Modeling and Automation of Industrial Processes

MSc in Electrical and Computer Engineering
Scientific Area of Systems, Decision, and Control

2022 / 2023

Group: ____
___ - _____
___ - _____
___ - _____
___ - _____

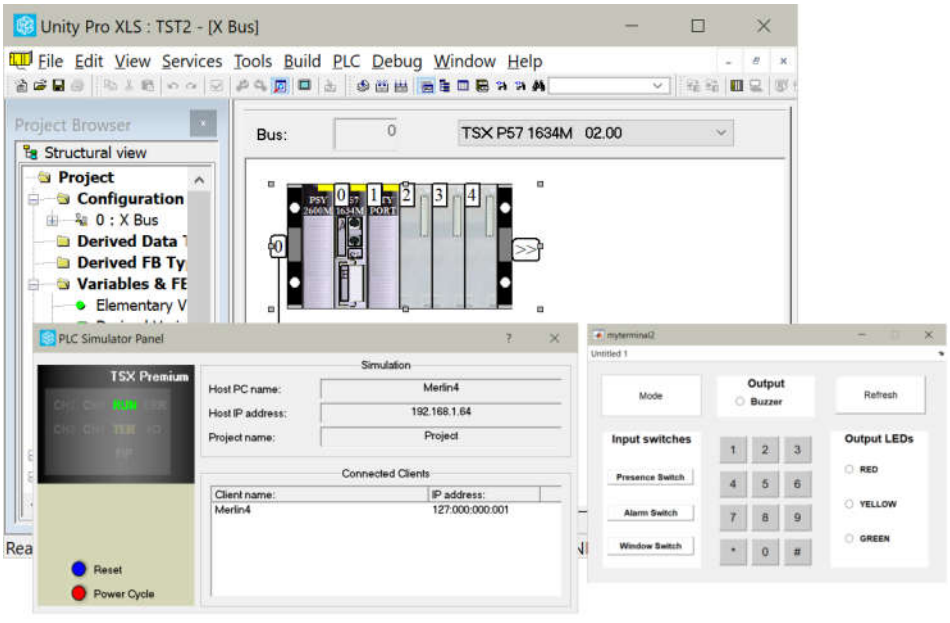
2nd Lab. - Alarm System for Intrusion Detection

This work aims the implementation of an intrusion detection alarm system, in a restricted space as a single room retail store. The intrusion will be detected resorting to an infrared sensor, installed in such a way that it points towards the main entrance of the space to be protected. An open/close switch is also installed on a window of the aforementioned space.

The automation system that constitutes the alarm is to be implemented in the Schneider **PLCs available in the laboratory**, model Premium TSX P57 1634M or TSX P57 2634M, see Fig1(a) or (b). In addition, there is available **simulated-hardware**, where digital inputs and outputs are PLC memory locations, which are read and written by the fieldbus Modbus, see Fig1(c).

(a) Setup with one input and one output modules

(b) Setup based on a single input and output module



(c) Simulated-hardware setup. The input and output are mapped on the PLC memory.

Fig.A1: PLC and terminal setups, real (a, b) and simulated (c). The virtual terminal, shown in (c), formed by switches and LEDs, is simulated in Matlab, and connects to the PLC simulator based on Modbus.

Part A - Alarm based on Switches

The objective of this first part of the work is to implement the basic functioning of the alarm using the switches, LEDs, and buzzer. Are expected to be implemented just the general characteristics of the alarm: (mode 1) **OFF**, (mode 2) **Presence Detection** and (mode 3) **Active**. See the description of these general characteristics in Annex A. (The advanced characteristics will be considered in the other parts of the work.)

Implementation details

To implement the general characteristics of the alarm in a clear manner, it is important to separate into one sub-routine the mode 2, Presence Detector, and to another sub-routine the mode 3, Active. More sub-routines can be added in case they help making the code clearer. In this part of the work **the infrared sensor is replaced by the two positions switch**.

