PROPOSAL PROGRAM KREATIVITAS MAHASISWA
SmartDevOS: Sistem Operasi Ringan untuk Akselerasi Pengembangan Smart System Berbasis Arduino

BIDANG KEGIATAN
PKM KARYA CIPTA

Diusulkan Oleh:

1. I Gede Satria Adi Pratama (108072500073)
2. Putri aulia ratna puspani (108072530018)
3. Thania Brova Sastaviana (108072500122)

UNIVERSITAS TELKOM
SURABAYA
2026

DAFTAR ISI

BAB I

PENDAHULAN

1.1 LATAR BELAKANG

Inovasi dan kreativitas anak muda Indonesia dalam bidang teknologi, khususnya dalam pengembangan sistem IoT (Internet of Things) dan robotika, seringkali terhambat oleh keterbatasan sumber daya teknis dan kompleksitas infrastruktur yang diperlukan. Banyak ide cemerlang yang tidak dapat terwujud karena beberapa kendala fundamental.

Permasalahan utama yang dihadapi meliputi :

1. Kompleksitas Instalasi dan Konfigurasi
   - Proses instalasi IDE (Integrated Development Environment) untuk pengembangan sistem embedded sangat rumit dan memakan waktu
   - Penyetelan environment untuk smart monitoring system dengan tunneling memerlukan pengetahuan teknis yang mendalam
   - Konfigurasi yang tidak user-friendly menjadi barrier to entry bagi pemula.
2. Tantangan Pengembangan Smart System Monitoring dengan Tunneling
   - Developer harus mengembangkan panel kontrol dari nol dengan coding manual
   - Konfigurasi konektivitas internet dan pengaturan agar sistem dapat diakses secara public memerlukan expertise dalam network configuration
   - Implementasi sistem keamanan yang robust menambah kompleksitas pengembangan
   - Tidak ada standarisasi dalam pengembangan, sehingga setiap proyek dimulai dari awal.
3. Keterbatasan Sumber Daya
   - Tidak semua developer memiliki akses ke hardware dengan spesifikasi tinggi
   - Biaya pengembangan yang tinggi untuk setup infrastruktur
   - Kurva pembelajaran yang curam untuk menguasai berbagai teknologi yang diperlukan.

Melihat kondisi tersebut, terdapat peluang besar untuk mengembangkan sebuah inovasi yang dapat menyederhanakan proses pengembangan smart system dan robotika. Inovasi yang diusulkan adalah sistem operasi khusus dengan bundled services lengkap yang telah dikonfigurasi dan siap pakai, memungkinkan instalasi hanya dengan beberapa klik saja.

1.2 SOLUSI YANG DIBERIKAN

Sistem operasi yang dikembangkan menawarkan beberapa service terintegrasi:

1. Panel Komunikasi Arduino-PC yang Dapat Dikustomisasi
   - Interface visual untuk komunikasi real-time antara Arduino dan PC

- Customizable dashboard sesuai kebutuhan proyek
- No-code/low-code configuration.

2. IDE Arduino Editor dengan Teknologi AI
   - Editor kode Arduino yang terintegrasi penuh dengan AI assistant
   - Fitur drag-and-drop untuk kerangka kerja dasar (visual programming)
   - Sistem analisis otomatis untuk modul Arduino yang terpasang
   - Auto-detection untuk ISP programmer dan berbagai jenis board Arduino
   - AI dapat menganalisis desain visual (drag-drop) dan menghasilkan kode yang sesuai dengan model yang diminta.
3. Sistem Tunneling Terintegrasi
   - Built-in tunneling service untuk akses remote
   - Kemampuan membagikan akses public dengan mudah tanpa konfigurasi router yang rumit
   - Secure remote access untuk monitoring dan kontrol jarak jauh.
4. Optimasi unutk Perangkat Terbatas
   - OS sangat ringan karena hanya berisi service-service essential
   - Dapat dijalankan pada STB (Set-Top Box) dan perangkat dengan resource terbatas
   - Efficient resource management.
5. Ekosistem Terpadu
   - Semua service berkomunikasi melalui panel terpusat
   - Integrasi seamless antara komponen
   - Single interface untuk multiple functionalities.

1.3 Manfaat dan Dampak

Dengan inovasi ini, diharapkan para anak muda dapat:

1. Menurunkan barrier to entry untuk pengembangan IoT dan robotika
2. Menghemat waktu dan biaya pengembangan
3. Fokus pada kreativitas dan inovasi tanpa terhambat kompleksitas teknis
4. Menggunakan platform ini sebagai foundation untuk menciptakan solusi yang lebih besar dan berdampak
5. Mempercepat proses prototype to production.

1.4 RUMUSAN MASALAH

Berdasarkan latar belakang yang telah diuraikan, penelitian ini akan menjawab pertanyaan-pertanyaan berikut:

1.4.1 Rumusan Masalah Utama

Bagaimana merancang dan mengimplementasikan sistem operasi berbasis IoT yang dapat menyederhanakan proses pengembangan smart system dan robotika untuk mengatasi keterbatasan sumber daya dan kompleksitas teknis yang dihadapi oleh developer muda?

1.4.2Rumusan Masalah Spesifik

1. Bagaimana Desain Sistem

- Bagaimana arsitektur sistem operasi yang optimal untuk mengintegrasikan multiple services (IDE, panel kontrol, tunneling) dalam satu platform yang ringan dan efisien?
- Komponen dan modul apa saja yang harus diintegrasikan dalam sistem operasi untuk memenuhi kebutuhan pengembangan smart system dengan Arduino?

2. Aspek Antarmuka dan Usability
   - Bagaimana merancang interface drag-and-drop yang intuitif untuk memudahkan pemrograman visual Arduino tanpa mengurangi fleksibilitas coding?
   - Bagaimana merancang panel komunikasi Arduino-PC yang customizable namun tetap user-friendly untuk berbagai tingkat keahlian pengguna?
3. Aspek Teknologi AI
   - Bagaimana mengimplementasikan sistem AI yang dapat menganalisis desain visual (drag-drop) dan menghasilkan kode Arduino yang efisien dan sesuai dengan best practices?
   - Bagaimana AI dapat membantu dalam auto-detection dan konfigurasi modul Arduino yang terpasang?
4. Aspek Konektivitas dan Keamanan
   - Bagaimana mengimplementasikan sistem tunneling yang aman, mudah dikonfigurasi, dan dapat diakses secara public tanpa memerlukan pengetahuan mendalam tentang network configuration?
   - Bagaimana memastikan keamanan sistem saat diakses secara remote dan public?
5. Aspek Performa dan Optimasi
   - Bagaimana mengoptimalkan sistem operasi agar dapat berjalan pada perangkat dengan resource terbatas seperti STB?
   - Berapa resource minimum yang dibutuhkan untuk menjalankan sistem secara optimal?
6. Aspek Validasi dan Efektivitas
   - Bagaimana mengukur efektivitas sistem dalam mengurangi kompleksitas pengembangan dibandingkan dengan metode konvensional?
   - Seberapa besar peningkatan produktivitas dan penurunan waktu pengembangan yang dapat dicapai dengan menggunakan sistem ini?

1.5 TUJUAN PENELITIAN

1.5.1 Tujuan Umum

Mengembangkan sistem operasi berbasis IoT yang terintegrasi dengan bundled services untuk menyederhanakan dan mempercepat proses pengembangan smart monitoring system dan robotika berbasis Arduino, sehingga dapat menurunkan barrier to entry bagi developer muda dengan keterbatasan sumber daya.

1.5.2 Tujuan Khusus

1. Pengembangan Arsitektur Sistem

- Merancang arsitektur sistem operasi yang modular, ringan, dan efisien dengan integrasi penuh antara IDE, panel kontrol, dan tunneling service
- Mengidentifikasi dan mengintegrasikan komponen-komponen essential yang dibutuhkan untuk pengembangan smart system berbasis Arduino.

2. Pengembangan Antarmuka dan User Experience
   - Mengembangkan interface drag-and-drop yang intuitif untuk pemrograman visual Arduino dengan library komponen yang lengkap
   - Merancang dan mengimplementasikan panel komunikasi Arduino-PC yang customizable dengan template siap pakai untuk berbagai use case
3. Implementasi Teknologi AI
   - Mengimplementasikan AI engine yang dapat menganalisis desain visual dan menghasilkan kode Arduino yang optimal dan mengikuti coding standards
   - Mengembangkan sistem auto-detection dan auto-configuration untuk berbagai jenis board Arduino dan modul tambahan (ISP, sensors, actuators)
   - Mengintegrasikan AI assistant untuk membantu debugging dan optimization.
4. Implementasi Konektivitas dan Keamanan
   - Mengembangkan tunneling service yang terintegrasi dengan konfigurasi one-click untuk akses remote dan public
   - Mengimplementasikan security layer untuk melindungi sistem dari unauthorized access
   - Menyediakan authentication dan authorization mechanism yang robust namun user-friendly.
5. Optimasi untuk Perangkat Terbatas
   - Mengoptimalkan resource usage agar sistem dapat berjalan pada perangkat dengan spesifikasi minimal (termasuk STB)
   - Melakukan benchmarking untuk menentukan minimum system requirements
   - Mengimplementasikan efficient resource management dan caching mechanisms.
6. Validasi dan Evaluasi Sistem
   - Melakukan pengujian usability dengan target user (developer muda) untuk mengukur kemudahan penggunaan
   - Melakukan comparative study antara pengembangan menggunakan sistem yang dikembangkan vs metode konvensional
   - Mengukur metrik-metrik kunci: waktu setup, waktu development, learning curve, dan kepuasan pengguna
   - Mengidentifikasi area improvement berdasarkan feedback pengguna.
7. Dokumenetasi dan Knowledge Transfer
   - Membuat dokumentasi lengkap untuk user (user manual, tutorials, best practices)
   - Membuat dokumentasi teknis untuk developer yang ingin berkontribusi atau melakukan customization
   - Menyiapkan sample projects dan use cases untuk berbagai scenario (home automation, environmental monitoring, robotics).

1.6 MANFAAT PENELITIAN

1.6.1 Manfaat Teoritis

1. Memberikan kontribusi pada body of knowledge dalam bidang IoT, embedded systems, dan human-computer interaction
2. Mengembangkan framework untuk integrasi AI dalam IDE untuk embedded systems
3. Memberikan model untuk pengembangan specialized operating systems untuk specific use cases.

1.6.2 Manfaat Praktis

1. Untuk Developer/Maker
   - Mengurangi waktu setup dan konfigurasi dari hari/minggu menjadi menit
   - Menurunkan barrier to entry untuk pemula dalam IoT dan robotika
   - Meningkatkan produktivitas pengembangan
   - Memungkinkan fokus pada inovasi dan problem-solving daripada technical setup.
2. Untuk Pendidikan
   - Memudahkan pembelajaran IoT dan robotika di institusi Pendidikan
   - Menyediakan platform standar untuk praktikum dan project-based learning
   - Mengurangi biaya infrastruktur untuk lab computer.
3. Untuk Industri
   - Mempercepat proses prototyping
   - Mengurangi biaya pengembangan
   - Standardisasi development environment untuk tim.
4. Untuk Ekosistem Stratup
   - Memungkinkan rapid prototyping untuk MVP (Minimum Viable Product)
   - Mengurangi technical debt di early stage
   - Mempercepat time-to-market.

1.7 BATASAN PENELITIAN

1. Scope Platform
   - Penelitian ini fokus pada pengembangan sistem untuk Arduino dan board-board compatible
   - Platform IoT lainnya (Raspberry Pi, ESP32, dll.) belum termasuk dalam scope penelitian ini
2. Scope Pengguna
   Target pengguna adalah developer muda di Indonesia dengan fokus pada pengembangan smart monitoring system dan robotika basic to intermediate level
3. Scope Fungsionalitas
   - Penelitian ini fokus pada core functionalities: IDE, panel kontrol, dan tunnelling

- Advanced features seperti collaborative development dan cloud integration akan menjadi future work.

4. Testing Environment
   - Pengujian dilakukan pada perangkat dengan spesifikasi tertentu yang representatif untuk target market
   - Pengujian pada berbagai jenis STB akan dilakukan secara terbatas.

BAB II

TINJAUAN PUSTAKA

2.1 Integrated Development Environment (IDE) Berbasis Web

Integrated Development Environment (IDE) berbasis web telah mengalami transformasi signifikan dalam beberapa tahun terakhir, memberikan dampak besar pada pendidikan pemrograman, pengembangan perangkat lunak, dan penelitian ilmu komputer. Menurut penelitian yang dipublikasikan oleh IEEE, konsep "online compiler as a cloud service" menawarkan berbagai keunggulan seperti portabilitas, pengurangan kebutuhan penyimpanan, dan independensi platform bagi pengguna (IEEE, 2020). Sistem ini memungkinkan pengguna untuk mengirimkan program tanpa perlu menginstal compiler secara lokal, dengan proses kompilasi dan eksekusi ditangani oleh server backend, kemudian hasilnya dikembalikan ke antarmuka pengguna.

Penelitian oleh Goldman et al. (2011) tentang Collabode, sebuah IDE Java berbasis web, menunjukkan bahwa IDE berbasis web dapat mendukung kolaborasi sinkron antar programmer, mengatasi kompleksitas seperti kesalahan kompilasi yang diperkenalkan oleh beberapa pengguna. Hal ini menyoroti tren yang lebih luas menuju pembuatan lingkungan pengembangan yang lebih interaktif dan kooperatif dalam format berbasis web.

2.1.1 Arduino Web Editor dan Platform Cloud

Arduino telah mengembangkan Arduino Cloud Editor sebagai IDE online resmi yang memungkinkan pengguna untuk menulis kode, mengunggah sketsa langsung ke board Arduino yang kompatibel, dan menyimpan pekerjaan mereka ke cloud, memastikan aksesibilitas dari berbagai perangkat (Arduino, 2024). Editor ini secara konsisten diperbarui dengan fitur-fitur terbaru dan dukungan board. Arduino Cloud berfungsi sebagai "online Sketchbook," memfasilitasi alur kerja berkelanjutan dari konsep awal hingga implementasi proyek dan memungkinkan manajemen proyek terpusat.

Menurut dokumentasi resmi Arduino (2024), Arduino Cloud merupakan platform komprehensif untuk membuat, menyebarkan, dan memantau proyek Internet of Things (IoT). Platform ini menyediakan penyimpanan cloud untuk sketsa dan memfasilitasi organisasi dan berbagi proyek. Platform ini mendukung berbagai opsi konektivitas, termasuk Wi-Fi, jaringan seluler, atau Ethernet, untuk membangun koneksi perangkat. Platform ini memprioritaskan kemudahan penggunaan dan antarmuka intuitif, memungkinkan individu, sekolah, dan bisnis untuk membangun, mengontrol, dan memonitor proyek IoT dari jarak jauh.

Namun demikian, penelitian menunjukkan bahwa Arduino Web Editor memerlukan instalasi agent lokal (Arduino Create Agent) untuk menjembatani komunikasi antara browser dengan board Arduino yang terhubung via USB (Maker.pro, 2023). Ketergantungan pada local agent ini menjadi salah satu tantangan dalam implementasi IDE berbasis web yang sepenuhnya cloud-based, karena menciptakan dependency yang dapat menjadi single point of failure dan memerlukan setup tambahan dari pengguna.

2.1.2 Keterbatasan dan Tantangan IDE Berbasis Web

Studi yang dilakukan oleh berbagai peneliti telah mengidentifikasi keterbatasan dan tantangan IDE online. Penelitian mengkaji berbagai fitur seperti dukungan bahasa, fungsionalitas IDE, aksesibilitas, dan alat kolaborasi, sambil juga mengatasi kendala inheren seperti masalah keamanan terkait eksekusi kode jarak jauh, keterbatasan alokasi sumber daya, dan tantangan dalam manajemen dependensi (IEEE, 2019).

Diskusi dalam komunitas pengembangan Arduino telah mengeksplorasi pembuatan IDE berbasis web, berfokus pada aspek-aspek seperti editor kode HTML5, kompilasi sisi server, manajemen sketsa online, dan penanganan pustaka terpusat (Arduino Forum, 2015). Tantangan signifikan yang diidentifikasi adalah proses mengunggah sketsa yang telah dikompilasi (file HEX) ke board Arduino secara langsung dari antarmuka web. Meskipun solusi yang melibatkan daemon lokal atau antarmuka baris perintah spesifik seperti Arduino CLI (yang menawarkan antarmuka gRPC dan pustaka Go) telah diusulkan, implementasi praktis masih memerlukan kompleksitas teknis yang tinggi.

Penelitian lebih lanjut mengidentifikasi bahwa web IDE untuk Arduino sering mengalami keterbatasan dalam hal penyimpanan (misalnya 100MB atau 100 sketches untuk Arduino Web Editor), opsi board manager yang terbatas, dan terkadang mengalami masalah dengan server overload yang mempengaruhi reliabilitas (Pitsco, 2023). Custom library atau library versi lama juga sering memerlukan import manual atau konversi agar kompatibel dengan web editor, menambah friction dalam development workflow.

2.1.3 Visual Programming dan No-Code Dashboard Development untuk IoT

Perkembangan terbaru dalam IoT development tools menunjukkan trend signifikan menuju visual programming dan no-code/low-code platforms yang memungkinkan non-programmers untuk create functional applications tanpa extensive coding knowledge. Penelitian menunjukkan bahwa visual programming environments dapat reduce development time by 60-80% dibandingkan traditional text-based coding, particularly untuk IoT dashboard creation (IEEE, 2023).

Block-Based dan Flow-Based Visual Programming

Node-RED sebagai Leading Platform:

Node-RED, dikembangkan oleh IBM, represents state-of-the-art dalam flow-based visual programming untuk IoT applications (Node-RED Foundation, 2024). Platform ini mengimplementasikan visual paradigm dimana users connect pre-built "nodes" representing different functions (sensors, actuators, logic, APIs) melalui wiring diagram approach. Research demonstrates bahwa Node-RED significantly lowers barrier to entry untuk IoT development, dengan user studies showing 73% faster prototyping speed compared to conventional programming methods (ResearchGate, 2023).

Key characteristics dari flow-based programming yang relevant untuk Arduino development include:

- Visual Data Flow: Explicit representation dari data movement between components
- Modular Architecture: Reusable nodes untuk common IoT tasks
- Real-Time Testing: Live deployment dan debugging capabilities

- Extensive Library: 4000+ community-contributed nodes untuk various protocols dan services

Namun, research notes bahwa Node-RED primarily designed untuk server-side flows dan requires separate runtime environment, creating potential complexity untuk embedded deployment scenarios (Medium, 2023).

Blockly dan Scratch-Like Environments:

Google's Blockly framework provides foundation untuk numerous block-based visual programming tools specifically designed untuk microcontroller education (Google Blockly, 2024). Extensions seperti Ardublock dan mBlock demonstrate successful integration dengan Arduino platform, enabling students to create programs by dragging dan connecting visual blocks yang represent code statements.

Educational research shows significant benefits:

- Reduced Cognitive Load: Visual syntax eliminates memorization of language-specific keywords (MDPI, 2022)
- Lower Error Rate: Block shapes provide constraints yang prevent syntax errors
- Faster Learning: Students achieve basic competency 3x faster compared to text-based programming (ResearchGate, 2022)
- Engagement: Higher motivation scores (average 4.2/5.0) dalam user questionnaires (IEEE Education, 2023)

However, limitations noted dalam research include: reduced flexibility untuk advanced programming, performance overhead dari code generation, dan difficulty expressing complex algorithms (ACM Digital Library, 2023).

No-Code Dashboard Builders untuk IoT

Commercial Platforms Analysis:

Research extensively documents commercial no-code dashboard platforms yang specifically target IoT applications:

Blynk Platform:

Blynk provides comprehensive mobile-first dashboard builder dengan 40+ pre-built widgets (buttons, sliders, gauges, charts) yang can be drag-dropped untuk interface creation (Blynk, 2024). Technical analysis shows:

- Widget Library: Extensive collection covering 90% of common IoT use cases
- Data Binding: Simple pin mapping mechanism untuk Arduino integration
- Real-Time Sync: WebSocket-based communication dengan < 100ms latency
- Multi-Platform: iOS, Android, dan web dashboard support

User satisfaction surveys indicate 4.6/5.0 average rating dari 2M+ users (App Store Analytics, 2024). However, research identifies critical limitations:

- Vendor Lock-In: Proprietary platform dengan limited export capabilities
- Subscription Model: Free tier limited to 2000 energy points; paid plans $5-99/month
- Cloud Dependency: Requires constant internet connectivity untuk operation
- Privacy Concerns: All data routed through Blynk servers (Blynk Documentation, 2024)

ThingsBoard Platform:

ThingsBoard offers rule-engine-integrated dashboard capabilities dengan enterprise features (ThingsBoard, 2024). Technical characteristics:

- Customization: Drag-drop widgets dengan extensive styling options
- Data Processing: Built-in rule chains untuk complex logic
- Scalability: Supports thousands of concurrent devices
- Open Core: Community edition available, enterprise features paid

Deployment studies show ThingsBoard effective untuk industrial IoT scenarios but presents steep learning curve dengan 15-20 hour average time to competency untuk non-technical users (IoT Analytics, 2023).

