PLC-Based Swimming Pool System

The PoolOS Project

By
Dr. Alexander Pozhitkov, PhD

When Alexander wanted to automate the pump and heater for his swimming pool, he found it was a complex problem. His solution was to develop a program called PoolOS that runs on a PLC. The systems automate the operations of the heater and pump over a 24-hour period or for short periods on-demand.

The project described in this article serves as a guide for readers who want to learn about and experiment with PLCs (programmable logic controllers). It began with the need to run a pump and a heater periodically for a backyard swimming pool. The heater is a flow-through gas burner with an internal thermostat and safety controls. Initially I thought of it as a simple task. Why don't I just use a timer? However, the reality turned out more complex.

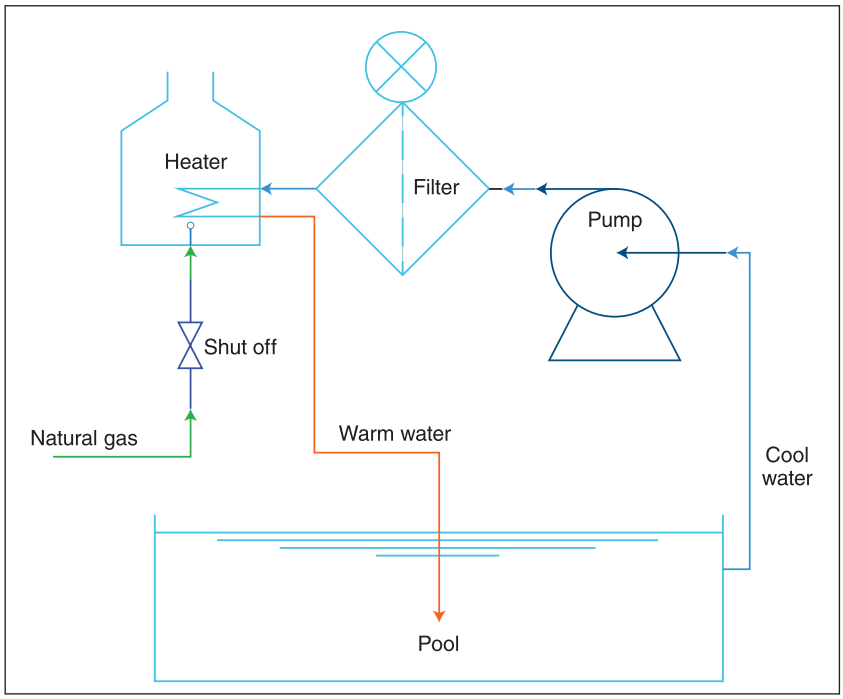


FIGURE 1
Diagram of pool heating and circulation process

The requirements for operation of the pump and the heater were:

1) Pump must circulate the water for 8 hours at night in a 24-hour cycle.
2) Heater turns on or off depending on the temperature.
3) Heater is allowed to heat only when the pump is running.
4) Pump must not stop if the burner is still active.
5) The operator must be able to activate the pump manually for a short period of time (about 20 minutes) during the off part of the cycle, to perform pool maintenance (mixing chlorine, vacuuming and so forth).

As you might guess, the requirements are not appropriate for a simple timer—even if the timer has multiple setpoints. Similar requirements actually are conceivable for a small chemical or fermentation plant! Fulfilling such requirements is thus a likely job for a microcontroller of some sort. I could have built a Microchip PIC MCU-based system. However, that would require all sorts of precautions for isolation, fault-proofing, housing and so on. Instead, I used an industrial PLC. The total cost of the PLC and its programming software is under $230. It is certainly a bit of an investment, but after all, the pool where you swim, the 240VAC pump and the gas burner should be treated more seriously than a breadboard-type project.

COMPONENTS

A diagram of the process is shown in **Figure 1**, with a photo of actual setup shown in **Figure 2**. Wires (14AWG) run through rigid

and flexible conduits. The enclosure housing the PLC is mounted on a strut that is secured into the ground with concrete. An inset in the Figure 2 photo shows the enclosure with its door open. The electrical schematic is shown in **Figure 3**. The schematic looks relatively simple—with all the heavy lifting done by the PLC. Physical grounding is not shown, but the pump, heater, conduits and housing are grounded (see busbar and green wires in the inset of Figure 2). Let's see how it operates and focus on the particulars of the heater operation before we look at the PLC program.

