Summer Fellowship Report

On

Electroblocks(Arduino) - Bug fixing and Further Improvements

Submitted by

Jyoti Kumari

Pranav Prakash Ranjan

National Institute of Technology Rourkela

Under the guidance of

Prof.Kannan M. Moudgalya

Chemical Engineering Department

IIT Bombay

Rajesh Kushalkar

Sr. Manager Open Source Hardware,

IIT BOMBAY

Noah Glaser

Pratik Bhosale

Research Associate,

IIT Bombay

July 2024

Acknowledgement

I would like to take this opportunity to express my heartfelt gratitude to everyone who played a pivotal role in making my summer internship with FOSSEE - Electroblocks (Arduino) an enriching and memorable experience.

First and foremost, I extend my deepest thanks to Noah Glaser, Prof. Rajesh Kumar and Prof. Pratik Bhosale for believing in my abilities and selecting me for this invaluable project. Their unwavering support and constant encouragement have been the driving force behind my progress.

I owe a great debt of gratitude to my mentors, whose guidance and wisdom have been indispensable throughout my internship. Their expert advice and timely assistance not only helped me complete my tasks successfully but also sharpened both my technical and non-technical skills.

I also want to express my appreciation to my colleagues working on various projects. Their constructive feedback and critical insights have significantly contributed to my growth and development.

Pranav Prakash Ranjan

Jyoti Kumari

Declaration

I declare that this written submission represents our ideas in my own words and whenever other's ideas or words 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 thesis. I also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken when needed.

Pranav Prakash Ranjan

Jyoti Kumari

Index

List Of Figures

Chapter 1

Introduction

Electroblocks is an innovative platform that simplifies learning and experimenting with electronics and programming using Arduino. It offers modular, plug-and-play components that can be easily connected, eliminating complex wiring. Compatible with the Arduino ecosystem, users can program projects with the Arduino IDE. Its user-friendly interface and clear labeling make it accessible for all skill levels, enhancing understanding of electronics and coding. Ideal for STEM education, Electroblocks supports a wide range of projects, from simple to complex. By combining Arduino's flexibility with modular simplicity, Electroblocks empowers creativity and makes prototyping straightforward for beginners and experienced makers alike.

1.1 Project Overview

During this fellowship, I worked on the existing project which was already completed to some extent. For the most part, my main job was to find and fix any bugs or errors in the project and to add the new features that were needed.

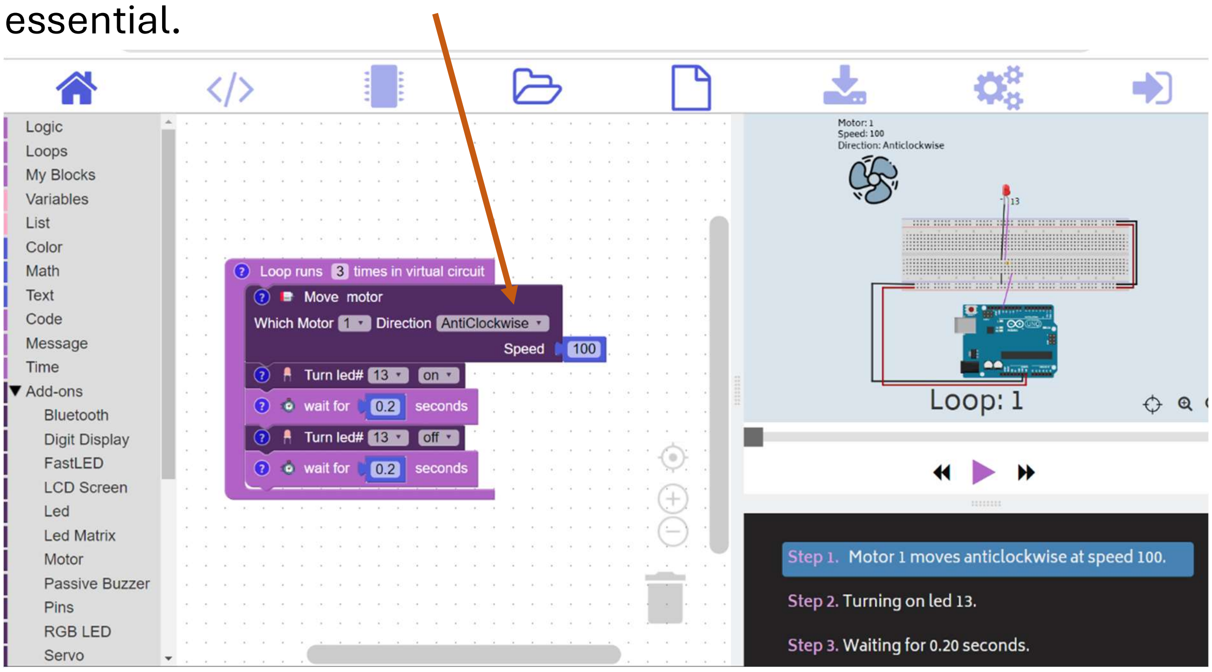
Chapter 2

Feature Additions

During the internship, I added a few features to the project, most of these features aimed at improving the accessibility of the website and facilitate some stuff in the website.

