Summer Internship Report
on

Arduino Architecture, Compilation, and Hardware Connection Logic in OpenHW Studio

Submitted by

Poojitha S K

B.Tech (Computer Science and Business Systems)

Sri Eshwar College of Engineering,Coimbatore

Prof. Prabhu Ramachandran

Principal Investigator, Department of Aerospace Engineering, Indian Institute of Technology Bombay

July 2026

Acknowledgement

I would like to express my deepest gratitude to **Prof. Prabhu Ramachandran** for providing me the opportunity to be a part of the **FOSSEE** internship program and for supporting open-source engineering tool development at IIT Bombay. His guidance and vision have encouraged students and researchers to actively contribute to the open-source ecosystem.

My sincere appreciation extends to my mentor, **Rajesh Kushalkar**, for his continual support, technical guidance, and encouragement throughout the duration of this project. His insights and feedback played a key role in refining ideas, overcoming challenges, and ensuring timely completion of the tasks assigned to me.

I would also like to thank my internal mentors, **Pratik Bhosale (Project Research Associate), Pratik Nemane (Project Research Assistant)**, for their valuable guidance, coordination, and technical inputs during the internship. Their mentorship contributed significantly to the clarity, progress, and successful execution of the work.

I also owe a great debt of gratitude to the **OpenHW Studio** development team and the open-source community. Their expert advice and timely assistance not only helped me complete my tasks successfully but also sharpened both my technical and non-technical skills.

This internship has been an enriching learning experience, allowing me to work closely with on the **OpenHW Studio** , an open source web simulation platform. The knowledge acquired during this period will undoubtedly support my future academic and professional pursuits.

I would also like to thank the entire **FOSSEE** team for their coordination, assistance, and timely interactions at various stages of this work. Their collective efforts ensured smooth workflow, resource accessibility, and effective project execution.

Poojitha S K

Declaration

I hereby declare that this written submission represents my own ideas and the work I contributed to during my internship. Whenever the ideas or words of others have been included, I have adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this report.

I also declare that I have adhered to all principles of academic honesty and integrity, and I have not misrepresented, fabricated, or falsified any idea, data, fact, or source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and may also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been obtained when required.

Poojitha S K

Contents

Chapter 1

Introduction

OpenHW Studio is a browser-based, open-source hardware simulation and learning platform developed by the FOSSEE project at IIT Bombay, designed to make embedded systems education highly accessible without the need for physical hardware. It provides a comprehensive, plugin-free environment that offers real-time, instruction-level emulation for popular microcontrollers such as the Arduino Uno, Raspberry Pi Pico, ESP32, and STM32. The platform caters to all skill levels by supporting both block-based programming for beginners and full C++ code for advanced users, while also featuring built-in digital logic components (like flip-flops and buffer gates), smart debugging tools like logic analyzers and serial plotters, and a dedicated classroom mode that allows educators to push projects and grade submissions in real-time.

1.1 Project Overview

During my summer internship, I contributed to the full-stack development of the OpenHW Studio platform. I built secure backend routes and middleware for user authentication and developed the corresponding frontend interfaces. Additionally, I designed interactive simulation components for the platform's emulator, ensuring their accuracy by actively testing them against physical hardware via USB. Beyond core development, I configured gamified guided projects, assisted with Blockly-based coding features, and wrote comprehensive documentation to maintain the project's long-term accessibility