To help the readability of the proposed solution, the identifiers used in the PLC program must have readable names. For example, instead of using "%I0.2.0" it is more readable to use "I_switch1_pos1". The following naming convention is suggested: **I_xyz** for inputs, **O_xyz** for outputs, **M_xyz** for memory variables, **TM_xyz** for timers, **SR_xyz** for sub-routines, etc, where xyz denotes meaningful names.

To help develop and test the PLC program, a simulated terminal is available as a Matlab graphical window, see bottom-right of Fig.A1(c). One may use both physical and virtual inputs / outputs, by having some additional initial and end sections within the project. See an example of a project combining physical and virtual IO in the webpage:
http://users.isr.ist.utl.pt/~jag/course_utils/plc_log/plc_log_str.htm

In the end of the project, each group delivers a ZIP file containing (i) the report as a PDF file, and (ii) the PLC project, more precisely Unity Pro files with extensions STA and STU.

Report questions

Note: this guide is distributed in an editable form so that it can be filled and then submitted online.

A1. *(Real and simulated hardware diagram)* Draw a hardware diagram corresponding to the PLC and terminal hardware (Fig. A1a) combined with the simulated terminal (Fig. A1c). Figure A2, to be edited (completed) by your group, shows hardware components, **switches**, **LEDs** and **buzzer**, that are to connect to the **PLC**. Do not forget to include the **power suppliers**. The virtual terminal,

myterminal5, allows writing to and reading from the **memory** of the PLC, runs in a PC and in that way is also connected to the PLC. Indicate in the drawing the physical addresses for the real inputs and outputs.

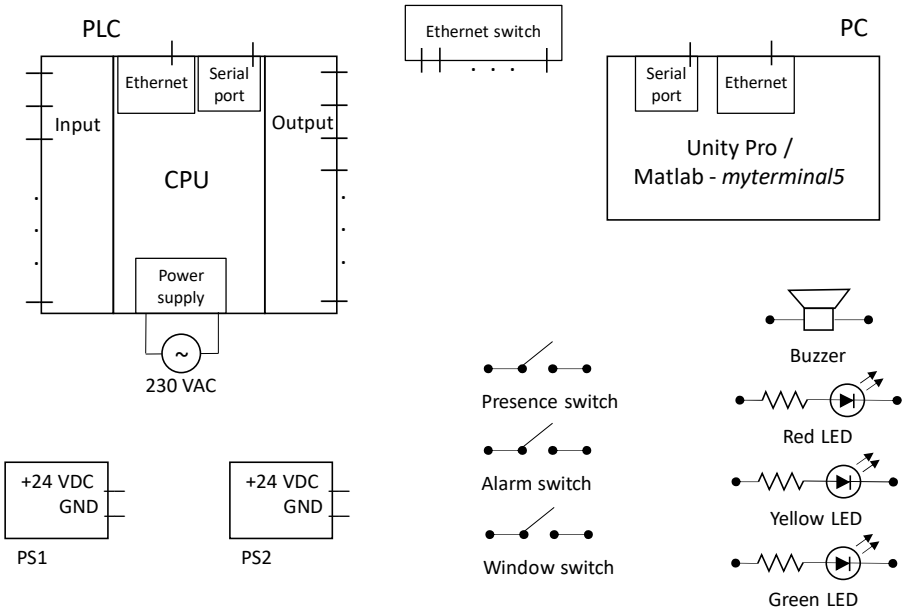


Fig.A2: Components to create a real and simulated hardware diagram. Connect these components to make a complete diagram (right-click the figure and "object > open" to complete the connections).

A2. *(Real and Simulated-hardware identification)* Identify the inputs (switches) and outputs (LEDs and buzzer) of the simulated intrusion-detection alarm-console and fill the next table.

Input	**PLC Identifier**	
Choose the name to use	***Input address %I***	***Memory address %M***

Output	**PLC Identifier**	
Choose the name to use	***Output address %Q***	***Memory address %M***

(Insert as many lines as needed to the tables)

A3. *(Hardware test program)* Design and test a **Structured Text** program that verifies the proper functioning of the output devices available in the alarm console. More precisely, the program must beep the buzzer 0.5 seconds and then turn ON the three lamps (LEDs), one lamp after the other, 0.5 seconds each. Create **one project with one MAST section** to test this sub-routine.

