Benchmarking of OpenPLC Runtime on C-DAC VEGA RISC-V Microcontroller

Vinukumar AppukuttanNair Retnakumari[1], Premjith Achem Veettil[2], Sivakumaran Natarajan[3] and Sri Ram Shankar Rajadurai[4]

[1]Department of Instrumentation and Control Engineering, National Institute of Technology, Tiruchirappalli, Tamil Nadu, India
[2]Hardware Design Group, Center for Development of Advanced Computing, Thiruvananthapuram, Kerala, India
[3]Department of Instrumentation and Control Engineering, National Institute of Technology, Tiruchirappalli, Tamil Nadu, India
[4]Department of Instrumentation and Control Engineering, National Institute of Technology, Tiruchirappalli, Tamil Nadu, India

Correspondence to:
Vinukumar AppukuttanNair Retnakumari,
vinu@cdac.in

Additional material is published online only. To view please visit the journal online.

Cite this as: Retnakumari VA, Veettil PA, N Sivakumaran and Rajadurai SRS. Benchmarking of OpenPLC Runtime on C-DAC VEGA RISC-V Microcontroller. Premier Journal of Science 2025;15:100135

DOI: https://doi.org/10.70389/PJS.100135

Peer Review

Received: 14 August 2025
Last revised: 25 September 2025
Accepted: 29 September 2025
Version accepted: 3
Published: 1 December 2025

ABSTRACT

Modern industrial automation systems are no longer restricted to proprietary Programmable Logic Controllers (PLC). They can be designed, developed, and commissioned on inexpensive yet powerful open-source microcontroller hardware. Open-source control software and communication protocols used in such systems provide increased protection against cyberattacks compared to proprietary controllers.[1–3] OpenPLC[1] is an open-source programming environment developed by Autonomy Logic that provides a platform for deploying industrial automation codes on hardware from different vendors. This work demonstrates the testing and benchmarking of OpenPLC functional blocks, ladder diagrams, etc., on common commercially available microcontroller boards and particularly on the indigenously developed C-DAC ARIES v2 microcontroller board with the THEJAS32 VEGA processor as a PLC. The RISC-V Processor Core and instruction set that run on the VEGA processor support a wide range of applications, including AI-based real-time process automation solutions.[4] This work leads to the establishment of a solution framework powered by modular hardware with support for software that uses an extendable RISC-V Instruction Set to build more optimized industrial control system applications[5–7] using Ladder Logic. This work demonstrates complex tasks such as fast data movement within internal memory, matrix multiplications, and crypto-computations in real-time on the microcontroller hardware, thus conveying that the solution is even capable of hosting an edge-level AI system for modern-day industrial applications. The tasks involved include a benchmarking algorithm that is executed on a candidate microcontroller hardware board. In addition to bare benchmarking using a C-code emulator for the respective boards, an extended version of the benchmarking was also carried out by encapsulating the benchmark algorithm as a PLC Ladder block. The benchmarking block helps control system designers select the right microcontroller unit for automation controllers at the field level, considering the availability of boards, cost, performance, security, and ease of integration.

Keywords: OpenPLC benchmarking, VEGA ARIES V2 board, RISC-V THEJAS32 processor, Ladder logic performance, Coremark-inspired MCU evaluation

Introduction

Programmable logic controllers (PLCs) have dominated the field of industrial automation in the past several decades as a successor to relay logic circuits. However, the proprietary and relatively expensive nature of PLC controller hardware has hindered automation in many industrial applications.[1–4] OpenPLC[1] is an open-source programming environment developed by AutonomyLogic that provides a platform for deploying industrial automation codes on hardware from different vendors. This study focuses on the creation of a low-cost system based on OpenPLC, which is comparable to the proprietary versions already used in industrial automation. The software framework of OpenPLC is known for its modular, simplified architecture, and expansion capabilities. OpenPLC comprises an integrated development environment (IDE) that is made of OpenPLC Editor, wherein the user can define the logic in any of the five programming languages compliant with the IEC 61131-3 standard[9,12] and an OpenPLC Runtime application, a software suite that renders the user's code to be compiled to run on Open Hardware,[8,10,12] a term used here to collectively describe popular non-proprietary embedded microcontroller boards. Figure 1 shows a schematic of the five programming possibilities on the OpenPLC Editor compliant with the IEC 61131-3 standard that can be compiled using the Matiec Compiler and run on the Open Hardware.

This work focuses on the development and demonstration of OpenPLC compatibility on the VEGA ARIES board[5–7] indigenously developed by C-DAC, based on the THEJAS32 SoC,[21] while scripting and deploying the automation logic using OpenPLC Editor and Runtime software packages. A custom benchmarking algorithm was executed on the VEGA ARIES board to compare its performance with other microcontroller hardware boards. Security is a major concern for modern control systems. However, security discussions were excluded from the implemented benchmarking block.

Related-Work

OpenPLC has also been extensively used in cybersecurity research. In the last few years, there have been many reports of cyber-attacks on industrial control systems, sabotaging the normal operation of SCADA and PLC systems.[12,13] Several defense mechanisms can be employed to protect critical assets from attacks. However, the proprietary nature of automation products from standard vendors that use custom microcontroller hardware and firmware to deploy controllers has significantly limited research and solution development for improving cybersecurity measures.[12]

The Board support package (BSP) for PLC controllers is a software that includes device drivers for digital and analog I/O channels and various serial communication modules. In proprietary PLC systems, such software is integrated into the runtime module of a specific PLC. This renders the design of commercial PLC hardware closed and inaccessible to system integrators. However, the design of open-source hardware breaks the barrier of the specific and private nature of system design

Ethical approval: N/a
Consent: N/a
Funding: No industry funding
Conflicts of interest: N/a
Author contribution: Vinukumar AppukuttanNair Retnakumari, Premjith Achem Veettil, Sivakumaran Natarajan and Sri Ram Shankar Rajadurai – Conceptualization, Writing – original draft, review and editing
Guarantor: Vinukumar AppukuttanNair Retnakumari
Provenance and peer-review: Unsolicited and externally peer-reviewed
Data availability statement: N/a

and enables better usage of the system through open licensing.

The Arduino family of microcontroller boards is open-source hardware that is programmed using the open-source programming environment Arduino IDE. When used in the implementation of real-time control systems,[3] these boards have several advantages: small size, lower power rating, faster development cycle, and better maintenance owing to the support of the open-source community. These boards can also be used for telemetry applications owing to their reliability and robustness in communication,[11] and OpenPLC ensures the compatibility of these boards for conventional industrial automation applications.

With the advent of Industry 5.0, industrial controllers must integrate more advanced computation capabilities for cloud computation support, communication through complex networks, and multi-processor designs.[10] This demands a newer generation of Arduino-like open-hardware and open-source software, where OpenPLC provides a user-friendly development environment for control and automation.[12] Porting the OpenPLC software framework[17,18] on commercial PLCs and benchmarking the performance is also possible. However, the focus of this study is limited to the implementation of the benchmarking block in Arduino-like open hardware.[19,20]

RISC-V (Reduced Instruction Set Computer Architecture – 5th version) is an open-source instruction set that conforms to modularity and extensibility with custom extensions[14–16] of the instruction set. C-DAC VEGA Processors[21] are the first of their kind developed in India with RISC-V support on silicon with a unique compiler and application development environment. In terms of computing efficiency, compactness, and timing aspects, ARIES boards can be compared to Arduino UNO, ESP-32, and STM-32 boards.[5] CAN Bus communications is also demonstrated with VEGA Processor in industrial applications.[7] The board support package provided by the VEGA system developers is fully compatible with the Arduino IDE software, including its command line extension called "arduino-cli." This enables the VEGA ARIES board to port the OpenPLC Runtime. Because the benchmarking target is the RISC-V microcontroller and the work is focused on limited 5 types of boards, which could be enhanced with contemporary boards such as Raspberry Pi, this will be handled in future work. The compliance, interrupt / debug interface support and energy / performance trade-offs of the open-source RISC-V cores (some of them being BOOM, Xiangshan, XuanTie C910, and CVA6S+) have been discussed by Fu et al.[25] They disclose in their work that the high-performance cores are undergoing some catch-up process however; there persist toolchain and verification gaps. Valente et al., has expounded on real-time predictability as well as open IP to RISC-V SoCs that model latency limits, predictable interconnects, and shared memory in heterogeneous RISC-V designs.[25] Niauronis et al., compares the parameters of the speed / execution time between various PLC hardware, including both open-source embedded controllers (Arduino, ESP8266) under OpenPLC, a commercial PLC (WAGO PFC200). It offers a point of comparison in the ability of constrained hardware to deal with IEC 61131 workloads.[24] In Chmiel et al.,[26] the blocks of timer functionality are used as requirements of IEC 61131-3, and implemented in FPGA hardware, further reporting on performance measurement, and hardware analysis of implementation costs and performance at run-time.[26]