---

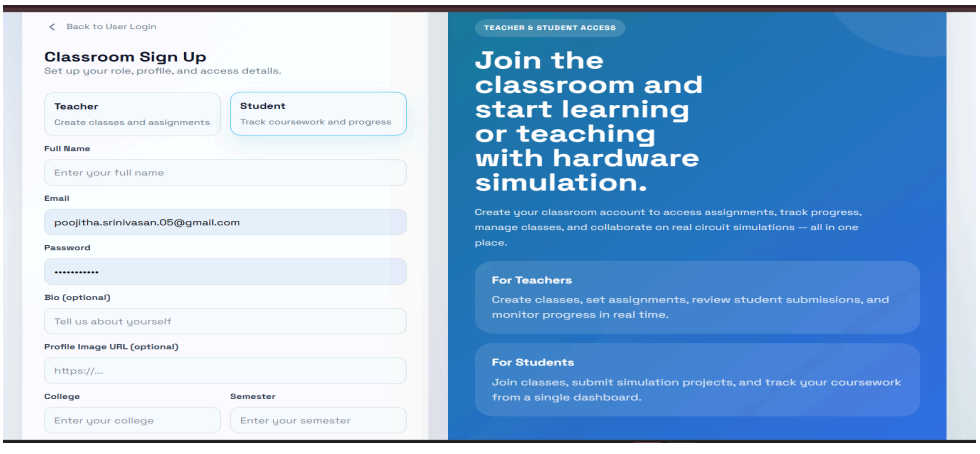
Chapter 2

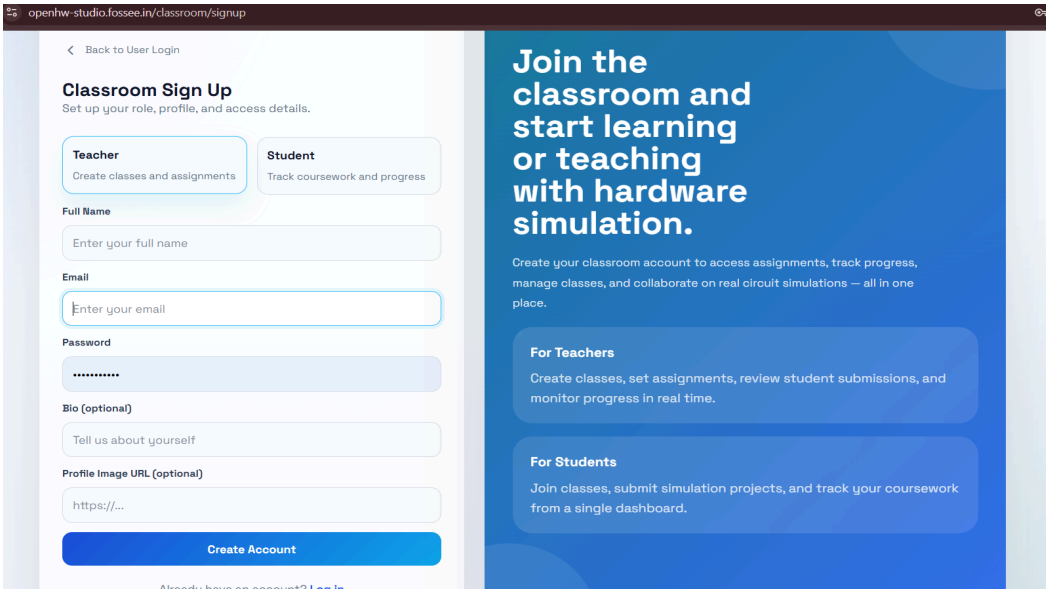
Designing Backend User Access and Management

2.1. signup_route

Handles new user registration and account creation through the following key processes:

- **Input Validation:** Ensures that the user-provided data (such as email, username, and password) meets specific formatting and complexity requirements to maintain data integrity and prevent injection attacks.
- **Duplicate Checking:** Queries the database to verify that the provided email or username does not already exist, preventing the creation of duplicate accounts.
- **Password Security:** Utilizes cryptographic hashing and salting techniques to securely encode the user's password before it is stored, ensuring that raw passwords are never exposed in the database.
- **Database Integration:** Securely commits the new user's profile information and hashed credentials into the database system.
- **Response Generation:** Returns a success status to the frontend upon completion, often accompanied by an initial authentication token so the user is immediately logged in after signing up.

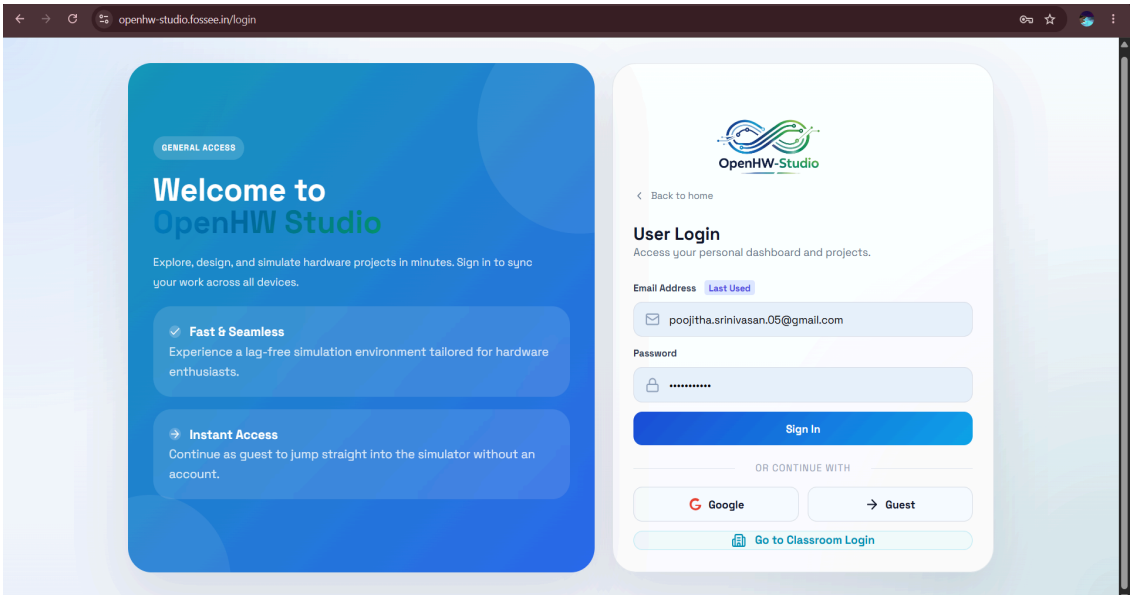




2.2. login_route

Manages user authentication and session creation through the following steps:

- **Credential Verification**: Receives the user's login attempt and queries the database to ensure that the provided email or username exists within the system.
- **Password Validation**: Securely compares the provided plaintext password against the cryptographically hashed password stored in the database to verify authenticity.
- **Token Generation**: Upon successful password verification, generates a secure, time-bound session token (such as a JSON Web Token - JWT) that contains essential user identity claims.
- **Session Management**: Returns the generated token to the frontend client, allowing it to be stored (e.g., in secure HTTP-only cookies or local storage) to maintain persistent, authorized access to the platform without needing to log in repeatedly.



2.3. forgot_password_route

Facilitates secure account recovery through the following mechanisms:

- **User Identification**: Receives a password reset request and verifies that the provided email address corresponds to an active account in the database.
- **Token Generation**: Generates a secure, cryptographically random, and time-sensitive recovery token (e.g., expiring in 15 minutes) to ensure the reset request cannot be exploited indefinitely.
- **Email Dispatch**: Integrates with a backend mailing service (like Nodemailer) to dispatch an automated email to the user, containing a secure reset link embedded with the generated recovery token.
- **Password Update**: Handles the final submission by validating the recovery token, ensuring it hasn't expired, and securely hashing and storing the user's newly chosen password in the database.

2.4 auth_middleware

Acts as a strict security filter for the backend application by implementing the following technical mechanisms:

- **Request Interception & Header Parsing:** Intercepts incoming HTTP requests directed at protected API endpoints and parses the Authorization header to extract the Bearer token payload.