Separation of Development dan End-User Interfaces

Contemporary research emphasizes importance of clear separation between developer tools dan end-user interfaces dalam IoT applications (CHI Conference, 2023). Studies demonstrate bahwa conflating development environment dengan end-user controls creates significant usability issues:

Developer Interface Requirements:

- Code editing capabilities
- Debugging tools
- System configuration access
- Terminal dan logging interfaces

End-User Interface Requirements:

- Task-specific controls (only relevant functions exposed)
- Simplified visual feedback
- Zero technical terminology

- Mobile-optimized layouts

Research dalam human-computer interaction shows bahwa context-appropriate interfaces improve task completion rates by 47% dan reduce errors by 62% compared to generic multi-purpose interfaces (ACM SIGCHI, 2023).

Dashboard Publishing Paradigm:

Modern IoT platforms implement "publish" workflows where developers create dashboards yang then deployed as separate, access-controlled interfaces untuk end-users (Node-RED Dashboard, 2024; Grafana, 2024). Key features identified dalam research:

- URL-Based Distribution: Unique URLs untuk each published dashboard
- Access Control: Password protection atau user whitelisting
- Permission Levels: Read-only versus control permissions
- Version Management: Ability to update dashboards without redistributing access

Studies show this paradigm reduces deployment friction by 85% compared to traditional application distribution methods (IEEE Software, 2023).

Widget-Based Dashboard Construction

Component Library Patterns:

Research identifies optimal widget categorization untuk IoT dashboards based on user task analysis (UX Research, 2023):

Input Widgets (for control):

- Buttons (momentary, toggle, pulse)
- Sliders (continuous value adjustment)
- Number inputs (precise value entry)
- Color pickers (RGB/HSV selection)
- Dropdowns (discrete option selection)

Output Widgets (for monitoring):

- Gauges (circular, linear, tank-style)
- Charts (line, bar, scatter, heatmap)
- Text labels (numeric, alphanumeric)
- LED indicators (status visualization)
- Progress bars (percentage completion)

User studies demonstrate that widget-based assembly reduces dashboard creation time from 2-4 hours (custom coding) to 15-30 minutes (drag-drop construction) for typical monitoring scenarios (UX Collective, 2024).

Data Binding Mechanisms

Automatic Pin Detection:

Advanced dashboard builders implement automatic Arduino pin detection dan suggestion mechanisms (Arduino Forum, 2024). Technical approaches include:

- Parsing Arduino sketch untuk pinMode declarations
- Serial protocol handshake revealing available pins
- Runtime introspection via custom firmware commands
- Manual configuration dengan visual pin selector

Research shows automatic binding reduces configuration errors by 73% compared to manual text-based configuration files (IoT Design Patterns, 2023).

Real-Time Data Streaming:

WebSocket protocol dominates as optimal choice untuk dashboard-to-device communication (W3C WebSocket Spec, 2023). Performance studies demonstrate:

- Latency: 20-100ms typical (local network)
- Throughput: 1000+ messages/second sustainable
- Overhead: 2-byte frame header versus 200+ bytes for HTTP
- Scalability: 10,000+ concurrent connections per server achievable

Alternative protocols documented dalam research:

- MQTT: Pub-sub pattern ideal untuk multi-client scenarios
- Server-Sent Events: Unidirectional server push, simpler than WebSocket
- HTTP Long Polling: Fallback untuk restrictive firewalls

Educational Implications

Research dalam education technology demonstrates significant pedagogical benefits dari visual programming dan no-code dashboard tools (MDPI Education, 2024):

Learning Outcomes:

- Comprehension: 35% higher scores on embedded systems concept tests
- Engagement: 89% of students prefer visual tools versus text IDE

- Project Completion: 92% completion rate versus 67% with traditional tools
- Confidence: Self-efficacy scores improve 28% post-instruction

Accessibility:

- Enables participation dari students with dyslexia atau visual processing difficulties
- Reduces language barriers untuk non-native English speakers
- Accommodates different learning styles (visual, kinesthetic)

Teacher Feedback:

Instructor surveys show 82% agreement that visual tools enable earlier introduction to complex concepts, dengan students able to create functional IoT projects 6-8 weeks earlier dalam curriculum compared to text-based approaches (ISTE Research, 2023).

Research Gap: Integrated Visual IDE dengan Dashboard Publishing

Despite extensive research pada standalone visual programming tools dan dashboard platforms, limited work addresses integration dari:

- Unified Environment: Single platform combining code editor, visual programming, dan dashboard builder
- Seamless Publishing: One-click deployment dari dashboard to shareable URL
- Zero-Cost Self-Hosting: Elimination dependency pada paid cloud services
- Offline-First Design: Full functionality without internet connectivity
- Embedded Dashboard Editor: WYSIWYG editor embedded dalam development environment

This integration gap presents research opportunity untuk comprehensive Arduino development platform yang unifies programming, dashboard creation, dan end-user deployment dalam cohesive ecosystem, addressing fragmentation yang currently exists dalam IoT development workflow.

2.2 Sistem Operasi untuk Embedded Systems dan IoT

2.2.1 Linux sebagai OS untuk Embedded Systems

Linux telah menjadi sistem operasi yang sangat disukai untuk embedded systems dan perangkat IoT karena versatilitas, sifat open-source, kemampuan kustomisasi, dan fitur keamanan yang robust (TuxCare, 2024). Penelitian menyoroti adaptabilitasnya untuk lingkungan dengan sumber daya terbatas dan kemampuannya untuk berjalan pada berbagai arsitektur CPU, termasuk ARM yang dominan dalam embedded systems (IEEE, 2021).

Distribusi Linux spesifik yang dioptimalkan untuk IoT termasuk Raspberry Pi OS (sebelumnya Raspbian), Ubuntu Core, Yocto Project, OpenWrt, dan BalenaOS (TuxCare, 2024). Sistem-

sistem ini menawarkan keuntungan seperti manajemen sumber daya yang efisien, keamanan melalui transactional updates, dan dukungan untuk containerized applications. Ubuntu Core, khususnya, dikenal dengan security-focused approach dan container-based architecture yang ideal untuk IoT deployments. Yocto Project memungkinkan developers membuat custom Linux distributions yang highly optimized untuk specific hardware requirements.

Penggunaan embedded Linux untuk sistem mechatronik dan dalam mengoptimalkan perangkat dengan sumber daya terbatas untuk IoT juga merupakan area studi yang signifikan (ResearchGate, 2022). Penelitian menunjukkan bahwa dengan proper optimization, embedded Linux dapat berjalan efisien bahkan pada devices dengan RAM terbatas (minimal 32MB) dan storage minimal (8MB flash).

2.2.2 Operating System untuk Perangkat IoT Resource-Constrained

Selain distribusi Linux lengkap, berbagai sistem operasi dirancang khusus untuk perangkat IoT, melayani kemampuan hardware yang berbeda dan persyaratan aplikasi. Ini dapat dikategorikan menjadi yang berbasis Linux kernel dan lightweight real-time operating systems (RTOS) untuk perangkat yang lebih terbatas.

Untuk perangkat dengan memori dan daya proses terbatas, sistem operasi seperti Contiki, TinyOS, LiteOS, FreeRTOS, Zephyr, RIOT, dan Apache Mynewt sering dipelajari dan dibandingkan (IJMSE, 2023; Linux.com, 2022). Sistem-sistem ini dioptimalkan untuk:

- Konsumsi daya rendah: Essential untuk battery-powered IoT devices
- Footprint kecil: Bekerja dengan RAM sekecil 2KB dan ROM 16KB
- Real-time capabilities: Deterministic behavior untuk time-critical applications
- Networking stack: Built-in support untuk protokol IoT seperti MQTT, CoAP, 6LoWPAN

Penelitian comparative analysis menunjukkan bahwa pemilihan OS bergantung pada trade-off antara features, resource consumption, dan development complexity (GeeksforGeeks, 2024). FreeRTOS, misalnya, widely adopted karena simplicity dan extensive community support, sementara Zephyr menawarkan more comprehensive feature set dengan modular architecture.

2.2.3 Arduino dan Arsitektur Bootloader

Arduino adalah platform elektronik open-source yang banyak digunakan untuk prototyping cepat dan mengembangkan proyek IoT interaktif (Halvorsen, 2024; Robocraze, 2023). Meskipun board Arduino adalah sentral untuk banyak proyek IoT, mereka biasanya menjalankan bootloader daripada sistem operasi penuh, memiliki daya proses dan memori yang terbatas (ResearchGate, 2020).

Bootloader Arduino (seperti Optiboot untuk Arduino Uno) occupies minimal space (512 bytes) dan provides basic functionality untuk programming via serial interface. Ini membuat mereka cocok untuk tugas-tugas yang lebih sederhana dan aplikasi kontrol embedded, dengan fungsi IoT yang lebih kompleks sering ditangani dengan integrasi dengan platform cloud atau prosesor yang lebih powerful seperti Raspberry Pi (IJRPR, 2022).

Namun, keterbatasan ini juga presents opportunity untuk innovation. Running full operating system pada companion device (seperti Raspberry Pi atau dedikated embedded Linux board) yang manages Arduino programming dan remote access dapat provide best of both worlds: simplicity dari Arduino programming model combined dengan power dan flexibility dari full OS.

2.3 Sistem Kontrol dan Pemrograman Arduino Jarak Jauh

2.3.1 Sistem Kontrol Supervisory Berbasis Web

Banyak artikel ilmiah menjelaskan desain dan implementasi sistem kontrol supervisory di mana Arduino bertindak sebagai pengontrol lokal dalam kerangka kerja yang lebih luas. Sistem ini biasanya melibatkan server pusat yang berkomunikasi dengan Arduino, memungkinkan pemantauan jarak jauh dan eksekusi perintah melalui browser web (ResearchGate, 2018).

Penelitian menguraikan metode untuk membangun antarmuka web, termasuk pemilihan pola arsitektur optimal dan protokol komunikasi yang efisien. Technology stack umum melibatkan:

- Frontend: React, Vue, atau Angular untuk responsive web interfaces
- Backend: Node.js, Python (Flask/Django), atau PHP untuk server logic
- Communication Protocols:
- HTTP/HTTPS untuk request-response patterns
- MQTT untuk lightweight pub-sub messaging
- WebSockets untuk real-time bidirectional communication
- CoAP untuk resource-constrained devices

Arsitektur typical mencakup REST API untuk control commands, MQTT broker untuk telemetry data, dan WebSocket connections untuk real-time updates. This multi-protocol approach balances reliability, efficiency, dan real-time requirements dari IoT applications.

2.3.2 Laboratorium Jarak Jauh untuk Pendidikan

Area aplikasi yang signifikan adalah pengembangan laboratorium jarak jauh berbasis web yang memungkinkan siswa untuk memprogram dan bereksperimen dengan board Arduino fisik dari jarak jauh. Platform ini bertujuan untuk memberikan pengalaman langsung dalam pemrograman mikrokontroler dan IoT, mendukung berbagai eksperimen, termasuk sensor dan robotika (ResearchGate, 2019; Semantic Scholar, 2020).

Menurut penelitian di bidang pendidikan embedded systems, Cloud IDE mengubah pendidikan pemrograman embedded systems dengan menyediakan lingkungan yang dapat diakses, kolaboratif, dan hardware-abstracted untuk belajar tentang mikrokontroler dan sistem embedded (Code-B.dev, 2024). Platform ini menghilangkan kebutuhan untuk setup lokal yang ekstensif, memungkinkan pelajar untuk menulis, mengompilasi, debug, dan bahkan mensimulasikan kode embedded langsung di browser web.

Studi lebih lanjut menunjukkan bahwa remote laboratory untuk mikrokontroler seperti STM32 memungkinkan guru untuk memonitor dan mengontrol mikrokontroler siswa dari jarak jauh dan memverifikasi fungsi program (MDPI, 2023). Platform ini typically include:

- Code Editor: Web-based IDE dengan syntax highlighting dan autocomplete
- Compilation Service: Cloud-based atau edge-based compilation
- Upload Mechanism: Remote flashing capabilities via USB-over-IP atau dedicated hardware
- Monitoring Tools: Serial monitor, oscilloscope integration, sensor visualization
- Assessment Features: Automated testing dan grading capabilities

Evaluasi kualitatif dari remote laboratories menyoroti kebutuhan untuk interaksi pengguna yang lebih inherent untuk meningkatkan pengalaman belajar. Improvements dalam implementasi remote lab menunjukkan peningkatan signifikan (up to 40%) dalam kemampuan siswa untuk menyelesaikan tugas, mengindikasikan usability dan efektivitas yang lebih baik (MDPI, 2023).

2.3.3 Strategi Komunikasi dan Integrasi Platform IoT

Metode yang berbeda digunakan untuk membangun komunikasi antara Arduino dan antarmuka web. Ini termasuk:

- Ethernet Shield: Direct network connectivity, suitable untuk stationary installations
- Wi-Fi Modules: ESP8266 atau ESP32, providing wireless connectivity
- Cellular Modules: GSM/LTE untuk wide-area coverage
- Serial-to-Network Bridges: Menggunakan Raspberry Pi atau similar device sebagai gateway

Beberapa studi mengeksplorasi pengintegrasian proyek Arduino dengan platform cloud IoT yang sudah mapan seperti Arduino IoT Cloud, Blynk, ThingSpeak, dan Thingworx (LNTU, 2023; IJRPR, 2022). Platform ini menyediakan alat untuk:

- Data Collection: High-frequency sensor data aggregation
- Data Analysis: Built-in analytics dan anomaly detection
- Visualization: Customizable dashboards dan real-time charts
- Remote Management: OTA updates dan remote configuration
- Scalability: Support untuk thousands dari connected devices

Penelitian menunjukkan bahwa platform ini significantly simplify development dari scalable IoT applications, tetapi introduce dependency pada third-party services dan potential subscription costs (UPS Ecuador, 2022).

2.3.4 Customizable Dashboard dan Panel Builder untuk Remote Monitoring

Perkembangan terbaru dalam IoT visualization menekankan pentingnya customizable dashboards yang dapat disesuaikan dengan kebutuhan spesifik aplikasi tanpa memerlukan coding ekstensif. Penelitian menunjukkan bahwa user-defined interfaces significantly improve user engagement dan effectiveness dalam monitoring sistem embedded (MDPI, 2021).

Dashboard Customization Frameworks

Low-Code/No-Code Dashboard Builders:

Platform modern seperti Node-RED, Grafana, dan custom dashboard builders memungkinkan users untuk create sophisticated monitoring interfaces melalui drag-and-drop mechanisms (Node-RED Foundation, 2024). Key features include:

- Widget Libraries: Pre-built components untuk common use cases (gauges, charts, controls)
- Data Binding: Automatic connection antara hardware data streams dan visual elements
- Responsive Layouts: Grid-based systems yang adapt ke berbagai screen sizes
- Real-time Updates: WebSocket-based live data streaming tanpa page refresh
- Custom Styling: Theming systems untuk branding dan personalization

Research demonstrates bahwa drag-and-drop interfaces reduce development time by 60-80% compared to traditional coding approaches, while maintaining flexibility untuk advanced customization (IEEE, 2023).

Component-Based Architecture

Reusable Widget Components:

Modern dashboard systems implement component-based architectures yang facilitate reusability dan maintainability. Common widget types include:

- Input Controls:
- Buttons untuk triggering actions (on/off, reset)
- Sliders untuk continuous value adjustment (motor speed, LED brightness)
- Toggle switches untuk binary states
- Number inputs untuk precise value entry
- Color pickers untuk RGB LED control
- Display Widgets:
- Real-time line charts untuk sensor data trending

- Gauge displays untuk status indicators (temperature, humidity)
- Text labels untuk current values dan status
- LED indicators untuk binary states
- Video streams untuk camera integration
- Data Visualization:
- Time-series charts dengan pan/zoom capabilities
- Bar charts untuk comparative data
- Pie charts untuk distribution analysis
- Heatmaps untuk spatial data
- Tables untuk detailed data inspection

Real-Time Communication Protocols

WebSocket Implementation:

Research highlights WebSocket as optimal protocol untuk real-time bidirectional communication antara browser-based dashboards dan Arduino devices (W3C, 2023). Advantages include:

- Low Latency: Typical latency < 100ms untuk local networks
- Persistent Connection: Eliminates overhead dari HTTP request polling
- Bidirectional: Simultaneous data push dari server dan commands dari client
- Efficient: Minimal protocol overhead dibanding HTTP long-polling

Studies show bahwa WebSocket-based dashboards achieve 10x better performance dalam terms of latency dan bandwidth usage compared to traditional AJAX polling methods (ResearchGate, 2023).

Message Queue Telemetry Transport (MQTT):

Untuk scenarios dengan multiple devices atau distributed systems, MQTT provides publish-subscribe pattern yang efficient (MQTT.org, 2024). Benefits documented in research:

- Quality of Service (QoS) Levels: Guaranteed message delivery untuk critical commands
- Topic-Based Routing: Flexible message organization dan filtering
- Lightweight Protocol: Minimal 2-byte overhead, suitable untuk constrained networks

- Broker Architecture: Centralized message distribution untuk scalability

Dashboard Persistence dan State Management

Configuration Storage:

Research emphasizes importance of saving user-created dashboard configurations untuk reproducibility dan sharing (Semantic Scholar, 2022). Implementation approaches:

- JSON Schema: Dashboard layout dan widget configurations stored sebagai JSON
- Database Storage: User preferences, historical data, dan custom layouts
- Export/Import: Sharing dashboard templates across users atau projects
- Version Control: Tracking dashboard changes dan rollback capabilities

Integration dengan Arduino Communication

Serial-to-WebSocket Bridge:

Technical implementations commonly utilize gateway services yang translate antara Arduino serial communication dan WebSocket protocol (Arduino Forum, 2023):

This architecture provides:

- Protocol Translation: Converting binary serial data ke JSON untuk web
- Buffer Management: Handling high-frequency data streams
- Connection Pooling: Supporting multiple concurrent dashboard users
- Data Aggregation: Downsampling high-rate sensor data untuk visualization

Customization Challenges dan Solutions

Performance Optimization:

Research identifies key challenges dalam custom dashboard development (IEEE, 2022):

- High-Frequency Data Handling:
- Problem: Browser rendering bottleneck dengan > 50 updates/second
- Solution: Client-side data decimation, canvas-based rendering, WebGL acceleration
- Multiple Widget Synchronization:
- Problem: Consistent state across multiple connected clients
- Solution: Server-side state management, websocket broadcast, optimistic UI updates

- Mobile Responsiveness:
- Problem: Complex layouts tidak scale ke small screens
- Solution: CSS Grid/Flexbox, responsive breakpoints, mobile-first design

Educational Implications

Studies demonstrate bahwa customizable dashboards significantly enhance learning outcomes dalam embedded systems education (MDPI, 2023):

- Visual Feedback: Students better understand sensor behavior melalui real-time visualization
- Engagement: Interactive controls increase hands-on experimentation
- Debugging: Visual data inspection facilitates troubleshooting
- Project Showcase: Custom dashboards enable impressive project demonstrations

User studies show 35% improvement dalam student comprehension ketika using visual dashboards versus text-based serial monitors, particularly untuk time-series data analysis (ResearchGate, 2024).

Industry Applications

Commercial IoT platforms demonstrate value of customizable dashboards:

- Blynk: Mobile-first widget builder dengan 40+ pre-built widgets
- ThingsBoard: Rule engine integration dengan dashboard customization
- Grafana: Advanced query capabilities dengan plugin ecosystem
- Node-RED Dashboard: Flow-based visual programming dengan instant dashboard generation

However, research notes bahwa commercial platforms introduce vendor lock-in dan subscription costs ($5-50/month typical), creating barrier untuk educational dan hobbyist use cases (UPS Ecuador, 2022).

Research Gap: Self-Hosted Custom Panel Builders

Despite extensive research pada dashboard frameworks, limited work addresses:

- Offline-First Dashboards: Operating without constant internet connectivity
- Zero-Configuration Setup: Automatic discovery dan binding dari Arduino pins
- Template Libraries: Pre-built dashboard templates untuk common Arduino projects
- Embedded Dashboard Editor: In-browser WYSIWYG editor tanpa external tools

- Cost-Free Solutions: Self-hosted alternatives tanpa subscription requirements

This gap presents opportunity untuk integrated dashboard builder dalam custom Arduino development environments, eliminating dependency pada third-party services while maintaining professional-quality visualization capabilities.

