CEC Continuing Education Center

DesignNews

Getting Started in Automation with Arduino

DAY 4: Understanding the IEC 61131-3 Functional Programming Language Specification

Sponsored by

DigiKey

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

Dr. Don Wilcher

Visit ‘Lecturer Profile’ in your console for more details.

Course Kit and Materials

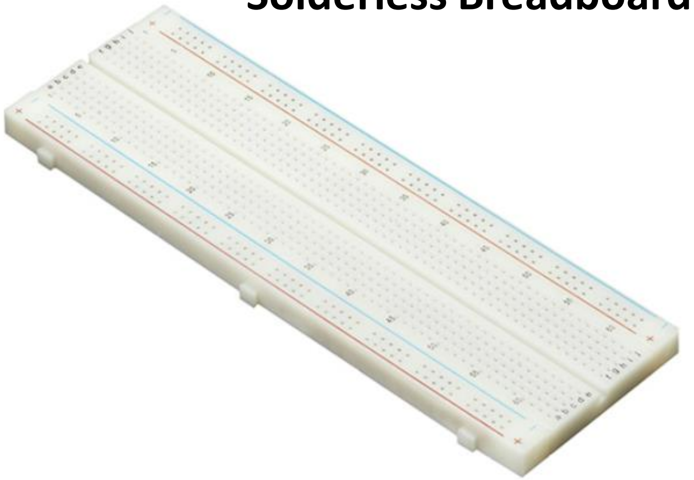
Arduino Opta

12VDC @ 500mA Wall Mount Power Supply

DC Motor: Medium Torque

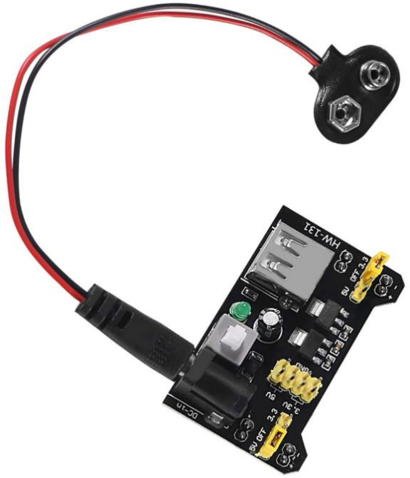
Adafruit Parts Pal Kit

Solderless Breadboard



Jumper Wires: Male to Male

Solderless Breadboard Power Supply



Agenda:

- Overview of the IEC 61131-3 Functional Programming Language Specification
- AI-assisted Automation Application: Simple Security System (OR Gate)
- Lab: Simple Security System Simulator

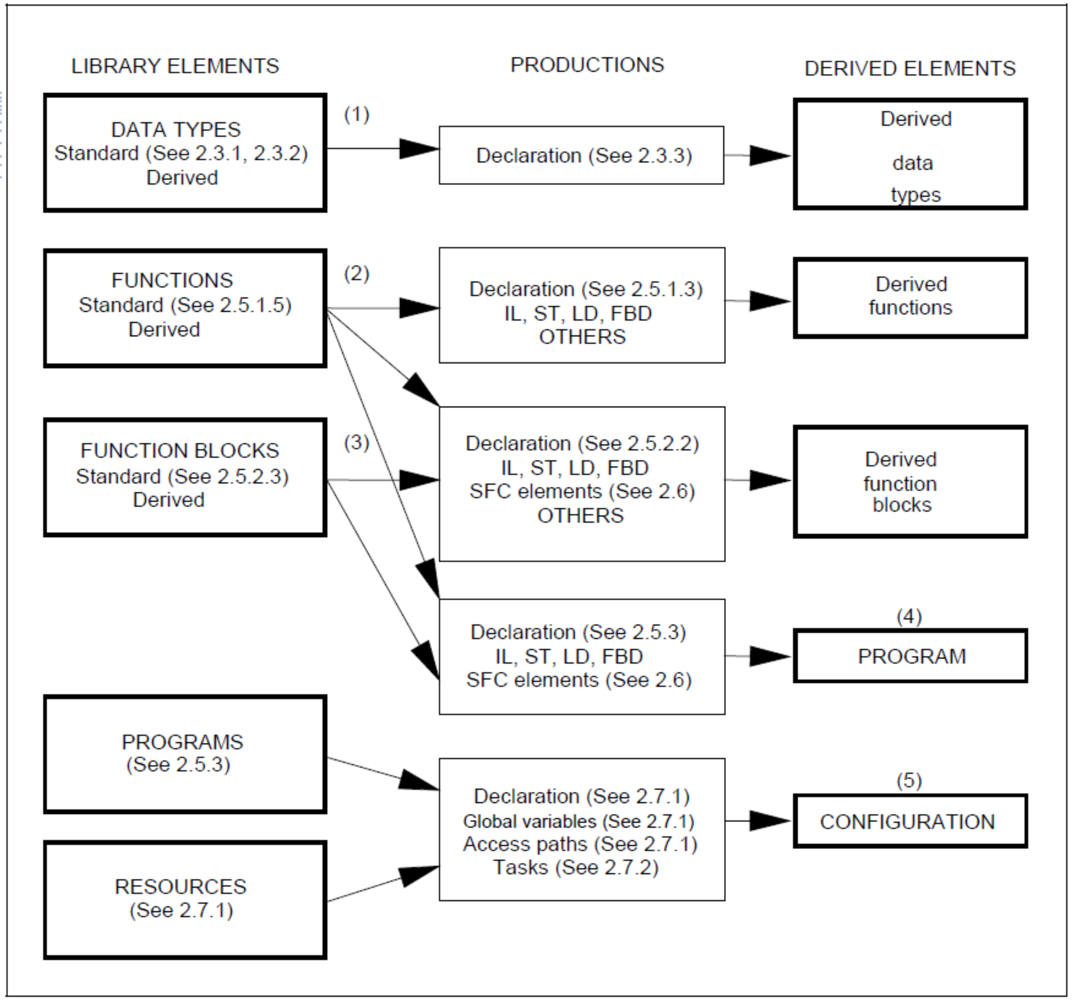
Seminal Research Perspective

“Programmable Logic Controller (PLC) is the most important component in industrial automation, and it has become one of the three pillars (robots, PLC, and CAD/CAM) of the modern industrial control technology”(Liao, 2007).

Overview of the IEC 6113-3 Functional Programming Language Specification

IEC 61131-3 Programming Model

Illustration courtesy of IEC 6113-3 Standard, Second Edition 2003.



LIBRARY ELEMENTS

PRODUCTIONS

DERIVED ELEMENTS

DATA TYPES
Standard (See 2.3.1, 2.3.2)
Derived

(1)

Declaration (See 2.3.3)

Derived data types

FUNCTIONS
Standard (See 2.5.1.5)
Derived

(2)

Declaration (See 2.5.1.3)
IL, ST, LD, FBD
OTHERS

Derived functions

FUNCTION BLOCKS
Standard (See 2.5.2.3)
Derived

(3)

Declaration (See 2.5.2.2)
IL, ST, LD, FBD
SFC elements (See 2.6)
OTHERS

Derived function blocks

Declaration (See 2.5.3)
IL, ST, LD, FBD
SFC elements (See 2.6)

(4)
PROGRAM

PROGRAMS
(See 2.5.3)

RESOURCES
(See 2.7.1)

Declaration (See 2.7.1)
Global variables (See 2.7.1)
Access paths (See 2.7.1)
Tasks (See 2.7.2)

(5)
CONFIGURATION

Overview of the IEC 6113-3 Functional Programming Language Specification. . .

The subclass **Program** block is the main **Declaration** element of interest in programming the Arduino Opta microPLC.

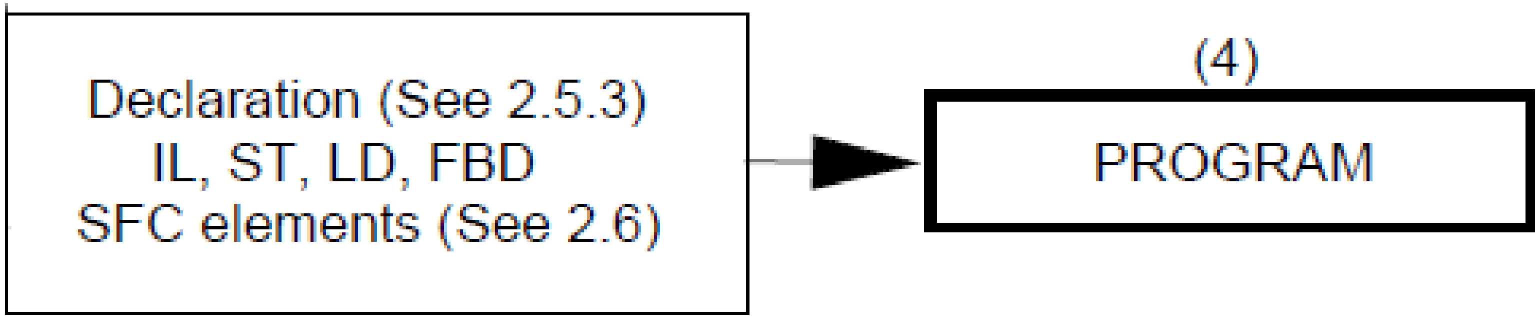


Illustration courtesy of IEC 6113-3 Standard, Second Edition 2003.

Question 1

The subclass Program block is the main setup element of interest in programming the Arduino Opta microPLC.

a) True

b) False

Overview of the IEC 6113-3 Functional Programming Language Specification. . .

- A **program** is defined as a logical assembly of all the programming language elements and constructs necessary for the intended (IEC 61131-3, p.83, 2003):

  a) signal processing for processes

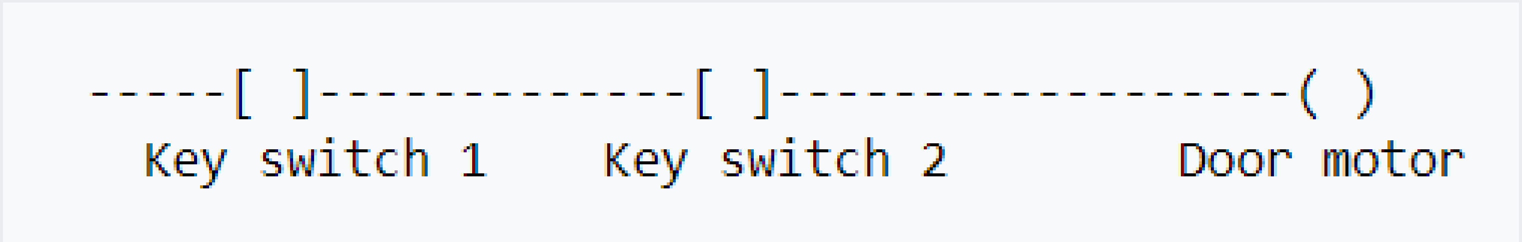
  b) control of a machine
- A program allows such signal processing and machine control within a programmable controller system.
- A network of programming elements defines the program's construction.