Heater: The heater is powered by a 240VAC source that needs to be constantly energized. Also, the heater features a "remote switch" (RS). When RS is ON, the control panel of the heater is enabled. The operator can allow or disallow heating and adjust temperature. When RS is OFF, the control panel is disabled, and the heater appears OFF. However, if the RS goes OFF while the heater is burning gas to heat the water, the control panel goes dark and the flame is extinguished, but the fan will continue blowing for several minutes to make sure the heater cools down. This, incidentally, is the reason why the heater cannot be energized or de-energized on the power-source level. It is important not to turn off the pump during such an event, because the heater needs to give off its residual heat to water. Also, the heater will not turn on if it does not detect adequate water pressure, that is, a running pump.

PLC: In this system, the PLC turns the pump and the heater RS ON or OFF. The PLC used for this project is the EASY-E4-UC-12RCX1 from Eaton. The datasheet [1] provides a detailed description of the unit. Briefly, the PLC features 8 inputs, 4 of which may be analog, and 4 SPST relays to control external circuitry. Also, the PLC has Ethernet and memory stick ports. The Ethernet port is used for communication and programming. The memory stick may be used to log the data and to boot the device (as an alternative to programming via Ethernet).

The PLC is powered by a doorbell transformer (available at Home Depot) and a stabilized waterproof power supply (a great variety of those is available on Amazon.com). Interestingly, the PLC takes a wide range of permissible power: 10.2VDC to 28.8VDC and 20.4VAC to 26.4VAC. Yes, you can feed AC to the PLC without problems. I thought of using the doorbell transformer to power the PLC directly. However, its nominal "24V" winding actually gave around 30V, which seemed risky. A tap from that winding, "16V," provided 18V, which clearly was not enough. Nothing's easy, as always! As for the pushbutton S1, when pressed it sets logic "1" on the input I02 of the PLC. In the PLC world even the AC voltage

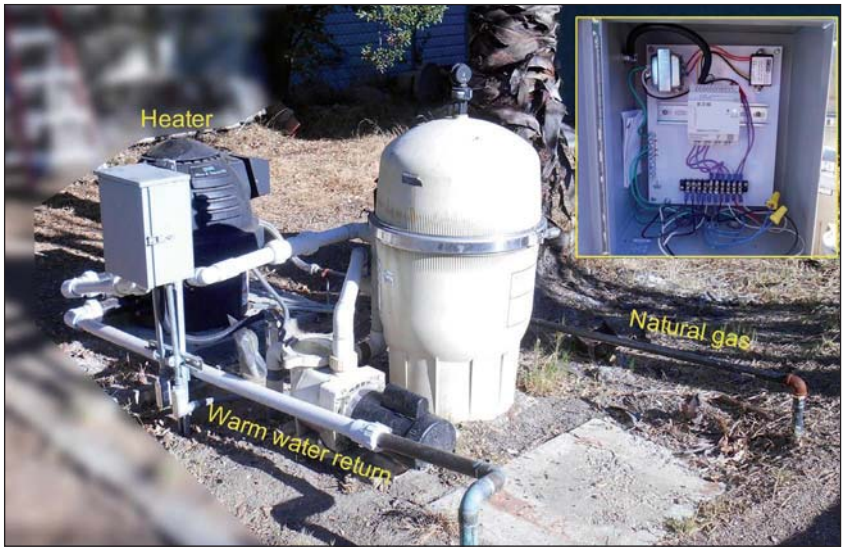


FIGURE 2
Actual implementation on the property. Inset outlined in yellow is the PLC enclosure with its door open.

