Semester Long Internship Report

on

OpenHW Studio: Expanding the Arduino Simulator with New Components and Fixes

Submitted by

Snehal Phatale

Pune Institute of Computer Technology, Pune

Under the Guidance of

Prof. Prabhu Ramachandran

Principal Investigator,

Department of Aerospace Engineering,

Indian Institute of Technology Bombay

June 2026

Acknowledgement

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

I would also like to extend my heartfelt thanks to **Prof. Rajesh Kushalkar**, **Senior Project Manager, Open Source Hardware (OSHW), FOSSEE, IIT Bombay**, for his constant encouragement, valuable suggestions, and unwavering support throughout the internship. His guidance played a significant role in enhancing both my technical knowledge and professional development.

I am also thankful to my mentors, **Mr. Pratik Bhosale**, **Project Research Associate, and Mr. Pratik Nemane, Project Research Assistant,** for their continuous mentorship, technical expertise, and thoughtful feedback. Their readiness to clarify doubts, provide practical insights, and guide me through challenges greatly contributed to the successful completion of my assigned work.

I would also like to acknowledge the support of my fellow interns and the colleagues at **OpenHW Studio**. Working alongside such enthusiastic and cooperative individuals made the internship an enjoyable and enriching experience. The collaborative environment encouraged knowledge sharing and helped me grow .

Lastly, I express my sincere appreciation to everyone associated with the **FOSSEE Project, IIT Bombay**, for creating an environment that promotes innovation, teamwork, and continuous learning. This internship has been a valuable milestone in my academic journey, strengthening my technical abilities, improving my problem-solving skills, and inspiring me to continue contributing to open-source software community.

Snehal Phatale
Pune Institute of Computer Technology

Declaration

The internship was a valuable learning experience that allowed me to contribute to the development of virtual hardware modules, implement functional enhancements, and participate in debugging activities within OpenHW Studio. It provided practical exposure to open-source software development, microcontroller simulation platforms, and collaborative software engineering practices. The knowledge, technical skills, and hands-on experience gained during this internship have strengthened my understanding of real-world development workflows and will serve as a strong foundation for my future academic projects and professional career.

I would also like to express my sincere gratitude to the entire FOSSEE team for their continuous guidance, technical support, and encouragement throughout the internship. Their collaborative approach, willingness to share knowledge, and supportive work environment made this experience both enriching and rewarding, enabling me to successfully complete the responsibilities and tasks assigned to me.

Snehal Phatale
Pune Institute of Computer Technology

Index

Chapter 1

Introduction

OpenHW-Studio is an open-source, cloud-based electronics simulation and learning platform developed under the FOSSEE Project at IIT Bombay with the objective of transforming the way embedded systems and electronics are taught and learned. It provides a comprehensive browser-based environment where students, educators, researchers, and electronics enthusiasts can design, simulate, and test electronic circuits without the need for physical hardware. By integrating circuit design, programming, simulation, classroom management, guided learning, and project development into a single platform, OpenHW-Studio makes electronics education more accessible, interactive, and scalable.

Traditional electronics laboratories often require expensive hardware kits, dedicated laboratory infrastructure, and continuous maintenance, making practical learning difficult for many educational institutions. OpenHW-Studio addresses these challenges by providing a virtual laboratory that enables users to perform experiments anytime and from anywhere using only a web browser. The platform supports realistic microcontroller-based simulations, allowing learners to understand hardware behavior, debug programs, and verify circuit functionality before implementing them on physical devices. This significantly reduces development costs while encouraging experimentation, innovation, and self-paced learning.

The platform is designed to support multiple categories of users, including guest users, students, teachers. Guest users can explore the simulator, build circuits, write programs, and execute simulations without creating an account, while registered users gain access to cloud-based project storage, classroom participation, assignment submission, progress tracking, and personalized learning features. Teachers can create virtual classrooms, monitor student activity and evaluate student performance through an integrated classroom management system.

OpenHW-Studio combines several advanced educational features within a unified ecosystem. Users can create circuits using an intuitive drag-and-drop interface, manually connect components, validate circuit connections in real time, and receive immediate error feedback. The platform supports both block-based programming for beginners and text-based programming for advanced users, allowing learners to gradually transition from visual programming concepts to professional embedded software development. Additional tools such as the Serial

Monitor enable real-time debugging and visualization of sensor data, closely replicating the experience of working with actual hardware.

A distinguishing feature of OpenHW-Studio is its intelligent learning assistance. The simulator automatically suggests or inserts supporting electronic components where appropriate, such as current-limiting resistors for LEDs or potentiometers for LCD modules, helping beginners construct correct circuits while still allowing experienced users complete flexibility to modify the design. The platform also incorporates guided projects, example circuits, structured learning pathways, and progressive difficulty levels that encourage systematic skill development.

To further enhance learner engagement, OpenHW-Studio integrates gamification elements including experience points, coins, badges, and level-based component unlocking. As users complete simulations, guided experiments, and assignments, they earn rewards that unlock more advanced electronic components and projects, creating a motivating and interactive learning environment. Classroom workflows, assignment management, and extend the platform beyond simulation into a complete digital learning ecosystem.

From a technical perspective, OpenHW-Studio follows a modular and scalable architecture that supports multiple microcontroller platforms. This focuses on simulation of Arduino Uno and Raspberry Pi Pico boards, ESP32 and STM32 microcontrollers. The system is designed to execute entirely within modern web browsers without requiring additional software installations or plugins, ensuring platform independence and broad accessibility.

Development of OpenHW-Studio follows modern open-source software engineering practices. The project is collaboratively developed using GitHub, where contributors work through feature branches, pull requests, code reviews, issue tracking, and continuous documentation. The source code is organized into dedicated repositories for frontend development, backend services, simulation engines, documentation, and example projects, enabling efficient collaboration among multiple development teams while maintaining high standards of code quality, maintainability, and scalability.

Overall, OpenHW-Studio represents a comprehensive educational platform that combines realistic electronics simulation, embedded programming, classroom collaboration, guided experimentation, and gamified learning into a unified web application. By bridging the gap between theoretical concepts and practical implementation, the platform provides an effective, and research-friendly environment for teaching and learning modern electronics and embedded

systems, while fostering collaboration, innovation, and open-source software development.

