MUHANDISLIK & IQTISODIYOT

ijtimoiy-iqtisodiy, innovatsion texnik,
fan va ta'limga oid ilmiy-amaliy jurnal

2026-YIL
IYUN/6-SON, II-QISM

Milliy nashrlar
OAK: https://oak.uz/pages/4802

05.00.00 - Texnika fanlari
08.00.00 - Iqtisodiyot fanlar

ISSN: 3060-463X

muhandislik
& iqtisodiyot
ijtimoiy-iqtisodiy, innovatsion texnik,
fan va ta’limga oid ilmiy-amaliy jurnal

Elektron nashr, 2026-yil, iyun.

Bosh muharrir:
Zokirova Nodira Kalandarovna, iqtisodiyot fanlari doktori, DSc, professor

Bosh muharrir o‘rinbosari:
Shakarov Zafar G‘afforovich, iqtisodiyot fanlari bo‘yicha falsafa doktori, PhD, dotsent

Tahrir hay’ati:
Abduraxmanov Kalandar Xodjayevich, O‘z FA akademigi, iqtisodiyot fanlari doktori, professor
Sharipov Kongratbay Avezimbetovich, texnika fanlari doktori, professor
Maxkamov Baxtiyor Shuxratovich, iqtisodiyot fanlari doktori, professor
Abduraxmanova Gulnora Kalandarovna, iqtisodiyot fanlari doktori, professor
Shaumarov Said Sanatovich, texnika fanlari doktori, professor
Turayev Bahodir Xatamovich, iqtisodiyot fanlari doktori, professor
Nasimov Dilmurod Abdulloyevich, iqtisodiyot fanlari doktori, professor
Allayeva Gulchexra Jalgasovna, iqtisodiyot fanlari doktori, professor
Arabov Nurali Uralovich, iqtisodiyot fanlari doktori, professor
Maxmudov Odiljon Xolmirzayevich, iqtisodiyot fanlari doktori, professor
Xamrayeva Sayyora Nasimovna, iqtisodiyot fanlari doktori, professor
Bobonazarova Jamila Xolmurodovna, iqtisodiyot fanlari doktori, professor
Irmatova Aziza Baxromovna, iqtisodiyot fanlari doktori, professor
Bo‘taboyev Mahammadjon To‘ychiyevich, iqtisodiyot fanlari doktori, professor
Shamshiyeva Nargizaxon Nosirxuja kizi, iqtisodiyot fanlari doktori, professor,
Xolmuxamedov Muhsinjon Murodullayevich, iqtisodiyot fanlari nomzodi, dotsent
Xodjayeva Nodiraxon Abdurashidovna, iqtisodiyot fanlari nomzodi, dotsent
Amanov Otabek Amankulovich, iqtisodiyot fanlari bo‘yicha falsafa doktori (PhD), dotsent
Toxirov Jaloliddin Ochil o‘g‘li, texnika fanlari bo‘yicha falsafa doktori (PhD)
Qurbonov Samandar Pulatovich, iqtisodiyot fanlari bo‘yicha falsafa doktori (PhD)
Zikriyoyev Aziz Sadulloyevich, iqtisodiyot fanlari bo‘yicha falsafa doktori (PhD)
Tabayev Azamat Zaripbayevich, iqtisodiyot fanlari bo‘yicha falsafa doktori (PhD)
Sxay Lana Aleksandrovna, iqtisodiyot fanlari bo‘yicha falsafa doktori (PhD), dotsent
Ismoilova Gulnora Fayzullayevna, iqtisodiyot fanlari nomzodi, dotsent
Djumaniyazov Umrbek Ilxamovich, iqtisodiyot fanlari nomzodi, dotsent
Kasimova Nargiza Sabitdjanovna, iqtisodiyot fanlari nomzodi, dotsent
Kalanova Moxigul Baxritdinovna, dotsent
Ashurzoda Luiza Muxtarovna, iqtisodiyot fanlari bo‘yicha falsafa doktori (PhD)
Sharipov Sardor Begmaxmat o‘g‘li, iqtisodiyot fanlari bo‘yicha falsafa doktori (PhD)
Tursunov Ulug‘bek Sativoldiyevich, iqtisodiyot fanlari doktori (DSc), dotsent
Bauyetdinov Majit Janizaqovich, Toshkent davlat iqtisodiyot universiteti dotsenti, PhD
Botirov Bozorbek Musurmon o‘g‘li, Texnika fanlari bo‘yicha falsafa doktori (PhD)
Sultonov Shavkatjon Abdullayevich, Kimyo fanlari doktori, (DSc)
Jo‘raeva Malohat Muhammadovna, filologiya fanlari doktori (DSc), professor.
Yusupov Maxamadamin Abduxamidovich, iqtisodiyot fanlari nomzodi (DSc), professor
Kalonova Moxigul Baxritdinovna, iqtisodiyot fanlari nomzodi (PhD), dotsent
Mirzayev Kulmamat Djanzakovich, iqtisodiyot fanlari nomzodi (DSc), professor.
Karimova Nilufar Sadirdin qizi, iqtisodiyot fanlari bo‘yicha falsafa doktori (PhD)
Norboyev Odil Abrayevich, iqtisodiyot fanlari bo‘yicha falsafa doktori (PhD), dotsent
Nasimov Dilmurod Abdulloyevich, iqtisodiyot fanlari doktori (DSc), professor
Mirzayev Kulmamat Djanzakovich, iqtisodiyot fanlari doktori (DSc), professor
Karimova Nilufar Sadirdin qizi, iqtisodiyot fanlari bo‘yicha falsafa doktori (PhD)
Pardaev Umidjon Uralovich, iqtisodiyot fanlari doktori (DSc), professor
Xolmirzayev Ulug‘bek Abdulazizovich, Iqtisodiyot fanlari doktori (DSc)