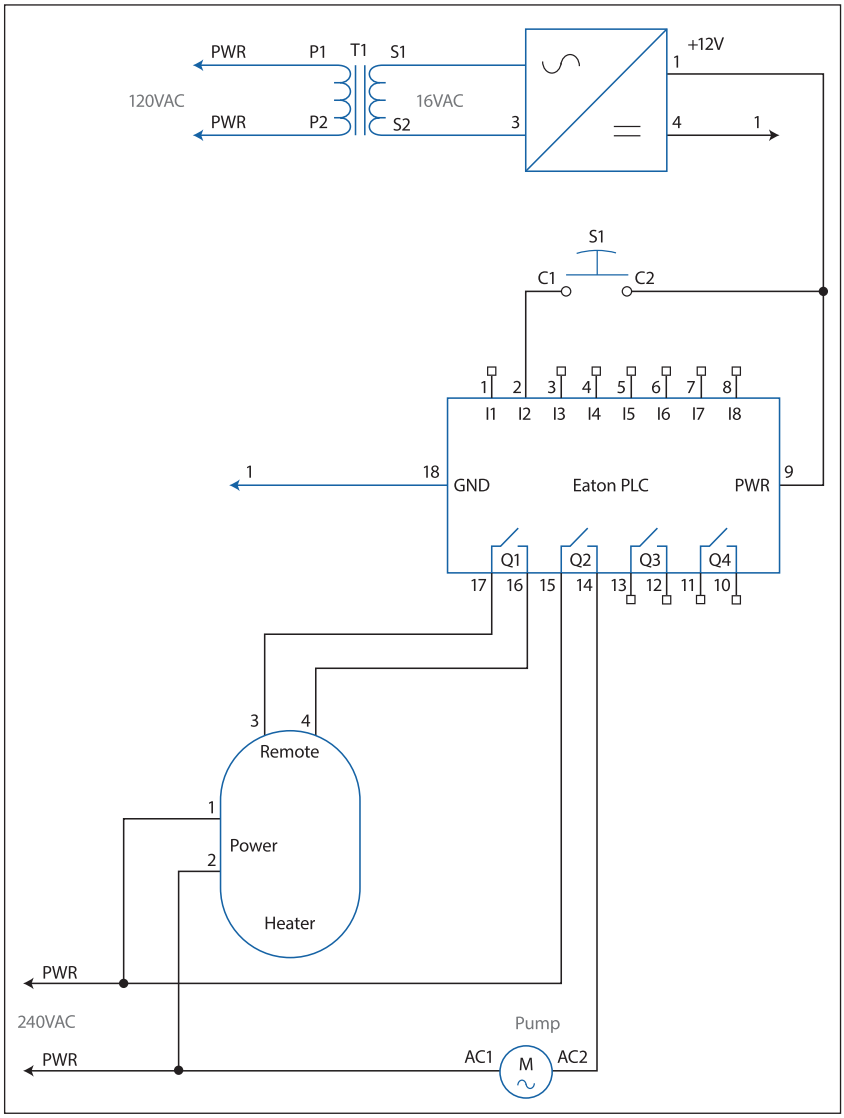


FIGURE 3
Schematic of PLC controlling heater and pump

24-Hour Cycle	Pushbutton S1 Pressed Manually
1. Turn on pump	
2. Turn on RS	1. Turn on pump, if off
3. Wait 8 hours	2. Keep RS unchanged
4. Turn off RS	3. Wait 20 minutes
5. Wait 15 minutes	4. Turn off pump if it was off initially
6. Turn off pump	

TABLE 1
PoolOS operating program

will be considered as logic "1" if the potential is substantially different from the GND.

PROGRAMMING

I named the operating system "PoolOS." Given the particulars of the heater operation, the program in **Table 1** needs to be implemented to fulfill the requirements indicated earlier.

The PLC was programmed with Eaton's easySoft 7 software. It is free to download and play with; however, a license is required to program the unit. The software supports all sorts of Eaton PLCs, not just the one described here. There are many YouTube tutorials on how to use it, and the manual under the *?* menu is well written. Here I will go through the PoolOS program to give readers the incentive to create their own programs. The PoolOS program code is provided on *Circuit Cellar's* code and files download webpage.