This work describes the OpenPLC-VEGA test setup and the algorithm used to benchmark its performance. This work discusses the development efforts to ensure the compatibility of VEGA Processors to the OpenPLC environment and the development of PLC benchmarking logic blocks. Further sections discuss the results of testing the custom process logic and enabling Modbus

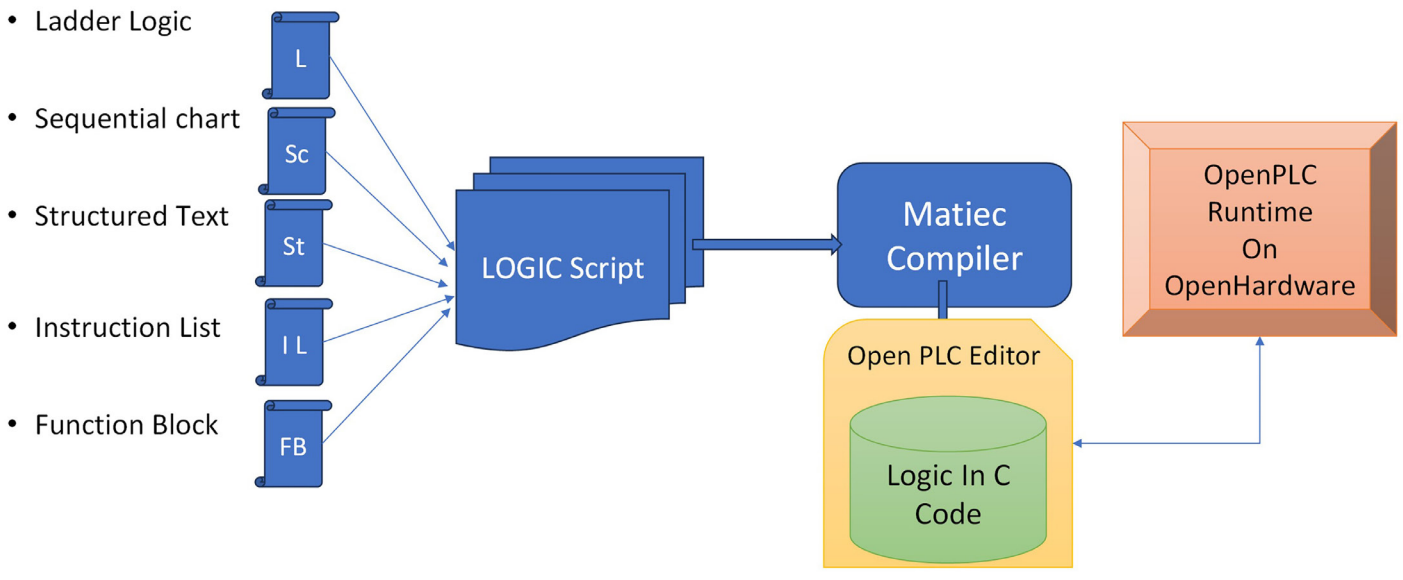

Fig 1 | OpenPLC Workflow showing IEC61131-3 compatible programming languages and software getting deployed on OpenHardware

communication in the developed setup. The execution results of the logic on various microcontroller hardware platforms are presented. Finally, the conclusions are presented.

Description of Setup

Establishment of VEGA ARIES V2 Board as OpenPLC Hardware

The setup for deploying the OpenPLC Runtime on the VEGA ARIES board consisted of a Personal Computer with Microsoft Windows Operating System acting as the engineering station. The system needs to be installed with software for Arduino-IDE and OpenPLC Editor.[17,19] The board support package for the VEGA ARIES v2 board can be downloaded[20] and installed using the Arduino IDE through the Boards Manager interface of the Arduino IDE. The VEGA ARIES board can now be connected to the USB port of the workstation. Any of the available Arduino Sketch files from the examples menu can be opened, compiled, and uploaded to ensure smooth operation of the setup. The steps described in the following sections need to be carried out to establish the VEGA ARIES board as PLC hardware and to detect the board in OpenPLC. The hardware I/O architecture of the ARIES board carrying the THEJAS32 SoC is illustrated in Figure 2. The signals to be identified from ARIES board are listed in Table 1.

The power requirements for the board are as follows: The Input Voltage required for the board is 7-12V, DC. The Current per I/O Pin is 12 mA, and the IO Voltage is 3.3 V. The memory footprint is 256 KB SRAM. The code-size metrics are highly dependent on the application, OpenPLC framework, Arduino software framework, BSP of OpenPLC, and Arduino BSP for the VEGA processor.

Modification of HALS Registry

To make the OpenPLC Editor detect the ARIES board, an appropriate entry should be inserted in the OpenPLC HALS file in the path "OpenPLC\Arduino." Insertion of VEGA ARIES board details as a new board entry can be carried out through the 5 steps listed below:

- Mapping of Field I/O driver,
- Setting of Arduino Board package reference,
- Reference for last hardware board package update,
- Setting of Arduino package core reference, and,
- Definition/mapping of the SCADA field signal type supported by the board for field operations.

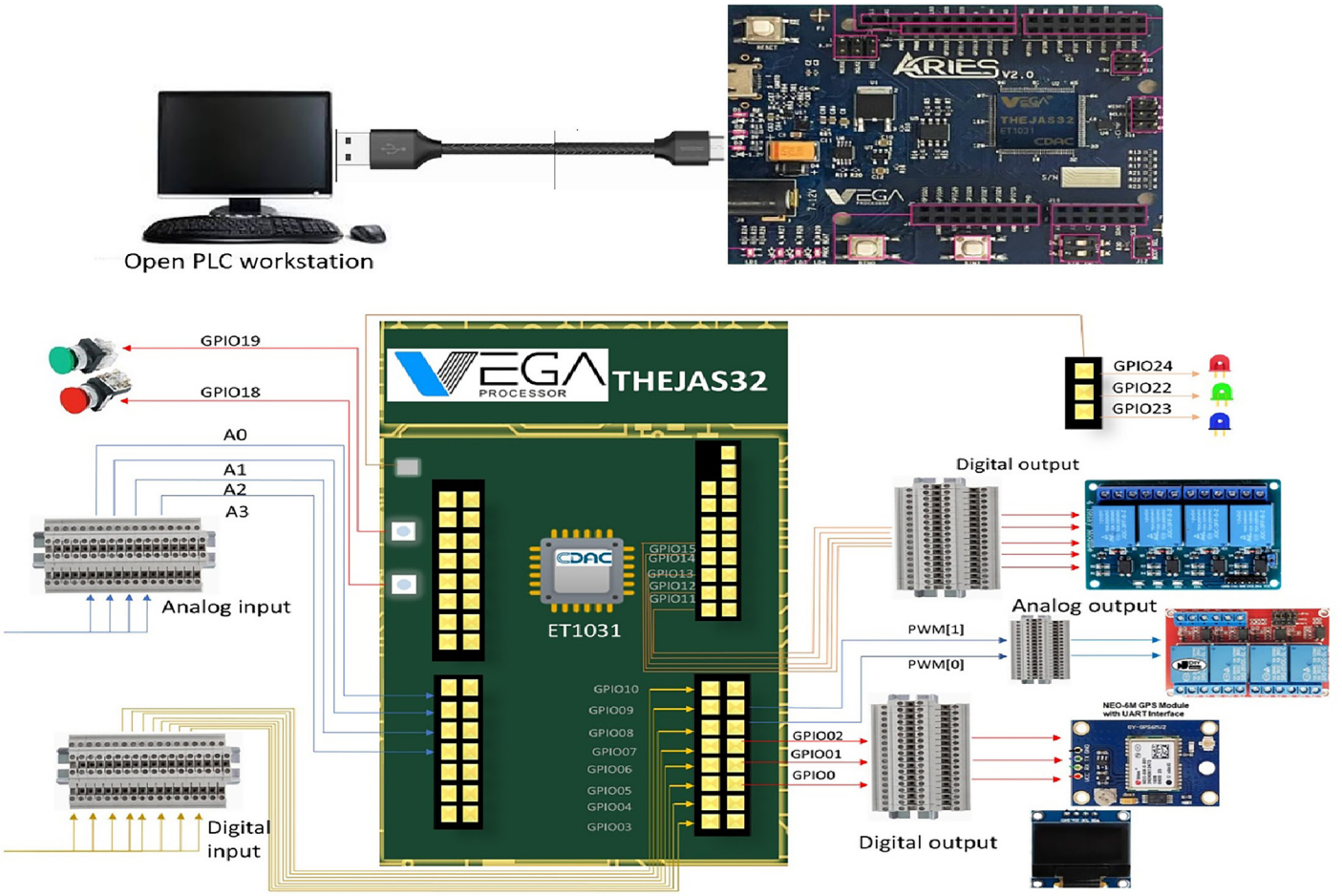

Fig 2 | Schematic describing the peripheral devices and the utilization of ARIES board containing the THEJAS32 SoC as a PLC Hardware board interfaced to a workstation using USB