A4. *(Multiple accesses to the buzzer)* Create a **Structured Text** sub-routine where the buzzer can be turned ON by one of two alternative memory variables (EBOOL), and can be unconditionally turned OFF by another memory variable (EBOOL). Create a fourth way of actuating the buzzer, also commanded by a memory variable (EBOOL), where the buzzer sounds with a periodic signal, 1 second ON and 2 seconds OFF. If the buzzer is **commanded to be simultaneously ON and alternating ON/OFF** then **select alternating ON/OFF**. Demonstrate this sub-routine by using switches or pushbuttons on the virtual terminal.

A5. *(Demonstration on the hardware)* Design one or more **Ladder** or **Structured Text** sections to solve the aforementioned automation problem. Create one *main* section in **Structured Text** where the memory variable M_MODE is used to indicate the current working mode of the complete system and actions are taken according to it.

A6. *(Experiment time plot)* Use data logging to document an experiment, i.e. make a time plot of binary variables, considering two usages of the alarm: (i) the alarm is activated, stays activated for a while, and then is deactivated without any intrusion, (ii) the alarm is activated, then, upon intrusion, activates the buzzer, and finally the alarm is disarmed. Start by downloading the demonstration file `data_log.zip` and reading more instructions from the website:
http://users.isr.ist.utl.pt/~jag/course_utils/plc_log/plc_data_log.htm
Notes: The referred zip file has one PLC project and Matlab scripts demonstrating PLC data logging.

Part B - Read Keyboard

The second part of this laboratory assignment consists of the implementation of an automated solution to the identification of a key pressed, on the available keyboard with 12 keys. The keyboard is central to the user to interact with the intrusion detection alarm system under design. The solution foreseen is the design of one or more **Structured Text** subroutine sections, optionally in Ladder, to identify the key pressed.

The keyboards that are used in the laboratory have 12 keys arranged in a 4x3 matrix, see Fig.B1. The terminology to be used in the laboratory is the following: the three columns are named by the digits `1,2,3` and the four lines by the letters `a,b,c,d`.

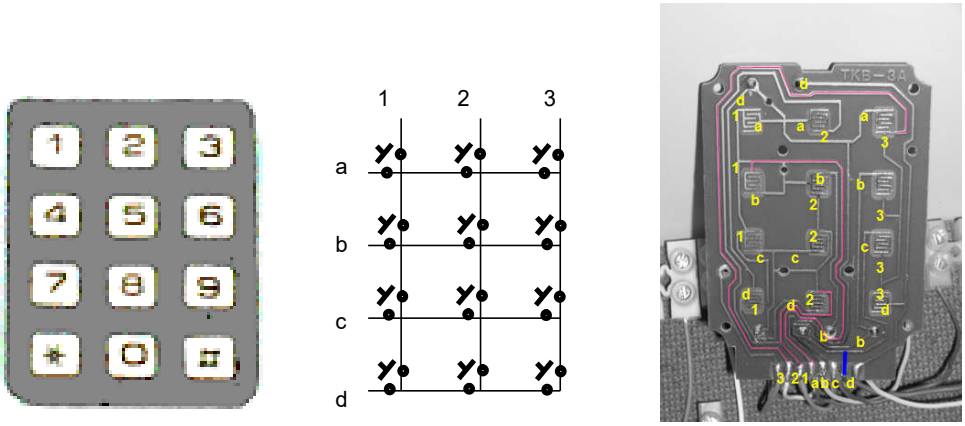


Figure B1: An image and an electrical diagram of the 4x3 keyboard (keypad).

A simulated keyboard terminal, which can interact with the PLC, is shown in Fig.A1(c). It is a simulated keyboard created with Matlab. The communication with the PLC is done through Modbus. As in the real keyboard, the keyboard columns must be energized before being possible to observe energized keyboard lines upon pressing the buttons.