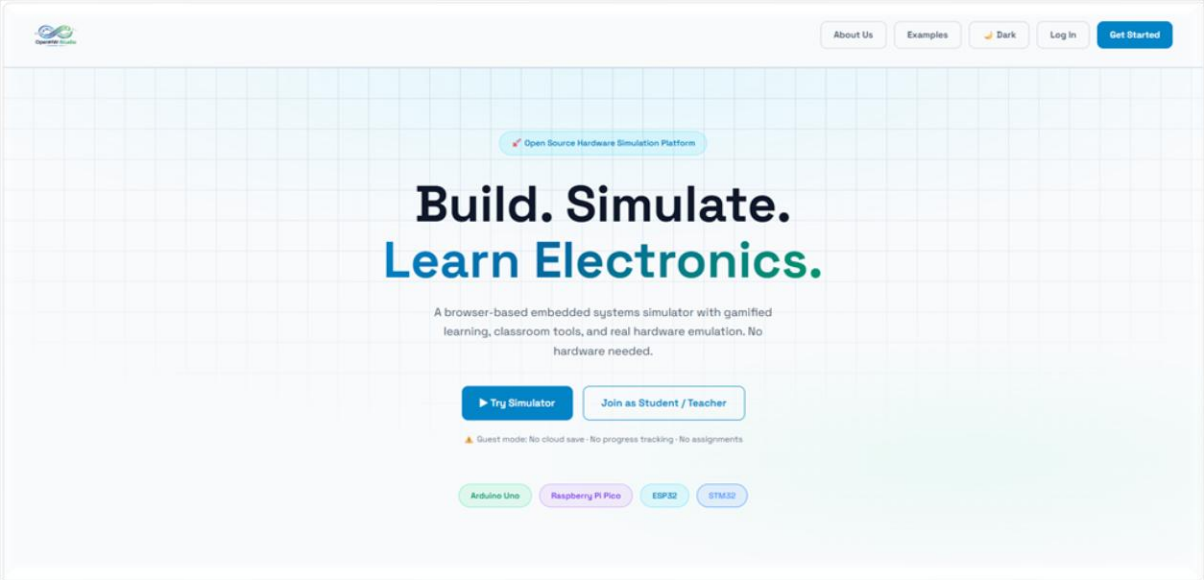

Figure 1: Landing page

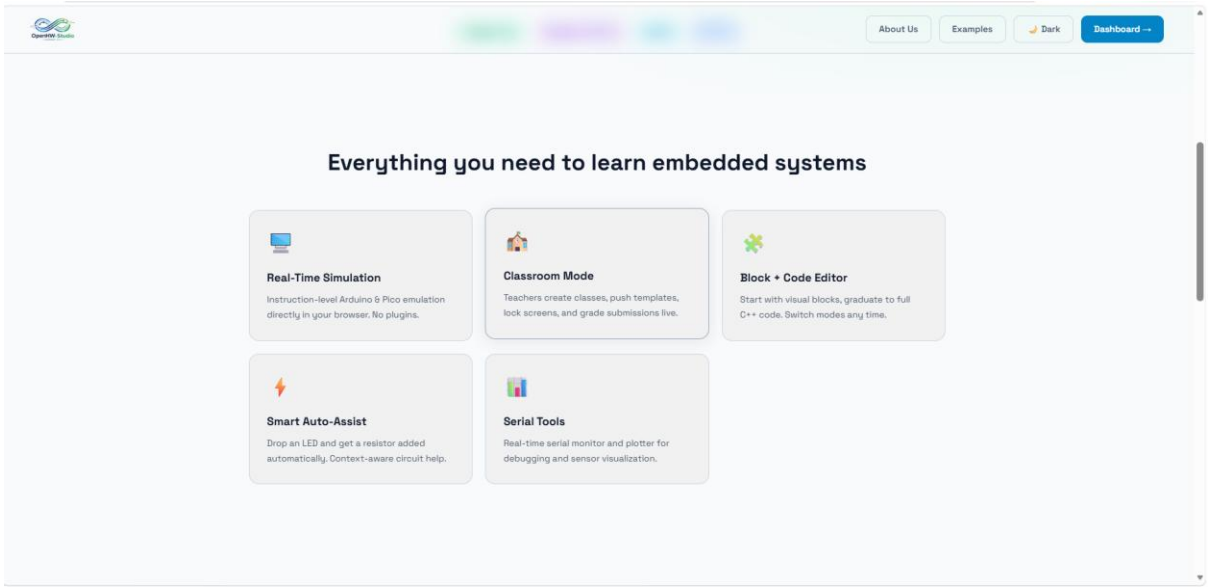

Figure 2: Landing page

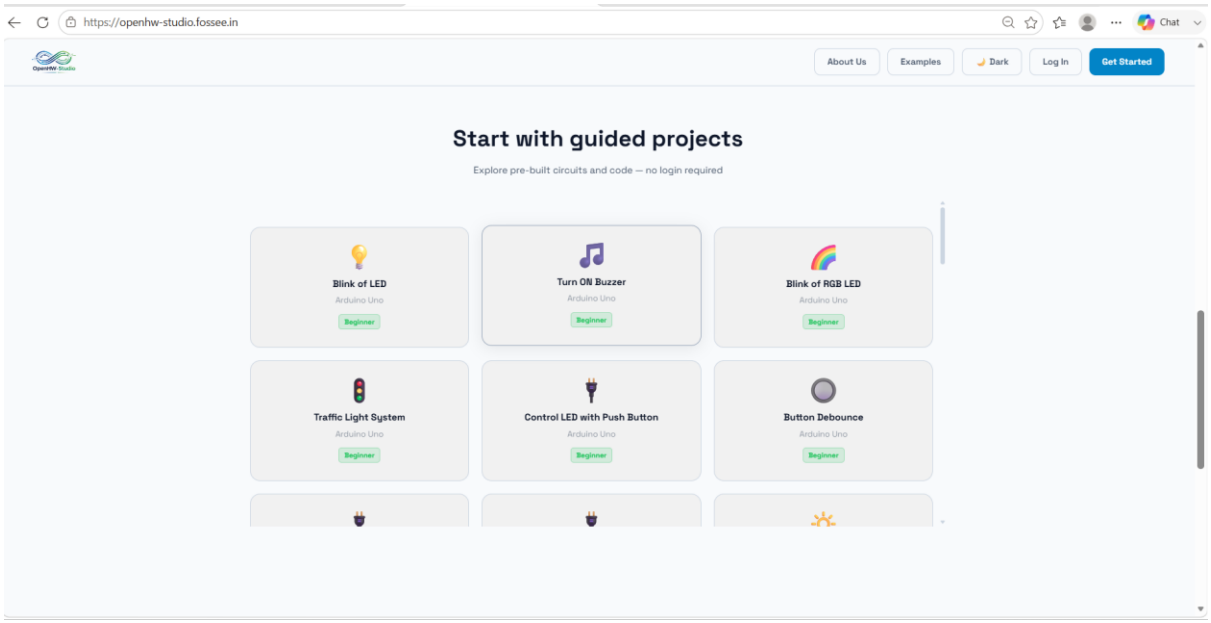

Figure 3: Landing page

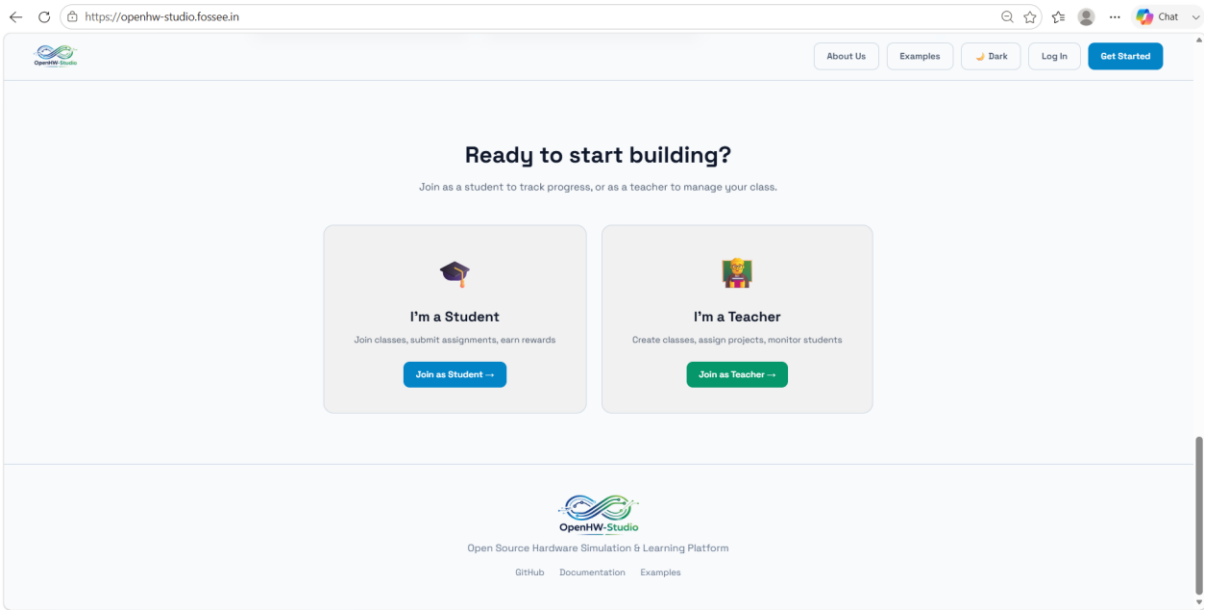

Figure 4: Landing page

1.1 Project Overview

The platform is designed to provide an interactive environment where students, educators, and electronics enthusiasts can design electronic circuits, write programs, simulate hardware, and learn embedded systems concepts entirely through a web browser without requiring physical electronic components. Its primary objective is to make practical electronics education more accessible, engaging, and cost-effective while promoting open-source learning and collaborative development.

The platform integrates circuit design, programming, simulation, classroom management, guided learning, and project management into a unified web application. Users can create electronic circuits using a drag-and-drop interface, connect various electronic components, write programs using either Blockly-based visual programming or Arduino C/C++ code, and execute simulations in real time. The simulator enables users to observe the behavior of electronic components and verify the correctness of their designs before implementing them on physical hardware.

