Arduino Circuit Simulator

Web-Based Electronics Simulation Platform with Auto-Wiring and Code Generation

Project Title:	Arduino Circuit Simulator
Organization:	FOSSEE OSHW - IIT Bombay
Internship Program:	Semester Long Internship - 2025
Candidate Name:	MD KAIF NEZAMI
Department:	Electronics and Communication Engineering
Semester:	3rd Semester
Institution:	BIT Sindri
Email:	mdkaif.ece24@bitsindri.ac.in
Submission Date:	February 2, 2025

Executive Summary

This report presents the development of a web-based Arduino simulator as part of the FOSSEE OSHW internship screening task. The primary objective was to create an interactive platform enabling users to build Arduino circuits visually with automatic wiring and real-time code generation capabilities.

The project successfully implements all three screening task requirements: a drag-and-drop component interface, automatic pin assignment with conflict prevention, and functional logic-level simulation with code generation. The simulator supports LED and Push Button components with the Arduino Uno board, automatically wiring them to default pins (LED to Pin 10, Button to Pin 2) while allowing user-configurable pin reassignment.

Built entirely with standard web technologies (HTML5, CSS3, JavaScript), the application requires no external frameworks or libraries, ensuring lightweight deployment and broad browser compatibility. The modular architecture separates UI logic from simulation logic, facilitating future maintenance and extensibility.

Project Resources

Resource	URL
Live Demo	https://698082fc7a017caae8fb4240--lustrous-faun-c12ff9.netlify.app/
GitHub Repository	https://github.com/nezamimdkaif/arduino-simulator
Demo Video	https://www.youtube.com/watch?v=d4xub4YjHz8

1. Introduction

1.1 Background and Motivation

Electronics and embedded systems education requires substantial hands-on experimentation for effective learning. However, access to physical Arduino boards and electronic components is not universally available due to cost constraints, laboratory availability, and geographical limitations. Web-based simulation platforms address these barriers by providing accessible, risk-free environments for circuit experimentation and code development.

Existing platforms such as Wokwi have demonstrated the educational efficacy of browser-based Arduino simulators. The FOSSEE initiative at IIT Bombay develops open-source tools to democratize engineering education in India. This screening task provided an opportunity to contribute to this mission by implementing a functional Arduino simulator prototype.

1.2 Project Objectives

The screening task specified three progressive implementation requirements:

Task	Objective	Description
Task 1	Web-Based Interface	Develop a drag-and-drop interface for Arduino Uno, LED, and Push Button components
Task 2	Auto-Wiring & Pin Config	Implement automatic component-to-pin connections with default mappings (LED→Pin 10, Button→Pin 2)
Task 3	Code Gen & Simulation	Generate syntactically correct Arduino code with automatic updates on configuration changes

1.3 Scope and Constraints

To maintain project focus and demonstrate core competencies, the following scope limitations were established:

• Single board support: Arduino Uno only

• Component library: LED and Push Button exclusively

• Pin availability: Digital pins 2-13

• Simulation type: Logic-level (HIGH/LOW) without analog capabilities

• Default configuration: LED on Pin 10, Push Button on Pin 2

2. System Architecture and Implementation

2.1 Technology Stack

The application employs standard web technologies to ensure maximum compatibility and minimal deployment complexity:

Technology	Purpose	Rationale
HTML5	Document structure	Universal browser support, native drag-and-drop API
CSS3	Visual styling	Modern styling capabilities without preprocessors
JavaScript ES6	Application logic	Native browser execution, no compilation required
Canvas API	Component rendering	Hardware-accelerated graphics

2.2 Project Structure

The codebase follows a modular organization pattern separating concerns for maintainability:

```
arduino-simulator/
|-- index.html (Application entry point)
|-- assets/
| |-- css/
| | +-- style.css (Styling and layout)
| |-- icons/ (Component visual assets)
| +-- js/
| |-- main.js (UI logic, drag-drop, code generation)
| +-- simulation.js (Simulation engine and state management)
+-- README.md (Technical documentation)
```

This separation enables independent development and testing of UI components (main.js) and simulation logic (simulation.js), reducing coupling and facilitating future enhancements.

3. Detailed Implementation

3.1 Task 1: Web-Based Interface

The interface architecture implements a three-panel layout optimized for circuit building workflow:

• **Component Palette (Left Panel):** Displays draggable component cards for Arduino Uno, LED, and Push Button with visual icons.

• **Circuit Canvas (Center Panel):** Primary workspace for component placement and circuit assembly with drop zone visualization.

• **Code Panel (Right Panel):** Real-time Arduino code display with syntax highlighting and circuit status indicators.

The drag-and-drop functionality utilizes the HTML5 Drag and Drop API with event handlers for dragstart, dragover, and drop events. Cross-browser compatibility was achieved through proper event.preventDefault() usage and dataTransfer object manipulation.

Control buttons (Start Simulation, Stop Simulation, Generate Code, Auto Wire, Clear Canvas) provide intuitive simulation management. The Auto Wire button offers manual wiring visualization, though actual wiring occurs automatically during component placement.

