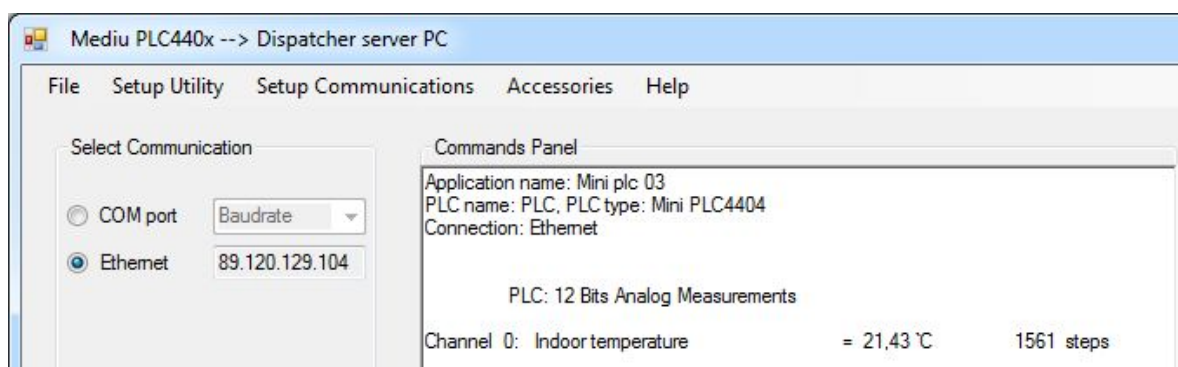
```
Measured_Value (Host, AnChNr) = (AD_Conv_Result (AnChNr) * 53727) / 4096 -17749 -150
```

Vezi SUB UseTransducer() in System\_Analog.inc.

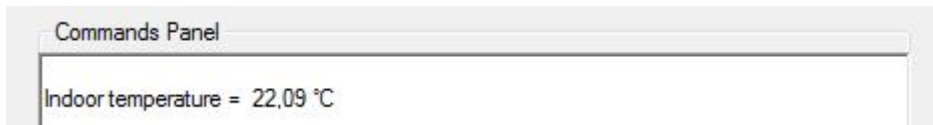
### 3.1. Transmiterea comenzii de la Dispatcher server PC

Sunt doua posibilitati:

1. Transmiteti serial RS232 sau in Internet comanda generala [Get PLC analog status](#):



2. Transmiteți serial RS232 sau în Internet comanda specifică [Get Indoor temperature](#):



În acest scop se declară comanda aplicației:

```
#define GET_TEMPERATURE          "Get indoor temperature"
```

Comanda provenind de la KEYboardul stației Dispatcher server PC este recepționată și executată în subrutină [ExecuteCustomerCommands](#) (BYTE CmdWay)

Comanda nu modifică starea aplicației, din acest motiv nu este salvat evenimentul de comandă. În acest caz sunt două soluții:

1. Folosirea subrutinei consacrate [ValidateReceivedCmd](#) (...)

'Transmite valoarea Temperaturii măsurate prin RS232 și Internet.

```
BYTE cmdDetected
```

```
CALL ValidateReceivedCmd(CmdWay, NO_EVENT, GET_TEMPERATURE, cmdDetected)
```

```
IF cmdDetected = TRUE THEN
```

```
  STRING SendMessage$(50)
```

```
  'Transmite valoarea măsurată a Temperaturii interioare, cu două zecimale
```

```
  SendMessage$ = CRLF + HOST_ANALOG_CH_0_NAME + " = " + STR$(Measured_Value(0, 0),  
    PRINT_USING_12DOT34) + " " + GRAD_C
```

```
  CALL SendSerialAllFeatures(SendMessage$, "", KEY)
```

```
ENDIF
```

2. Identificarea directă a comenzii în variabila globală [Keyboard\\_Subcommand\\$](#)

'Transmite valoarea Temperaturii măsurate prin RS232 și Internet.

```
IF CmdWay = KEY AND INSTR(GET_TEMPERATURE, Keyboard_Subcommand$, 0, 0) > -1 THEN
```

```
  STRING SendMessage$(50)
```

```
  'Transmite valoarea măsurată a Temperaturii interioare, cu două zecimale
```

```
  SendMessage$ = CRLF + HOST_ANALOG_CH_0_NAME + " = " + STR$(Measured_Value(0, 0),  
    PRINT_USING_12DOT34) + " " + GRAD_C
```

```
  CALL SendSerialAllFeatures(SendMessage$, "", KEY)
```

```
ENDIF
```

## 3.2. Transmiterea comenzii prin SMS

În acest scop se declară o comandă SMS mai scurtă:

```
#ifdef GSM
```

```
  'Utilizatorul transmite prin SMS comanda pentru recepționarea temperaturii măsurate.
```

```
  'Declara sintagma folosită în comandă SMS
```

```
  #define SMS_TEMPERATURE          "temperatura"
```

```
  'Sintagma "temperatura" transmisă de utilizator, este mai scurtă decât comanda
```

```
  "'Get indoor temperature" și produce același efect.
```

```
#endif
```

Comanda prin SMS este executata in subrutina [ExecuteCustomerCommands \(BYTE CmdWay\)](#)

'Transmite valoarea Temperaturii masurate prin SMS.

```
#ifdef GSM
  IF CmdWay = SMS AND INSTR(SMS_TEMPERATURE, Received_SMS_Message$, 0, 4) > -1 THEN
    'pregateste mesajul SMS de raspuns
    Send_SMS_Message$ = Send_SMS_Message$ + HOST_ANALOG_CH_0_NAME + " = " +
                        STR$(Measured_Value(0, 0), PRINT_USING_12DOT34) + " " + GRAD_C
  ENDIF
#endif
```

Nota: Identificarea comenzilor SMS care implica marimi analogice are intotdeauna o solutie particulara, conform cazului de mai sus. Comanda este executata de taskurile sistem cuprinse in [System\\_Gsm.inc](#).



Nota: Avantajul comenzilor SMS este ca un singur mesaj SMS poate cuprinde mai multe comenzi distincte, separate prin orice caracter mai puțin punct și virgula, în limita a max 160 caractere. Vezi [House\\_Heating.tig](#).

## Simularea comenzilor SMS

Este foarte utila la elaborarea programelor. In [System\\_Gsm.inc](#) selectati primele doua optiuni:

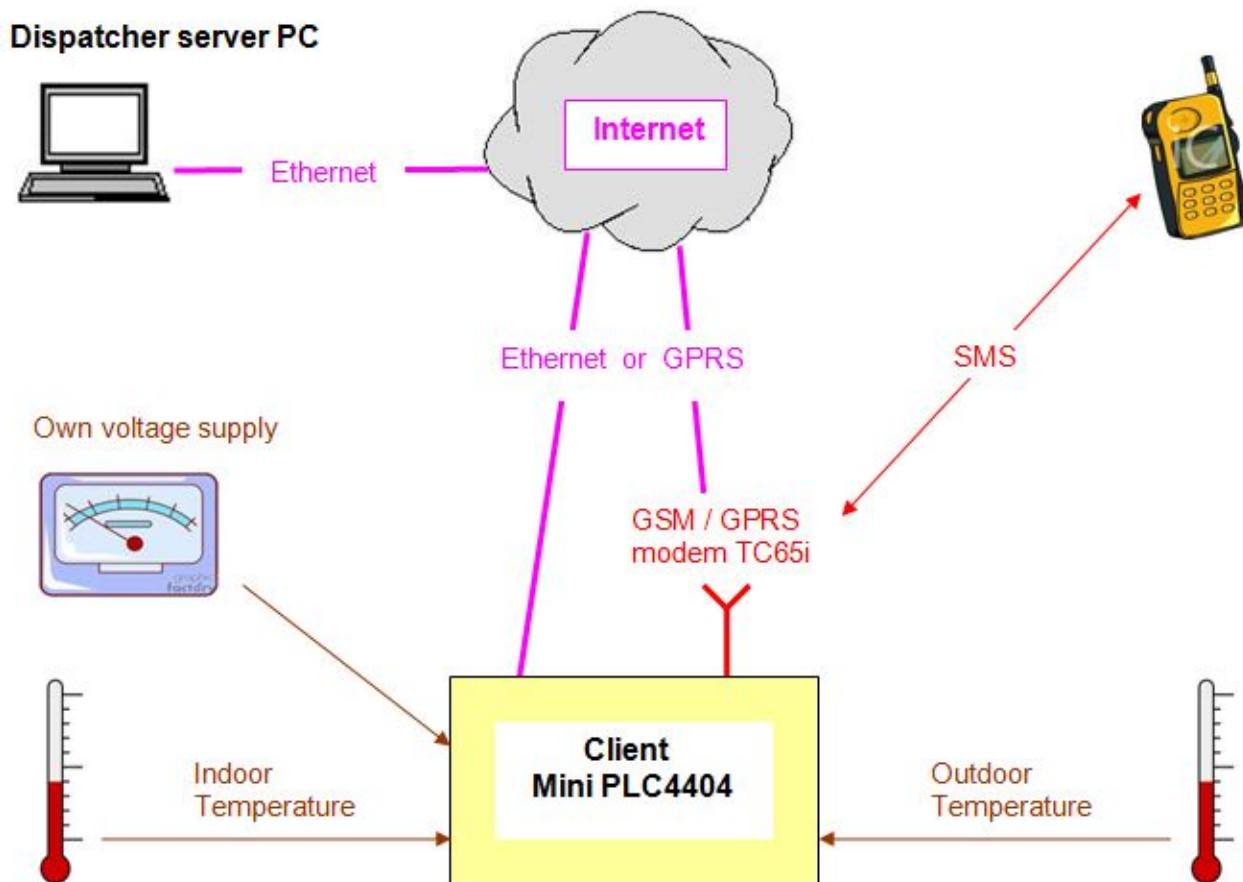
```
#define GSM_REPORT_ENABLED          'print debugg info on Ser1, 38400 baud
#define SIMULATE_SMS                'if enabled no SMS will be sent!
```

Introduceti temporar secventa de cod in [TASK MAIN](#):

```
WAIT_DURATION 1000      'milisecunde
WHILE 1 = 1
  STRING SimulateSmsCmd$(30)
  IF BIT(First_Dig_Input,0) = 0 THEN          'se aduce 1 secunda manual la masa prima intrare digitala
    PRINT #Ser, #1, CRLF; "BIT(First_Dig_Input,0) = 0"      'monitorizare cu Terminal Windows
    WAIT_DURATION 2000

    SimulateSmsCmd$ = "temPeraTura = 24,1"
    CALL SimulateIncommingSms(SimulateSmsCmd$, "0745659909")
  ENDIF
  WAIT_DURATION 200
ENDWHILE
```

#### 4. Aplicatie → Programarea limitelor marimilor analogice masurate



Programul [Mini\\_plc\\_04.tig](#) foloseste propriile intrari analogice ale automatului programabil:

- Temperatura interioara este masurata de traductorul KTY81-210 montat in interiorul automatului programabil.
- Temperatura exterioara este masurata de traductorul KTY81-210 livrat impreuna cu automatul programabil. Traductorul se conecteaza cu un cablu bifilar 2x0,75 mm<sup>2</sup> la distanta max 50 m.
- Tensiunea de alimentare e masurata de divizorul rezistiv intern.

Programul foloseste procedura limitelor marimilor analogice, cuprinsa in [System\\_Mes\\_Limits\\_And\\_Data\\_Aquisition.inc](#)

Limitele marimilor analogice sunt folosite in patru scopuri:

- Salvarea evenimentelor de comanda atunci cand utilizatorul modifica valoarea limitelor
- Salvarea evenimentelor atunci cand valoarea masurata depaseste valoarea limitei
- Transmiterea mesajelor de alarma at cand valoarea masurata depaseste valoarea limitei
- Inchiderea buclilor de reglaj automat → vezi [Mini\\_plc\\_07.tig](#)

Programul [Mini\\_plc\\_04.tig](#) prezinta primele trei scopuri.

Aplicatia este configurabila pe baza optiunilor selectate in program.