2.1 Changed from Forward backward to clockwise and anti-clockwise.

When you change the control scheme from forward/backward movement to clockwise/anticlockwise rotation, it alters how users interact with the system. This change could be for a robotic system, a user interface element, or any application where directional control is essential.



```
Loop runs 3 times in virtual circuit
    Move motor
    Which Motor 1 Direction AntiClockwise
                               Speed 100
    Turn led# 13 on
    wait for 0.2 seconds
    Turn led# 13 off
    wait for 0.2 seconds
```

```
Motor: 1
Speed: 100
Direction: Anticlockwise
```

Step 1. Motor 1 moves anticlockwise at speed 100.

Step 2. Turning on led 13.

Step 3. Waiting for 0.20 seconds.

Fig 2.1

Here's how to handle this transition effectively:

Understanding the Control Change

Forward/Backward:

- Forward: Moves the entity in a straight line ahead.
- Backward: Moves the entity in a straight line behind.

Clockwise/Anticlockwise:

- Clockwise: Rotates the entity to the right.
- Anticlockwise: Rotates the entity to the left.

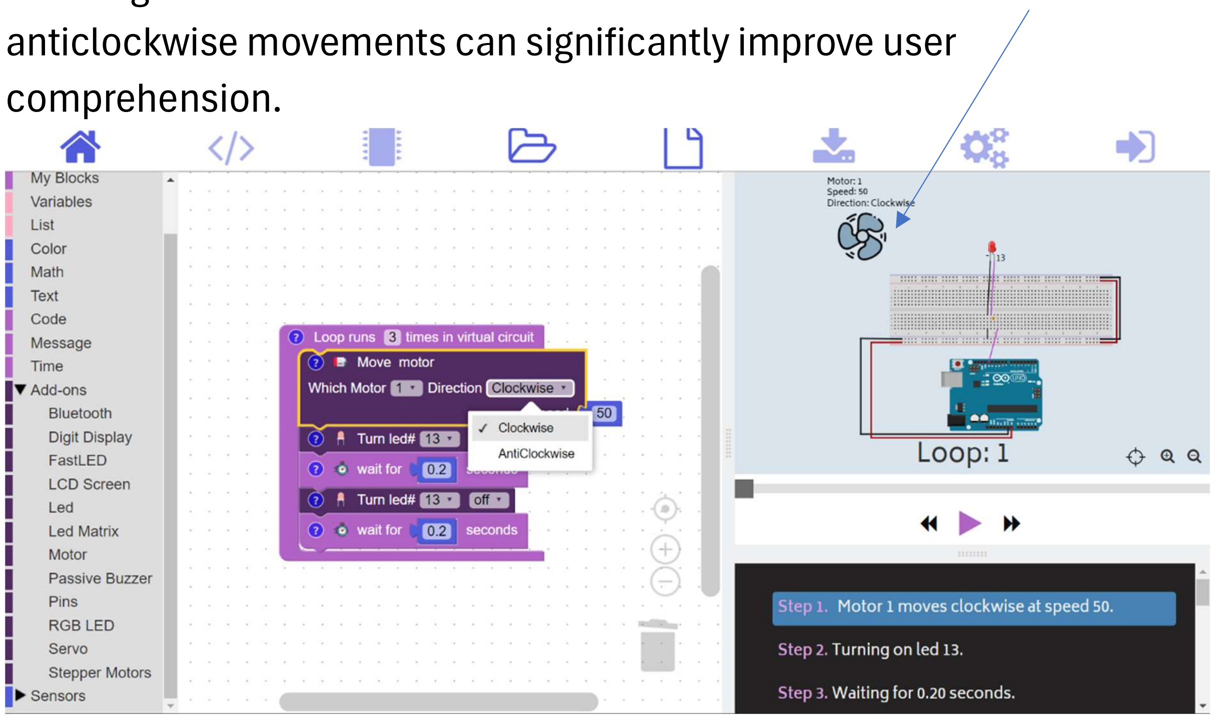
Steps to Implement the Control Change

1. **Update Control Inputs:**
    - Replace forward and backward commands with clockwise and anticlockwise commands.
    - Ensure that the input method (e.g., buttons, joystick, keyboard) corresponds to the new rotational commands.
2. **Adjust the Logic:**
    - Modify the underlying logic to handle rotational movement instead of linear movement.
    - For instance, if you are controlling a robot, change the motor control commands to rotate the robot instead of moving it forward or backward.
3. **Recalibrate Movements:**
    - If applicable, calibrate the speed and degree of rotation for the clockwise and anticlockwise commands to ensure smooth operation.
    - Determine the appropriate units of rotation (e.g., degrees per second) for precise control.
4. **Update the User Interface:**
    - Change any visual indicators or instructions from forward/backward to clockwise/anticlockwise.
    - Ensure that users can easily understand and adapt to the new control scheme.
5. **Test and Validate:**
    - Conduct thorough testing to ensure the new controls work as expected.
    - Gather user feedback to identify any issues or areas of confusion with the new controls.

By following these steps, you can smoothly transition from forward/backward controls to clockwise/anticlockwise controls, ensuring that users can effectively and intuitively interact with the system.