- **Cryptographic Verification:** Utilizes a backend secret key to verify the JSON Web Token's (JWT) cryptographic signature (e.g., using the HS256 algorithm) and strictly evaluates the exp (expiration) claim to ensure the session is still valid.

- **Context Injection & Delegation:** Upon successful verification, decodes the JWT payload and injects essential claims (such as the User ID and roles) directly into the request object (e.g., req.user). It then safely delegates control to the next middleware or route controller in the stack.

- **Error Handling & Request Termination:** Aborts the request lifecycle immediately if token verification fails, returning a standardized 401 Unauthorized JSON response for missing/tampered tokens, or a 403 Forbidden response if the user lacks the necessary permissions for that route.

Chapter 3. Guided Projects Creation

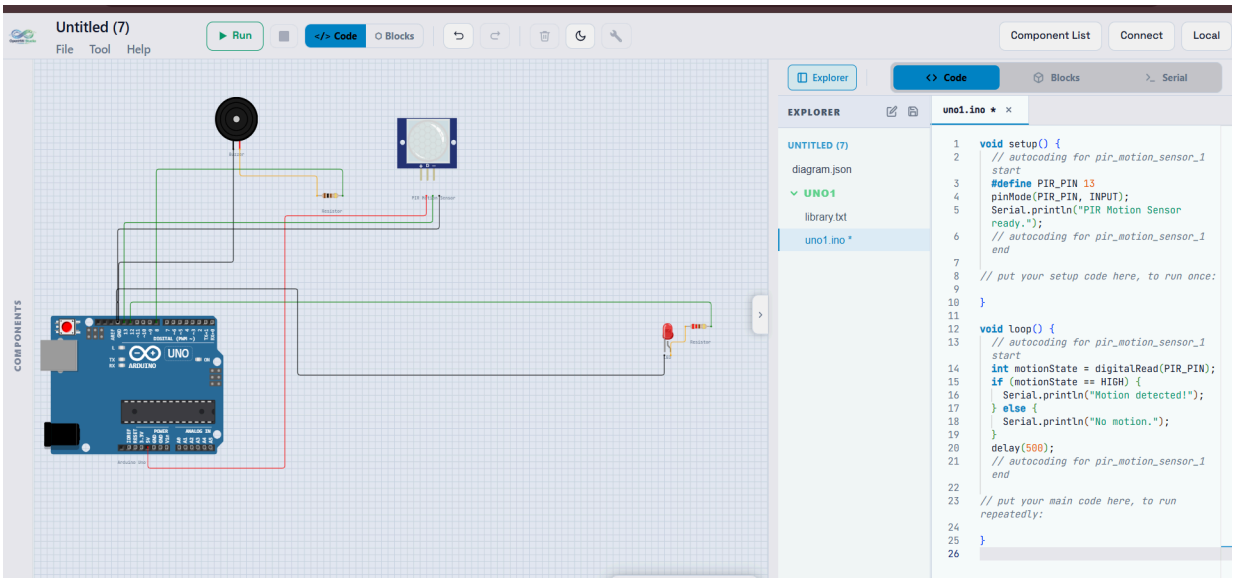
1. Visual Asset & Project Architecture This module involved architecting the structured learning modules and their associated UI assets for the platform's simulator. The following technical contributions were made:

- **UI Asset Generation & Binding** Designed and exported high-fidelity graphical assets (PNGs) representing real-world hardware components and schematic wiring. Bound these graphical assets to the frontend UI components to serve as dynamic, visual overlays during step-by-step project execution.

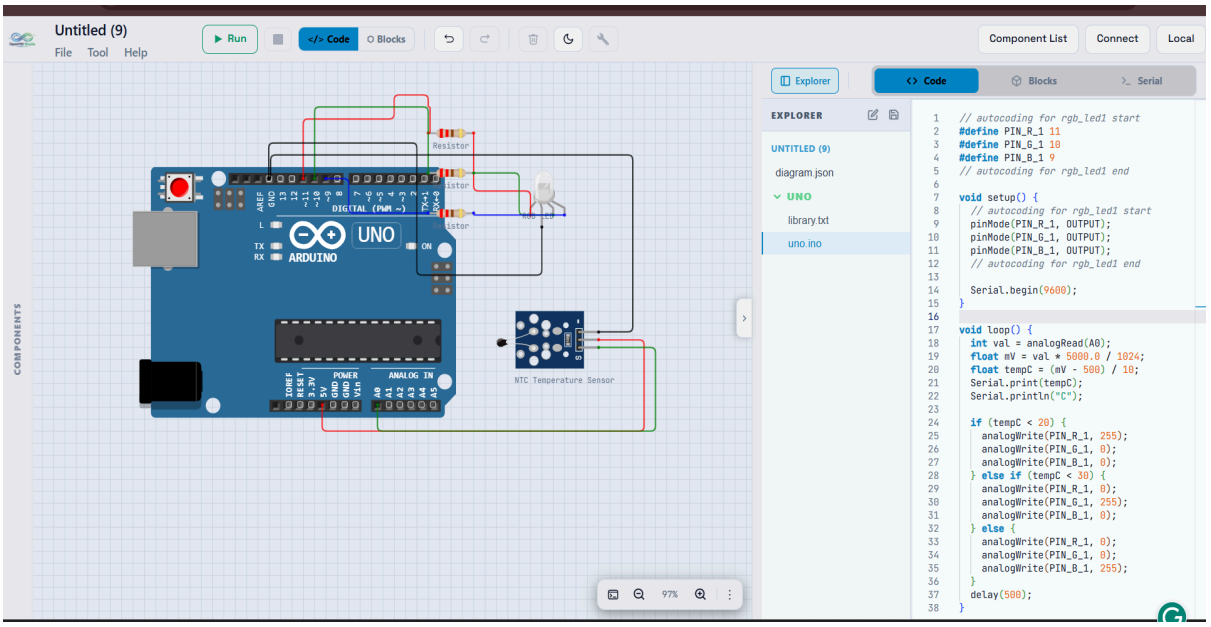
- **Project Configuration & Flow State** Architected the end-to-end logical flow for the guided tutorials. This involved configuring the underlying tutorial metadata (such as step indices and validation triggers) to manage the state progression of the module, ensuring strict synchronization between the user's actions and the instructional UI.