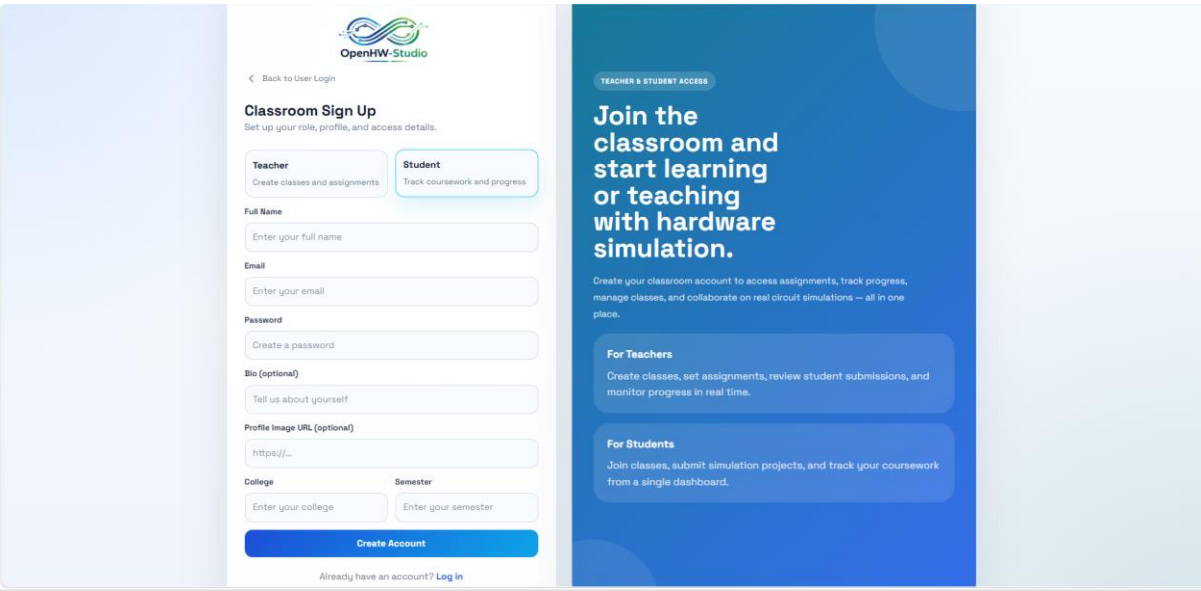

Figure 5: Sign Up page for student

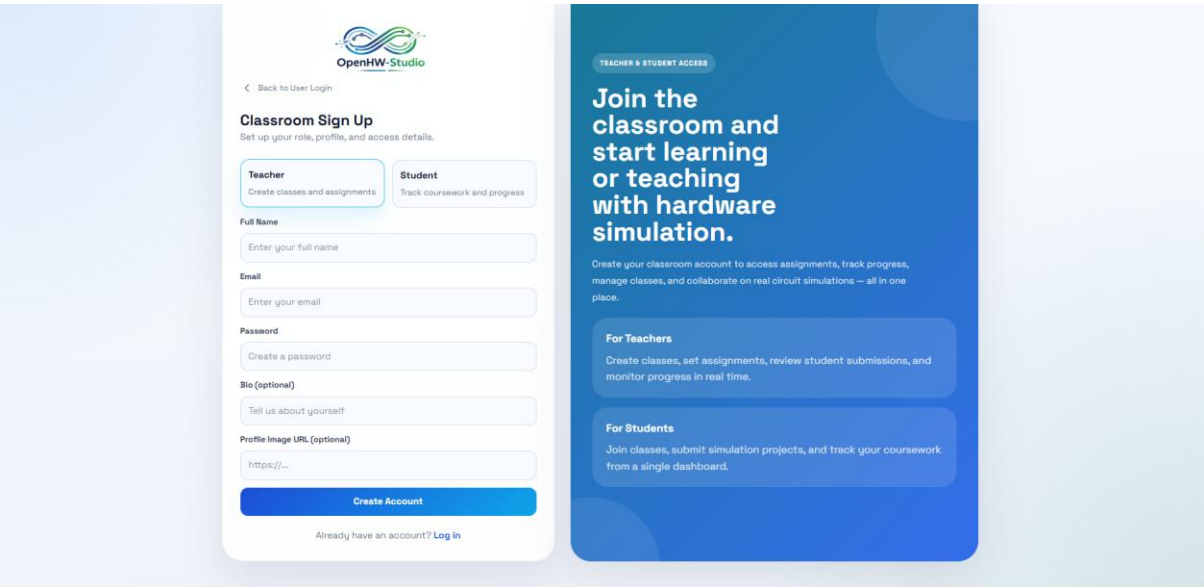

Figure 6: Sign Up page for teacher

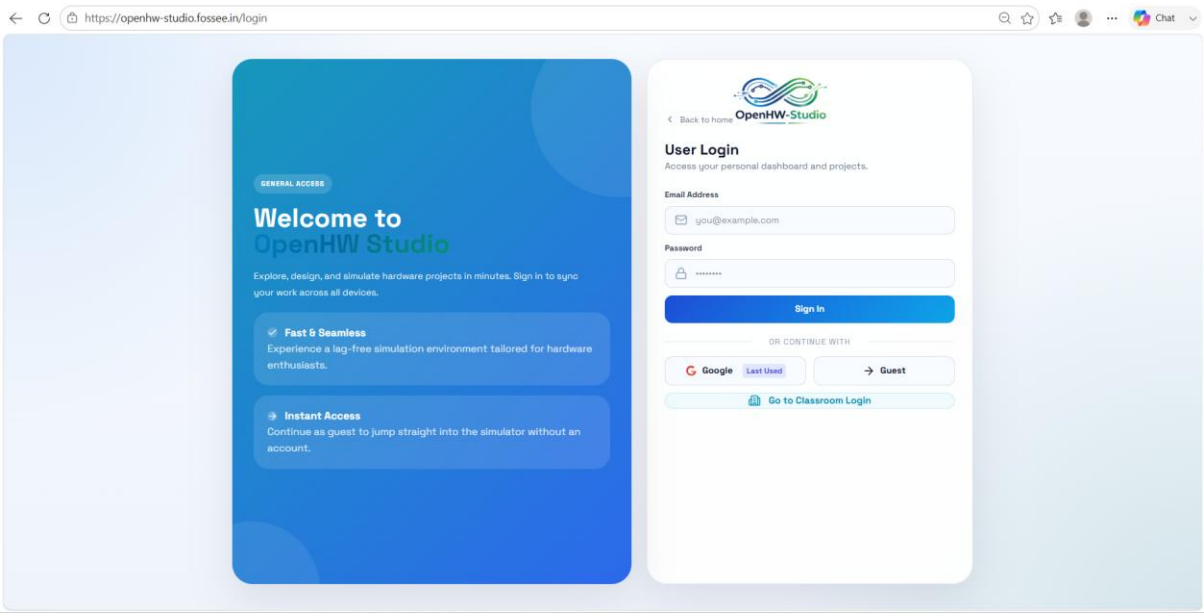

Figure 7: User Login Page of OpenHW-Studio

OpenHW-Studio supports multiple categories of users, including guest users, students, teachers. Guest users can explore the simulator, build circuits, write programs, and run simulations without creating an account. Registered students gain additional features such as cloud project storage, assignment submission, progress tracking, and access to classroom activities. Teachers can create and manage classes, assign practical exercises, monitor student progress, evaluate

submissions, and provide feedback through integrated classroom management tools.

The platform provides multiple programming modes to accommodate users with different experience levels. Beginners can create programs using a Blockly-based visual programming interface, while advanced users can develop applications using text-based Arduino programming. The simulator also includes tools such as a Serial Monitor for debugging and visualizing sensor data during program execution.

One of the key features of OpenHW-Studio is the Guided Projects module, which provides structured, step-by-step practical experiments that help learners understand embedded systems concepts through interactive simulations. These projects cover a wide range of topics, including LED control, sensors, displays, motors, communication modules, and Internet of Things (IoT) applications. By combining theoretical concepts with practical experimentation, Guided Projects provide an effective learning experience for users at different skill levels.

The platform supports the simulation of various electronic components such as LEDs, RGB LEDs, push buttons, buzzers, potentiometers, LCD displays, seven-segment displays, servo motors, DC motors, ultrasonic sensors, temperature sensors, light sensors, gas sensors, Bluetooth modules, and several other embedded system peripherals. It also includes intelligent features such as automatic component assistance, real-time circuit validation, error highlighting, and interactive debugging to improve usability and reduce common design errors. OpenHW-Studio has been designed with a modular and scalable architecture that supports multiple microcontroller platform, including Arduino Uno, Raspberry Pi Pico, ESP32, and STM32. The platform is developed collaboratively using GitHub, where contributors implement new features, review code, perform testing, fix bugs, and continuously improve the software following standard open-source development practices.

Overall, OpenHW-Studio serves as a comprehensive educational platform that combines electronics simulation, programming, guided learning, classroom management, and collaborative development into a single integrated environment. It enables users to learn, experiment, and develop embedded systems applications efficiently while fostering innovation through open-source software.