The PLC may be programmed in four different ways: 1) easy Device programming, 2) ladder diagram, 3) structured text, and 4) functional block diagram (FBD). After experimenting and playing around with these modes, the FBD seemed the most intuitive.

The screenshots in **Figure 4** show the entire program in two scrolled views. The left side of the screen shows the elements available for programming. Most notable are the *Contacts* and *Coils*. The *Contacts* represent either inputs of the PLC (for example, I01-I08), or numeric constants. *Coils* are either actual relays or *Markers*, which are internal variables or flags of the program.

As we look down the list on the left, under *Manufacturer function blocks*, all sorts of interesting things emerge, such as timers, signal filters and PID controllers. Our primary interest at this point is a timing relay, *T*. Selecting the relay and dragging it to the right side places it into *net 0001*. The nets or networks (having nothing to do with Ethernet) may be considered program lines, which the PLC is executing one by one.

Clicking on the relay block brings up a configuration sheet on the bottom of the screen (not shown here). The operating mode of T01—timer #1 – is configured as "flashing," which means certain time ON (8 hours), and certain time OFF (16 hours). The values for ON and OFF times are given as *Contacts* configured as constants connected to the I1 and I2 inputs of T01 (not to be confused with the PLC inputs). The input RE connected to I01 (actual PLC input) is to reset the timer. By the way, power cycling of the PLC will reset T01 as well. Setting "1" on I01 is done by temporarily connecting I01 to PWR. I did not implement an actual pushbutton for this function, since it is rarely needed. A constant "1" on input EN of T01 keeps the timer enabled permanently. Finally, the output Q1 of T01 is connected to the coils Q01 and M01. *Coil* Q01 is actually an SPST relay that controls the remote switch of the heater. M01, however, is a "marker" that receives the same signal as Q01 and will be used further in the program.

Thus far, we have controlled the heater, but it still requires circulating water. Here comes the *net 0002*, where the pump is taken care of. Another timer, T02, is configured as *off delayed*, which means that it will stay on for some time after its EN input goes low. Remember M01? It is still high from the *net 0001*, keeping T02 enabled. After T01 turns OFF and hence M01, T02 will be still active for 15 minutes, as determined by a constant on its I1 input. The output Q1 of T02 is connected to an OR gate, together with a signal from the marker M02. Both signals determine the state of the relay Q02, which operates the pump. You may ask why T01 and T02 could not be on the same net, and simply connected with appropriate "wires." The rules of the FBD prohibit such an arrangement, along with having more than one *Coil* on the same net. There are only a few rules for FBD, and they are clearly defined in the manual.

How is the state of the M02 defined? This is happening on *net 0003*. There is a third timer, T03, also operating as *off delayed*. It is momentarily enabled by raising "1" on I02 of the PLC (push button S1); then T03 stays on for 20 minutes, enabling the pump to run on demand.

DEBUGGING AND DEPLOYMENT

Once the program is prepared, it can be simulated to debug (see *Simulation* button, lower left side of Figure 4). You can define how

ABOUT THE AUTHOR

Dr. Alexander Pozhitkov has an MSc degree in Chemistry and a PhD in Genetics from Albertus Magnus University in Cologne, Germany. His expertise is interdisciplinary research involving molecular biology, physical chemistry, software, and electrical engineering. Alex has worked in academia (University of Washington and Planck Institute) and in the private sector (MidNite Solar). Currently, he is a researcher at the City of Hope medical center in Los Angeles, CA.

For detailed article references and additional resources go to:
www.circuitcellar.com/article-materials
Reference [1] as marked in the article can be found there.

RESOURCES

Eaton | www.eaton.com

Best Test & Control Products by Saelig

unique electronics

www.saelig.com

The Saelig Difference?
www.saelig.com/values.htm

Siglent: a global leader for electronic test and measurement instruments providing high accuracy, and dependable quality at value prices. Benchtop and Handheld Oscilloscopes, AWGs, RF Signal Generators, Power Supplies, Swept and Realtime Spectrum Analyzers, Network Analyzers, DMMs DC Loads.