- **Hardware Simulation Implementations** Configured and integrated a comprehensive suite of guided projects designed to simulate core embedded systems concepts, including GPIO manipulation, Pulse Width Modulation (PWM), and Analog-to-Digital Conversion (ADC). The exactly implemented projects include:

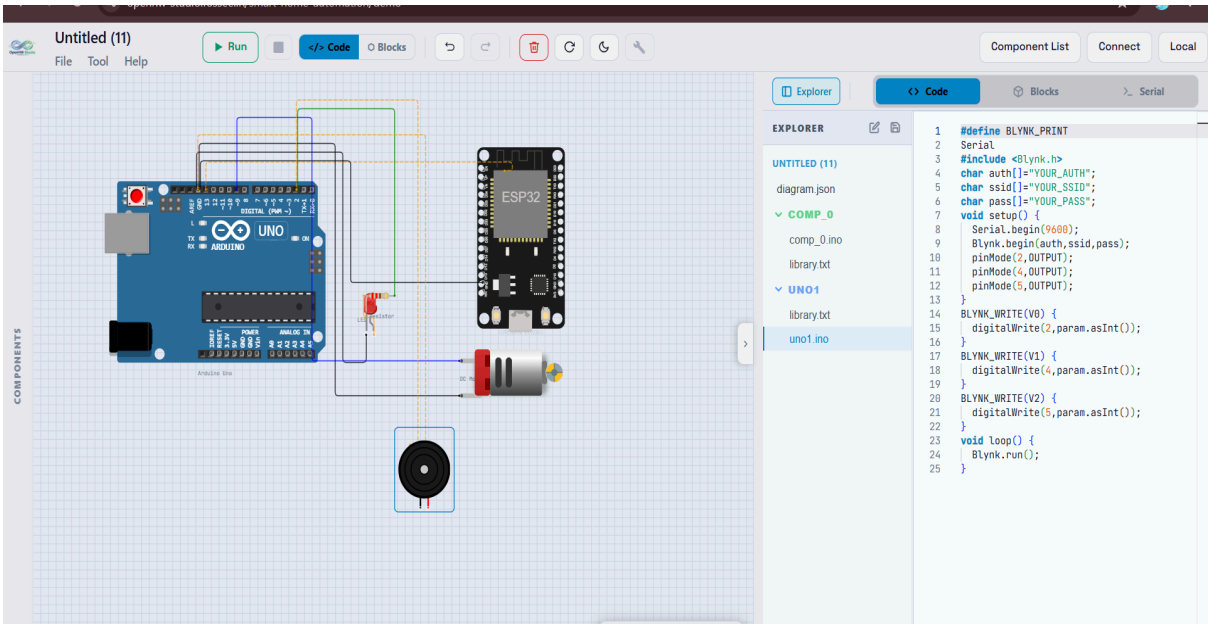
Motion Sensor with Alarm: Simulating digital interrupts and logic-level triggers.



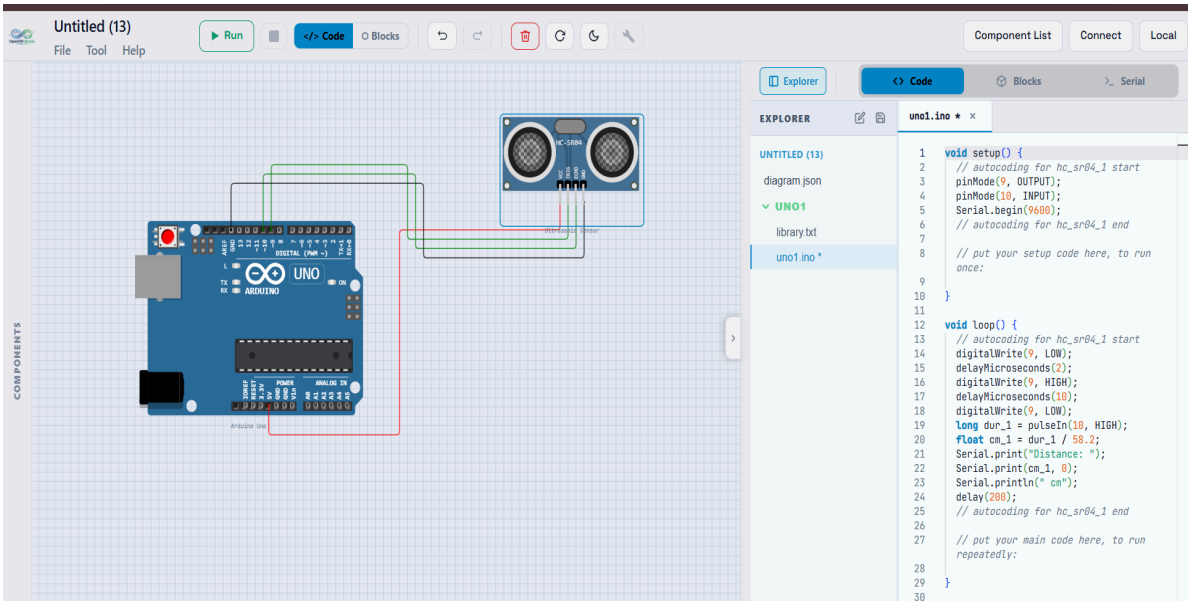
- **Temperature Indicator using RGB LED:** Simulating analog sensor reading and multi-state visual feedback.