1.2 Internship Overview

As part of my semester long internship under the FOSSEE Project, IIT Bombay, I contributed to the development and testing of various modules of OpenHW-

Studio, an open-source web-based electronics simulation and learning platform. My primary responsibilities included contributing to the implementation of the Guided Projects module, which provides structured, step-by-step practical experiments to help users learn embedded systems concepts through interactive simulations. I also assisted in parts of the user authentication system, including the login workflow, first-time password setup, and the initial implementation of the forgot password process. In addition, I was responsible for testing the Block Programming (Blockly) module by verifying the functionality of different programming blocks, ensuring their proper execution within the simulator, and testing their integration with Guided Projects. Throughout the internship, I collaborated with mentors and team members to identify issues, verify fixes, and improve the overall reliability and user experience of the platform, gaining practical experience in backend development, software testing, debugging, and collaborative open-source software development.

Chapter 2

Work Contributions / Feature Additions

2.1 User Authentication Module

One of my contributions during the internship was assisting in the development of the User Authentication Module of OpenHW-Studio. Authentication is one of the fundamental components of any web application because it ensures that only authorized users can access platform resources and services. In OpenHW-Studio, the authentication module plays a crucial role in managing user identities, protecting user data, and providing secure access to features such as classroom management, cloud project storage, assignment submission, and personalized dashboards. Since different users, including students, teachers, have different access privileges, implementing a reliable authentication system is essential for maintaining the overall security and integrity of the platform.

My work focused on assisting in the development of different parts of the login and account management system. I contributed to implementing and testing portions of the login functionality while supporting the password setup process for new users. The authentication workflow was designed to provide a simple yet secure user experience by allowing users to enter their credentials, verify their identity, and access the platform according to their assigned role. Throughout the implementation process, attention was given to both security and usability so that users could easily access their accounts while preventing unauthorized access.

A significant part of my contribution involved working on user input validation. I implemented password validation requirements, verified login credentials, handled invalid user inputs, and ensured that meaningful error messages were displayed whenever required. Password validation included checking whether passwords satisfied the required security conditions, and preventing invalid or incomplete submissions. Login validation involved verifying that mandatory fields such as email and password were entered correctly before authentication requests were processed. These validation mechanisms reduced the possibility of incorrect user inputs and improved the overall reliability of the authentication system.

In addition to implementing validation logic, I worked on improving the user experience by ensuring that appropriate feedback was provided whenever authentication failed. Instead of allowing users to proceed with incorrect information, the system displayed clear validation messages indicating missing fields, invalid credentials, incorrect password formats, or unsuccessful login attempts. Providing informative feedback helps users quickly identify and correct

errors, resulting in a smoother and more user-friendly login experience while also reducing confusion during account creation and sign-in.

The authentication module was designed to support secure account management throughout the user lifecycle. During implementation and testing, I verified different stages of user interaction, including account login, password setup, password confirmation, and authentication validation. Each stage required careful testing to ensure that users experienced a seamless workflow from entering their credentials to successfully accessing the platform. By focusing on both functionality and usability, the authentication module contributes to a secure and reliable entry point for all OpenHW-Studio users.

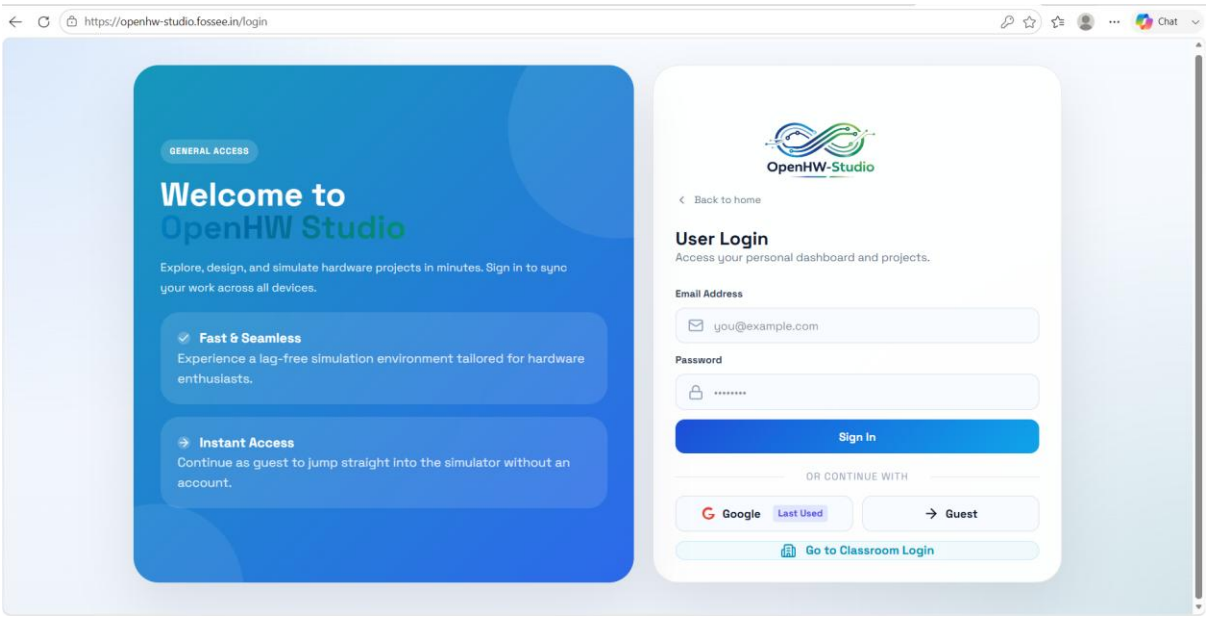

Figure 8: User Login Page of OpenHW-Studio

Figure 4 illustrates the User Login Page of OpenHW-Studio. The login interface is designed with a clean and intuitive layout that allows users to sign in using their registered email address and password. The page provides clearly labeled input fields, a password visibility option, validation support, and navigation links for account creation and password recovery. The simple interface minimizes user confusion while maintaining the security standards expected from a modern web application. Since the login page serves as the first point of interaction for registered users, particular emphasis was placed on ensuring that it remained responsive, user-friendly, and functionally reliable.

In addition to implementing authentication-related features, I participated in testing various user scenarios to ensure that the login workflow functioned correctly under different conditions. Functional testing was carried out using

multiple combinations of valid and invalid user inputs to verify that the authentication system responded correctly in every situation. This included verifying successful login attempts using valid credentials, unsuccessful login attempts using incorrect passwords, invalid email formats, empty input fields, duplicate password entries during account creation, and other possible user interactions. Each scenario was carefully evaluated to ensure that the application behaved as expected and displayed appropriate responses.

I also verified password creation and confirmation rules during account registration and password setup. The testing process involved confirming that passwords satisfying all validation requirements were accepted successfully, while passwords failing the defined criteria generated meaningful validation messages. Particular attention was given to ensuring that confirmation passwords matched the original password, mandatory fields were not left empty, and invalid inputs were rejected before submission. These validation checks significantly improved the robustness of the authentication process by preventing common user errors during account creation.

Another important aspect of my work involved verifying the effectiveness. This included checking for incomplete forms, invalid credentials, incorrect password formats, and other exceptional scenarios that could affect the login workflow. Through repeated execution of these test cases, I helped identify inconsistencies and contributed to improving the stability of the authentication module.

Testing also involved verifying the consistency of error handling throughout the authentication workflow. Whenever invalid inputs were detected, the application displayed descriptive error messages that guided users toward correcting their mistakes. I confirmed that these messages appeared at the appropriate time, accurately described the identified problem, and disappeared once valid information was entered. Providing immediate and meaningful feedback plays an important role in enhancing usability while reducing the possibility of repeated user errors.

Throughout the internship, I collaborated with other team members by integrating my work with the existing frontend and backend modules while following the project's GitHub-based development workflow. Development activities were carried out using feature branches to ensure that new changes remained isolated until they were fully tested and reviewed. After implementing the required functionality, I performed local testing, committed the modifications with meaningful commit messages, and participated in the pull request review process before the changes were merged into the main development branch. This

workflow helped maintain code quality while encouraging collaborative development among contributors.

Working within a large open-source project also exposed me to modern software engineering practices such as version control, issue tracking, collaborative debugging, and continuous code integration. Interacting with mentors and fellow contributors allowed me to understand how different software modules are integrated, reviewed, tested, and maintained throughout the development lifecycle. These collaborative experiences enhanced both my technical knowledge and my understanding of professional software development workflows.

Overall, working on the User Authentication Module provided me with valuable practical experience in implementing secure user management features, validating user inputs, testing authentication workflows, debugging login-related issues, and improving the overall user experience. It strengthened my understanding of authentication systems, account management, software testing, and secure web application development while providing practical exposure to collaborative open-source software engineering. The knowledge and skills gained while contributing to this module will serve as a strong foundation for future work involving backend development, authentication systems, and modern web application security.