Overview of the IEC 6113-3 Functional Programming Language Specification. . .

- A **network** is a maximal set of interconnected elements, excluding the left and right rails (IEC 61131-3, p.135, 2003).
- Graphical languages represent the flow of a conceptual quantity through one or more networks representing a control plan (IEC 6113-3, p. 135, 2003).
- There are three graphical language flows used in a control plan.
  a) Power – aligns with relay ladder diagrams (LD)
  b) Signal – aligns with function blocks (FB)
  c) Activity – aligns with sequential function charts (SFC)

Overview of the IEC 6113-3 Functional Programming Language Specification. . .

The Power flow graphical language (LD) is primarily used in building PLC applications.



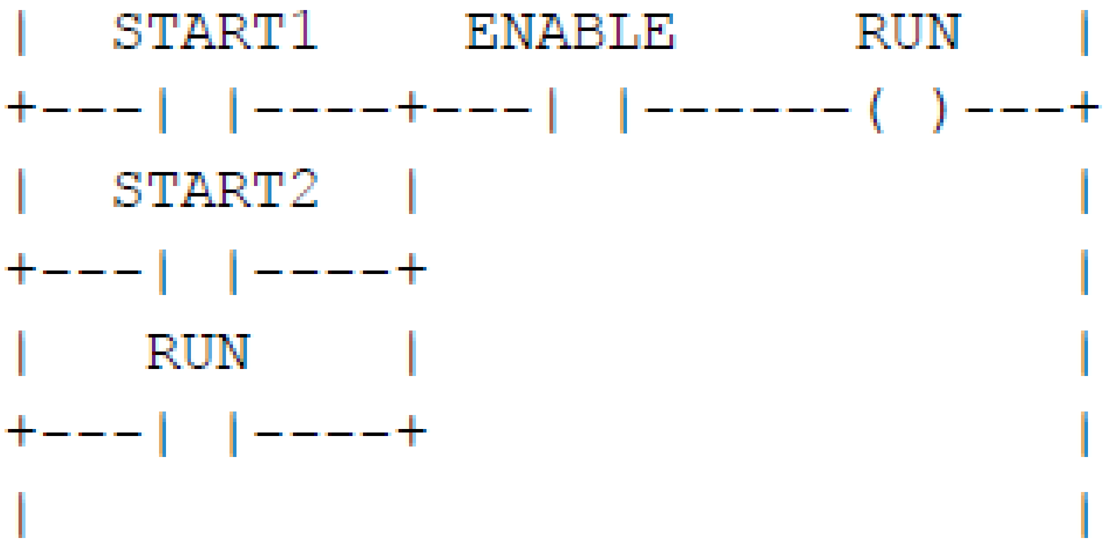
```
-----[ ]-------------[ ]-------------------( )
  Key switch 1     Key switch 2         Door motor
```

An Example LD Power Flow graphical language

Illustration courtesy of https://en.wikipedia.org/wiki/Ladder_logic

Overview of the IEC 6113-3 Functional Programming Language Specification. . .

IEC 61131-3 of an LD (Power Flow) program



```
|  START1    ENABLE     RUN    |
+---| |----+---| |-------( )---+
|  START2   |                  |
+---| |----+                   |
|   RUN     |                  |
+---| |----+                   |
|                              |
```

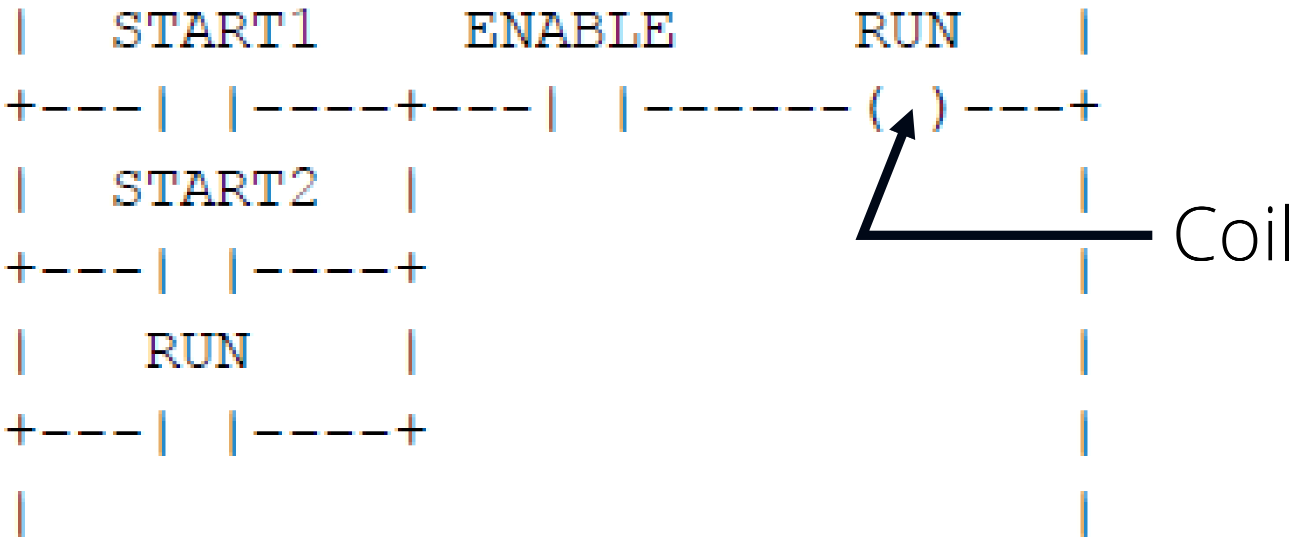
Illustration courtesy of IEC 6113-3 Standard, Second Edition 2003.

Overview of the IEC 6113-3 Functional Programming Language Specification. . .

- A contact is an element that imports a state to the horizontal link on its right side, equal to the Boolean function of the state of the horizontal link on its left side.
  a) The right and left sides of the horizontal link are the power rail.
  b) The horizontal link is the ladder diagram rung.
- A coil copies the state of the link on its left to the link on its right without modification. It stores an appropriate function of the state or transition of the left link into the associated Boolean variables (IEC 61131-3, p.142, 2003).

Overview of the IEC 6113-3 Functional Programming Language Specification. . .

Identification of the coil on an LD.



```
|  START1    ENABLE    RUN    |
+---| |----+---| |------( )---+
|  START2  |                  |
+---| |----+                  |
|   RUN    |                  |
+---| |----+                  |
|                             |
```

Coil

Illustration courtesy of IEC 6113-3 Standard, Second Edition 2003.

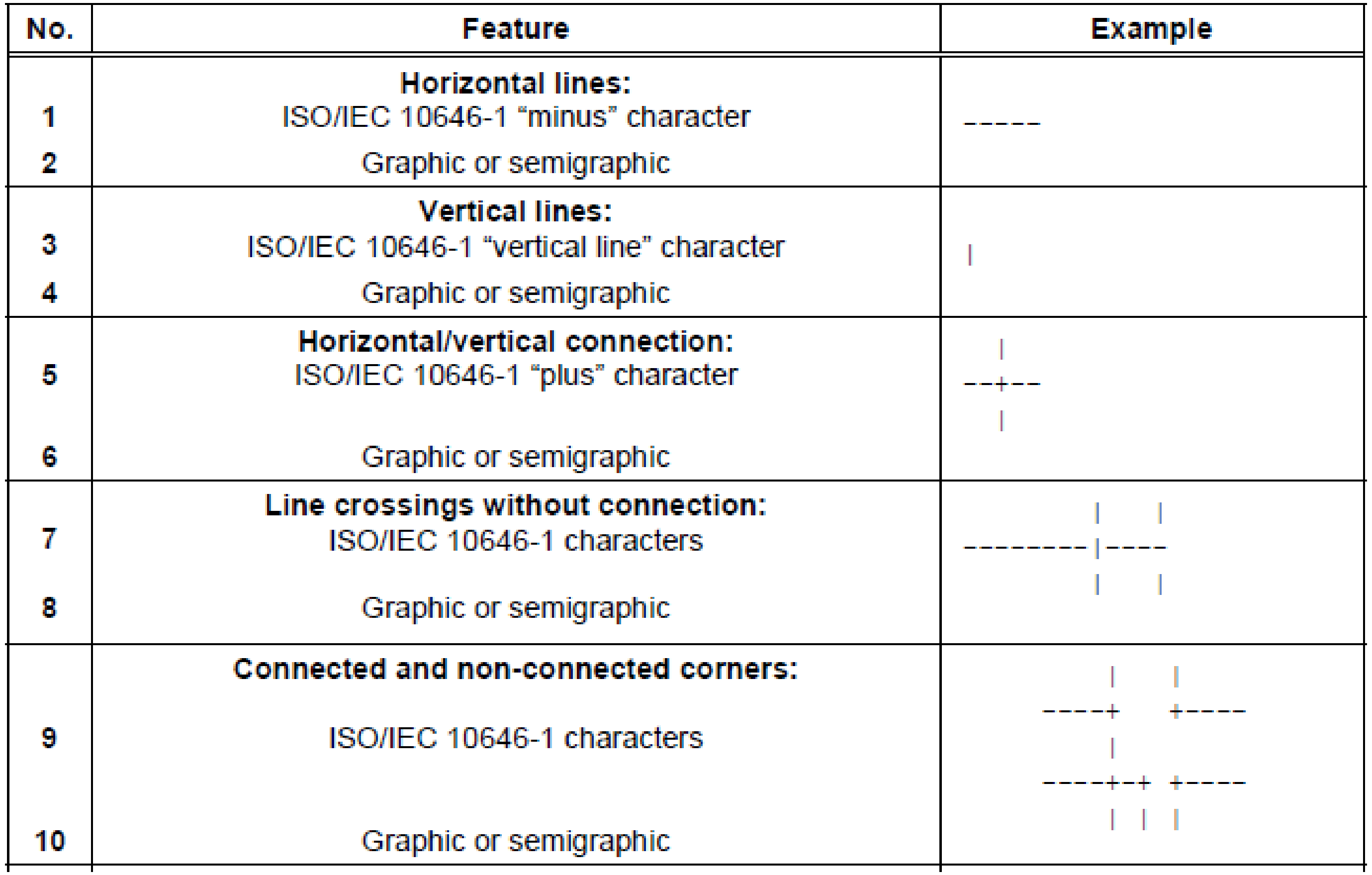
Question 2

A contact is an element that ________ a state to the horizontal link on its right side.

a) exports
b) transports
c) imports
d) transmits

Overview of the IEC 6113-3 Functional Programming Language Specification. . .