Development of ARIES I/O Driver for OpenPLC

The board support package (BSP) framework for OpenPLC requires the signals identified in Table 1 to be processed in the software for data acquisition functions. The driver functions were implemented such that each function was encapsulated with a call to the corresponding Arduino API function. OpenPLC defines the following functions: (i) hardwareInit, (ii) updateInputBuffers, and (iii) updateOutputBuffers. Figure 3 shows the configuration process of the OpenPLC with the VEGA BSP. The metadata file for the ARIES board, along with the relevant pinout configuration, is discussed in our previous work.[22]

Building Process Logic Using OpenPLC

The process logic defined by the user in the OpenPLC Editor in the workstation can then be compiled to run on the engineering workstation as a standalone system. Furthermore, it can be built for a target embedded board, such as the "C-DAC ARIES v2" board. The UART controller on the VEGA board is enabled as a serial port that works over the Universal Serial Bus (USB) interface, which is connected to the workstation. The OpenPLC Editor further communicates with the board via a USB interface. The process logic, defined as a ladder diagram in graphical format, is first converted to the OpenPLC IEC 61131 Structured Text (ST) script (Figure 1). An equivalent C file for the script was generated using the matiec compiler. The C code was compiled using tools linked to the Arduino IDE C compiler. The command-line variant of the Arduino IDE coordinates the compilation process in OpenPLC. The steps for generating the binary code for the target PLC hardware are as follows:

1. Defining logic in OpenPLC Editor,
2. Compiling PLC script and building binary code,
3. Uploading binary code to VEGA hardware,
4. Execution of the binary code.

Benchmarking Bare Hardware for Arduino Boards

The hardware boards selected for benchmarking were Arduino Nano, Arduino Uno, ESP 32, ESP 8266, STM32 Bluepill, and C-DAC ARIES V2. It can be noted that all the microcontroller units (MCU) in the selected boards are compatible with the Arduino IDE. The benchmarking methodology adopted in this study is similar to the CoreMark methodology.[23] However, this study aims to pack the algorithm within a PLC Ladder Logic function block. The benchmarking methods adopted here consist of tasks such as linked list creation,

Table 1 | Signal mapping of ARIES board with OpenPLC hardware

Sl. No.	Signal Details		
	PLC Signal Type	Signal Count	ARIES Signal Interface Type
1	Digital Input	8	GPIO
2	Push button switches for debugging	2	GPIO
3	Digital Output pins	8	GPIO
4	Debug LEDs	3	GPIO
5	Analog Input pins	4	Analog inputs
6	Analog output channels	2	PWM channels

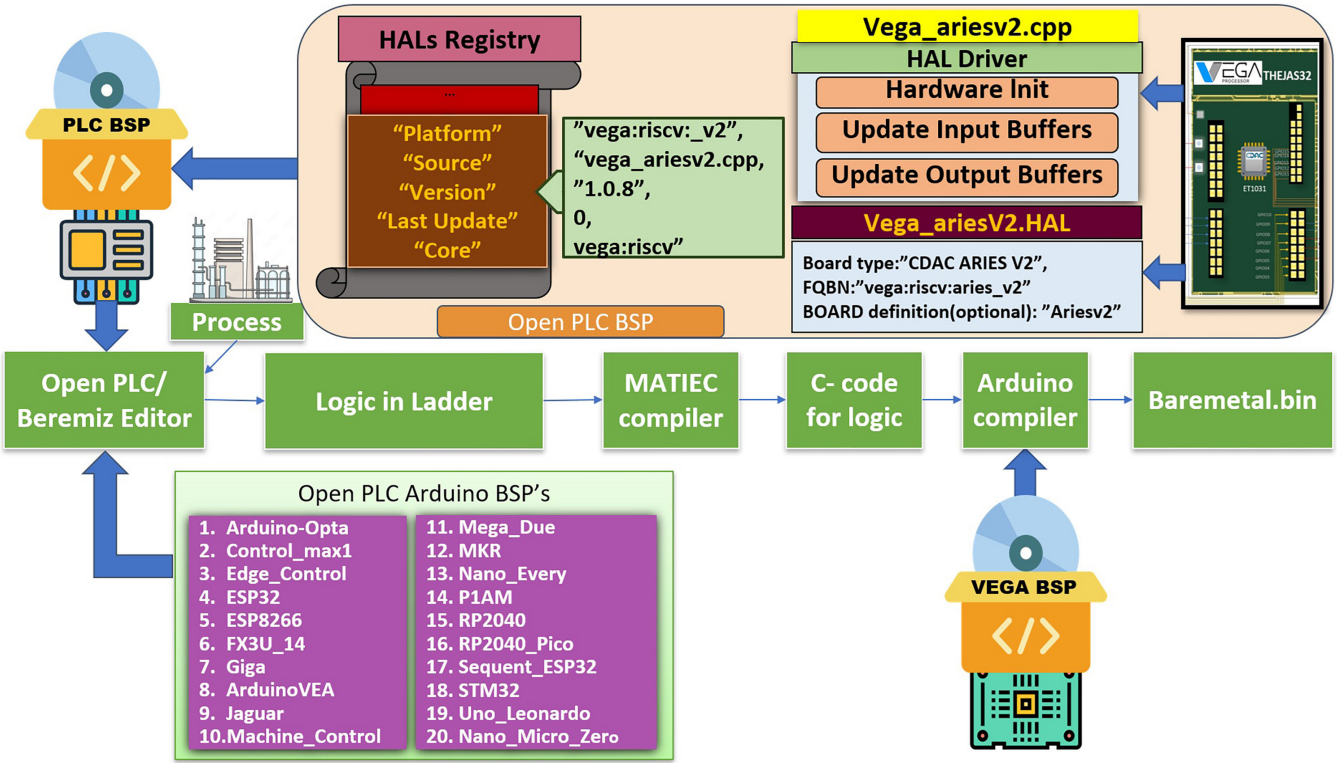

Fig 3 | The PLC configuration process and the addition of VEGA ARIES software packages to the OpenPLC VEGA Arduino BSP along with the necessary updates in the HAL registry

DOI: https://doi.org/10.70389/PJS.100135 | Premier Journal of Science 2025;15:100135

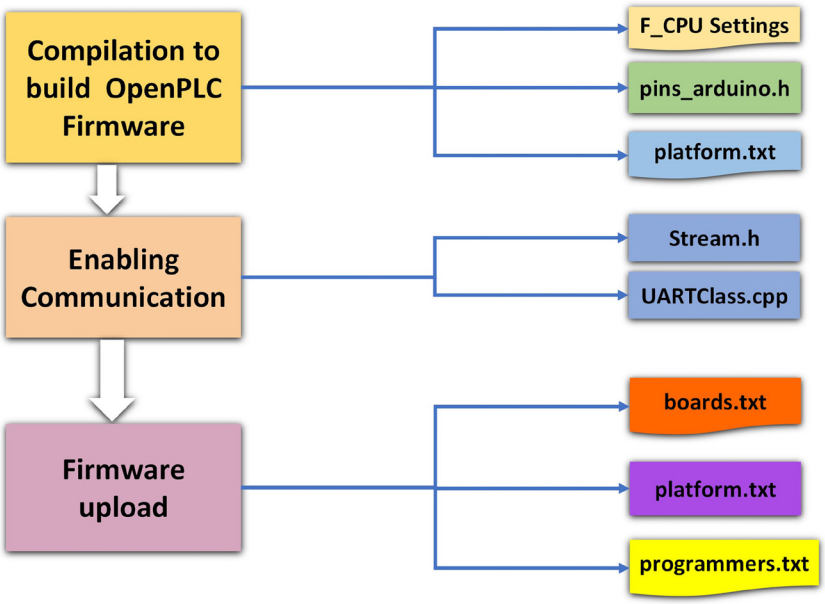

Fig 4 | Schematic diagram of the relationship between necessary files modified for the compilation of OpenPLC program in VEGA-Arduino package and its upload to ARIES board

reversal, matrix multiplication, and state machine traversal. These computational models are often used in many applications, including those based on real-time AI. To perform a rational benchmark, the benchmarking code was first applied to run on the bare MCU using the Arduino IDE. In addition, an extended variant was created by implementing the benchmarking algorithm as a PLC function block to run from the ladder logic. The challenge was that PLC blocks normally cannot have a very extended working time. Hence, the benchmarking algorithms were executed as split chunks, where timers were tuned to collect benchmark timing strictly when the benchmark algorithms ran on the MCUs.