'Declara provenienta marimilor analogice:

```
#define USE_OWN_PLC_ANALOG_INPUTS "OwnAI"
```

'Declara particularitatile fiecarei marimi analogice:

```
#define FIRST_MES_NAME "Temp inside"
#define FIRST_MES_CHANNEL 0 'masoara pe canalul analogic 0
#define FIRST_MES_DOT_POSITION DECIMAL_12DOT34 'ex ±0,12 ... ±12,34 °C
#define FIRST_MES_MES_UNIT GRAD_C °C
```

```
#define SECOND_MES_NAME "Temp outside"
#define SECOND_MES_CHANNEL 2 'masoara pe canalul analogic 2
#define SECOND_MES_DOT_POSITION DECIMAL_12DOT34 'ex ±0,12 ... ±12,34 °C
#define SECOND_MES_MES_UNIT GRAD_C °C
```

```
#define THIRD_MES_NAME "Own voltage"
#define THIRD_MES_CHANNEL 1 'masoara pe canalul analogic 1
#define THIRD_MES_DOT_POSITION DECIMAL_12DOT34 'ex ±0,12 ... ±12,34 V
#define THIRD_MES_MES_UNIT "V" 'Volți
```

'Declara procedura de utilizare a limitelor, cuprinsa in [System\\_Mes\\_Limits\\_And\\_Data\\_Aquisition.inc](#)

```
#define USE_MEASURES_LIMITS
```

Declara daca aplicatia opereaza cu limite pentru evenimente si alarme. Se definesc parametrii limitelor pentru fiecare marime in parte.

'Nu se declara limite pentru a treia valoare analogica masurata = propria tensiune de alimentare

'Declara daca Temperatura interioara foloseste limite programate de utilizator

'Declara mesajul de comanda in forma text caruia i se atribuie un numar unic in gama 1...10

```
#define FIRST_MES_CMD_CHANGE_LIMIT "Utilizator modifica limita temp int la "
#define FIRST_MES_CMD_CHANGE_LIMIT_nr 1
```

```
#ifndef FIRST_MES_CMD_CHANGE_LIMIT
```

'Declara limitele de functionare normala a Temperaturii interioare

```
#define FIRST_MES_MIN_VAL_LIMIT 5 °C
#define FIRST_MES_MAX_VAL_LIMIT 27.5 °C
#define FIRST_MES_LIMIT_INCREMENT 2.5 °C
```

'Valori limita in gama 5...27,5°C pot fi selectate in Dispatcher server PC, panoul Valori masurate si limite.

'Valorile limita pot fi editate de utilizator cu precizie de doua zecimale.

'Programul corecteaza valori introduse de utilizator in afara limitelor de functionare normala.

'Declara scopul utilizarii limitelor programate de utilizator.

'Declara daca Temperatura interioara foloseste evenimente sau alarme sau nu foloseste.

'Selecteaza una din optiuni sau niciuna:

```
#define FIRST_MES_EVENT_ALARM ANALOG_EVENT 'salveaza evenimentele in 'FLASH
#define FIRST_MES_EVENT_ALARM ANALOG_ALARM 'transmite mesaje de 'alarma prin Email si SMS
```

```
#ifndef FIRST_MES_EVENT_ALARM
```

'Temperatura interioara foloseste mesaje text dedicate in caz de eveniment sau alarma.

'Fiecarui mesaj text ii este alocat un numar unic in gama 1...10.

'Defineste mesajele folosite in caz de eveniment sau alarma.

```
#define FIRST_MES_RISE_ABOVE_LIMIT "Temp int urca peste "
#define FIRST_MES_RISE_ABOVE_LIMIT_nr 2
```

```
#define FIRST_MES_DROP_BELOW_LIMIT "Temp int coboara sub "
#define FIRST_MES_DROP_BELOW_LIMIT_nr 3
```

'Declara valoarea histerezei  
`#define FIRST_MES_HYSTERESIS` "5" '5°C, valoare editabila

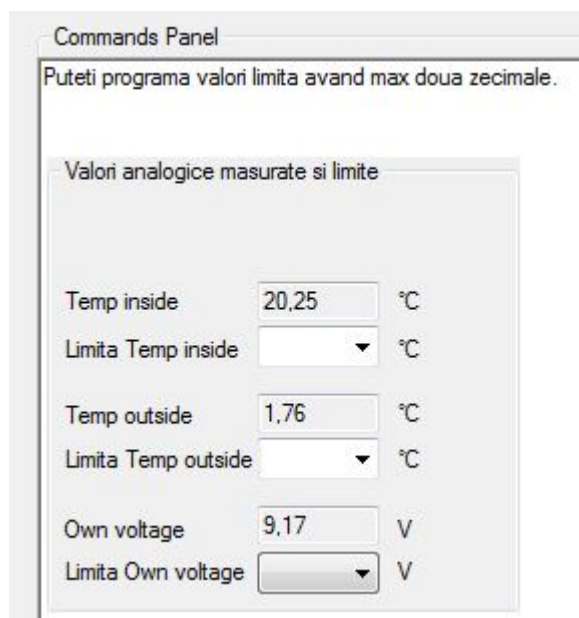
'Selecteaza una din optiuni:  
`#define FIRST_MES_ABOVE_BELOW` ABOVE 'depasirea in sus a limitei  
`#define FIRST_MES_ABOVE_BELOW` BELOW 'depasirea in jos a limitei

```
#endif 'FIRST_MES_EVENT_ALARM  
#endif 'FIRST_MES_CMD_CHANGE_LIMIT
```

In subrutina sistem `DeliverCustomerCommands(...)` se declara comanda sistem `GET_MES_VAL_AND_LIMITS`

In final, subrutina sistem `ExecuteCustomerCommands(...)` apeleaza subrutina sistem `ExecuteCustomerLimitsCommands(...)`.

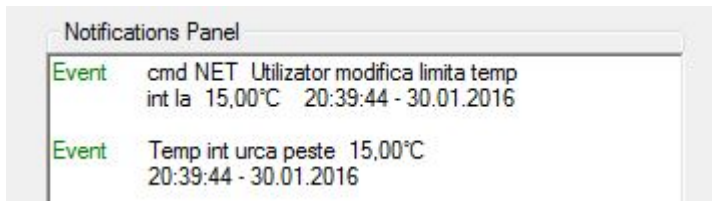
Procedura limitelor marimilor analogice este implementata in Dispatcher server PC. Selectati comanda [Get values and set limits](#).



Limitele marimilor analogice pot fi modificate astfel:

- Se selecteaza valori in casutele limitelor
- Se editeaza manual valori cu precizie max doua zecimale

Modificati limita Temperaturii interioare la 15°C si apasati butonul [Send Command](#).  
Panoul de notificari prezinta:



Primul eveniment atesta comanda utilizatorului.

Al doilea eveniment atesta ca Temperatura interioara (20,25°C) a depasit valoarea limita programata de utilizator (15°C).

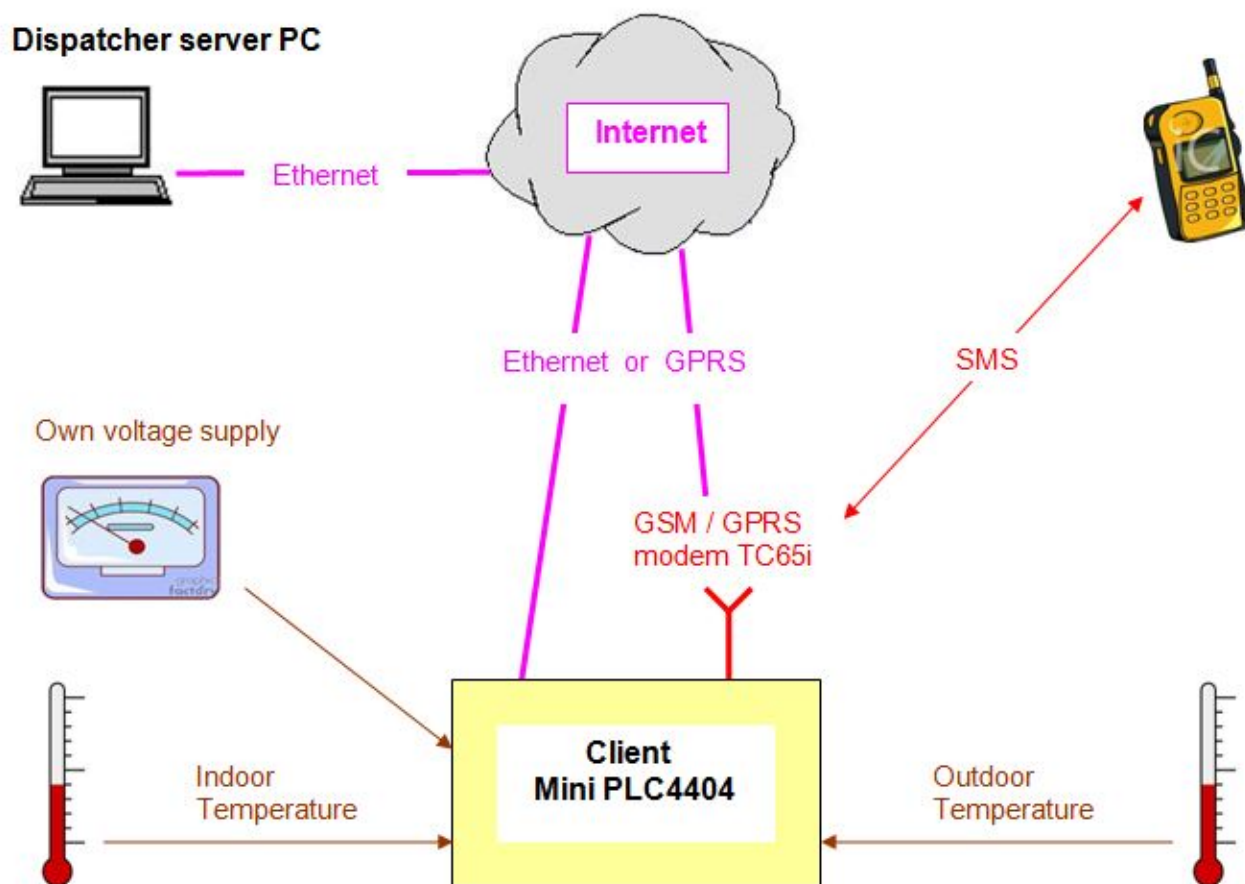
Atunci cand valorile masurate depasesc limitele programate de utilizator, [TASK EventAlarm](#) se ocupa de salvarea evenimentelor si transmiterea mesajelor de alarma in Dispatcher server PC, Email si SMS.

```
#ifdef FIRST_MES_EVENT_ALARM                                'Temperatura interioara
  IF FIRST_MES_ABOVE_BELOW = ABOVE THEN
    CALL SaveEvent_SendAlarm (Host, 0, AnalogLimitValue(Host, 0), HysteresisVal(0),
                              TEMP_INT_RISE_ABOVE_LIMIT)
  ELSE 'cazul BELOW
    CALL SaveEvent_SendAlarm (Host, 0, AnalogLimitValue(Host, 0), HysteresisVal(0),
                              TEMP_INT_DROP_BELOW_LIMIT)
  ENDIF
#endif
```

Salvarea in Flash a evenimentelor de comanda se face in [SUB ExecuteCustomerCommands \(\)](#) de catre subrutina [ExecuteCustomerLimitsCommands\(CmdWay\)](#).

Salvarea in Flash a evenimentelor la depasirea limitelor programate de utilizator si transmiterea mesajelor de alarma In Dispatcher server PC, Email si SMS sunt asigurate de procedurile sistem cuprinse in [System\\_Mes\\_Limits\\_And\\_Data\\_Aquisition.inc](#) si [System\\_Events.inc](#).

## 5. Aplicatie → Achizitia in SRAM a valorilor analogice si prezentarea graficelor



Programul este [Mini\\_plc\\_05.tig](#) si foloseste propriile intrari analogice ale automatului programabil:

- Temperatura interioara este masurata de traductorul KTY81-210 inclus in automatul programabil.
- Temperatura exterioara este masurata de traductorul KTY81-210 livrat impreuna cu automatul programabil. Traductorul se conecteaza cu un cablu bifilar 2x0,75 mm<sup>2</sup> la distanta max 50 m.
- Tensiunea de alimentare e masurata de divizorul rezistiv intern.

Procedura grafica este implementata in:

- [System\\_Mes\\_Limits\\_And\\_Data\\_Aquisition.inc](#)
- Dispatcher server PC.

Valorile analogice sunt achizitionate din 5 in 5 minute. Pe durata unei zile se acumuleaza astfel 288 puncte.

In urma comenzilor transmise de la Dispatcher server PC, automatul programabil livreaza valorile achizitionate, prezentate grafic pe durata ultimelor 14~28 zile, in functie de ziua selectata in calendar.

Automatul programabil achizitioneaza valorile masurate in SRAM-ul ramas neutilizat:

```
#define SAVE_DATA_IN_SRAM          "SRAM"
```

Orice program incarcat in automatul programabil foloseste doar o parte din SRAM-ul total:

- Cea mai mare parte este utilizata de functiile si procedurile sistemului de operare Tiger BASIC insusi.
- Alte resurse SRAM sunt consumate de programul de fata impreuna cu fisierele [.inc](#) utilizate.

