SMARTICS Journal, Vol.11 No.02 2025. p45-52
ISSN online: 2476-9754, ISSN print: 2623-0429
DOI : https://doi.org/10.21067/smartics.v11i2.13749

Analisis Komparatif Efektivitas dan Kapabilitas Implementasi Laboratorium *Virtual Internet of Things* (IoT) Berbasis Emulasi Mikrokontroler dengan Dukungan MicroPython

San Vitores Jemalut[a], Muhammad Priyono Tri Sulistyanto[b*], Moh Ahsan[c]
[a]Program Studi Teknik Informatika, Universtas PGRI Kanjuruhan Malang, Malang, Indonesia
[*]correspondence email : m.priyono.ts@unikama.ac.id

Abstract*—This research analyzes the implementation and effectiveness of virtual laboratories in Internet of Things (IoT) education through microcontroller emulation, specifically with MicroPython support. Physical infrastructure constraints and operational costs have prompted a shift in learning paradigms toward virtual solutions. This paper compares the capabilities of two primary emulator categories—online solutions (Case Study: Wokwi) and offline solutions (Case Study: QEMU/Espressif Fork)—based on accessibility, emulation fidelity, and pedagogical effectiveness. The results indicate that online emulators like Wokwi excel in accessibility and ease of end-to-end integration (Wi-Fi simulation), which has been proven to yield significant improvements in student competency (average N-Gain score of 74.6%) in academic case studies. Functional case study implementation using Wokwi with Thingsboard demonstrates effectiveness in illustrating end-to-end IoT workflows from virtual sensors to the cloud via the MQTT protocol. Conversely, QEMU provides deeper register-level emulation accuracy, establishing it as a vital instrument for advanced firmware development and debugging. A hybrid implementation model and the use of free cloud platforms are recommended for introductory-level education.

***Index Terms*— Virtual Laboratory; Internet of Things; MicroPython; Microcontroller Emulation; N-Gain Score**

***Abstrak*—Penelitian ini menganalisis implementasi dan efektivitas laboratorium virtual (Virtual Lab) dalam pendidikan Internet of Things (IoT) melalui emulasi mikrokontroler, khususnya dengan dukungan bahasa pemrograman MicroPython. Keterbatasan infrastruktur fisik dan biaya operasional mendorong pergeseran paradigma pembelajaran menuju solusi virtual. Makalah ini membandingkan kapabilitas dua kategori emulator utama—solusi online (Studi Kasus: Wokwi) dan solusi offline (Studi Kasus: QEMU/Espressif Fork)—berdasarkan faktor aksesibilitas, fidelitas emulasi, dan efektivitas pedagogis. Hasil menunjukkan bahwa emulator online seperti Wokwi unggul dalam aspek aksesibilitas dan kemudahan integrasi end-to-end (Simulasi Wi-Fi), yang terbukti menghasilkan peningkatan kompetensi peserta didik yang signifikan (N-Gain score rata-rata 74,6%) dalam studi kasus akademik. Implementasi studi kasus menggunakan Wokwi dengan Thingsboard membuktikan efektivitas dalam mendemonstrasikan alur kerja IoT dari sensor virtual ke cloud melalui protokol MQTT. Sementara itu, QEMU menawarkan akurasi emulasi tingkat register yang lebih dalam, menjadikannya instrumen penting untuk pengembangan dan debugging firmware tingkat lanjut. Disarankan sebuah model implementasi hibrida dan pemanfaatan platform cloud gratis untuk pembelajaran tingkat pengantar..**

***Kata Kunci*— Laboratorium Virtual; Internet of Things; MicroPython; Emulasi Mikrokontroler; N-Gain**

I. PENDAHULUAN

Domain *Internet of Things* (IoT) telah bertransisi menjadi pilar fundamental yang mendorong revolusi Industri 4.0. Penguasaan kompetensi interdisipliner yang mencakup pemrograman sistem embedded, komunikasi jaringan (Wi-Fi, MQTT), dan integrasi cloud adalah esensial [1]. Meskipun demikian, penyediaan fasilitas pendidikan IoT secara konvensional menghadapi kendala signifikan, utamanya terkait biaya pengadaan, pemeliharaan perangkat keras (misalnya, mikrokontroler ESP32, sensor, dan aktuator), serta masalah skalabilitas untuk memenuhi rasio perangkat per siswa yang memadai. Keterbatasan infrastruktur fisik ini, sebagaimana didokumentasikan dalam studi kasus spesifik, menyoroti defisit

ketersediaan perangkat praktikum IoT dan mitigasi pemahaman konseptual siswa sebagai permasalahan prioritas yang harus diatasi dalam konteks pengajaran kejuruan [2] .