Development

The compilation of a sample ladder logic was carried out as mentioned in the previous sections. The software framework used for the compilation was Arduino Version 2.3.4 with GCC compiler. The hardware abstraction layer package provided by the CDAC was version 1.0.8. The compilation process resulted in a few compilation errors, which necessitated corrections in the software. To enable OpenPLC Runtime to execute on the VEGA hardware for the first time, the following changes were incorporated in the VEGA-Arduino package: (i) changes in the compile and build configurations, (ii) enabling communication, and (iii) firmware upload. These changes resulted in the release of the updated 'vega-arduino package 1.0.8' as part of the Arduino IDE. These changes were made based on the previous 'vega-arduino package 1.0.7'.

The OpenPLC Editor command "Transfer to PLC" was executed. Upon applying the listed changes, by selecting the board with the specified package version and the serial port for firmware download, the compilation was enabled for the first time. The process of enabling the OpenPLC Runtime can be symbolically presented as a flow diagram, as shown in Figure 4.

Changes in Compile and Build Configurations

The CPU frequency must be configured to compile the OpenPLC source code. In addition, the "regtype" typecast used for handling the pins in register sets of VEGA was enabled in the file "pins_arduino.h". For various ladder and function blocks related to the process logic implementation, OpenPLC generates a structured text file and the corresponding C source code file. The C files were further compiled using "gcc compiler" set from the Arduino platform specification file called "platform.txt". The C and C++ compiler flags were set to suppress 'no risk' warnings to create a clean code compile. These modifications enabled a clean compilation of the OpenPLC source code by creating binary files and subsequently the 'Baremetal.ino' sketch file.

Enabling Communication

The Universal Serial Bus (USB) serial communication interface of the ARIES board was used for both terminal operations and raw data communications using the Universal Asynchronous Receiver-Transmitter (UART) interface of the VEGA processor. On the Arduino BSP for VEGA processors, there is a driver file that contains the details for enabling and updating the UART functionality for various fieldbus protocols, including the Modbus protocol. A file named "stream.h" of the VEGA-Arduino package was modified so that the underlying UART driver could provide the data characters received in the UART buffer at various stages of data exchange in the communication layer of the application.

Firmware Upload

To ensure that the control logic is set in a PLC controller persistently, even after a reset/reboot of the controller, the flash memory storage of the ARIES board was used for the storage of PLC program data. The 'flash support' in the boot loader program for the ARIES board communicates with an application called 'vegaflasher' that uses serial communication protocols. To flash the control logic on the VEGA processor, the following changes were made to the VEGA-Arduino package file to configure the code upload process:

1. Adding new entries to support different upload methods and protocols in 'boards.txt' file,
2. Configuring details of vegaflasher in 'platform.txt' file,
3. Setting Serial protocol details for vegaflasher and
4. Setting the default Flash mode for the vegaflasher in the 'programmers.txt' file.

Design, Development of Benchmarking Software on Arduino IDE

The benchmarking of the MCU core was carried out by running the 'barebenchmarking.ino' Arduino Sketch file from the Arduino IDE.

The Arduino packages for the MCU boards selected for benchmarking were installed in the Arduino IDE on

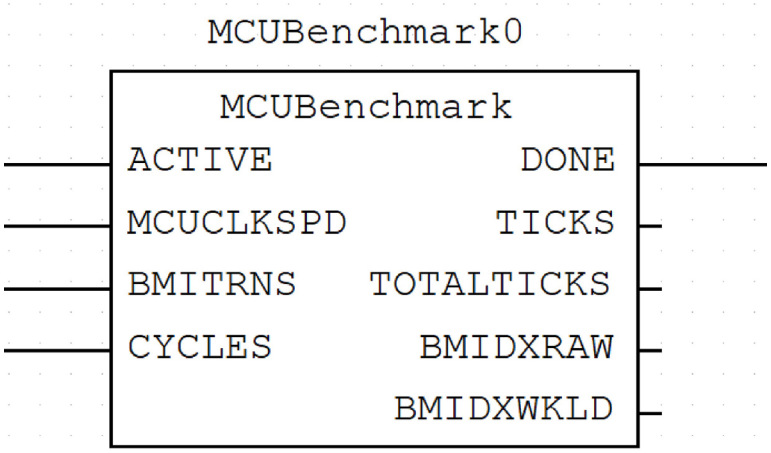

Fig 5 | The ladder logic used for testing PLC functionality

Table 2 | Arduino packages for MCU boards to be benchmarked

Sl. No.	Board Name	Package Name
1	CDAC VEGA ARIES V2	VEGA ARIES Boards By CDAC
2	ARDUINO UNO	Arduino AVR boards
3	ARDUINO NANO	Arduino AVR boards
4	ESP 32	ESP32-DOIT DEVKIT V1
5	NODE MCU	ESP8266 Node MCU
6	STM32 BluePill	STM32 MCU based board (BLUEPILL_F103C8)

the workstation. Benchmarking was performed using Arduino version 2.3.4. To complete the benchmarking, the boards listed in Table 2 were connected to the workstation. The benchmarking program written in C was compiled using the Arduino IDE. The benchmarking algorithm executed iterations for 20000 cycles, calculated the benchmark index normalized with the clock frequency of the processor (per MHz), displayed the benchmark value, and finally exited the software.

Design, Development of Benchmarking Block for OpenPLC

Although the bare benchmarking performed using the Arduino IDE validated the performance of the MCU, applications based on Ladder logic blocks embedded with scripts, diagrams, or functions could impose an additional computational load on the PLC hardware. Therefore, to verify the performance of the automation controller with PLC ladder logic, the PLC logic was viewed as an application on an MCU, and application-level benchmarking was evaluated. Therefore, the bare benchmarking logic was split into subfunctions and implemented as function blocks for the ladder diagram. The design of the benchmarking function block is shown in Figure 5.

The Coremarking Algorithm, outlined as a flowchart in Figure 6, was implemented as a 4-input, 5-output MCU Benchmarking block in the Ladder Logic program. The input signal ACTIVE enables the execution of the benchmarking logic. The MCUCLKSPD parameter (or f_{CLK} in equation (4)) is the MCU clock speed in MHz. This parameter enables the evaluation of performance per clock speed (per MHz). The BMITRNS parameter (or $N_{iter-cycl}$ in eq. (2)) is the number of iterations of the core marking algorithm to cover in one block execution. If the block waits for the benchmarking to be completed, then the state machine load imposed by various blocks cannot be identified accurately. The input parameter CYCLES ensures that the benchmark calculation is performed and updated at the end of each cycle. The five output signals include DONE, which is a Boolean function that toggles on the completion of each iteration. TICKS (or N_{TK} in eq. (1)) is the number of CLOCK TICKS taken in milliseconds for the execution of one cycle of benchmarking iterations. TOTALTICKS is the clock tick counted in milliseconds since the start of the PLC in the execution of the number of benchmarking iterations specified by the product of the CYCLES and BMITRNS parameters. The output signal BMIDXRAW (or i_{BM} in eq. (4)) is the Raw Benchmark Index and BMIDXWKLD is the Benchmark index calculated for the additional PLC functionality orchestration load injected by the OpenPLC based Ladder Logic framework. It can be concluded that a processor that shows a higher benchmark index can be considered faster for the given clock speed.

The expressions used in the evaluation of the Benchmark Index parameters are shown in equations 1–4.

$$T_i = N_{TK}/1000 \quad (1)$$

$$N_{1s} = N_{iter-cycl}/T_i \quad (2)$$

$$i_{iter} = round(N_{1s}) \quad (3)$$

$$i_{BM} = i_{iter}/f_{CLK} \quad (4)$$

Here T_i is the total time (in seconds) taken to execute one cycle consisting of $N_{iter-cycl}$ iterations of Benchmarking block. It is obtained from N_{TK}, the number of Ticks (in milliseconds) counted during execution of one cycle of iterations. N_{1s} is the iterations performed in 1 second.

The Iteration Index i_{iter} is the rounded figure of N_{1s} to the nearest integer. The overall Benchmarking Index i_{BM} is the ratio of the Iteration index normalized with respect to the clock frequency f_{CLK} (in MHz) of the processor.