muhandislik & iqtisodiyot

ijtimoiy-iqtisodiy, innovatsion texnik,
fan va ta’limga oid ilmiy-amaliy jurnal

05.01.00 – Axborot texnologiyalari, boshqaruv va kompyuter grafikasi
05.01.01 – Muhandislik geometriyasi va kompyuter grafikasi. Audio va video texnologiyalari
05.01.02 – Tizimli tahlil, boshqaruv va axborotni qayta ishlash
05.01.03 – Informatikaning nazariy asoslari
05.01.04 – Hisoblash mashinalari, majmualari va kompyuter tarmoqlarining matematik va dasturiy ta’minoti
05.01.05 – Axborotlarni himoyalash usullari va tizimlari. Axborot xavfsizligi
05.01.06 – Hisoblash texnikasi va boshqaruv tizimlarining elementlari va qurilmalari
05.01.07 – Matematik modellashtirish
05.01.11 – Raqamli texnologiyalar va sun’iy intellekt
05.02.00 – Mashinasozlik va mashinashunoslik
05.02.08 – Yer usti majmualari va uchish apparatlari
05.03.02 – Metrologiya va metrologiya ta’minoti
05.04.01 – Telekommunikasiya va kompyuter tizimlari, telekommunikasiya tarmoqlari va qurilmalari. Axborotlarni taqsimlash
05.05.03 – Yorug‘lik texnikasi. Maxsus yoritish texnologiyasi
05.05.05 – Issiqlik texnikasining nazariy asoslari
05.05.06 – Qayta tiklanadigan energiya turlari asosidagi energiya qurilmalari
05.06.01 – To‘qimachilik va yengil sanoat ishlab chiqarishlari materialshunosligi
05.08.03 – Temir yo‘l transportini ishlatish
05.08.06 – “G‘ildirakli va gusenisali mashinalar va ularni ishlatish” (texnika fanlari)
05.09.01 – Qurilish konstruksiyalari, bino va inshootlar
05.09.04 – Suv ta’minoti. Kanalizatsiya. Suv havzalarini muhofazalovchi qurilish tizimlari
10.00.06 – Qiyosiy adabiyotshunoslik, chog‘ishtirma tilshunoslik va tarjimashunoslik
10.00.04 – Yevropa, Amerika va Avstraliya xalqlari tili va adabiyoti
08.00.01 – Iqtisodiyot nazariyasi
08.00.02 – Makroiqtisodiyot
08.00.03 – Sanoat iqtisodiyoti
08.00.04 – Qishloq xo‘jaligi iqtisodiyoti
08.00.05 – Xizmat ko‘rsatish tarmoqlari iqtisodiyoti
08.00.06 – Ekonometrika va statistika
08.00.07 – Moliya, pul muomalasi va kredit
08.00.08 – Buxgalteriya hisobi, iqtisodiy tahlil va audit
08.00.09 – Jahon iqtisodiyoti
08.00.10 – Demografiya. Mehnat iqtisodiyoti
08.00.11 – Marketing
08.00.12 – Mintaqaviy iqtisodiyot
08.00.13 – Menejment
08.00.14 – Iqtisodiyotda axborot tizimlari va texnologiyalari
08.00.15 – Tadbirkorlik va kichik biznes iqtisodiyoti
08.00.16 – Raqamli iqtisodiyot va xalqaro raqamli integratsiya
08.00.17 – Turizm va mehmonxona faoliyati

Ma’lumot uchun, OAK Rayosatining 2024-yil 28-avgustdagi 360/5-son qarori bilan “Dissertatsiyalar asosiy ilmiy natijalarini chop etishga tavsiya etilgan milliy ilmiy nashrlar ro‘yxati”ga texnika va iqtisodiyot fanlari bo‘yicha “Muhandislik va iqtisodiyot” jurnali ro‘yxatga kiritilgan.

Muassis: “Tadbirkor va ishbilarmon” MChJ
Hamkorlarimiz:
1. Toshkent shahridagi G.V.Plexanov nomidagi Rossiya iqtisodiyot universiteti
2. Toshkent davlat iqtisodiyot universiteti
3. Toshkent irrigatsiya va qishloq xo‘jaligini mexanizatsiyalash muhandislari instituti” milliy tadqiqot universiteti
4. Islom Karimov nomidagi Toshkent davlat texnika universiteti
5. Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti
6. Toshkent davlat transport universiteti
7. Toshkent arxitektura-qurilish universiteti
8. Toshkent kimyo-texnologiya universiteti
9. Jizzax politexnika instituti

MUNDARIJA

LOGICLABUZ: A BROWSER-NATIVE CO-SIMULATION PLATFORM FOR THE VIRTUALIZATION OF MULTI-MCU ROBOTIC SYSTEMS

Ergashev Adizbek Kamol ugli
Tashkent University of Information Technologies
Researcher at the Department of Systematic and Practical Programming
ORCID: 0009-0006-2847-260X

Abstract: Faithful virtualization of a microcontroller-based robotic system requires three capabilities that existing tools rarely combine: cycle-accurate execution of the real instruction set, a continuous-domain electrical model of the surrounding circuit, and a scheduler that couples the two at interactive speed. This paper presents Logiclabuz, a browser-native co-simulation platform that provides all three. The simulator implements the full instruction set of three AVR variants, drives a library of more than fifty electrically modeled components through a DC circuit solver with a fault layer, and infers multi-MCU UART, SPI, and I^2C links from the wire topology. Two design choices carry the performance: a unified data-space buffer that removes per-region branching from the instruction hot path, and a two-tier scheduler that amortizes peripheral evaluation while preserving cycle-accurate timer reads through linear interpolation. On a ten-workload benchmark, Logiclabuz runs 1.39 to 4.58 times faster than avr8js, the closest open browser comparator, reaching 59 to 171 million simulated cycles per second; the scheduler alone accounts for an 11.7 to 43.6 times speedup over per-cycle evaluation. A conformance harness against simavr 1.7 and avr8js 0.21.0 establishes bit-for-bit agreement on the CPU instruction subset, and a 60-circuit benchmark yields a macro F1 of 0.81 for the fault detector. The platform is deployed at https://logiclab.uz.

Keywords: embedded systems; microcontroller simulation; co-simulation; cycle-accurate emulation; circuit virtualization; educational tools; AVR.