2.2 Development and Implementation of Guided Projects

One of my major contributions during the internship was working on the Guided Projects module of OpenHW-Studio. Guided Projects are designed to provide students with a structured learning experience by guiding them through practical embedded systems experiments within the simulator. Unlike a traditional simulator where users begin with a blank workspace, Guided Projects provide step-by-step instructions, predefined circuit configurations, and programming tasks that help learners understand both hardware and software concepts in a systematic manner. The primary objective of this module is to simplify the learning process by breaking complex embedded systems concepts into smaller, manageable activities that students can complete independently. Each guided project is carefully designed to provide clear instructions, expected outcomes, and hands-on interaction with electronic components, allowing learners to gain confidence while gradually developing practical skills.

The Guided Projects module serves as an important educational component of OpenHW-Studio because it bridges the gap between theoretical knowledge and

practical implementation. Rather than expecting users to understand every hardware connection and programming concept from the beginning, the platform provides a structured workflow that introduces concepts progressively. Students can follow detailed instructions, assemble virtual circuits, write or execute code, observe simulation outputs, and understand how different electronic components interact within an embedded system. This guided learning methodology helps reduce the learning curve for beginners while also providing more advanced learners with opportunities to explore complex projects involving multiple sensors, actuators, communication modules, and control systems.

As part of the backend team, I contributed to the implementation and integration of Guided Projects so that users could access learning modules smoothly within the platform. My work involved supporting the backend functionality required for loading project data, verifying project execution, and ensuring that the guided workflow functioned correctly. I worked with the project structure to ensure that the necessary files, project metadata, source code, and instructional content were correctly linked and loaded whenever a user selected a guided project. This required careful verification of project configurations and coordination with other modules of the platform to ensure seamless execution within the simulator.

In addition to implementation, I participated in validating the overall project workflow from project selection to successful execution. This included verifying that users could navigate through guided instructions without interruptions, ensuring that circuit components appeared correctly within the simulation environment, checking that project descriptions and instructional content were displayed accurately, and confirming that each project could be executed without unexpected errors. These activities contributed to improving the stability and usability of the Guided Projects module while ensuring a smooth learning experience for end users.

The Guided Projects covered multiple difficulty levels to support learners with different levels of experience. At the Beginner Level, projects focused on introducing fundamental electronics and programming concepts. These projects were designed to help students understand digital input/output operations, analog sensing, serial communication, display interfaces, and basic embedded programming. Through simple experiments such as LED blinking, button interfacing, buzzer control, potentiometer reading, LDR sensing, LCD interfacing, and serial communication, beginners could gradually develop an understanding of essential embedded system concepts before progressing to more advanced topics. The structured nature of these projects enables students to build confidence while learning both circuit design and programming fundamentals.

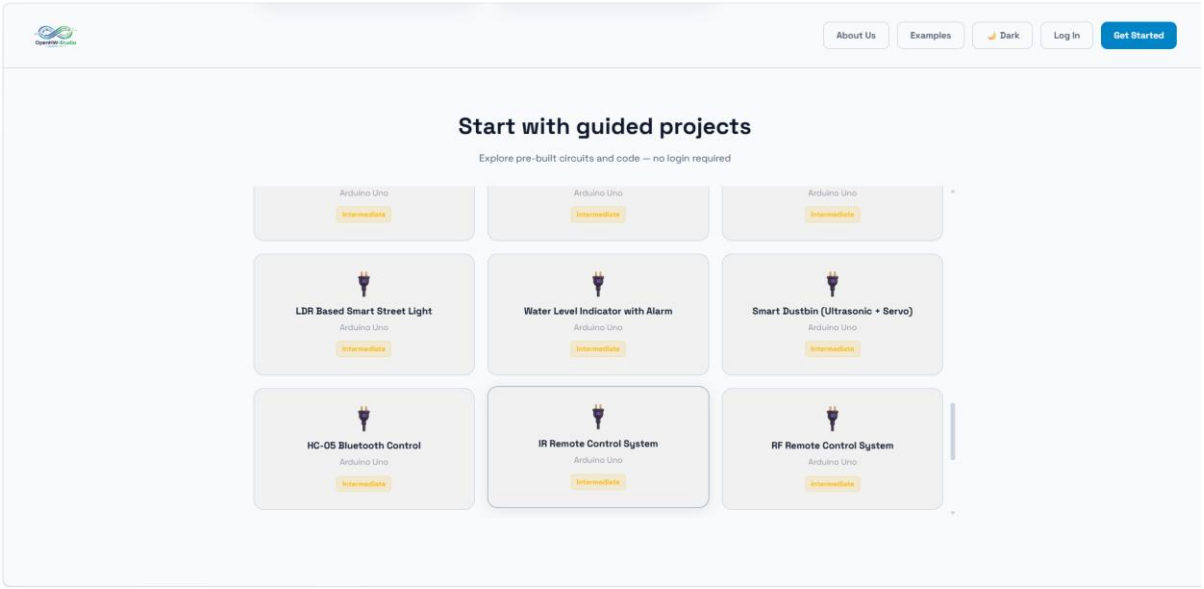

Figure 9: Guided Projects (Intermediate Level)

The Intermediate Level introduced more advanced embedded system concepts by combining sensors, actuators, displays, and communication modules. Guided Projects includes RGB LED, Ultrasonic Distance Measurement, Motion Sensor with Alarm, Gas Sensor with LED Indicator, DHT22 Sensor Display on LCD, Servo Motor Rotation, Servo Motor Control using Potentiometer, DC Motor Control using PWM, Automatic Fan Speed Control, LDR-Based Smart Street Light, Water Level Indicator with Alarm, Smart Dustbin using Ultrasonic Sensor and Servo Motor, HC-05 Bluetooth Control, IR Remote Control System, RF Remote Control System, and several other embedded systems experiments. These projects required students to integrate multiple hardware components within a single application, understand sensor data processing, implement control logic, and analyze simulation outputs. The intermediate-level projects provided learners with practical exposure to solving real-world engineering problems while reinforcing concepts learned in the beginner stage.

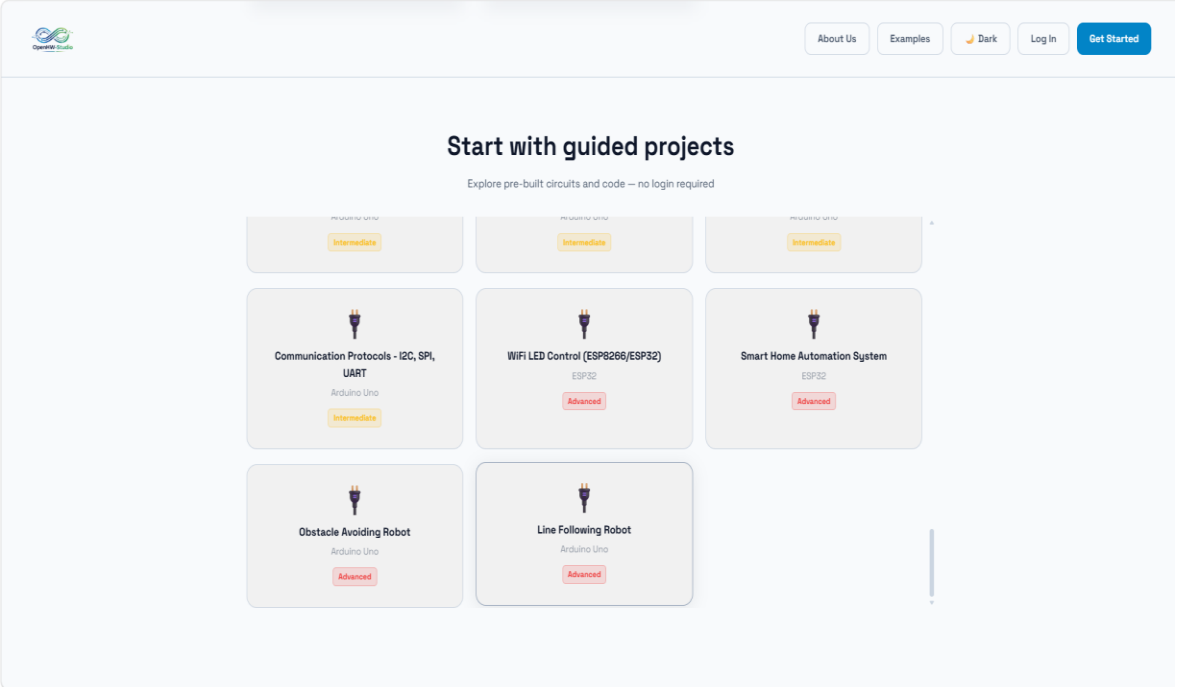

Figure 10: Guided Projects (Advanced Level)

The platform also includes Advanced Level Guided Projects that expose learners to practical Internet of Things (IoT), automation, and robotics applications. These projects provide students with real-world applications of embedded systems by integrating multiple peripherals, communication protocols, and intelligent control mechanisms within a single project. Advanced Guided Projects are intended to simulate practical engineering applications where learners must understand both hardware integration and software development. This level prepares students for industry-oriented projects by encouraging them to apply their accumulated knowledge to more complex embedded system designs.