Modifying OpenPLC to Hold MCU Benchmark Block

To insert the MCU Benchmark block into OpenPLC, six new files were added at various file locations, as listed in Table 3. The file "MCU_BenchMarkLib.xml" encapsulates the visual representation of the MCU benchmark block to be added to the list of libraries from which the PLC programmer can drag the block to the logic editor. The files "plc_cpu_bm*" is for the simulation to run on the Workstation and the "plc_mcu_bm*" are for the logic to run on the Arduino board, which is the essence of this work. In the case of the Arduino variant, for better organization in the OpenPLC application, we created unique files in the Modules' directory of the OpenPLC Editor to encapsulate private C logic related to benchmarking. The file with the extension "*. h" contains the block data structure definition, init code definition, and block

DOI: https://doi.org/10.70389/PJS.100135 | Premier Journal of Science 2025;15:100135

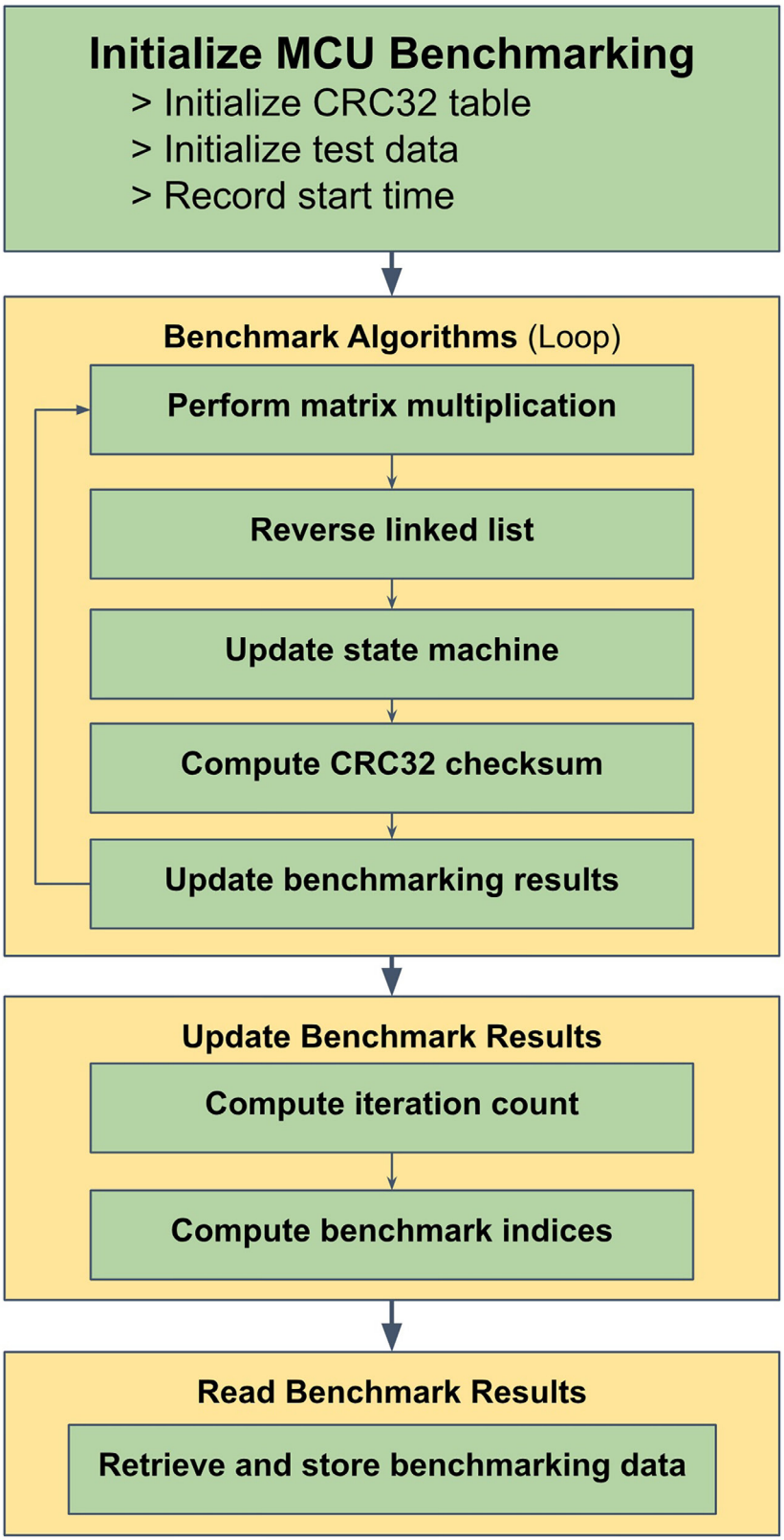

Fig 6 | Flowchart for the OpenPLC benchmarking algorithm which is developed in both to run directly on the specimen boards and embedded as a block in the OpenPLC software

logic body definition. To ensure the correctness of the logic, the block was first executed in the simulator on the workstation before being run on the Arduino. The simulation can be performed on either Windows or Linux computers. In addition to the creation of the six files mentioned in Table 3, to make the block run, the reference of the block had to be registered in files such as "definitions.py," appropriate files in accompanying folders for files such as "Standard_FB.txt" (which is a master structure text file for all blocks), "iec_std_fb.h" (which is the master C code abstraction for all PLC blocks), etc. An undocumented information of enabling macro "USE_ARDBMRKBLOCK" was also done to include the file "ardmcubmrk.c". With this, the OpenPLC Editor displayed the block "MCUBenchmarking." The block could be added to the logic, and the system was compiled for PLC boards such as CDAC VEGA, ESP32, and NODE MCU. Figure 7 shows the leaf nodes of the tree structure where the files of the OpenPLC Editor were installed, the folders to be used, and the files to be modified.

Testing and Results

Testing of Ladder Logic on ARIES v2 Through OpenPLC Runtime

After applying the changes in both OpenPLC and VEGA Arduino package files, the compilation process was successful, and the target binary file was created. As part of the last stage of OpenPLC testing for the ARIES board, the binary image of the PLC code had to be uploaded to the flasher chip of the board, which could be carried out using the "Transfer To PLC" command available in the Download menu/dialog of the OpenPLC Editor. The firmware was uploaded using the "vegaflasher" tool, which was integrated into the VEGA-Arduino package folder. Once the image file was uploaded, the bootloader started and shifted the execution control to the PLC code. To verify that the PLC code was executed by the ARIES board, it was tested with a simple LED blinking program, as shown in Figure 8, and Modbus communication. The logic consisted of two timer blocks: an ON-delay timer and an OFF-delay timer operating at 20 ms intervals interfaced to a debug blinking LED. The timer blocks in the rung operate in such a way as to obtain a toggled operation in each cycle. The logic was successfully tested using the OpenPLC Runtime. The total ecosystem of the OpenPLC Runtime with software on the workstation and Open Runtime firmware for the VEGA ARIES V2 board is shown in Figure 9.

Testing of Modbus Communication Through OpenPLC Runtime on ARIES v2

Modbus communication was tested by configuring the workstation as the Modbus Master and the VEGA Open PLC hardware as the Modbus Slave. The rung in the test ladder containing the timer blocks in Figure 8 was connected to a contact/coil combination named "BlinkyLED." The associated Boolean flag was defined using the IEC 61131 notation "%QX1.0". The status of the timer block was shared with the engineering workstation using the Modbus data communication protocol. To ensure that the Modbus packets from the board were received at the workstation correctly and to prevent any erroneous communication, the UART controller on