In prezenta aplicatie contam pe minim 12K SRAM ramas disponibil.

```
#define SRAM_LENGT          12096 'asigura salvarea valorilor pe durata a 14 zile -> trei marimi analogice
```

Procedura grafica elaborata in Dispatcher server PC asigura:

- Scalare automata pe axele Y si X. Axa X reprezinta timpul
- Specificarea zilei sau intervalului in care sunt prezentate graficele
- Prezentarea 1~3 grafice in aria geometrica disponibila

'Declara portiunile verticale neutilizate ale fiecarui grafic:

```
#define FIRST_GRAPH_BORDER      1      '+ - 1 °C in sus si in jos fata de valorile masurate extreme  
#define SECOND_GRAPH_BORDER    1      '+ - 1 °C in sus si in jos fata de valorile masurate extreme  
#define THIRD_GRAPH_BORDER     1      '+ - 1 V in sus si in jos fata de valorile masurate extreme
```

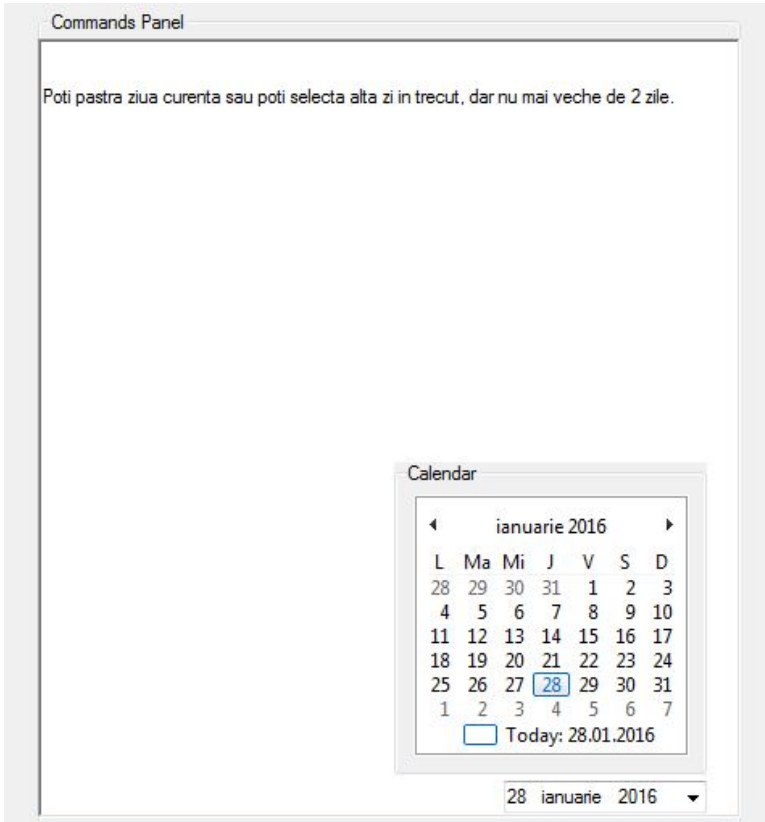
Nota: La experimentarea aplicatiilor este foarte util simularea intervalelor de timp pentru achizitia datelor.

'se valideaza doar pentru simulari necesare la elaborarea aplicatiei !!!!

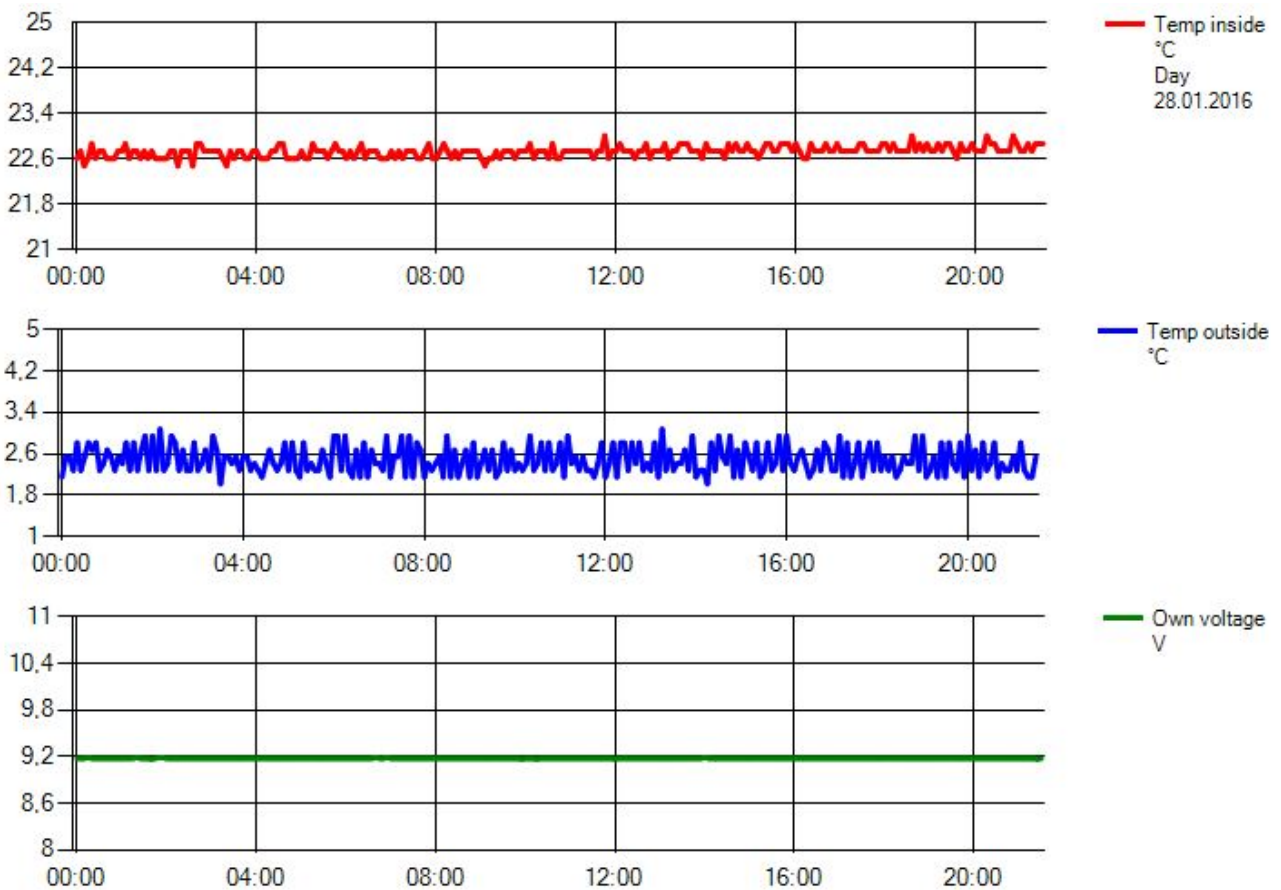
```
#define SIMULATE_SAVING_INTERVAL  
#ifdef SIMULATE_SAVING_INTERVAL  
    'Similarile sunt foarte utile la elaborarea si verificarea aplicatiei  
    'Durate lungi de timp, de ordinul saptamanilor sunt simulate in cursul mai multor ore  
    #define SIMULATED_SAVE_INTERVAL_IN_SECONDS      2  
    '2 secunde -> 288 masuratori se efectueaza simulat in 576 secunde = 9,6 minute  
#else 'functionare normala Internet  
#endif 'SIMULATE_SAVING_INTERVAL
```

In final, subrutina sistem `ExecuteCustomerCommands(...)` apeleaza subrutina sistem `ExecuteGraphCommands_When_SaveDataInSram(...)`.

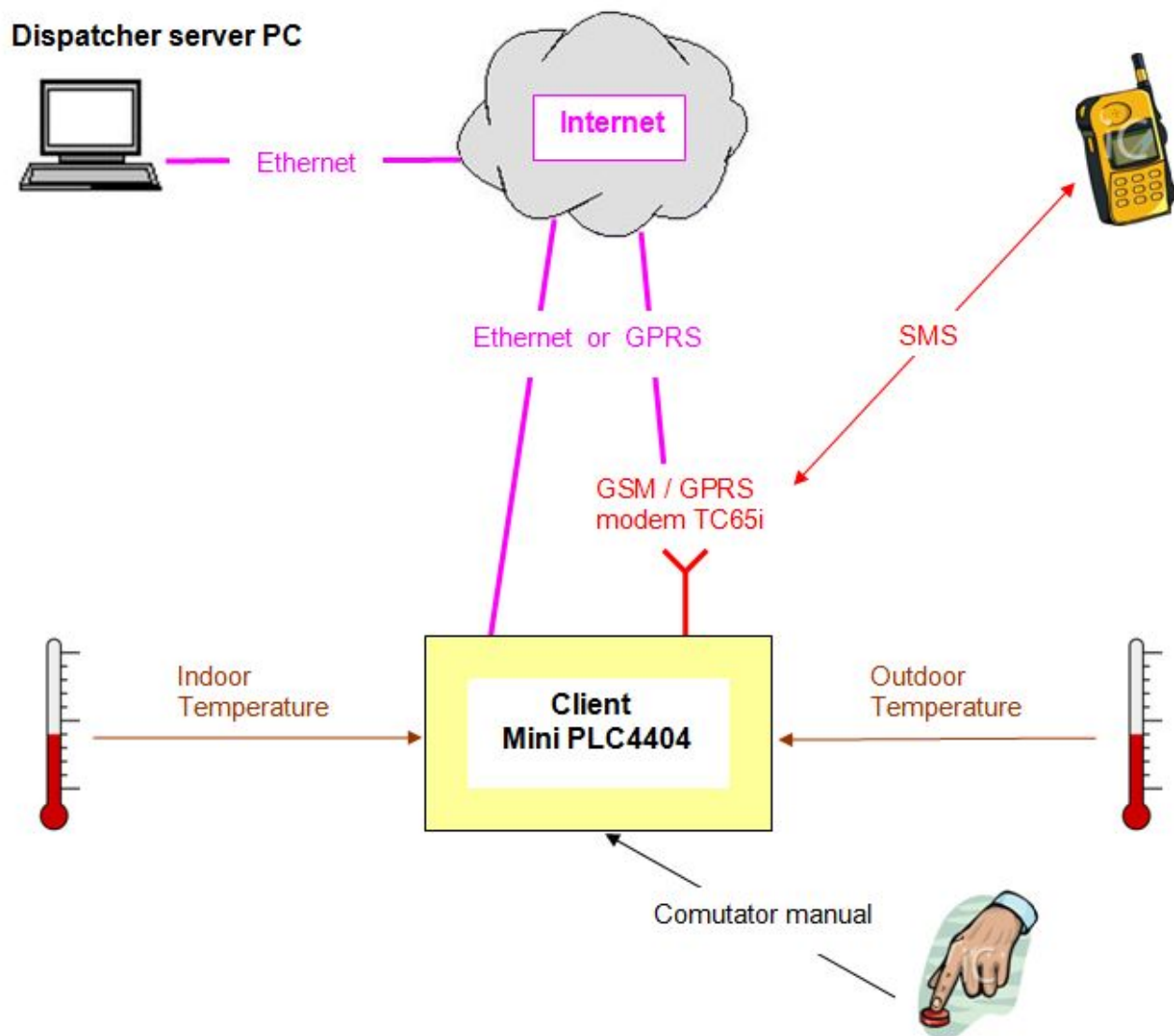
Selectati comanda comanda [Get graph data](#) in Dispatcher server PC



Selectati in calendar ziua dorita. Pentru ziua curenta se obtin graficele:



## 6. Aplicatie → Achizitia in SRAM a valorilor analogice, digitale si prezentarea graficelor



Programul este [Mini\\_plc\\_06.tig](#) si foloseste doar resursele automatului programabil:

- Temperatura interioara este masurata de traductorul KTY81-210 inclus in automatul programabil.
- Temperatura exterioara este masurata de traductorul KTY81-210 livrat impreuna cu automatul programabil. Traductorul se conecteaza cu un cablu bifilar 2x0,75 mm<sup>2</sup> la distanta max 50 m.

In practica se cer uneori grafice pentru evolutia in timp a marimilor digitale.

In scop demonstrativ, utilizatorul comuta manual un comutator pe prima intrare digitala a automatului programabil.

Nu se declara a treia marime analogica. In locul ei se declara marimea digitala care pastreaza toate atributele marimii analogice pe care o inlocuieste.

Configurarile sunt facute automat in [System\\_Mes\\_Limits\\_And\\_Data\\_Aquisition.inc](#).

```
'Declara marimea digitala
#define THIRD_DIGITAL_NAME          "This Input"
```

Valorile marimii digitale sunt determinate in [SUB Customer\\_Dig\\_In\\_Out\\_Exe\(\)](#).