Pico Technology: a world-leader in PC-based instrumentation, recognized for providing innovative, cost-effective alternatives to traditional benchtop test equipment and data acquisition products. Pico's USB-connected scopes are compact & lightweight - ideal for field work.

Rigol Technology: test instruments that compete equally with industry leaders but for a more affordable price. Digital and Mixed Signal Oscilloscopes, Swept and Realtime Spectrum Analyzers, RF Signal Generators, AWGs, DMMs and DC Loads.

Cambrionix: world leaders in intelligent USB charge and sync technology to enable smarter handling of mobile devices with optimal charging profiles and fast and remotely controllable software syncing.

ABI Electronics: a leading European supplier of test and measurement equipment for fast PCB fault-finding using logical test sequence software and mix-and-match test modules. Fast-to-learn, minimizes equipment downtime, fights obsolescence!

Saelig-Kerr: easy-to-use PIC-based motion control boards and ICs which integrate high level motion control functions and power amplifiers into a single board to provide everything needed to control a motor from a PC or other host controller.

Alan Lowne, CEO

"I founded Saelig in 1988 to search the world for unique electronic control components and test & measurement equipment, including: economical oscilloscopes, PC and RF spectrum analyzers, USB analyzers, AWGs, RF sources, DMMs, PSUs, data-loggers, industrial and panel PCs, EMI enclosures, USB serial converter cables and motor controls."

"Great team support, took care of my questions and treat me as a VIP customer." **JA**

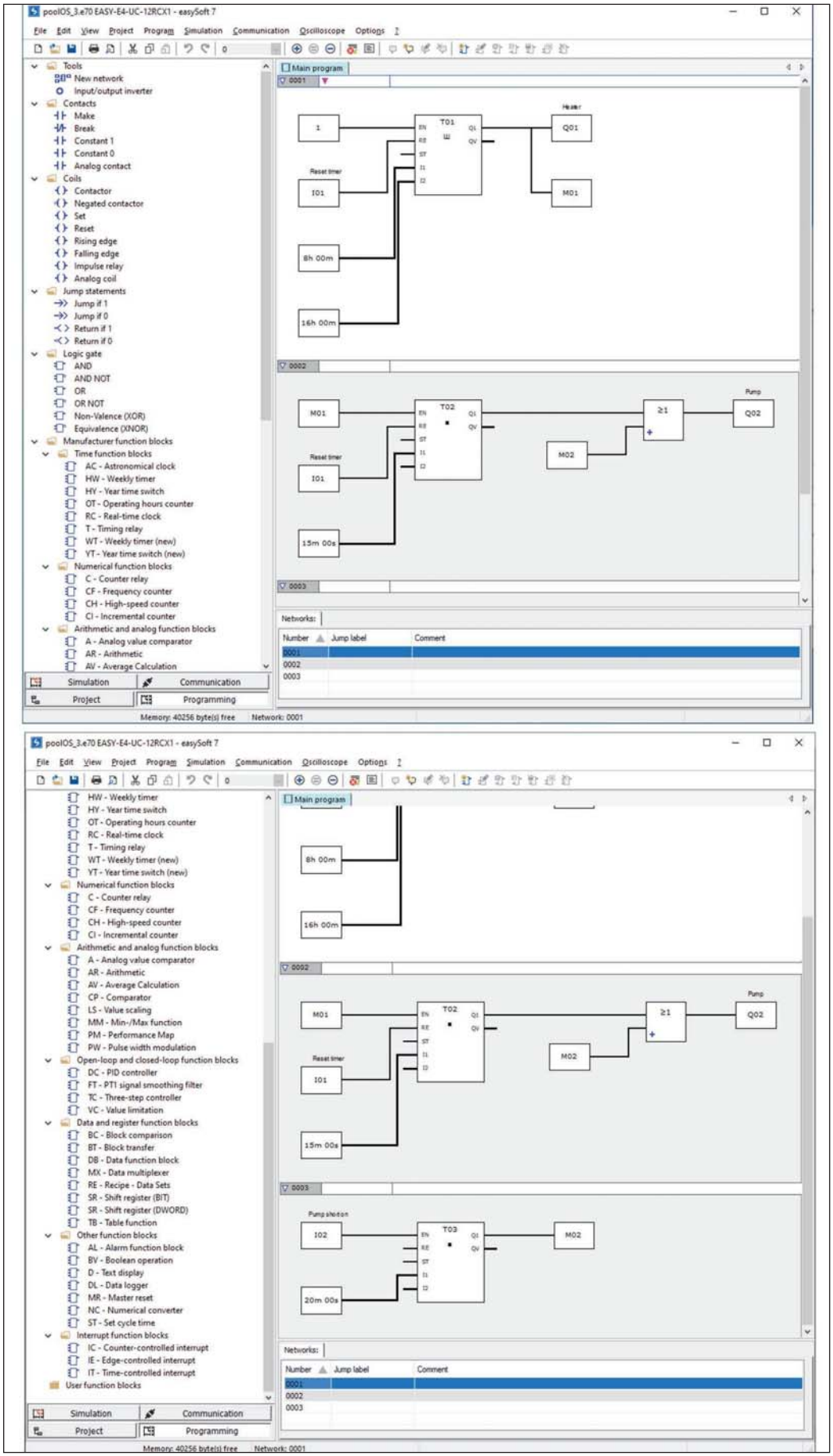
"Reading about the core values of your business impressed me." **RL**