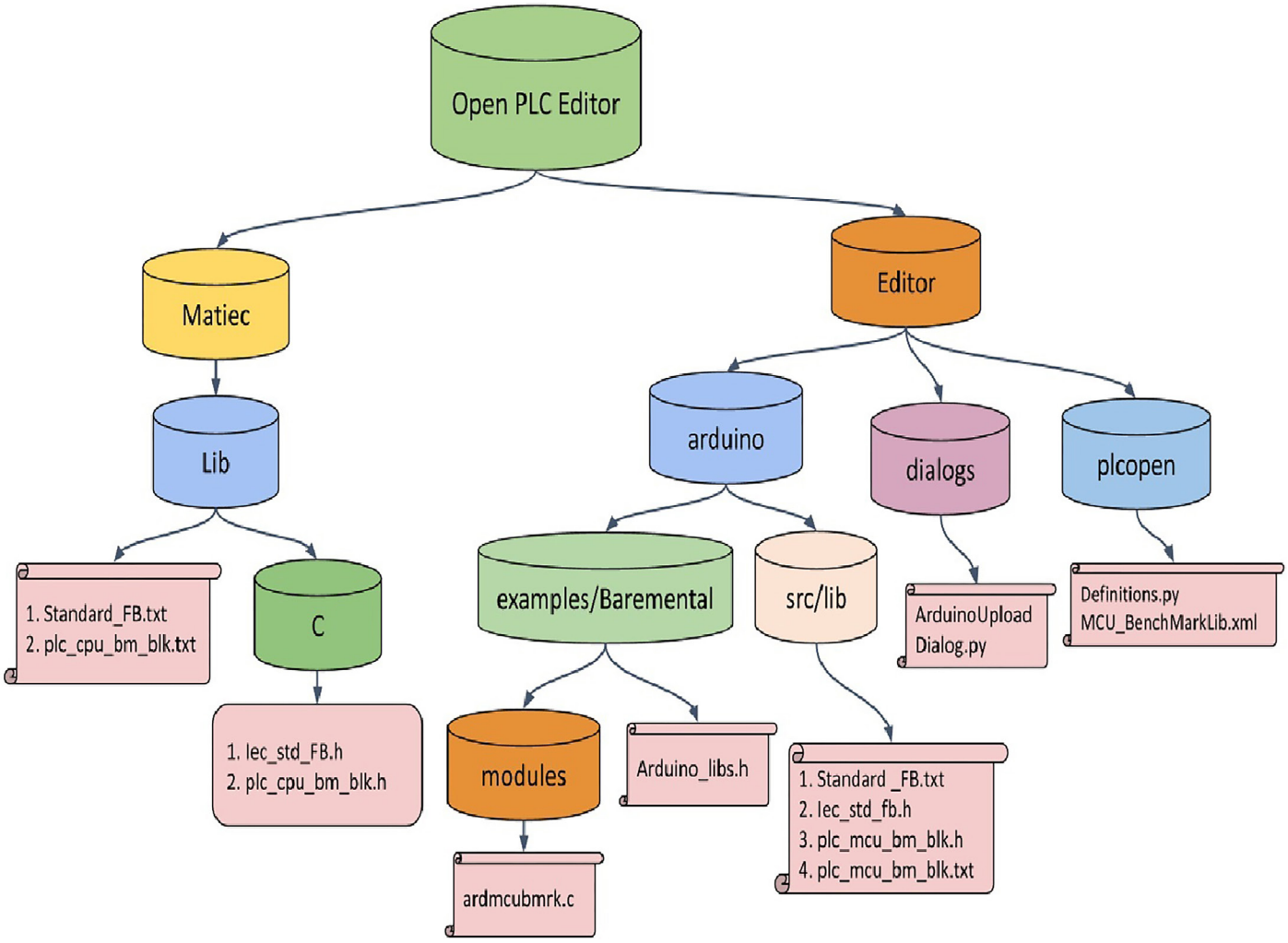

Fig 7 | The architecture of OpenPLC Editor folder structure which mentions in detail the files to be added and modified

Table 3 | Files added to openplc to insert the new benchmarking block

Sl. No	Filename	Path
1	MCU_BenchMarkLib.xml	OpenPLC_Editor\editor\plcopen
2	plc_cpu_bm_blk.h	OpenPLC_Editor\matiec\lib\C
3	plc_cpu_bm_blk.txt	OpenPLC_Editor\matiec\lib
4	plc_mcu_bm_blk.h	OpenPLC_Editor\editor\arduino\src\lib
5	plc_mcu_bm_blk.txt	OpenPLC_Editor\editor\arduino\src\lib
6	ardmcubmrk.c	OpenPLC_Editor\editor\arduino\examples\Baremetal\modules

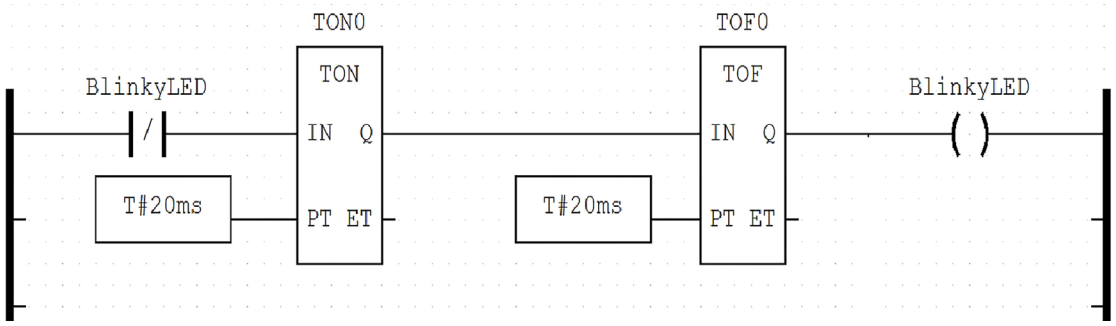

Fig 8 | The ladder logic used for testing PLC functionality

DOI: https://doi.org/10.70389/PJS.100135 | Premier Journal of Science 2025;15:100135

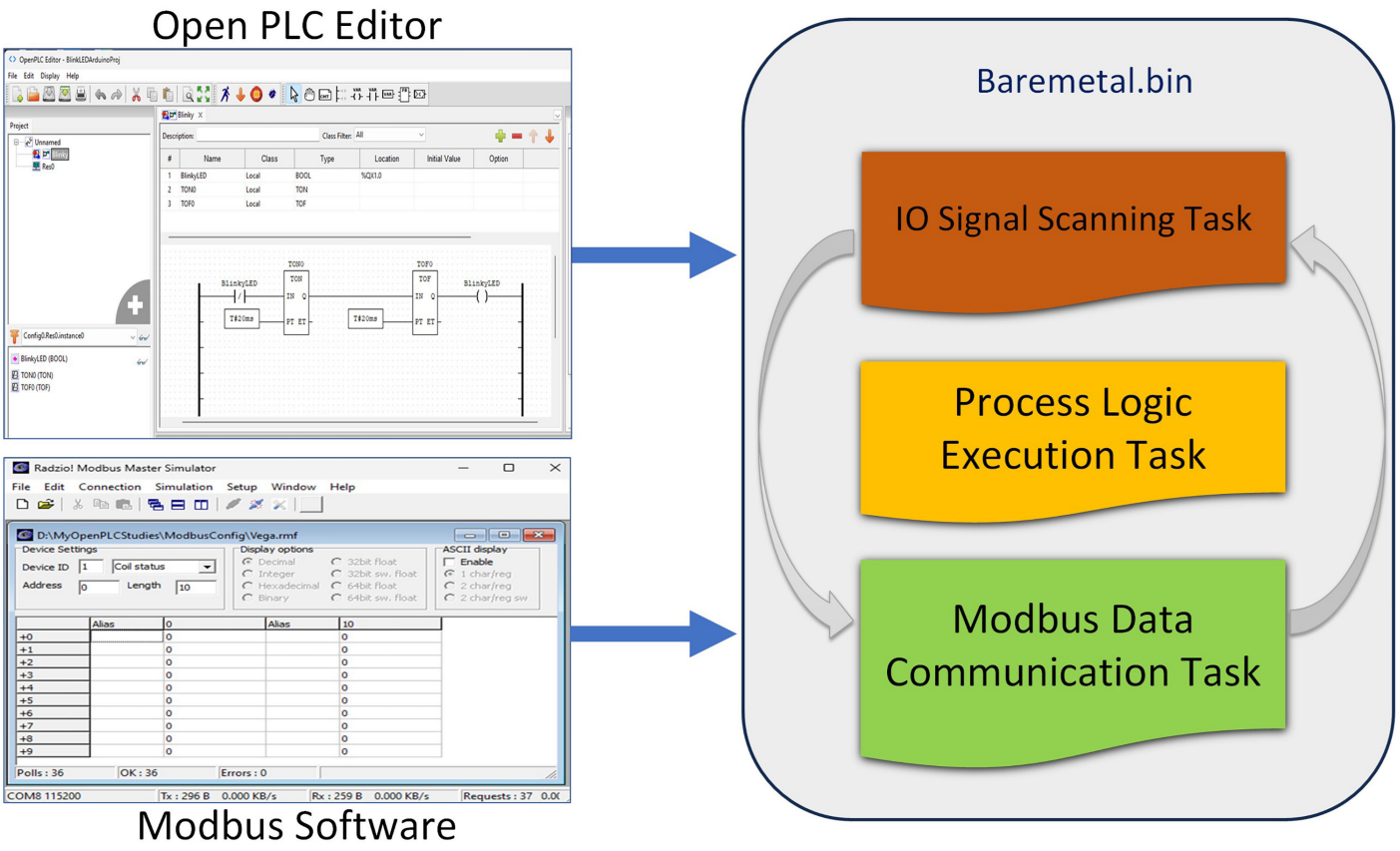

Fig 9 | Software applications on PLC workstation interacting with ARIES board loaded with BareMetal firmware encapsulating PLC tasks

the ARIES v2 board was enabled for a First In First Out (FIFO) based operation. Therefore, the UART Class was modified to support FIFO as well as interrupt-based communication. Thus, the UART driver was integrated with the source code for the soft FIFO and interrupt handler functions. Once the board was tested after the mentioned modification, the request response sequence of Modbus communication still showed erratic results. The Modbus packet response received in the OpenPLC workstation showed a significant number of Modbus packet drops. It was also observed that the interrupt service routine for the serial port was disabled after a few minutes of operation.

Improvements in code for Communication: The UART communication was enhanced by implementing an interrupt-based design with a Software FIFO buffer attached to the driver. The interrupt-based approach allows the system to respond immediately to incoming and outgoing data events without requiring continuous polling. The integrated FIFO buffer temporarily stores data in a first-in-first-out manner, improving data reliability, particularly for high data throughput. The header file supporting the interrupt-based UART class and FIFO buffer was also modified. Modifications in the 'UARTClass.cpp' and 'UARTClass.h' files were integrated with features for interrupt-based UART functionality, in addition to a Soft FIFO implementation. The FIFO buffer handles incoming data queues, enabling efficient serial communication. The 'UART-Class.h' file includes the necessary definitions and declarations to support interrupt and FIFO mechanisms.