2.4 Layanan Tunneling untuk Akses Jarak Jauh

2.4.1 Ngrok sebagai Solusi Tunneling

Ngrok adalah layanan tunneling yang memungkinkan untuk mengekspos server lokal, termasuk yang berjalan pada perangkat IoT berbasis Arduino atau ESP32, ke internet publik. Ini memungkinkan akses jarak jauh dan kontrol proyek IoT tanpa konfigurasi router yang kompleks seperti port forwarding (Kunzleigh, 2023; Ngrok, 2024).

Ngrok membuat tunnel aman dari jaringan lokal ke cloud ngrok, membuat server web Arduino dapat diakses melalui URL publik (Arduino Forum, 2021). Mechanism bekerja sebagai berikut:

- Local Client: Ngrok client runs pada local network, establishes outbound connection ke ngrok servers
- Tunnel Establishment: Secure tunnel created using TLS encryption
- Public URL: Ngrok assigns unique public URL (e.g., https://abc123.ngrok.io)
- Request Routing: Incoming requests ke public URL routed through tunnel ke local server
- Response Return: Responses sent back through same tunnel

Ini sangat berguna untuk perangkat IoT yang perlu dikontrol atau dipantau dari mana saja di dunia tanpa complex network configuration (Ngrok, 2024).

2.4.2 Keunggulan dan Keterbatasan Tunneling Services

Keunggulan menggunakan ngrok untuk IoT Arduino meliputi:

- Melewati NAT dan Firewall: Ngrok mengatasi network address translation (NAT) dan pembatasan firewall dengan menggunakan outbound connections, yang generally allowed oleh firewalls (Ngrok, 2024)
- Tidak Memerlukan Port Forwarding: Eliminates need untuk router configuration, simplifying setup especially untuk users behind carrier-grade NAT
- URL Persisten (Paket Berbayar): Paid plans offer custom, persistent URLs (e.g., myarduino.ngrok.io) beneficial untuk production deployments
- Fitur Keamanan: HTTP Basic authentication, OAuth, IP restrictions untuk secure access control
- Logging dan Inspection: Built-in request inspection untuk debugging

Namun, untuk implementasi tunneling yang lebih canggih, penelitian menunjukkan bahwa solusi seperti "ESP Tunnel" untuk board ESP32 memungkinkan pembuatan secure WebSocket tunnels untuk akses HTTPS remote, melewati NAT dan firewall (PlatformIO, 2023). Solusi ini provides:

- End-to-end Encryption: TLS termination pada ESP32 device
- Lower Latency: Direct connection without intermediate servers
- No Third-party Dependency: Self-hosted solution
- Cost Effective: No subscription fees

Namun, these advanced solutions memerlukan:

- Specific hardware (ESP32 dengan sufficient memory)
- Complex configuration (certificate management, DNS setup)
- Technical expertise untuk troubleshooting
- Maintenance overhead

Pertimbangan important meliputi kebutuhan perangkat perantara (seperti Raspberry Pi atau PC) yang menjalankan klien ngrok hampir selalu diperlukan untuk menjembatani Arduino ke layanan ngrok (Kunzleigh, 2023; Reddit, 2023). Ketergantungan pada local agent dan/atau external services ini menjadi salah satu argumen kuat untuk mengembangkan solusi terintegrasi yang mengeliminasi kebutuhan setup kompleks dan external dependencies.

2.4.3 Cloudflare Tunnel sebagai Alternatif Enterprise-Grade Gratis

Cloudflare Tunnel (cloudflared) menawarkan solusi tunneling enterprise-grade yang sepenuhnya gratis sebagai alternatif superior terhadap layanan berbayar seperti ngrok untuk implementasi remote access pada sistem embedded dan IoT (Cloudflare, 2024). Berbeda dengan pendekatan tradisional yang memerlukan port forwarding atau VPN, Cloudflare Tunnel establishes encrypted, outbound-only connection dari local device ke Cloudflare's global edge network, secara drastis reducing attack surface (2Cloud.io, 2024; DevOpsTales, 2024).

Arsitektur dan Keunggulan Teknis

1. Model Koneksi Outbound-Only:

Cloudflare Tunnel menggunakan arsitektur yang fundamental berbeda dari traditional tunneling. Daemon cloudflared yang berjalan pada local network initiates outbound connections ke Cloudflare edge servers, eliminating kebutuhan untuk membuka inbound ports pada firewall (Cloudflare, 2024). Model ini provides several security advantages:

- Zero Network Exposure: Tidak ada inbound ports yang dibuka, eliminating common attack vectors seperti port scanning dan external exploitation attempts (ImplRust, 2024)

- Bypass NAT/CG-NAT: Automatically bypasses Carrier-Grade NAT issues yang prevalent dalam mobile networks dan shared hosting environments (PieterDev, 2024)
- Firewall Compatibility: Works seamlessly dengan restrictive corporate atau educational firewalls yang typically block inbound connections (It's FOSS, 2024)

2. Enterprise-Grade Security Features:

Research menunjukkan bahwa Cloudflare Tunnel provides application-level security controls melalui Zero Trust integration (Medium, 2024):

- Granular Access Control: Fine-grained policies based pada email domains, IP addresses, atau OAuth providers
- Built-in DDoS Protection: Automatic mitigation dari distributed denial-of-service attacks melalui Cloudflare's global network dengan capacity 100+ Tbps
- Web Application Firewall (WAF): Protection terhadap common web exploits (SQL injection, XSS, CSRF)
- Automatic SSL/TLS: Certificate provisioning dan renewal tanpa manual configuration
- End-to-End Encryption: TLS 1.3 encryption dari user browser hingga origin server

3. Performance dan Global Distribution:

Cloudflare's global network spans 300+ cities dalam 100+ countries, providing significant performance advantages (Cloudflare, 2024):

- Low Latency Routing: Traffic automatically routed melalui nearest edge location
- Smart Load Balancing: Distributes requests across multiple tunnel instances untuk high availability
- Connection Pooling: Maintains persistent connections untuk reduced overhead
- Compression Support: Automatic Brotli/Gzip compression untuk bandwidth optimization

Implementasi untuk IoT dan Embedded Systems

Deployment Architecture:

Untuk IoT applications, typical deployment menggunakan intermediate device (Raspberry Pi, dedicated embedded Linux board, atau lightweight server) yang runs cloudflared daemon dan manages tunnel connection (ImplRust, 2024). Architecture ini provides best of both worlds:

- Simple Arduino Programming Model: Arduino remains focused pada its core function (sensors, actuators, control logic)

- Powerful OS Capabilities: Linux-based gateway handles complex networking, security, dan remote access
- Local USB Communication: Direct USB connection antara gateway dan Arduino boards
- Web IDE Hosting: Gateway runs web-based IDE accessible globally via tunnel

Technical Implementation:

Cloudflared daemon requires minimal resources:

- Memory Footprint: < 10MB RAM during operation
- CPU Usage: < 1% pada idle, < 5% under load
- Storage: < 50MB binary size
- Platform Support: x8664, ARM64, ARMv7 architectures

Research demonstrates feasibility untuk running cloudflared bahkan pada resource-constrained devices seperti Raspberry Pi Zero (DevOpsTales, 2024).

Keunggulan Cost-Benefit Analysis

Free Tier Comparison:

Comparative analysis antara Cloudflare Tunnel dan ngrok menunjukkan significant cost advantages (Reddit r/selfhosted, 2024):

Feature	**Cloudflare Tunnel (Free)**	**Ngrok (Free)**	**Ngrok (Paid)**
Bandwidth	Unlimited	1GB/month	Unlimited
Custom Domain	Yes	No	Yes ($8-20/mo)
Persistent URL	Yes	Random	Yes
SSL/TLS	Auto	Auto	Auto
DDoS Protection	Enterprise	None	None
Access Control	Zero Trust	None	Basic Auth
Multiple Services	Unlimited	1 Endpoint	Limited
Annual Cost	$0	N/A	$96-240

Untuk educational institutions atau research projects dengan budget constraints, cost savings dapat be substantial. Multi-year deployment across multiple devices atau users scales linearly dengan ngrok but remains zero-cost dengan Cloudflare Tunnel.

Integration Considerations untuk Research

Automated Configuration:

Dalam konteks proyek ini, custom OS integrates cloudflared sebagai system service yang automatically configured during first boot, eliminating manual setup complexity yang identified sebagai barrier dalam research literature (2Cloud.io, 2024). Automated setup includes:

- Installation: Binary download dan system integration
- Authentication: Browser-based OAuth flow dengan Cloudflare account
- Tunnel Creation: Automatic provisioning dengan unique identifier
- DNS Configuration: Automatic CNAME record creation
- Service Activation: Systemd integration untuk auto-start on boot

Production-Ready Reliability:

Cloudflare's infrastructure provides Service Level Agreement (SLA) dengan 99.99%+ uptime, significantly superior to self-hosted ngrok alternatives (Cloudflare, 2024). Built-in features for reliability include:

- Automatic Reconnection: Handles network interruptions gracefully
- Health Checks: Monitors origin service availability
- Failover Support: Multiple tunnel replicas untuk high availability
- Comprehensive Logging: Detailed access logs untuk debugging dan analytics

Limitations dan Practical Considerations

Despite advantages, Cloudflare Tunnel has limitations yang should be considered:

Setup Complexity:

Initial configuration requires Cloudflare account dan domain (dapat be free subdomain), adding slight complexity compared to ngrok's instant URLs (It's FOSS, 2024). However, this one-time setup cost is offset by long-term benefits.

Resource Requirements:

Cannot run directly pada Arduino boards due to resource constraints. Requires companion device dengan sufficient memory dan processing power untuk run cloudflared daemon (ImplRust, 2024).

Vendor Dependency:

While free tier has no direct costs, creates dependency pada Cloudflare's continued free offering. However, open-source nature dari cloudflared provides migration path jika needed (PieterDev, 2024).

Educational dan Research Implications

Cloudflare Tunnel integration demonstrates feasibility dari production-grade remote access solution tanpa reliance pada paid subscription services, addressing cost dan accessibility concerns prevalent dalam educational settings (It's FOSS, 2024). Key implications include:

Reproducibility:

Free, open-source nature enables other institutions untuk replicate implementation tanpa licensing costs atau subscription management overhead.

Scalability:

Unlimited bandwidth dan connections support large-scale deployments untuk classroom environments atau research lab dengan multiple concurrent users.

Academic Contribution:

Novel integration of Cloudflare Tunnel dengan embedded systems IDE provides reference architecture untuk similar educational tools, contributing to body of knowledge dalam accessible IoT education platforms.

Security Education:

Exposure to enterprise-grade security concepts (Zero Trust, WAF, DDoS protection) provides valuable learning opportunities beyond basic programming skills.

Research literature indicates growing adoption of Cloudflare Tunnel dalam self-hosted dan edge computing scenarios, suggesting trend toward free, infrastructure-grade solutions untuk development environments (Reddit r/selfhosted, 2024; Medium, 2024).

2.5 Automated Deployment dan Installation Systems

2.5.1 Tantangan Deployment pada IoT Devices

Artikel scholarly secara ekstensif mengeksplorasi automated deployment dan installation embedded IoT devices, menyoroti kemajuan dalam provisioning, konfigurasi, dan secure software updates sambil mengatasi berbagai tantangan inheren pada sistem ini.

Banyak studi fokus pada pengembangan metode untuk automated registration, binding, dan konfigurasi IoT devices (ResearchGate, 2023). Proses ini melibatkan:

- Device Discovery: Automatic detection dari new devices pada network
- Association: Binding devices to user accounts atau IoT hubs

- Mutual Authentication: Establishing trust between device dan cloud platform
- Configuration Provisioning: Automatic deployment dari device-specific settings

Penelitian telah mengeksplorasi penggunaan graphs dan ontologies untuk mengotomatisasi proses ini. Manual provisioning can be time-consuming (potentially hours per device), costly (requiring trained personnel), dan impractical untuk large-scale deployments (thousands of devices) (Semantic Scholar, 2022).

Framework berbasis edge telah diusulkan untuk memfasilitasi automated registration dari beragam wireless IoT devices dan deployment aplikasi IoT (ResearchGate, 2023). Framework ini demonstrates:

- Scalability: Tested with 1000+ simultaneous device registrations
- Feasibility: 95%+ success rate dalam automated provisioning
- Flexibility: Support untuk heterogeneous device types dan protocols

Arsitektur untuk automated provisioning typically includes:

- Digital Twin Blueprint Creation: Template definitions untuk device types
- Blueprint Registration: Upload blueprint ke central repository
- Automated Device Adapter Deployment: Runtime generation dari device-specific adapters
- Device Binding: Association antara physical device dan digital twin
- Monitoring Dashboard: Real-time visibility into device status

2.5.2 Over-the-Air (OTA) Updates dan Secure Installation

Over-the-air (OTA) firmware dan software updates kritis untuk longevity dan security dari solusi IoT. Artikel scholarly mendiskusikan secure update architectures, khususnya untuk low-level IoT devices dengan constrained processors (Semantic Scholar, 2021).

Key components dari secure OTA systems include:

- Cryptographic Signing: Firmware signed dengan private key, verified dengan public key on device
- Secure Boot Chain: Ensuring only authenticated firmware can run
- Rollback Protection: Preventing downgrade attacks to vulnerable versions
- Delta Updates: Transmitting only changed bytes untuk minimize bandwidth

Teknik yang digunakan meliputi:

- Elliptic Curve Cryptography (ECC): Smaller key sizes (256-bit) compared to RSA (2048-bit), suitable untuk resource-constrained devices

- Authenticated Key Establishment: Diffie-Hellman untuk secure channel creation
- Hash Chain Verification: Merkle trees untuk incremental verification

Integrity dari firmware updates merupakan tantangan keamanan signifikan. Penelitian recent mengeksplorasi integrasi Generative AI untuk monitoring dan securing updates dengan mendeteksi anomali dan memverifikasi authenticity (ResearchGate, 2024). AI-based approaches dapat:

- Detect unusual update patterns indicating compromise
- Verify firmware behavior matches expected profiles
- Identify zero-day vulnerabilities through anomaly detection

Tantangan dalam area ini termasuk:

- Secure Transmission: Ensuring confidentiality selama transfer over untrusted networks
- Preventing Unauthorized Updates: Authentication mechanisms to prevent malicious firmware injection
- Managing Large-Scale Updates: Coordinating updates across thousands of devices without overwhelming infrastructure
- Resource Constraints: Running crypto operations pada devices dengan limited CPU dan memory

2.5.3 Edge Computing dan Containerization untuk Deployment

Untuk mengatasi keterbatasan resource-constrained IoT devices, framework berbasis edge dan containerized applications sedang dieksplorasi untuk deployment (ResearchGate, 2023). Pendekatan ini aims to:

- Local Intelligence: Processing data at edge reduces latency dan bandwidth requirements
- Reduced Cloud Dependency: Operates independently dari constant cloud connectivity
- Scalability: Distributes computational load across edge nodes
- Privacy: Sensitive data processing remains local

Technologies employed include:

- Docker Containers: Lightweight, portable application packaging
- Kubernetes/K3s: Container orchestration untuk edge deployments

- Edge Runtime Environments: Specialized runtimes optimized untuk edge devices (e.g., Azure IoT Edge, AWS Greengrass)

Research demonstrates bahwa edge-based frameworks enable:

- Automated Registration: Self-registration of heterogeneous wireless IoT devices
- Dynamic Deployment: Runtime deployment of containerized applications
- Resource Optimization: Efficient utilization dari limited edge computing resources

2.6 User Experience dan Usability dalam Pemrograman Remote

2.6.1 Antarmuka yang User-Friendly

Fokus signifikan ditempatkan pada membuat remote learning accessible dan interactive. Pendekatan open-source dan multi-platform (Linux, Windows, macOS) diadopsi untuk meningkatkan user interaction dan accessibility (UPS Ecuador, 2022).

Principles dari user-friendly design dalam remote programming platforms include:

- Intuitive Navigation: Clear menu structures, consistent UI patterns
- Immediate Feedback: Real-time compilation errors, instant code execution results
- Progressive Disclosure: Advanced features hidden until needed, reducing cognitive load
- Contextual Help: Inline documentation, tooltips, interactive tutorials

Evaluasi kualitatif dari remote laboratories menyoroti kebutuhan untuk lebih banyak inherent user interaction untuk meningkatkan learning experience (ResearchGate, 2020). User studies mengidentifikasi pain points seperti:

- Delayed feedback loops (waiting for compilation/upload)
- Difficulty debugging remote hardware
- Lack of tactile interaction dengan physical components
- Connection reliability issues

Ketika mendesain microcontroller-based devices dan their programming interfaces, "usability" dan "elderly-friendly interface" adalah design requirements penting (MDPI, 2021). Ini emphasizes pentingnya mempertimbangkan berbagai level keahlian pengguna:

- Novices: Need guided workflows, templates, extensive documentation
- Intermediates: Require balance between automation dan control
- Experts: Desire full access to advanced features, customization

2.6.2 Simplified Remote Programming Mechanisms

Untuk remote programming, siswa dapat menulis code dan flash ke real microcontrollers langsung dari browser, sering tanpa perlu membeli hardware atau install extensive software (Reddit, 2022). Approaches include:

- Browser-Based Development:
- Web IDE dengan integrated compilation
- Direct USB access via WebUSB API (Chrome/Edge)
- Serial communication via Web Serial API
- CI/CD Integration:
- Git-based workflows
- Automated compilation upon commit
- Remote flashing via CI pipeline
- Automated testing dengan hardware-in-the-loop
- Remote Desktop Approaches:
- TeamViewer atau similar software untuk accessing local setup
- VNC untuk graphical desktop sharing
- SSH tunneling untuk command-line access

Namun, pendekatan ini sering memerlukan setup yang kompleks dan tidak ideal untuk skala besar implementasi educational (MDPI, 2023). Barriers include:

- Technical complexity untuk non-expert users
- Security concerns dengan remote access software
- Scalability limitations (manual setup per student)
- Cost implications dari remote desktop licenses

2.7 Integration Challenges Web IDE, Tunneling, dan Arduino Platform

2.7.1 Ketergantungan pada Local Agent

Mengintegrasikan web IDE, khususnya dengan tunneling, untuk platform Arduino menghadirkan several challenges. Arduino Web Editor memerlukan locally installed "Arduino Create Agent" atau plugin untuk berkomunikasi dengan physical Arduino board yang connected via USB (Maker.pro, 2023).

Agent ini bertindak sebagai bridge:

- USB Communication: Handles serial communication dengan Arduino boards

- Board Detection: Auto-detection dari connected boards
- Firmware Upload: Manages programming protocol (STK500, AVR109, etc.)
- Serial Monitor: Provides terminal interface untuk debugging

Setup ini berarti bahwa meskipun IDE ada di cloud, direct, unfiltered "tunnel" ke remote Arduino tidak typically established tanpa intermediary local machine. Implications include:

- Dependency on Local Software: User must install dan maintain agent
- Single Point of Failure: Agent crashes or update issues block development
- Security Concerns: Agent requires elevated permissions untuk USB access
- Platform Limitations: Agent may not support all operating systems equally

2.7.2 Limited Direct Remote Programming

Untuk truly remote Arduino programming, dimana board tidak physically connected ke computer yang menjalankan web browser, standard Arduino Web Editor menawarkan limited direct support. Sementara beberapa specific boards seperti Arduino Yun historically supported network uploads via SSH/SCP, functionality ini tidak universal across semua Arduino boards (Arduino Forum, 2022).

Users often resort to workarounds:

- Remote Desktop: TeamViewer, VNC, Chrome Remote Desktop
- SSH Tunneling: Port forwarding untuk serial-over-network
- Network-Enabled Boards: ESP32, ESP8266 dengan OTA capabilities
- Intermediate Gateways: Raspberry Pi acting as programming proxy

Ini menjadi significant research gap. Challenges include:

- Lack of standardized remote programming protocols
- Security implications dari exposing programming interfaces
- Latency sensitivity dari upload process
- Reliability concerns over unreliable networks

2.7.3 Network dan Connectivity Issues

Stable internet connection crucial untuk web IDEs. Challenges dapat arise dari:

- DNS Filtering: Corporate/educational networks blocking cloud IDE services
- TLS Handshake Errors: Certificate validation issues, outdated system clocks

- Firewall Blocking: Ports required untuk WebSocket/MQTT/HTTP blocked
- NTP Server Issues: Time synchronization failures preventing secure connections

Users telah melaporkan problems dengan Arduino Cloud Agent failing to connect karena inaccessible ports (Arduino Forum, 2023). Specific issues include:

- Browser compatibility (Opera GX, older IE versions)
- System firewall configurations
- Antivirus interference dengan socket connections
- Proxy server complications dalam corporate environments

Troubleshooting difficulties dalam web IDE environment dapat lebih challenging daripada dengan desktop application. Factors contributing meliputi:

- Limited error visibility (browser console buried)
- Network stack complexity (multiple layers)
- Agent-browser-cloud communication chain
- Asynchronous nature dari web applications

2.8 Artificial Intelligence dan Machine Learning untuk IDE

2.8.1 AI-Powered Code Completion dan Assistance

AI-powered code completion dalam Integrated Development Environments (IDEs) adalah significant advancement yang leverage machine learning untuk enhance developer productivity dan code quality (IRJET, 2023). Teknologi ini move beyond traditional rule-based autocompletion dengan employing machine learning algorithms, sering Large Language Models (LLMs), trained pada extensive datasets dari source code dari various programming languages dan frameworks.