Representation of Lines:
Lines can be extended by the use of connectors (IEC 61131-3, p.135, 2003)

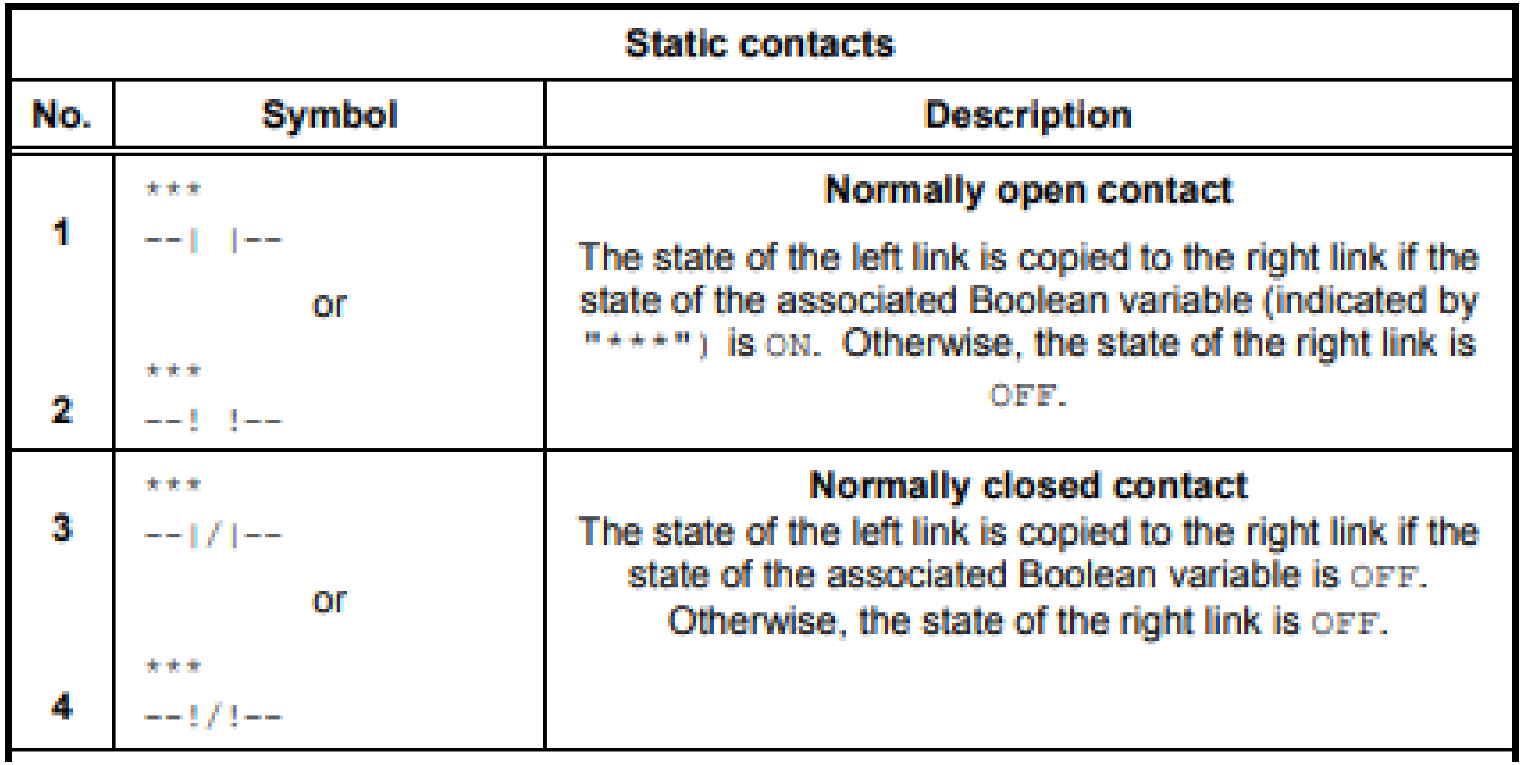


<table>
<tr><th>No.</th><th>Feature</th><th>Example</th></tr>
<tr><td>
1
2</td><td><b>Horizontal lines:</b>
ISO/IEC 10646-1 “minus” character
Graphic or semigraphic</td><td>-----</td></tr>
<tr><td>
3
4</td><td><b>Vertical lines:</b>
ISO/IEC 10646-1 “vertical line” character
Graphic or semigraphic</td><td>|</td></tr>
<tr><td>
5
6</td><td><b>Horizontal/vertical connection:</b>
ISO/IEC 10646-1 “plus” character
Graphic or semigraphic</td><td> |
--+--
  |</td></tr>
<tr><td>
7
8</td><td><b>Line crossings without connection:</b>
ISO/IEC 10646-1 characters
Graphic or semigraphic</td><td>| |
--------|----
| |</td></tr>
<tr><td>
9
10</td><td><b>Connected and non-connected corners:</b>
ISO/IEC 10646-1 characters
Graphic or semigraphic</td><td>| |
----+   +-----
|
----+-+ +-----
| | |</td></tr>
</table>

Illustration courtesy of IEC 6113-3 Standard, Second Edition 2003.

Overview of the IEC 6113-3 Functional Programming Language Specification. . .

Representation of Contacts:
A contact is an element which imparts a state to the horizontal link on its right side (IEC 61131-3, p.140, 2003)

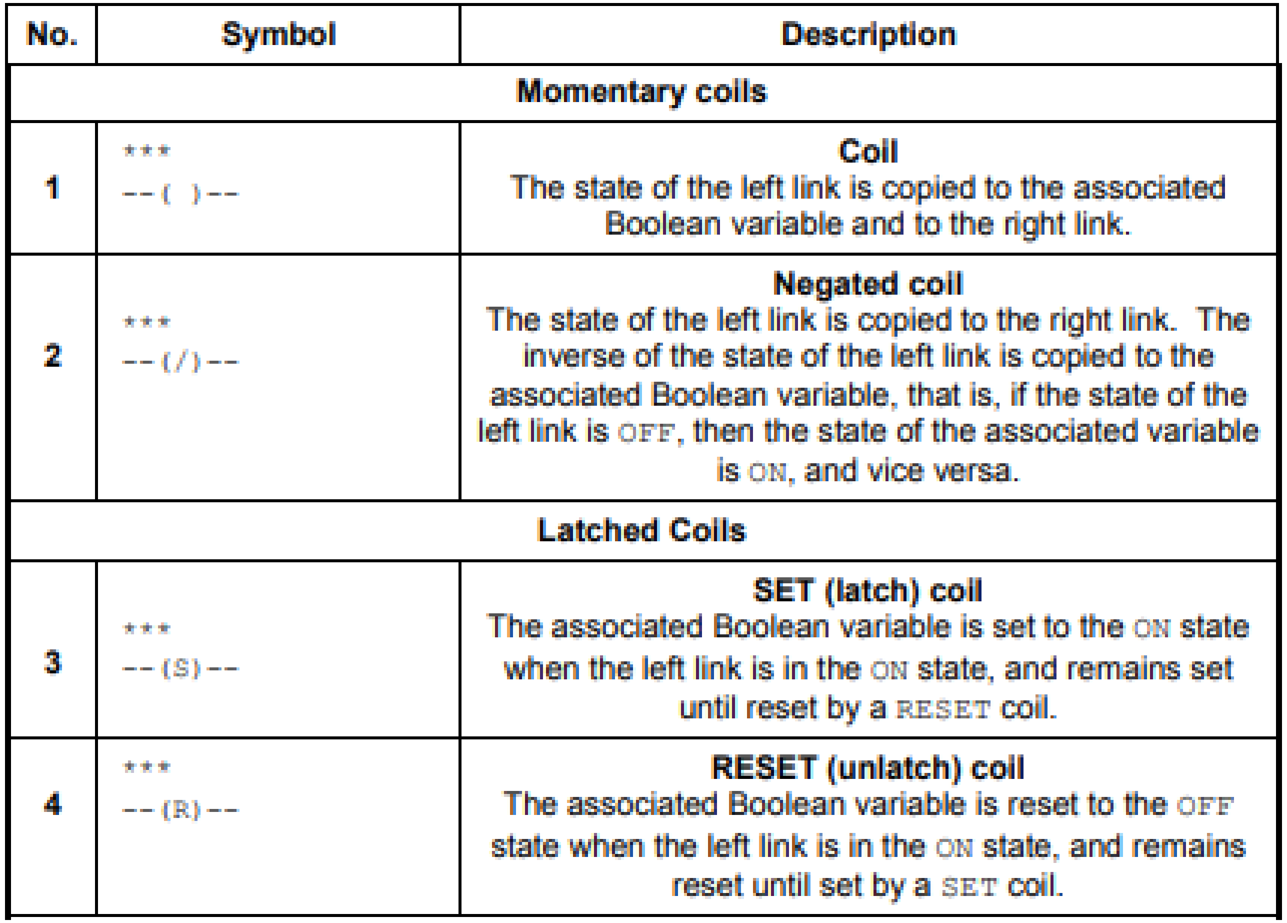


<table>
<tr><th colspan="3">Static contacts</th></tr>
<tr><th>No.</th><th>Symbol</th><th>Description</th></tr>
<tr><td>1</td><td>***
--| |--</td><td rowspan="3"><b>Normally open contact</b>
The state of the left link is copied to the right link if the state of the associated Boolean variable (indicated by "***") is ON. Otherwise, the state of the right link is OFF.</td></tr>
<tr><td></td><td>or</td></tr>
<tr><td>2</td><td>***
--! !--</td></tr>
<tr><td>3</td><td>***
--|/|--</td><td rowspan="3"><b>Normally closed contact</b>
The state of the left link is copied to the right link if the state of the associated Boolean variable is OFF. Otherwise, the state of the right link is OFF.</td></tr>
<tr><td></td><td>or</td></tr>
<tr><td>4</td><td>***
--!/!--</td></tr>
</table>

Illustration courtesy of IEC 6113-3 Standard, Second Edition 2003.

Overview of the IEC 6113-3 Functional Programming Language Specification. . .