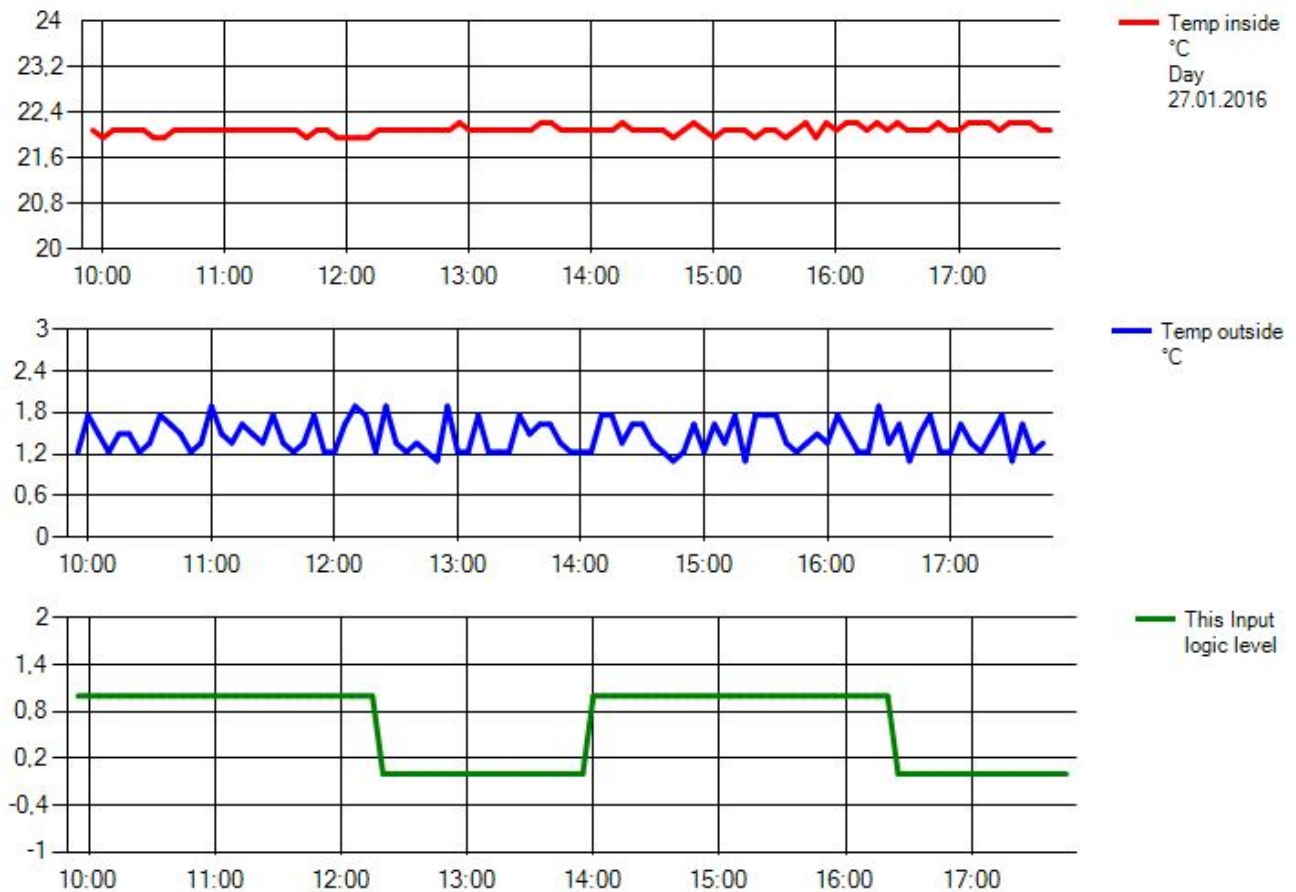
```
#ifndef THIRD_DIGITAL_NAME
    DigitalValue(Host, 2) = First_Dig_Input
#endif 'SECOND_DIGITAL_NAME
```

Pentru economie de timp se selecteaza optiunea `#define SIMULATE_SAVING_INTERVAL`.

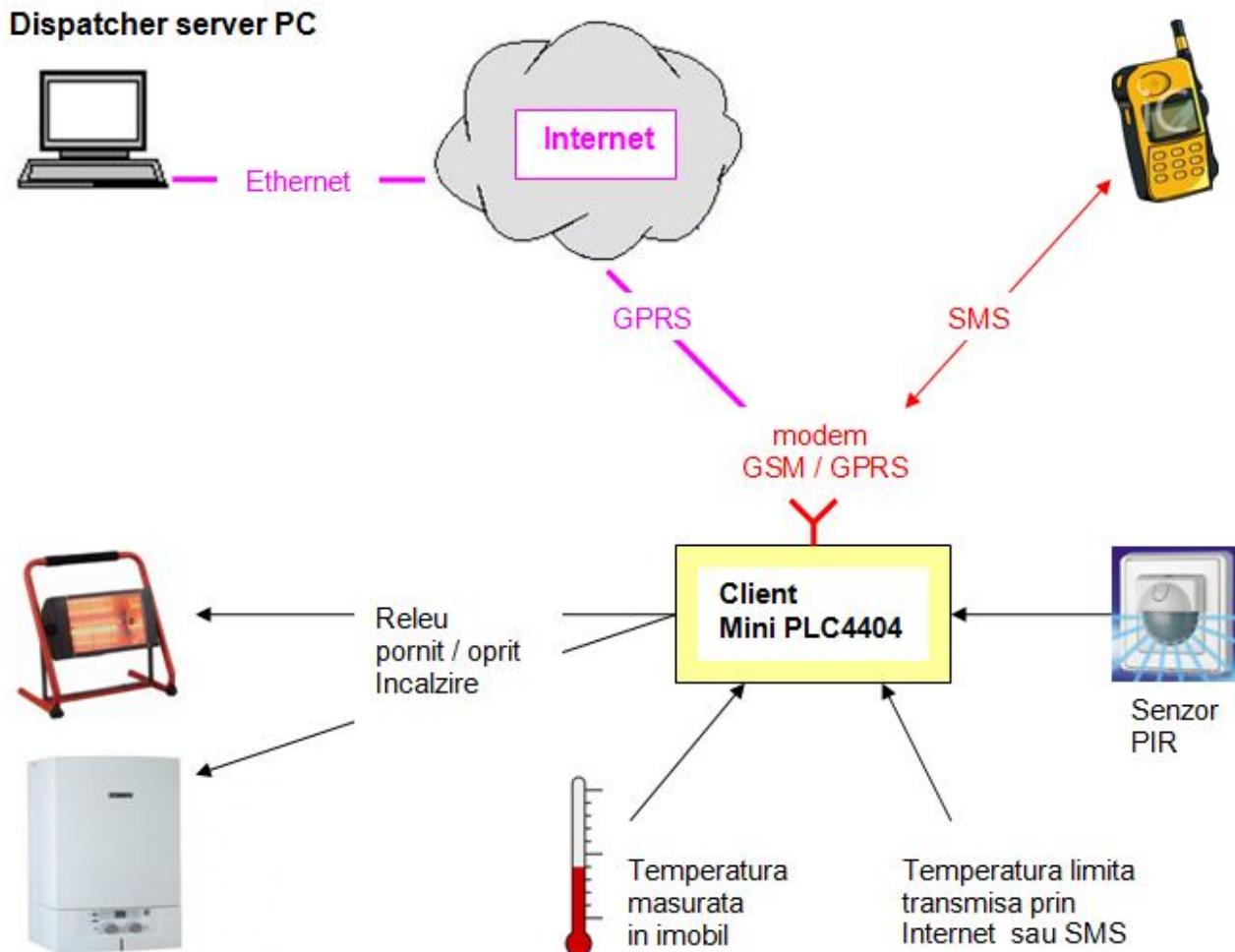
Selectati comanda [Get values and limits](#).

Selectati in calendar ziua dorita.

Apasati butonul [Send Command](#).



## 7. Aplicatie → Controlul in bucla inchisa al temperaturii, salvarea evenimentelor, transmiterea mesajelor de alarma si prezentarea graficelor



Programul este [Mini\\_plc\\_07.tig](#) si foloseste traductorul de temperatura KTY81-210 intern care masoara temperatura ambianta intr-una dintre camere.

Automatul programabil receptioneaza prin Internet sau SMS temperatura limita (de referinta) transmisa de utilizator.

Sunt utilizate diverse surse de caldura proiectate pentru a fi controlate de termostate locale: radiatoare electrice, panouri radiante murale, centrale termice pe gaz sau alt combustibil.

Automatul programabil indeplineste functia de termostat cu limita de temperatura comandata la distanta.

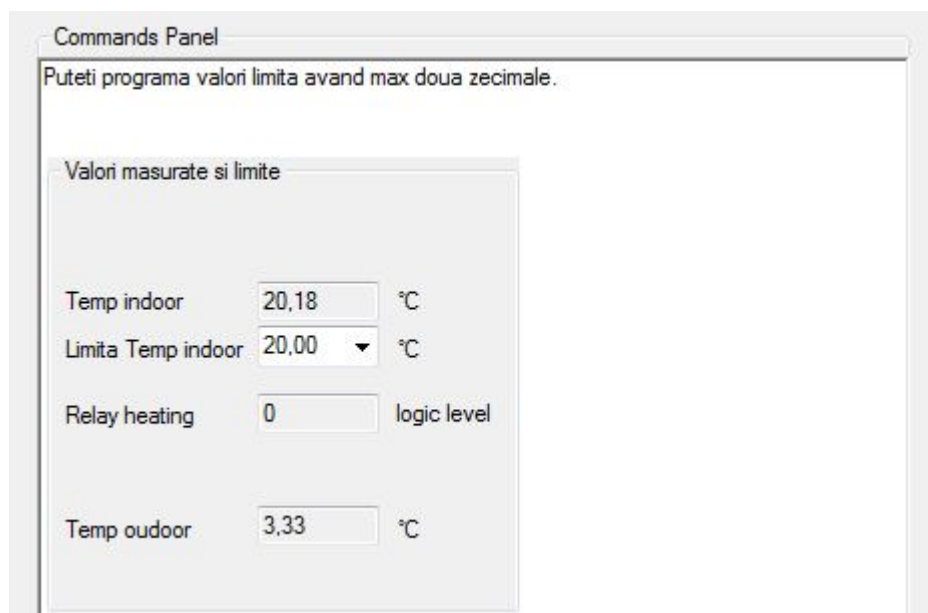
**Termostatul local este inlocuit de Mini PLC4404** care porneste si opreste in mod repetat sursa de caldura astfel incat temperatura masurata sa fie egala cu temperatura limita. Precizia asigurata este de  $\pm 0,3 \text{ }^\circ\text{C}$ , conform histerezei definita in program.

```
#define TEMP_HYSTERESIS          "0,30"          '+0,3 °C
```

## 7.1. Modificarea limitei de temperatura in Internet

Transmiteti comanda generala [Enable customer commands](#) si parola.

Selectati comanda [Get values and set limits](#)



The screenshot shows a web interface titled "Commands Panel". At the top, it says "Puteti programa valori limita avand max doua zecimale." Below this is a section titled "Valori masurate si limite". It contains four rows of controls: "Temp indoor" with a text input field containing "20,18" and a "°C" label; "Limita Temp indoor" with a dropdown menu showing "20,00" and a "°C" label; "Relay heating" with a text input field containing "0" and a "logic level" label; and "Temp outdoor" with a text input field containing "3,33" and a "°C" label.

Panoul prezinta valoarea temperaturii interioare, limita initiala a temperaturii interioare, starea logica a releului de iesire si valoarea temperaturii exterioare.

