Iron Coder Design Draft

Julian Beloiu, Rebecca Diaz, Ethan Morandi, Eric Tobon, Josue Vicente
Oct. 8th 2024

Introduction

Purpose / Need

Iron Coder is an embedded systems IDE that combines elements of hardware development and programming in the Rust language. Iron Coder already has a strong foundation. Our team's goal is to improve the current features and add additional features to make it a more complete IDE. Improvements to the current Iron Coder will make it easier for students and entry-level developers to get into embedded systems development with Rust. Eventually, Iron Coder can be used as a valuable educational tool and lead to the wider adoption of Rust in embedded development.

Domain & Prior Art

Iron Coder is an Integrated Development Environment for embedded development in Rust with a primary focus on education and introducing newcomers to embedded Rust development. There are many other education-focused IDEs for embedded development such as Arduino, TinkerCad, Fritzing, and Wokwi.

Arduino IDE is an open source embedded IDE for C++ development with Arduino's microcontrollers. The IDE's main feature includes a code editor that allows users to load programs onto their boards [1]. In addition, Arduino IDE offers many example programs for newcomers to explore and is widely considered the pioneer for educational embedded development. Iron Coder differentiates itself from Arduino IDE by offering the chance to explore Rust embedded development rather than C++. With a number of resources available for embedded development in C++, Iron Coder will offer users an opportunity to learn a new language for embedded development and be part of a growing community.

TinkerCad is an open source 3D design editor for helping beginners learn about electronics, coding, and object modeling. Users can connect basic components and sensors to a microcontroller and simulate C++ code with their design [2]. Unlike Arduino's IDE, TinkerCad and Iron Coder share the capability of offering a hardware design and schematic editor. However, while TinkerCad offers 3D modeling and printing capabilities, Iron Coder focuses solely on introducing beginners to electronics and coding. Additionally, Iron Coder will expose users to Rust development rather than C++.

Fritzing streamlines the printed circuit board development process [3]. By starting with their schematic editor, users get the opportunity to prototype hardware designs, create schematics, run code, and develop PCBs from their designs [4]. Although proposed elements of Iron Coder take inspiration from their more traditional schematic editor, Iron Coder will not focus on PCB development. Instead, Iron Coder aims to provide a fun introduction to Rust embedded development. Users will learn the Rust programming language as well as microprocessor application fundamentals with Iron Coder. This is done by providing example code for sample projects and the generation of boilerplate code from hardware designs.

Unlike many of the other embedded IDEs, Wokwi has Rust and MicroPython development options [5]. As an open source embedded Rust development application that offers a schematic editor, code editor, and simulation, it shares many similarities with Iron Coder. However, Wokwi has a limited variety in microcontroller options with only 4 families of boards being supported according to the site's documentation [6]. Iron Coder will offer a number of different microcontrollers options. Users will also have the ability to add their own using Rust library crates and toml files, and explore a variety of different microcontrollers.

Impact & Risk Assessment

One impact of Iron Coder is that as an open-source software and hardware platform, it will contribute to the development of new ideas by allowing users to take the source material and add their own development to it. This can serve as a valuable educational tool for those that are new to embedded development with Rust. It can also eventually lead to the wider adoption of Rust in an embedded systems context. As an IDE for embedded systems, Iron Coder gives users the freedom to design any embedded device that Iron Coder supports. While these systems will mainly help people learn more about embedded development with Rust, there is a risk that systems could be designed to cause harm to people. It is ultimately up to users to determine what their overall system will accomplish when designing. Granted, there is risk involved with allowing anyone to upload any type of board or support certain features that may lead to harmful designs. This will need to be accounted for through the implementation of the board loading feature.

Statement of Work

Core Feature	Target Date for Completion	Person Responsible	Description	Requirements + time
Example Code for Blinking LED	Nov 20, 2024	Eric	Example code for blinking LED that can be preloaded into the IDE and simulated	Rust egui rustc and cargo rust_gpiozero 15 hours
Example Code for LCD Screen	Nov 20, 2024	Ethan	Example code for an LCD screen that can be preloaded into the IDE and simulated	Rust rustc and cargo 15 hours
Example Code for Alarm Clock	Nov 20, 2024	Rebecca	Example code for alarm system that can be preloaded into the IDE and simulated	Rust Egui rustc and cargo symphonia 15 hours
Automation of Adding New Boards	Feb 24, 2025	Rebecca Ethan	Add functionality that allow users to quickly add new microcontrollers to the list of boards compatible for Iron Coder	Rust egui rustc and cargo toml image 60 hours
Debugger	Feb 24, 2025	Ethan Eric	Embedded system debugger features would include watch	Rust rustc and cargo egui headcrab

			windows for register and variables, break points, manual stepping.	40 hours
Simulator	Feb 24, 2025	Eric	Allow users to run their code on the board and view graphical outputs of components (ex. blinking LEDs)	Rust rustc and cargo egui embedded-graphics crate 40 hours
Online Forum	Feb 24, 2025	Julian	Create an online forum that will serve as a place to share projects that were completed with Iron Coder	Rust PostgreSQL Yew.rs SQLX Docker 60 hours

