

Project Report

Arduino Simulator – OSHW Screening Tasks 1–3

Introduction

This project presents a **web-based Arduino Simulator** developed as part of the **Open Source Hardware (OSHW) screening tasks**.

The simulator provides an interactive interface for building simple Arduino circuits, configuring pins, generating Arduino code automatically, and performing **logic-level simulation**.

The implementation is divided into **three incremental tasks**, each extending the functionality of the previous one while maintaining strict constraints and clarity.

Objectives

The objectives of this project are:

- To design a clean and minimal **web-based Arduino simulation interface**
- To support **drag-and-drop circuit construction**
- To implement **automatic pin assignment and configuration**
- To generate **valid Arduino code automatically**
- To provide **logic-level simulation** that mirrors the generated code
- To ensure strict compliance with OSHW screening requirements

Technologies Used

- **HTML5** – Application structure
- **CSS3** – Layout and visual styling
- **JavaScript (Vanilla)** – Core logic and simulation engine
- **Browser DOM APIs** – Drag-and-drop and UI interactions

No external libraries or frameworks were used to keep the implementation lightweight and transparent.

Task 1: Web-Based Arduino Simulation Interface

4.1 Description

Task 1 focused on building the basic simulator interface.

4.2 Features Implemented

- Component palette with:
 - Arduino Uno
 - LED
 - Push Button
- Drag-and-drop placement on a canvas
- Single Arduino constraint
- Movable components

- Right-click component removal
- Read-only code view panel
- Start/Stop controls with visual feedback

4.3 Outcome

A functional, minimal interface capable of representing Arduino circuits visually.

Task 2: Pin Management and Auto-Wiring

5.1 Description

Task 2 extended the simulator to support **automatic wiring** and **user-configurable pin assignments**.

5.2 Constraints Enforced

- Exactly **one Arduino Uno**
- Exactly **one LED**
- Exactly **one Push Button**
- No additional components allowed

5.3 Auto-Wiring Rules

- LED defaults to **Digital Pin D10**
- Push Button defaults to **Digital Pin D2**
- Component addition is blocked if the default pin is unavailable

- No fallback pin assignment is allowed

5.4 Pin Configuration

- Pins selectable from **D2 to D13**
- Pin conflicts prevented via dropdown filtering
- Changes reflected instantly in generated code

5.5 Code Generation

Automatically generates valid Arduino code using:

- `pinMode`
- `digitalRead`
- `digitalWrite`
- One-to-one LED–Button mapping

6. Task 3: Logic-Level Simulation

6.1 Description

Task 3 introduced a **logic-level simulation engine** tightly coupled with the generated Arduino code.

6.2 Simulation Model

- Button uses `INPUT_PULLUP`
- Button pressed → LOW

- Button released → HIGH
- LED is **Active-High**
- LED ON when pin is HIGH
- LED OFF when pin is LOW

6.3 Simulator Engine

- Centralized simulation logic
- Start/Stop lifecycle control
- Real-time LED visual updates
- Immediate adaptation to pin reconfiguration

6.4 Code and Simulation Alignment

The simulation behavior exactly mirrors the generated Arduino code logic:

```
int buttonState = digitalRead(buttonPin);
digitalWrite(ledPin, buttonState == LOW ? HIGH : LOW);
```

Verification and Testing

Manual Tests Performed

- Component constraints validation
- Default pin assignment verification

- Pin conflict prevention
- Code regeneration on pin changes
- Button-to-LED simulation correctness
- Start/Stop behavior validation

All tests passed successfully.

Limitations

- Logic-level simulation only (no electrical modeling)
- Single LED and Button supported
- No breadboards, wires, or resistors
- No timing or PWM simulation

These limitations are intentional to align with screening requirements.

Conclusion

The Arduino Simulator successfully fulfills **Tasks 1, 2, and 3** of the OSHW screening process.

The project demonstrates strong fundamentals in **UI design, state management, code generation, and simulation consistency**, while maintaining clarity and correctness.

The simulator provides a solid foundation for future extensions while remaining strictly compliant with the scope of the screening tasks.