Annotatsiya: Mikrokontrollerga asoslangan robototexnik tizimni ishonchli virtuallashtirish mavjud vositalarda kamdan-kam hollarda birga uchraydigan uchta imkoniyatni ta'minlaydi: mikrokontroller buyruqlar to'plamini takt aniqlikda (cycle-accurate) bajarish, atrofdagi elektr zanjirining uzluksiz sohadagi modeli va bu ikkalasini interaktiv tezlikda bog'laydigan muvofiqlashtiruvchi (scheduler). Ushbu maqolada uchala imkoniyatni ham ta'minlaydigan, brauzerda ishlaydigan birgalikda simulyatsiya (co-simulation) platformasi — Logiclabuz taqdim etiladi. Simulyator uchta AVR variantining to'liq buyruqlar to'plamini amalga oshiradi, elektr jihatdan modellashtirilgan ellikdan ortiq komponentlar to'plami va nosozlik qatlamiga ega bo'lgan o'zgarmas tok (DC) zanjiri yechuvchisi orqali boshqaradi hamda ko'p mikrokontrollerli UART, SPI va I^2C ulanishlarini simlar topologiyasidan xulosa qiladi. Unumdorlikni ikkita loyihaviy yechim ta'minlaydi: buyruqlarning tezkor yo'ldan hududlar bo'yicha tarmoqlanishni bartaraf etadigan yagona ma'lumotlar fazosi buferi va periferiya hisob-kitobini taqsimlab, shu bilan birga taymer o'qishlarini chiziqli interpolyatsiya orqali takt aniqlikda saqlaydigan ikki bosqichli rejalashtiruvchi. O'nta turli dasturdan iborat sinovda Logiclabuz eng yaqin ochiq brauzer raqibi avr8js'ga nisbatan 1,39–4,58 baravar tez ishlaydi va sekundiga 59–171 million simulyatsiya qilingan taktga erishadi; rejalashtiruvchining o'zi har takt bo'yicha baholashga nisbatan 11,7–43,6 baravar tezlashishni beradi. simavr 1.7 va avr8js 0.21.0 ga nisbatan o'tkazilgan muvofiqlik sinovi protsessor buyruqlarining qism-to'plamida bitma-bit moslikni tasdiqlaydi, 60 ta sxemadan iborat sinov esa nosozlik aniqlagich uchun makro F1 = 0,81 natijasini beradi. Platforma https://logiclab.uz manzilida joylashtirilgan.

Kalit so'zlar: o'rnatilgan tizimlar; mikrokontroller simulyatsiyasi; birgalikda simulyatsiya; takt aniqlikdagi emulyatsiya; elektr zanjirlarni virtuallashtirish; ta'lim vositalari; AVR.

Аннотация: Достоверная виртуализация робототехнической системы на базе микроконтроллера требует трёх возможностей, которые в существующих инструментах сочетаются редко: потактово-точного (cycle-accurate) исполнения реального набора инструкций, модели окружающей электрической цепи в непрерывной области и планировщика, связывающего их с интерактивной скоростью. В статье представлена Logiclabuz - работающая в браузере платформа совместного моделирования (co-simulation), обеспечивающая все три возможности. Симулятор реализует полный набор инструкций трёх вариантов AVR, управляет библиотекой из более чем пятидесяти электрически смоделированных компонентов через решатель цепей постоянного тока (DC) с слоем диагностики неисправностей и выводит межмикроконтроллерные связи UART, SPI и I²C из топологии соединений. Производительность обеспечивают два проектных решения: единый буфер пространства данных, устраняющий ветвление по регионам на «горячем» пути инструкций, и двухуровневый планировщик, который амортизирует вычисление периферии, сохраняя потактовую точность чтения таймеров за счёт линейной интерполяции. На наборе из десяти нагрузок Logiclabuz работает в 1.39 - 4.58 раза быстрее avr8js ближайшего открытого браузерного аналога - достигая 59 - 171 миллиона смоделированных тактов в секунду; сам планировщик даёт ускорение в 11.7 - 43.6 раза по сравнению с потактовым вычислением. Проверка соответствия относительно simavr 1.7 и avr8js 0.21.0 устанавливает побитовое совпадение на подмножестве инструкций процессора, а тест из 60 схем даёт макро-F1 = 0.81 для детектора неисправностей. Платформа развёрнута по адресу https://logiclab.uz.

Ключевые слова: встраиваемые системы; моделирование микроконтроллеров; совместное моделирование; потактово-точная эмуляция; виртуализация электрических цепей; образовательные инструменты; AVR.

INTRODUCTION

A microcontroller-based robotic system is seldom a single controller in isolation. A line-following vehicle, an automated greenhouse, or a security demonstrator typically integrates two or more controllers, a dozen sensors, one or two actuators, a display, and several inter-device protocols such as UART, SPI, I²C, and 1-Wire. To virtualize such a system, that is, to execute its unmodified firmware against a model of its electrical and protocol environment with sufficient fidelity for instruction and protocol debugging, three capabilities are required together. The first is cycle-accurate execution of the real instruction set, so that timing-dependent protocols behave as on hardware. The second is a continuous-domain electrical model that responds to GPIO writes, PWM duty cycles, and ADC sampling, and that can flag unsafe topologies such as overcurrent or a missing current-limiting resistor. The third is a co-simulation scheduler that couples the two without degrading interactivity.

Existing platforms provide at most two of the three. simavr [1] and the avr8js core of Wokwi [2] are cycle-accurate but expose little analog feedback; SimulIDE [3] and TinkerCad Circuits [4] model analog circuits but are not browser-native, lack multi-MCU support, or are not cycle-accurate; and Proteus VSM [5] approaches full coverage but is proprietary, desktop-only, and costly for the classrooms it would most benefit. The cost barrier matters in lower-resource settings, where the shortage of accessible, high-fidelity simulation tools is a documented obstacle to robotics education [6].

This paper presents Logiclabuz, an open-architecture platform that addresses all three requirements in the browser. It is deployed at logiclab.uz and runs client-side as a Progressive Web Application, with no server component other than a sandboxed compiler service for C and Arduino sources. Its contributions are (i) a complete, cycle-accurate, multi-chip AVR simulator implemented in TypeScript without WebAssembly or native code; (ii) a unified-buffer data-space layout that removes per-region branching from the instruction hot path; (iii) a two-tier scheduler with linear-interpolated timer reads, whose formal correctness is established in a companion paper [7] and measured here; (iv) a continuous-domain circuit solver with a fault layer that maps electrical anomalies to ten diagnostic codes; and (v) a wire-topology-inferred multi-MCU protocol binder that detects UART, SPI, and I²C links from the drawn circuit without manual bus configuration. The paper follows the IMRAD structure: Section 2 reviews related work, Section 3 describes the architecture and evaluation methodology, Section 4 reports the results, and Sections 5 and 6 discuss and conclude.

REVIEW OF LITERATURE ON THE SUBJECT

The reference open-source AVR emulator is simavr [1], a headless C engine that is cycle-accurate on most instructions and that we use as the conformance and performance reference. Wokwi [8] is a commercial browser IDE built on a closed core; its open predecessor avr8js [2] implements a smaller AVR subset in TypeScript and is the closest publicly available comparator to the Logiclabuz core, against which Section 4 benchmarks directly. On the analog side, the SPICE family [9] provides high-fidelity transient analysis at a per-nanosecond cost far beyond an interactive co-simulator's budget; SimulIDE [3], pairing a simavr-derived core with an analog canvas, is the closest free competitor but is desktop-only with limited multi-MCU support. General co-simulation standards couple heavyweight models over millisecond-scale steps, a different cost regime from the microsecond-budget, MCU-and-DC-circuit specialization pursued here and formalized in the companion paper [7].