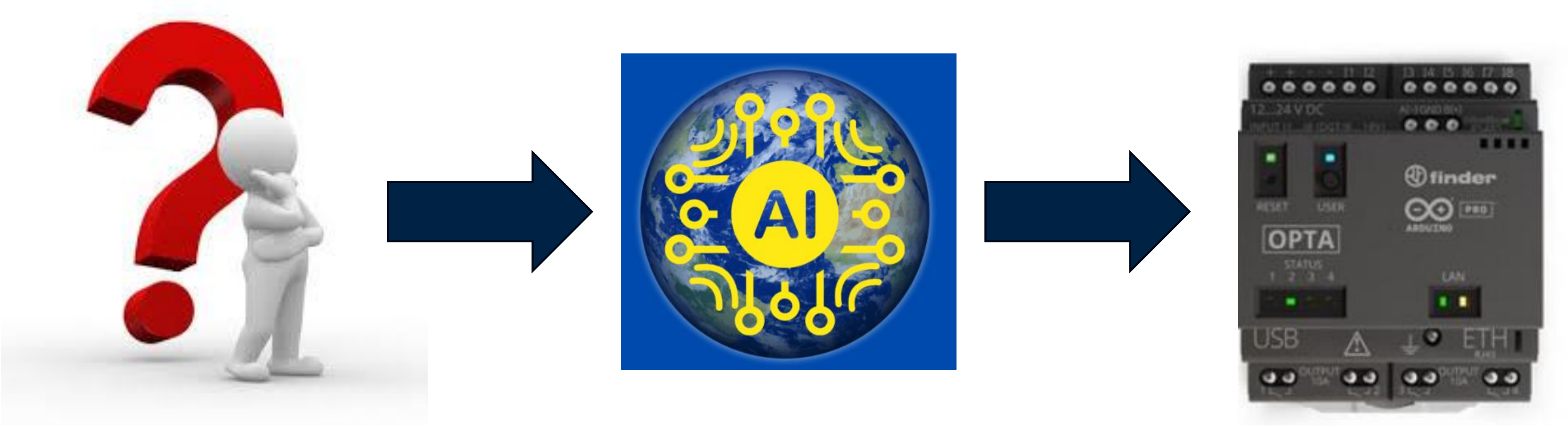
Representation of Coils:
A coil copies the state of the link on its left side (IEC 61131-3, p.140, 2003)



No.	Symbol	Description
		Momentary coils
1	*** --( )--	**Coil** The state of the left link is copied to the associated Boolean variable and to the right link.
2	*** --(/)--	**Negated coil** The state of the left link is copied to the right link. The inverse of the state of the left link is copied to the associated Boolean variable, that is, if the state of the left link is OFF, then the state of the associated variable is ON, and vice versa.
		Latched Coils
3	*** --(S)--	**SET (latch) coil** The associated Boolean variable is set to the ON state when the left link is in the ON state, and remains set until reset by a RESET coil.
4	*** --(R)--	**RESET (unlatch) coil** The associated Boolean variable is reset to the OFF state when the left link is in the ON state, and remains reset until set by a SET coil.

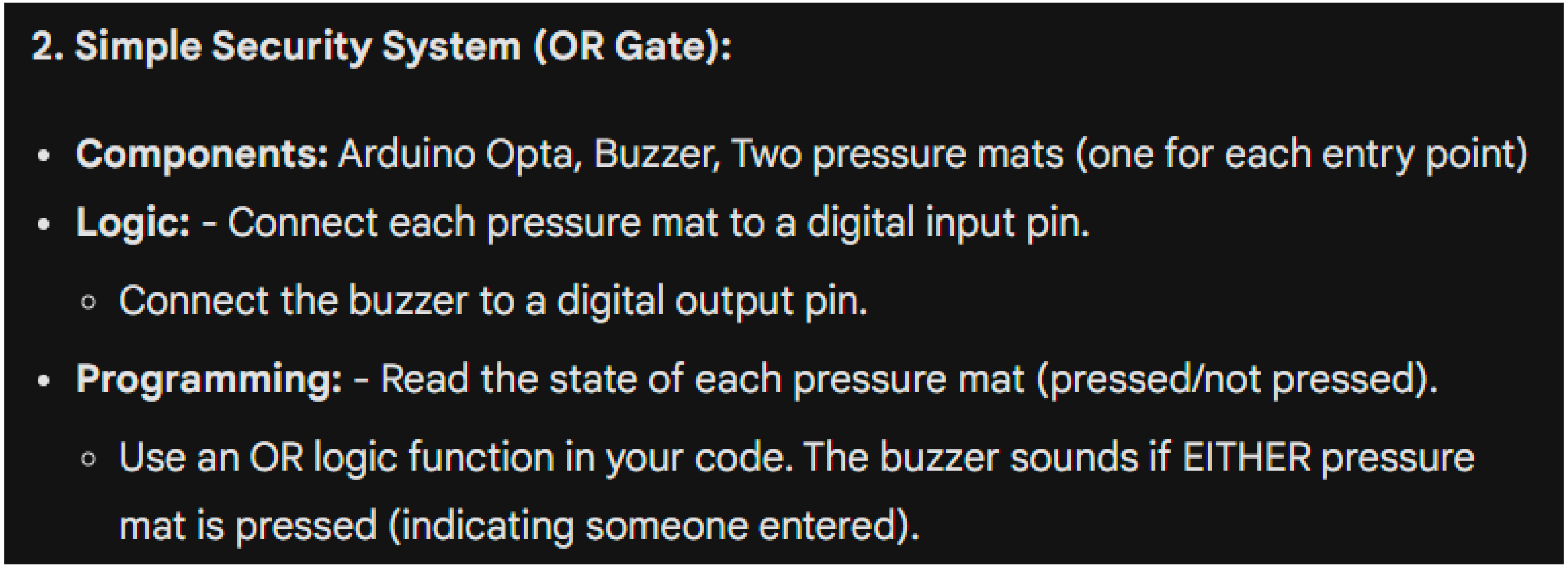
Illustration courtesy of IEC 6113-3 Standard, Second Edition 2003.

AI-assisted Application: Simple Security System (OR Gate)



What Automation application can be generated for the Arduino Opta using a Large Language Model?

AI-assisted Application: Simple Security System (OR Gate). . .



2. **Simple Security System (OR Gate):**

- **Components:** Arduino Opta, Buzzer, Two pressure mats (one for each entry point)
- **Logic:** - Connect each pressure mat to a digital input pin.
  - Connect the buzzer to a digital output pin.
- **Programming:** - Read the state of each pressure mat (pressed/not pressed).
  - Use an OR logic function in your code. The buzzer sounds if EITHER pressure mat is pressed (indicating someone entered).

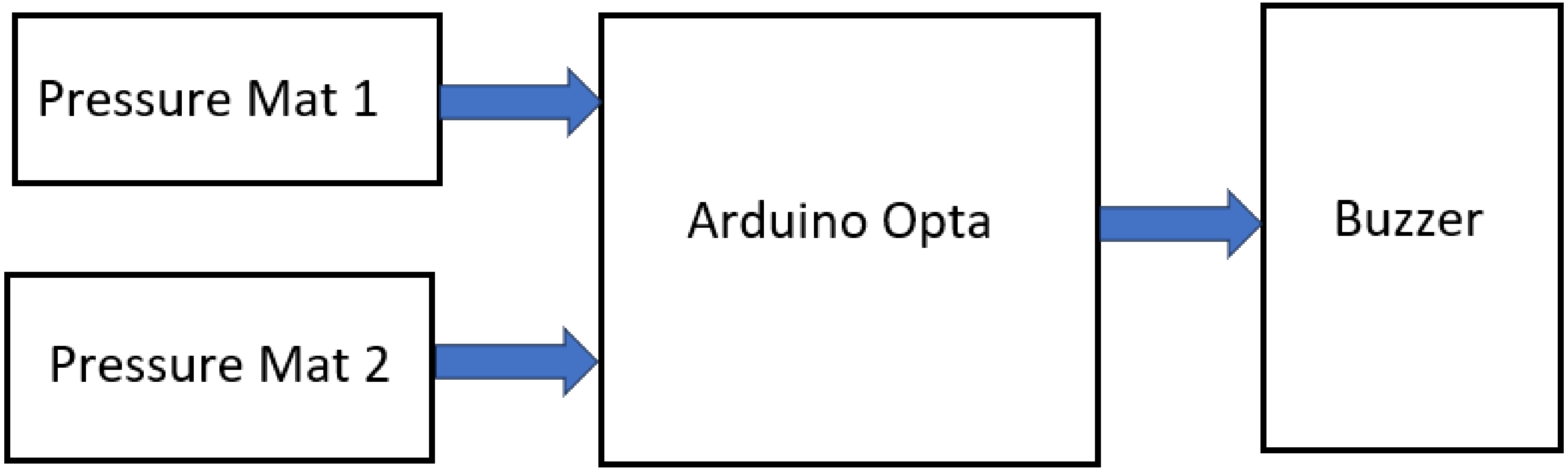
A Physical Simulator will be built using the Arduino Opta micro-PLC, two pushbutton switches, and a buzzer. The IEC 61131-3 Specification will provide guidance in building the LD program.

Question 3

On slide 21, which AI LLM was used to create the Simple Security System Application?

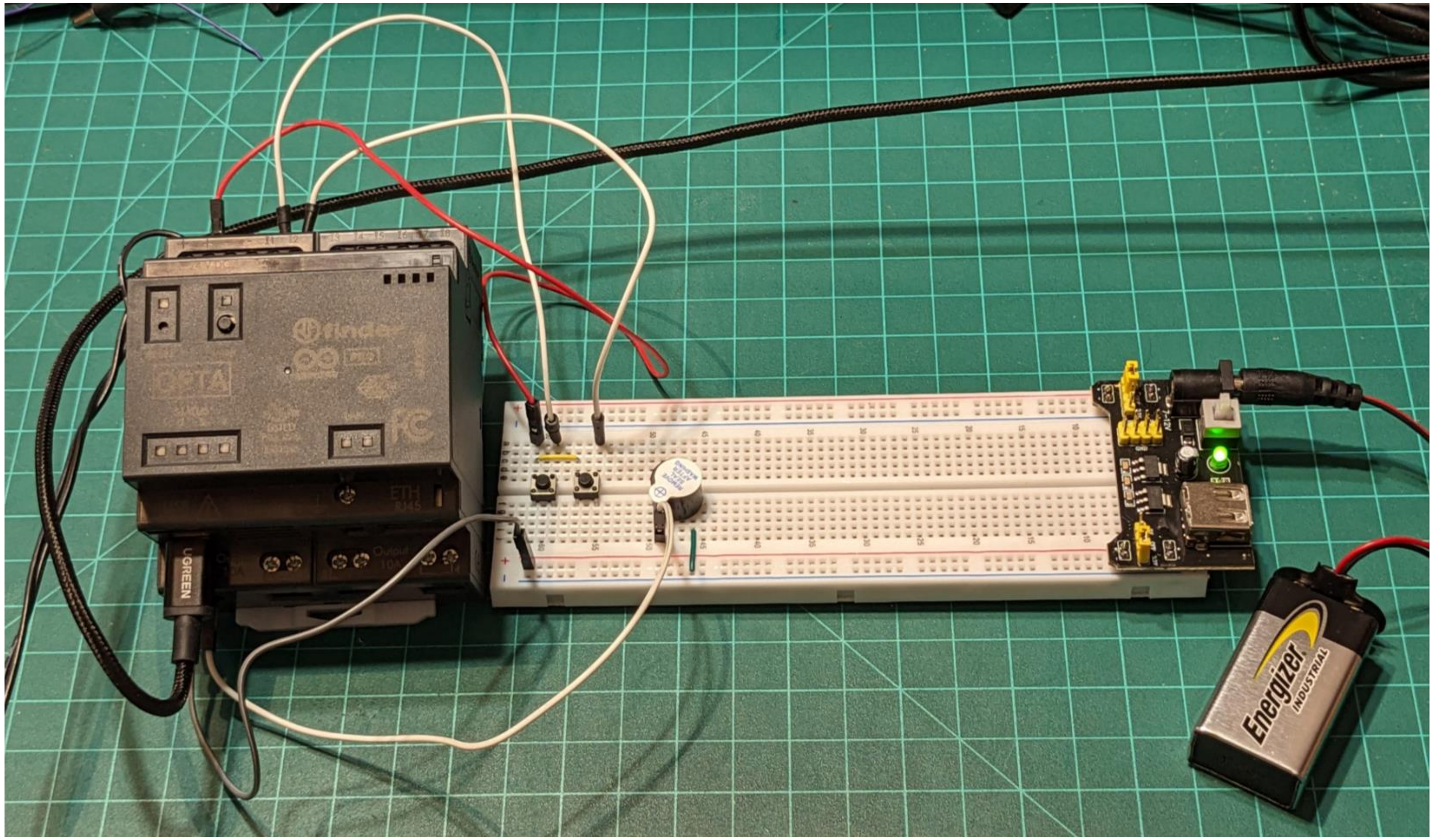
- **a) Dalle2**
- **b) Co-Pilot**
- **c) Gemini**
- **d) ChatGPT**

AI-assisted Application: Simple Security System (OR Gate). . .