Secondary Task	**Target Date for Completion**	**Person Responsible**	**Description**	**Requirements + time**
Add New Basic Components	Nov 20, 2024	Rebecca	Add basic components such as LEDs, speakers, switches, etc	Rust rustc and cargo toml image egui 20 hours
CLI	Dec 2, 2024	Ethan	Current CLI is read only, making it fully	Rust egui rustc and

			usable within the IDE.	cargo
Wire GUI Enhancements	Nov 20, 2024	Ethan	Changing wire to use different curve types away from bezier curves.	Rust egui rustc and cargo 5 hours
Fix Functional Bugs	Nov 20, 2024	Julian	Making IronCoder better to use, and fixing existing bugs.	Rust rustc and cargo
Quick Start Documentation	Feb 24, 2025	Rebecca	Tutorial for new users and developers that demonstrate how to install, build, and use the application (including documentation for new features)	Markdown 5 hours
Enhance Documentation of Compatible Boards	Nov 20, 2024	Julian	Enhance the current viewable documentation of the boards within the application	Rust rustc and cargo
Schematic View	Feb 24, 2025	Eric	Create a minimalistic schematic view of the user's working design	Rust rustc and cargo Graphvis toml 60 hours

Deliverable Artifacts

Example Code

Description

Example code used to display the functionality of IronCoder. Demonstrating the simulator, debugger, and wire improvements.

Dissemination Plan

The example code will be displayed as a default code that can be used as a template. The user will be able to load it up to simulate and if the user wishes they can debug the code. This would allow them to practice with the code and IDE.

Accessibility / Maintenance Plan

Example code can be expanded upon to demonstrate different components that are not currently usable in IronCoder. As the project goes along, there could be different code examples that are made for the specific board types that are being added as well.

Simulator and Debugger

Description

Code simulator where users can input code and run it on the hardware viewer, with the ability to view registers and variable states as the program runs.

Dissemination Plan

The simulation environment will have a code view on one screen with an option within the code editor to run a hardware simulation. The registers, variables, and other information will be available in a collapsible side window on the hardware viewer.

Accessibility / Maintenance Plan

The simulation and debugger allow users to understand how their code functions on hardware systems. The simulation feature could be expanded in the future to perform real-time simulations in sync with updates to the code, and more relevant information can be added to the debugger.

Online Forum

Description

The online forum will serve as a place to share projects that were completed with Iron Coder. It will also give users a chance to ask questions, respond to questions, browse documentation, and more.

Dissemination Plan

The forum will be a website that is hosted on the internet. Actix, PostgreSQL, and Docker will be used for the backend API and Yew.rs will be used for the frontend.

Accessibility / Maintenance Plan

The website will have to be user friendly, protect user data, and have some form of flagging mechanism for potentially inappropriate content. There will need to be a set of guidelines for contributing to the forum. Also, there will need to be moderators that enforce the contribution guidelines and decide what to do with flagged posts.

Feature that Automates Adding New Boards

Description

Add a menu selection to the “add board” menu that allows the user add a new board to the list of compatible boards within Iron Coder. This will streamline the existing process of creating the board.toml file as well as adding the required crates.

Dissemination Plan

The feature will be added to the existing add board menu. The user will input information about their new board for the automation script to reference.

Accessibility / Maintenance Plan

The feature will be designed to accept the addition of a variety of different boards. This would allow users to contribute different hardware platforms to IronCoder, making more hardware available within the hardware editor.

Documentation of Compatible Boards

Description

Within the hardware editor of Iron Coder improve the existing documentation by adding manuals and improving the description of pin functions for boards and peripherals. Having direct access to this documentation creates a seamless development environment that can meet all embedded system design needs.

Dissemination Plan

The manuals for peripherals or boards would be linked directly to the device within the hardware editor and would be opened by clicking the link from the right click option menu. Pinouts are already included but would be improved to be easily readable.

Accessibility / Maintenance Plan

Any updates to manuals would easily be replaceable within the code base, since the manuals would simply be linked to the online version of a devices manual. Pinouts would need to be checked if they are ever changed, but since it is physically linked to the hardware there are not many cases where it would need to be changed for a specific board.

Schematic View

Description

Schematic view of the hardware designs, providing a complete look into how the hardware has been built and assembled within the hardware editor. This could make debugging hardware designs easier and provide a more structured view of the hardware

Dissemination Plan

The schematic view would be an option available within the hardware editor screen of IronCoder. It would load a new window with a schematic of the design using information from the hardware editor.

Accessibility / Maintenance Plan

The schematic viewer could be further developed to provide more formal schematics similar to what is seen for real hardware products. The schematic viewer could help simplify the development of more complex hardware systems within the IronCoder environment.

Iron Coder Quickstart Documentation

Description

Tutorial for new users to get started using IronCoder and become familiar with the different processes available on the platform. This will present users with a walkthrough of IronCoder’s features as well as how to install and build the application.

Dissemination Plan

The tutorial will be available in the menu of IronCoder and will be presented within a new window. The tutorial pages will contain text explanations, images, and navigation to locate specific features.

Accessibility / Maintenance Plan

This documentation will help get new users up to speed with using IronCoder. The tutorial documentation could be developed in the future to be a fully functional walkthrough of IronCoder’s features, allowing users to interact with the IronCoder development suite while learning about how to use it.

Iron Coder Enhancements and Bug Fixes

Description

There are bugs with Iron Coder that need fixing. We would also like to create a generally upgraded user experience for Iron Coder. Some examples include the red box that appears around boards while moving them and wires being displayed on top of the board menu when it is open.

Dissemination Plan

Bugs will be addressed on a per case basis depending on the level of impact that they have on IronCoder’s functionality. There is no standardized way that these changes will look as each bug will be unique.

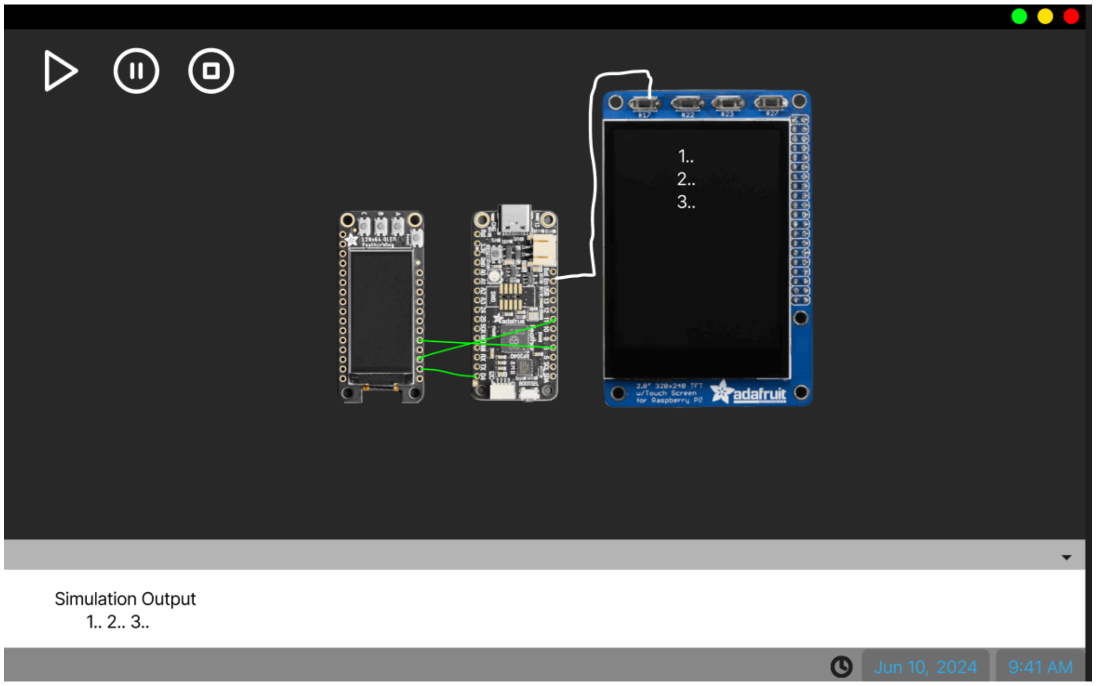
Accessibility / Maintenance Plan

Fixing bugs and making user experience improvements will draw more people to use IronCoder. Bugs will need to be regularly checked for and addressed, as is normal with most software projects.

Mockups

Interfaces

Simulator

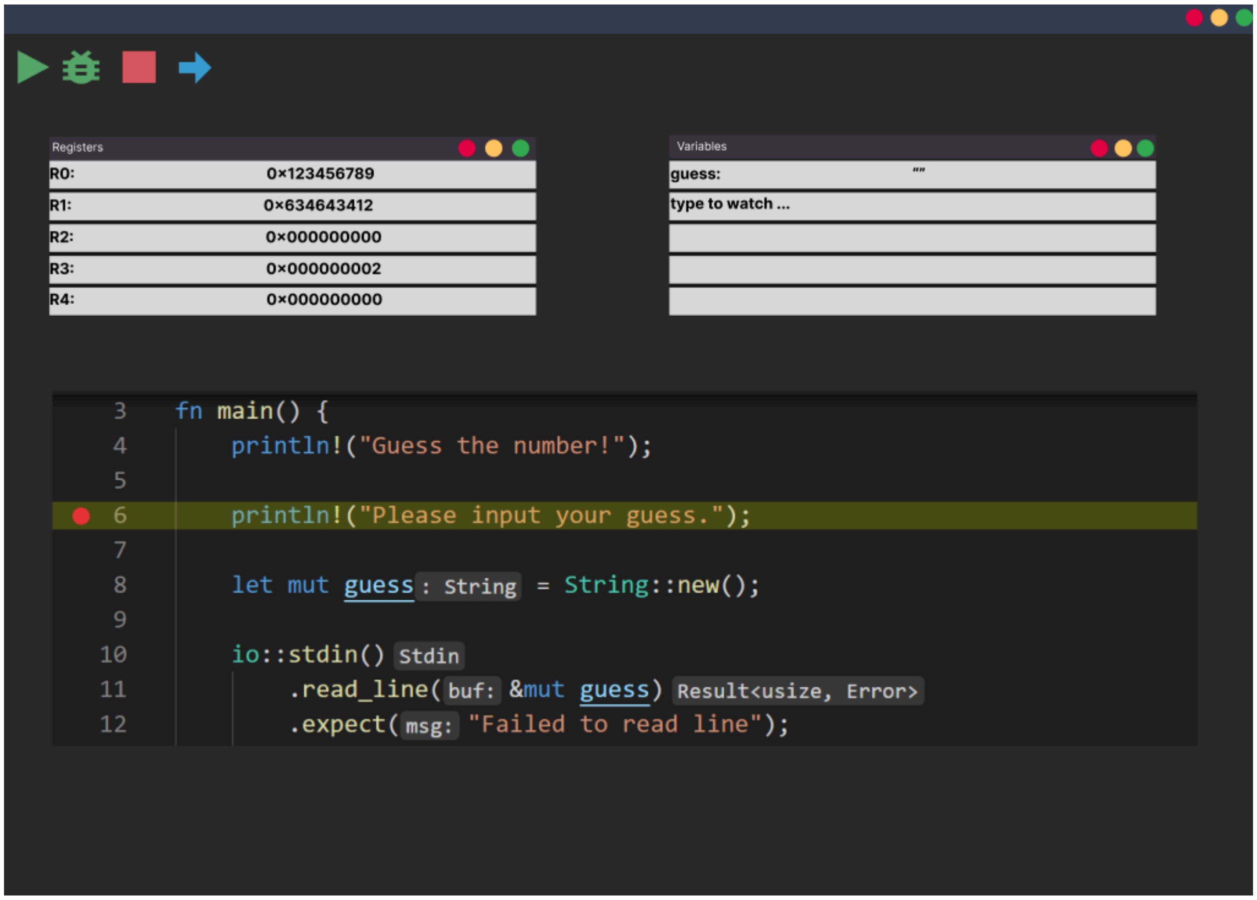


Simulation Output
1.. 2.. 3..

Features:

- Simulation Output Window: Allows user to simulate and see logs as they are output for testing
- Play button: Will give the user the ability to begin or restart the simulation
- Pause button: Allows users to pause the simulation to not end the program but to halt the program temporarily.
- Stop Button: Users should be able to immediately stop the simulation especially if the program is not functioning properly.
- Timer
- Real Time Simulation: Components in the simulation should be able to output as they would outside of simulation. Such as an output to the screen.
- Time/Timer: The user should be able to see how long the program is running or other data to track usage.

Debugger


Registers

R0:	0x123456789
R1:	0x634643412
R2:	0x000000000
R3:	0x000000002
R4:	0x000000000

Variables

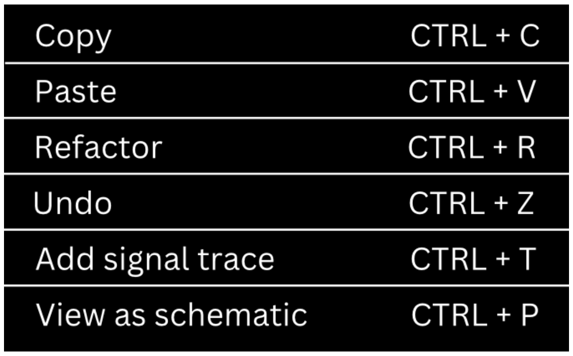
guess:	""
type to watch ...	

```
fn main() {
    println!("Guess the number!");

    println!("Please input your guess.");

    let mut guess: String = String::new();

    io::stdin() Stdin
        .read_line(buf: &mut guess) Result<usize, Error>
        .expect(msg: "Failed to read line");
```

Features:

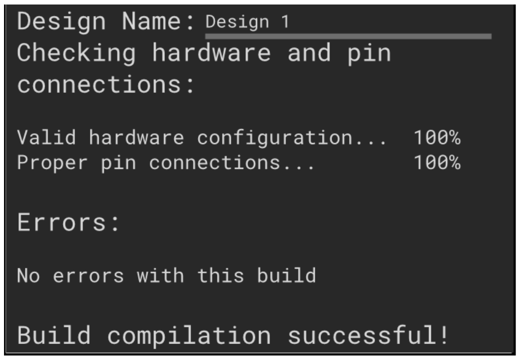
- Breakpoints: Stops execution at a certain line of code, can be placed at any line within the program.
- Register window: Shows current register contents pertaining to specific board the project is targeting.
- Variable window: Shows current variables and their values, any variable name can be typed in to watch.
- Step button (blue arrow): Allows for single step execution of the program
- Debug button (green bug): Starts debugging of program.
- Stop button (red square): Stops debugging execution.
- Continue button (green triangle): Runs program to next breakpoint or till end of execution.

Schematic View



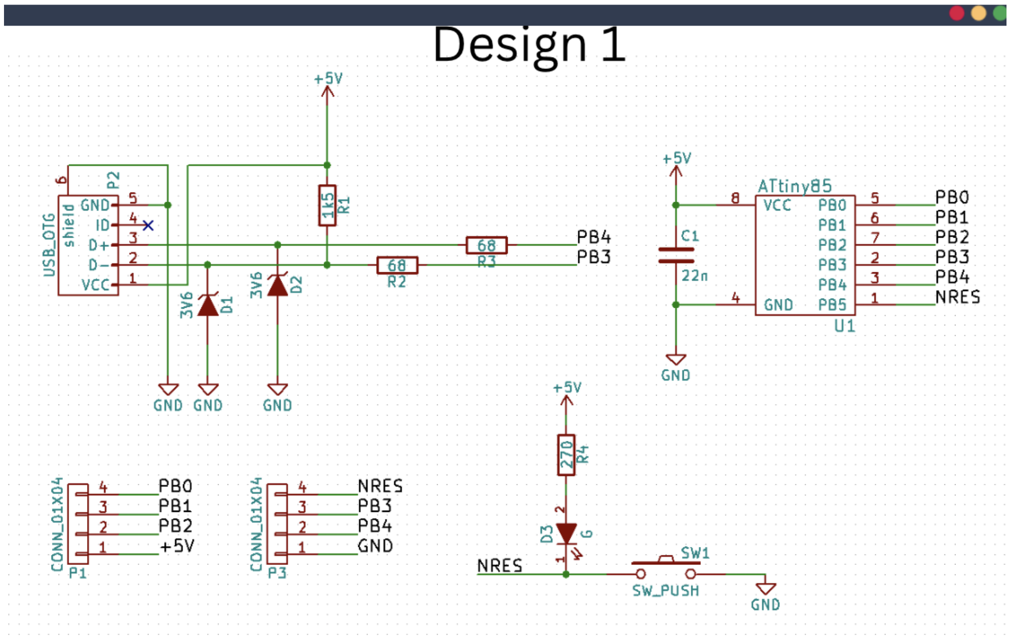
Copy	CTRL + C
Paste	CTRL + V
Refactor	CTRL + R
Undo	CTRL + Z
Add signal trace	CTRL + T
View as schematic	CTRL + P

Menu



```
Design Name: Design 1
Checking hardware and pin
connections:

Valid hardware configuration...  100%
Proper pin connections...        100%

Errors:

No errors with this build

Build compilation successful!
```

Schematic Generator Log

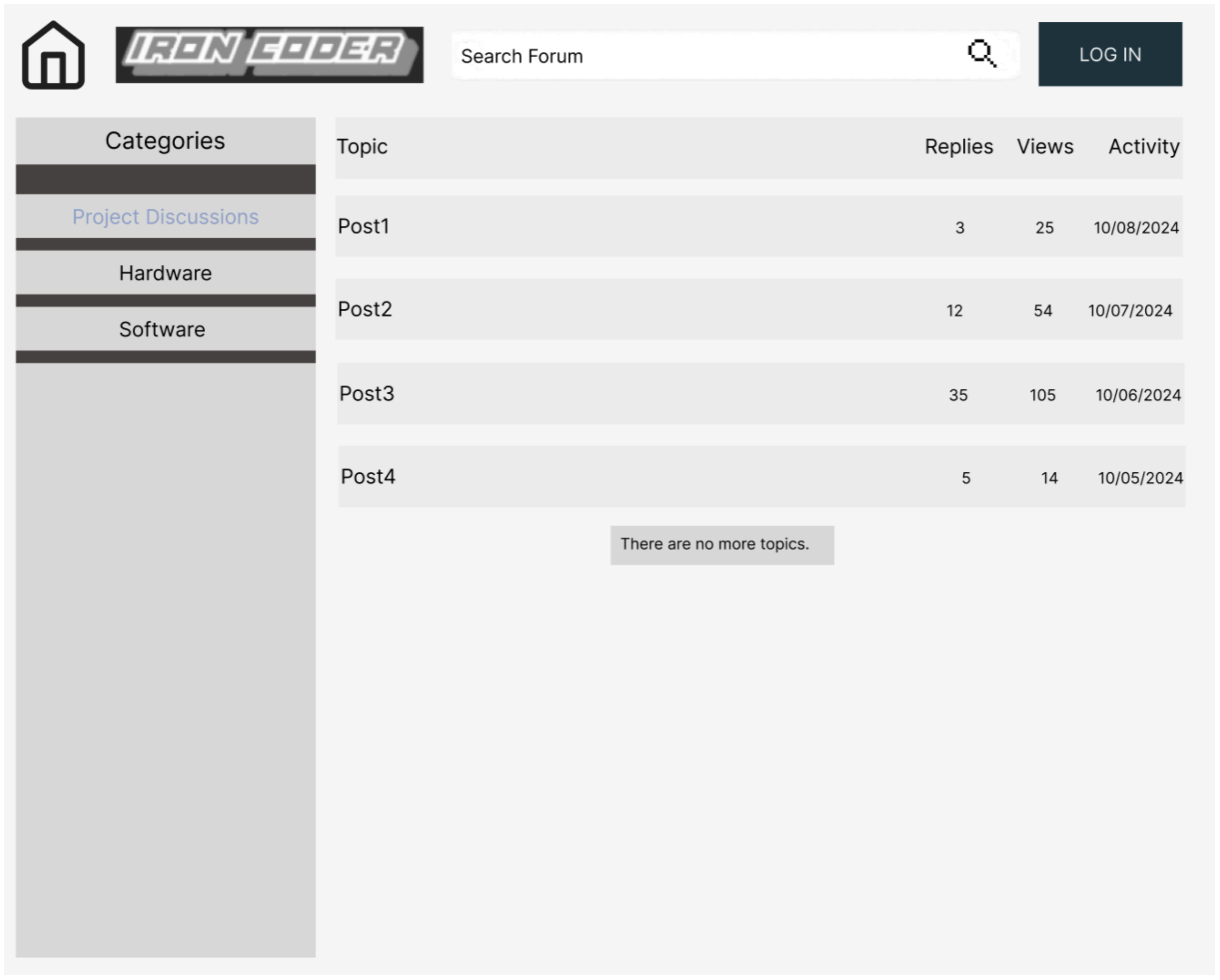


Example Schematic View

Features:

- User can generate a non-editable Schematic View of user's current design inside the Iron Coder editor
- Schematic View will consist of block diagrams for all components on the board along with wire connections
- Users can troubleshoot their design with the error messages in the schematic generation log
- User will generate the schematic view through the "right-click" menu
- User can exit the Schematic View by closing the window

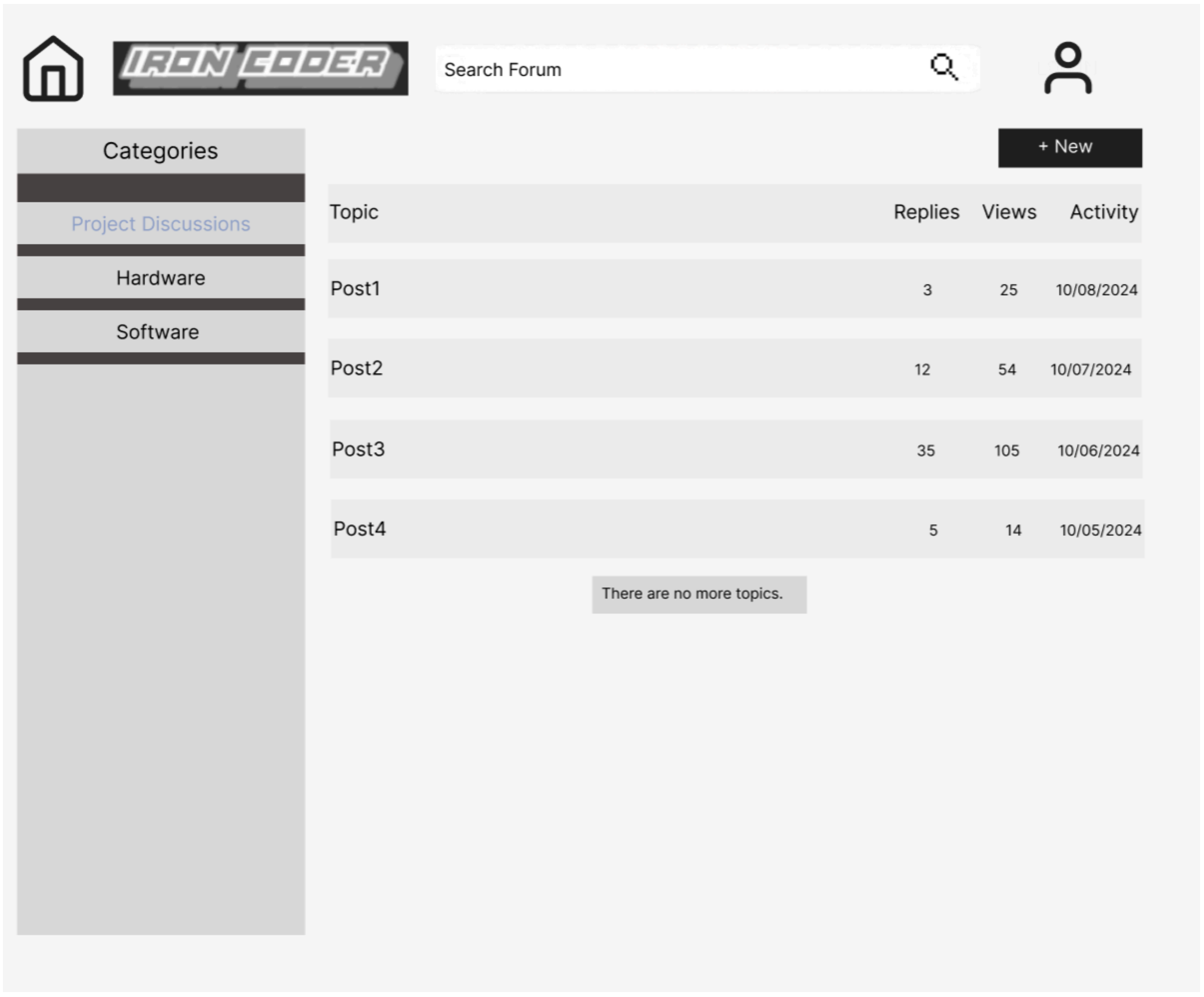
Online Forum (not logged)



Topic	Replies	Views	Activity
Post1	3	25	10/08/2024
Post2	12	54	10/07/2024
Post3	35	105	10/06/2024
Post4	5	14	10/05/2024

There are no more topics.

Online Forum (logged in)



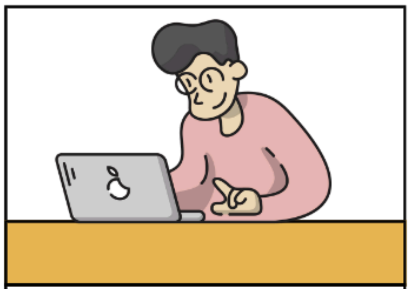
Topic	Replies	Views	Activity
Post1	3	25	10/08/2024
Post2	12	54	10/07/2024
Post3	35	105	10/06/2024
Post4	5	14	10/05/2024

Features:

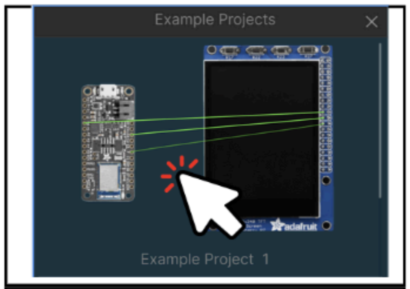
- When not logged in, you can only view the forum posts.
- When logged in, you have the option to create posts
- Left side has the forum categories.
- Search bar on top.

Storyboards

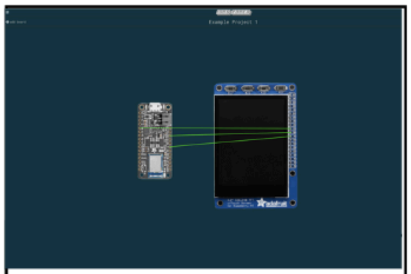
LOADING EXAMPLE PROJECTS



James wants to load an example project.



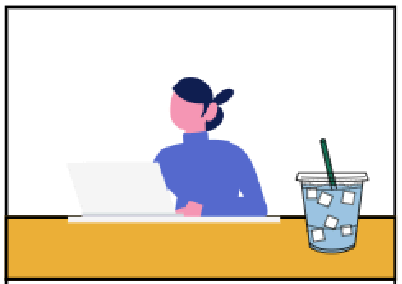
He can navigate to the “Example Projects” tab and select an example project



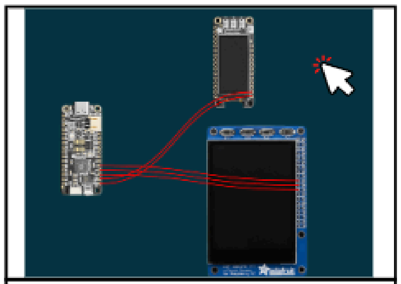
The example project will load in the hardware viewer

The code for the example project will appear after clicking “start development”

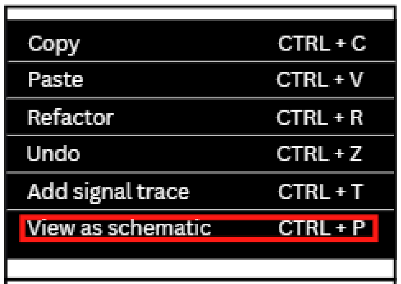
VIEWING HARDWARE SCHEMATICS



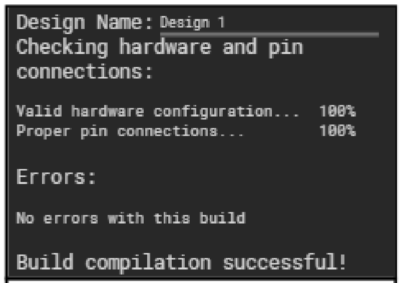
Rebecca wants to view the design that she created within the hardware editor as a schematic



She can right click within the hardware editor interface

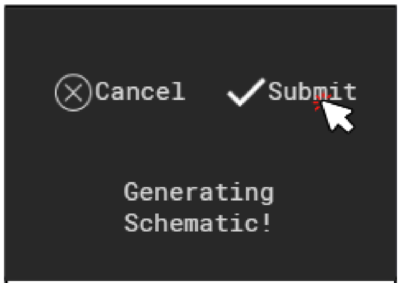


This will bring up a menu of options, with a schematic view listed as one of the options

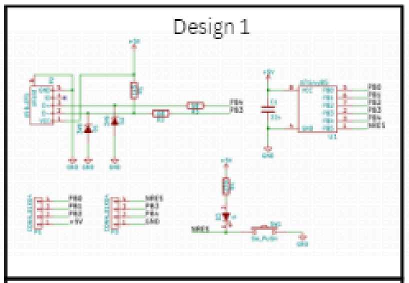


```
Design Name: Design 1
Checking hardware and pin
connections:

Valid hardware configuration...   100%
Proper pin connections...         100%

Errors:

No errors with this build

Build compilation successful!
```

Rebecca can view the progress of her design being converted to a schematic, with any errors being listed on this page

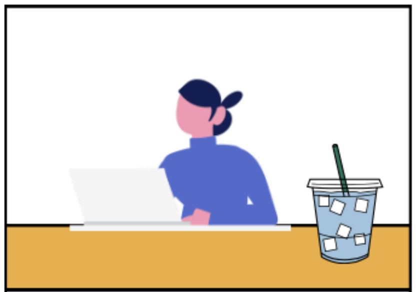


After confirming, the schematic will be generated. Clicking cancel will return to the hardware editor.

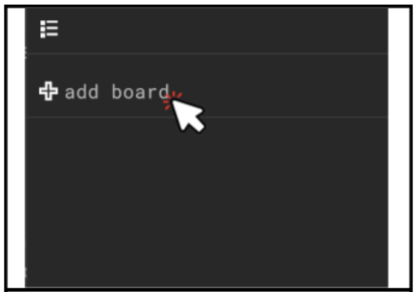


The schematic will appear in a new window once it is generated.

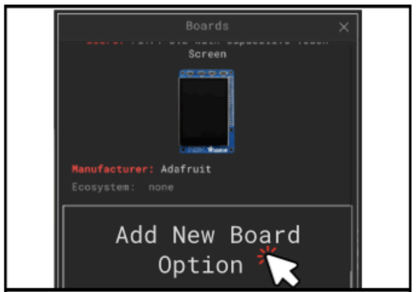
ADDING NEW BOARDS



Rebecca wants to work with a microcontroller that is not currently listed in the add board menu.

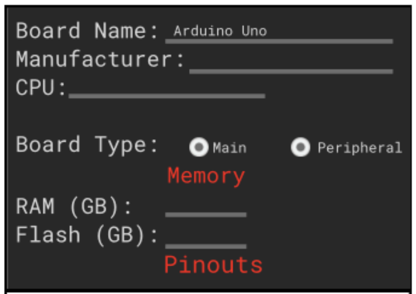


She can click on the add board option at the top left corner of the page.



Manufacturer: Adafruit
Ecosystem: none

A menu will appear. By scrolling down to the bottom of the menu, she can click on the "Add New Board Option."



Board Name: Arduino Uno
Manufacturer:
CPU:

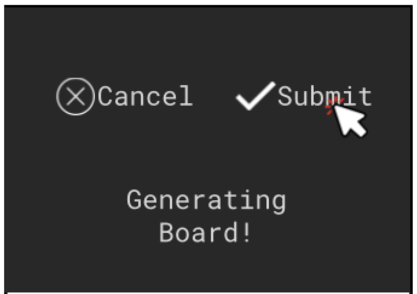
Board Type: ○ Main ○ Peripheral

Memory

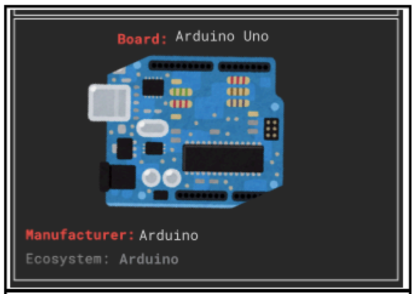
RAM (GB):
Flash (GB):

Pinouts

Rebecca can fill out the following form with all necessary information as well as upload a photo of the board.



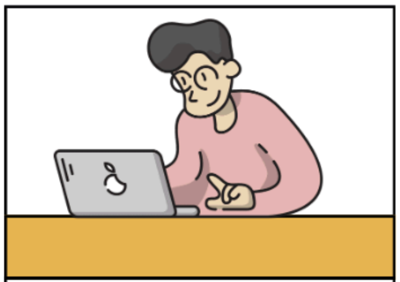
After, clicking submit, the board files will be generated. Clicking cancel will return to the menu. Any input errors are addressed here as well.



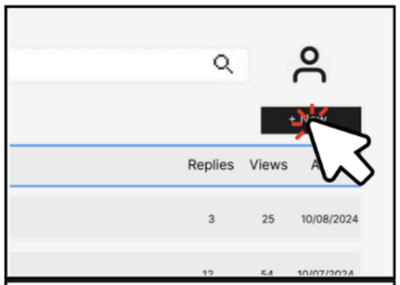
Board: Arduino Uno
Manufacturer: Arduino
Ecosystem: Arduino

The board will appear in the menu once it is generated.

CREATING A POST ON FORUM

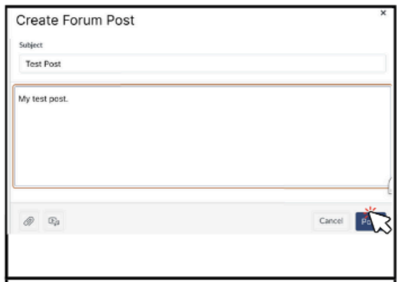


James wants to create a post on the Iron Coder forum.



Replies	Views	A
3	25	10/08/2024
12	54	10/07/2024

He can click on + New when logged in to his Iron Coder forum account

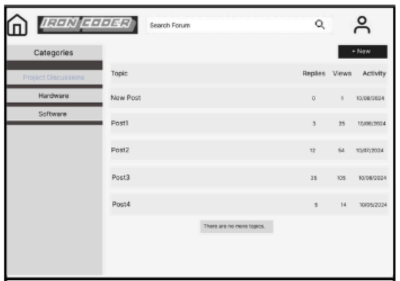


Create Forum Post

Subject: Test Post

My test post.

He can write a Subject, Post Body, and attach files before posting.



Topic	Replies	Views	Activity
New Post	0	1	10/08/2024
Post1	3	25	10/08/2024
Post2	12	54	10/07/2024
Post3	35	105	10/06/2024
Post4	5	14	10/05/2024

There are no more topics.

The new post will appear at the top of the page.

References

[1] What is Arduino Software (IDE), and how use it ? (2024) AndProf. Available at: https://andprof.com/tools/what-is-arduino-software-ide-and-how-use-it/ (Accessed: 07 October 2024).

[2] 10 reasons to use tinkercad (2024) All3DP. Available at: https://all3dp.com/2/what-are-the-pros-of-using-tinkercad/ (Accessed: 07 October 2024).

[3] What is fritzing, how does it work and how to use it? (2023) *Soldered*. Available at: https://soldered.com/learn/what-is-fritzing-how-does-it-work-and-how-to-use-it/#:~:text=Fritzing%20Fab%20allows%20us%20to%20create%20plates%20designed%20by%20Fritzing.&text=This%20option%20allows%20you%20to,the%20board%20and%20COM%20port. (Accessed: 07 October 2024).

[4] Fritzing - Electronics Made Easy (no date) *Fritzing*. Available at: https://fritzing.org/about/context (Accessed: 07 October 2024).

[5] "Wokwi," Woki, [Online]. Available: https://wokwi.com/. [Accessed 08 October 2024].

[6] Supported hardware | *wokwi docs* (2023) *Wokwi Docs*. Available at: https://docs.wokwi.com/getting-started/supported-hardware (Accessed: 08 October 2024).