Table 1 summarizes the comparison. The intersection of cycle-accurate execution, multi-MCU support, browser-native deployment, and a continuous-domain electrical model with a pedagogical fault layer is, to our knowledge, not occupied by any prior platform (Table 1).

Table 1. Feature comparison across browser-native and desktop MCU simulation platforms[1] (● full, ◐ partial, ○ absent, - not applicable)

System	Cycle-acc. MCU	Analog circuit	Fault layer	Multi-MCU	Browser-native	Open source	Multi-lingual
Logiclabuz	●	●	●	●	●	◐ (proprietary)	●
Wokwi	●	◐	○	◐ named buses	●	◐ partial	◐
simavr	●	○	○	◐ programmatic	○	●	○
SimulIDE	●	●	○	◐	○	●	◐
TinkerCad	◐	●	○	○	●	○	◐
Proteus VSM	●	●	○	●	○	○	○
avr8js	◐	○	○	○	●	●	○
Microcorruption	● (MSP430)	—	—	○	●	○	○

A distinct line of work concerns the pedagogy of MCU platforms. Wolffe et al. [10] survey online simulators for computer-organization courses, and recent studies document browser-based simulators in microcontroller teaching [11]. Within the national context of this work, robotics and embedded-systems education in Uzbekistan is expanding but constrained by tool access: Abdullaeva and Khurana [6] identify the shortage of accessible simulation platforms as a barrier in the school system, and Kholkhujaev et al. [12] survey the regional robotics and simulator market without identifying a browser-native, cycle-accurate option. Related institutional work on hybrid FPGA-and-microcontroller teaching benches [13] and the IEEE-sponsored ICISCT conference series hosted at TUIT indicate active regional interest, yet we did not locate prior Uzbek-authored work on cycle-accurate microcontroller simulation, a gap this platform is intended to help close. This paper reports the engineering and technical evaluation; classroom outcome studies are deferred to future work.

RESEARCH METHODOLOGY

Logiclabuz is a Bun monorepo of three packages: the distributable AVR simulator library, the Vue 3 web IDE, and a small HTTP service wrapping `arduino-cli`. Figure 1 shows the overall structure: the browser tab hosts the IDE and a Web-Worker-isolated simulator, while the stateless compiler service builds C and Arduino sources to Intel HEX over HTTP, so the deployed application is a static Progressive Web Application backed only by that service. Within the core, a `Chip` base class aggregates a CPU, memory, and a peripheral set; concrete subclasses (`ATmega328P`, `ATmega2560`, `ATtiny85`) instantiate the peripherals and their register addresses. A single `InstructionSet` class implements decode-and-execute for all 130 ATmega328P opcodes, parameterized by memory sizes so that it is shared across variants; per-instruction unit tests confirm behavior and cycle count against the AVR Instruction Set Manual. Factoring the decoder out of any specific chip makes multi-chip support, and an eventual non-AVR port, require only a new chip class while reusing the surrounding infrastructure (Figure 1).

---

[1] Source: Compiled by the author.

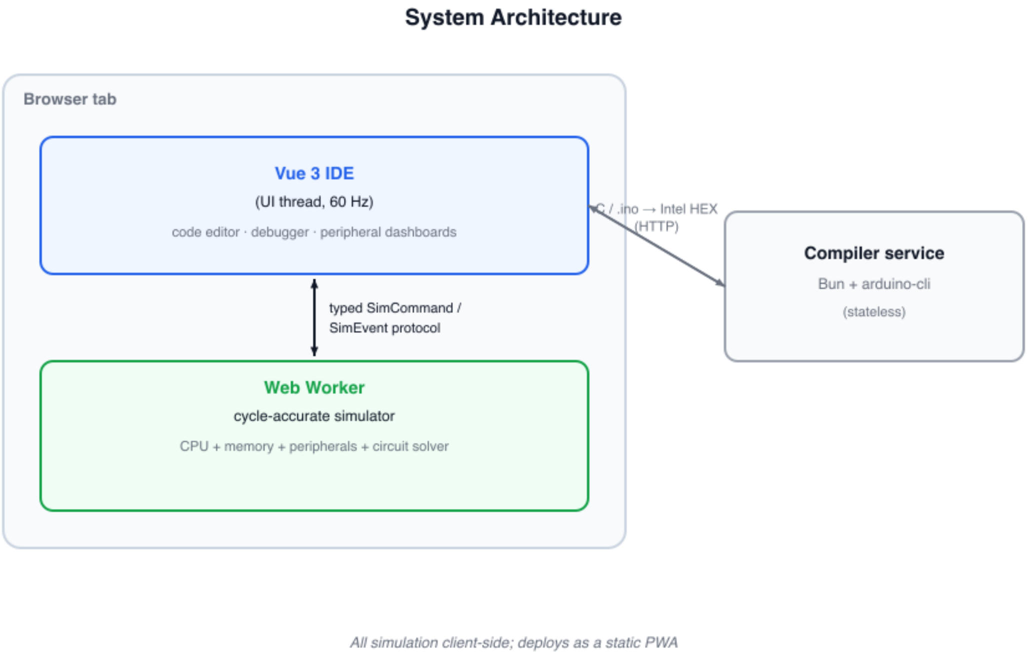

Figure 1. System architecture[2]

The browser tab hosts the Vue 3 IDE and a Web-Worker-isolated, cycle-accurate simulator; a stateless compiler service builds C and Arduino sources to Intel HEX over HTTP.

The AVR data space is a flat, 8-bit-addressed region whose first 32 bytes alias the register file, the next 64 are standard I/O registers, the next 160 are extended I/O, and the remainder is internal SRAM. A naive implementation allocates one typed array per region and dispatches on the address, incurring a branch on every memory instruction. Logiclabuz instead allocates one backing buffer and aliases typed-array views over disjoint regions:

```
const dataBuffer = new ArrayBuffer(totalDataSize);
this.dataSpace = new Uint8Array(dataBuffer);
this.ioRegisters = new Uint8Array(dataBuffer, 0x20, 64);
this.extendedIoRegisters = new Uint8Array(dataBuffer, 0x60, extendedIoSize);
```

A write through any view is visible to every other because all share the buffer, so the hot path performs a single unconditional indexed access; the register file occupies the same buffer at the same offsets. Four fast-path side-effect hooks (`onIoWriteFast`, `onIoReadFast`, `onExtIoWriteFast`, `onExtIoReadFast`) attach at the two I/O ranges. The technique is standard in C emulators but uncommon in JavaScript, where the saved per-instruction branch is the main benefit.