A Physical Simulator Block Diagram

Lab: Simple Security System Simulator



Lab: Simple Security System Simulator. . .

Lab Objectives:

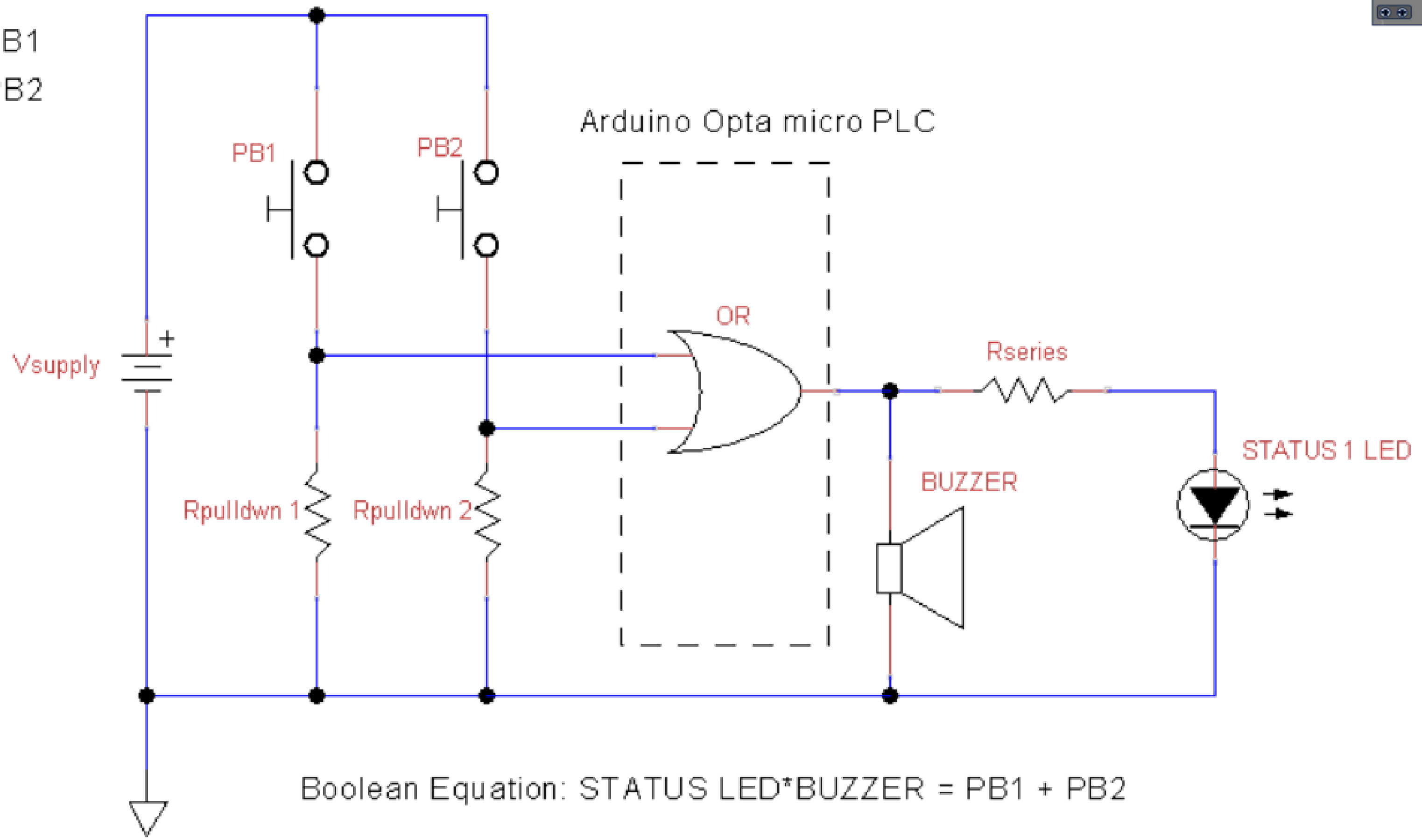
- Participants will learn to set up communications using the Arduino PLC IDE.
- Participants will learn to create a Ladder Diagram Logic Function program using the Arduino PLC IDE.
- Participants will learn how to branch a bit instruction.
- Participants will learn how program and control an Arduino Opta Relay.
- Participants will learn to download, run, and test a Ladder Diagram Logic Function program.

Lab: Simple Security System Simulator. . .

Digital IC Concept: Discrete logic gate circuit simulating a Simple Security System.

Press_Mat 1 = PB1

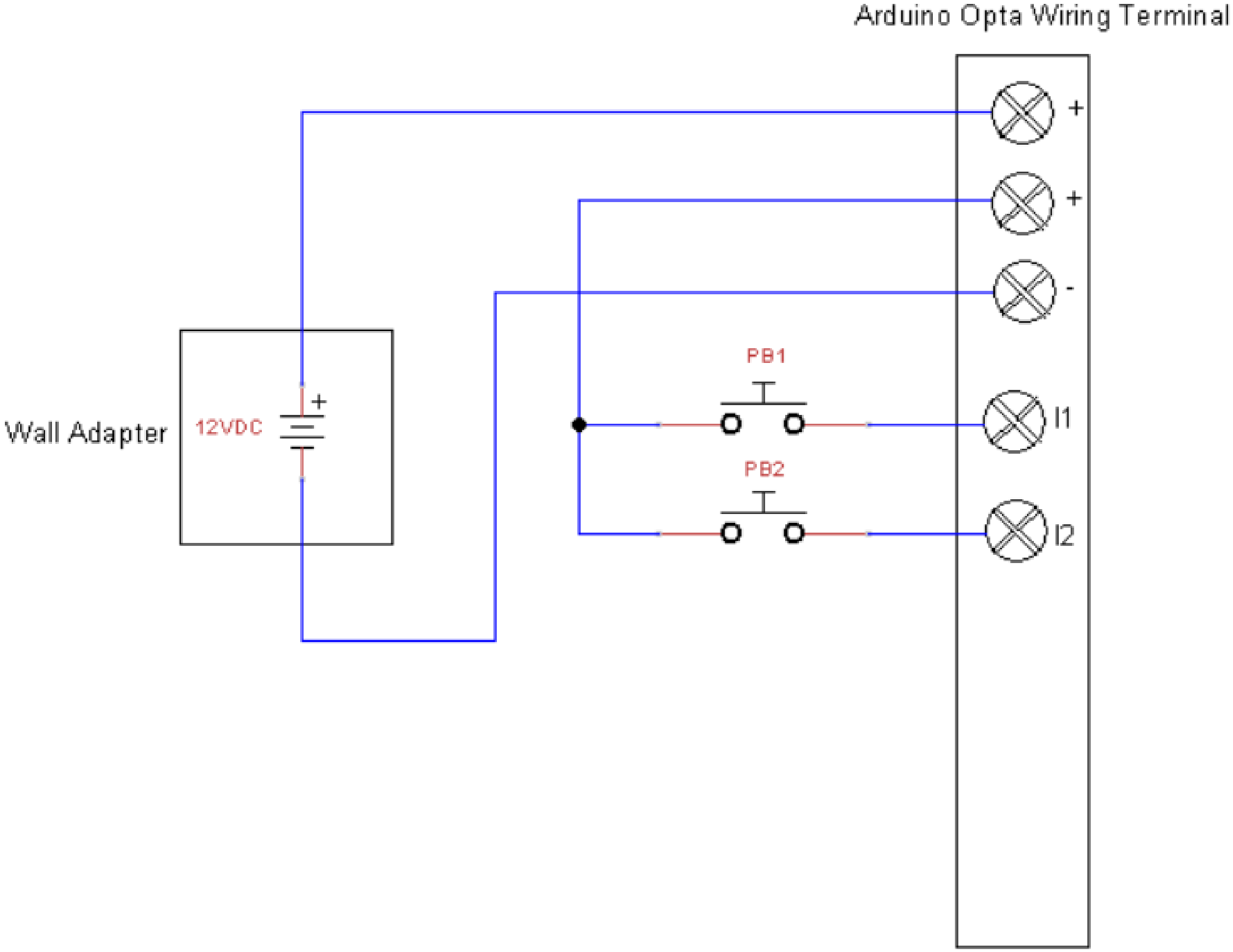
Press_Mat 2 = PB2



Boolean Equation: STATUS LED*BUZZER = PB1 + PB2

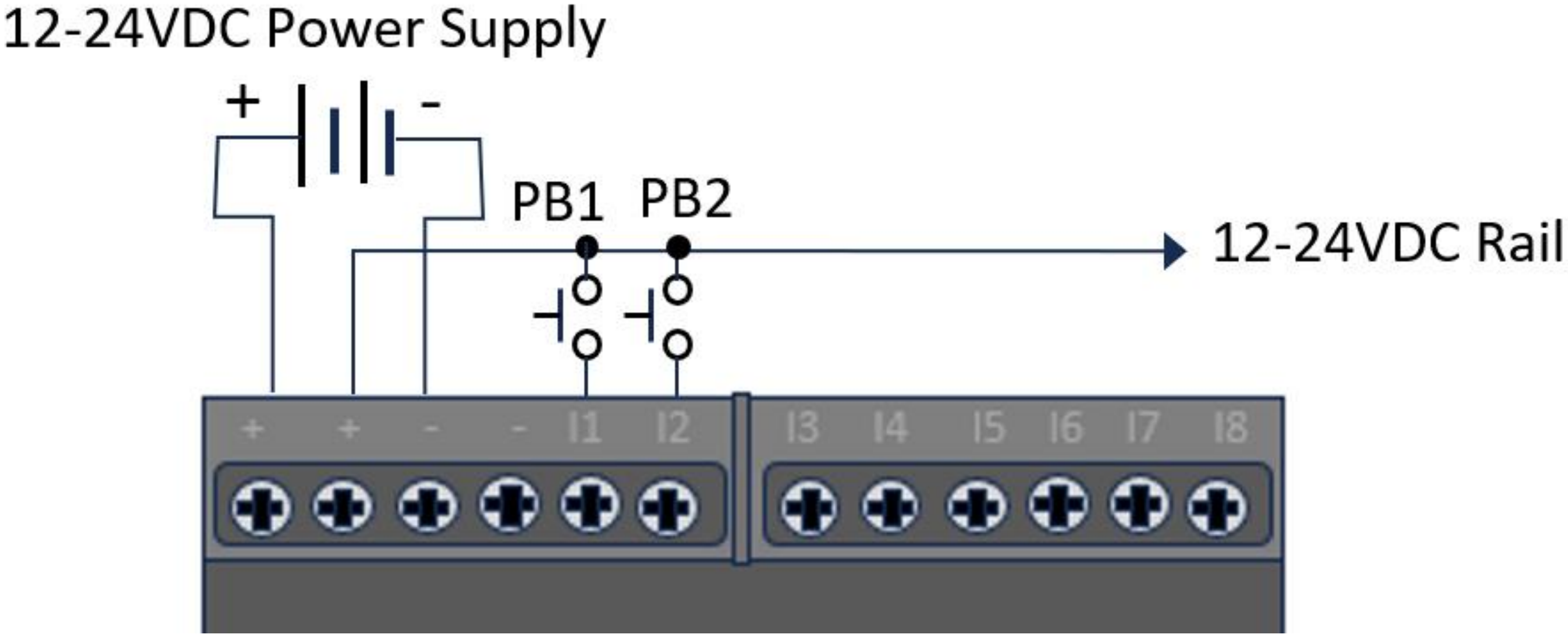
Wiring 2 Pushbutton Switches To the Arduino Opta. . .