Oleh karena itu, adopsi laboratorium virtual menjadi imperatif strategis. Laboratorium virtual menawarkan skalabilitas tak terbatas, pengurangan biaya operasional, dan peningkatan aksesibilitas, secara efektif memitigasi hambatan yang timbul dari infrastruktur fisik konvensional.

Dalam ranah sistem embedded, perbedaan antara emulasi dan simulasi adalah krusial. Emulasi didefinisikan sebagai replikasi akurat perilaku Central Processing Unit (CPU), memori, dan periferal, yang memungkinkan firmware asli dieksekusi dalam lingkungan virtual. Sebaliknya, simulasi lebih berfokus pada replikasi fungsionalitas sistem pada tingkat logis.

Untuk mencapai validitas pedagogis yang tinggi dalam pembelajaran IoT, emulasi wajib bersifat tingkat lanjut. Simulasi I/O statis tidak memadai untuk mencapai tujuan pembelajaran yang berfokus pada konektivitas. Esensi IoT adalah komunikasi, sehingga kemampuan emulator untuk mereplikasi fitur konektivitas kritis, seperti simulasi Wi-Fi, menjadi faktor kausal [2], [3].

Ketersediaan simulasi Wi-Fi, yang merupakan fitur unggulan dari simulator online mutakhir, memfasilitasi peragaan aliran data ujung ke ujung (End-to-End Data Flow). Hal ini secara langsung memungkinkan praktik integrasi dengan platform IoT eksternal (misalnya, Thingsboard). Kemampuan untuk memverifikasi koneksi cloud tanpa kompleksitas konfigurasi jaringan fisik adalah validasi utama efektivitas pedagogis virtual lab. Dengan mengeliminasi hambatan networking fisik, peserta didik dapat mengalihkan fokus ke lapisan protokol aplikasi (MQTT atau HTTP), yang merupakan inti dari rekayasa sistem IoT [4].

II. TINJAUAN PUSTAKA

MicroPython, sebuah implementasi bahasa Python 3 yang dioptimalkan untuk lingkungan mikrokontroler, memegang peranan vital dalam mempercepat kurva pembelajaran di bidang IoT. Dengan mereduksi kompleksitas sintaksis dari bahasa pemrograman tingkat rendah seperti C/C++, MicroPython memungkinkan peserta didik untuk berkonsentrasi pada logika IoT, desain sistem, dan interaksi perangkat keras.

MicroPython memiliki portabilitas yang ekstensif, mendukung hampir seluruh arsitektur mikrokontroler IoT utama, termasuk seri ESP32, ESP8266, RP2040 (Raspberry Pi Pico), dan STM32 [5]. Fleksibilitas ini menjamin bahwa kompetensi yang diperoleh dalam laboratorium virtual menggunakan MicroPython bersifat transferable dan dapat diterapkan pada beragam hardware fisik.

Fitur inti MicroPython adalah Read-Evaluate-Print Loop (REPL), yang memfasilitasi interaksi real-time dengan mikrokontroler [6]. REPL sangat penting untuk proses debugging interaktif. Oleh karena itu, setiap lingkungan emulasi yang digunakan harus mampu mereplikasi shell REPL ini, baik melalui port serial virtual maupun mekanisme berbasis jaringan

Mikrokontroler keluarga ESP dari Espressif, khususnya ESP32, telah menjadi standar de facto dalam prototyping dan pengembangan akademik IoT, didukung oleh integrasi fitur Wi-Fi dan Bluetooth [5], [6]. Arsitektur ini, yang umumnya berbasis CPU Xtensa, menyediakan platform yang efisien dan hemat biaya.

Implementasi MicroPython pada perangkat ini memerlukan flashing firmware MicroPython spesifik (.bin file) ke memori mikrokontroler. Proses ini menciptakan lapisan runtime yang berbeda dari firmware berbasis C/C++ (misalnya Arduino atau Espressif IDF). Dengan demikian, emulator harus mampu memuat dan mengeksekusi konteks runtime MicroPython secara akurat.

Lingkungan REPL adalah fitur yang membedakan pengalaman pemrograman MicroPython dari tradisi embedded sistem konvensional. Akses REPL secara fisik biasanya terjadi melalui port serial (UART0), terhubung ke GPIO1 (TX) dan GPIO3 (RX) dengan baudrate 115200 . Emulator harus menyediakan terminal virtual yang memetakan output UART ini, memungkinkan eksekusi perintah Python secara interaktif, yang ditandai dengan prompt >>> .