Ticking every peripheral on every CPU cycle costs O(n) per instruction in the number of active peripherals, roughly ten function calls for a fully configured ATmega328P, which precludes interactive speed. Logiclabuz uses a two-tier scheduler: the inner loop fetches, decodes, and executes one instruction and touches no peripheral state beyond the data-space writes the instruction performs, while every `peripheralInterval` cycles (default 512) the outer loop advances all peripherals by the elapsed cycle count and synchronizes their state. This trades a bounded worst-case latency for peripheral-visible events against a large amortized speedup, measured at 11.7 to 43.6 times over the eager baseline (§ 4.3); the companion paper [7] proves the conditions under which the lazy schedule is observationally equivalent to the eager one.

Two refinements preserve correctness under lazy synchronization. First, registers whose writes must take effect before the next instruction reads them, such as `UDR0` and `SPDR`, are forwarded synchronously through the write hooks. Second, registers whose value is a known linear function of the cycle count, principally the Timer 0 counter `TCNT0` and the overflow flag `TIFR0`, are interpolated on read from the prescaler divisor, remainder, and counter base cached at the last synchronization, as `(base + ⌊(rem + Δc) / div⌋) mod 256`. The interpolation is exact while the prescaler configuration and counter base are unchanged, which lets tight timer-polling loops interact with the lazy schedule without observable drift.

The simulator runs in a dedicated Web Worker that communicates with the UI thread through a typed

---

[2] Source: Compiled by the author.

message protocol of commands and state events. Isolation is essential: without it a fast simulated loop would block rendering, whereas with it the UI paints at 60 Hz independently of simulator load, and a worker can in principle be spawned per chip for parallel multi-MCU simulation. The C and Arduino path is served by a stateless Bun service that compiles a posted source with `arduino-cli` under an FQBN allowlist, a 64 KB source cap, and a 30 s timeout, returning Intel HEX that the browser loads into flash. Keeping simulation client-side and compilation in a small isolated service lets the application deploy as a static, offline-capable PWA while the compiler retains the resource isolation a multi-tenant build requires.

The electrical model is invoked from the worker after each topology change or GPIO transition. It is a DC series-path solver rather than a SPICE-class transient simulator, chosen to fit the per-synchronization cost budget. The `NetBuilder` groups pins into electrical nets using union-find; wires and internal shorts union their endpoints, whereas resistor, diode, and voltage-divider terminals stay separate because current flows through them. Each component maps to one of eight electrical models (voltage source, ground, resistor, diode, short, voltage divider, open input, ADC input). The `SeriesSolver` enumerates paths from each driven net to ground and applies elementary series-circuit analysis, $I = (V_{source} - \Sigma V_f) / \Sigma R$, treating a zero-resistance driven path as a short and an insufficient source voltage as a non-conducting path, with PWM duty scaling the source voltage; per-component voltage, current, and power follow by back-propagation. The cost is $O(P \cdot L)$ for P source-to-ground paths of maximum length L, under one millisecond for the circuits encountered in practice. Figure 2 shows a worked example (Figure 2).

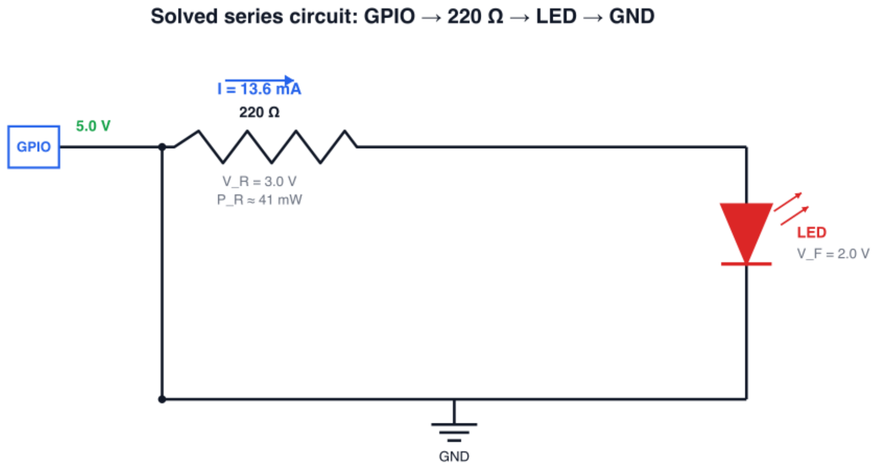
Solved series circuit: GPIO → 220 Ω → LED → GND

Figure 2. A solved series circuit[3]

A GPIO output drives a 220 Ω resistor in series with a red LED to ground on a 5 V rail. With $V_f = 2.0$ V, $I = (5.0 - 2.0)/220 = 13.6$ mA, $V_R = 3.0$ V, and $P_R = I^2 R \approx 41$ mW.

The `FaultDetector` consumes the solved circuit in a single $O(|E|)$ pass and emits diagnostics from ten codes (`led_no_resistor`, `led_overcurrent`, `led_dim`, `led_reverse`, `gpio_overcurrent`, `chip_overcurrent`, `resistor_overpower`, `short_circuit`, `floating_input`, `overvoltage`), each with a severity, the offending component, and the value and limit that triggered it. The contribution is one of framing: where a SPICE tool reports a current the learner must interpret, the detector reports a named, actionable diagnosis, for example that an LED conducts 41 mA above its 20 mA limit and needs a series resistor of at least 150 Ω. Fault messages are localized into English, Russian, and Uzbek. DC-only modeling leaves transient, inductive, and AC phenomena out of scope (Section 5).

Each modeled component is a triple of a definition file, a Vue rendering component, and a simulator class, which keeps the library extensible without core changes. The library spans more than fifty components across boards, passive and power parts, light emitters and displays, digital and analog sensors, bus sensors (DHT11/22, DS18B20 over 1-Wire, HC-SR04), motor and output modules, and clock and communication modules; sensors with proprietary protocols are simulated at the bus level. A multi-MCU circuit could be configured by asking the learner to name a virtual bus, the approach taken by comparable platforms; Logiclabuz instead infers the bus from the wiring. The `interChipBridge` walks the union-find net structure and, for each board pair, consults a per-chip table mapping peripheral functions to pins; when board A's TX net coincides with board B's RX net

---

[3] Source: Compiled by the author.

and vice versa, it installs a byte-forwarding hook between the USARTs, and the same pattern recognizes SPI and I^{2}C pairings, at $O(B^2 + W)$ cost for B boards and W wires. This makes the canonical distributed-sensor exercise, a master polling a slave over UART, buildable exactly as on a physical breadboard. We did not locate peer-reviewed prior art for inferring a multi-MCU protocol binding from a static schematic; the companion paper [7] gives the complexity analysis.