2.2 Created animation, to show the movement in both directions.

Creating an animation to demonstrate both clockwise and anticlockwise movements can significantly improve user comprehension.



```
Loop runs 3 times in virtual circuit
    Move motor
    Which Motor 1 Direction Clockwise 50
    Turn led# 13
    wait for 0.2 seconds
    Turn led# 13 off
    wait for 0.2 seconds
```

Motor: 1
Speed: 50
Direction: Clockwise

Loop: 1

```
Step 1. Motor 1 moves clockwise at speed 50.
Step 2. Turning on led 13.
Step 3. Waiting for 0.20 seconds.
```

Fig 2.2

Here’s how you can effectively plan and implement this concept.

Objectives:

- Visually demonstrate the difference between clockwise and anticlockwise rotations.
- Make the animation intuitive and easy to understand for users.

Key Elements:

1. **Rotating Object:** Choose a clear and easily recognizable object such as an arrow, gear, or dial to represent rotation.
2. **Direction Indicators:** Include markers or labels that indicate "Clockwise" and "Anticlockwise."
3. **Animation Phases:**
    - **Initial Phase:** Start with the object in a neutral, stationary position.
    - **Clockwise Rotation:** Animate the object rotating clockwise for a few seconds.
    - **Return to Neutral:** Bring the object back to the neutral position.

    - **Anticlockwise Rotation:** Animate the object rotating anticlockwise for a few seconds.
    - **Loop:** Optionally, loop the animation for continuous demonstration.

Implementation Steps:

1. **Design the Visuals:**
    - Use a design tool to create the rotating object and any directional markers or labels.
    - Ensure the design is simple and the rotation direction is clearly indicated.
2. **Animation Planning:**
    - **Clockwise Rotation:** Plan for the object to rotate to the right. This can be visualized as the top of the object moving to the right and downwards.
    - **Anticlockwise Rotation:** Plan for the object to rotate to the left. This is the opposite direction, with the top of the object moving to the left and downwards.
    - **Neutral Position:** Design a starting and resting position for the object where no rotation is happening.
3. **User Interface:**
    - Integrate the animation into your user interface in a prominent location where users can easily see and understand it.
    - Ensure there are clear buttons or triggers for starting each type of rotation (e.g., "Start Clockwise" and "Start Anticlockwise").
4. **Testing and Feedback:**
    - Test the animation with real users to ensure it is clear and effective.
    - Collect feedback on whether the rotation directions are easily distinguishable and if the animation helps in understanding the control scheme.

5. **Refinement:**
   - Based on feedback, refine the animation for smoother transitions and better clarity.
   - Ensure the speed and smoothness of the rotation are appropriate and do not cause any visual discomfort.

By following these steps, you can create an effective and intuitive animation that demonstrates both clockwise and anticlockwise movements, helping users understand the new control scheme and making the interaction more engaging.

2.3 Created dropdown for number of motors.

When you add a dropdown to select the number of motors, it allows users to configure the system according to their specific needs. This feature can be particularly useful in applications such as robotics, automation systems, or any scenario where motor control is required.

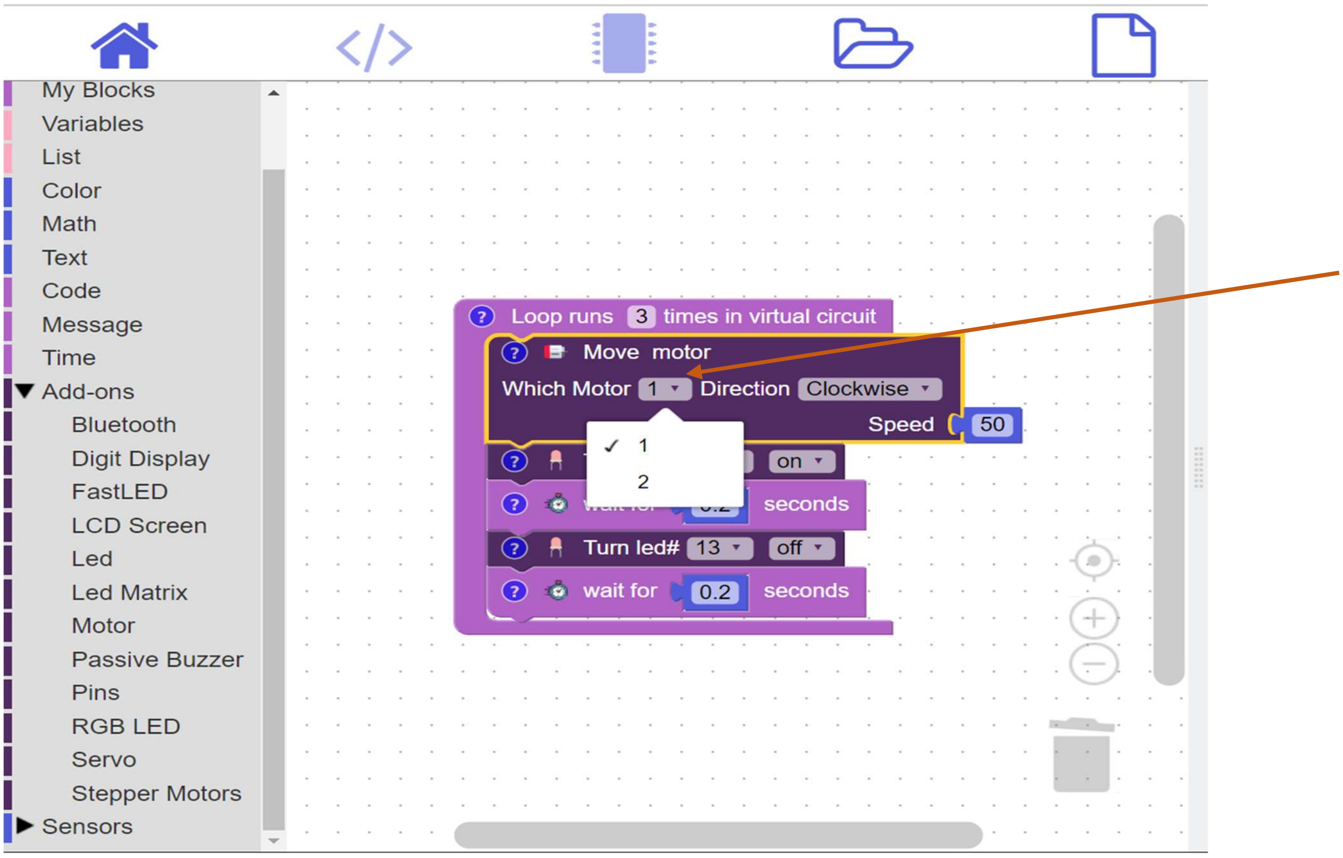


fig 2.3

Steps to Implement the Dropdown

1. **Define the Dropdown Options:**
    - Determine the range of motor numbers that can be selected (e.g., 1 to 6 motors).
    - Populate the dropdown with these options.
2. **Handle Selection:**
    - Capture the user's selection.

    - Adjust the interface or system configuration based on the selected number of motors.
3. **Update the Interface:**
    - Dynamically update the interface to display controls or indicators for the selected number of motors.
    - Ensure that the system or application logic adjusts to handle the specified number of motors.

Conceptual Approach

1. Dropdown Options:

- Provide a range of options that the user can select from, such as 1 to 6 motors.
- The dropdown should be easily accessible and clearly labeled.

2. Capture User Selection:

- When the user selects a number of motors, capture this selection.
- Use this selection to adjust the rest of the interface or system configuration accordingly.

3. Dynamically Update Interface:

- Based on the selected number of motors, dynamically display controls or indicators for each motor.
- Ensure that the controls are intuitive and provide clear feedback for each motor.

User Experience Considerations

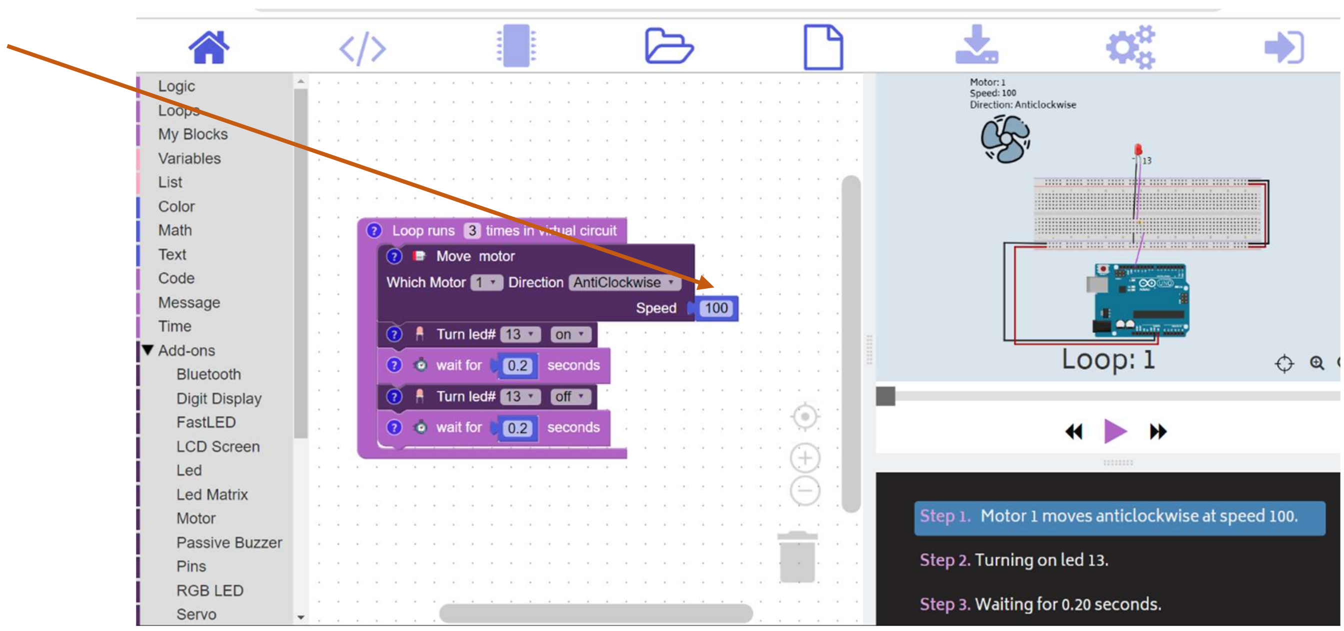
1. **Clear Labeling:**
    - Clearly label the dropdown to indicate its purpose, such as "Select Number of Motors."

2. **Responsive Interface:**
    - Ensure that the interface responds immediately to the user's selection by displaying the appropriate number of controls.
3. **Intuitive Controls:**
    - Design the motor controls to be intuitive and easy to use.
    - Provide clear feedback for each motor, such as indicating the current state or speed.
4. **Accessibility:**
    - Ensure the dropdown and controls are accessible to all users, including those with disabilities.
    - Provide keyboard navigation and screen reader support if applicable.

By implementing a dropdown for selecting the number of motors and dynamically updating the interface, you provide a flexible and user-friendly way for users to configure and control their motors. This approach ensures that the system can adapt to various configurations and user needs, enhancing the overall user experience.

2.4 Implementing Speed Control for Motors

Adding the ability to change the speed of each motor enhances the control and flexibility of your system.



Step 1. Motor 1 moves anticlockwise at speed 100.
Step 2. Turning on led 13.
Step 3. Waiting for 0.20 seconds.

Fig 2.4

Here's how you can conceptualize and implement this feature.

Steps to Implement Speed Control

1. **Dropdown for Motor Selection:**
   - Provide a dropdown to select the number of motors.
   - Dynamically generate controls for each motor based on the selection.
2. **Speed Control Sliders:**
   - For each motor, include a slider or input field to adjust the speed.
   - Ensure the speed range is appropriate for your motors (e.g., 0 to 100% or specific RPM values).
3. **Real-time Feedback:**
   - Display the current speed next to each slider.

    - Update the motor's speed in real-time as the user adjusts the slider.

Conceptual Approach

1. Dropdown Options for Motor Selection:

- Provide options like 1 to 6 motors.
- When the user selects a number, dynamically generate controls for each motor.

2. Speed Control Interface:

- For each motor, display a slider to adjust the speed.
- Optionally, include input fields for precise speed adjustments.

3. Real-time Speed Adjustment:

- As the user adjusts the slider, update the motor's speed immediately.
- Display the current speed next to the slider for visual feedback.

User Experience Considerations

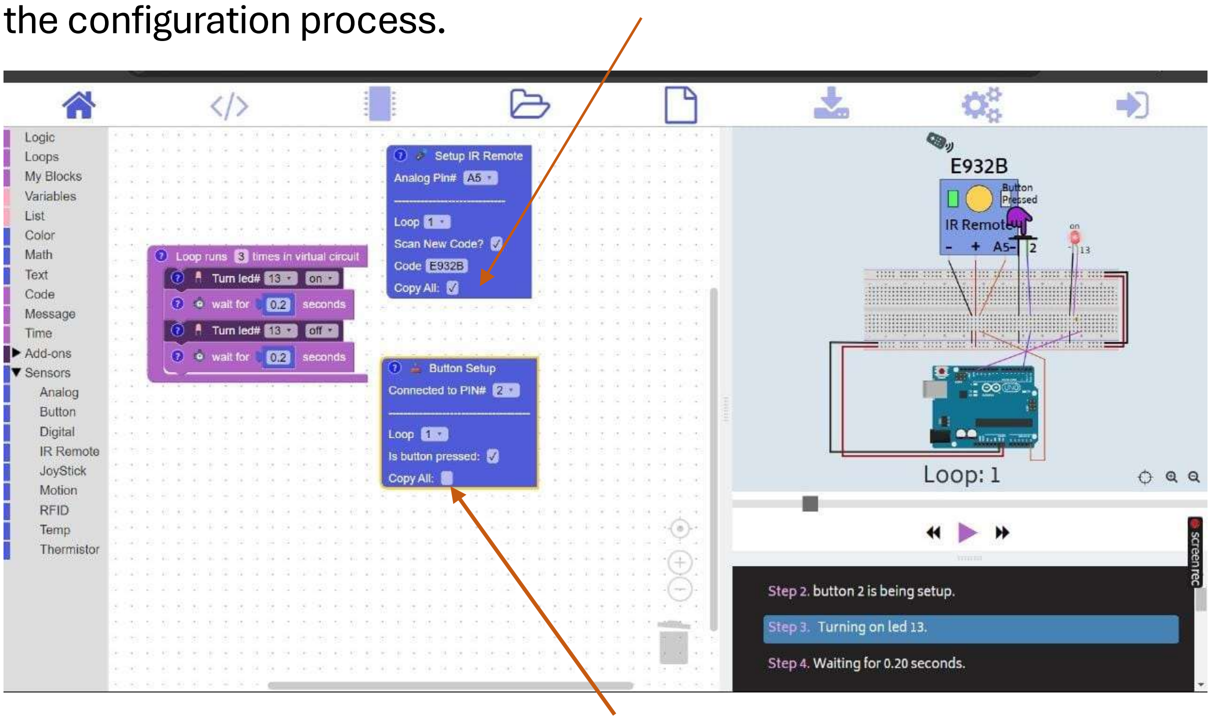
1. **Clear and Intuitive Controls:**
    - Label each motor and its corresponding speed control clearly.
    - Ensure the sliders are responsive and easy to use.
2. **Real-time Feedback:**
    - Provide immediate visual feedback for each motor's speed.
    - Consider using indicators or gauges to show the current speed.
3. **Error Handling:**

  - Handle cases where speed values might be out of range or invalid.
  - Provide error messages or warnings as necessary.

By implementing these features, you create a flexible and user-friendly interface that allows users to control the speed of each motor individually. This enhances the overall functionality and usability of your system, making it more adaptable to various use cases and user needs.

2.5 Added a checkbox in sensor setup block to apply the changes to all the loops

Adding a checkbox to apply changes to all loops in the sensor setup block can significantly enhance the user experience by streamlining the configuration process.



```
Loop runs 3 times in virtual circuit
  Turn led# 13 on
  wait for 0.2 seconds
  Turn led# 13 off
  wait for 0.2 seconds
```

```
Setup IR Remote
Analog Pin# A5
Loop 1
Scan New Code? ✓
Code E932B
Copy All: ✓
```

```
Button Setup
Connected to PIN# 2
Loop 1
Is button pressed: ✓
Copy All:
```

```
Step 2. button 2 is being setup.
Step 3. Turning on led 13.
Step 4. Waiting for 0.20 seconds.
```

Fig 2.5

Here’s how you can conceptualize and implement this feature effectively.

Conceptual Approach

1. **Checkbox Integration:**
   - Add a checkbox labeled "Apply changes to all loops" in the sensor setup block.
   - When checked, any changes made to one loop will automatically apply to all loops.
2. **User Interaction Flow:**
   - Users configure settings for one loop.

    - If the checkbox is checked, the same settings propagate to all loops.
    - If unchecked, users can configure each loop individually.

3. **Real-time Feedback:**
    - Provide immediate visual feedback to indicate that changes have been applied to all loops.
    - Display a confirmation message or indicator to reassure users.

Steps to Implement the Feature

1. Designing the Interface:

- **Checkbox Placement:** Place the checkbox within the sensor setup block in a prominent position.
- **Labeling:** Clearly label the checkbox as "Apply changes to all loops" for clarity.

2. Capturing User Input:

- When the checkbox is checked, listen for changes in the loop settings.
- Capture the configuration settings from the active loop.

3. Applying Changes:

- If the checkbox is checked, apply the captured settings to all loops.
- Update the configuration for each loop dynamically.

4. Providing Feedback:

- Display a message or visual indicator confirming that changes have been applied to all loops.
- Ensure that users can see which settings have been propagated.

Example Scenario

Use Case: Sensor Configuration Interface

1. **User Scenario:**
    - The user is configuring sensors for multiple loops in a system.
    - They want to apply the same settings to all loops to save time.
2. **Checkbox Interaction:**
    - The user checks the "Apply changes to all loops" checkbox.
    - They configure settings for Loop 1 (e.g., threshold values, sensitivity).
3. **Applying Settings:**
    - As the user changes settings for Loop 1, the same changes are applied to Loops 2, 3, and 4 automatically.
    - The interface shows a confirmation message: "Changes applied to all loops."
4. **User Feedback:**
    - The user sees real-time updates indicating that all loops have been configured with the same settings.
    - If the user unchecks the checkbox, they can adjust each loop individually without affecting the others.

User Experience Considerations

1. **Clear Instructions:**
    - Provide clear instructions or tooltips explaining the checkbox functionality.

- Ensure users understand that changes will be applied to all loops when the checkbox is checked.

2. **Feedback Mechanism:**
   - Implement visual feedback such as highlighting or a confirmation message to show that changes have been applied.
   - Consider adding an "undo" option in case users want to revert the changes.
3. **Accessibility:**
   - Ensure the checkbox is easily accessible and usable with keyboard navigation.
   - Provide appropriate labels and ARIA attributes for screen readers.
4. **Error Handling:**
   - Handle scenarios where applying changes to all loops might cause conflicts or errors.
   - Provide informative error messages or prompts to guide users.

By following these steps and considerations, you can implement an effective feature that allows users to apply changes to all loops with a single action, enhancing the efficiency and usability of your sensor configuration interface.

2.6 Created a page in Electroblocks to show list of projects readily available

Creating a page within Electroblocks to showcase a list of readily available projects can provide users with inspiration and resources to get started quickly. This page should be user-friendly, informative, and visually appealing.

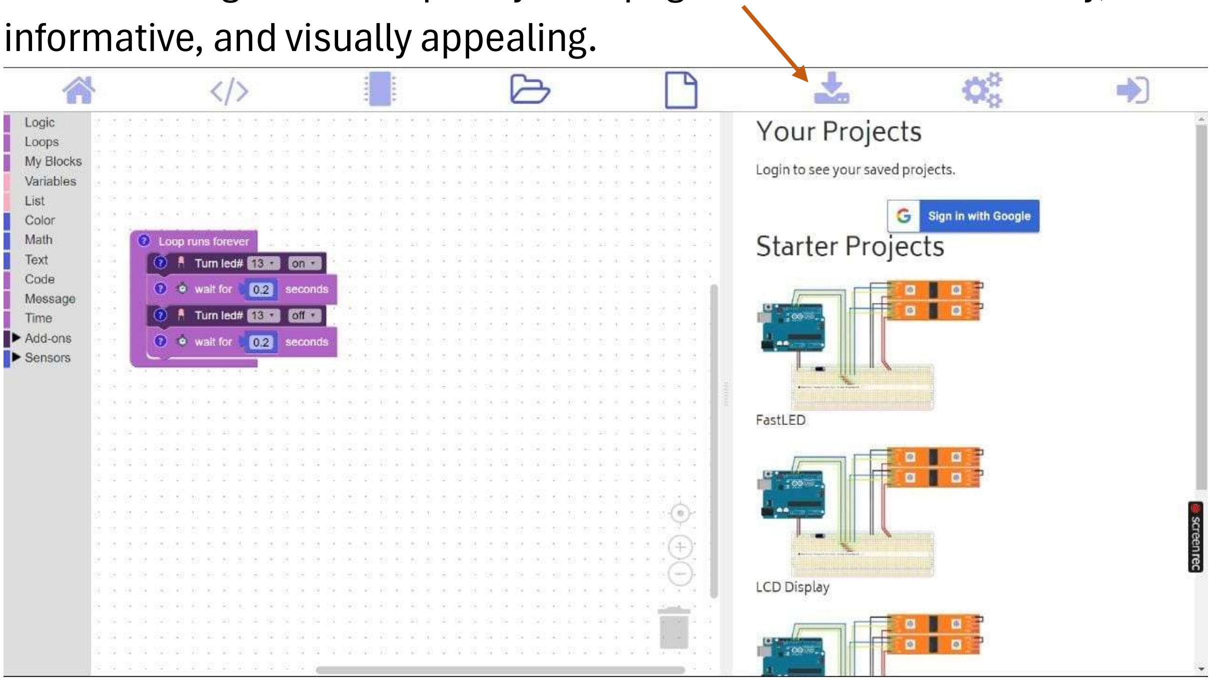


Fig 2.6

Key Components

1. **Header:**
   - Title: "Available Projects"
   - Brief description or introduction to the projects list.
2. **Project List:**
   - Each project should have a card or tile layout.
   - Information on each project should include:
     - Project title
     - Brief description
     - Difficulty level
     - Required components
     - Link to detailed instructions or tutorial
     - Image or thumbnail
3. **Filters and Search:**

    - Allow users to filter projects by difficulty level, components required, or categories (e.g., robotics, automation, sensors).
    - Include a search bar for quick access to specific projects.
4. **Footer:**
    - Additional resources or links
    - Contact information or support links

Layout and Design

1. Header:

- **Title:** "Available Projects"
- **Description:** A short paragraph introducing the projects list and encouraging users to explore and try out different projects.

2. Project List:

- Use a grid layout for better visual organization.
- Each project card should include:
    - **Title:** Clearly display the project name.
    - **Description:** A brief summary of the project.
    - **Difficulty Level:** Indicate whether the project is Beginner, Intermediate, or Advanced.
    - **Components:** List key components required for the project.
    - **Link:** A button or link to view detailed instructions or a tutorial.
    - **Image:** An illustrative thumbnail or image related to the project.

3. Filters and Search:

- **Filters:** Dropdown menus or checkboxes to filter projects by:
    - Difficulty level (Beginner, Intermediate, Advanced)
    - Components required (Arduino, sensors, motors, etc.)
    - Categories (Robotics, Automation, Sensors, etc.)
- **Search Bar:** A search input at the top for users to quickly find specific projects.

4. Footer: Include links to additional resources, such as tutorials, community forums, or documentation.

- Provide contact information or links to support channels.