Selain REPL serial, MicroPython mendukung WebREPL, yang memfasilitasi akses prompt melalui jaringan Wi-Fi, diakses via peramban. Fitur WebREPL ini memiliki relevansi tinggi untuk virtual lab online. Emulator berbasis cloud yang memiliki kapabilitas simulasi jaringan, seperti Wokwi, dapat meniru fungsi WebREPL. Interaksi terminal dapat dilakukan langsung melalui browser tanpa memerlukan konfigurasi jaringan lokal, memberikan pengalaman REPL yang autentik dalam lingkungan cloud [5], [7].

III. METODE

Penelitian ini menggunakan metode tinjauan pustaka naratif [8] (narrative review atau traditional review) untuk mengidentifikasi, menilai, dan menginterpretasikan literatur yang relevan mengenai laboratorium virtual IoT berbasis MicroPython. Berbeda dengan tinjauan sistematis yang kaku, metode naratif memberikan fleksibilitas untuk mendiskusikan perkembangan teknologi, membandingkan berbagai platform emulasi, dan memberikan interpretasi holistik terhadap tren penelitian terbaru tanpa protokol pencarian literatur yang ketat.

Literatur dikumpulkan melalui penelusuran pada pangkalan data akademik seperti Google Scholar, IEEE Xplore, dan Science Direct menggunakan kata kunci yang dinamis, antara lain: "IoT virtual lab" atau “Laboratorium virtual”, "MicroPython emulation", "Wokwi", dan "QEMU ESP32" . Pemilihan artikel didasarkan pada relevansi tematik terhadap implementasi teknis mikrokontroler virtual dan efektivitasnya dalam lingkungan pendidikan . Artikel yang dipilih mencakup studi eksperimental, laporan teknis, dan dokumentasi resmi dari pengembang emulator yang diterbitkan dalam rentang waktu sepuluh tahun terakhir.

Data yang diperoleh dari berbagai sumber disintesis secara kualitatif berdasarkan tema-tema utama yang muncul dalam narasi pengembangan IoT . Fokus analisis meliputi aspek aksesibilitas platform (online vs. offline), fidelitas emulasi terhadap perangkat keras asli, serta kemampuan dukungan konektivitas Wi-Fi yang krusial bagi sistem IoT . Penulis memberikan interpretasi kritis terhadap kelebihan dan kekurangan masing-masing metode berdasarkan data sekunder yang tersedia.

Meskipun artikel ini bersifat ulasan naratif, untuk memperkuat argumen mengenai efektivitas pedagogis, penelitian ini juga menginkoporasi analisis data sekunder berupa hasil eksperimen pendidikan yang diukur melalui Normalized Gain (N-gain) score . Metode ini digunakan untuk memvalidasi narasi mengenai peningkatan kompetensi siswa setelah menggunakan alat emulasi tertentu dalam proses pembelajaran .

IV. HASIL DAN PEMBAHASAN

4.1. Implementasi Laboratorium Virtual Berbasis Emulator Online (Studi Kasus Wokwi)

Wokwi berfungsi sebagai simulator elektronik online berbasis cloud yang dieksekusi langsung di peramban web, menghilangkan kebutuhan akan instalasi perangkat lunak lokal yang kompleks [9]. Hal ini menghasilkan learning curve yang rendah dan tingkat aksesibilitas yang unggul.

Wokwi mendukung spektrum emulasi yang luas, termasuk platform populer seperti ESP32, ESP8266, Arduino, STM32, dan Raspberry Pi Pico [9]. Platform ini secara eksplisit mendukung simulasi proyek MicroPython. Selain CPU, Wokwi mensimulasikan berbagai periferal seperti layar, sensor, dan motor.

Diferensiator kunci Wokwi dalam kerangka pembelajaran IoT adalah kapabilitas simulasi Wi-Fi terintegrasi. Kemampuan ini vital dalam memfasilitasi koneksi proyek yang diemulasikan ke jaringan eksternal. Simulasi jaringan memungkinkan peragaan konsep IoT yang melibatkan komunikasi eksternal secara fungsional, contohnya ESP32 NTP Clock atau MicroPython MQTT Weather Logger.