The debugger wraps the CPU with before-step and memory-access hooks and supports conditional breakpoints, memory watchpoints, and call-stack-aware stepping. Time-travel debugging is provided by a checkpoint ring buffer that snapshots CPU state at a fixed instruction interval (default 1,000 instructions over 100 slots, giving 100,000 instructions of reverse history at constant memory); reverse stepping restores the nearest preceding checkpoint and replays forward. Checkpoint-based time travel is established in record-and-replay systems [14] but is, to our reading, absent from prior browser-based MCU emulators; the constant-memory variant suits the small state and high instruction rate of an MCU. The core is about 12,000 lines of dependency-free TypeScript; the web package uses Vue 3, Vite, and CodeMirror. The platform deploys as a service-worker-cached PWA that runs offline after first load, a property chosen for intermittent-connectivity classrooms, and is distributed under a proprietary license.

We evaluate four axes. *Conformance:* beyond a per-instruction unit suite (4,533 assertions over all 130 opcodes), a ten-program harness runs each byte-identical binary at instruction granularity on Logiclabuz, simavr 1.7 [1], and avr8js 0.21.0 [2] and compares PC, SP, SREG, all 32 registers, the SRAM window `0x0100-0x011F`, and the cycle count, passing only if every field matches both references. *Throughput:* ten short programs span CPU-bound to interrupt-bound behavior; each binary, identical across engines, runs for a 10^7-cycle budget over five repetitions on Apple Silicon (Bun 1.3.11), both engines headless, with the avr8js runner instantiating the full peripheral set to avoid under-charging it. *Scheduler ablation:* the same workloads run with `peripheralInterval` swept over {1, 64, 512, 4096}. *Drift:* a separate benchmark diffs architectural state (67 fields at 10 checkpoints) and event timing against the N = 1 eager schedule. *Fault recall:* thirty circuits each engineered to trigger one fault category, plus thirty controls, are scored per code. All harnesses and raw data are in `research/bench/`.

ANALYSIS AND RESULTS

Six of the ten conformance programs match both references bit-for-bit on PC, SP, SREG, the full register file, the SRAM window, and the cycle count. These cover arithmetic and logic, branches, the SREG flag chain, stack and pointer operations, and the atomic I/O bit operations, and constitute the evidence for the cycle-accuracy claim on the CPU instruction subset. The four remaining programs agree on PC, SP, SREG, and SRAM but differ in registers populated by peripheral I/O reads; the divergences arise from a TIFR0 compare-flag latching difference, an undriven-channel ADC default, unspecified UDRE0 behavior with the transmitter disabled, and an unconnected-MISO default (§ 5). We therefore state the claim precisely: the Logiclabuz instruction set is bit-cycle-accurate relative to simavr 1.7 and avr8js 0.21.0 on the CPU instruction subset, with the listed peripheral behaviors as documented limitations.

Table 2 compares Logiclabuz at its default `peripheralInterval = 512` against avr8js at its documented default. Logiclabuz runs 1.39 to 4.58 times faster across the ten workloads, reaching 59 to 171 million simulated cycles per second, equivalently 3.7 to 10.7 times the ATmega328P’s 16 MHz real-time clock. Eight workloads fall in a 1.39 to 2.24 band; the outliers are interpreted in Section 5. The coefficient of variation is at most 5 % on nine of ten workloads. Figure 3 plots the comparison. To our knowledge no peer-reviewed benchmark of avr8js against simavr exists in the prior literature, so Table 2 is the first published cross-implementation throughput comparison for browser-deployed AVR emulation (Table 2; Figure 3).

Table 2. Throughput at default configuration; Throughput in millions of cycles per second; real-time factor (RTF) is throughput over the 16 MHz nominal clock[4]

Workload	Logiclabuz Mcyc/s	avr8js Mcyc/s	Ratio	Logiclabuz RTF	avr8js RTF
`cpu_loop`	130.9	91.1	**1.44×**	8.2×	5.7×
`fib`	109.7	67.7	**1.62×**	6.9×	4.2×
`blink_loop`	59.5	38.6	**1.54×**	3.7×	2.4×
`usart_tx`	149.2	82.1	**1.82×**	9.3×	5.1×
`adc_read`	154.2	85.6	**1.80×**	9.6×	5.3×

---

[4] Source: Compiled by the author.

Workload	Logiclabuz Mcyc/s	avr8js Mcyc/s	Ratio	Logiclabuz RTF	avr8js RTF
`spi_xfer`	138.8	63.0	**2.20×**	8.7×	3.9×
`pwm_setup`	116.4	83.9	**1.39×**	7.3×	5.2×
`timer_isr`	110.4	24.1	**4.58×**	6.9×	1.5×
`recursive_call`	171.3	103.1	**1.66×**	10.7×	6.4×
`arduino_blink`	149.9	66.9	**2.24×**	9.4×	4.2×

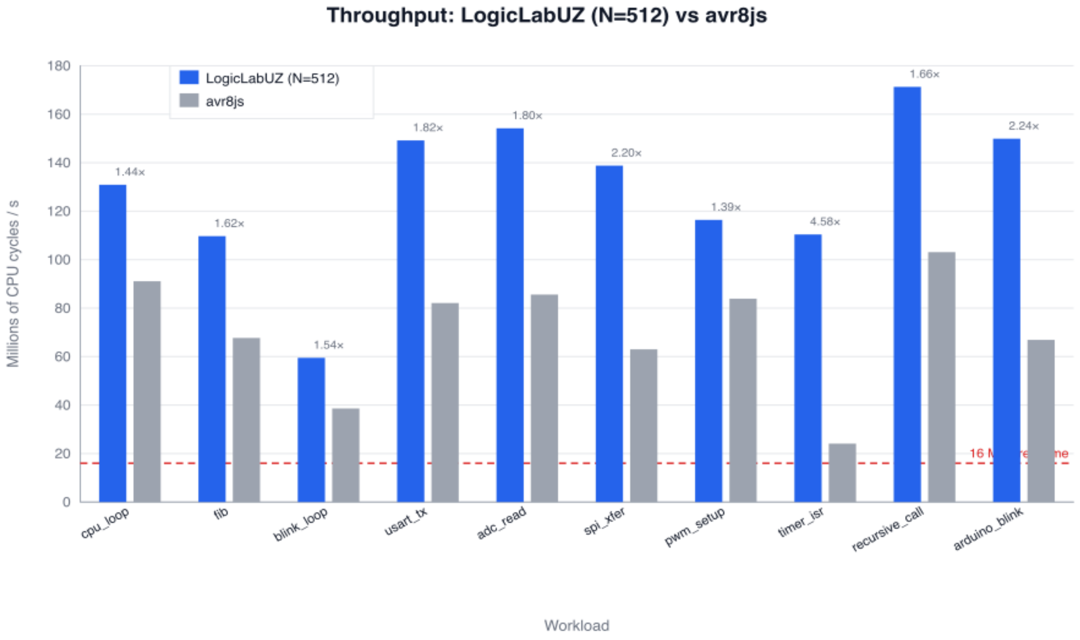

Figure 3. Throughput of Logiclabuz (peripheralInterval = 512) versus avr8js across the ten workloads, on a logarithmic scale; the dashed line marks 16 MHz real time[5]