During development, I worked on implementing parts of these Guided Projects and verifying that they loaded correctly within the simulator. My responsibilities included checking project configurations, validating project resources, ensuring that the required files were properly linked, and confirming that each project executed correctly inside the simulation environment. I also assisted in verifying that the predefined circuit layouts, source code, instructional steps, and expected outputs were synchronized correctly so that users could complete projects without unnecessary interruptions. These verification activities helped maintain consistency across different Guided Projects available on the platform.

A significant portion of my work involved extensive testing of the Guided Projects module. I tested project navigation, validated instructional content,

checked simulator compatibility, verified correct execution of the provided code, and confirmed that the expected outputs matched the intended learning objectives of each project. I executed projects under different scenarios to ensure that instructions appeared in the correct sequence, code executed without errors, and simulation results accurately reflected the implemented logic. Whenever inconsistencies or unexpected behavior were observed, I documented the issues, communicated them with mentors and team members, and participated in verifying the fixes after implementation.

I also worked on ensuring that the Guided Projects remained compatible with the evolving simulator environment. Since OpenHW-Studio is actively developed, updates to different modules can affect project execution. Therefore, regression testing was performed whenever necessary to verify that previously functioning Guided Projects continued to operate correctly after new features or modifications were introduced. This process improved my understanding of software maintenance, quality assurance, and the importance of continuous testing throughout the software development lifecycle.

Throughout the internship, I collaborated closely with mentors and fellow contributors using the project's GitHub-based development workflow. Code modifications, project updates, testing activities, and issue reporting were managed using feature branches, commits, pull requests, and code reviews. This collaborative development process provided valuable exposure to modern software engineering practices while ensuring that all changes were thoroughly reviewed before being integrated into the main project.

Working on the Guided Projects module gave me valuable experience in educational software development, backend integration, software testing, debugging, feature validation, and collaborative software engineering. More importantly, it helped me understand how carefully designed educational content can significantly enhance student learning by combining theoretical concepts with practical simulations. Through this experience, I gained a deeper appreciation of how structured practical exercises, interactive simulations, and guided experimentation can make embedded systems education more engaging, effective, and accessible. The knowledge and practical skills acquired while working on this module have strengthened my understanding of both software development and educational technology, making this one of the most valuable learning experiences of my internship.

2.3 Testing of Block Programming

As part of the OpenHW-Studio project, I performed comprehensive testing of the Block Programming module to ensure the correctness, reliability, and stability of the simulator. Block Programming is one of the core learning features of OpenHW-Studio, designed primarily for beginners who are new to embedded systems and programming. Instead of writing source code manually, users can create programs by dragging and connecting visual programming blocks that represent different programming constructs and hardware operations. The platform automatically converts these blocks into executable code, enabling users to understand programming logic through a visual interface. Since Block Programming serves as an entry point for many students, ensuring its correctness and usability was an essential part of the development and testing process.

The primary objective of testing this module was to verify that each Blockly block functioned correctly both individually and when combined with other blocks to create complete embedded system applications. Testing was performed across different categories of blocks to ensure that the generated programs executed successfully within the simulator and produced outputs that matched the expected behavior. I carefully evaluated the interaction between different blocks, verified the correctness of generated logic, and ensured that users could create functional programs without encountering unexpected errors or inconsistencies.

One of the first stages of testing involved the Basic Blocks, which are commonly used by beginners while learning programming concepts. I tested the functionality of blocks such as Clear Screen, Show Icon, Show Number, Show String, Heart Icons, Plot Bar Graph, and several other display-related blocks. Each block was executed individually as well as in combination with other blocks to verify correct behavior. I confirmed that text, numbers, icons, and graphical outputs were displayed accurately within the simulator. I also tested multiple execution scenarios to ensure that display updates occurred correctly, previous outputs were cleared when required, and sequential display operations functioned without conflicts. These tests ensured that learners would receive accurate visual feedback while experimenting with different programming concepts.

The Control Blocks formed another important category of testing because they determine the overall execution flow of Blockly programs. I verified the behavior of blocks such as On Start, Forever, Wait, Repeat, and other control structures under different execution conditions. Multiple combinations of nested loops, repeated execution, delays, and sequential operations were tested to ensure that program flow matched the intended logic. Special attention was given to verifying that initialization code executed correctly before entering continuous

execution loops, that delay functions behaved accurately, and that repetitive execution produced consistent results throughout the simulation. These tests were essential in confirming that users could implement reliable program logic using visual programming blocks.

I also performed extensive testing of Output Blocks, which control various hardware components available within the simulator. This included validating LED plotting, brightness control, digital output, analog output (PWM), servo motor rotation, and servo pulse generation using different sample circuits. Multiple output devices were tested individually and together to ensure that they responded correctly to different input values and program logic. PWM values were varied across their supported range to verify analog output behavior, while servo motors were tested using different angle values to confirm smooth and accurate movement within the simulator. These tests ensured that the simulated hardware accurately reflected the behavior expected from actual embedded systems.

Another important aspect of testing involved the Math and Text Blocks, which provide essential computational and string manipulation capabilities within Blockly programs. I executed programs involving arithmetic operations, comparison operators, logical operators, random number generation, value mapping, minimum and maximum calculations, string manipulation, text concatenation, number-to-text conversion, and other utility functions under different scenarios. The objective was to verify that mathematical calculations produced accurate results and that text operations behaved consistently across different input values. Various edge cases and boundary conditions were also considered to ensure that the blocks remained stable under different operating conditions.

In addition to computation-related blocks, I verified the functionality of Input, Variables, Lists, and Sensor Blocks, which are essential for creating interactive embedded applications. Testing included digital inputs, analog inputs, push buttons, potentiometers, temperature sensors, light sensors, accelerometers, rotation sensors, compass heading, variable initialization, variable assignment, arithmetic operations on variables, list creation, list manipulation, and other supported Blockly blocks. Different input values were supplied during simulation to verify that the generated programs responded appropriately to changing sensor conditions. Variable operations were tested repeatedly to ensure correct storage, retrieval, updating, and utilization of data throughout program execution.

Sensor-related testing was particularly important because many Guided Projects rely on real-time sensor data to demonstrate embedded systems concepts. I

verified that sensor values were correctly read by Blockly programs and that corresponding outputs changed dynamically according to the simulated environmental conditions. Various scenarios involving changing light intensity, temperature values, analog voltages, digital signals, and motion detection were executed to ensure that the simulator accurately represented hardware behavior. These tests confirmed that learners could effectively understand sensor interfacing through interactive simulation without requiring physical hardware.

To evaluate the robustness of the Block Programming module, I executed complete Blockly programs involving multiple electronic components simultaneously. These included practical applications using LEDs, RGB LEDs, Buzzers, LCD Displays, Seven-Segment Displays, Sensors, Servo Motors, DC Motors, and several other supported devices. Rather than testing individual blocks in isolation, these complete programs simulated real embedded system applications where multiple hardware components interacted within the same project. I verified that the generated programs executed successfully, that outputs matched expected results, and that different components operated together without conflicts. These comprehensive tests provided greater confidence in the overall stability of the Blockly environment.

A significant part of my responsibilities involved performing integration testing between the Block Programming module and the Guided Projects module. Since many Guided Projects depend on Blockly-based programming, it was essential to ensure that both modules worked together seamlessly. I verified that Guided Projects loaded the correct Blockly workspace, that users could modify and execute Blockly programs without errors, and that generated outputs matched the instructional objectives of each project. This testing helped ensure consistency across the platform and improved the learning experience for students using guided practical exercises.

During testing, I carefully observed the behavior of the simulator under different user interactions and execution scenarios. Whenever inconsistencies, unexpected outputs, interface issues, or execution errors were identified, I documented the observations and reported them. After corrective changes were implemented, I performed regression testing to verify that the reported issues had been resolved successfully and that the fixes did not introduce new problems into existing functionality. Repeated testing cycles significantly contributed to improving the overall quality, reliability, and maintainability of the Block Programming module.