Implementation guidelines

A suggested name for the main subroutine identifying the key pressed is `SR_get_key`. If no key is pressed the `SR_get_key` subroutine must store -1 in a memory as the result. In case a digit, '0', '1', ... , or '9', is pressed, the subroutine stores the corresponding value in the memory. In case the key '*' or the key '#' is pressed, the subroutine stores in the memory 10 or 11, respectively. A 0.05sec beep signal (buzzer sound) shall be provided as soon as a key is detected.

The `SR_get_key` subroutine is desired to (i) reject multiple keys in the same column pressed at the same time, (ii) recognize key release, i.e. "key up", as soon as possible and then allow immediately to read another key, and (iii) not allowing to input by mistake two or more times the same key, in other words having a repeat delay preventing immediate input of multiple times the same key. Note: a "key up" flag is useful for other parts of the program to recognize that the `SR_get_key` subroutine has a key that was read and is prepared to be reset and therefore input another key (notice the similarity of the `SR_get_key` subroutine with the timers/counters' reset and done bits).

A subroutine named `SR_get_multiple_keys` must be built to hold up to 10 keys. This subroutine checks the output of `SR_get_key` to get keys one by one. The `SR_get_multiple_keys` subroutine must have a global variable allowing to reset (flush) the buffer. The recommended implementation language for this subroutine is **Structured Text**. Note: in order to use array indexing one has to enable the option "menu Tools -> Project Settings -> variables -> Directly represent array variables".

Report questions

Note: this guide is distributed in an editable form so that it can be filled and then submitted online.

B1. *(Keyboard IO identification)* Identify the PLC inputs and outputs, from/to the **keyboard**, to be used on the intrusion detection alarm system.

PLC Input (input address, %I)	PLC Identifier (memory address, %M)

PLC Output (output address, %Q)	PLC Identifier (memory address, %M)

(Insert as many lines as needed in the tables)

B2. Describe the reading/identifying strategy to be implemented to solve the keyboard reading problem. In other words, describe the implementations of the subroutines `SR_get_key` and `SR_get_multiple_keys`.

B3. Use the **logging method** (already used in part A) to show changes in the inputs when a person enters the code `9963` using the keyboard. Show **time plots** of the relevant input signals.

Part C - Integration

In this part of the work the 12-keys keyboard available in the console is used as the primary interaction device with the alarm system under design. Sequences of keys must be validated resorting to the subroutines implemented in the previous part of this work.

Implementation guidelines

The basic presence detection and active alarm system already developed is supposed to be running continuously. Note, however, that the function of selecting the operating mode (OFF / presence detection / active alarm) cannot be based just on switches. Selecting the operating mode must be moved to an upper integration level. Suggestion: Use a global variable M_ALARM_MODE to indicate and change the alarm mode.

Keyboard reading is also a function assumed to be running continuously. The upper integration level can validate codes saved in a buffer. Keyboard input is cleared if a timeout occurs, or any other condition recommends flushing keys already read.

Use the keyboard to replace the switches for changing the mode of the system. More in detail, let `#1*` set the off mode, `#2*` set the presence detector and `#3*` set the active mode. Suggestion: Consider the human and alarm interaction based on a prompt for person (alarm owner) commands, while the alarm is running in parallel, so the alarm goes off when the prompt is not used to stop it.

As noted in advanced characteristics of the alarm (annex A), entering, and leaving the ACTIVE mode implies entering a four digits code. To facilitate the interaction, any return to the OFF mode must be marked with 3-short-beeps done with the buzzer. A suggested timing is 0.02sec for each beep and a pause of 0.1sec in between.

Report questions

Note: this guide is distributed in an editable form so that it can be filled and then submitted online.

C1. Represent as a finite state machine the upper integration level of the complete system. Clearly identify the OFF mode, the presence detection mode, the active alarm mode, and the other states necessary for the complete system.

C2. Design a PLC section (program) representing the **upper integration level**. Describe the **methodology used to validate sequences of keys** of a pre-specified length. Discuss how to proceed if the input fails, e.g. someone enters a wrong code.