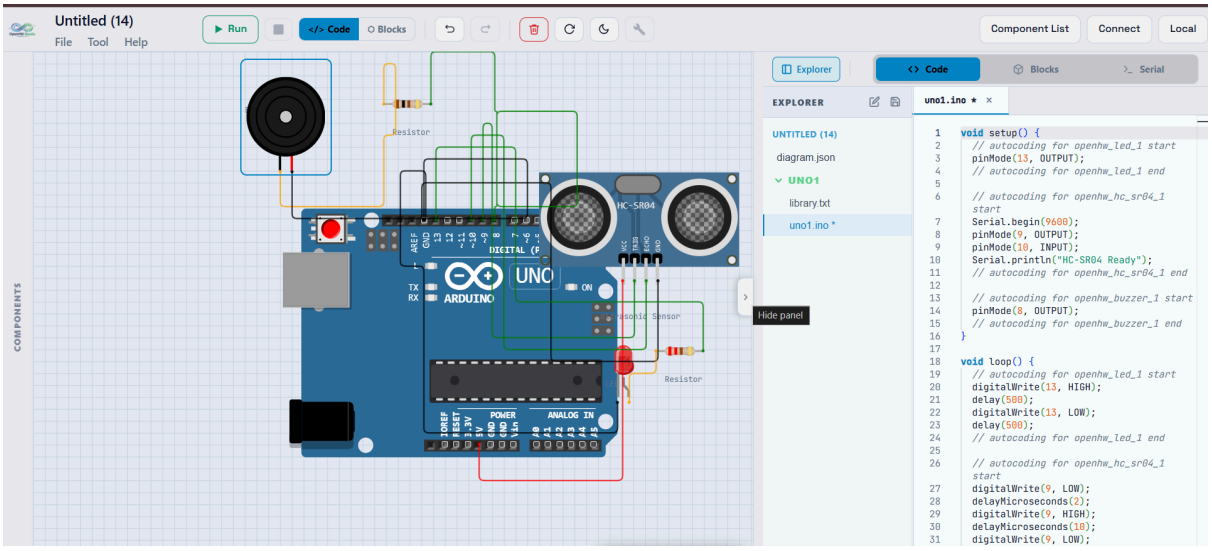
- **Smart Home Automation System:** Designing complex, multi-component integration logic.



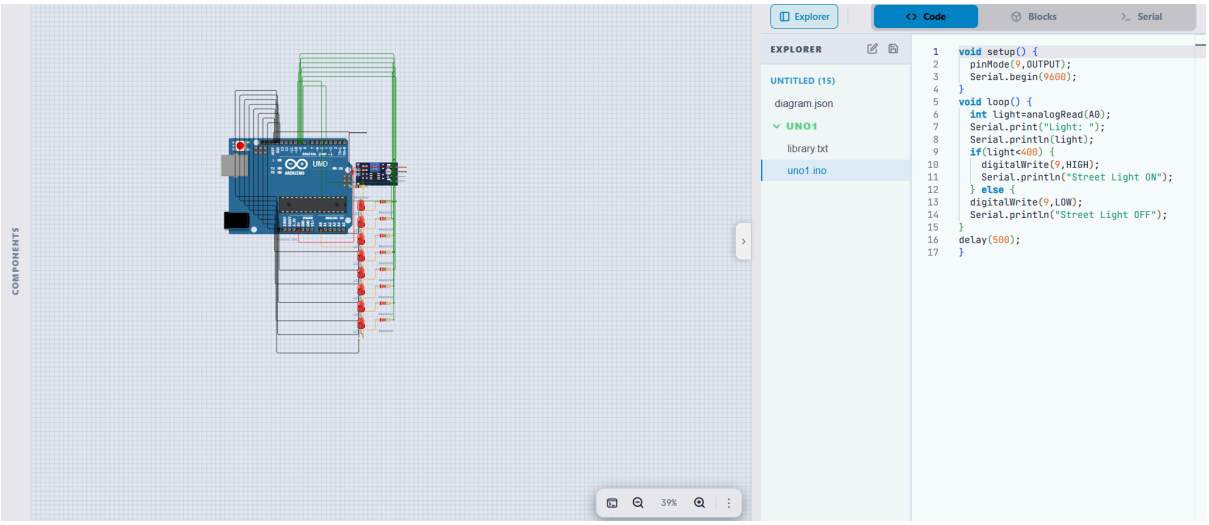
- **Ultrasonic Distance Measurement:** Emulating timing-based sensor data acquisition.



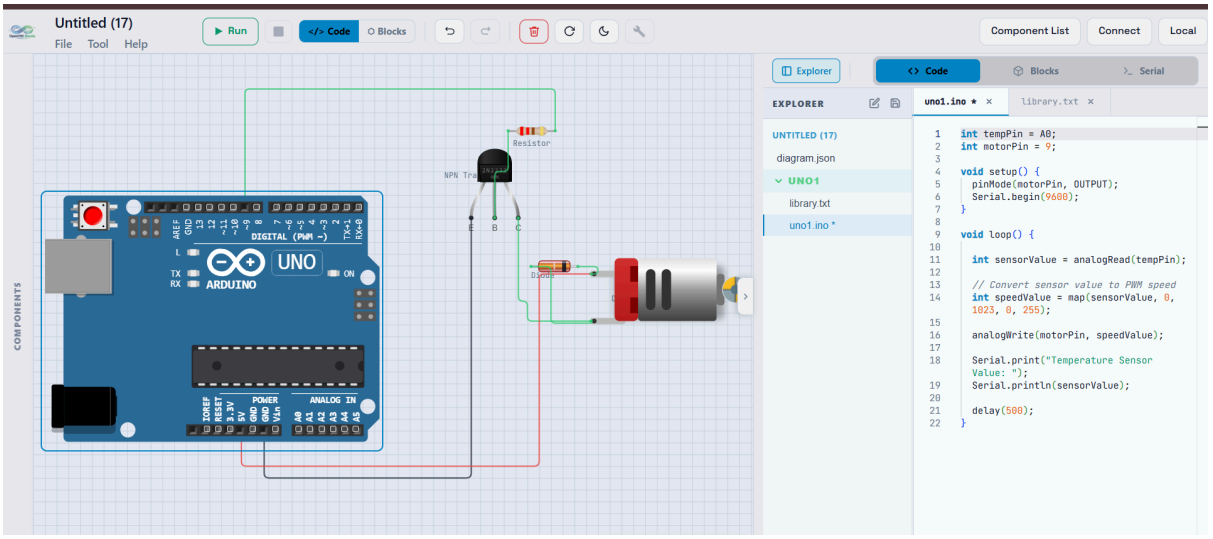
- **Water Level Indicator with Alarm:** Configuring threshold-based sensor logic and buzzer alerts.