Apart from verifying functional correctness, I also evaluated the usability of the Blockly interface from a learner's perspective. Since OpenHW-Studio is intended

for educational use, it is important that beginners can easily understand the available blocks, connect them correctly, and execute programs without unnecessary complexity. I verified that blocks could be dragged, connected, modified, deleted, and rearranged correctly within the workspace. I also checked that appropriate validation messages were displayed whenever users attempted invalid operations, helping maintain a smooth and user-friendly programming experience.

The testing activities performed during this internship strengthened my understanding of software quality assurance methodologies, functional testing, integration testing, regression testing, debugging, and verification of educational software systems. I gained practical experience in designing test scenarios, executing test cases, validating expected outputs, documenting software defects, reproducing reported issues, and confirming the effectiveness of implemented fixes. Working closely with mentors and developers also improved my ability to analyze software behavior systematically and identify the root causes of issues affecting application functionality.

Overall, testing the Block Programming module was one of the most significant aspects of my internship. It allowed me to work extensively with visual programming environments, embedded system simulations, hardware interaction, and educational software workflows. Through continuous testing, debugging, issue reporting, and validation, I contributed to improving the stability, accuracy, and reliability of the Blockly environment within OpenHW-Studio. This experience not only enhanced my technical knowledge of software testing and embedded systems but also provided valuable exposure to collaborative open-source software development and the importance of delivering high-quality educational software that offers a reliable and engaging learning experience for students.

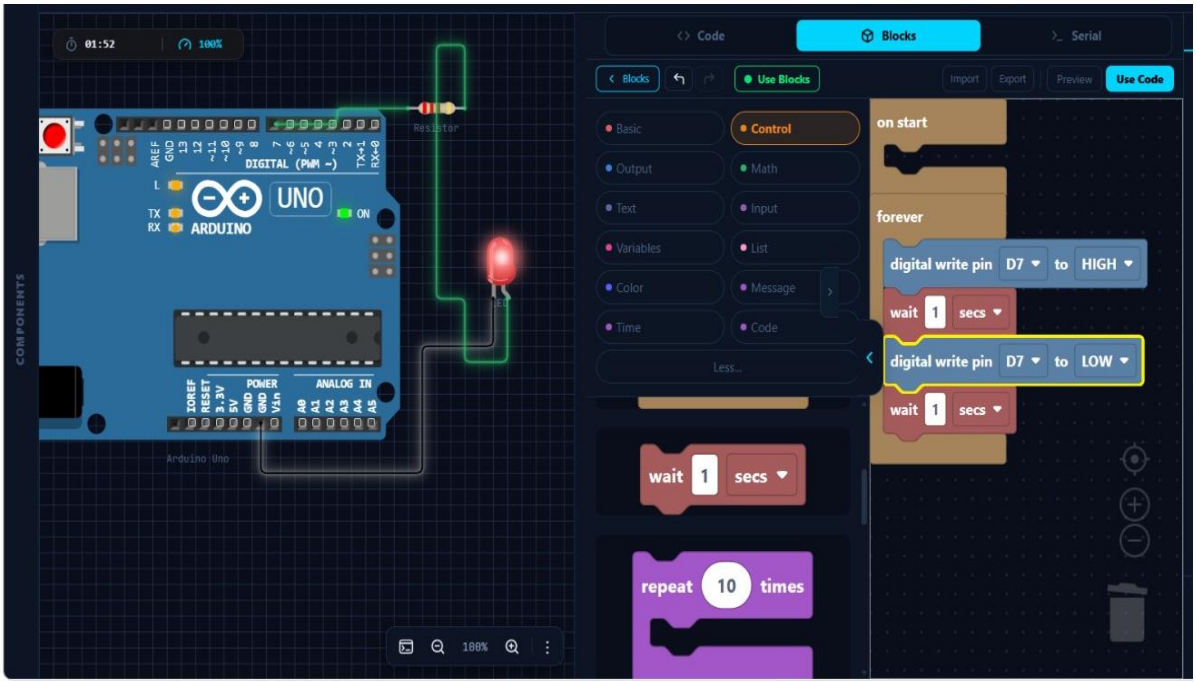

Figure 11: Block Programming Simulation - LED Control

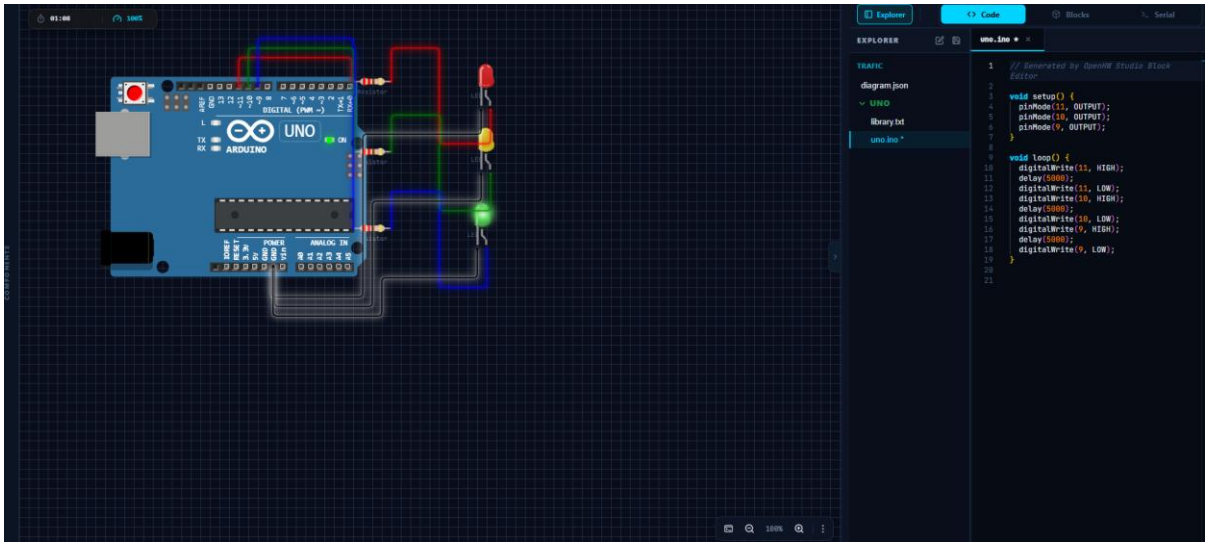

Figure 12: Traffic Light Simulation Using Block Programming (Blockly Code Visible)