Sweeping `peripheralInterval` isolates the scheduler's contribution. Disabling it (N = 1) drops throughput to 2.8-5.7 Mcyc/s, well below avr8js's 24-103 Mcyc/s on the same workloads, so the scheduler is the load-bearing optimization rather than an incremental one. Raising N to the default of 512 yields an 11.7 to 43.6 times speedup, smallest on the GPIO-heavy `blink_loop`, where each iteration triggers a write hook that cannot be amortized further, and largest on the peripheral-free `cpu_loop`. The marginal gain from 512 to 4096 is at most 10 %, placing 512 near the knee of the curve, which Figure 4 shows (Figure 4).

---

[5] Source: Compiled by the author.

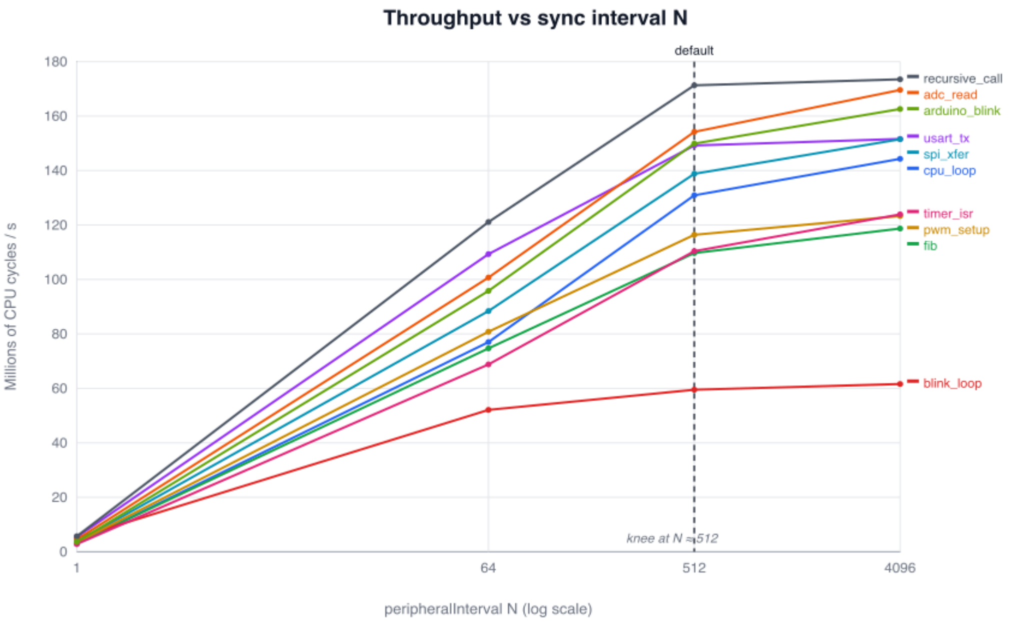

Figure 4. Throughput versus `peripheralInterval` N ∈ {1, 64, 512, 4096} for the ten workloads. Each curve rises steeply to N = 64, bends by N = 512, and is nearly flat thereafter, the empirical knee that justifies the default[6]

The drift benchmark validates correctness at the default. At N = 512, state drift is exactly zero on every state-diff workload, including the interpolated timer-polling path, across all 67 fields at all checkpoints. Event drift is bounded by N as predicted, with a worst case of 258 cycles for interrupt dispatch and 79 cycles for USART completion (means of 128 and 40). During its preparation the benchmark surfaced a real one-LSB TCNT0 defect under a mid-batch prescaler change, fixed by flushing pending ticks under the prior configuration before any timer-register write; the zero-drift result is post-fix.

Table 3 reports the fault detector across the ten categories: macro F1 is 0.809 and macro recall is 0.922. Five codes reach F1 ≥ 0.85, two reach F1 ≥ 0.80, and three (`led_overcurrent`, `chip_overcurrent`, `short_circuit`) fall to 0.54-0.57. These three share a precision ceiling: each detection is a physically real fault, but the benchmark labels only the primary fault per circuit, so correct co-detections (an unprotected LED simultaneously exceeds the LED, per-pin, and per-chip current limits) count as false positives. The `short_circuit` shortfall is structural, caused by a pushbutton component that lacks an electrical model, and the single `overvoltage` miss occurs where a series resistor drops a 9 V rail to a safe level. Recall is high across all codes; precision is the axis with remaining headroom (Table 3).

Table 3. Fault-detector evaluation across ten categories. Macro is the unweighted mean across codes; micro pools the counts before computing[7]

Fault code	TP	FP	FN	TN	Precision	Recall	F1
`led_no_resistor`	4	0	0	56	1.000	1.000	**1.000**
`led_dim`	2	0	0	58	1.000	1.000	**1.000**
`led_reverse`	3	0	0	57	1.000	1.000	**1.000**
`gpio_overcurrent`	4	0	1	55	1.000	0.800	**0.889**
`floating_input`	3	1	0	56	0.750	1.000	**0.857**
`resistor_overpower`	3	1	0	56	0.750	1.000	**0.857**
`overvoltage`	2	0	1	57	1.000	0.667	**0.800**
`led_overcurrent`	4	6	0	50	0.400	1.000	0.571
`chip_overcurrent`	2	3	0	55	0.400	1.000	0.571

---

[6] Source: Compiled by the author.
[7] Source: Compiled by the author.