A Prototyping Concept for a Digital Switch Simulator: Input Terminal Wiring Diagram



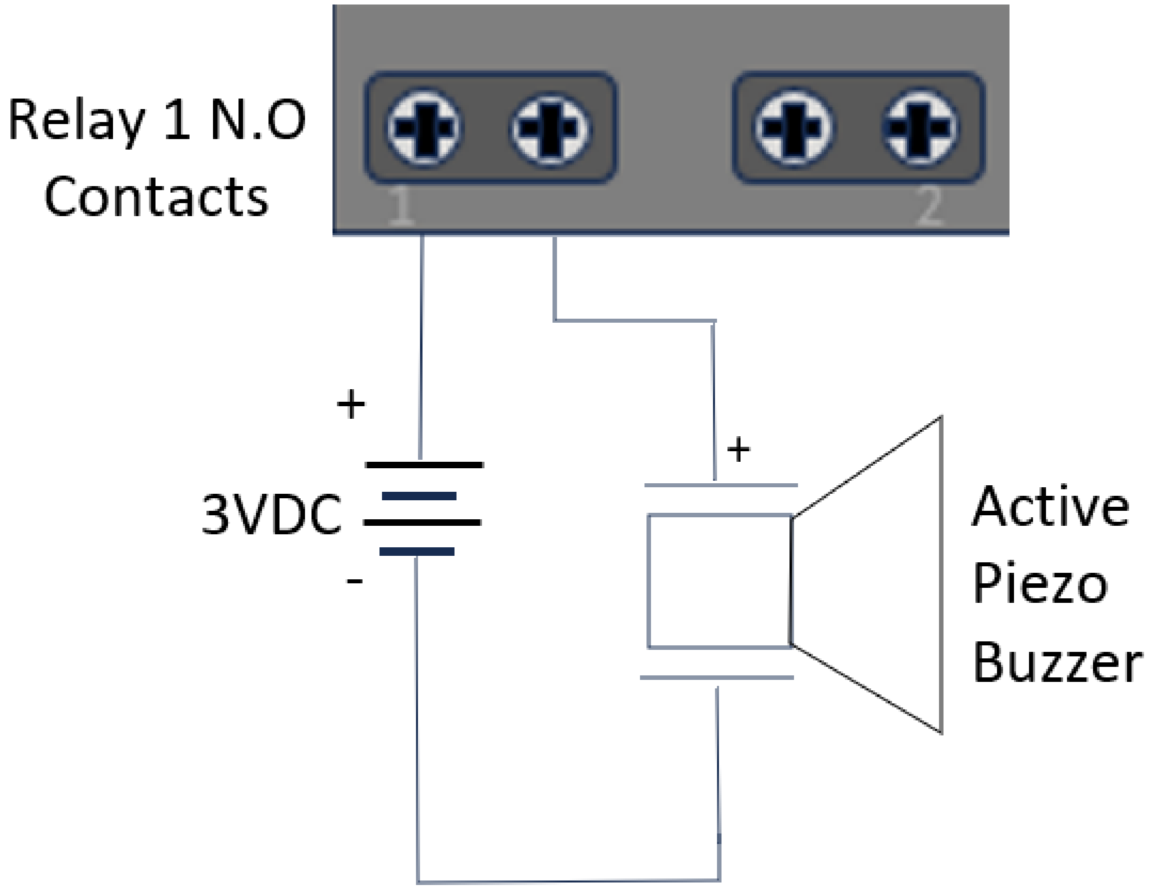
Wiring 2 Pushbutton Switches To the Arduino Opta. . .

2 tactile pushbutton switches are wired to the terminal points I1 and I2 of the Arduino Opta.

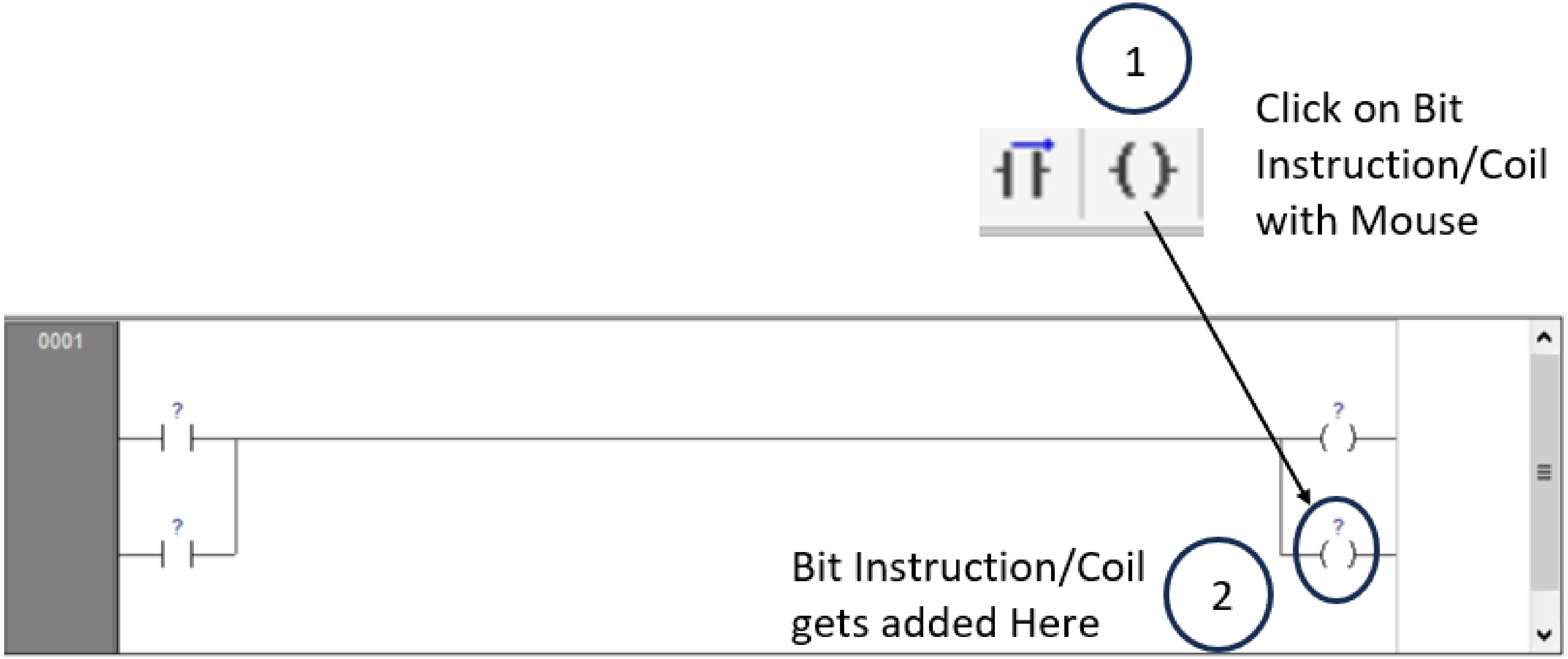


Wiring an Active Piezo Buzzer To the Arduino Opta

An Active Piezo Buzzer wired to the terminal point "1" Relay Contact of the Arduino Opta.