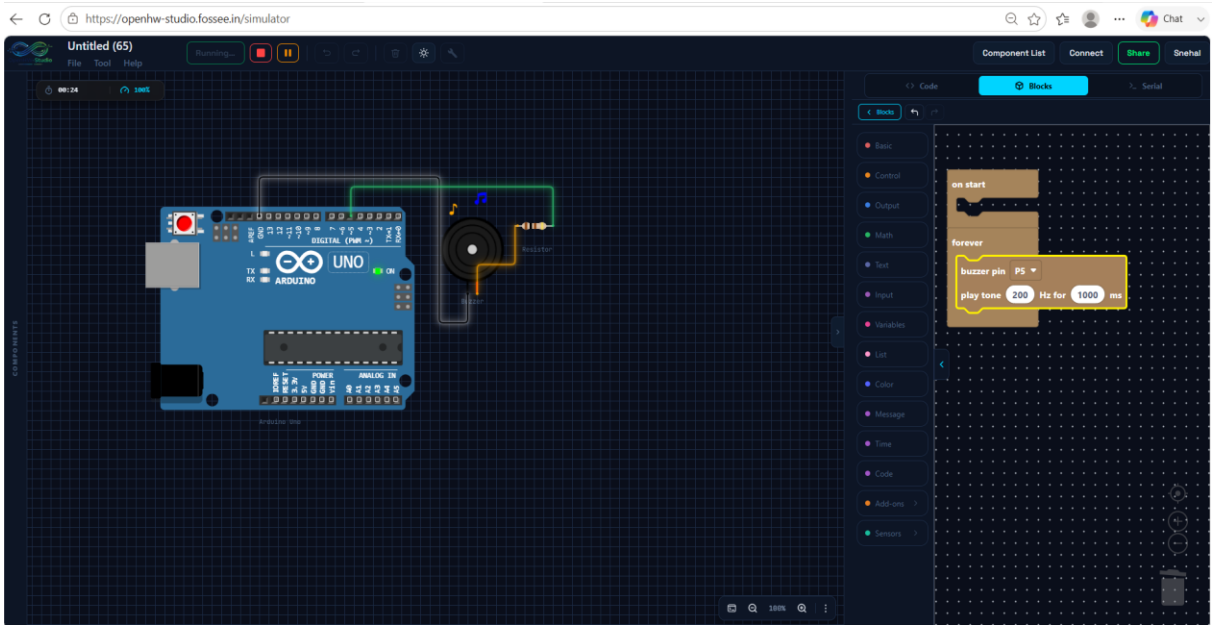

Figure 13: Buzzer Circuit Simulation with Blockly Workspace

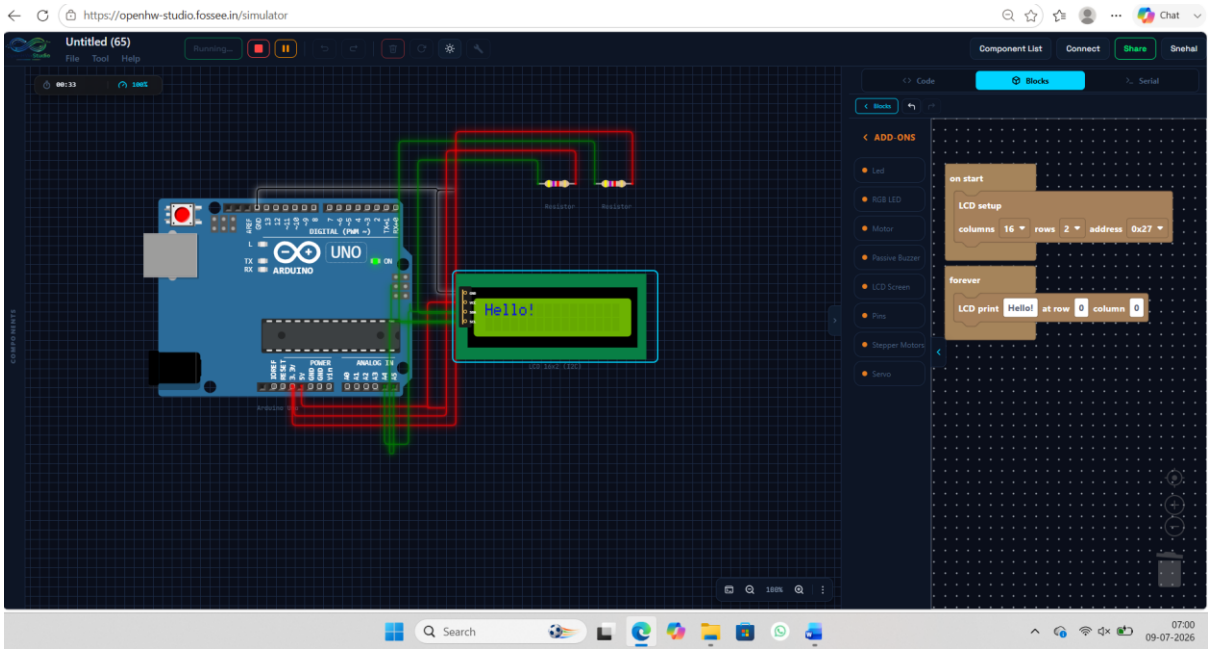

Figure 14: Block Programming Simulation - LCD Display

2.4 Summary

During my internship under the FOSSEE Project, IIT Bombay, I contributed to the backend development and testing of various platform modules. My work primarily focused on supporting the implementation of the user authentication system, contributing to the Guided Projects module, and performing functional testing of the Guided Projects and Block Programming features. I was involved in implementing parts of the login and password management functionalities, testing different user scenarios to ensure smooth operation, and validating that authentication workflows functioned correctly under various conditions. Additionally, I participated in testing various Guided Projects and Blockly programming blocks, identifying issues, verifying fixes, and helping improve the overall reliability and user experience of the platform. Beyond feature implementation, I actively performed functional, integration, and user-interface testing to ensure that newly developed modules interacted correctly with existing components of the platform. This involved executing multiple test cases, validating expected outputs, reproducing reported issues, documenting bugs, and verifying that fixes resolved the identified problems without introducing new defects. Through systematic testing, I contributed to improving the stability and usability of several platform features before their integration into the main development branch. I also collaborated with mentors and fellow contributors through the project's GitHub-based development workflow. My responsibilities included working on feature branches, committing code with meaningful commit messages, creating and reviewing pull requests, and resolving merge conflicts when necessary. This collaborative workflow helped me understand modern software development practices, version control strategies, and the importance of maintaining clean, well-documented, and maintainable code in a large open-source project. Working on OpenHW-Studio provided me with valuable exposure to the complete software development lifecycle, from feature implementation and debugging to testing and deployment. The internship strengthened my understanding of backend web development, authentication mechanisms, software quality assurance, debugging techniques, and collaborative open-source development. It also enhanced my analytical thinking, problem-solving abilities, and technical communication skills while giving me practical experience of contributing to a real-world educational software platform used for electronics simulation and embedded systems learning.

2.5 References

1. OpenHW-Studio — Live Platform: https://openhw-studio.fossee.in/
2. OpenHW-Studio — https://openhw-studio.fossee.in/about
3. OpenHW-Studio GitHub Repository (FOSSEE Project)- https://github.com/OpenHW-Studio
4. https://github.com/OpenHW-Studio/openhw-studio-backend
5. https://github.com/OpenHW-Studio/openhw-studio-examples

Chapter 3

Conclusion

3.1 Overview of Learning Experience

The internship under the FOSSEE, IIT Bombay, was a highly enriching learning experience that allowed me to bridge the gap between theoretical knowledge gained during my academic coursework and its practical application in a real-world software development environment. Throughout the internship, I had the opportunity to contribute to the development and testing of a large-scale, open-source educational platform designed for electronics simulation and embedded systems learning. Working on an actively developed project exposed me to industry-standard software engineering practices and provided valuable insights into the complete software development lifecycle, from feature implementation and testing to debugging, documentation, and collaborative development.

3.2 Contributions to the User Authentication Module

One of my major contributions was assisting in the implementation and testing of the User Authentication Module. Working on the login system, password setup process, and input validation helped me understand how secure authentication mechanisms are designed and integrated into modern web applications. I gained practical knowledge of handling user credentials, validating user inputs, managing authentication workflows, and ensuring that users receive appropriate feedback during different authentication scenarios. This experience strengthened my understanding of backend application logic and highlighted the importance of security, reliability, and usability in user account management systems.

3.3 Contributions to Guided Projects and Software Testing

Another significant aspect of my internship was contributing to the Guided Projects module and performing extensive testing of various platform features. I participated in verifying the functionality of multiple guided projects, ensuring that project instructions, code execution, and simulation workflows operated as expected. I also tested Blockly programming blocks across different categories, including control blocks, input/output blocks, mathematical operations, variables, sensors, and complete project integrations. Executing numerous test cases enabled me to identify defects, validate bug fixes, and improve the overall stability and quality of the platform. This experience enhanced my understanding of software testing methodologies, quality assurance practices, and systematic debugging techniques.

3.4 Open-Source Collaboration

The internship also introduced me to collaborative open-source software development using GitHub. I worked with feature branches, commits, pull requests, and code reviews while following structured development workflows. Collaborating with mentors and fellow contributors taught me the importance of version control, code organization, documentation, issue tracking, and effective communication within a software development team. These practices not only improved my technical proficiency but also helped me develop professional habits required in collaborative software engineering environments.

3.5 Technical and Professional Skills Gained

Beyond technical skills, the internship significantly improved my analytical thinking, problem-solving abilities, and attention to detail. Debugging software issues, validating platform functionality, reproducing reported bugs, and verifying solutions required a systematic and logical approach to problem solving. I learned the importance of thoroughly testing software under different user scenarios and ensuring that every implemented feature meets both functional and usability requirements before integration into the main project.

3.6 Understanding of OpenHW-Studio

Working on OpenHW-Studio also gave me a broader understanding of how modern educational technology platforms are designed and developed. I observed how multiple modules including authentication, circuit simulation, Blockly programming, guided learning, classroom management, project storage, and collaborative development work together to create a comprehensive learning ecosystem. This exposure broadened my understanding of scalable software architecture and demonstrated how different software components interact to deliver a seamless user experience.

3.7 Learning Outcomes

Overall, the OpenHW-Studio internship was an invaluable opportunity that enhanced both my technical and professional competencies. It strengthened my knowledge of backend development, authentication systems, software testing, debugging, feature integration, version control, and collaborative open-source development. More importantly, it increased my confidence in working on large-scale software projects and reinforced the importance of writing reliable, maintainable, and user-focused software. The knowledge and practical experience gained during this internship will serve as a strong foundation for my future academic projects, research work, and professional career in software engineering and embedded systems development.

Chapter 4

Future Work

OpenHW-Studio has significant potential to further enhance its Guided Projects module and provide a more engaging learning experience for students. Some possible future improvements include: Expanding the Guided Projects library by adding more beginner, intermediate, and advanced embedded systems experiments covering a wider range of sensors, actuators, and communication modules.