"Nice website and well done on photos and specs." **KV**

"Accurate website and friendly sales people, not to mention the best prices around!" **BN**

• Saelig Company, Inc. • www.saelig.com • sales@saelig.com • 1-888-7SAELIG • (1-888-772-3544)

FIGURE 4
PoolOS program as functional block diagram, two scrolled views



long the "hour" lasts during the simulation process. The PoolOS was simulated with "1 hour" lasting 1 minute. Hence, the entire process was run through within 24 minutes *in silico* (in computer simulation). Throughout the simulation, the state of the timers, inputs and outputs may be monitored, and the PLC inputs may be set to "1" at arbitrary times.

When the program is found to be working satisfactorily, it must be uploaded into the PLC. The easiest way is to directly connect the PLC to a PC with an Ethernet cable. Do not attempt connecting the PLC to a hub, a router or a PC with bridged Ethernet adapters. The reason is that a brand-new PLC will not be able to communicate in an existing subnet, because it has its own IP address that is unlikely to match your subnet. Instead, when a PC with an Ethernet adaptor that has no configuration "sees" a PLC, it will autoconfigure itself with an IP address to establish a subnet of two devices: PC and PLC.

Click the *Communication* button to bring up the interface shown in **Figure 5**. First click *Search* to find your PLC, and save its profile. After that, click *Online* to establish communication. With communication established, many parameters may be changed, including the network configuration, debouncing, time and date. Most importantly, the program may be uploaded into the PLC, and the unit may be started. While communication is established, the status of the PLC and the program may be monitored in real time. Once everything is completed, click *Offline* and unplug the Ethernet cable.

Unleash the power of PLC for your next project!

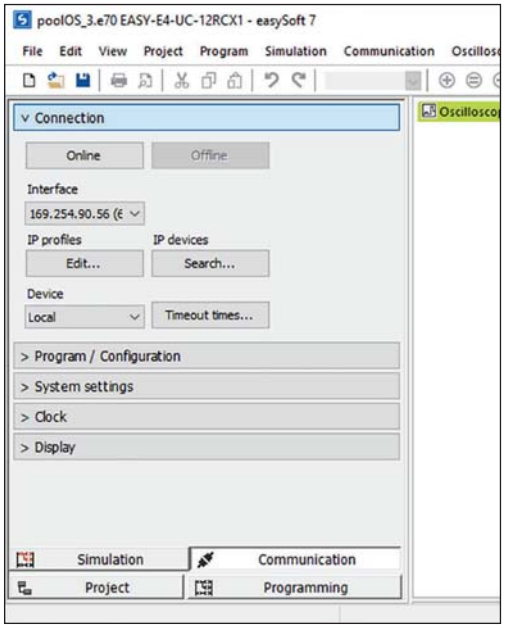


FIGURE 5
PLC-PC communication interface