Simulasi Wi-Fi ini juga merupakan prasyarat teknis untuk integrasi dengan platform IoT cloud seperti Thingsboard. Berdasarkan studi kasus akademis, Wokwi digunakan secara efektif untuk praktik IoT berbasis ESP32 dan Thingsboard. Kemampuan ini memungkinkan siswa melewati kompleksitas setup jaringan fisik dan langsung memfokuskan studi pada logika aplikasi dan lapisan protokol IoT. Dengan demikian, Wokwi mempercepat pemahaman konsep IoT end-to-end dengan mengeliminasi hambatan teknis perangkat keras jaringan.

Wokwi tidak bersifat monolithic; platform ini menawarkan ekstensi untuk Visual Studio Code (VS Code). Integrasi ini memungkinkan pengembang dan siswa untuk menulis kode secara lokal dalam lingkungan IDE profesional yang sudah familiar, sambil tetap memanfaatkan mesin simulasi cloud Wokwi. Pendekatan blended ini mengkombinasikan aksesibilitas cloud dengan efisiensi alur kerja pengembangan lokal.

Skenario implementasi fungsional di laboratorium virtual berfokus pada peragaan pengiriman data sensor (suhu/kelembapan) dari mikrokontroler ESP32 yang diemulasikan ke platform cloud Thingboard, memanfaatkan MicroPython dan protokol Message Queuing Telemetry Transport (MQTT).

Proses ini dimulai di lingkungan emulasi online (Wokwi) dengan menyiapkan: (1) Mikrokontroler Virtual (ESP32), (2) Periferal Virtual (Sensor DHT22 virtual), dan (3) Konfigurasi Jaringan Virtual. Keunggulan simulasi Wi-Fi pada Wokwi memungkinkan proyek diemulasikan untuk terhubung ke internet yang disimulasikan dan berinteraksi dengan layanan cloud eksternal.

Inti dari implementasi adalah skrip MicroPython (main.py) yang menjalankan fungsi-fungsi krusial:

Inisialisasi Wi-Fi Virtual: (1) Menginisialisasi koneksi jaringan menggunakan kredensial yang diemulasikan, (2) Membaca data suhu dan kelembapan dari sensor virtual, (2) Pengiriman Data MQTT: Menggunakan pustaka MicroPython MQTT untuk terhubung ke broker Thingsboard dan mengirimkan data sensor ke endpoint telemetri. Pada saat kode berjalan di emulator, peserta didik berinteraksi dengan dashboard Thingsboard. Langkah-langkah integrasi meliputi: (1) Registrasi Perangkat di Thingsboard untuk mendapatkan Access Token yang berfungsi sebagai kredensial MQTT, dan (2) Penerimaan dan Visualisasi Data dengan membuat dashboard visual (Widget) yang menampilkan grafik real-time dari data yang dikirimkan oleh ESP32 virtual.

Model implementasi ini berhasil mendemonstrasikan alur kerja IoT end-to-end—dari pengambilan data sensor di perangkat embedded (ditemulasikan) hingga visualisasi di cloud—tanpa memerlukan perangkat keras fisik, memberikan bukti nyata efektivitas pedagogis virtual lab.

4.2 Implementasi Laboratorium Virtual Berbasis Emulator Offline (Studi Kasus QEMU)

Quick Emulator (QEMU) adalah emulator open-source yang memfasilitasi emulasi penuh sistem (full-system emulation), mencakup CPU dan periferal. Espressif, produsen mikrokontroler ESP32, memelihara fork QEMU spesifik yang dirancang untuk mendukung arsitektur ESP32. Fork ini secara akurat mengimplementasikan emulasi CPU Xtensa, memori, dan beberapa periferal utama ESP32 [3].

QEMU untuk ESP32 diorientasikan untuk debugging tingkat register dan pengujian firmware yang mendalam. Walaupun MicroPython bytecode dapat dieksekusi di atas emulasi CPU/memori QEMU, fokus utama dukungan Espressif adalah pada debugging aplikasi yang dikembangkan menggunakan Espressif IoT Development Framework (ESP-IDF) berbasis C/C++.

Pemanfaatan QEMU untuk ESP32 terikat erat dengan ekosistem pengembangan ESP-IDF. Proses instalasi dan konfigurasi QEMU jauh lebih kompleks dibandingkan emulator berbasis browser. Pengguna diwajibkan menginstal toolchain dan dependensi sistem spesifik (misalnya, libgcrypt20, libglib2.0-0, dan libslirp0 pada sistem Linux) sebelum dapat menggunakan biner QEMU yang telah dikompilasi.