C3. Upload the program to the PLC, execute it and comment on the results. Use logging-methodologies (as in part A) to **create plots that illustrate the functioning** of the system.

C4. Indicate the **minimum time pressing keys** till successfully deactivating the active alarm mode. Indicate also the maximum time, if applicable.

C5. (Facultative) Use strings logging to report the following experiment: (1) the alarm is activated using a four keys code, (2) the alarm is deactivated using the same keys code. Show a print screen of the window opened when terminal option "Logged strings show" is selected. See sample code for string writing and logging in:
http://users.isr.ist.utl.pt/~jag/course_utils/plc_log/plc_log_str.htm

C6. Discuss possible future improvements to the alarm system. Given the list of MSc dissertations:
http://users.isr.ist.utl.pt/~jag/msc/thesis/plc_works.htm
indicate parts of these MSc works you would like to see included in this work.

Annex A - Detailed Functional Specifications of the Alarm

General Characteristics of the Alarm:

The Alarm has three main modes of operation, OFF, Presence Detector and Active. The three modes are selected by a three positions switch. The three modes operate as detailed next:

(Mode 1) OFF – this mode deactivates the alarm completely.

(Mode 2) PRESENCE DETECTOR – the infrared sensor is used to detect the movement on the room/space, that be signalized resorting both to a lamp and to the buzzer on the panel. The lamp should be on for 5 seconds, upon the detection of each person, and an acoustic signal with the duration of 1 second should be emitted.

(Mode 3) ACTIVE – in this mode the alarm is to be used.

Detailed specifications for mode 3, **ACTIVE**, are the following:

a) When requested for activation, a period of 30 seconds of inactivity is set to allow the user to abandon the space, and afterwards remains permanently activated.

b) Upon intrusion detection, by the infrared sensor or the window switch, the alarm evolves to the warning phase.

c) The alarm lights a warning on the panel and after 5 seconds the buzzer must be activated. The warning must be a periodic signal with 1 second on and 2 seconds off.

d) The alarm can be deactivated pressing the # key on the command panel.

Advanced Characteristics of the Alarm:

An advanced alternative for the alarm activation/deactivation consists on the use of a code previously set by the human owner (e.g. 9887). To implement the activation function, the following procedure must be implemented:

a) switch the alarm mode to ACTIVE.

b) introduce the activation code (e.g. 9887).

c) press #, and wait for 30 seconds to allow the user to abandon the space.

d) start the intrusion detection function, i.e. the alarm is fully operational.

To deactivate the alarm, upon intrusion detection or to allow the use of the space, the following instructions must be accomplished:

a) Introduce the secret code (the same as the activation one, e.g. 9887).

b) Press #

c) Change the alarm mode to a mode other the ACTIVE.

Special Characteristics of the Alarm:

A safer mode of operation for the intrusion detection alarm is to allow the user to change the activation/deactivation code. The code 1234 is initially used, as a factory preset. To change the code, the following operations must be done:

a) Press *, followed by the pre-programmed code.

b) Introduced the new code to be used, finished by *

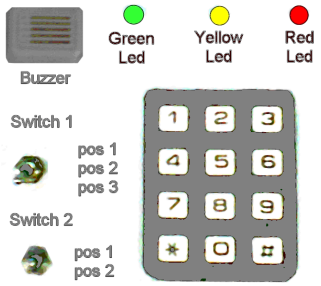
In the case where a mistake occurs, press the code **** to reset the code to the factory default.

Available Material:

In the laboratory there are six different working places, all with similar PLCs but different consoles. All workplaces have a PLC Schneider model P57. All of them have a power supply with 24V and/or 12V and a desktop PC, with the Unity Pro v6 development software and the PLC manuals, in PDF format.

In each workplace there will be also an alarm console with the following components:

1 three positions switch
1 two positions switch
3 LEDs
1 keyboard (4x3 buttons)
1 buzzer (12V)



The solution for this automation problem must be based on the languages described on the IEC-61131-3 standard, i.e. ladder diagrams, instruction list and structured text.