Architectural Change in VEGA CPU Setting for OpenPLC Runtime

The PLC logic for Modbus communication was tested again after the modifications discussed in the earlier sections were applied. The Modbus communication module was still erratic with the UART operating in the interrupt mode. Nevertheless, the packet drop was reduced. In addition, the period for which the system worked also increased, from ~120s before to 300s–600s after the modifications were incorporated. However, the results were not comparable with the performance of Arduino boards tested by the developers of OpenPLC.[1,12] Therefore, an implementation based on interrupt routines was considered to streamline Modbus communication from the RISC-V processor. The MCU core of the VEGA processor can be configured for both basic integer instruction set and multiply divide instruction set architecture. Based on this, changes aimed at optimizing architectural parameters and improving the performance of delay functions were implemented. The built MCU configuration was changed from 'RV32IM' to 'RV32I.' This switch to working with a microcontroller architecture emulator built as 'rv32i' resulted in improved compatibility

with interrupt-based operations. This was achieved by modifying the 'boards.txt' file with "aries_v2.build.mcu=rv32i." To accommodate this change, the delay function in "wiring.c" was modified to eliminate the use of instructions with multiplication operations. These modifications resulted in improved efficiency and faster execution on the specified hardware. Therefore, Modbus communication worked well without any packet loss.

Table 4 | Results of bare benchmarking on the candidate microcontroller boards

S. No	Board Name	MCU Clock Speed (MHz) f_{CLK}	Total Ticks (20000 iters.) N_{TK}	Time taken (s) T_i	Iters/ Sec N_{1s}	Bench-marking Index i_{BM}
1	CDAC VEGA ARIES V2	100	1191	1.19	16792.6	167.93
2	ARDUINO UNO	16	4380	4.38	4566.21	285.39
3	ARDUINO NANO	16	4380	4.38	4566.21	285.39
4	ESP32-DOIT DEVKIT V1	80	627	0.63	31897.9	389.72
5	ESP8266 Node MCU	80	1484	1.48	13477.1	168.46
6	STM32 MCU BluePill _F103C8	72	2077	2.08	9629.3	133.74

Benchmarking Various Arduino Boards Using Bare Benchmarking Source Code

The bare benchmarking code was implemented on the listed MCU boards using the Arduino IDE, as mentioned in Table 2. The program also had an external interface to blink an LED and show the result on the Arduino IDE Serial Terminal. The benchmarking results are presented in Table 4. As shown in Table 4, all six candidate MCU boards were able to complete the cycle in a few seconds, with ESP-32 being the fastest, taking only 0.63s, with the benchmarking index being approximately 389. Arduino boards took more than 4s, partly due to the lower clock speed of only 16 MHz. It can be noted that the THEJAS32 processor in the VEGA ARIES v2 board and the ESP8266 processor were based on the RISC-V instruction set, thus yielding similar benchmarking index values.

Testing of OpenPLC on a set of Arduino-Like Boards for MCU Performance Benchmarking

By following the steps outlined earlier the Benchmarking block was embedded in Ladder Logic and executed. Figure 10 shows the Ladder Logic program

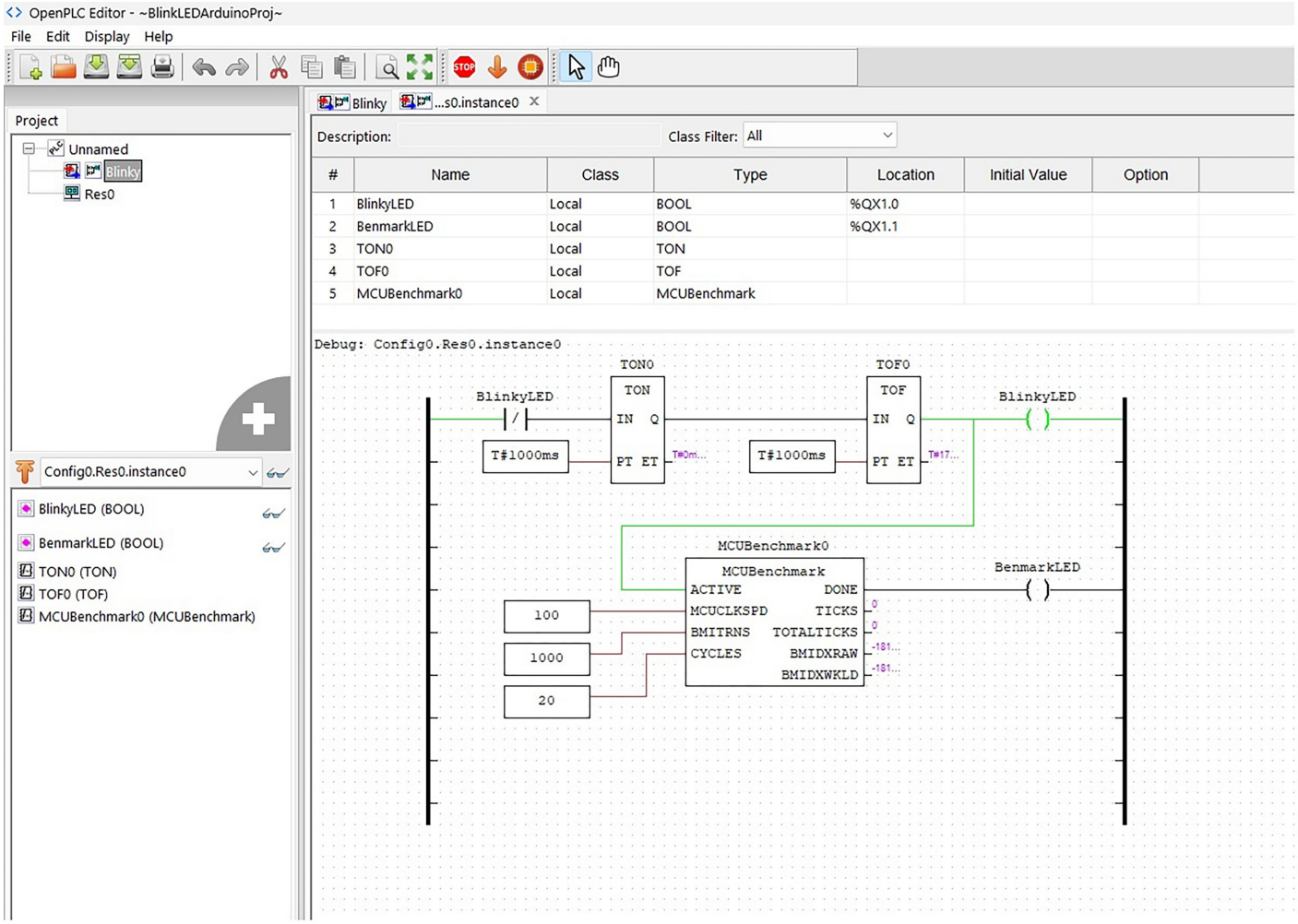

Fig 10 | OpenPLC runtime environment showing the Ladder logic program that periodically runs the 'MCUBenchmark' block

DOI: https://doi.org/10.70389/PJS.100135 | Premier Journal of Science 2025;15:100135

Table 5 | Results of benchmarking on the candidate microcontroller boards

S No	Board Name	MCU Clock Speed (MHz) f_{CLK}	Total Ticks (20 cycles of 1000 iters.) N_{TK}	Raw Bench-marking Index i_{BM}	OpenPLC Bench-marking Index (BMIDXWKLD)
1	CDAC VEGA ARIES V2	100	6869	319	320
2	ESP32-DOIT DEVKIT V1	80	5911	389	389
3	ESP8266 Node MCU	80	6290	174	174

in OpenPLC Runtime used for the benchmarking of the MCU core through a block named 'MCUBenchmark.' The timer-based ladder logic program for the blinking LED was used to generate the Enable (ACTIVE) signal for the block. The other input parameters for the block are MCU Clock speed (in MHz) chosen as per the MCU specification, no. of iterations BMITRNS within a benchmarking cycle was 1000, and the no. of such cycles (CYCLES parameter) was set to 20. The output status signal DONE from the block goes high after the benchmarking tasks are completed. Table 5 compares the benchmarking results for the different candidate MCU boards. Among the boards, the Arduino Uno and Nano boards could not execute the benchmarking algorithm block owing to limitations in their data memory. While the STM32 Microcontroller did not have any compiler errors, serious runtime errors occurred during the execution of the code by OpenPLC. Therefore, in the Bluepill board, bare benchmarking alone was effective. It can be seen that newer generation boards, such as VEGA, ESP32, and ESP-8266, demonstrated remarkably fast performance in running the Benchmarking block, wherein each cycle of 1000 iterations could be completed within 0.4s.