Modificarea limitei de temperatura se face astfel:

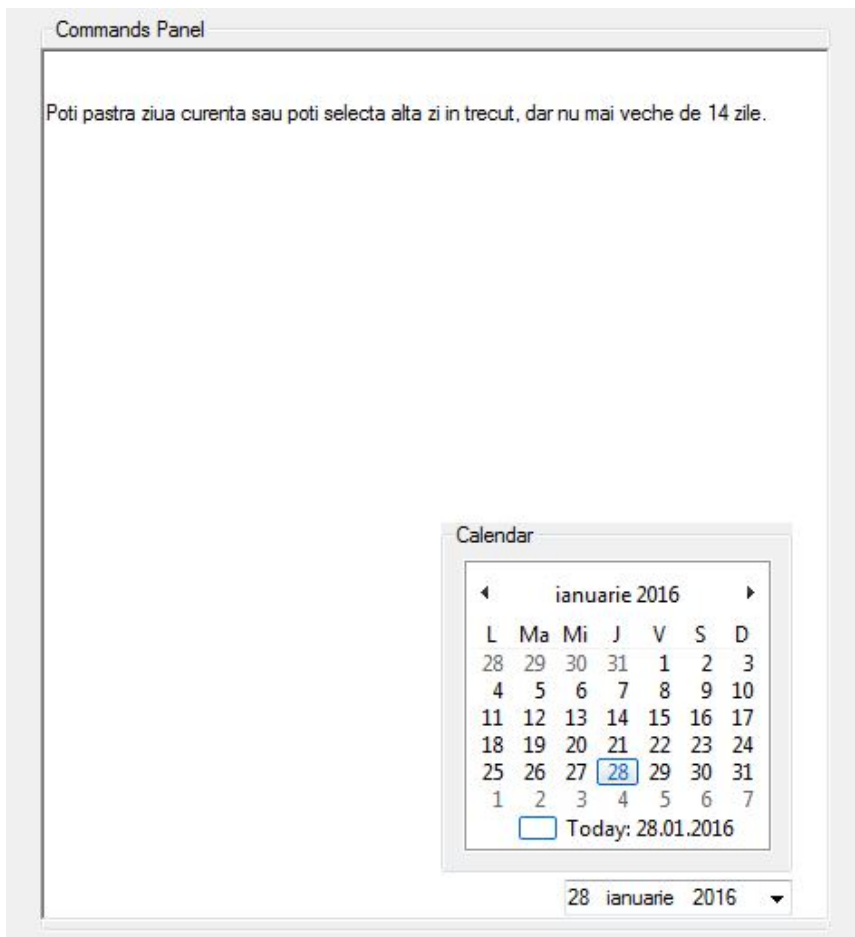
- Selectati alta valoare in casuta limitei
- Scrieti alta valoare de exemplu [22](#). Se pot introduce valori cuprinse intre [4](#) ~ [26](#) °C. Valorile pot contine zecimale ex. [21,34](#) sau [21.7](#) Se foloseste punctul sau virgula zecimala

Apasati butonul [Send Command](#). Noua valoare limita este transmisa automatului programabil.

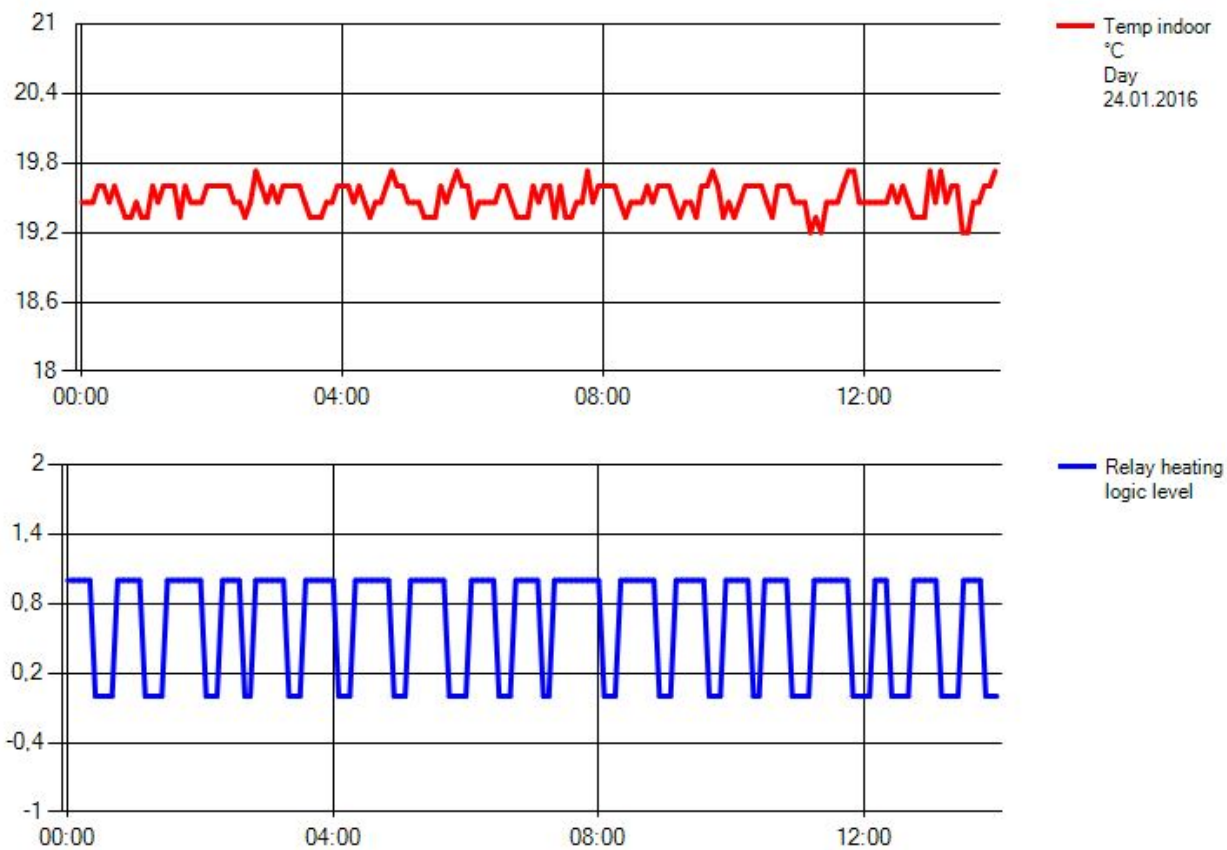
## 7.2. Prezentarea graficelor

Automatul programabil achizitioneaza din 5 in 5 minute valorile temperaturii interioare si starea logica a releului de iesire pe durata ultimelor 14 zile.

Selectati comanda [Get graph data](#). Aplicatia raspunde cu numarul zilelor in care exista achizitii de date.



Selectati in calendar ziua dorita si apasati butonul [Send Command](#).



Primul grafic arata evolutia foarte stransa a temperaturii in jurul valorii de 19,5 °C.

Al doilea grafic arata frecventa si duratele in care centrala debiteaza caldura.

Nota: Extinderea acestei aplicatii este [House\\_Heating.tig](#) care introduce suplimentar asigurarea securitatii locatiei, realizata cu senzori PIR.

### 7.3. Modificarea limitei de temperatura prin SMS

In baza configurarilor GSM, automatul programabil Mini PLC4404 GSM-GPRS recunoaste doar numerele GSM a max doi utilizatorilor. De la telefonul mobil pot fi lansate doua comenzi SMS:

- **temperatura = 21** Stabileste o noua valoare limita temperaturii. In partea dreapta a semnelui egal se specifica valoarea temperaturii limita intre limitele + 4 ~ 26 °C. Valorile pot fi transmise si sub forma zecimala ex: 22,34 sau 22.56
- **temperatura** Cere temperaturile masurate si limita existente in acel moment.

Transmiteti prin SMS comanda: **temperatura = 20** Pot fi folosite litere mari sau mici.

Mini PLC4404, receptioneaza, executa comanda SMS si transmite imediat utilizatorului un raport SMS cuprinzand noua valoare a limitei de temperatura si valoarea masurata a temperaturii. In acest fel exista certitudinea executiei comenzii SMS.

Comanda SMS



Raport SMS receptionat



Sintagma **temperatura** este definita in program:

```
#ifdef GSM
#define SMS_TEMPERATURE          "temperatura"
#endif
```

Executia comenzilor SMS este indeplinita de subrutina utilizator `UseSmsCommand ()` apelata in subrutina `ExecuteCustomerCommands ()`.

## 7.4. Bucla de reglaj automat a temperaturii

Declara marimea digitala folosita in aplicatie. Marimile digitale pastreaza attributele marimilor analogice neutilizate.

```
#define SECOND_DIGITAL_NAME          "Relay heating"

WORD Relay_Heating                   'conectat la Relay_1_Dig_Out, iesire dig. generalizata
```

Bucla de reglaj are la baza compararea valorii masurate cu valoarea limita si histereza de valoare mica, doar 0,3 °C. Procedura e executata in [TASK Analog\\_Data](#).

```
'Variabila personalizata Relay_Heating ia valoarea variabilei generalizate Relay_1_Dig_Out
Relay_Heating = Relay_1_Dig_Out
```

```
'Opreste incalzirea at cand valoarea limita programata de utilizator nu se incadreaza in plaja admisa
IF AnalogLimitValue(Host, 0) > MaxTemperature OR AnalogLimitValue(Host, 0) < FreeseLimit THEN
    RES_BIT Relay_Heating, 7          'reseteaza bitul de comanda
ELSE
```

```
    'Porneste incalzirea at cand temperatura masurata coboara sub limita histerezei
```

```
    IF AnalogLimitValue(Host, 0) - Measured_Value(Host, FIRST_ANALOG_CHANNEL) > TempHisteresis
        AND BIT(Relay_Heating, 7) = FALSE THEN
        SET_BIT Relay_Heating, 7      'seteaza bitul de comanda
    ENDIF
```

```
    'Opreste incalzirea at cand temperatura masurata urca peste limita histerezei
```

```
    IF Measured_Value(Host, FIRST_ANALOG_CHANNEL) - AnalogLimitValue(Host, 0) > TempHisteresis
        AND BIT(Relay_Heating, 7) = TRUE THEN
        RES_BIT Relay_Heating, 7      'reseteaza bitul de comanda
    ENDIF
ENDIF
```

```
'Executa comanda iesirii digitale Relay_Heating: starile ON, OFF
CALL Execute_Relay_Out(Relay_Heating, NO_EVENT)
```

## 7.5. Transmiterea alarmelor in situatii de avarie a centralei termice pe gaz

Centrala termica poate intra in regim de avarie:

- La scaderea presiunii apei in cazan e necesar interventia umana
- La intreruperea alimentarii cu gaz. La revenirea alimentarii cu gaz e necesar interventia umana

```
'Centrala termica e declarata in stare de avarie at cand Temperatura interioara coboara sub limita critica
```

```
'Declara ca Temperatura interioara foloseste nivele critice de alarma
```

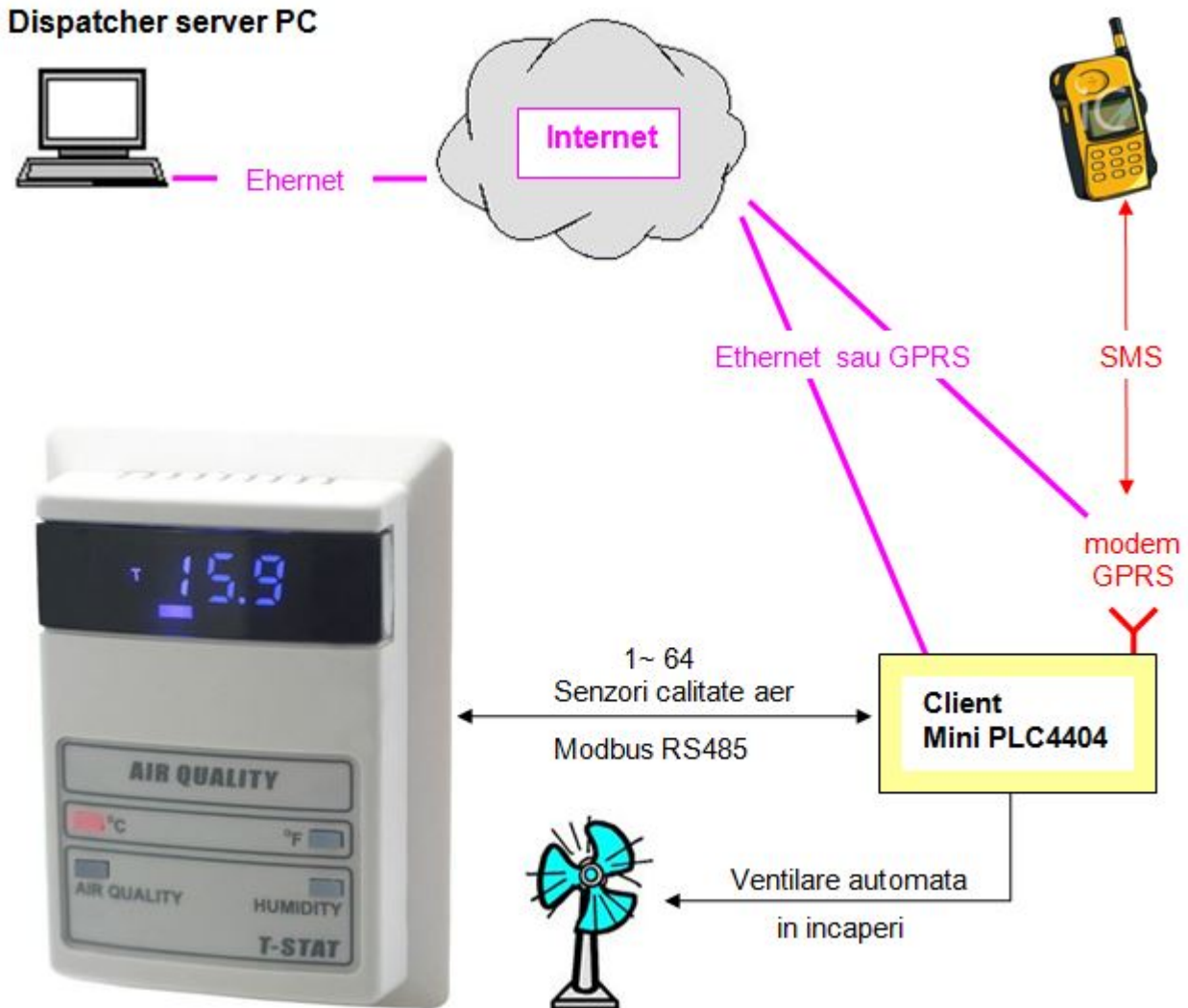
```
#define FIRST_MES_CRITIC_LIMIT
```

```
#define FIRST_MES_BELOW_CRITIC_LIMIT    "3,00"    ' 3 grade Celsius
```

```
'Revenirea in stare normala a centralei termice se produce at cand Temperatura interioara
'creste peste valoarea minima normala FIRST_ANALOG_ABOVE_CRITIC_LIMIT = 5°C
```

```
#define FIRST_MES_ABOVE_CRITIC_LIMIT    "5,00"    ' 5 grade Celsius
```

## 8. Aplicatie → Utilizarea senzorilor AQ-1 pentru calitatea aerului



Programul este [Air\\_Quality.tig](#) si este realizat special pentru [senzorii AQ-1](#).