Lab: Simple Security System Simulator…
Adding Parallel Bit Instruction



Click on Bit Instruction/Coil with Mouse

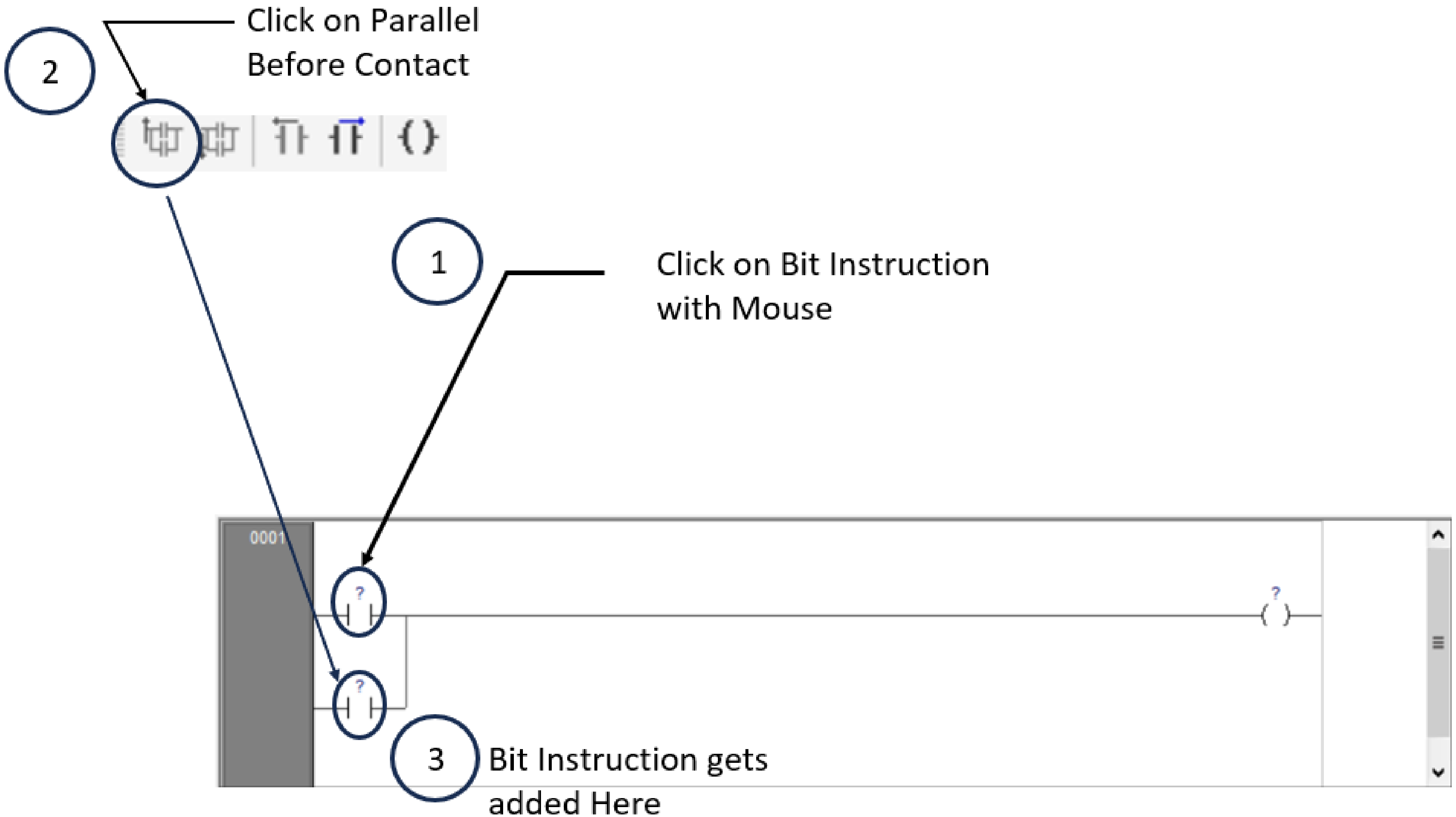
Bit Instruction/Coil gets added Here

Question 4

In reviewing slide 30, what bit instruction is used to add a parallel coil for an LD rung?

- **a) contact**
- **b) parallel coil**
- **c) coil**
- **d) none of the above**

Lab: Simple Security System Simulator. . . Adding Parallel Bit Instructions



2 Click on Parallel Before Contact

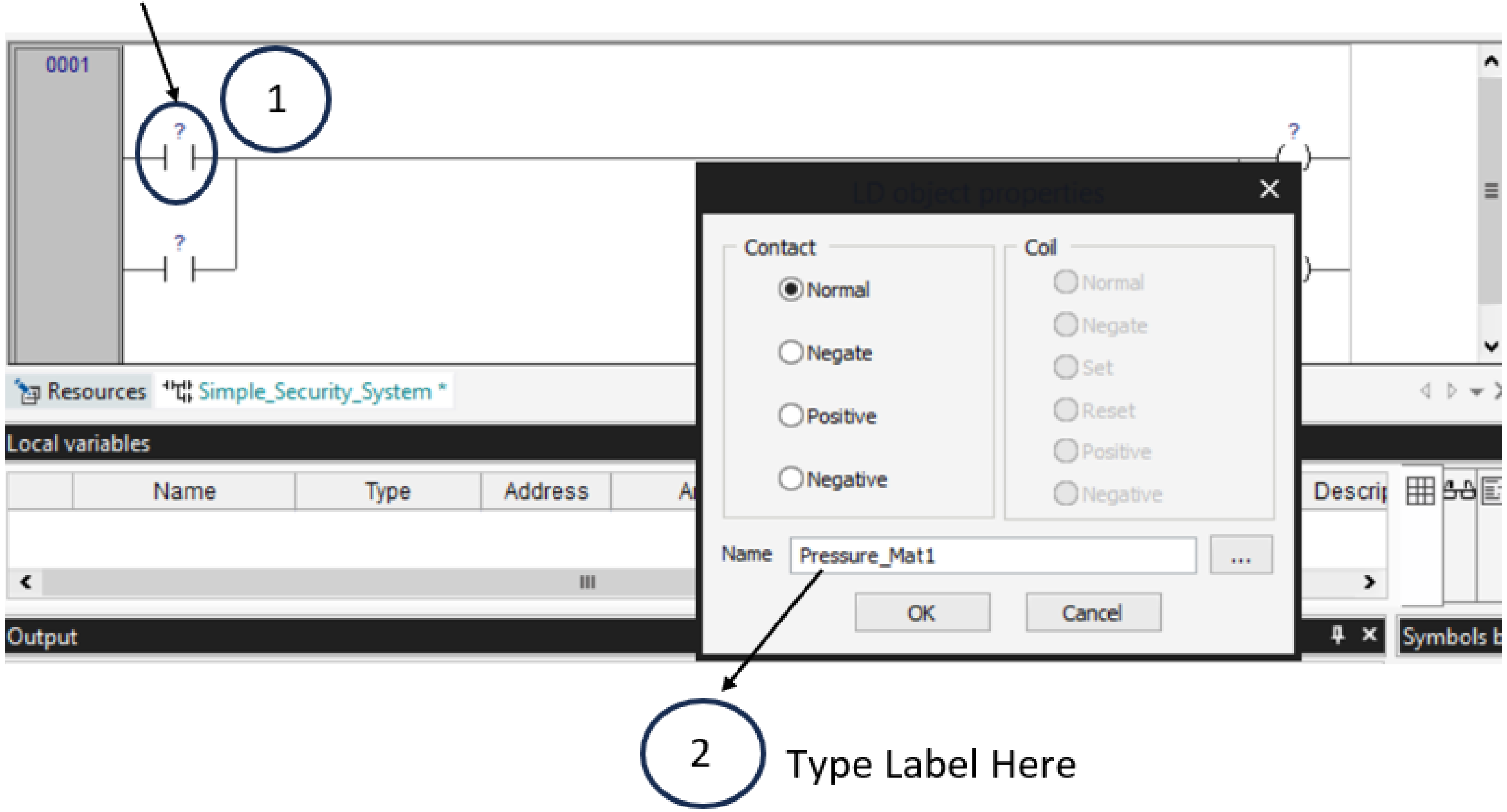
1 Click on Bit Instruction with Mouse

3 Bit Instruction gets added Here

Lab: Simple Security System Simulator. . . Labeling Bit Instructions

Repeat the Step for the remaining Bit Instructions.

Click on Bit Instruction/Contact



LD object properties

Contact
- ● Normal
- ○ Negate
- ○ Positive
- ○ Negative

Coil
- ○ Normal
- ○ Negate
- ○ Set
- ○ Reset
- ○ Positive
- ○ Negative

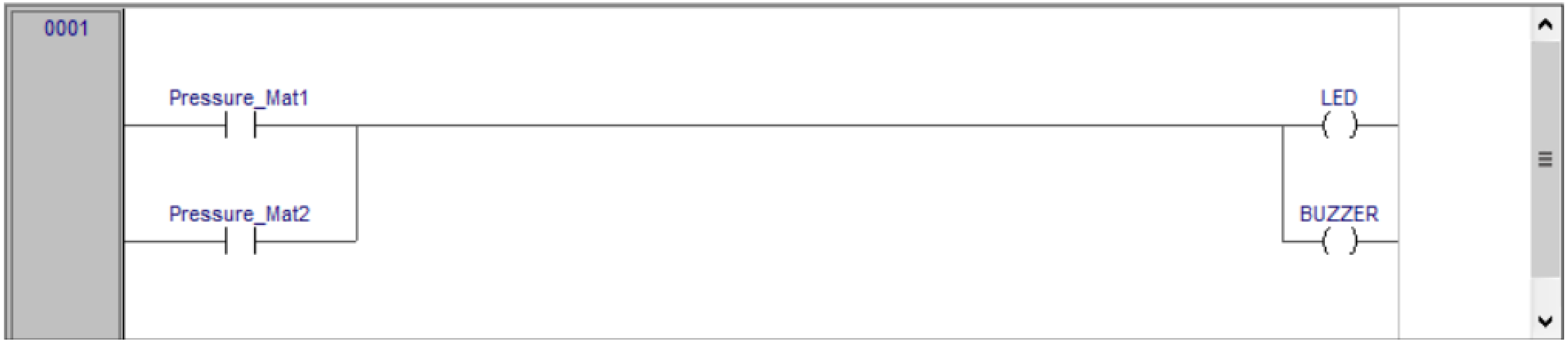
Name: Pressure_Mat1 ...

OK Cancel

Type Label Here

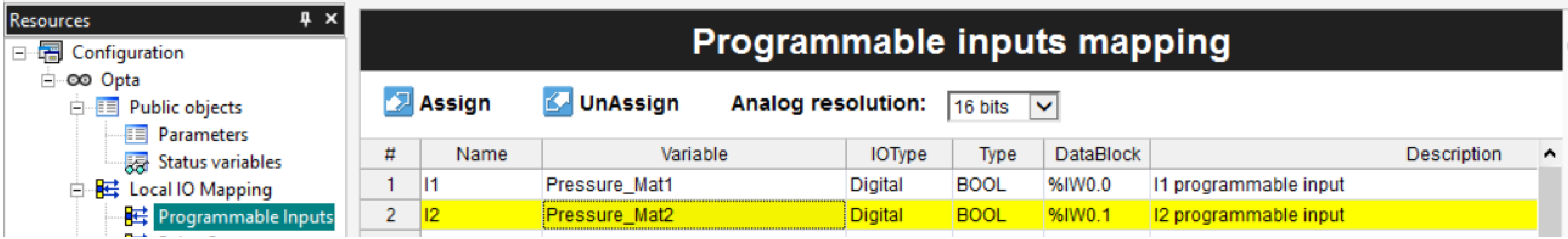
Lab: Simple Security System Simulator… Labeling Bit Instructions

Completely Labeled Bit Instructions



Lab: Simple Security System Simulator…
Labeling Bit Instructions

Labeling and Mapping Pressure_Mat1 and Pressure_Mat2 to I1 and I2 Terminals

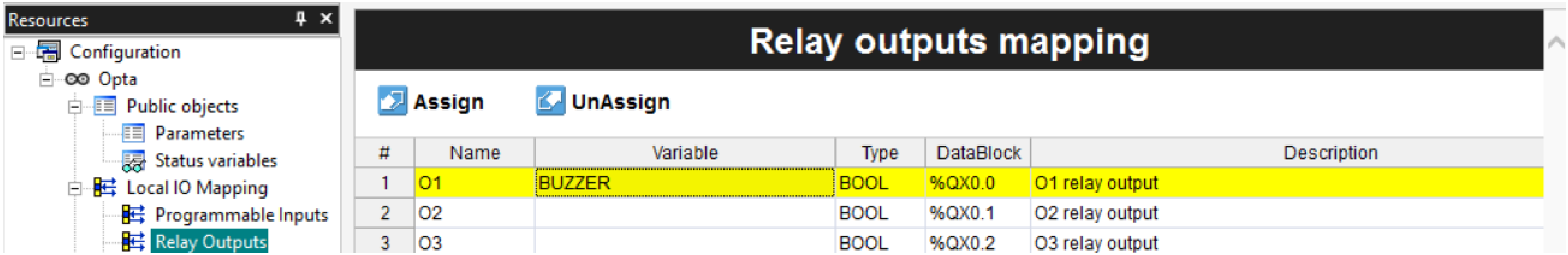


Analog resolution: 16 bits

#	Name	Variable	IOType	Type	DataBlock	Description
1	I1	Pressure_Mat1	Digital	BOOL	%IW0.0	I1 programmable input
2	I2	Pressure_Mat2	Digital	BOOL	%IW0.1	I2 programmable input