In comparison with the study in Table 4, the number of iterations was limited to 1000 in each cycle, so that the processor resources could also be utilized by other ladder blocks in between the cycles. The parameter BMIDXWKLD is the Benchmarking Index computed by OpenPLC while the Ladder logic was indefinitely running. This was similar to the value of Benchmarking Index computed for a limited number of cycles. This is a testament to the fairly stable execution of memory-intensive operations, such as linked list creation, reversal, and retrieval, by OpenPLC logic running on the processor. This underscores the reliability of OpenPLC as a platform for developing complex control algorithms and executing them in real time. Through the unique efforts of enabling MCU Benchmarking PLC block onto Arduino-complaint boards by the project team, it was understood that low-scale MCU on boards, such as Arduino Nano and Uno, were not able to compile and run the benchmark block in the system.

Conclusion

In this study, an open-source PLC application (OpenPLC) running on a RISC-V open hardware platform was demonstrated. The logic was developed using an open-source editor compatible with the IEC61131-3 standard. This work presents a systematic methodology to enable OpenPLC Runtime application to work on the 'C-DAC VEGA ARIES v2 board', an indigenous, low-cost, open-source embedded board employing the indigenous THEJAS32 SoC. The work included the modification of the Arduino-IDE package for the board, driver files for its I/O channels, and OpenPLC Runtime configuration files. The software suites for OpenPLC and Arduino-IDE were set up at the workstation. Through successive phases of code development and modifications in the OpenPLC framework, the test application was successfully run on the ARIES board.

To gear up the PLC for modern computing requirements, such as intelligent edge devices, application-aware benchmarking were performed. Accordingly, a custom benchmarking PLC function block was developed and added to the OpenPLC Library. Considering the complexity of the benchmarking logic, the demonstration could herald a remarkable push in the applications of inexpensive, yet powerful MCU boards towards advanced automation functions. Future investigations will include optimizing the automation logic using RISC-V instructions wherever possible.

An advanced IEC61131-3 compatible instruction set, which also implements AI/ML functions, can be developed based on the RISC-V instruction set architecture for future applications. Using FPGA platforms that emulate next-generation RISC-V processors, these instructions can be optimized for speed and efficient hardware resource utilization. To this end, the C-DAC VEGA THEJAS32 processor-based ARIES board running RISC-V instructions, completely complying with open hardware requirements and now made compatible with OpenPLC, promises a high-quality, low-cost platform for future development activities in industrial automation.

References

1 Alves TR, Buratto M, de Souza FM, Rodrigues TV. OpenPLC: An open source alternative to automation. In: IEEE Global Humanitarian Technology Conference (GHTC); 2014. p. 585–9. https://doi.org/10.1109/GHTC.2014.6970342

2 Rakitin I, Markova VI. Open source hardware - advantages and applications. In: International Conference on Biomedical Innovations and Applications (BIA), Varna, Bulgaria. 2022. p. 21–4. https://doi.org/10.1109/BIA52594.2022.9831292

3 Chenguang W, Zhiqiang H, Yi Y, Lingbo G, and Ling W. Control system design for micro AUV based on open source hardware. In: IEEE International Conference on Information and Automation (ICIA); 2018. p. 980–4. https://doi.org/10.1109/ICInfA.2018.8812537

4 Eassa H, Adly I, Issa H. RISC-V based implementation of programmable logic controller on FPGA for Industry 4.0. In: 31st International Conference on Microelectronics (ICM), Cairo, Egypt. 2019. p. 98–102. https://doi.org/10.1109/ICM48031.2019.9021939

5 Kaur R, Dash B, Shiney JO, Singh S. Revolutionizing CanSat technology with Vega processors: a comparative study. In: 2nd International Conference on Edge Computing and Applications (ICECAA), Namakkal, India. 2023. p. 1276–82. https://doi.org/10.1109/ICECAA58104.2023.10212104

6 Biswas S, Chatterjee S, Pandit S, Das A. Hardware prototyping of handwritten character recognition using VEGA soft core processor. In: 8th International Conference on Computers and Devices for Communication (CODEC), Kolkata, India. 2023. p. 1–2. https://doi.org/10.1109/CODEC60112.2023.10466133

7 Lakshmi BS, Anish S, Vinukumar AR, Divya DS. Design and development of secure CAN bus communication protocol for industrial field devices using indigenously developed VEGA processor. Unpublished work.

8 Nayyar A, Puri V. A review of Arduino board's, Lilypad's and Arduino shields. In: 3rd International Conference on Computing for Sustainable Global Development, NewDelhi. 2016. p. 1485–92.

9 Bashev V, Anureev I, Zyubin V. The Post language: process-oriented extension for IEC 61131–3 structured text. In: International Russian Automation Conference (RusAutoCon), Sochi, Russia. 2020. p. 994–9. https://doi.org/10.1109/RusAutoCon49822.2020.9208049

10 Halim DK, Ming TC, Song NM, Hartono D. Arduino based IDE for embedded multi-processor system on-chip. In: 5th International Conference on New Media Studies (CONMEDIA), Bali, Indonesia,2019. p. 135–8. https://doi.org/10.1109/CONMEDIA46929.2019.8981862

11 Matijevic M, Cvjetkovic V. Overview of architectures with Arduino boards as building blocks for data acquisition and control systems. In: 13th International Conference on Remote Engineering and Virtual Instrumentation (REV), Madrid, Spain, 2016. p. 56–63. https://doi.org/10.1109/REV.2016.7444440

12 Alves T, Morris T. OpenPLC: An IEC 61131–3 compliant open source industrial controller for cyber security research. Comput Secur. 2018;78:364–79. https://doi.org/10.1016/j.cose.2018.07.007

13 Iacobelli A, Rinieri L, Melis A, Sadi AA, Prandini M, Callegati F. Detection of ladder logic bombs in PLC control programs: an architecture based on formal verification. In: 7th International Conference on Industrial Cyber-Physical Systems (ICPS), St. Louis,USA. 2024. p. 1–7. https://doi.org/10.1109/ICPS59941.2024.10639995

14 Cui E, Li T, Wei Q. RISC-V instruction set architecture extensions: a survey. IEEE Access. 2023;11:24696–711. https://doi.org/10.1109/ACCESS.2023.3246491

15 Jiahui L, Morimoto T, Ogita T, Kawamata R, Ziming W, Tsutsumi T. Microprocessor instruction design tool for RISC-V architecture. In: 22nd International Symposium on Communications and Information Technologies (ISCIT); 2023. p. 1–6. https://doi.org/10.1109/ISCIT57293.2023.10376034

16 Raveendran A, Patil VB, Selvakumar D, Desalphine V. A RISC-V instruction set processor-micro-architecture design and analysis. In: International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA), Bengaluru, India. 2016. p. 1–7. https://doi.org/10.1109/VLSI-SATA.2016.7593047

17 Download OpenPLC webpage [Internet]. Available from: https://autonomylogic.com/download-windows

18 OpenPLC editor webpage. Available from: https://autonomylogic.com/docs/installing-openplc-editor/

19 Arduino software webpage [Internet]. Available from: https://www.arduino.cc/en/software

20 Arduino command line interface specifications webpage. Available from: https://arduino.github.io/arduino-cli/0.34/platform-specification/

21 VEGA ARIES v2 board webpage. Available from: https://vegaprocessors.in/devboards/ariesv2.php

22 Vinukumar AR, Premjith AV, Rajadurai SRS, Natarajan S. Implementation and testing of programmable logic controller (PLC) functionality on C-DAC VEGA RISC-V microcontroller using OpenPLC. In: 4th International Conference on Emerging Electronics and Automation (E2A 2024), Silchar, India.

23 Gal-On S, Levy M. Exploring CoreMark™ – a benchmark maximizing simplicity and efficacy. The Embedded Microprocessor Benchmark Consortium. 2013;6(23):87. Available from: https://www.eembc.org/techlit/articles/coremark-whitepaper.pdf

24 Niauronis S. OpenPLC hardware speed performance comparison. Prof Stud Theory Pract. 2023;27(1):65– 71.

25 Fu Z, Tedeschi R, Ottavi G, Wistoff N, Fuguet C, Rossi D, Benini L. Ramping Up Open-Source RISC-V Cores: Assessing the Energy Efficiency of Superscalar, Out-of-Order Execution. In 22nd ACM International Conference on Computing Frontiers (CF'25), May 28–30, 2025, Cagliari, Italy. ACM, New York, NY, USA, p. 12-20. https://doi.org/10.1145/3719276.3725186

26 Chmiel M; Czerwinski R, Malcher A. FPGA Implementation of IEC 61131-3-Based Hardware-Aided Timers for Programmable Logic Controllers. Electronics. 2023; 12:4255. p. 1-23. https://doi.org/10.3390/electronics12204255