Ketergantungan pada toolchain lokal dan prasyarat instalasi yang ekstensif ini menimbulkan hambatan pedagogis yang signifikan untuk laboratorium pengantar. Meskipun QEMU adalah perangkat yang kuat, kompleksitas setup ini menempatkannya sebagai instrumen penelitian dan pengembangan (R&D) firmware daripada sebagai alat pembelajaran massal yang mudah diakses. Eksekusi aplikasi di QEMU dilakukan menggunakan perintah idf.py qemu monitor, yang menghubungkan konsol IDF Monitor ke port UART yang diemulasikan.

QEMU unggul dalam emulasi deep pada lapisan register. Emulator ini mendukung pengujian I2C tanpa perangkat keras fisik melalui emulasi sensor I2C virtual, seperti sensor TMP105. Kapabilitas untuk memanipulasi keadaan hardware virtual secara langsung merupakan kekuatan utama QEMU yang memfasilitasi pengujian skenario ekstrem dan validasi driver perangkat keras yang kritis.

4.3 Analisis Komparatif Mendalam Emulator IoT (Online vs. Offline)

Perbandingan terstruktur antara paradigma online dan offline adalah esensial untuk mengidentifikasi instrumen yang paling tepat untuk tujuan pembelajaran spesifik. Emulator online (Wokwi) menawarkan aksesibilitas dan kemudahan penggunaan yang superior. Pengguna hanya memerlukan peramban web, menghilangkan dependensi pada sistem operasi lokal dan kebutuhan instalasi. Sebaliknya, emulator offline (QEMU) membutuhkan instalasi toolchain yang spesifik dan kompleks.

Meskipun QEMU adalah open source, Total Cost of Ownership (TCO) QEMU dapat lebih tinggi di lingkungan akademik karena kebutuhan akan dukungan teknis instruktur yang substansial untuk mengelola lingkungan toolchain yang beragam.

Terdapat dikotomi fungsional dalam fidelitas emulasi antara kedua paradigma. Wokwi menyediakan fungsionalitas yang broad (luas), berfokus pada visualisasi dan pengalaman pengguna, dengan simulasi komponen visual dan I/O. Secara kritis, Wokwi secara eksplisit mendukung simulasi Wi-Fi dan koneksi real-time ke platform IoT eksternal, yang fundamental untuk simulasi konsep IoT penuh. Sebaliknya, QEMU menawarkan fungsionalitas yang deep (dalam). Fokusnya adalah pada emulasi CPU, memori, dan register secara akurat. Meskipun cakupan periferal yang diemulasikan (misalnya, sensor I2C virtual) lebih terbatas, tingkat akurasinya pada lapisan hardware jauh lebih tinggi. QEMU lebih berorientasi pada pengujian firmware inti, dan simulasi jaringan tingkat tinggi untuk koneksi cloud eksternal lebih menantang dibandingkan lingkungan online. Tabel 1 berikut menyajikan ringkasan perbandingan teknis dan operasional antara emulator *online* dan *offline* dalam konteks pembelajaran IoT berbasis MicroPython.

Tabel 1 Perbandingan Teknis Spesifik (Wokwi vs. QEMU)

Aspek Komparasi	Emulator Online (Contoh: Wokwi)	Emulator Offline (Contoh: QEMU)
Aksesibilitas	Sangat Tinggi (Hanya butuh peramban/browser), tidak tergantung OS lokal.	Rendah hingga Menengah (Membutuhkan instalasi perangkat lunak dan *toolchain* spesifik).
Kebutuhan Hardware Lokal	Sangat Rendah (Ringan, semua komputasi di *cloud*).	Tinggi (Membutuhkan CPU dan RAM yang cukup untuk menjalankan simulasi/VM).
Simulasi Jaringan (Iot)	Unggul (Simulasi WiFi/Koneksi *real-time* ke broker MQTT/Platform IoT dimungkinkan).	Terbatas (Membutuhkan *networking* kompleks/virtualisasi, fokus pada *low-level stack*).
Debugging Tingkat Rendah	Terbatas (Fokus pada kode tingkat tinggi/MicroPython REPL).	Sangat Unggul (Akses penuh ke CPU, memori, register, GDB).
Kurva Pembelajaran	Landai (Intuitif, desain grafis).	Curam (Membutuhkan pemahaman arsitektur sistem operasi dan *toolchain*).
Dukungan Micropython (Eksekusi)	Penuh (Langsung melalui browser).	Tidak langsung (Membutuhkan flashing firmware MicroPython ke emulasi hardware).
Jenis Emulasi	Simulasi Fungsional dan Logika Perangkat Keras.	Emulasi Penuh Sistem (CPU, Memori, Register).
Simulasi Periferal	Luas (Sensor populer, display, motor, I/O digital/analog).	Terbatas namun deep (CPU, Memori, UART, I2C virtual sensor TMP105).
Integrasi Ide	VS Code Extension tersedia.	Terintegrasi erat dengan Espressif IDF melalui idf.py qemu monitor.
Kesesuaian Target Pengguna	Pendidikan, Prototyping Cepat.	Pengembang Firmware, Debugging Sistem Kritis, Riset Teknis.