Fault code	TP	FP	FN	TN	Precision	Recall	F1
`short_circuit`	3	4	1	52	0.429	0.750	0.545
Macro average					**0.773**	**0.922**	**0.809**
Micro pooled	30	15	3	552	**0.667**	**0.909**	**0.769**

The throughput advantage follows from the scheduler, the unified data-space buffer, the fast-path bypass, and hook-based rather than polled peripheral delivery, in roughly that order of impact. The margin is workload-dependent: it is smallest (1.39×) on `pwm_setup`, where avr8js incurs no side effect on the hot register and both engines approach a pure-CPU floor, and largest (4.58×) on `timer_isr`, where roughly 39,000 interrupt dispatches expose the per-event cost difference. The conformance divergences are confined to peripheral reads and reflect a mix of one genuine defect (TIFR0 flag latching), two underspecified hardware behaviors on which the two references themselves disagree, and one deliberate fault-detection default; none affect CPU state. The scheduler's correctness rests on monotone peripherals being interpolable, which the drift benchmark confirms at zero state drift and bounded event drift; the formal argument appears in the companion paper [7].

Logiclabuz targets interactive learning and protocol-level prototyping, not analog design or sub-cycle-deterministic verification; the DC solver models steady-state resistive and diode behavior and omits transient, inductive, and AC phenomena. The principal threats to validity are engine and host bias: V8 JIT behavior makes absolute throughput host-dependent (an independent re-run varied within ±4 %), so the stable Logiclabuz-to-avr8js ratio is the safer figure to cite; the workloads are microbenchmarks on a single host; and the fault benchmark uses hand labels rather than a SPICE ground truth, so its figures are category-level recall on engineered cases.

The architecture is not AVR-specific. The data-space layout, the scheduler, the worker isolation, the topology binder, and the fault detector are chip-agnostic; porting to an ESP32 or RISC-V target requires a new decoder and chip subclass and reuses the rest. We claim novelty not for any single component but for their combination: a cycle-accurate browser MCU, a DC analog model with a pedagogical fault layer, topology-inferred multi-MCU binding, and a lazy-synchronization scheduler with interpolated reads, deployed and in continuous use.

CONCLUSIONS AND SUGGESTIONS

We presented Logiclabuz, a browser-native co-simulation platform that virtualizes multi-microcontroller robotic systems through a cycle-accurate AVR simulator, a DC circuit solver with a pedagogical fault layer, a library of more than fifty modeled components, a topology-inferred multi-MCU binder, and a time-travel debugger. A unified data-space layout and a two-tier scheduler with interpolated timer reads deliver a 1.39 to 4.58 times throughput advantage over avr8js and an 11.7 to 43.6 times internal speedup over per-cycle evaluation; conformance against simavr and avr8js is bit-for-bit on the CPU instruction subset, and the fault detector attains a macro F1 of 0.81. The immediate roadmap is to resolve the four peripheral conformance divergences, add the missing electrical model that limits short-circuit recall, and provide DWARF-based source-level debugging. Extension to ESP32 and RISC-V targets and a classroom outcome study, drawing on the deployed user base at logiclab.uz, are the longer-term directions.

References:

1. J.-M. Pollet, "Simavr: A lean, mean and hackable AVR simulator for linux & co." https://github.com/buserror/simavr.
2. U. Wasserman, "avr8js: AVR 8-bit simulator written in TypeScript." https://github.com/wokwi/avr8js.
3. S. Garcia, "SimulIDE: A real-time electronic circuit simulator." https://www.simulide.com.
4. Autodesk, "TinkerCad circuits." https://www.tinkercad.com/circuits.
5. Labcenter Electronics, "Proteus design suite — VSM." https://www.labcenter.com.
6. I. Y. Abdullaeva and K. Khurana, "The impediments to the process of implementing robotics in the school education system in Uzbekistan," *SAGE Open*, vol. 14, no. 2, 2024, DOI: 10.1177/21582440241254595.
7. A. Ergashev, "Lazy-synchronization with interpolated state for discrete-continuous co-simulation of microcontroller-driven circuits." 2026.
8. U. Wasserman, "Wokwi: Online IoT and microcontroller simulator." https://wokwi.com.
9. L. W. Nagel, "SPICE: A computer program to simulate semiconductor circuits," University of California, Berkeley, Memorandum No. ERL-M520, 1975.
10. G. S. Wolffe, W. Yurcik, H. Osborne, and M. A. Holliday, "Teaching computer organization/architecture with limited resources using simulators," in *Proceedings of the 33rd SIGCSE technical symposium on computer science education (SIGCSE 2002)*, Cincinnati, KY, USA: ACM, 2002, pp. 176-180. DOI: 10.1145/563340.563408.
11. J. Lima, D. A. Gomes, J. E. Souto, and T. F. Vieira, "Use of Tinkercad platform for teaching electronics subject in post-

secondary technical courses," in *Proceedings of the 9th international conference on technological ecosystems for enhancing multiculturality (TEEM 2021)*, Barcelona, Spain: ACM, 2021, pp. 670-675. DOI: 10.1145/3486011.3486517.

12. J. Kholkhujaev, F. Khusnuddinov, and S. Kengesbayeva, "Review of industrial robot and robotic simulator market and available education platforms for robotics learning in Uzbekistan," *Acta of Turin Polytechnic University in Tashkent*, vol. 14, no. 2, pp. 61-64, 2024, Available: https://acta.polito.uz/index.php/journal/article/view/287
13. P. Galkin, R. Umiarov, and O. Grigorieva, "Design embedded system testbench based on FPGA and microcontrollers for TATU smart lab as education component of Industry 4.0," in *Proceedings of the 2019 IEEE 2nd ukraine conference on electrical and computer engineering (UKRCON)*, 2019, pp. 628-633. DOI: 10.1109/UKRCON.2019.8879996.
14. R. O'Callahan, C. Jones, N. Froyd, K. Huey, A. Noll, and N. Partush, "Engineering record and replay for deployability," in *Proceedings of the 2017 USENIX annual technical conference (USENIX ATC '17)*, Santa Clara, CA, USA: USENIX Association, 2017, pp. 377-389.

muhandislik & iqtisodiyot

ijtimoiy-iqtisodiy, innovatsion texnik,
fan va ta’limga oid ilmiy-amaliy jurnal

Ingliz tili muharriri: Feruz Hakimov

Musahhih: Zokir Alibekov

Sahifalovchi va dizayner: Abdurahmon Qurbonov

2026. № 6

“Muhandislik va iqtisodiyot” jurnali 26.06.2023-yildan
O‘zbekiston Respublikasi Prezidenti Adminstratsiyasi huzuridagi
Axborot va ommaviy kommunikatsiyalar agentligi tomonidan
№S–5669245 reyestr raqami tartibi bo‘yicha ro‘yxatdan o‘tkazilgan.
Litsenziya raqami: №095310.

Manzilimiz: Toshkent shahri Yunusobod
tumani 15-mavze 19-uy

☎ +998 93 718 40 07 https://muhandislik-iqtisodiyot.uz/index.php/journal t.me/yait_2100