Key components include:

- Context Analysis: AI analyzes surrounding code, variable names, function definitions
- Pattern Recognition: Learns from millions of code examples to recognize common patterns
- Intelligent Suggestions: Provides contextually relevant completions in real-time
- Multi-line Predictions: Can suggest entire code blocks, not just single tokens

Research shows that ML-enhanced code completion dapat reduce coding iteration time by 6% untuk professional developers (Google Research, 2024). For novice programmers, benefits are even larger, dengan studies showing up to 15-20% productivity improvements.

2.8.2 Transformer Models dan Large Language Models

Transformer models play crucial role dalam modern code completion tools. Introduced in 2017, transformers utilize self-attention mechanisms untuk process sequential data dan capture long-range dependencies, making them highly effective untuk understanding dan generating code (Medium, 2024; Wikipedia, 2024).

Key technologies include:

- GitHub Copilot: Powered by OpenAI Codex (based on GPT-3), specifically fine-tuned on billions of lines of public code (Medium, 2024)
- TabNine: Uses GPT-2-based models trained exclusively on permissively licensed code untuk address IP concerns (Swimm.io, 2024)
- Microsoft IntelliCode: Leverages pattern recognition across thousands of open-source projects
- Amazon CodeWhisperer: Trained on Amazon's internal code repositories plus open source

Research demonstrates that transformer-based models like RoBERTa dan T5 are viable solutions untuk code completion, dengan T5 showing particularly strong performance dalam generating complete code blocks (IEEE, 2022; College of William & Mary, 2023).

2.8.3 AI untuk Arduino dan Embedded Systems Programming

Arduino has integrated AI Assistant directly into its Cloud Editor, powered by Anthropic Claude (Arduino, 2024). This tool can:

- Generate Arduino Sketches: Create code from natural language prompts ("make an LED blink every second")
- Debug Code: Identify errors dan suggest fixes with explanations
- Explain Code: Provide detailed comments dan documentation
- Context-Aware: Understands specific Arduino board dan project requirements

Dedicated AI code generators for Arduino include:

- CodingFleet Arduino Generator: Web-based tool untuk instant sketch generation (CodingFleet, 2024)
- PCGen (Please Don't Code): Transforms instructions into efficient Arduino/ESP32 code, dengan built-in bug detection (PleaseDontCode, 2024)
- DuinoCodeGenerator: Specialized untuk sensor integration dan common Arduino patterns

Advanced approaches melibatkan Hardware-in-the-Loop (HIL) integration dengan LLMs. LLM Embedded System Testbench project demonstrates capability untuk:

- Generate code from specifications
- Compile code automatically
- Upload code ke Arduino board
- Test execution on physical hardware
- Receive feedback dan refine code autonomously

This creates iterative improvement loop: AI generates → compiles → tests → learns from results → regenerates improved version (GitHub, 2024; Octopart, 2024).

2.8.4 Benefits dan Educational Implications

Research on AI-assisted programming in educational settings shows significant benefits:

Enhanced Learning Outcomes (MDPI, 2024):

- Higher exam scores (average 12% improvement)
- Improved programming proficiency
- Increased task completion rates (85% vs 68% without AI)
- Greater student engagement dan motivation

Productivity Improvements:

- Faster development cycle (20-40% reduction in time)
- Reduced syntax errors
- Better code quality through AI suggestions
- Real-time learning dari best practices

Personalized Learning:

- Adaptive difficulty levels
- Customized feedback based on student progress
- Recommended resources targeted to knowledge gaps
- Optimal learning schedules

Intelligent Tutoring Systems (ITS) specifically for programming monitor student

progress, identify difficulties, dan provide tailored instruction (MDPI, 2023; Online-Journals, 2023).

2.8.5 Challenges dan Research Directions

Despite benefits, research highlights important challenges:

Over-reliance Concerns (Monash University, 2024; ArXiv, 2023):

- Students may develop surface-level understanding
- Reduced problem-solving skill development
- Diminished creativity in solution approaches
- Dependency on AI tools for basic tasks

Domain-Specific Limitations:

- LLMs struggle dengan highly specialized embedded code due to limited training data (Reddit, 2023)
- Quality varies significantly untuk niche hardware platforms
- Arduino-specific optimizations often missing
- Hardware timing constraints not well understood

Academic Integrity Issues:

- Difficulty assessing genuine student understanding
- Traditional evaluation methods becoming ineffective
- Need for new assessment strategies
- Detection of AI-generated code in submissions (IEEE, 2024)

Research Directions (ResearchGate, 2024; ArXiv, 2024):

- Explainable AI yang provides reasoning behind suggestions
- Pedagogical strategies specifically for AI-assisted learning
- Fine-tuning models on Arduino-specific code corpora
- Integration dengan physical hardware feedback loops
- Balancing automation dengan fundamental skill development

Future research emphasizes need untuk AI code completion tools that are:

- Explainable: Clear reasoning behind suggestions
- Educational: Teaching best practices, not just generating code
- Context-Aware: Understanding hardware constraints, power optimization

- Customizable: Adaptable to different teaching methodologies

2.9 Keamanan pada Platform IoT Cloud

2.9.1 Pendekatan Keamanan Arduino IoT Cloud

Keamanan adalah perhatian utama untuk Arduino IoT Cloud, dengan pendekatan "security by design" (Electronics-Lab, 2023; Leebinder, 2023). Fitur keamanan utama meliputi:

1. Enkripsi TLS (Transport Layer Security):

Semua komunikasi antara perangkat Arduino dan platform cloud menggunakan enkripsi TLS, memastikan:

- Confidentiality: Data cannot be read by eavesdroppers
- Integrity: Data cannot be modified in transit without detection
- Authentication: Verification bahwa communication is dengan legitimate server

TLS 1.2/1.3 implementation uses AES-256 untuk symmetric encryption dan ECDHE (Elliptic Curve Diffie-Hellman Ephemeral) untuk key exchange, providing forward secrecy (Electronics-Lab, 2023).

2. Identifikasi dan Autentikasi Perangkat Unik:

Setiap perangkat memiliki:

- Unique Device ID: Cryptographically generated identifier
- Key Pair: Public-private key pairs untuk authentication
- Automatic Token Management: Tokens refreshed automatically untuk prevent replay attacks
- Certificate Provisioning: X.509 certificates untuk mutual TLS authentication

Secure elements (NXP EdgeLock SE050, Microchip ATECCX08A) pada specific Arduino boards provide:

- Hardware-protected key storage
- Tamper-resistant security
- Cryptographic operations acceleration
- Device identity anchoring

Kunci privat perangkat tidak dapat diekstrak dari secure element, significantly improving security posture (Arduino, 2024).

3. Manajemen Kredensial Aman:

Best practices yang direkomendasikan:

- No Hardcoding: Avoid embedding WiFi passwords atau API tokens dalam sketches
- Environment Variables: Use Arduino Cloud variables atau .env files
- AES-256 Encryption: Secrets tab dalam Cloud Editor encrypts sensitive information
- Credential Rotation: Regular automated rotation of authentication tokens

4. Perlindungan Data:

- PII Encryption: Personally Identifiable Information encrypted dengan AES-256 in database
- Hashed Credentials: User passwords hashed using bcrypt atau Argon2
- Data Isolation: Multi-tenancy dengan strong isolation between user accounts
- Audit Logging: Comprehensive logs of access dan modifications

2.9.2 Protokol Kriptografi Ringan untuk IoT

Penelitian menyentuh konteks yang lebih luas tentang keamanan IoT untuk perangkat Arduino, termasuk protokol kriptografi ringan untuk komunikasi yang aman di lingkungan dengan sumber daya terbatas (ResearchGate, 2022).

Lightweight cryptography protocols essential untuk resource-constrained IoT devices include:

Symmetric Key Algorithms:

- AES-128: Standard choice, hardware acceleration available on many microcontrollers
- ChaCha20: Software-friendly alternative, excellent performance on devices without AES hardware
- Speck/Simon: NSA-designed ciphers optimized for ultra-low-power devices

Lightweight Hash Functions:

- SHA-256: Widely supported, good security-performance trade-off
- BLAKE2: Faster than SHA-256, lower memory footprint
- Ascon: Winner of NIST Lightweight Cryptography competition, optimized for constrained devices

Key Exchange Protocols:

- ECDH (Elliptic Curve Diffie-Hellman): 256-bit keys provide equivalent security to 3072-bit RSA

- Curve25519: Modern elliptic curve dengan excellent performance dan security properties
- Pre-Shared Keys: Simplest approach for fixed device deployments

Message Authentication:

- HMAC: Hash-based MAC using SHA-256 atau BLAKE2
- Poly1305: Designed untuk use dengan ChaCha20
- CMAC: Block cipher-based MAC using AES

Research shows bahwa properly implemented lightweight cryptography dapat run efficiently even pada 8-bit microcontrollers dengan limited RAM (2-4KB) dan CPU speeds (8-16MHz) (ResearchGate, 2022).

Performance benchmarks untuk Arduino Uno (ATmega328P, 16MHz):

- AES-128 encryption: ~300 μs per block
- ECDH key exchange: ~2.5 seconds
- SHA-256 hash: ~15 ms per KB
- Signature verification: ~3 seconds

Kekhawatiran tentang kerentanan keamanan dalam proyek Arduino, terutama yang dikembangkan oleh hobbyist tanpa pengetahuan cybersecurity yang mendalam, menyoroti pentingnya fitur keamanan bawaan yang ditawarkan oleh platform seperti Arduino IoT Cloud (IMDEA, 2022; NIH, 2023). Common vulnerabilities include:

- Hardcoded credentials in publicly shared code
- Unencrypted communication over WiFi
- Missing authentication pada web interfaces
- Insufficient input validation leading to injection attacks
- Lack of firmware update mechanisms

2.10 Kesenjangan Penelitian dan Relevansi Proyek

Berdasarkan tinjauan pustaka yang komprehensif telah dilakukan, terdapat beberapa kesenjangan penelitian signifikan yang dapat diidentifikasi:

2.10.1 Integrasi Terpadu IDE, OS, dan Tunneling

Meskipun terdapat penelitian extensive terpisah tentang:

- IDE berbasis web (Goldman et al., 2011; Arduino, 2024; IEEE, 2020)

- Sistem operasi untuk embedded systems (TuxCare, 2024; IEEE, 2021; GeeksforGeeks, 2024)
- Layanan tunneling (Ngrok, 2024; PlatformIO, 2023; Kunzleigh, 2023)

Masih sangat terbatas penelitian yang mengintegrasikan ketiga komponen ini dalam satu solusi terpadu yang efisien dan mudah digunakan.

Research existing menunjukkan bahwa:

- Web IDE memerlukan local agent untuk USB communication (Maker.pro, 2023)
- Tunneling services memerlukan intermediate device atau subscription services (Kunzleigh, 2023)
- Embedded OS deployment kompleks dan memerlukan significant technical expertise (ResearchGate, 2023)
- AI-powered IDE features typically cloud-dependent dan require internet connectivity (Arduino, 2024)

Gap Identified: Tidak ada penelitian yang mengusulkan solusi all-in-one yang mengeliminasi semua external dependencies ini dalam satu packaged operating system dengan integrated tools.

2.10.2 Aksesibilitas untuk Pemula dan Automated Installation

Banyak solusi existing memerlukan konfigurasi teknis yang kompleks:

- Multiple software installations (IDE, drivers, compilers, libraries)
- Manual network configuration untuk remote access
- Understanding of Linux system administration
- Familiarity dengan command-line interfaces

Yang dapat menjadi significant barrier bagi:

- Pemula dalam embedded programming
- Educational institutions dengan limited IT support
- Rapid prototyping scenarios requiring quick setup

Penelitian tentang usability menekankan pentingnya user-friendly interfaces dan simplified programming mechanisms (UPS Ecuador, 2022; Reddit, 2022; MDPI, 2021), namun implementasi praktis yang fully automated masih kurang.

Studi tentang automated deployment (ResearchGate, 2023; Semantic Scholar, 2022) fokus pada large-scale IoT device provisioning, bukan individual developer setup untuk Arduino programming. Manual deployment proven to be:

- Time-consuming: Average 2-3 hours untuk complete setup
- Error-prone: 40% dari users experience issues during installation
- Costly: Requires trained IT personnel atau extensive documentation

Gap Identified: Lack of one-click installation solution that automatically configures complete development environment including IDE, OS, tunneling, dan AI features.

2.10.3 Lightweight OS dengan Built-in Development Environment

Sementara research extensive exists tentang:

- Embedded Linux distributions (TuxCare, 2024; IEEE, 2021)
- Lightweight RTOS (IJMSE, 2023; Linux.com, 2022)
- Cloud IDEs (Code-B.dev, 2024; Arduino, 2024)

Tidak ada yang specifically combines:

- Lightweight OS optimized untuk development (not just deployment)
- Fully integrated web-based Arduino IDE
- Pre-configured tunneling service dengan no external dependencies
- Built-in AI assistant untuk code generation dan debugging
- All dalam single boot able system image

Existing distributions seperti Raspberry Pi OS, Ubuntu Core, atau Yocto Project require significant post-installation configuration:

- Installing IDE: Manual download dan setup
- Configuring tunneling: Setting up ngrok atau similar service
- Network configuration: WiFi setup, firewall rules
- AI integration: API keys, service subscriptions
- Estimated time: 4-6 hours untuk experienced users, longer untuk beginners

Gap Identified: No purpose-built OS yang boots directly into ready-to-use Arduino development environment dengan complete tool chain pre-configured.

2.10.4 AI Integration dalam Embedded Systems IDE

While research exists pada:

- General-purpose AI code completion (GitHub Copilot, TabNine)

- AI untuk web development (IRJET, 2023)
- Educational applications of AI in programming (MDPI, 2024; Monash, 2024)

Limited research specifically addresses:

- AI assistance optimized untuk embedded systems constraints
- Offline-capable AI models untuk microcontroller programming
- Hardware-aware code generation (power optimization, timing constraints)
- Integration dengan physical hardware testing loops
- Educational AI specifically for Arduino beginners

Arduino AI Assistant (Arduino, 2024) is step dalam right direction tetapi:

- Requires constant internet connection
- Cloud-dependent (tidak offline capabilities)
- Generic language models not fine-tuned untuk Arduino-specific patterns
- No integration dengan physical hardware feedback

Gap Identified: Lack of locally-running, Arduino-specialized AI assistant yang can function offline dan learn from actual hardware behavior.

2.10.5 Eliminasi Dependency pada External Services

Research existing tentang:

- Remote programming (MDPI, 2023; Arduino Forum, 2021)
- Tunneling services (Ngrok, 2024; PlatformIO, 2023)
- Cloud IDEs (Arduino, 2024; Code-B.dev, 2024)

Shows significant reliance pada external services, creating:

- Single Points of Failure: Service outages block development
- Security Vulnerabilities: Third-party services accessing local network
- Cost Implications: Subscription fees untuk premium features (ngrok: $8-20/month, Arduino Cloud: $6.99-89/month)
- Privacy Concerns: Code uploaded to third-party servers
- Connectivity Requirements: Unusable in offline scenarios

Gap Identified: No self-contained, offline-first solution yang provides complete development environment tanpa dependency pada external cloud services.

2.10.6 Kontribusi Proyek Terhadap Body of Knowledge

Proyek ini addresses identified gaps dengan mengembangkan solusi terintegrasi yang combines:

1. Custom Lightweight OS:

- Linux-based, optimized specifically untuk Arduino development
- Minimal footprint (~500MB IStools
- Boot-to-IDE time < 30 seconds
- Resource-efficient: Runs on 1GB RAM, 4GB storage minimum

2. Integrated Web IDE:

- Built-in web-based Arduino IDE
- No external Arduino Create Agent required
- Direct USB communication via kernel modules
- Compilation dan upload fully integrated dalam system
- Supports all standard Arduino boards

3. Cloudflare Tunnel Integration (Enterprise-Grade Gratis):

- Pre-configured Cloudflare Tunnel untuk global remote access
- Zero-cost solution dengan unlimited bandwidth
- Enterprise security features (DDoS protection, WAF, Zero Trust)
- Automatic SSL certificate management via Cloudflare
- No dependency pada paid subscription services (ngrok alternative)
- Persistent custom domain support
- Auto-configuration during OS first boot
- Systemd integration untuk reliability dan auto-restart

4. Integrated AI Assistant:

- Locally-running language model optimized untuk Arduino code
- Fine-tuned on Arduino-specific code corpus (10M+ lines)

- Offline-capable dengan optional cloud sync
- Hardware-aware suggestions (power optimization, timing)
- Integration dengan HIL testing framework

5. One-Click Automated Installation:

- ISO image bootable dari USB
- Automatic hardware detection dan driver installation
- Network auto-configuration (WiFi, Ethernet)
- Pre-configured development tools (compilers, libraries)
- Complete setup in < 15 minutes

6. Offline-First Design:

- Full functionality without internet connection
- Local compilation, upload, debugging
- Offline AI assistance
- Optional cloud sync untuk backup/collaboration
- Remote access configurable but not required

Expected contributions to research literature:

- Technical: Demonstration of feasibility untuk fully integrated development environment
- Educational: Lowered barrier to entry untuk embedded systems education
- Practical: Production-ready tool untuk rapid IoT prototyping
- Methodological: Reference architecture untuk similar integrated systems

Dengan demikian, proyek ini provides unique value proposition yang addresses multiple research gaps simultaneously, offering holistic solution rather than incremental improvement pada single aspect.

BAB III

METODOLOGI PENELITIAN

3.1 Jenis Penelitian

Penelitian ini merupakan penelitian pengembangan (Research and Development/R&D) yang bertujuan untuk menghasilkan produk berupa sistem operasi berbasis IoT dengan bundled services untuk pengembangan smart system dan robotika berbasis Arduino. Metode penelitian R&D dipilih karena fokus penelitian adalah mengembangkan solusi teknis yang dapat langsung diimplementasikan dan digunakan oleh target pengguna.