4.3. Analisis Komparatif Platform Cloud IoT: Gratis vs. Berbayar

Pemilihan platform cloud merupakan komponen integral dalam desain Laboratorium Virtual IoT. Platform cloud diklasifikasikan berdasarkan model biaya, yang secara signifikan memengaruhi fitur, skalabilitas, dan kompleksitas implementasi.Platform gratis (misalnya, Thingsboard Community Edition, ThingSpeak, Blynk, Ubidots Free Tier) ideal untuk konteks pendidikan, prototyping cepat, dan proyek skala kecil [10]. Perbandingan kelebihan dan kekurangan dari penggunaan platform cloud IoT dapat dilihat pada Tabel 2

Tabel 2 Kelebihan dan kekurangan platform cloud IoT gratis

Kelebihan	Kekurangan atau batasan
Biaya Nol: Eliminasi biaya awal, cocok untuk eksperimen dan lingkungan akademik	Keterbatasan Fitur: Tidak mencakup fitur canggih seperti Advanced Role-Based Access Control (RBAC), Single Sign-On (SSO), atau Reporting yang dijadwalkan
Fleksibilitas Thingsboard CE: Bersifat open-source, fleksibel dalam protokol (MQTT, CoAP, HTTP), dan mendukung deployment lokal (self-managed)	Pembatasan Skala: Keterbatasan jumlah perangkat, volume data, dan laju pengiriman data (misalnya, batasan pengiriman data per menit pada ThingSpeak)
Kemudahan Penggunaan: Platform seperti Blynk menawarkan antarmuka yang sangat intuitif, memfasilitasi visualisasi data secara cepat	Dukungan Teknis Terbatas: Ketersediaan dukungan teknis yang terbatas

Sedangkan platform berbayar (misalnya, Thingsboard Professional Edition, AWS IoT Core, Azure IoT Hub) dirancang untuk memenuhi kebutuhan skalabilitas industri, keamanan tingkat lanjut, dan integrasi sistem yang kompleks, komparasi antara kelebihan dari platform IoT berbayar dapat ditampilkan pada Tabel 3.

Tabel 3 Kelebihan dan kekurangan platform cloud IoT berbayar atau profesional

Kelebihan	Kekurangan dan Kompleksitas
Skalabilitas dan Keandalan Industri: Keandalan tingkat industri dan integrasi mulus dengan ekosistem layanan cloud yang lebih luas (misalnya, AWS Lambda)	Biaya Tinggi: Membutuhkan biaya lisensi dan operasional yang substansia
Fitur Lanjutan: Menawarkan fitur-fitur seperti Advanced RBAC, SSO, White-Labeling, Reporting & Scheduling, dan Solution Templates	Kurva Pembelajaran Curam: Kompleksitas yang tinggi, memerlukan pemahaman mendalam tentang arsitektur cloud, kebijakan keamanan (IAM/RBAC), dan model penetapan harga, yang kurang sesuai untuk lingkungan pembelajaran pengantar
Integrasi Sistem: Mendukung integrasi sistem eksternal yang kompleks (misalnya, AWS IoT, Azure IoT, Kafka)	

Untuk Laboratorium Virtual tingkat pengantar yang berfokus pada MicroPython dan emulasi (seperti Wokwi), penggunaan platform cloud gratis, khususnya Thingsboard Community Edition, sangat direkomendasikan. Hal ini memungkinkan peserta didik untuk memfokuskan upaya pada logika coding dan protokol IoT (MQTT) tanpa terhambat oleh kompleksitas konfigurasi akun cloud berbayar atau biaya finansial. Platform berbayar sebaiknya dicadangkan untuk modul spesialisasi yang mensimulasikan skenario enterprise tingkat lanjut (misalnya, stress test skalabilitas atau integrasi backend).