Senzorii AQ-1 masoara calitatea aerului la poluanti CO<sub>2</sub>, fum si compusi organici volatili VOC. De asemenea masoara umiditatea si temperatura.

Gama valorilor masurate este:

- Temperatura -30 ~ +70 °C
- Umiditatea 0 ~ 100 %RH
- Calitatea aerului in valori absolute 0 ~ 1000, patru intervale:
  - Fine 0 ~ 200
  - Fair 200 ~ 400
  - Poor 400 ~ 600
  - Bad 600 ~ 1000

Declara provenienta marimilor analogice:

```
#define AIR_QUALITY_SENZOR_AQ_1 "AQ-1"
```

Senzorul calitate aer foloseste procedura Modbus RTU RS485 cuprinsa in [Sistem\\_Modbus\\_RS485.inc](#)

```
#define MODBUS_RS485 "Modbus"
```

'FOARTE IMPORTANT: adresele RS485 ale senzorilor trebuie setate individual din butoanele senzorului, conform documentatiei .pdf a senzorului.

'Atunci cand este folosit un singur senzor, adresa RS485 = 1, pentru al doilea senzor adresa RS485 = 2, 'samd.

[Convertorul RS232-RS485](#) este conectat la portul Ser1 al automatului programabil. Comunicatia se desfasoara la 19.200 baud.



Aplicatia poate folosi max 64 senzori distribuiti pe distanta max 1000m.

'Declara numarul senzorilor folositi in aplicatie

```
#define AQ_SENZORS_TOTAL_NUMBER 2
```

'Declara ca aplicatia opereaza cu limite ale marimilor analogice

```
#define USE_MEASURES_LIMITS
```

'Declara daca aplicatia asigura ventilarea automata a incaperilor atunci cand valoarea Calitatii aerului scade sub limita programata de utilizator

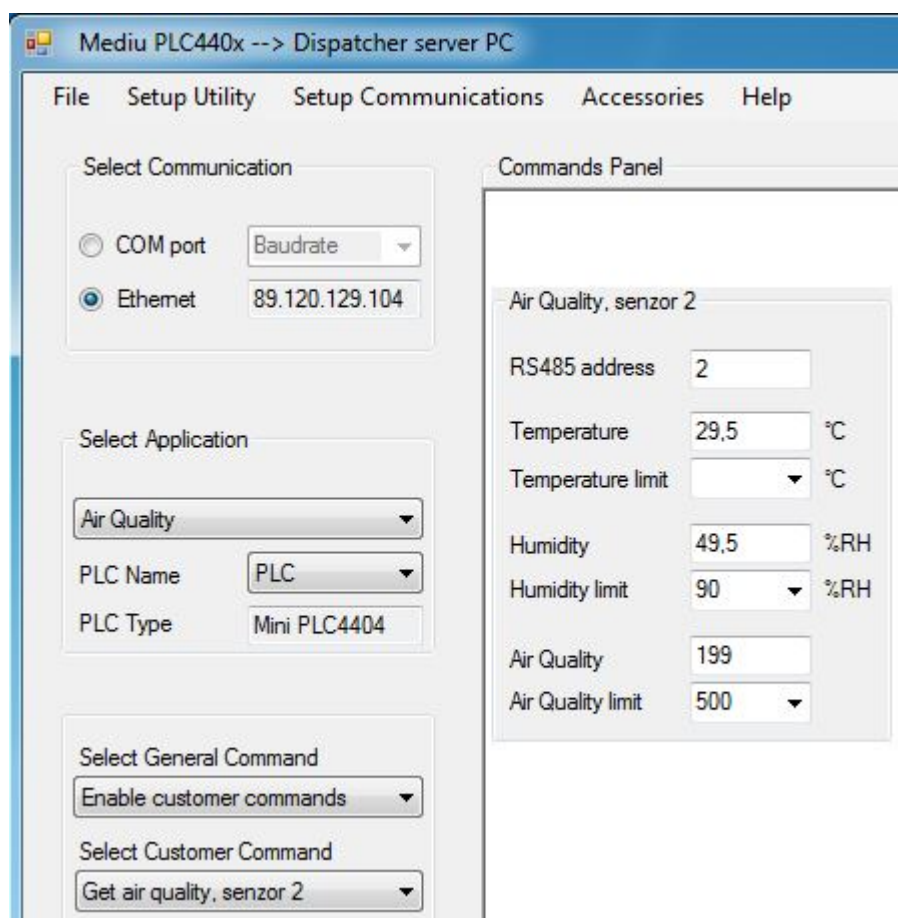
```
#define VENTILATION 'OPTIONAL'
```

## 8.1. Programarea valorilor limita in Internet de la statia Dispatcher server PC

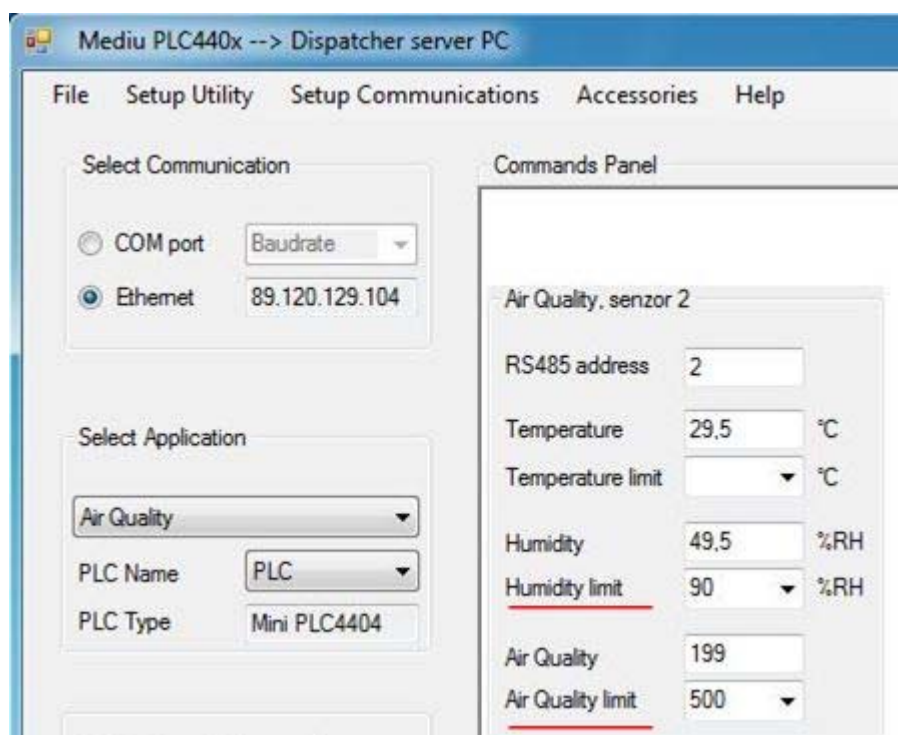
Pentru fiecare senzor cu adresa **n** in retea RS485, cutia [Select Customer Command](#) pune la dispozitie doua comenzi specifice:

- [Get air quality, senzor n](#)
- [Get graph data, senzor n](#)

**Selectati** comanda [Get air quality, senzor 2](#). Automatul programabil raspunde cu valorile masurate de senzorul cu adresa 2 in retea RS485:



In acest panou, utilizatorul programeaza valori limita ale marimilor masurate, individual pentru fiecare senzor, in functie de necesitati.



**Apasati** butonul [Send Command](#). Automatul programabil intra in posesia noilor valori de alarma programate de utilizator.

## 8.2. Configurarea aplicatiei

Aplicatia este configurabila pe baza optiunilor selectate in programul [Air\\_Quality.tig](#).

```
'Declara daca Temperatura foloseste limite programate de utilizator
'Declara mesajul de comanda in format text caruia i se atribuie un numar unic in gama 1...10
#define FIRST_MES_CMD_CHANGE_LIMIT "Utilizator modifica limita Temperatura la "
#define FIRST_MES_CMD_CHANGE_LIMIT_nr 1

#ifdef FIRST_ANALOG_CMD_CHANGE_LIMIT

'Declara limitele de functionare normala ale Temperaturii
#define FIRST_MES_MIN_VAL_LIMIT -30 °C 'valoare editabila
#define FIRST_MES_MAX_VAL_LIMIT 60 °C 'valoare ditabila
#define FIRST_MES_LIMIT_INCREMENT 5 °C 'valoare editabila

'Valorile limita ale Temperaturii sunt programate in Dispatcher server PC:
' - Selecteaza una din valorile disponibile.
' Pentru Temperatura pot fi selectate valori in gama - 30 ... 60°C, increment 5°C.
' - Editeaza manual orice valoare cu precizie de o zecimala.

'Declara scopul utilizarii limitelor programate de utilizator.
'Declara daca Temperatura foloseste evenimente sau alarme sau nu foloseste.

'Selecteaza una din optiuni sau niciuna:
#define FIRST_MES_EVENT_ALARM ANALOG_EVENT 'salveaza evenimentele in FLASH
#define FIRST_MES_EVENT_ALARM ANALOG_ALARM 'transmite mesaje de alarma prin
'Email si SMS

#ifdef FIRST_MES_EVENT_ALARM

'Declara valoarea histerezei
#define FIRST_MES_HYSTERESIS "5" '5°C 'valoare editabila

'Selecteaza una din optiuni:
#define FIRST_MES_ABOVE_BELOW ABOVE 'depasirea in sus a limitei
#define FIRST_MES_ABOVE_BELOW BELOW 'depasirea in jos a limitei

'Temperatura interioara foloseste mesaje text dedicate in caz de eveniment sau alarma.
'Fiecarui mesaj text ii este alocat un numar unic in gama 1...10.
'Defineste mesajele folosite in caz de eveniment sau alarma.

#define FIRST_MES_RISE_ABOVE_LIMIT "Temperatura creste peste "
#define FIRST_MES_RISE_ABOVE_LIMIT_nr 2

#define FIRST_MES_DROP_BELOW_LIMIT "Temperatura coboara sub "
#define FIRST_MES_DROP_BELOW_LIMIT_nr 3
#endif 'FIRST_MES_EVENT_ALARM
#endif 'FIRST_MES_CMD_CHANGE_LIMIT
```

## Selectarea optionala a ventilatiei

Aplicatia asigura ventilarea automata a doua incaperi deservite de primi doi senzori de Calitate a aerului.

Bucula de reglaj automat a valorii Calitate a aerului asigura ventilarea incaperii pana cand Valoarea Calitatii aerului coboara sub limita programata de utilizator.

```
#define VENTILATION          'OPTIONAL
```

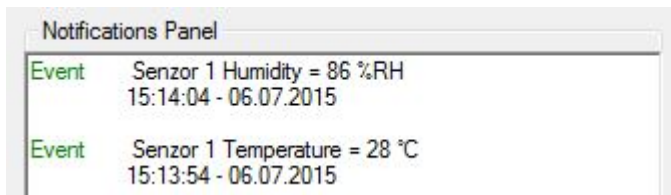
'Declara numarul senzorilor pentru care se asigura ventilatia in incaperi

```
#define VENTILATION_NR      1          'poate fi 1...2
```

### 8.3. Evenimente si mesaje de alarma

Automatul programabil salveaza depasirea valorilor limita programate de utilizator ca **evenimente** sau **alarme** conform configurarii aplicatiei.