- **LDR Based Smart Street Light:** Implementing closed-loop control based on simulated ambient light data.



- **DC Motor Control using PWM:** Configuring variable Pulse Width Modulation for DC motor speed control.



Custom Asset Implementations for Projects Successfully created the visual assets (PNGs) and configured the simulation environments for a comprehensive suite of guided projects. I designed the graphical workflows and implemented the underlying simulation concepts for the following specific experiments:

- Motion Sensor with Alarm: Simulating digital interrupts and logic-level triggers.
- Ultrasonic Distance Measurement: Emulating timing-based sensor data acquisition.
- Temperature Indicator using RGB LED: Simulating analog sensor reading and multi-state visual feedback.
- Smart Home Automation System: Designing complex, multi-component integration logic.
- Automatic Fan Speed Control: Implementing logic for environmental-based motor control.
- Smart Dustbin: Integrating ultrasonic distance sensing with servo motor actuation.
- Water Level Indicator with Alarm: Configuring threshold-based sensor logic and buzzer alerts.
- LDR Based Smart Street Light: Implementing closed-loop control based on simulated ambient light data.
- DC Motor using PWM: Configuring variable Pulse Width Modulation for DC motor speed control.
- Servo Motor Control using Potentiometer: Mapping analog inputs to precise servomotor positional rotation.
- Gas Sensor with LED: Simulating environmental sensor threshold detection and state signaling**.**

Chapter 4 . Emulator Component Design

1. Interactive Hardware Simulation Components This module involved designing and implementing the visual and structural assets used within the platform's core hardware simulation environment. The following technical contributions were made:

● **Component Creation & Visual Design** Designed custom, high-fidelity scalable vector graphics (SVGs) for a wide array of embedded systems hardware. This ensured that the simulation assets were visually intuitive, responsive, and accurately represented their physical counterparts. The specific components designed include:

- **Microcontrollers:** STM32 Bluepill
- **Sensors:** PIR Motion Sensor, Rain Drop Module, NTC Thermometer, MQ2 Gas Sensor, LDR Sensor Module, and DHT11/DHT22 Temperature & Humidity Sensors
- **Actuators & Outputs:** Servo Motor, Stepper Motor (including Biaxial), Relay Module, and NeoPixel Ring
- **Inputs & Controls:** Rotary Potentiometer, Analog Joystick, Slide Switch, DPDT Switch, standard Push Buttons, and 6mm Push Buttons
- **Modules:** RTC (Real-Time Clock) Module

● **Breadboard Alignment & Configuration Correction** Modified and maintained the associated configuration metadata for these hardware components. A major focus of this work involved meticulously recalculating and correcting component dimensions and pin anchors so they aligned perfectly with the standard breadboard grid spacing within the UI, enabling realistic and error-free circuit wiring.

● **Component Logic Integration** Integrated the newly designed visual assets with the underlying emulator engine. This involved binding the graphical SVG states to the simulation data, allowing the frontend components to dynamically react to voltage signals, environmental thresholds, and mechanical state changes in real-time.