4.3. Evaluasi Kinerja dan Dampak Pedagogis Virtual Lab

Efektivitas implementasi virtual lab dievaluasi menggunakan metodologi standar akademik berupa pre-test dan post-test, yang hasilnya dihitung melalui Normalized Gain (N-gain) score. Analisis N-gain digunakan untuk mengukur keefektifan sebuah perlakuan pembelajaran dalam meningkatkan pemahaman konsep peserta didik dengan membandingkan gain yang diperoleh dengan gain maksimum yang mungkin didapat. Metode N-gain score dihitung menggunakan rumus yang diperkenalkan oleh Hake [11] sebagai berikut:

$$g = \frac{S_{post} - S_{pre}}{S_{max} - S_{pre}}$$

Dimana g= Normalized Gain (N-gain), S_{post} = skor rata-rata post-test, S_{pre} = skor rata-rata pre-test, dan S_{max} = skor maksimum ideal. Hasil perhitungan N-gain kemudian diinterpretasikan ke dalam tiga kategori klasifikasi menurut Hake (1998): Tinggi, jika $g > 0.7$; Sedang (Moderate), jika $0.3 \leq g \leq 0.7$; dan Rendah, jika $g < 0.3$.

Studi kasus menunjukkan bahwa intervensi pelatihan yang mengaplikasikan simulasi Wokwi menghasilkan luaran yang signifikan: peningkatan pengetahuan dan kompetensi siswa mengenai Sistem IoT - ESP 32 Thingsboard seperti yang terlihat pada Tabel 4 .Tingkat kemampuan awal peserta pelatihan (nilai rata-rata 33.00) dikategorikan rendah. Pasca-pelatihan, kemampuan peserta mengalami peningkatan substansial dengan nilai rata-rata mencapai 83.00. Efektivitas pelatihan ini terkonfirmasi oleh N-Gain score rata-rata sebesar 74,6% atau 0,746 [2], [4]. Berdasarkan kriteria Hake, skor tersebut termasuk dalam kategori Tinggi (Efektif).Secara spesifik:Materi Pengenalan Thingsboard dan Wokwi mencapai N-Gain 76.0% (Efektif). Materi Pengenalan IoT mencapai N-Gain 71.6% (Efektif). Tingginya N-Gain score, khususnya pada materi yang melibatkan Thingsboard dan Wokwi, mengindikasikan bahwa simulasi berhasil mengeliminasi hambatan terkait perangkat keras dan setup jaringan yang sebelumnya menjadi kendala utama. Hal ini memungkinkan peserta didik untuk secara efisien memfokuskan studi pada abstraksi sistem IoT. Peserta dinyatakan mampu mengimplementasikan sistem IoT berbasis ESP32 dan Thingsboard menggunakan simulasi Wokwi

Tabel 4 Peningkatan Kompetensi Siswa Melalui Simulasi Wokwi

Materi Pelatihan	Nilai Awal (Pre-Test)	Nilai Akhir (Post-Test)	N-Gain Score (%)	Tafsiran Efektivitas
Tren Industri 4.0	33,00	84,00	76,1	Efektif
Pengenalan IoT	40,00	83,00	71,6	Efektif
Pengenalan Thingsboard dan Wokwi	25,00	82,00	76,0	Efektif
Rata-rata Keseluruhan	33,00	83,00	74,6	Efektif

Meskipun emulator menawarkan solusi skalabel, keduanya memiliki batasan inheren. Tantangan implementasi Wokwi terletak pada keterbatasan emulasi untuk periferal yang sangat spesifik atau kustom, serta perilaku timing yang sangat sensitif. Sebaliknya, QEMU menghadapi tantangan utama pada kompleksitas instalasi dan konfigurasi yang tinggi, yang sulit untuk diterapkan dalam skala besar pada lingkungan kelas yang heterogen

V. SIMPULAN

Implementasi laboratorium virtual dalam pembelajaran IoT melalui emulasi mikrokontroler dan MicroPython terbukti sebagai pendekatan yang strategis dan efektif untuk memitigasi keterbatasan perangkat keras fisik dan mengakselerasi kurva pembelajaran. MicroPython merupakan bahasa yang efisien karena dukungan REPL dan kompatibilitasnya yang luas dengan platform IoT dominan.

Wokwi (Online) adalah instrumen yang unggul secara Pedagogis karena aksesibilitas universal dan kapabilitas simulasi jaringan (Wi-Fi) yang memfasilitasi praktik konsep IoT end-to-end yang otentik. Bukti empiris N-Gain score rata-rata 74,6% secara kuat mendukung keefektifan pedagogis Wokwi.