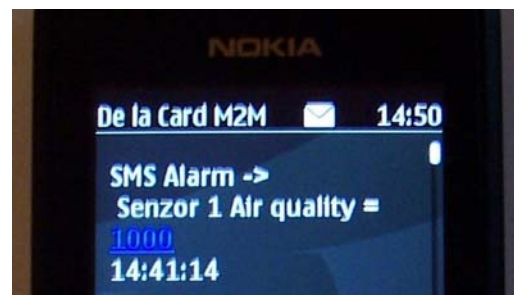
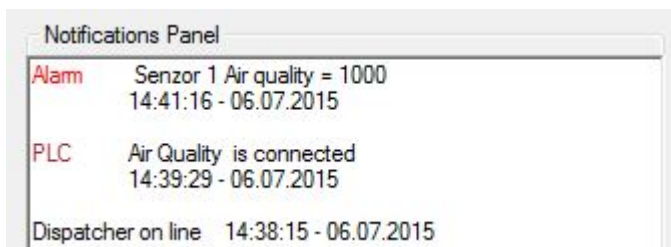
Evenimentele la depasirea limitelor sunt prezentate astfel:



Evenimentele sunt vizualizate cu ajutorul comenzii [Get last events](#).

Atunci cand sunt configurate alarme, automatul programabil transmite utilizatorului mesaje de alarma, pe urmatoarele cai:

- Notificare in Dispatcher server PC
- SMS
- Email



## 8.4. Transmiterea comenzilor SMS

Pentru obtinerea prin SMS a valorilor masurate de senzorul de calitate aer cu adresa `n` in retea RS485, se transmite comanda SMS `AQn`. `AQ` poate fi scris cu litere mari sau mici.

Automatul programabil raspunde cu urmatorul SMS.



'Receptionarea comenzilor SMS este un caz particular.

'Avantajul comenzilor receptionate prin SMS este ca pot fi executate mai multe comenzi individuale cuprinse 'in acelasi SMS, in limita a max 160 caractere.

'Comenzile SMS pot fi scrise cu litere mari sau mici, fara diacriticele limbii romane.

### #ifdef GSM

'Pentru modificarea limitelor, utilizatorul transmite comenzile SMS:

- ' 1. Comanda pentru modificarea limitei Temperaturii: ex "temperatura = 21"  
' sau "temperatura = 21,3"
- ' 2. Comanda pentru modificarea limitei Umiditatii: ex "umiditate = 70"  
' sau "umiditate = 70,9"
- ' 3. Comanda pentru modificarea limitei Calitatii aerului: ex "aq = 500"

'Pentru receptionarea valorilor masurate, utilizatorul transmite comanda SMS "aq"

'declara sintagmele folosite in comenzile SMS

```
#define SMS_TEMPERATURA "temperatura"  
#define SMS_UMIDITATE "umiditate"  
#define SMS_CALITATE_AER "aq"
```

#endif 'GSM

## 8.5. Prezentarea graficelor

Resursele disponibile SRAM ale modului computer Tiny Tiger sunt folosite pentru salvarea din cinci in cinci minute a valorilor furnizate de senzori.

'Declara ca aplicatia achizitioneaza date in SRAM in scopul prezentarii graficelor

```
#define USE_MEASURES_GRAPHs "Use analog graphs"  
#define SAVE_DATA_IN_SRAM "SRAM"
```

'Declara numarul senzorilor pentru care se fac achizitii de date

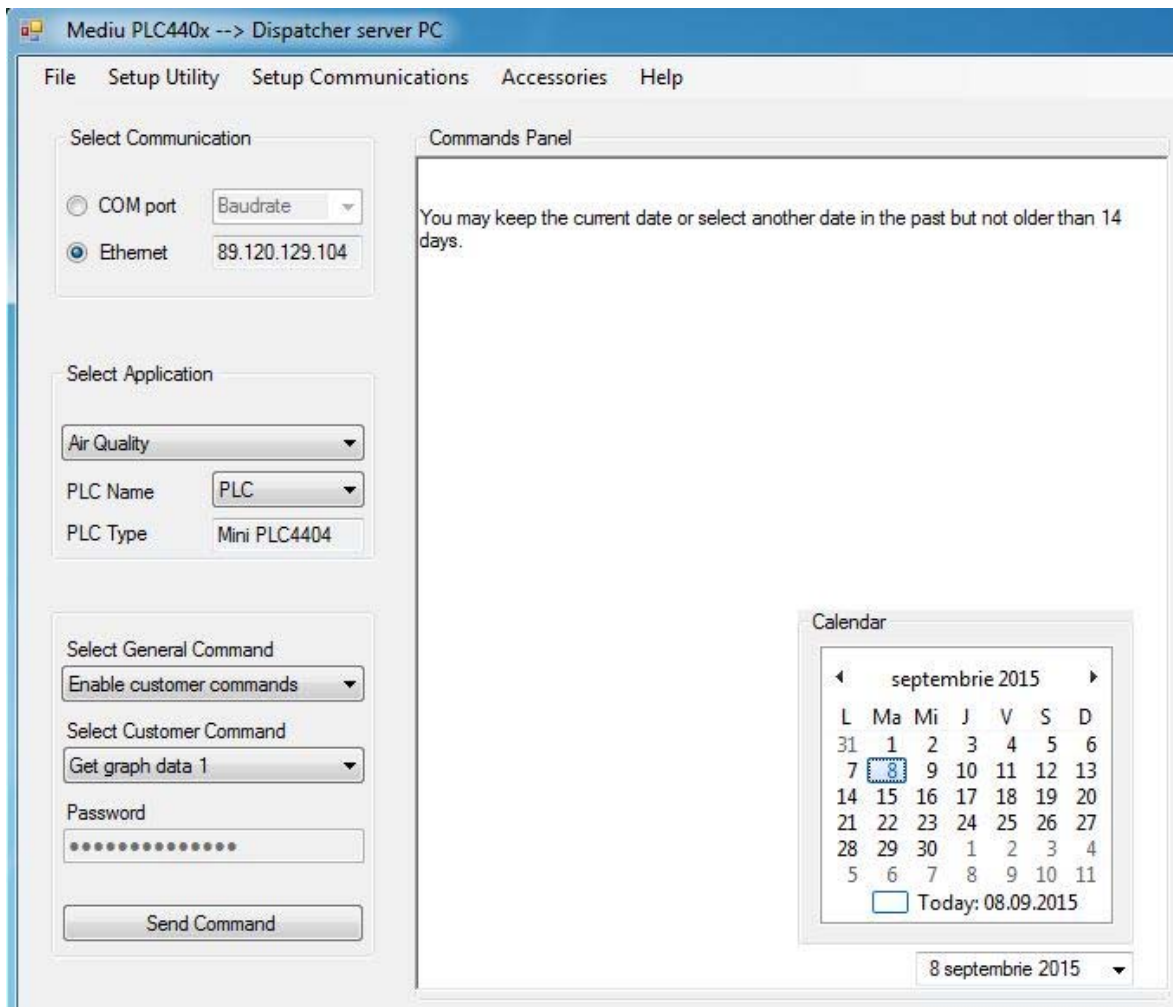
```
#define AQ_SENZORS_BUILD_GRAPHs_NUMBER 1
```

Nota: `AQ_SENZORS_BUILD_GRAPHs_NUMBER` <= `AQ_SENZORS_TOTAL_NUMBER`

Resursele disponibile SRAM ale automatului programabil se impart egal intre acesti senzori:

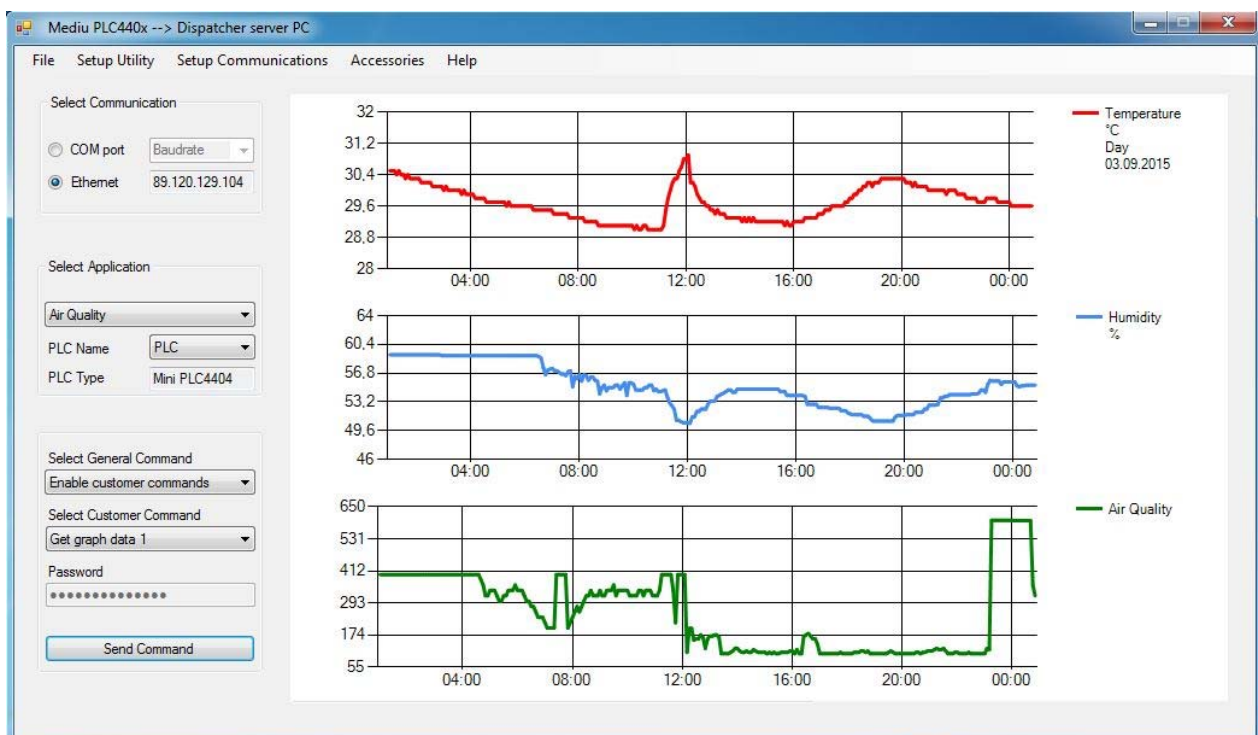
- Pentru un singur senzor, resursele SRAM asigura achizitia datelor din cinci in cinci minute pe durata ultimelor 15 zile
- Pentru doi senzori, achizitia datelor se intinde pe 7 zile samd

**Selectati** comanda `Get graph data, senzor 1`.



Selectati in calendar ziua curenta sau alta zi in urma cu maxim 14 zile.

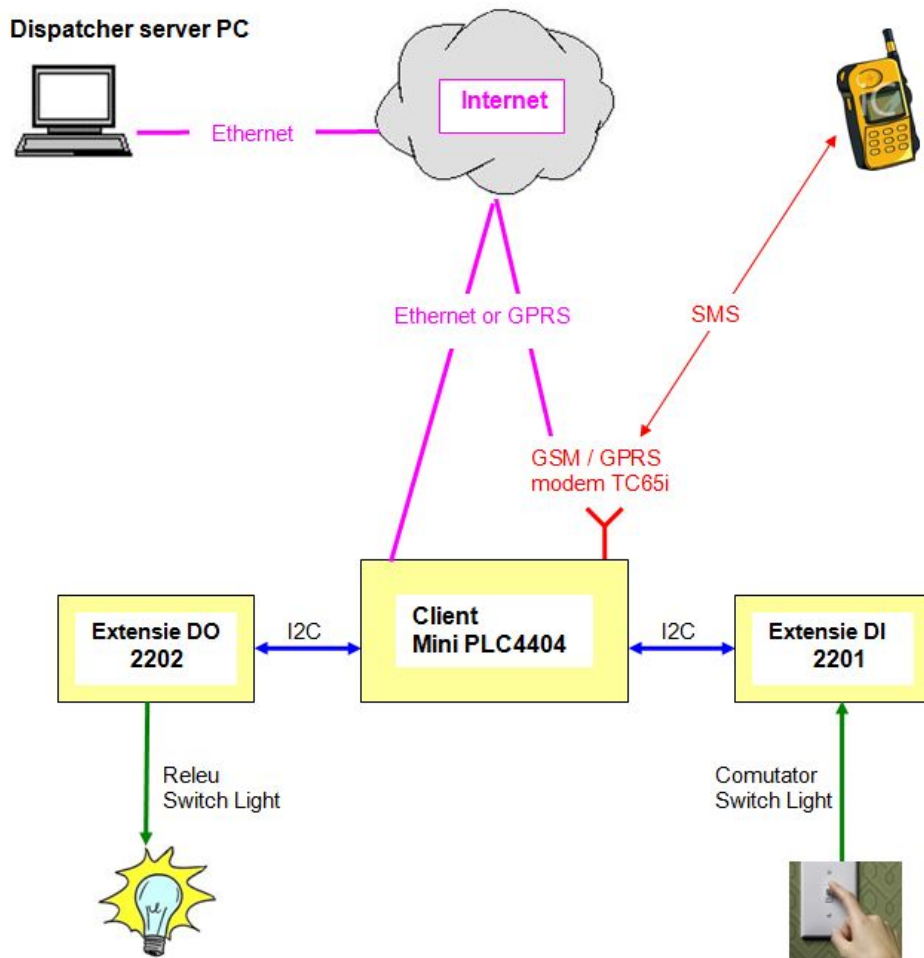
**Apasati** butonul **Send Command**. Ziua selectata este transmisa automatului programabil care livreaza datele achizitionate prezentate grafic.