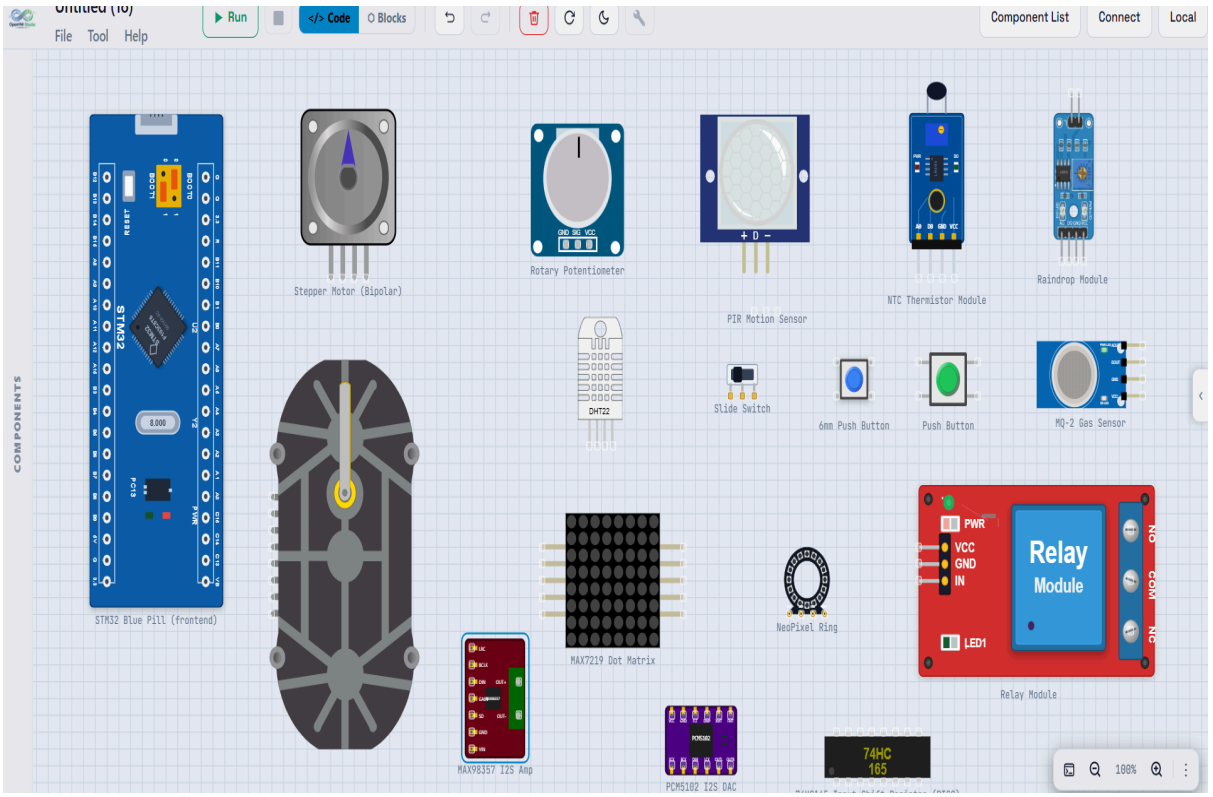


Figure 4.1: *A selection of hardware simulation components designed and implemented in the OpenHW Studio emulator, showcasing high-fidelity SVG assets and precise component configuration.*

Chapter 5.Hardware-in-the-Loop Testing (avrbro)

5.1 To support and validate the flashing of physical Arduino (AVR) boards, I conducted extensive testing on the platform's avrbro library integration. My specific contributions included:

- **Web Serial API Verification:** Actively tested the browser's native Web Serial API integration, verifying that the frontend could establish a stable, direct communication line with connected USB devices across various hardware configurations.

- **Deployment Process Testing:** Verified the end-to-end deployment flow. Once the backend compiled the code and returned the .hex string, I rigorously tested the avrbro parser to ensure it flashed the hex data directly and flawlessly onto the connected Arduino board, confirming that students could operate entirely without the desktop Arduino IDE.

- **Performance & Accuracy Benchmarking:** Monitored and verified upload performance, ensuring the highly optimized process consistently took approximately 8 seconds to fully upload compiled code to a physical board.

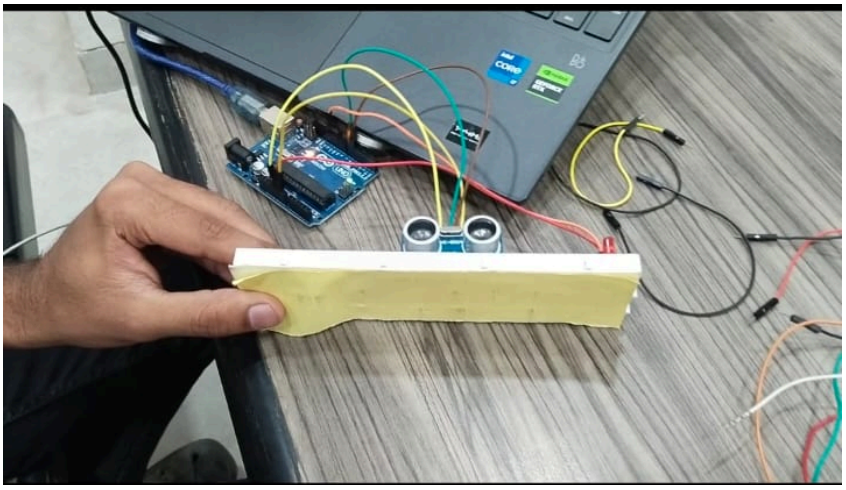
5.2 Component-Level Hardware Verification Experiments To ensure complete system reliability, I conducted a comprehensive set of hardware experiments by physically connecting various electronic components to an Arduino board and cross-verifying their behavior against the platform's simulation output. The following experimental validations were performed:

- **Output Components:** Verified that LEDs (Standard, RGB, and NeoPixel), Buzzers, and 7-Segment Displays responded correctly to the browser-flashed code, confirming accurate GPIO signal output**.**

- **Environmental Sensors:** Tested DHT22 (Temperature & Humidity), NTC Thermistor, MQ2 Gas Sensor, Soil Moisture Sensor, LDR Photoresistor, and PIR Motion Sensor , verifying that analog and digital sensor readings matched the platform's simulated data values**.**

- **Distance & Motion Sensors:** Conducted timing-based verification experiments using the HC-SR04 Ultrasonic Sensor, ensuring the echo pulse timing produced accurate real-world distance measurements consistent with the emulator output.

- **Motor & Actuator Control:** Physically tested DC Motors (via L293D driver), Servo Motors, and Stepper Motors to confirm that PWM signals, directional control, and positional accuracy matched the simulated behavior.
- **Displays & Communication:** Verified LCD1602 (I2C) and OLED SSD1306 displays, confirming that character rendering and graphical output from browser-flashed code matched the platform's simulated screen output.
- **Modules & Advanced Peripherals:** Tested the Relay Module, RTC DS1307, and MFRC522 RFID Reader to ensure correct hardware handshaking and data communication when driven by the browser-deployed firmware.