3.2 Task 2: Automatic Wiring and Pin Configuration

The automatic wiring system implements intelligent default pin assignment with dynamic conflict prevention:

Component	Default Pin	Pin Type	Configuration
LED	Digital Pin 10	OUTPUT	PWM-capable for future analog support
Push Button	Digital Pin 2	INPUT_PULLUP	Active-LOW with internal pull-up

Pin reassignment is implemented through dropdown menus on each component. The conflict prevention algorithm dynamically filters available pins by maintaining a registry of occupied pins. When a component occupies a pin, that pin is excluded from dropdown options for other components, preventing electrical conflicts.

3.3 Task 3: Code Generation and Simulation

The code generator produces syntactically correct Arduino C++ code following standard conventions:

```
void setup() {
pinMode(10, OUTPUT); // LED configuration
pinMode(2, INPUT_PULLUP); // Button with pull-up
}
void loop() {
int buttonState = digitalRead(2);
if (buttonState == LOW) { // Button pressed (active-LOW)
digitalWrite(10, HIGH); // Turn LED ON
} else {
digitalWrite(10, LOW); // Turn LED OFF
}
}
```

The simulation engine implements a 50ms execution loop (20Hz update rate) using JavaScript setInterval. This loop reads button state and updates LED state accordingly, mirroring Arduino execution timing. The button implements active-LOW logic matching real Arduino behavior with INPUT_PULLUP configuration.

4. Testing and Validation

4.1 Functional Testing

Comprehensive testing was conducted across all functional requirements:

Test Case	Expected Behavior	Result
Component drag-drop	Smooth dragging with visual feedback	✓ Pass
Default pin assignment	LED→10, Button→2 automatically	✓ Pass
Pin conflict prevention	Occupied pins unavailable	✓ Pass
Pin reassignment	Code updates automatically	✓ Pass
Code generation	Valid Arduino syntax	✓ Pass
Button simulation	LED ON when pressed	✓ Pass
Button release	LED OFF when released	✓ Pass
Simulation controls	Start/Stop function correctly	✓ Pass

4.2 Browser Compatibility

• Google Chrome 90+: Full functionality with optimal performance
• Mozilla Firefox 88+: Complete feature support
• Microsoft Edge 90+: Chromium-based, identical behavior
• Apple Safari 14+: Functional with WebKit adjustments

4.3 Performance Metrics

Metric	Measurement	Assessment
Initial page load	< 200ms	Excellent
Component drag latency	< 16ms (60 FPS)	Smooth
Code generation	< 10ms	Instantaneous
Simulation rate	50ms (20Hz)	Adequate
Memory footprint	< 15MB	Lightweight

5. Results and Discussion

5.1 Achievement Summary

Task 1 - Web Interface: Implemented fully functional drag-and-drop interface with component palette, circuit canvas, and real-time code viewer.

Task 2 - Auto-Wiring: Developed automatic pin assignment system with default configuration and dropdown-based reassignment with conflict prevention.

Task 3 - Code Generation & Simulation: Created Arduino code generator producing syntactically correct sketches with functional button-to-LED control logic.

5.2 Technical Challenges and Solutions

Challenge	Solution Implemented
Pin conflict management	Synchronous dropdown filtering with immediate registry updates
State synchronization	Event-driven architecture with cascading updates
Active-LOW button behavior	Correct INPUT_PULLUP implementation
Cross-browser drag-drop	Standard HTML5 API with proper event handling

6. Future Development Roadmap

6.1 Component Library Expansion

• **Passive Components:** Resistors, capacitors, potentiometers

• **Sensors:** Temperature, ultrasonic, light, motion sensors

• **Displays:** LCD, 7-segment, OLED modules

• **Actuators:** Servo motors, DC motors, stepper motors

• **Communication:** Bluetooth, WiFi, NRF24L01 modules

6.2 Platform Features

• Serial monitor for debugging and data visualization
• Analog support (ADC input, PWM output)
• Multi-board support (Mega, Nano, ESP32)
• Circuit persistence using localStorage
• Breadboard view for realistic visualization
• Code export as .ino files

7. Conclusion

This project successfully demonstrates a functional Arduino simulation platform meeting all screening task requirements. The implementation showcases systematic requirement analysis, clean code architecture, thorough testing, and proper documentation practices.

The modular design separating UI logic from simulation engine facilitates addition of new components and features without major refactoring. This project demonstrates competencies in requirement analysis, maintainable code architecture, user experience design, comprehensive testing, and technical documentation.

The FOSSEE initiative's mission to democratize quality engineering education aligns with my values and career aspirations. If selected for this internship, I am committed to expanding this simulator into a comprehensive educational tool benefiting students across India and globally.

Thank you for considering this application.

8. References

[1] Arduino Official Documentation. Available: https://www.arduino.cc/reference/

[2] Wokwi - Online Arduino Simulator. Available: https://wokwi.com/

[3] FOSSEE Open Source Hardware Project. Available: https://oshw.fossee.in/

[4] Wokwi Elements - Web Components. Available: https://github.com/wokwi/wokwi-elements

[5] MDN Web Docs - HTML5 Drag and Drop API

[6] MDN Web Docs - Canvas API

[7] FOSSEE Semester Long Internship 2025 - Screening Task Document

[8] Project Repository. Available: https://github.com/nezamimdkaif/arduino-simulator

[9] Live Demo. Available: https://698082fc7a017caae8fb4240--lustrous-faun-c12ff9.netlify.app/

[10] Demo Video. Available: https://www.youtube.com/watch?v=d4xub4YjHz8