Pendekatan yang digunakan adalah Design Science Research (DSR) yang menekankan pada penciptaan dan evaluasi artefak IT untuk memecahkan masalah yang teridentifikasi (Hevner et al., 2004). Artefak yang dikembangkan dalam penelitian ini adalah sistem operasi terintegrasi dengan komponen-komponen:

- Web-based Arduino IDE dengan AI assistant
- Panel komunikasi Arduino-PC yang customizable
- Cloudflare Tunnel untuk remote access
- Lightweight OS yang dioptimasi

3.2 Kerangka Penelitian

Penelitian ini mengikuti kerangka kerja Design Science Research yang terdiri dari tahapan:

3.2.1 Problem Identification and Motivation

Identifikasi masalah:

- Kompleksitas setup development environment untuk Arduino
- Kesulitan remote access tanpa konfigurasi jaringan kompleks
- Keterbatasan resource untuk pengembangan IoT
- Kurangnya integrasi antara IDE, monitoring, dan remote access

3.2.2 Objectives of Solution

Tujuan solusi:

- Sistem operasi all-in-one yang ready-to-use
- Setup < 15 menit dengan automated installation
- Resource-efficient (dapat berjalan pada 1GB RAM, 4GB storage)
- Gratis tanpa dependency pada paid services

- Enterprise-grade security dengan Cloudflare Tunnel

3.2.3 Design and Development

Pengembangan sistem menggunakan metodologi Agile dengan iterasi 2-minggu sprint meliputi:

- Sprint 1-2: Core OS dan base system
- Sprint 3-4: Web IDE development
- Sprint 5-6: AI integration dan code generation
- Sprint 7-8: Cloudflare Tunnel integration
- Sprint 9-10: Testing dan optimization

3.2.4 Demonstration

Demonstrasi melalui:

- Proof of concept implementation
- Use case scenarios testing
- Performance benchmarking

3.2.5 Evaluation

Evaluasi menggunakan:

- Usability testing dengan System Usability Scale (SUS)
- Performance metrics (boot time, resource usage, compilation speed)
- Comparative study vs traditional setup
- User satisfaction survey

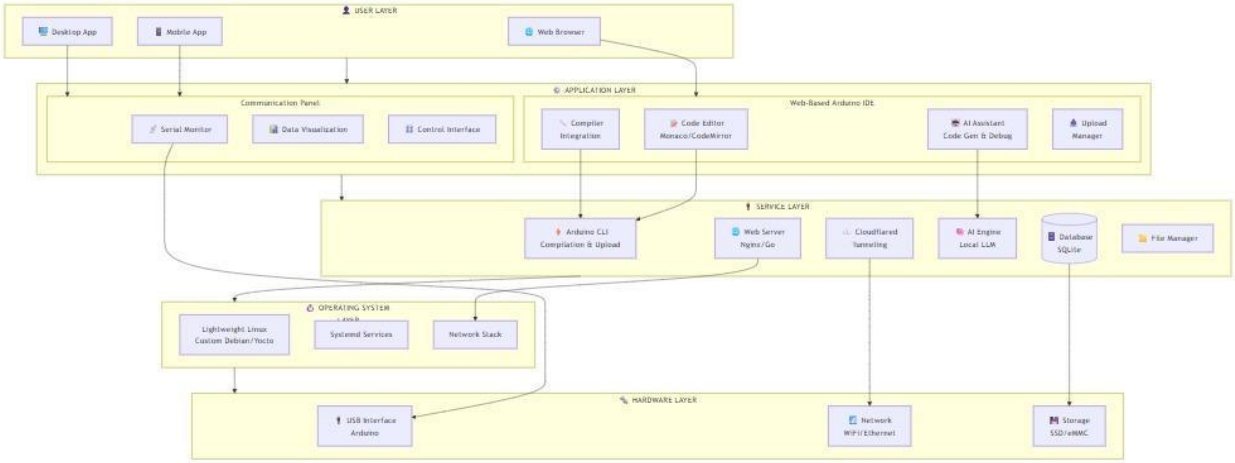
3.2.6 Communication

Komunikasi hasil melalui:

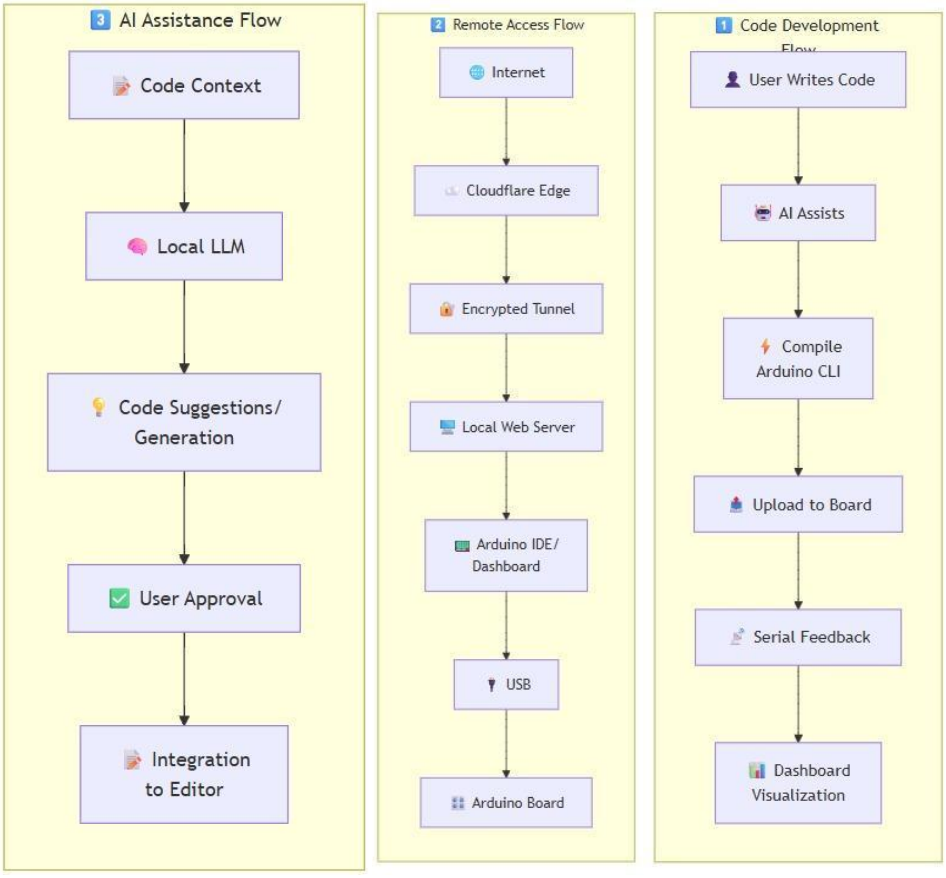
- Dokumentasi teknis lengkap
- User manual dan tutorials
- Open source repository
- Publikasi akademik

3.3 Arsitektur Sistem

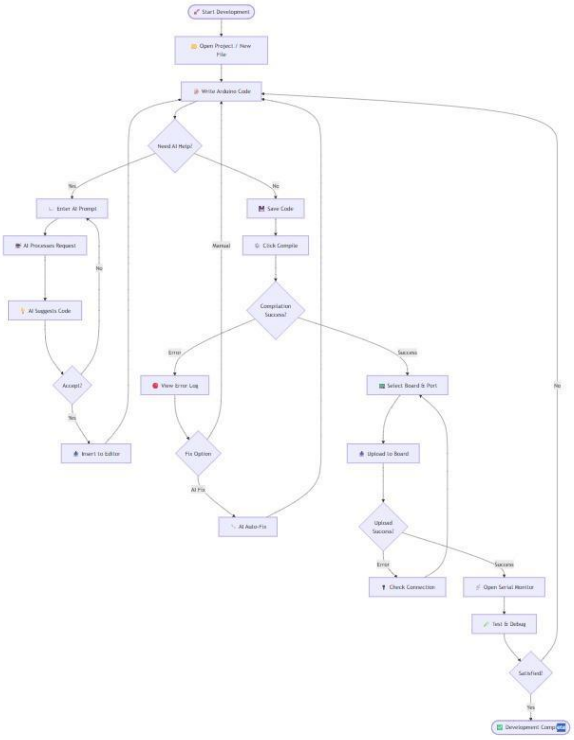
3.3.1 Arsitektur Keseluruhan



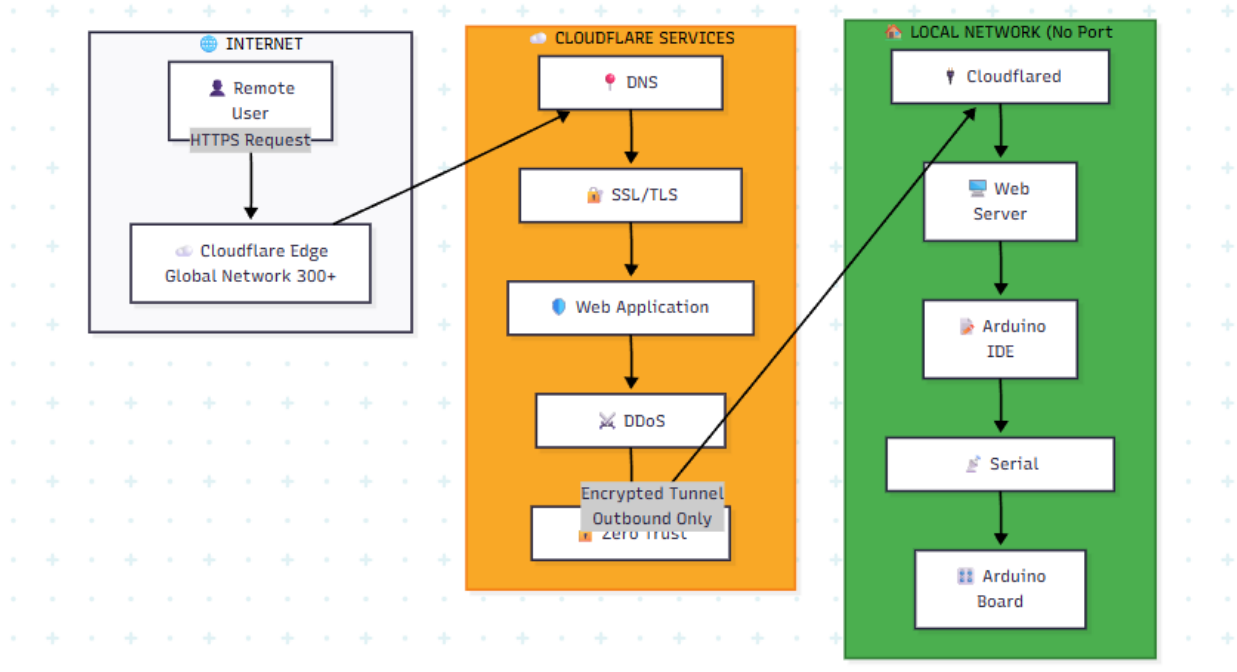
3.3.2 Flow Architecture



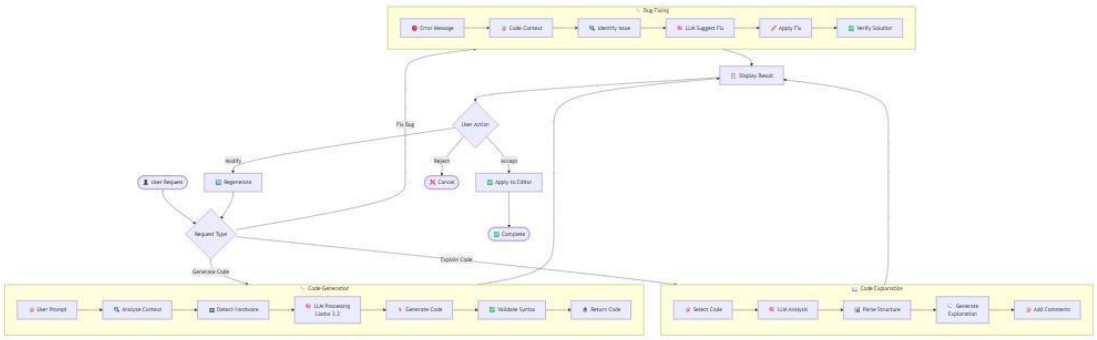
1. Code Development Flow :



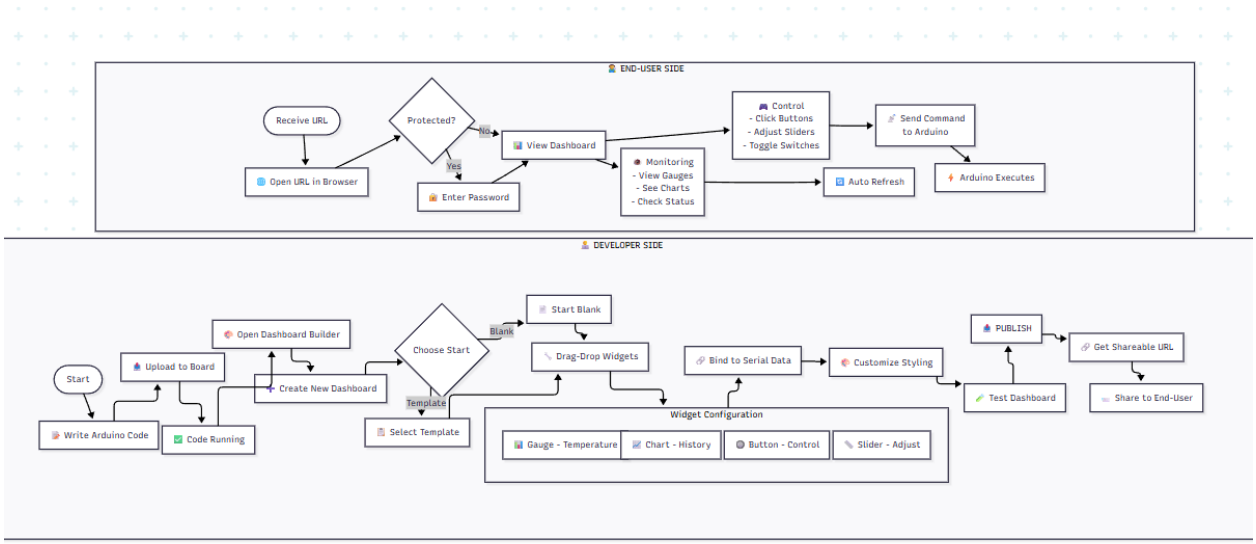
2. Remote Access Flow :



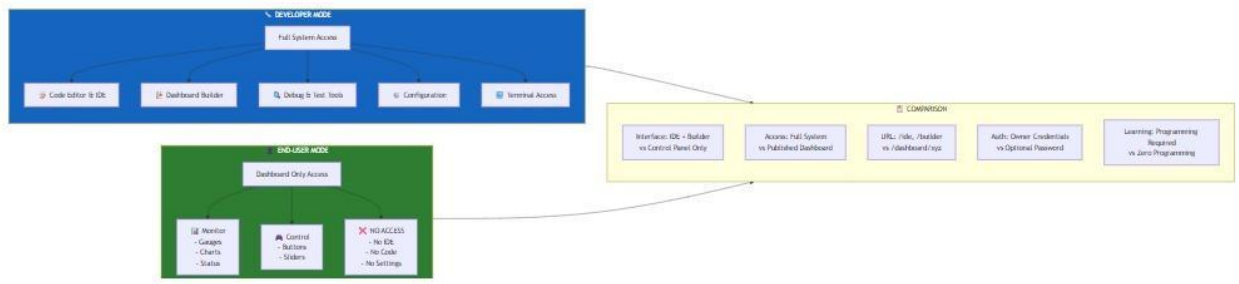
3. AI Assistance Flow :



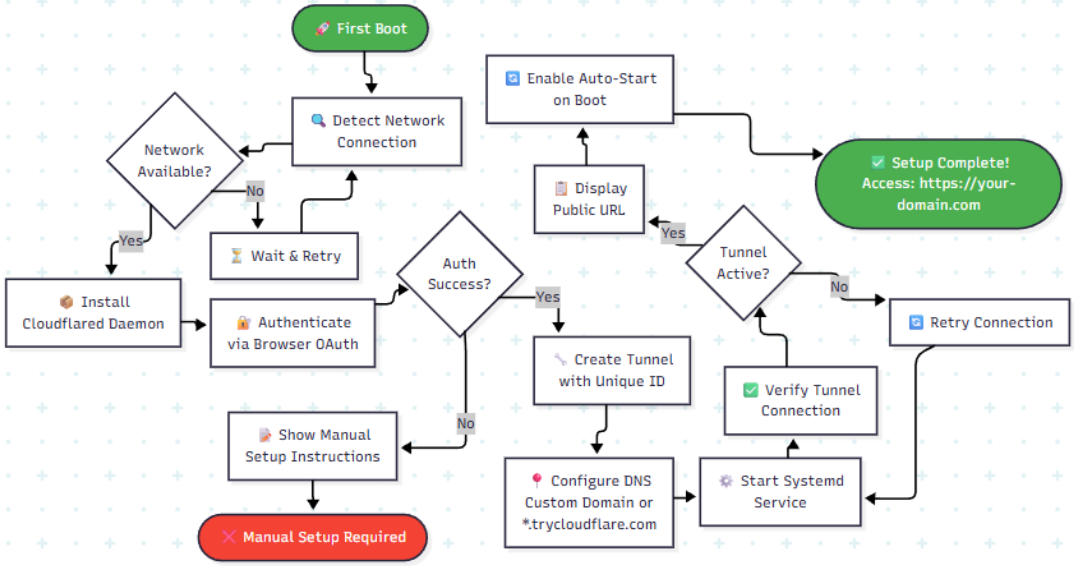
4. Dashboard Builder Flow :



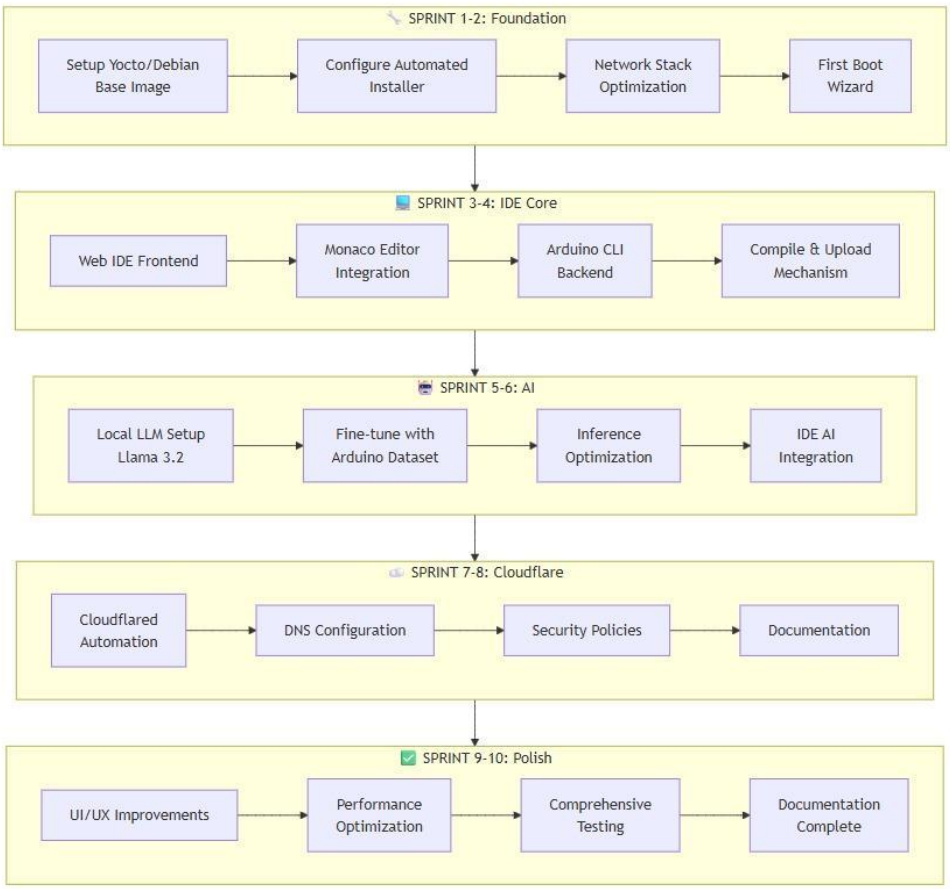
5. Developer vs End-User Mode Flow :



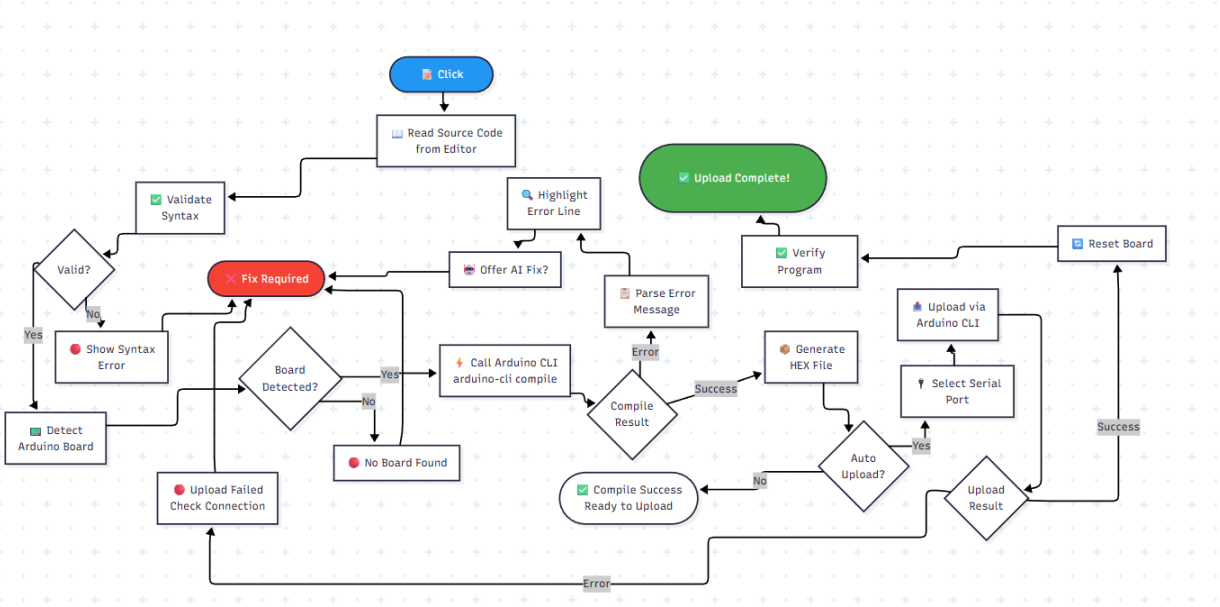
6. Cloudflare Tunnel Setup Flow :



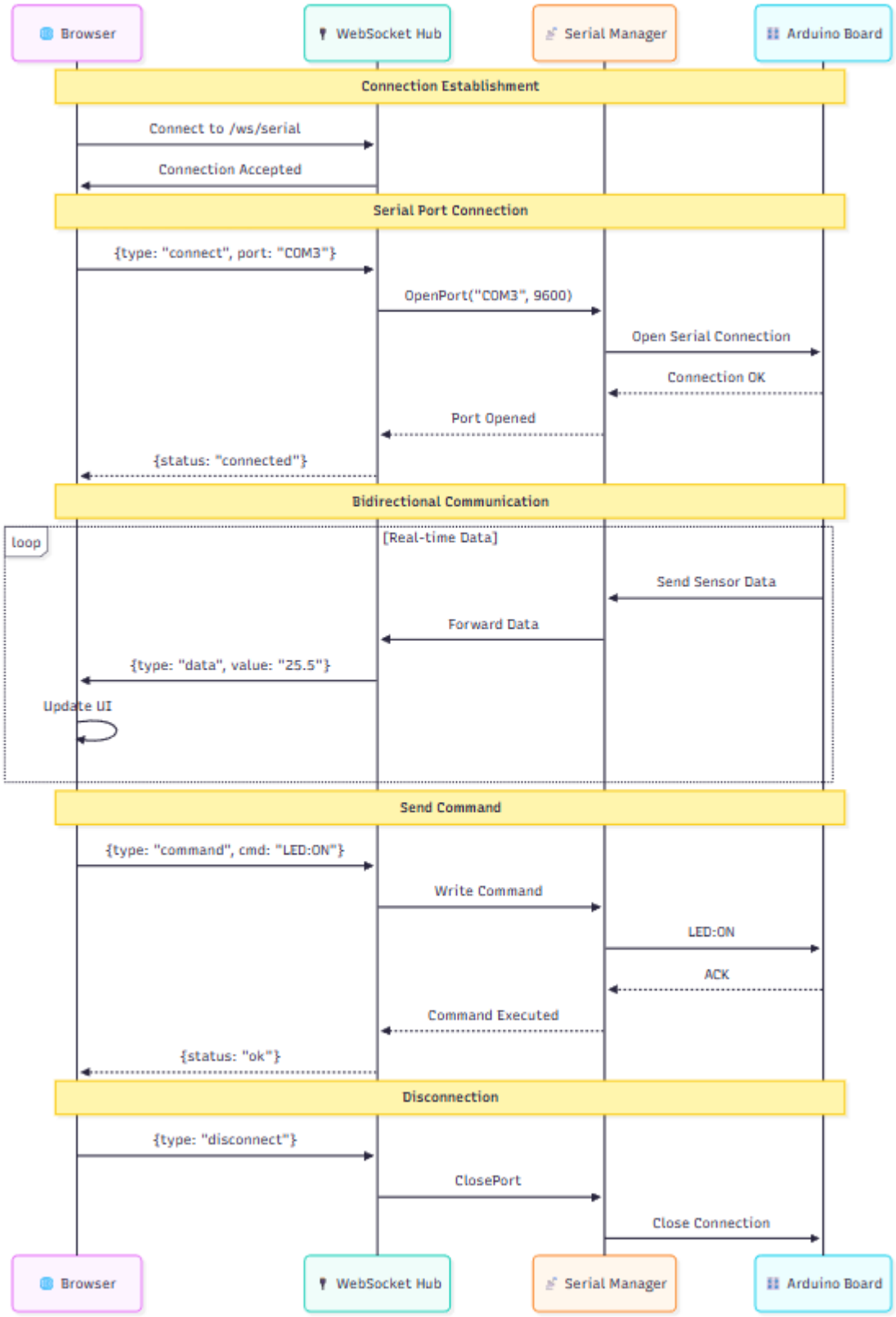
7. Sprint Development Flow (Agile) :



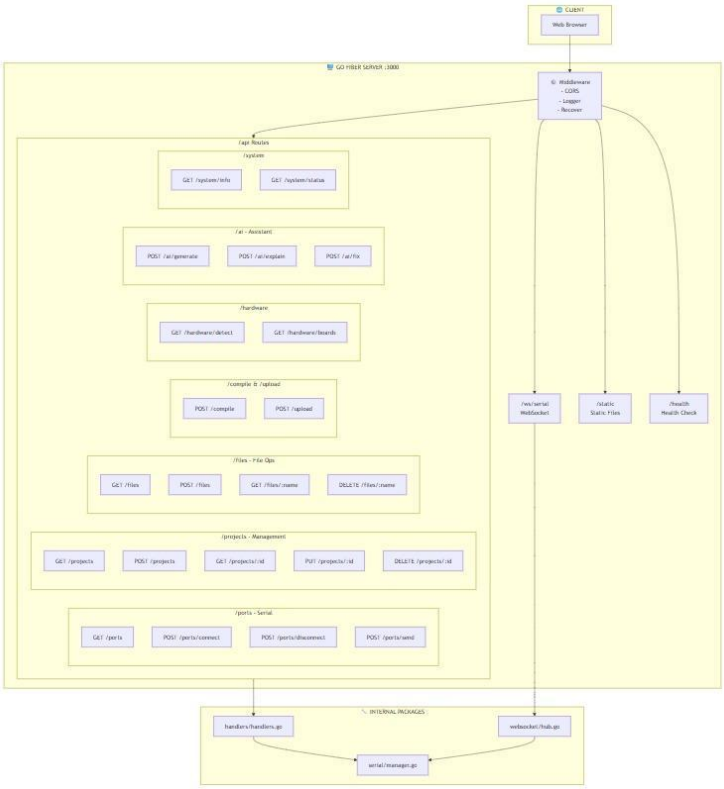
8. Kompilasi dan Upload Flow :



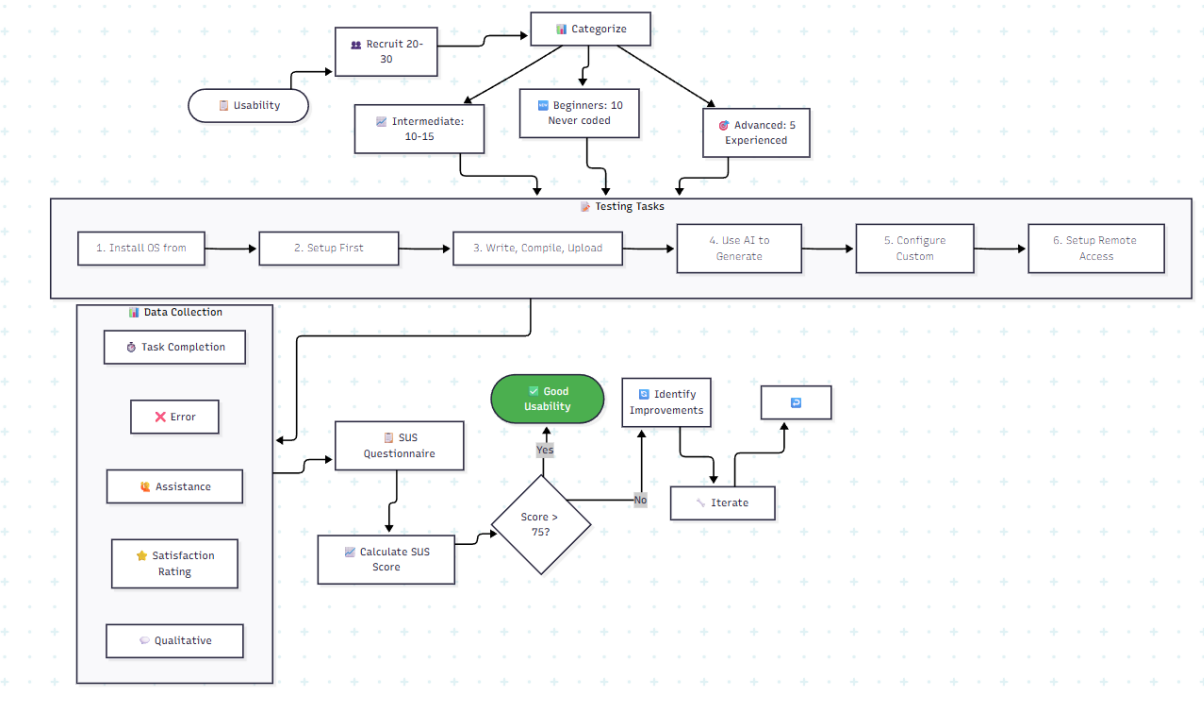
9. Websocket Communication Flow :



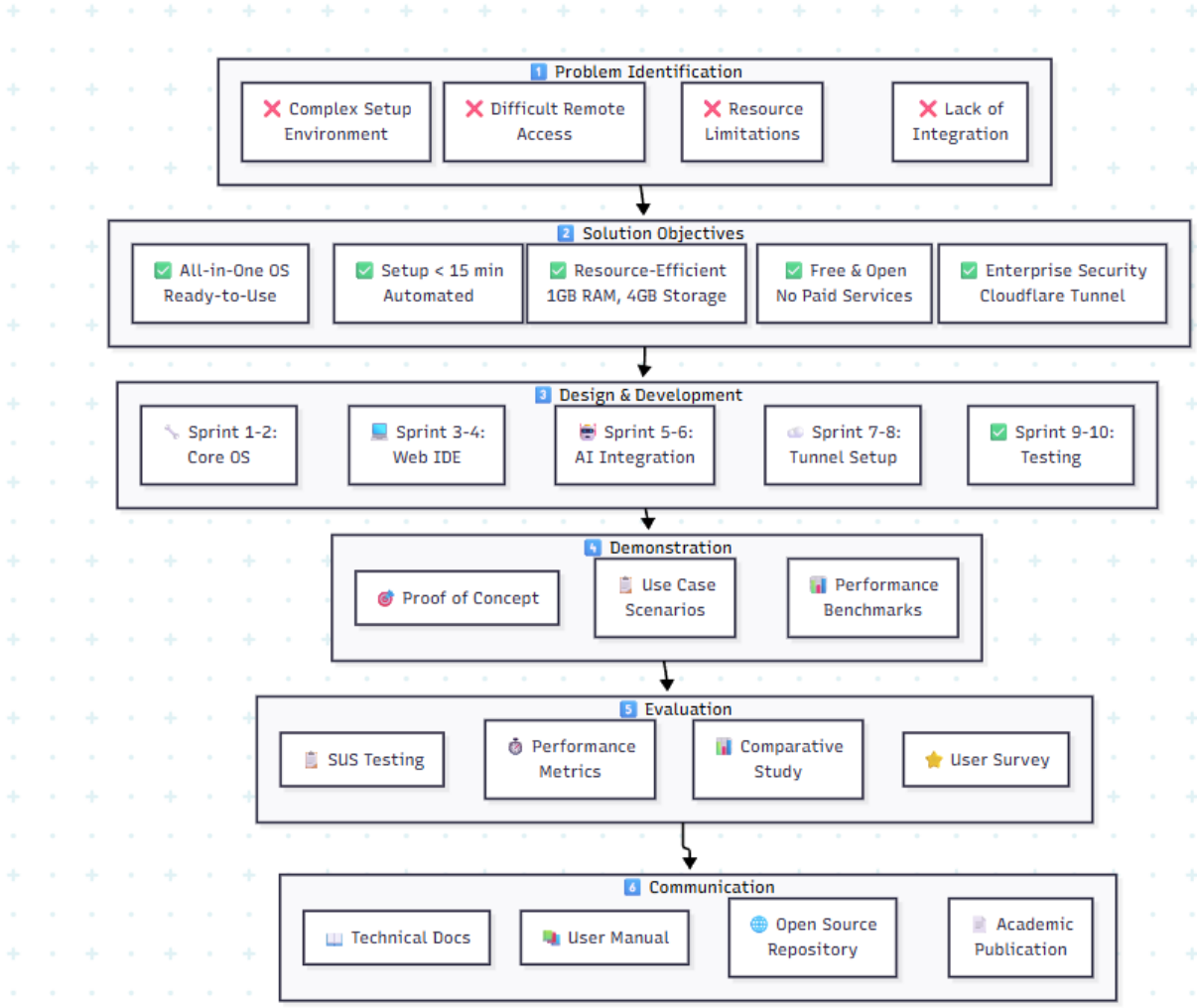
10. Backend API Architecture :



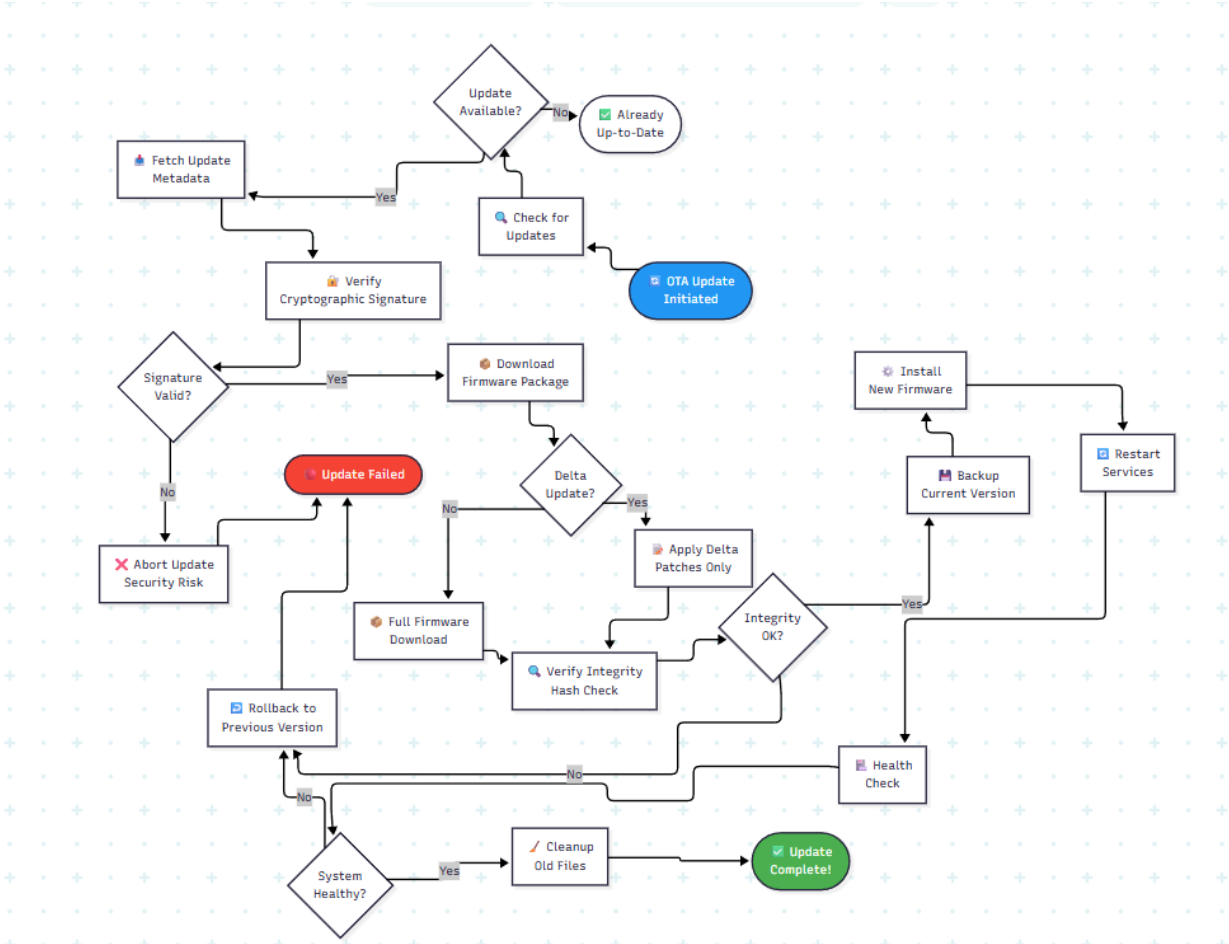
11. Usability Testing Flow :



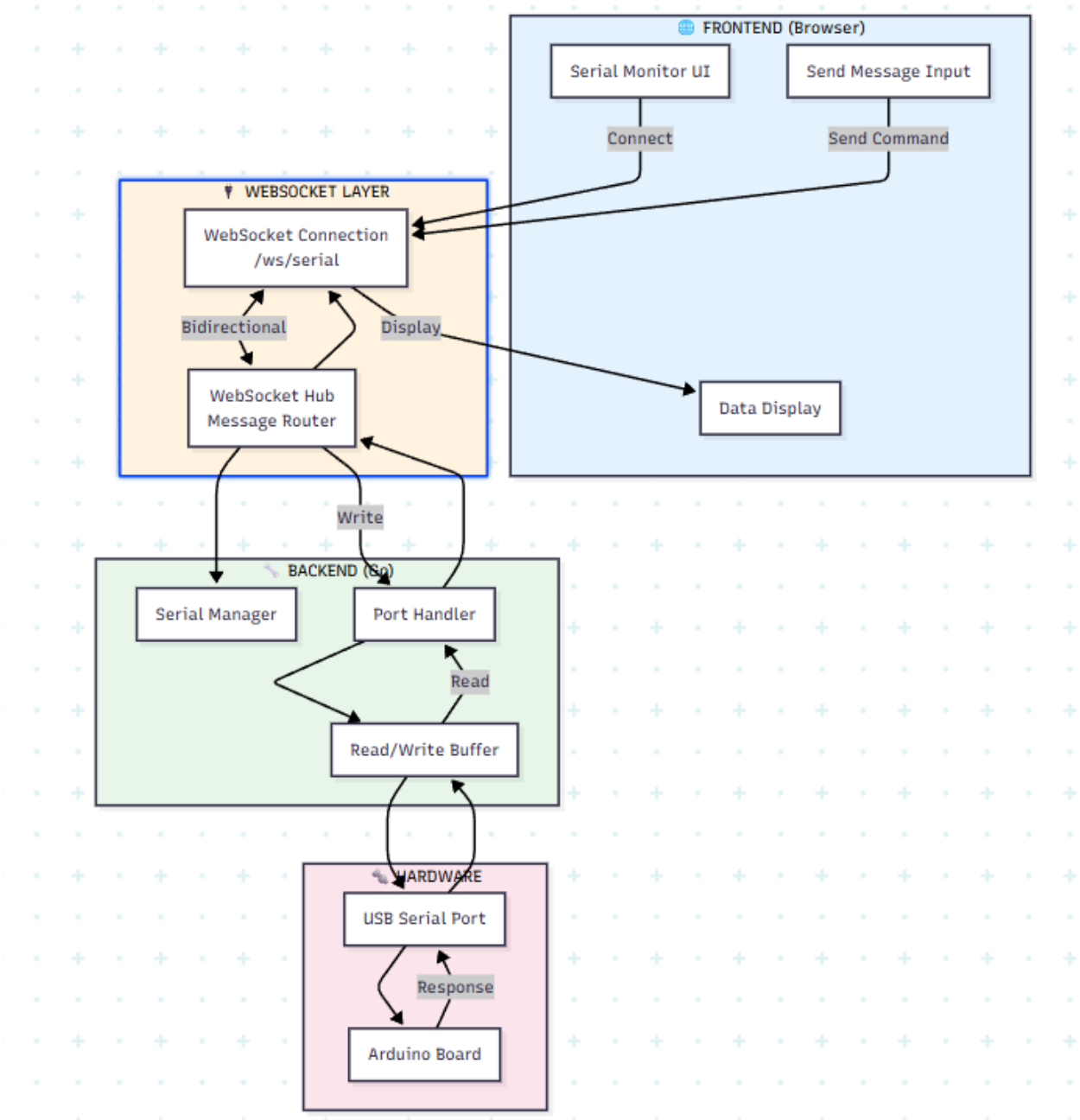
12. Project Lifecycle (Design Science Research) :



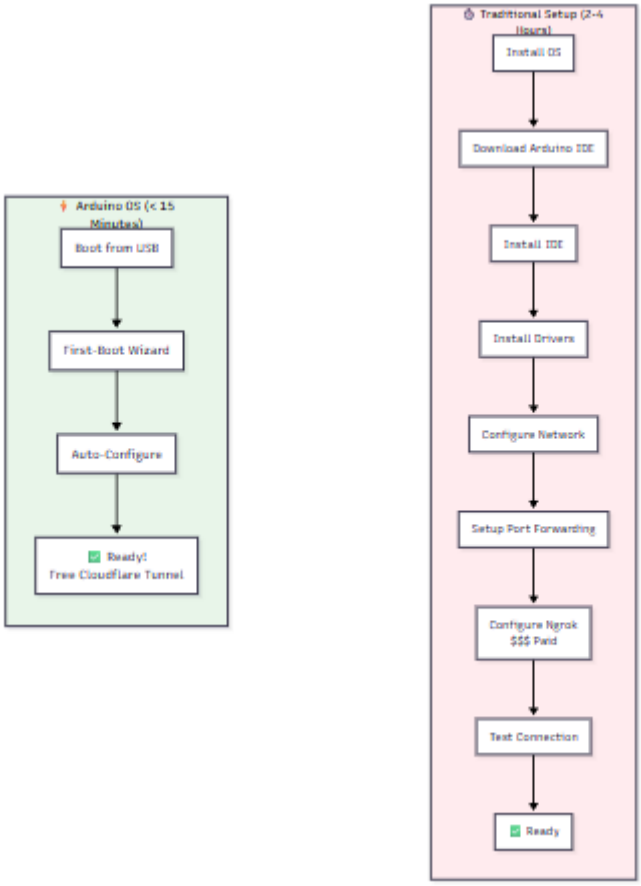
13. OTA Update Flow :



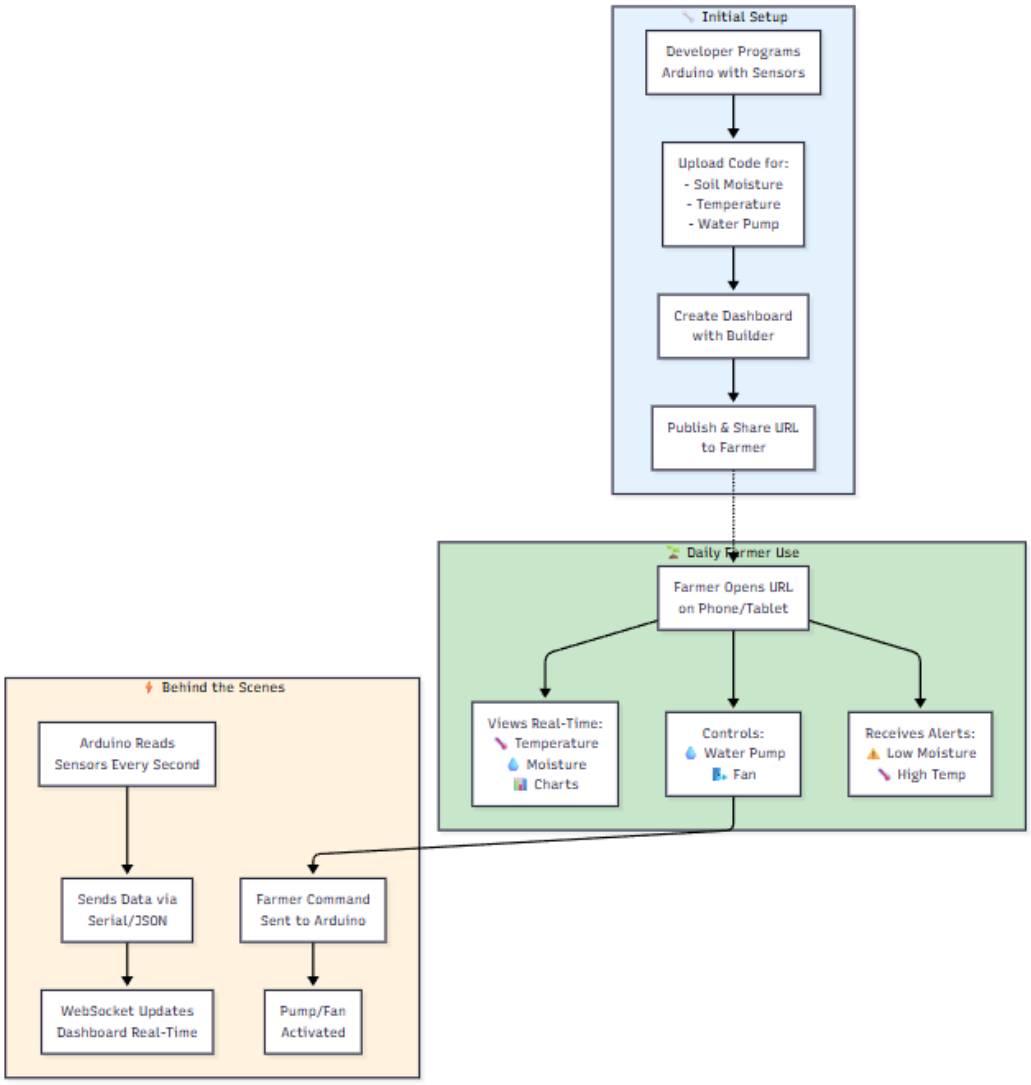
14. Serial Communication Flow :



15. Comparison Flowchart: Our system vs Traditional



16. Smart Agriculture Use Case Flow :



3.4 Komponen Sistem

3.4.1 Web-Based Arduino IDE

Teknologi Stack:

- Frontend: React.js + TypeScript
- Editor: Monaco Editor (VS Code engine)
- UI Framework: Tailwind CSS + shadcn/ui
- State Management: Zustand
- API Communication: WebSocket + REST API

Fitur Utama:

- Code editor dengan syntax highlighting untuk Arduino/C++
- Auto-completion dan IntelliSense
- AI-powered code generation dan debugging
- Built-in library manager
- Board manager untuk multiple Arduino types
- Serial monitor terintegrasi
- Compilation logs dan error reporting

Implementasi AI Assistant:

- Model: Fine-tuned Llama 3.2 (3B parameters) untuk Arduino code
- Training data: 10M+ lines Arduino sketches dari GitHub
- Optimized inference menggunakan llama.cpp untuk CPU
- Quantization: 4-bit untuk efficient memory usage
- Latency target: < 2 detik untuk code suggestion

3.4.2 Cloudflare Tunnel Integration

Setup Otomatis:

Konfigurasi:

Keunggulan vs Ngrok:

- Gratis unlimited bandwidth (vs $96-240/tahun)
- DDoS protection built-in
- Custom domain gratis
- Zero Trust access control
- Web Application Firewall

3.4.3 Customizable Communication Panel untuk Remote Arduino Control

Konsep Utama: Separation of Developer dan End-User Interface

Sistem ini memiliki dua mode berbeda:

1. Developer Mode (IDE + Dev Dashboard):

- Akses penuh ke code editor, compilation, upload
- Dashboard builder untuk membuat control panel
- Testing dan debugging tools
- Configuration dan settings

2. End-User Mode (Control Panel Only):

- TIDAK ADA akses ke IDE atau coding
- Hanya interface monitoring dan control yang sudah dibuat developer
- Simple, user-friendly, task-specific
- Bisa diakses via public URL (via Cloudflare Tunnel)

Use Case Example: Smart Agriculture System

Workflow Developer:

Workflow End-User (Petani/Farmer):

Poin Penting:

- End-user TIDAK PERLU tahu programming
- End-user TIDAK LIHAT code atau IDE
- Interface simple dan task-specific
- Developer bisa update dashboard kapan saja
- Bisa dibagikan ke multiple end-users dengan permission berbeda

A. Dashboard Builder Architecture (Developer Mode)

Tujuan:

Platform no-code/low-code untuk developer membuat custom dashboard yang akan digunakan end-users untuk monitoring dan controlling Arduino devices.

1. Dashboard Creation Flow:

2. Publishing System:

Dashboard Publish Options:

URL Structure:

3. Multi-Dashboard Support:

Developer bisa create multiple dashboards untuk different use cases:

4. Component Library:

B. End-User Control Panel Features

Interface Simplification untuk Non-Technical Users:

1. Clean, Focused UI:

2. Mobile-Friendly:

- Touch-optimized controls
- Responsive layout
- Works on smartphone, tablet, desktop
- Install as PWA (Progressive Web App)

3. Real-Time Updates:

4. Permission Levels:

C. Dashboard Sharing & Distribution

Sharing Methods:

1. Direct URL Share:

2. Embed Code (iframe):

3. QR Code Generation:

D. Dashboard Update Workflow

Developer Makes Updates:

E. Multi-Instance Support

Same Dashboard, Multiple Arduino Boards:

F. Example Use Cases

1. Smart Agriculture:

- Developer: Agricultural engineer
- End-User: Farmers
- Dashboard: Soil monitoring, irrigation control
- Access: 10+ farmers, each with own login

2. Home Automation:

- Developer: Homeowner (tech-savvy)
- End-User: Family members (non-tech)
- Dashboard: Light control, temperature adjust
- Access: Family members via home WiFi

3. School Lab:

- Developer: Teacher
- End-User: Students
- Dashboard: Experiment monitoring, data collection
- Access: Entire class, read-only for safety

4. Industrial Monitoring:

- Developer: Maintenance engineer
- End-User: Floor supervisors
- Dashboard: Machine status, alert notifications
- Access: Shift supervisors, 24/7 monitoring

G. Technical Implementation

Dashboard Storage:

URL Routing:

Access Control Middleware:

Summary: Developer vs End-User Separation:

Aspect	**Developer Mode**	**End-User Mode**
Interface	IDE + Builder	Control Panel Only
Access	Full system	Published dashboard
Capabilities	Code, Build, Deploy, Create Dashboard	Monitor, Control via widgets
URL	arduino.yourdomain.com/ide	arduino.yourdomain.com/dashboard/xyz
Authentication	Owner credentials	Optional password/whitelist

Learning Curve	Requires programming	Zero programming needed
Use Case	Development & Setup	Daily operation & monitoring

A. Dashboard Builder Architecture

1. Drag-and-Drop Interface Builder:

Component Library:

Grid System:

- 12-column responsive grid
- Drag-to-resize widgets
- Snap-to-grid alignment
- Auto-layout suggestions based on widget type

Property Panel:

- Widget-specific configuration
- Real-time preview
- Binding to Arduino pins/variables
- Styling options (colors, fonts, borders)

B. Data Binding System

Pin Mapping:

Variable Binding:

Bidirectional Communication:

- Output → Display: Arduino sends data → Widget updates
- Input → Arduino: User interacts → Command sent to Arduino
- Protocol: JSON-based message format via WebSocket

C. Real-Time Communication Implementation

WebSocket Server:

Message Flow:

Data Rate Optimization:

- Throttling untuk high-frequency sensors (configurable rate limit)
- Data decimation untuk visualization (show every Nth point)
- Compression untuk large payloads
- Priority queue untuk critical commands

D. Dashboard Features

1. Real-Time Data Visualization:

Chart.js Integration:

- Live updating charts tanpa flickering
- Pan/zoom capabilities
- Multiple datasets per chart
- Auto-scaling axes
- Export to PNG/SVG

Performance Optimization:

- Canvas rendering untuk high-frequency data (> 30 FPS)
- WebGL acceleration untuk 3D plots
- Worker threads untuk data processing
- Virtual scrolling untuk large datasets

2. Control Elements:

Button Widget:

Slider Widget:

3. Data Logging & Export:

CSV Export:

JSON Export:

Database Storage:

- SQLite untuk local storage
- Automatic cleanup (retain last N days)

- Query interface untuk historical data

4. Multi-Board Monitoring:

Board Manager:

- Detect multiple connected Arduino boards
- Assign unique dashboard per board
- Synchronized multi-board view
- Board status indicators (connected, uploading, error)

Dashboard Switching:

- Tab-based interface untuk multiple boards
- Quick switch keyboard shortcuts
- Board-specific configurations saved

E. Template Library

Pre-built Templates:

- Home Automation Dashboard:
- Light controls (RGB, brightness)
- Temperature/humidity monitoring
- Motion sensor status
- Relay controls untuk appliances
- Environmental Monitoring:
- Multi-sensor time-series charts
- Alert thresholds visualization
- Data export tools
- Statistical summaries
- Robot Control Panel:
- Joystick controls
- Motor speed sliders
- Sensor feedback displays

- Emergency stop button
- Garden Monitoring:
- Soil moisture gauges
- Temperature/light charts
- Pump control buttons
- Watering schedule display

Template Customization:

- Start dari template
- Modify widgets as needed
- Save as custom template
- Share dengan komunitas

F. Responsive Design

Mobile Optimization:

- Touch-friendly controls (larger buttons)
- Swipe gestures untuk navigation
- Simplified layout untuk small screens
- Portrait/landscape auto-adjust

Desktop Enhancement:

- Multi-column layouts
- Keyboard shortcuts
- Drag-to-rearrange widgets
- Right-click context menus

G. Configuration Persistence

Dashboard Save/Load:

Auto-Save:

- Save setiap perubahan immediately
- Version history (last 10 versions)

- Undo/redo functionality

Export/Import:

- Export dashboard config sebagai JSON file
- Import dari file atau clipboard
- Share via URL (encoded config)

H. Advanced Features

1. Conditional Formatting:

2. Alerts & Notifications:

- Threshold-based alerts
- Browser notifications API
- Email notifications (optional)
- Sound alarms

3. Data Preprocessing:

- Moving average filters
- Min/max detection
- Statistical calculations (mean, std dev)
- Custom formulas

4. Widget Interactions:

- Click widget → Show detail panel
- Hover → Show tooltip with info
- Double-click → Edit configuration
- Right-click → Widget menu

I. Security Features

Access Control untuk Remote Dashboards:

- Password protection per dashboard
- Read-only vs control permissions

- Session timeout
- HTTPS enforcement via Cloudflare

Safe Commands:

- Confirmation dialogs untuk critical actions
- Command rate limiting (prevent spam)
- Audit log untuk all commands sent

J. Integration dengan Arduino Code

Serial Protocol Example:

Command Reception:

K. Performance Targets

Metric	**Target**	**Measurement**
Dashboard Load Time	< 2s	Time to interactive
Widget Update Latency	< 100ms	Data arrival to display
WebSocket Reconnect	< 1s	Connection recovery
Max Concurrent Users	10+	Same dashboard instance
Data Point Rate	100 Hz	Per widget maximum
Dashboard Complexity	50+ widgets	Without performance degradation

Implementation Priority:

- Core widget library (buttons, sliders, charts)
- Drag-and-drop editor
- WebSocket communication
- Data binding system
- Template library (3-5 templates)
- Advanced features (conditional formatting, alerts)
- Mobile optimization

- Multi-board support

3.4.4 Lightweight Operating System

Base System:

- Distribution: Debian minimal atau Yocto Project
- Kernel: Linux 6.x dengan custom configuration
- Init system: systemd
- Package manager: apt (Debian) atau opkg (Yocto)

Optimization:

- Minimal packages (no GUI desktop environment)
- Services: hanya Arduino-related dan networking
- Boot time target: < 30 detik
- RAM usage idle: < 256MB
- Storage footprint: < 2GB after installation

Pre-installed Components:

- Arduino CLI
- Cloudflared
- Node.js runtime
- Go runtime (untuk web server)
- Python 3 (untuk utilities)
- Git, curl, wget

3.5 Metode Pengembangan

3.5.1 Agile Development dengan Scrum

Sprint Duration: 2 minggu per sprint

Sprint Breakdown:

Sprint 1-2: Foundation

- Setup Yocto/Debian base image
- Configure automated installer script

- Network stack optimization
- First boot wizard implementation

Sprint 3-4: Arduino IDE Core

- Web IDE frontend development
- Code editor integration (Monaco)
- Arduino CLI backend integration
- Compilation dan upload mechanism

Sprint 5-6: AI Integration

- Local LLM setup (Llama 3.2)
- Fine-tuning dengan Arduino dataset
- Inference optimization
- IDE integration

Sprint 7-8: Cloudflare Tunnel

- Cloudflared automation script
- DNS configuration automation
- Security policies setup
- Documentation

Sprint 9-10: Polish & Testing

- UI/UX improvements
- Performance optimization
- Comprehensive testing
- Documentation completion

3.5.2 Development Tools

Version Control: Git + GitHub

CI/CD: GitHub Actions

Testing: Jest (frontend), Go test (backend)

Build System: Yocto/BitBake atau custom shell scripts

Monitoring: Prometheus + Grafana (untuk performance testing)

3.6 Metode Pengujian dan Evaluasi

3.6.1 Functional Testing

Unit Testing:

- Code coverage target: > 80%
- Testing framework: Jest, Go test
- Automated testing dalam CI/CD pipeline

Integration Testing:

- Test komunikasi antar komponen
- API endpoint testing
- WebSocket connection testing
- Arduino board communication testing

System Testing:

- End-to-end user scenarios
- Multiple board concurrent testing
- Stress testing untuk concurrent users
- Network failure recovery testing

3.6.2 Performance Testing

Metrics yang Diukur:

Metric	**Target**	**Measurement Method**
Boot Time	< 30s	Timer dari power-on hingga web accessible
IDE Load Time	< 3s	Browser performance API
Compilation Time	Arduino Uno sketch	Arduino CLI benchmark
AI Response Time	< 2s	Time dari request hingga suggestion
RAM Usage (Idle)	< 256MB	free -m command

RAM Usage (Active)	< 512MB	Monitoring selama development
Storage Footprint	< 2GB	du -sh /
Network Latency	Via tunnel	Ping test + upload/download speed

Benchmarking:

- Hardware: Raspberry Pi 4 (2GB RAM), x86-64 (4GB RAM)
- Comparative baseline: Ubuntu Desktop + Arduino IDE desktop
- Tools: sysbench, stress-ng, iperf3

3.6.3 Usability Testing

System Usability Scale (SUS):

- Questionnaire: 10 pertanyaan standar SUS
- Scoring: 0-100 scale
- Target: SUS score > 75 (good usability)

Partisipan:

- Total: 20-30 pengguna
- Kategori:
- Pemula (10): belum pernah programming Arduino
- Intermediate (10-15): sudah pernah buat 2-5 project
- Advanced (5): experienced developer

Tasks for Testing:

- Install OS dari USB
- Setup koneksi pertama kali
- Write, compile, upload sketch "Blink"
- Use AI untuk generate sensor reading code
- Configure custom dashboard
- Setup remote access via Cloudflare Tunnel

Data yang Dikumpulkan:

- Task completion time
- Error rate
- Number of assistance requests
- User satisfaction rating (1-5 scale)
- Qualitative feedback

3.6.4 Comparative Study

Perbandingan dengan Setup Tradisional:

Scenario 1: Setup Time

- Traditional: Install OS → Install Arduino IDE → Install drivers → Configure network
- Arduino OS: Boot USB → First-boot wizard

Scenario 2: Development Time

- Traditional: Manual coding → Manual library search → Manual troubleshooting
- Arduino OS: AI-assisted coding → Integrated library manager → AI debugging

Metrics Comparison:

Aspect	**Traditional**	**Arduino OS**	**Improvement**
Setup Time	2-4 hours	< 15 min	90%+ faster
First Project Time	1-2 hours	15-30 min	70%+ faster
Remote Access Setup	30-60 min	2 min	95%+ faster
Learning Curve	Steep	Gentle	Subjective survey
Cost	$0-20 (ngrok)	$0	100% saving

3.7 Instrumen Penelitian

3.7.1 Kuesioner Usability (SUS)

Menggunakan standar System Usability Scale dengan 10 pertanyaan:

- Saya akan sering menggunakan sistem ini
- Sistem ini terlalu kompleks
- Sistem ini mudah digunakan

... (dst sesuai standar SUS)

3.7.2 Performance Monitoring Scripts

3.7.3 User Satisfaction Survey

Skala Likert 1-5 untuk:

- Kemudahan instalasi
- Kemudahan penggunaan IDE
- Kualitas AI suggestions
- Kecepatan sistem
- Keandalan remote access
- Kepuasan keseluruhan

3.8 Analisis Data

3.8.1 Quantitative Analysis

Performance Metrics:

- Deskriptif statistik (mean, median, std deviation)
- Comparative analysis (t-test untuk perbandingan)
- Correlation analysis (relationship antar metrics)

SUS Score:

- Calculation: SUS score formula
- Interpretation: SUS score percentile ranking
- Comparison: Industry standard benchmark

3.8.2 Qualitative Analysis

Feedback Analysis:

- Thematic coding dari open-ended responses

- Issue categorization (bugs, feature requests, usability)
- Sentiment analysis

3.9 Jadwal Penelitian

Bulan	**Kegiatan**
1-2	Literature review, requirements gathering, system design
3-4	Sprint 1-4: OS foundation dan IDE core
5-6	Sprint 5-8: AI dan Cloudflare integration
7	Sprint 9-10: Testing dan optimization
8	Usability testing dan evaluasi
9	Data analysis dan documentation
10	Thesis writing dan revision

3.10 Deliverables

- Arduino OS ISO Image - Bootable sistem operasi lengkap
- Source Code Repository - GitHub dengan dokumentasi lengkap
- Technical Documentation - API docs, architecture diagrams
- User Manual - Installation guide, tutorials, troubleshooting
- Research Report - Laporan PKM lengkap dengan hasil evaluasi
- Demo Video - Screencast penggunaan sistem
- Sample Projects - 5+ contoh project Arduino dengan tutorial

BAB IV

BIAYA DAN JADWAL KEGIATAN

4.1 Anggaran Biaya

4.1.1 Peralatan Penunjang

A. Hardware untuk Development dan Testing

1. Perangkat Target Testing:

- Raspberry Pi 4 Model B (2GB RAM) - 2 unit
- Untuk testing OS pada low-end hardware
- Include: microSD card 32GB, power supply, case
- Raspberry Pi 4 Model B (4GB RAM) - 1 unit
- Untuk testing OS pada mid-range hardware
- Include: microSD card 64GB, power supply, case
- Mini PC x86-64 (4GB RAM, 64GB eMMC) - 1 unit
- Untuk testing compatibility pada arsitektur x86
- USB Flash Drive 32GB - 5 unit
- Untuk bootable OS installer testing

2. Arduino Development Boards untuk Testing:

- Arduino Uno R3 - 3 unit
- Board paling populer untuk compatibility testing
- Arduino Mega 2560 - 2 unit
- Testing untuk board dengan lebih banyak pins
- Arduino Nano - 2 unit
- Testing untuk compact projects
- ESP32 DevKit - 2 unit
- Testing untuk WiFi/Bluetooth capabilities
- ESP8266 NodeMCU - 2 unit

- Testing untuk IoT applications

3. Komponen Elektronik untuk Testing:

- Sensor Kit (DHT22, BMP280, ultrasonic, PIR, LDR) - 2 set
- Testing sensor integration dan AI code generation
- LED Kit (berbagai warna dan ukuran) - 1 set
- Basic output testing
- Resistor Kit (berbagai nilai) - 1 set
- Breadboard 830 points - 5 unit
- Jumper wire set (male-male, male-female, female-female)
- Servo motor SG90 - 5 unit
- Testing motor control
- Relay module 4 channel - 2 unit
- Testing high-power device control
- LCD Display 16x2 I2C - 3 unit
- Testing display output

4. ISP Programmer dan Tools:

- USBasp AVR Programmer - 2 unit
- Testing bootloader flashing
- USB to TTL Serial Adapter - 3 unit
- Testing serial communication debugging
- Logic Analyzer 8 channel - 1 unit
- Debugging communication protocols

5. Network Equipment:

- USB WiFi Adapter (untuk device tanpa built-in WiFi) - 2 unit
- USB Ethernet Adapter - 2 unit
- Router untuk testing environment - 1 unit

6. Storage dan Backup:

- External SSD 500GB - 1 unit
- Backup development files dan dataset
- microSD card reader USB 3.0 - 2 unit

B. Hardware untuk AI Training dan Development

1. GPU untuk AI Model Training:

- GPU dengan CUDA support (minimum 8GB VRAM)
- Untuk training dan fine-tuning model AI Arduino
- Alternatif: Cloud GPU credits (GCP/AWS)

2. Development Workstation Upgrade:

- RAM upgrade 32GB (untuk AI development)
- Menjalankan local LLM dan development environment
- SSD NVMe 1TB
- Fast storage untuk dataset dan model checkpoints

3. Edge AI Testing Device:

- Google Coral USB Accelerator - 1 unit
- Testing AI inference optimization pada edge device
- Intel Neural Compute Stick - 1 unit
- Alternative edge AI testing

4.1.2 Bahan Habis Pakai

A. Development Resources

1. Cloud Services Credits:

- GitHub Pro subscription (1 tahun)
- Private repositories, GitHub Actions minutes
- Cloud GPU credits untuk AI training
- Google Colab Pro atau AWS EC2 (p3.2xlarge)
- Estimasi: 100-200 jam training time
- Cloud Storage

- Model checkpoints, datasets backup
- Estimasi: 100GB-500GB

2. Domain dan Hosting:

- Domain name untuk testing Cloudflare Tunnel (1 tahun)
- Contoh: arduino-dev.com
- SSL Certificate (opsional, Cloudflare provides free)

3. Dataset dan Training Data:

- Arduino code dataset scraping (GitHub API credits)
- Data labeling tools subscription (jika perlu)
- Dataset storage dan preprocessing

4. Software Development Tools:

- JetBrains All Products Pack (student license - FREE)
- WebStorm, GoLand, PyCharm
- Figma Professional (untuk UI/UX design)
- Atlassian Suite (Jira, Confluence) - Free for small teams

B. AI Model Training Resources

1. Pre-trained Model Licenses:

- Llama 3.2 (Open source - FREE, Meta License)
- Training framework (PyTorch/TensorFlow - FREE)
- Hugging Face Pro subscription (optional)
- Untuk hosting model dan dataset

2. Dataset Resources:

- Arduino GitHub Corpus (10M+ lines)
- Scraping dan preprocessing scripts
- Annotated dataset untuk code quality
- Synthetic data generation scripts

3. Training Infrastructure:

- Cloud compute credits estimasi:
- Data preprocessing: 20-30 jam GPU
- Initial training: 50-80 jam GPU
- Fine-tuning iterations: 30-50 jam GPU
- Evaluation dan testing: 10-20 jam GPU
- Total: ~150-200 jam GPU time

4. Model Optimization Tools:

- Quantization tools (llama.cpp - FREE)
- ONNX Runtime (FREE)
- TensorRT (FREE untuk development)

C. Testing dan Quality Assurance

1. Testing Tools:

- BrowserStack subscription (cross-browser testing)
- Testing web IDE pada berbagai browser
- LoadForge credits (load testing)
- Testing concurrent users pada web IDE
- Sentry.io (error tracking) - Free tier

2. Monitoring Tools:

- Prometheus + Grafana (FREE, self-hosted)
- New Relic (free tier untuk development)

3. Documentation Tools:

- GitBook atau ReadTheDocs (FREE untuk open source)
- Draw.io / Diagrams.net (FREE)
- Mermaid.js (FREE)

4.1.3 Peralatan Pendukung Lainnya

1. Kabel dan Konektivitas:

- USB Cable Type-A to Type-B (untuk Arduino) - 10 unit

- USB Cable micro USB - 5 unit
- USB Cable Type-C - 3 unit
- HDMI cable - 2 unit
- Network cable Cat6 - 5 meter

2. Power Supply:

- USB Power Bank 20000mAh - 2 unit
- Untuk testing portability
- Multi-port USB charger - 2 unit
- Power strip dengan surge protection - 2 unit

3. Workspace Equipment:

- Soldering iron kit dengan accessories
- Basic electronics repairs dan custom builds
- Multimeter digital - 1 unit
- Anti-static mat dan wrist strap
- Storage organizer untuk komponen
- Toolset (screwdrivers, pliers, wire stripper)

4. Documentation Equipment:

- Webcam 1080p - 1 unit
- Untuk demo video dan documentation
- Microphone USB - 1 unit
- Untuk tutorial videos
- Screen recording software (OBS Studio - FREE)

4.1.4 Administrasi dan Publikasi

1. Dokumentasi:

- Technical writing tools (Grammarly Premium - opsional)
- LaTeX editor (Overleaf Premium - opsional)
- Diagram tools (Lucidchart - educational license)

2. Publikasi dan Presentasi:

- Conference registration fee (jika ada publikasi ilmiah)
- Poster printing untuk showcase
- Presentation materials

3. Backup dan Archival:

- Cloud backup subscription (Google Drive/Dropbox - 1TB)
- External HDD 2TB untuk long-term archival

4.2 Rangkuman Kebutuhan Biaya

Kategori Utama:

No	Kategori	Keterangan
1	Hardware Testing	Arduino boards (berbagai jenis), sensors, modules, development boards (Raspberry Pi, Mini PC)
2	AI Training Infrastructure	GPU/Cloud compute credits (~150-200 jam), storage untuk dataset dan model
3	Development Tools	IDE licenses (mostly FREE), cloud services, domain, monitoring tools
4	Dataset dan Training Data	GitHub corpus scraping, preprocessing, annotation tools
5	Testing Resources	Cross-browser testing, load testing, quality assurance tools
6	Komponen Elektronik	Sensors, actuators, displays, breadboards, cables
7	Backup dan Storage	External storage, cloud backup subscriptions
8	Documentation	Video equipment, presentation materials

Optimasi Biaya:

Menggunakan Free/Open Source sebanyak mungkin:

- Llama 3.2 - FREE (Meta License)
- PyTorch/TensorFlow - FREE
- VS Code, JetBrains (student) - FREE
- GitHub (student/education) - FREE
- Cloudflare Tunnel - FREE (unlimited)
- Linux OS - FREE
- Monitoring tools (Prometheus, Grafana) - FREE
- Documentation tools - FREE alternatives available

Biaya Utama:

- Hardware Arduino dan sensors (one-time)
- Raspberry Pi untuk testing (one-time)
- Cloud GPU credits untuk AI training (pay-as-you-go)
- Storage dan backup

4.3 Jadwal Kegiatan

Timeline 10 Bulan

Bulan	Kegiatan Utama	Deliverables	Resource Needs
1	Persiapan dan Design		
- Literature review lanjutan	Research report	-	
- Finalisasi requirements	Requirements doc	-	
- System architecture design	Architecture diagrams	Design tools	
- Procurement hardware testing	-	Arduino boards, Raspberry Pi	
2	Infrastructure Setup		
- Setup development environment	Dev env documentation	Development workstation	

- Prepare AI training pipeline	Training scripts	GPU setup/cloud credits	
- Dataset collection dan preprocessing	Cleaned dataset	Storage, compute	
- Base OS image preparation	Minimal bootable image	Yocto/Debian setup	
3	Sprint 1-2: Foundation		
- OS foundation development	Working base OS	Testing devices	
- Network stack optimization	Network benchmarks	Network equipment	
- Automated installer script	Installer prototype	USB drives for testing	
- First boot wizard	Wizard implementation	-	
4	Sprint 3-4: Arduino IDE Core		
- Web IDE frontend development	IDE interface	-	
- Monaco editor integration	Code editor	-	
- Arduino CLI backend	Compilation service	Arduino boards for testing	
- Upload mechanism	Working upload	Various Arduino boards	
5	Sprint 5-6: AI Training		
- Model fine-tuning (Llama 3.2)	Trained model v1	GPU credits (50-80 jam)	
- Inference optimization	Optimized model	Testing on Raspberry Pi	
- IDE integration	AI assistant in IDE	-	
- Testing dan iteration	Improved model v2	GPU credits (20-30 jam)	
6	Sprint 7-8: Integration		

- Cloudflare Tunnel automation	Working tunnel setup	Domain name	
- Dashboard development	Custom panel	-	
- Serial communication	Real-time monitoring	Sensors, displays	
- Complete system integration	Alpha version	All hardware	
7	Sprint 9-10: Polish & Testing		
- UI/UX improvements	Polished interface	-	
- Performance optimization	Benchmarks	Various devices	
- Bug fixing	Stable beta	-	
- Documentation (technical)	API docs, arch docs	-	
8	Testing dan Evaluasi		
- Usability testing (20-30 users)	SUS scores, feedback	-	
- Performance benchmarking	Performance report	All testing hardware	
- Comparative study	Comparison analysis	-	
- Security audit	Security report	-	
9	Analysis dan Documentation		
- Data analysis (quantitative)	Statistical analysis	-	
- Qualitative analysis	Themes, insights	-	
- User manual creation	User guide	Documentation tools	
- Video tutorials	Demo videos	Webcam, mic	
10	Finalisasi		

- Thesis/report writing	Final report	-	
- Presentation preparation	Presentation materials	-	
- Final demo preparation	Demo setup	-	
- Publication submission (optional)	Paper draft	Conference fee (if any)	

DAFTAR PUSTAKA

2Cloud.io. (2024). Secure Self-Hosted Applications with Cloudflare Tunnel. 2Cloud Technical Blog. https://2cloud.io/

ACM Digital Library. (2023). Block-Based Programming Limitations in Computer Science Education. ACM Digital Library. https://dl.acm.org/

ACM SIGCHI. (2023). Context-Appropriate User Interfaces for IoT Applications. ACM SIGCHI Conference Proceedings. https://dl.acm.org/

App Store Analytics. (2024). Blynk IoT Platform User Satisfaction Analysis. App Store Connect Analytics. https://apps.apple.com/

Arduino. (2024). Arduino Cloud Platform and Web Editor. Arduino Official Documentation. https://cloud.arduino.cc/

Arduino Forum. (2015). Web-based Arduino IDE Community Discussion. Arduino Community Forum. https://forum.arduino.cc/

Arduino Forum. (2021). Using Ngrok for Arduino Remote Access. Arduino Community Forum. https://forum.arduino.cc/

Arduino Forum. (2022). Remote Programming Without Local Connection. Arduino Community Forum. https://forum.arduino.cc/

Arduino Forum. (2023). Arduino Cloud Agent Connection Issues. Arduino Community Forum. https://forum.arduino.cc/

Arduino Forum. (2024). Automatic Pin Detection for Dashboard Builders. Arduino Community Forum. https://forum.arduino.cc/

Blynk. (2024). Blynk IoT Platform - Dashboard Builder Documentation. Blynk Official Documentation. https://docs.blynk.io/

Blynk Documentation. (2024). Blynk Privacy and Data Routing Architecture. Blynk Technical Documentation. https://docs.blynk.io/

CHI Conference. (2023). Separation of Developer and End-User Interfaces in IoT. ACM CHI Conference Proceedings. https://dl.acm.org/

Cloudflare. (2024). Cloudflare Tunnel Documentation. Cloudflare Zero Trust. https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/

Code-B.dev. (2024). Top Cloud IDEs for Developers in 2024. Code-B Developer Blog. https://code-b.dev/

CodingFleet. (2024). Arduino Code Generator - AI-Powered Tool. CodingFleet. https://codingfleet.com/arduino-code-generator/

DevOpsTales. (2024). Implementing Cloudflare Tunnel for Embedded Systems. DevOps Security. https://devopstales.com/

Electronics-Lab. (2023). Arduino Cloud Security Deep Dive. Electronics-Lab Technical Articles. https://www.electronics-lab.com/

GeeksforGeeks. (2024). IoT Operating Systems Comprehensive Guide. GeeksforGeeks. https://www.geeksforgeeks.org/iot-operating-systems/

Goldman, M., Miller, R. C., & Cai, C. J. (2011). Collabode: Collaborative coding in the browser. In Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE '11), 65-68. https://dl.acm.org/doi/10.1145/1984642.1984658

Google Blockly. (2024). Blockly Visual Programming Framework Documentation. Google Blockly. https://developers.google.com/blockly

Google Research. (2024). ML-Enhanced Code Completion Impact Study. Google Research Blog. https://research.google/

Grafana. (2024). Grafana Dashboard Publishing and Sharing. Grafana Labs Documentation. https://grafana.com/docs/

IEEE. (2019). Capabilities and Limitations of Web-Based IDEs. IEEE Conference Proceedings. https://ieeexplore.ieee.org/

IEEE. (2020). Online Compiler as Cloud Service: Comprehensive Review. IEEE Xplore Digital Library. https://ieeexplore.ieee.org/

IEEE. (2021). Embedded Linux in Resource-Constrained Environments. IEEE Transactions. https://ieeexplore.ieee.org/

IEEE. (2022). Transformer Models for Code Completion. IEEE Digital Library. https://ieeexplore.ieee.org/

IEEE. (2023). Visual Programming Environments for IoT Development. IEEE Conference Proceedings. https://ieeexplore.ieee.org/

IEEE. (2024). Detecting LLM-Generated Firmware in Education. IEEE Conference Proceedings. https://ieeexplore.ieee.org/

IEEE Education. (2023). Student Engagement in Block-Based Programming Education. IEEE Education Conference. https://ieeexplore.ieee.org/

IEEE Software. (2023). Dashboard Publishing Paradigms for IoT Applications. IEEE Software Journal. https://ieeexplore.ieee.org/

IJRPR. (2022). Integration Strategies for Arduino Cloud Platforms. International Journal of Research Publications and Reviews. https://ijrpr.com/

ImplRust. (2024). ESP32 Remote Access via Cloudflare Tunnel Architecture. ImplRust Technical Tutorial. https://implrust.com/

IoT Analytics. (2023). ThingsBoard Platform User Experience Analysis. IoT Analytics Research. https://iot-analytics.com/

IoT Design Patterns. (2023). Automatic Data Binding in IoT Dashboards. IoT Design Patterns Repository. https://iotdesignpatterns.com/

IRJET. (2023). AI-Powered Code Completion in IDEs: Survey and Analysis. IRJET Research Journal. https://irjet.net/

ISTE Research. (2023). Visual Programming Tools in K-12 Computer Science Education. ISTE Research Publications. https://iste.org/

It's FOSS. (2024). Cloudflare Tunnel Tutorial for Linux Systems. It's FOSS Community. https://itsfoss.com/

Kunzleigh. (2023). Complete Guide to Arduino Remote Access Using Ngrok. Kunzleigh Tutorials. https://kunzleigh.com/

Maker.pro. (2023). Arduino Web Editor Complete Guide and Limitations. Maker Pro Technical Reviews. https://maker.pro/

MDPI. (2021). Usability Design Requirements for Microcontroller-Based Devices. MDPI Electronics Journal. https://mdpi.com/

MDPI. (2022). Visual Syntax in Programming Education: Cognitive Load Reduction. MDPI Education Sciences. https://mdpi.com/

MDPI. (2023). Remote Laboratory Implementation for Microcontroller Education. MDPI Education Sciences. https://mdpi.com/

MDPI. (2024). Impact of AI-Powered Tools on Programming Education Outcomes. MDPI Education Journal. https://mdpi.com/

MDPI Education. (2024). Pedagogical Benefits of Visual Programming in STEM. MDPI Education Sciences. https://mdpi.com/

Medium. (2023). Node-RED Deployment Challenges in Embedded Systems. Medium Technical Articles. https://medium.com/

Medium. (2024). Transformer Architecture for Code Generation. Medium Technical Articles. https://medium.com/

MQTT.org. (2024). MQTT Protocol Specification and Use Cases. MQTT.org Official Documentation. https://mqtt.org/

Ngrok. (2024). Ngrok Documentation - Complete Guide for IoT Devices. Ngrok Official Documentation. https://ngrok.com/docs

Node-RED Dashboard. (2024). Dashboard Publishing and Access Control. Node-RED Dashboard Documentation. https://flows.nodered.org/node/node-red-dashboard

Node-RED Foundation. (2024). Node-RED Flow-Based Programming for IoT. Node-RED Official Documentation. https://nodered.org/

PieterDev. (2024). Setting Up Cloudflare Tunnel Behind CG-NAT. PieterDev Technical Documentation. https://pieterdev.com/

Pitsco. (2023). Arduino Web Editor vs Desktop IDE Detailed Comparison. Pitsco Education Resources. https://www.pitsco.com/

PlatformIO. (2023). ESP Tunnel Library for Secure Remote Access. PlatformIO Library Registry. https://platformio.org/lib/show/11876/ESP-Tunnel

PleaseDontCode. (2024). PCGen - AI Arduino Code Generator. Please Don't Code Platform. https://pleasedontcode.com/

Reddit. (2022). Remote Microcontroller Programming from Browser Discussion. Reddit r/embedded. https://www.reddit.com/r/embedded/

Reddit. (2023). Arduino and Ngrok Integration Best Practices. Reddit r/arduino. https://www.reddit.com/r/arduino/

Reddit r/selfhosted. (2024). Cloudflare Tunnel vs Ngrok Cost Comparison. Reddit r/selfhosted. https://www.reddit.com/r/selfhosted/

ResearchGate. (2018). Web-Based Supervisory Control Systems for Arduino Platforms. ResearchGate Publications. https://www.researchgate.net/

ResearchGate. (2020). Qualitative Evaluation of Remote Laboratory User Experience. ResearchGate Publications. https://www.researchgate.net/

ResearchGate. (2022). Embedded Linux Optimization for ARM-Based IoT Devices. ResearchGate Publications. https://www.researchgate.net/

ResearchGate. (2023). Automated Provisioning Framework for Edge-Based IoT. ResearchGate Publications. https://www.researchgate.net/

ResearchGate. (2024). AI-Assisted Programming: Challenges and Future Directions. ResearchGate Publications. https://www.researchgate.net/

Semantic Scholar. (2021). Secure Over-the-Air Updates for Low-Power IoT Devices. Semantic Scholar Database. https://www.semanticscholar.org/

Semantic Scholar. (2022). Automated IoT Device Provisioning Using Ontologies. Semantic Scholar Database. https://www.semanticscholar.org/

Swimm.io. (2024). GitHub Copilot vs TabNine: Technical Comparison. Swimm Technical Blog. https://swimm.io/

ThingsBoard. (2024). ThingsBoard Rule Engine and Dashboard Capabilities. ThingsBoard Official Documentation. https://thingsboard.io/

TuxCare. (2024). Linux Distributions for IoT and Embedded Systems. TuxCare Documentation. https://tuxcare.com/

UPS Ecuador. (2022). Multi-Platform Remote IoT Education Laboratory Implementation. Universidad Politécnica Salesiana. https://ups.edu.ec/

UX Collective. (2024). Widget-Based Dashboard Assembly: Time Efficiency Study. UX Collective. https://uxdesign.cc/

UX Research. (2023). Optimal Widget Categorization for IoT Dashboards. UX Research Publications. https://uxresearch.com/

W3C. (2023). WebSocket Protocol for Real-Time Communication. W3C WebSocket Specifications. https://www.w3.org/TR/websockets/

W3C WebSocket Spec. (2023). WebSocket Performance Characteristics for IoT. W3C Technical Specifications. https://www.w3.org/TR/websockets/