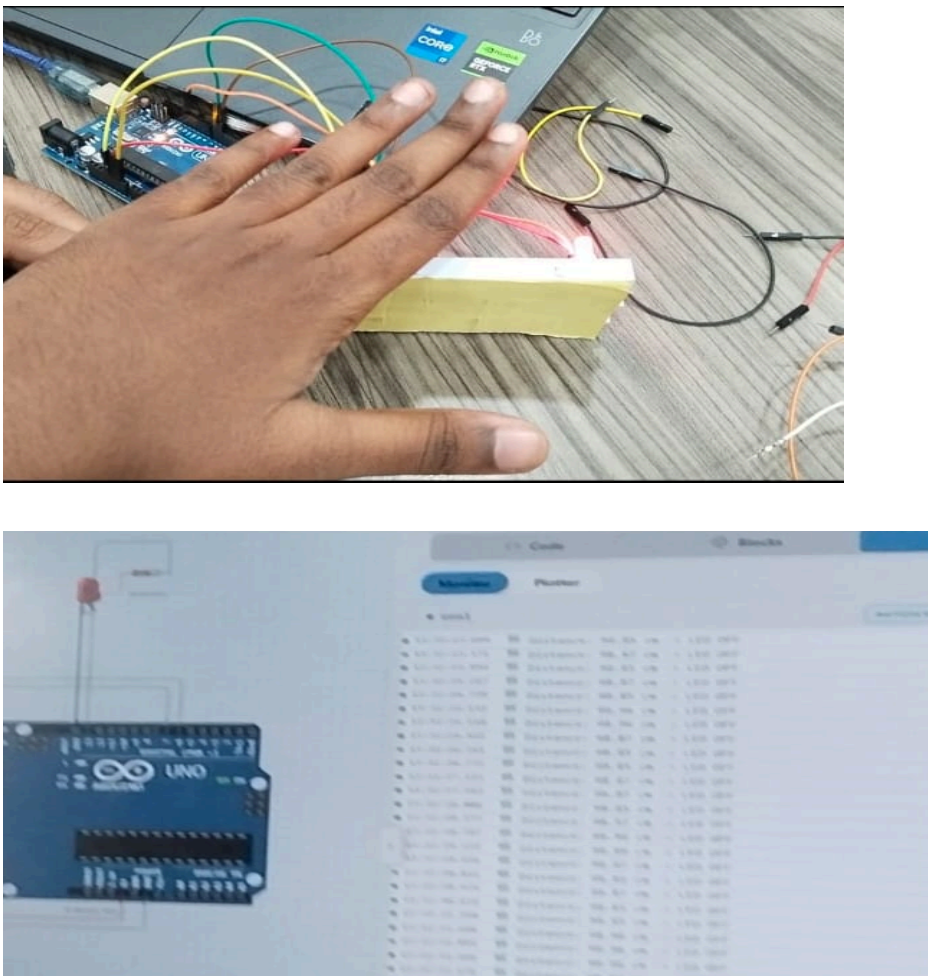


Figure 5.1: Hardware verification experiment demonstrating real-time distance measurement using the HC-SR04 Ultrasonic Sensor, with code flashed directly from the browser to a physical Arduino board.

Chapter 6. Conclusion

his internship has been a deeply enriching and rewarding experience both technically and creatively. By working across the full stack architecture of OpenHW Studio spanning backend authentication systems, frontend UI development, interactive emulator component design, guided project creation, and physical hardware verification, I gained invaluable insights into how modern web platforms are engineered from the ground up.

My work centered on bridging the gap between secure server side middleware and a high fidelity browser based hardware simulation environment. This allowed me to contribute to a platform that provides genuine educational value to students and engineers. The iterative process of designing components from scratch, achieving pixel perfect alignment on virtual breadboards, and validating their behavior against physical Arduino hardware was both challenging and deeply fulfilling.

I am confident that the enhancements I implemented during this internship significantly strengthen OpenHW Studio as both a robust educational platform and a versatile research tool. This experience has not only sharpened my technical proficiency but has also deepened my appreciation for building systems that make complex engineering concepts accessible and intuitive.

Chapter 7. Future Work

Looking ahead, the features and systems can be expanded in several promising directions:

1. **Advanced Authentication & Role Management:** Extending the current authentication system to support OAuth-based social login (Google, GitHub) and introducing fine-grained role-based access control (RBAC) for teachers, students, and administrators.

2. **Expanded Guided Project Library:** Building upon the existing suite of guided projects by introducing advanced-level tutorials covering communication protocols such as I2C, SPI, and UART, as well as IoT-based projects leveraging ESP32 WiFi connectivity.
3. **Automated Component Testing Pipeline:** Introducing a CI/CD-integrated automated testing framework for newly designed emulator components, enabling regression testing to verify that SVG pin alignments and simulation logic remain accurate across platform updates.
4. **Python Compiler Integration:** Expanding the backend to support MicroPython or CircuitPython, allowing students to program virtual hardware using Python in addition to the standard C++ Arduino environment.
5. **Enhanced Physical Deployment Support:** Extending the browser-based flashing pipeline to support additional microcontroller families beyond AVR, including ESP32 and STM32 boards, enabling a wider range of hardware verification experiments.

Chapter 8.References

The following open-source repositories, libraries, and tools were instrumental in the development and testing of the features implemented during this internship:

- **wokwi-elements:** A web component library for electronics simulation. https://github.com/wokwi/wokwi-elements
- **avrbro:** A Web Serial library for flashing AVR microcontrollers directly from the browser. https://github.com/wokwi/avrbro
- **esptool-js:** A JavaScript port of esptool.py for ESP32 firmware deployment over Web Serial. https://github.com/espressif/esptool-js
- **JSON Web Tokens (JWT):** Industry-standard RFC 7519 for secure, stateless authentication tokens. https://jwt.io
- **Node.js & Express.js:** The backend runtime and web framework used for building authentication routes and middleware. https://nodejs.org / https://expressjs.com
- **React.js:** The frontend JavaScript library used for building interactive UI components and authentication interfaces. https://react.dev

- **OpenHW Studio (FOSSEE, IIT Bombay):** The core platform this internship contributed to. https://fossee.in