Lab: Simple Security System Simulator…
Labeling Bit Instructions

Labeling and Mapping Buzzer
to Output Relay 1



Resources

- Configuration
  - Opta
    - Public objects
      - Parameters
      - Status variables
    - Local IO Mapping
      - Programmable Inputs
      - Relay Outputs

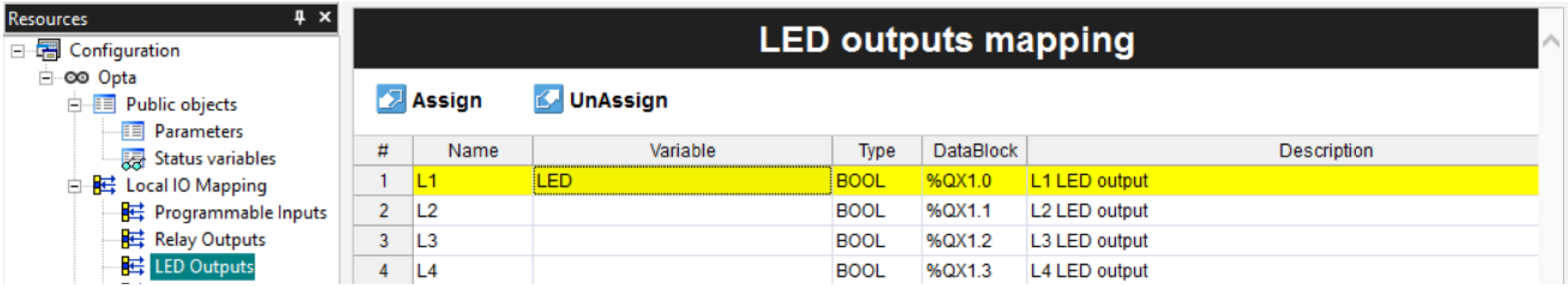
Relay outputs mapping

Assign UnAssign

#	Name	Variable	Type	DataBlock	Description
1	O1	BUZZER	BOOL	%QX0.0	O1 relay output
2	O2		BOOL	%QX0.1	O2 relay output
3	O3		BOOL	%QX0.2	O3 relay output

Lab: Simple Security System Simulator. . .
Labeling Bit Instructions

Labeling and Mapping LED to
STATUS LED 1

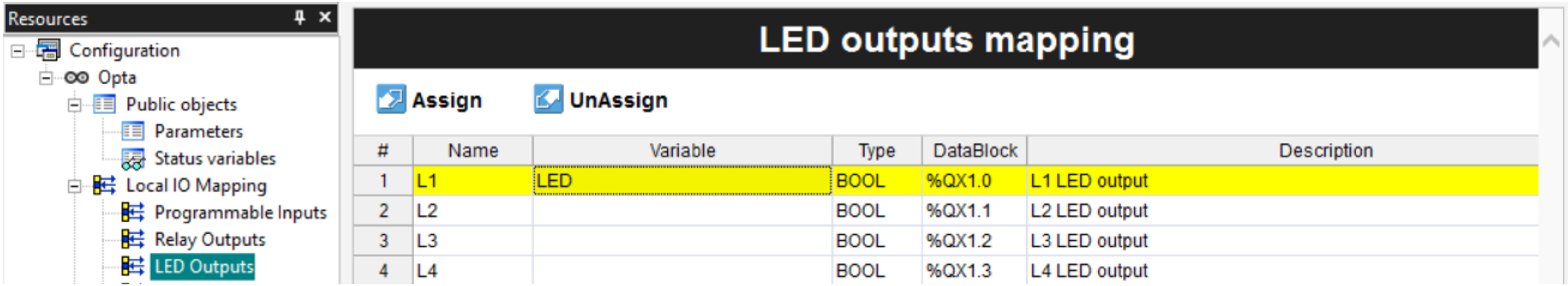


LED outputs mapping

#	Name	Variable	Type	DataBlock	Description
1	L1	LED	BOOL	%QX1.0	L1 LED output
2	L2		BOOL	%QX1.1	L2 LED output
3	L3		BOOL	%QX1.2	L3 LED output
4	L4		BOOL	%QX1.3	L4 LED output

Lab: Simple Security System Simulator… Labeling Bit Instructions

Labeling and Mapping LED to STATUS LED 1

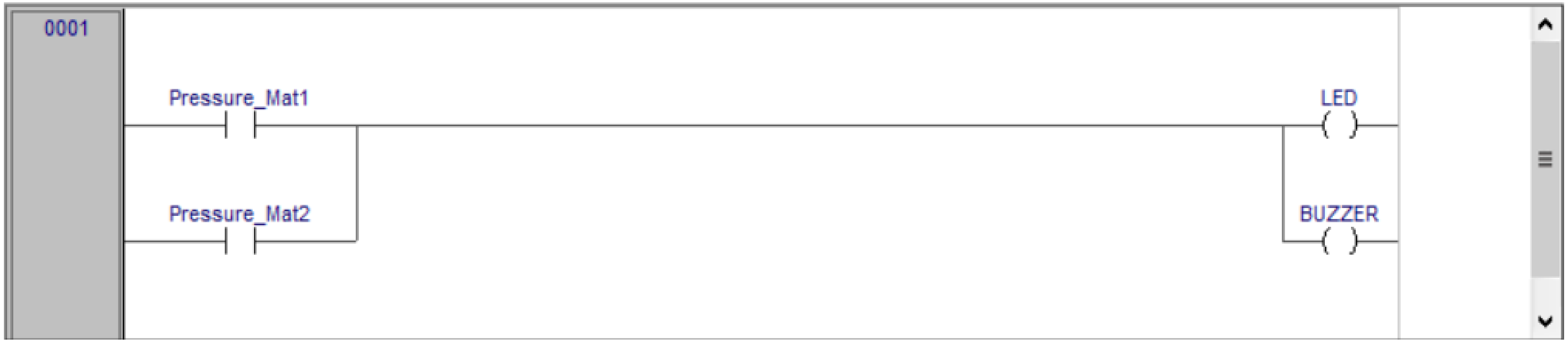


LED outputs mapping

#	Name	Variable	Type	DataBlock	Description
1	L1	LED	BOOL	%QX1.0	L1 LED output
2	L2		BOOL	%QX1.1	L2 LED output
3	L3		BOOL	%QX1.2	L3 LED output
4	L4		BOOL	%QX1.3	L4 LED output

Lab: Simple Security System Simulator…

Completely Simple Security System Simulator

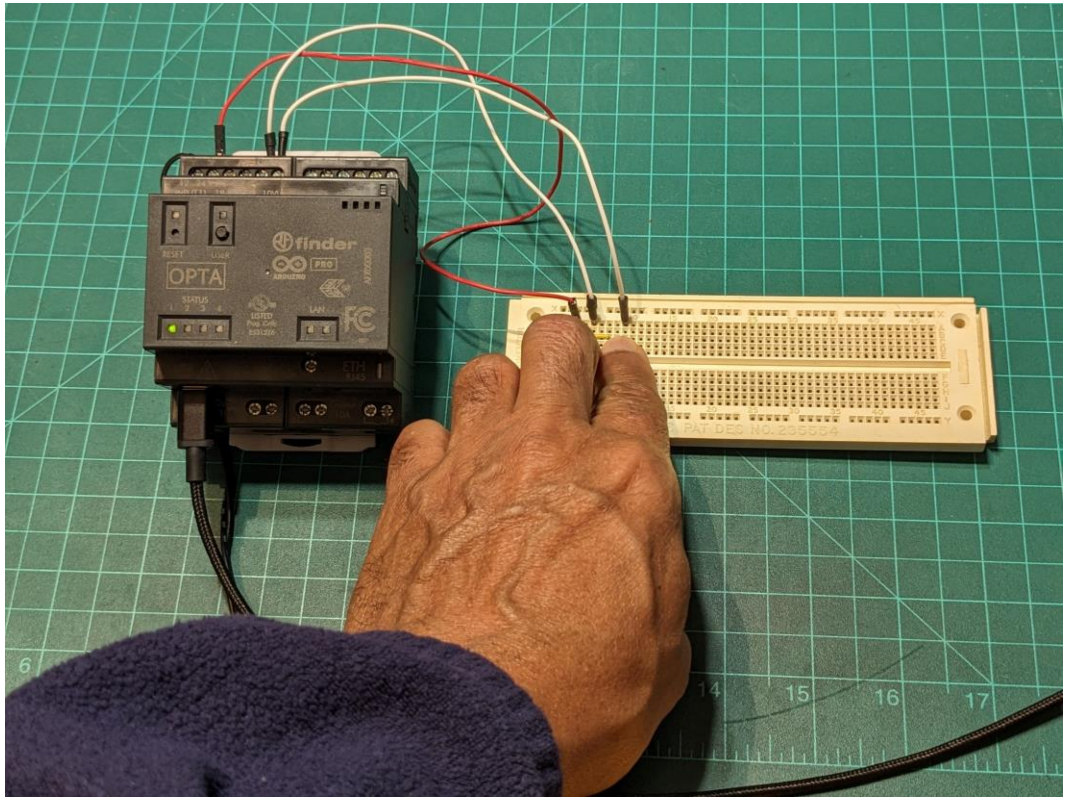


Download LD program to the Arduino Opta using slides 36 and 37 from Day 2 ppt/pdf.

Lab: Programming an Arduino Opta Ladder Diagram Logic Function. . .

The AND Gate Logic Function: Simulating An Automatic Light Control System

Functional Automatic Light Control System – AND Gate Logic Simulator



Watch Video Clip!

https://youtu.be/r1xgHIo0rBo

Question 5

In reviewing slide 38, what address (DataBlock) is associated with the LED variable?

- **a) %QX0.0**
- **b) %QX1.1**
- **c) %QX1.0**
- **d) %QX1.1**

Thank you for attending

Please consider the resources below:

Liao, C.C. (2007). *Programming and application of S7-200 plc* (3rd ed.). Mechanical Industry Press.

Mandal. R, Maity, T., Prasad, G.M., & Verma, R. P. (2015). Automation of underground coal mines using plc. *Journal of Mines, Metals, and Fuels*, 174 – 181. https://www.researchgate.net/publication/317038146_Automation_of_underground_coal_mines_using_PLC#:~:text=This%20paper%20presents%20applications%20of,flammable%20gases%20exceeds%20permissible%20limit

Wilcher. D. (2024, February 21). *Turn a raspberry pi into a plc using openplc*. https://control.com/technical-articles/turn-a-raspberry-pi-into-a-plc-using-openplc/

Course_Lab_project_code.zip folder: Github Repository: https://github.com/DWilcher/HCI_Electronics

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets