|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Industrial Processes Automation***  *MSc in Electrical and Computer Engineering*  *Scientific Area of Systems, Decision, and Control*  *Winter Semester 2020/2021* |  | *Group: \_\_\_\_*  *\_\_-\_\_\_\_\_*  *\_\_-\_\_\_\_\_*  *\_\_-\_\_\_\_\_*  *\_\_-\_\_\_\_\_* |

***1st Lab. - Alarm System for Intrusion Detection [[1]](#footnote-1)***

***Part B - Read Keyboard***

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 space. The automation system that constitutes the alarm is to be implemented in the Schneider PLCs available on the laboratory, model Premium TSX P57 1634M or TSX P57 2634M. *Important: while in effect the current suspension of presence classes, set to protect public health, this laboratory assignment will be based on* ***simulated-hardware****, where* ***digital inputs and outputs are PLC memory locations****, which are read and written by the fieldbus Modbus.*

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.

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.

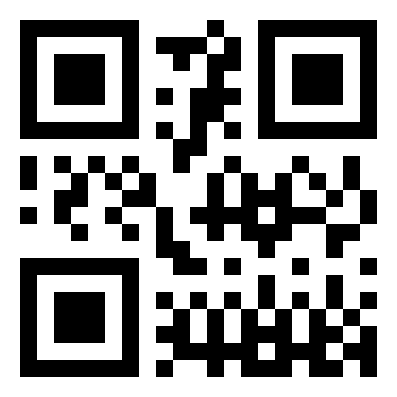
The SR\_get\_key subroutine is desired to (i) reject multiple keys in the same column pressed at the same time, (ii) recognize key release, i.e. "key up", as soon as possible and then allow immediately to read another key, and (iii) not allowing to input by mistake two or more times the same key, in 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).

An additional feature which is important is the possibility of buffering some keys. 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.*

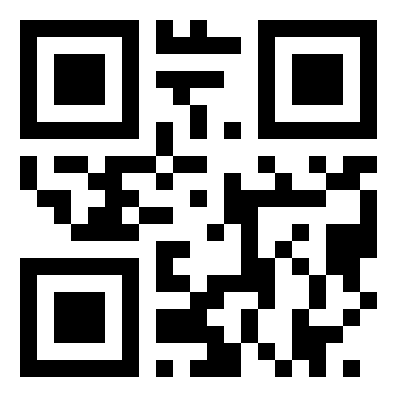
**B1.** *(Hardware identification)* Identify the PLC inputs and outputs, from/to the **keyboard**, to be used on the intrusion detection alarm system.



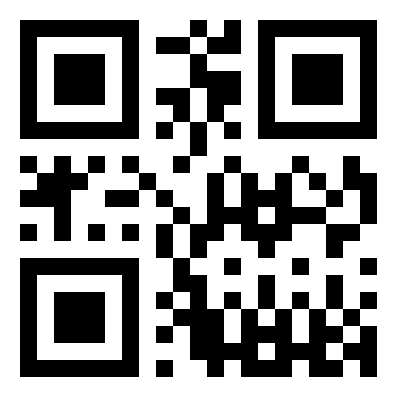
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PLC Input (chosen variable name)** | **PLC Identifier (memory address, %M)** |  | **PLC Output (chosen variable name)** | **PLC Identifier (memory address, %M)** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

*(Insert as many lines as needed in the tables)*

**B2.** Describe the reading/identifying strategy to be implemented to solve the keyboard reading problem.



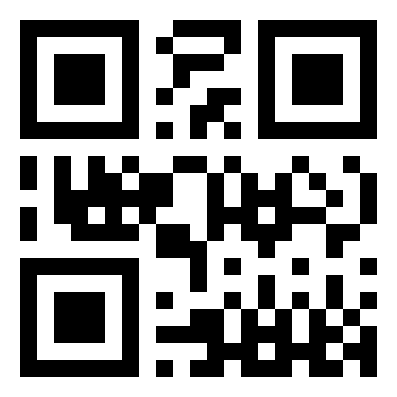
**B3.** Indicate the timer(s) and memory cell(s) used to implement SR\_get\_key.



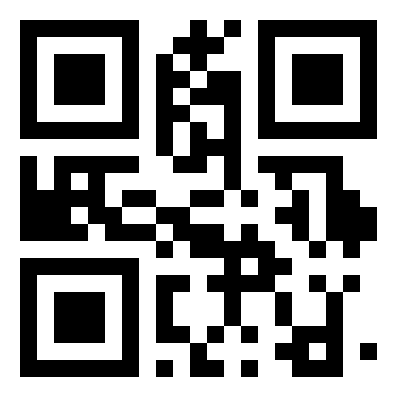
|  |  |  |
| --- | --- | --- |
| **Memory or timer (chosen variable name)** | **PLC physical address (if specified)** | **Short description of the memory or timer** |
|  |  |  |
|  |  |  |

*(Insert as many lines as needed in the table)*

**B4.** Upload to the PLC and run the program proposed for keyboard reading. Comment on the results obtained.



**B5.** *(Scan cycle period estimation)* Consider the PLC is running just the SR\_get\_key called by one section of MAST. How long is the scan cycle period? Suggestion: Write and test one **Structured Text program** based on a timer that counts 10 seconds and counts the number of scan cycles in those 10 seconds.