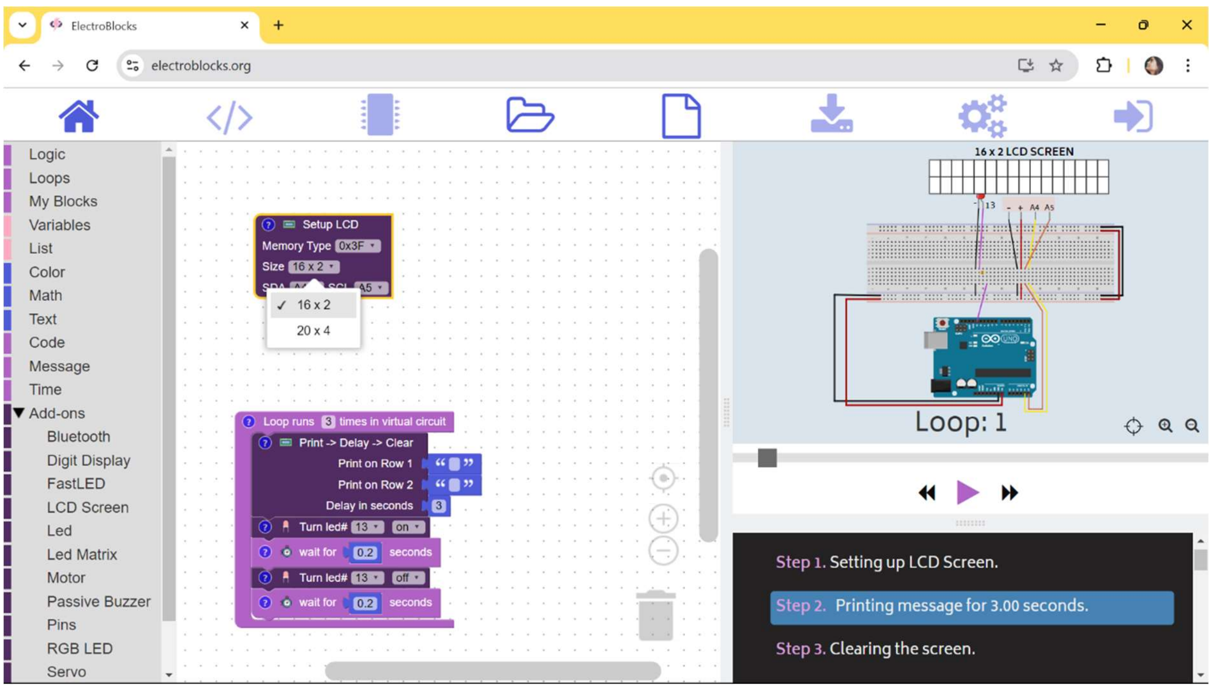
User Experience Considerations

1. **Ease of Navigation:**
    - Ensure the page is easy to navigate with clear headings and sections.
    - Use consistent design elements and spacing.
2. **Visual Appeal:**
    - Use high-quality images and a clean, modern design.
    - Make sure the text is readable with sufficient contrast.
3. **Responsiveness:**
    - Ensure the page is responsive and looks good on various devices, including desktops, tablets, and smartphones.
4. **Accessibility:**
    - Ensure the page is accessible to users with disabilities.
    - Use appropriate ARIA labels and ensure keyboard navigability.

By following these guidelines, you can create a functional and appealing projects list page in Electroblocks, making it easier for users to find and engage with available projects.

Chapter 3

Bug Fixing



Setup LCD
Memory Type 0x3F
Size 16 x 2
✓ 16 x 2
20 x 4

Loop runs 3 times in virtual circuit
Print -> Delay -> Clear
Print on Row 1 " "
Print on Row 2 " "
Delay in seconds 3
Turn led# 13 on
wait for 0.2 seconds
Turn led# 13 off
wait for 0.2 seconds

Step 1. Setting up LCD Screen.
Step 2. Printing message for 3.00 seconds.
Step 3. Clearing the screen.

3.1 Reducing / adding the number of rows when selected 16*2 / 20*4

Adjusting Content for Different Display Modes

When designing an interface that allows users to switch between different display modes, such as 16x2 and 20x4 LCD displays, you need to ensure the content fits appropriately within the constraints of each mode. Here’s how to handle this:

Understanding the Display Modes

16x2 Display:

- 16 characters per row
- rows

20x4 Display:

- 20 characters per row
- 4 rows

Steps to Adjust Content

1. Determine the Current Mode:

- Allow the user to select the display mode through a dropdown menu or similar interface element.
- Read the selected mode to know whether it's 16x2 or 20x4.

2. Prepare the Content:

- Collect or generate the content that needs to be displayed.
- Ensure the content is flexible enough to be split across multiple rows.

3. Adjust Content Based on Mode:

- Calculate the number of characters per row and the number of rows available.
- Split the content into segments that fit within the current display's constraints.
- For a 16x2 display, divide the content into chunks of 16 characters each and distribute them over 2 rows.
- For a 20x4 display, divide the content into chunks of 20 characters each and distribute them over 4 rows.

4. Display the Adjusted Content:

Clear any previous content displayed.

Render the new content, ensuring each segment is placed in its respective row.

Make sure to handle cases where content might overflow the available space.

User Experience Considerations

- Smooth Transitions:
- Implement smooth transitions when switching between modes to enhance the user experience.
- Avoid abrupt changes that can confuse the user.

Content Overflow Handling:

- For content longer than the display capacity, implement scrolling or pagination.
- Inform users if content is truncated to fit the display.

Real-Time Updates:

- Ensure that any change in the display mode updates the content in real-time without requiring a page refresh.

By following these steps, you can effectively adjust and display content on different LCD display modes, providing a seamless user experience.

Chapter 4

Conclusion

This project was truly special, like a work of art that I thoroughly enjoyed creating. Working on it and adding different features was a lot of fun, and I got to learn so many new things along the way. I'm really thankful to my mentors for their amazing guidance throughout the project. I believe this project has so much potential. It could be a great help to students and researchers who are exploring Electroblocks(Arduino). All the effort and creativity that went into this project make it a valuable tool for learning and researching. I'm excited about the impact this project could have and how it might make a difference for others.

Chapter 5

Future Work

- This project can be improved further by adding new features and fixing already present bugs.
- Create a virtual environment that does not require the internet for people with bad internet.
- It should work on a Windows machine as the server. You will need to get both the main Electro Blocks repo and the Electro Blocks server working.
- Expanded Sensor Integration
- User Interface Enhancements
- Automated testing and debugging tools
- Multi-language support
- Enhanced data visualization tools