QEMU (Offline) adalah instrumen yang unggul untuk Pengembangan Mendalam dan Riset (R&D) karena emulasi penuh sistem (CPU dan register) dan kemampuan debugging tingkat rendah (GDB) yang didukung oleh fork resmi Espressif.

Untuk mencapai keseimbangan optimal antara efektivitas pedagogis dan kedalaman teknis, direkomendasikan Model Laboratorium Virtual Hibrida (Blended Emulation). Solusi pertama ialah adopsi Wokwi sebagai Komponen Pengantar. Wokwi digunakan untuk modul pengantar dan konseptual, memfokuskan pembelajaran pada skrip MicroPython, REPL, dan integrasi cloud (MQTT/Thingsboard) . Penggunaan platform cloud gratis diutamakan untuk menghilangkan hambatan teknis dan finansial. Sedang solusi kedua adalah pemanfaatan QEMU untuk spesialisasi. QEMU dialokasikan di lingkungan terkontrol untuk modul tingkat lanjut yang menuntut analisis arsitektur, timing kritis, atau debugging firmware tingkat rendah. Solusi terakhir berupa integrasi lingkungan lokal. Dorongan penggunaan IDE lokal seperti Thonny atau VS Code/Wokwi Extension untuk menjembatani manajemen kode online ke deployment offline.

Pengembangan emulator IoT di masa depan diprediksi akan bergerak menuju konvergensi fungsionalitas, dengan emulator online berupaya meningkatkan fidelitas emulasi untuk menandingi akurasi debugging tingkat register yang saat ini menjadi domain alat offline seperti QEMU. Hal ini akan semakin memperkuat peran laboratorium virtual sebagai tulang punggung pendidikan teknologi di era digital.

DAFTAR PUSTAKA

[1] M. P. T. Sulistyanto, D. A. Nugraha, N. Sari, N. Karima, and W. Asrori, “Implementasi IoT (Internet of Things) dalam pembelajaran di Universitas Kanjuruhan Malang,” *SMARTICS Journal*, vol. 1, no. 1, pp. 20–23, 2015.

[2] Rahmawati, S. Amra, Hanafi, Ismaniar, and C. Yusnar, “Simulasi IoT berbasis ESP32 dan Thingsboard Bagi Siswa SMKN 5 Kota Lhokseumawe,” in *Proceeding Seminar Nasional Politeknik Negeri Lhokseumawe*, Lhokseumawe, Mar. 2024.

[3] “QEMU Emulator - ESP32 - — ESP-IDF Programming Guide v5.5.2 documentation.” Accessed: Sep. 01, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/tools/qemu.html

[4] M. N. A. Nur, R. Prajono, Y. A. Koedoes, I. Galugu, and A. Lolok, “Analisis Kebutuhan Sistem Laboratoroium Virtual IoT Terintegrasi SPADA Universitas Halu Oleo,” *Jurnal Fokus Elektroda : Energi Listrik, Telekomunikasi, Komputer, Elektronika dan Kendali)*, vol. 9, no. 4, pp. 218–222, Nov. 2024, doi: 10.33772/JFE.V9I4.986.

[5] “MicroPython - Python for microcontrollers.” Accessed: Sep. 01, 2025. [Online]. Available: https://micropython.org/

[6] "MicroPython tutorial for ESP32 — MicroPython latest documentation." Accessed: Sep. 01, 2025. [Online]. Available: https://docs.micropython.org/en/latest/esp32/tutorial/index.html
[7] "Getting Started with MicroPython on ESP32 and ESP8266 | Random Nerd Tutorials." Accessed: Sep. 01, 2025. [Online]. Available: https://randomnerdtutorials.com/getting-started-micropython-esp32-esp8266/
[8] E. T. Rother, "Systematic literature review X narrative review," *Acta Paulista de Enfermagem*, vol. 20, no. 2, pp. v–vi, 2007, doi: 10.1590/S0103-21002007000200001.
[9] "Welcome to Wokwi! | Wokwi Docs." Accessed: Sep. 01, 2025. [Online]. Available: https://docs.wokwi.com/?utm_source=wokwi
[10] "Rekomendasi Platform IoT Terbaik & Gratis yang Cocok untuk Pemula." Accessed: Jan. 16, 2026. [Online]. Available: https://www.cloudcomputing.id/pengetahuan-dasar/rekomendasi-platform-iot
[11] R. R. Hake, "Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses," *Am. J. Phys.*, 1998, doi: 10.1119/1.18809.