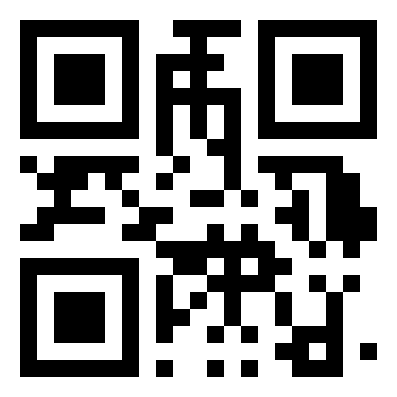


*Note: the PLC has system words indicating minimum and maximum scan-cycle times, however, unfortunately, they are not effective in simulation mode and therefore cannot be used.*

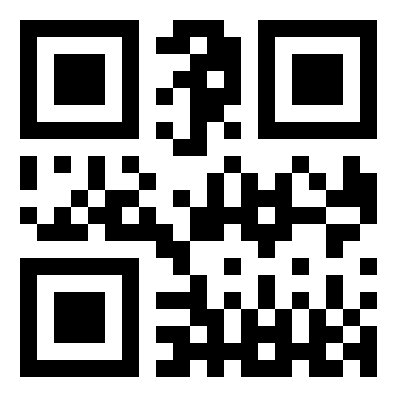
|  |
| --- |
| Average scan cycle time (program based on timer) = [msec] |

*(write here the ST program used to estimate the scan cycle period; add space as needed)*

**B6.** Use the **logging method indicated in part A question 9** to show changes in the inputs when a person enters the code 9963 using the keyboard. Show **time plots** of the relevant input signals.



**B7.** While running your PLC code in Unity Pro, run the following lines of code in Matlab. Note that is necessary to have the course software available in your computer.



login\_api

myterminal5

tkList= {0, [1 1], 2, [3 4], 4, [5 1 4], 6, [7 1:6], 8, 9};

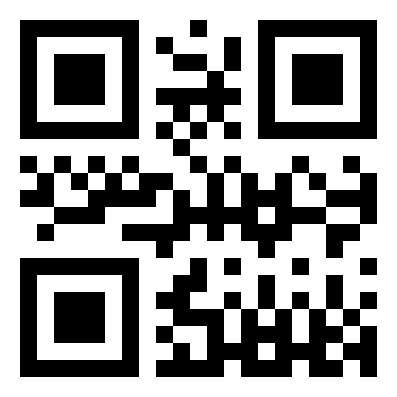
tu= tu\_sim( 'make\_tu', tkList );

myterminal5\_aux('tu\_keys\_run', tu);

a) Write in your report the value of matrix tu.

b) Describe the effect of these lines of code while your PLC code is running. Document the effects with time plots as was done in the previous question.

**B8.** Every hardware setup in the laboratory is based in one of two options, (i) a DMY 28FK input and output module, or (ii) one DEY16D2 input module and one DSY16T2 output module. Consider now an option where the outputs are implemented by a DSY16R5 module (see in the webpage of the course the item "HW IO Module" pointing to the file "PLC\_3\_HW\_IO\_Discrete.pdf"). What needs to be changed in the keyboard reading strategy?



# Annex A - 4x3 Keyboard

*(please do NOT include this annex in your report)*

The keyboards[[2]](#footnote-2) that are used in the laboratory have 12 keys arranged in a 4x3 matrix (see figure A1). 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 A1: An image and an electrical diagram of the 4x3 keyboard (keypad).

Figure A2 shows the inside of a keyboard. Notice that keyboards can be made simply of contacts which are short circuited when someone presses a button. For example, pressing key "eight" shorts the circuit of column 2 with line c. Most 4x3 keyboards have 4+3 wires. Some, as the one shown in the figure bellow, can have also an 8th wire which will not be used in the project.



Figure A2: Printed circuit board showing lines to be short-circuited by pressing buttons. Note there are no integrated circuits nor components, just lines (traces) and pads.

In figure A2 the pads with holes for wiring the keyboard to a terminal (or a PLC) are marked, from left-to-right, 3,2,1,a,b,c,d. This sorting is not standard among manufacturers. In most cases columns and lines are mixed and unsorted as the base printed circuit boards are simpler. Every 4x3 keyboard must be tested to identify the pads corresponding to the 3 columns and the 4 lines.

The simulated keyboard terminal is show in figure A3, a figure created with Matlab. The communication with the PLC is done through Modbus. As in the real keyboard, the keyboard columns have to be energized before being possible to observe energized keyboard lines upon pressing the buttons.

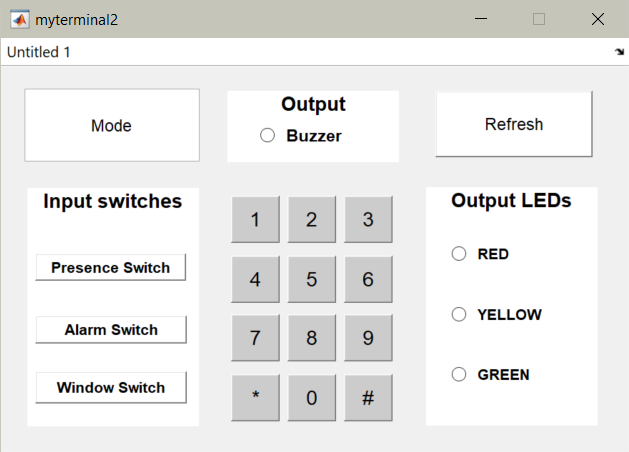


Figure A3: simulated terminal (Matlab figure), containing a numerical keyboard (keypad).

1. Original guide by Prof. Paulo J. Oliveira. 2021 revision by Prof. José Gaspar. [↑](#footnote-ref-1)
2. The keyboards used in the laboratory show just digits and therefore are more commonly known as numerical keypads or, simply, keypads. [↑](#footnote-ref-2)