## 9. Utilizarea pinilor configurabili

### 9.1. Aplicatie → Utilizarea modulelor de extensie I2C

Programul [Mini\\_plc\\_08.tig](#) prezinta implementarea comunicatiei I2C cu module [2201](#), [2202](#) de extensie a intrarilor si iesirilor digitale.



Automatul programabil trebuie dotat cu conectorul I2C conform [pdf Mini PLC4404](#). Strapul J3 conform capitol [4 pini configurabili](#) furnizeaza alimentare +5V modulelor de extensie.

**Sectiunea 1** cuprinde:

```
'declara procedura sistem I2C in baza careia sunt utilizate module de extensie
#define I2C
```

Configurarea pinilor se face in subrutina [Host\\_PLC\\_Device\\_Setup\(\)](#) din [System\\_In\\_Out.inc](#):

```
#ifndef I2C
    I2C_SETUP(6, 1, 6, 2, 0, 1)
    'maximum speed
    'sets L62 as I2C DIN
    'sets L60 as I2C DOUT
    'sets L61 as I2C CLK OUT
#endif 'I2C
```

Sursa de lumina urmareste starile logice ale comutatorului bipozitional [Switch Light](#).

Comenzile de aprindere, stingere a luminii pot fi lansate si prin Internet sau SMS, comanda cea mai recenta are prioritate indiferent de starea comutatorului local [Switch Ligh](#).

Comenzile pot fi transmise prin internet sau SMS:



## 9.2. Aplicatie → Comunicatie seriala in retea RS485



[Convertorul RS232-RS485](#) asigura comunicatia seriala intre automatul programabil Master Mini PLC4404 si Slaves RTU cu ajutorul protocolului [MODBUS](#):

- Max 1000m, atunci cand este folosita reseaua RS485 cablata
- 30 ~ 100m, atunci cand sunt folosite dispozitive Bluetooth RS485

Slavii RTU sunt de regula senzori inteligenti: fum, umiditate, temperatura, presiune, viteza, concentratie chimica etc.

Sunt realizate aplicatiile:

- [Monitorizarea si reglarea calitatii aerului in primarii, gari, spatii publice](#)
- [Grafice in ultimii trei ani cu memorare date in micro SD card](#)

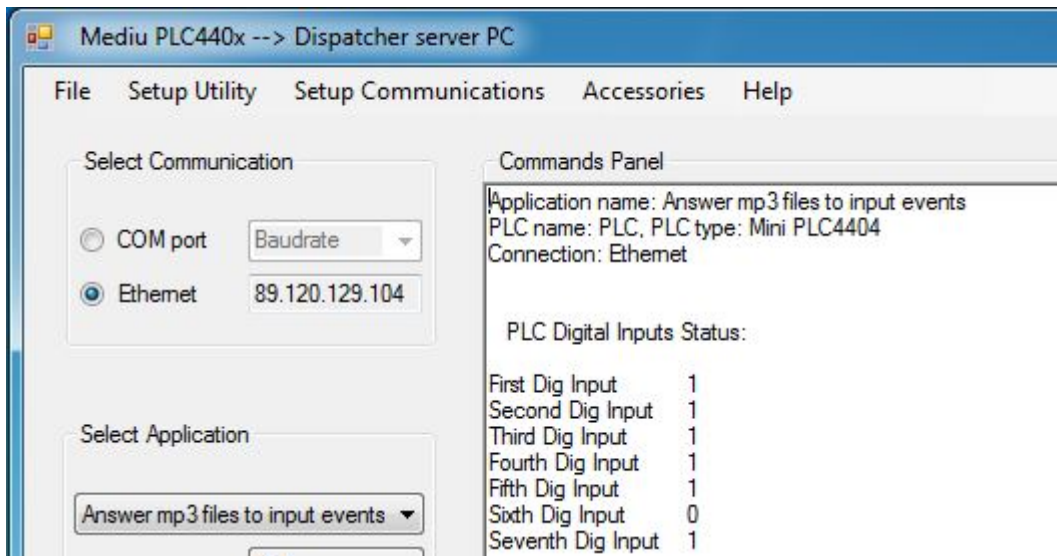
### 9.3. Aplicatie → Extensia intrarilor digitale

O aplicatie realizata este [Raspuns interactiv cu fisiere mp3 la evenimente](#).

Procedura sistem este declarata in **Sectiunea 1**:

'suplimenteaza la 7 numarul intrarilor digitale ale automatului programabil Mini PLC4404  
'cu ajutorul celor 4 pini configurabili care devin intrari digitale  
`#define USE_4_CONFIGURABLE_PINS_AS_DIG_INPUTS`

Raspunsul la comanda [Get PLC digital status](#) prezinta starea logica a celor 7 intrari digitale:



## 10. Comenzi simple si comenzi interactive

Sunt doua categorii de comenzi transmise de la Dispatcher server PC:

- Comenzile simple sunt facute intr-o singura etapa, la **apasarea** butonului [Send Command](#):
  - Modificarea starii logice a iesirilor digitale
  - Receptionarea unui raspuns elaborat, ex. coordonatele GPS
- Comenzile interactive se desfasoara in doua etape:
  - La **selectarea** comenzii automatul programabil transmite valorile pe care le cunoaste in panouri dedicate aplicatiei, apartinand Dispatcher server PC
  - La **apasarea** butonului [Send Command](#) automatul programabil intra in posesia noilor valori programate de utilizator in panourile dedicate. Exemple sunt toate aplicatiile cu marimi analogice, vezi limite si grafice.

## 11. Dezvoltarea aplicatiilor

In exemplele prezentate, utilizatorul poate opera modificari.

Seturi intregi de linii cod ale unui program pot fi copiate in alt program pentru indeplinirea propriilor scopuri.

Se verifica [Start\Compile](#) urmat de [Start\Run](#).

Utilizatorul poate elabora el insusi taskurile si subrutinele necesare.

Desigur, este nevoie de determinare, atentie si rabdare.

### 11.1. Depanarea aplicatiilor cu Terminalul Windows

In cursul realizarii aplicatiilor cu automate programabile sunt esentiale informatii text privind desfasurarea programului in diverse etape ale sale.

In acest scop este utilizat temporar portul Ser1 cu ajutorul optiunilor [REPORT\\_ENABLED](#) in fisierele sistem [System\\_Common\\_Internet.inc](#), [System\\_Gsm.inc](#) si [System\\_Keyboard.inc](#).

Terminalul Windows se conecteaza la portul Ser1.

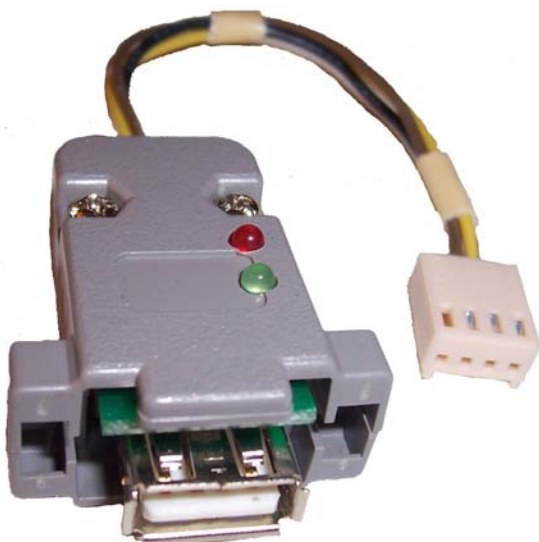
In laborator este foarte util sa folositi simultan trei porturi seriale ale PC-ului, la rata de baud 38400:

- Primul port alocat mediului de dezvoltare [Tiger BASIC](#)
- Al doilea port alocat interfeței grafice [Dispatcher server PC](#)
- Al treilea port alocat [Terminal Windows](#) pentru depanarea aplicatiilor

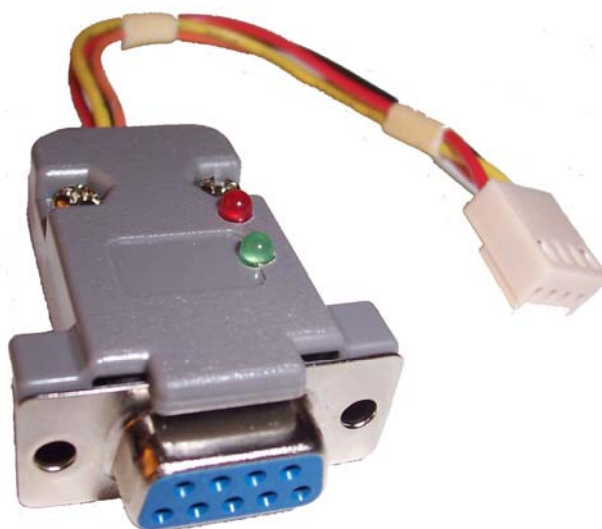
In acest fel, cele trei aplicatii Windows de mai sus pot fi deschise si utilizate simultan.

**Nota:** Automatele programabile cu porturi USB asigura accesul portului Ser1 la nivel 0~5V prin intermediul conectorului Con17. Citirea traficului se face cu unul din [dispozitivele](#):

Citeste trafic USB



Citeste trafic RS232



## 12. Anexe

Acest capitol e destinat utilizatorilor care doresc sa patrunda mai adanc in programarea automatelor programabile Mini PLC4404.

Programele utilizator se bazeaza pe librariile sistem cu extensie [.inc](#).

Taskurile si subrutinele sistem rezolva 95% din sarcini astfel incat programul utilizator sa fie usor de realizat, avand cat mai putine linii de cod.

Codul utilizator apeleaza taskuri si subrutine sistem puternice si configurabile. Configurarea campurilor unor subrutine fundamentale e prezentata in capitolele anterioare si in programele [.tig](#) accesibile pe site.

Desigur, nu am putut intui scopul fiecarui utilizator. Librariile sistem au fost dezvoltate in functie de probleme concrete cu care noi insine ne-am confruntat pana in acest moment.

Aplicatiile si librariile sistem sunt dezvoltate permanent si modificate fara preaviz. Din acest motiv e necesar sa descarcati cele mai recente fisiere.

Cu siguranta librariile sistem nu vor satisface utilizatori care abordeaza creativ noi proceduri si domenii de utilizare, nebanuite in acest moment.

Codul sursa Tiger BASIC pentru Mini PLC4404 este **open source**. Oricine il poate utiliza si opera modificari.

Atunci cand librariile sistem nu satisfac cerintele, utilizatorul va scrie codul necesar direct in programul utilizator, ex [clopotelul\\_scolii.tig](#) sau [street\\_light.tig](#).

[clopotelul\\_scolii.tig](#) si [street\\_light.tig](#) sunt atat de particulare si lipsite de sansa generalizarii, astfel incat nu are sens definirea unor noi librarii sistem.

In aceste cazuri codul utilizator poate ajunge la sute linii de cod in taskuri si subrutine cuprinzand functii elementare Tiger BASIC.

Funcitiile elementare Tiger BASIC sunt prezentate in [Help / Tiger BASIC IDE](#).

Informatii ample sunt prezentate in bibliografie.

**Nota:** Acest capitol va fi dezvoltat in viitor, in acord cu cerintele utilizatorilor.

## Bibliografie Tiger BASIC ~ Wilke-Technology GmbH

1. Modul computer [Econo Tiger](#)
2. Manual [Programare Tiger BASIC](#)
3. Noi functii introduse in [versiunea 5.2](#)
4. Noi functii si modernizari introduse [post versiunea 5.2](#)
5. Manual [Device Drivers](#)
6. Modul [ETHERNET Adapter EM01](#)
7. [Ghid de programare Tiger BASIC Sockets](#) pentru modulul ETHERNET Adapter EM01
8. 100 de aplicatii realizate de [Gunther Zielosko](#)