Robotics Repository Website Documentation

Created by: Pritom Paul
Email: paulp3@rpi.edu
Github repo link: https://github.com/PritomP25/Robotics-Repository-URP

Table of Contents

Overview

What is this project?

This project is a website designed to help students access resources from the “Yahboom” website more easily. It gathers educational material and presents it in a structured, accessible format.

Primary Goal

Simplify the discovery and retrieval of academic or robotics-related resources by providing a centralized interface for students to view and easily understand.

Getting Started: Setup & Run

This is a website created with just HTML, CSS, and JavaScript. There’s no backend.

To view or edit locally:

1. Clone the repo
2. Open index.html in a browser

Repository Structure

The root of the project contains the following key components:

- .vscode/
  - Editor workspace settings (**optional**, as its only contain prettier extension)
- assets/images/
  - Directory for image assets used throughout the site
- prism/
  - Contain files related to Prism.js for code syntax highletting. DON'T TOUCH, unless you add a new code language to display on the website. Otherwise you need to redownload added/updates Languages and Plugins
  - Link to the website for download: https://prismjs.com/
  - Useful resources: Design a Code Snippet Section using PRISM
- index.html
  - Main landing page of the site
- about.html
  - A page to describe the purpose or background of the website/project.
- dofbot.html
  - A page about a specific "DOFBOT" component or concept.
- style.css
  - Core styling rules and visual design of the site.
- script.js
  - JavaScript to handle interactivity, dynamic behaviors, or external data fetching.

Quick summary on the Functionality/Files

- index.html
  - Acts as the entry point; contains navigation, featured repository card, and layout structure.
- about.html
  - Provides background, mission, and links to original Yahboom repo website.
- dofbot.html
  - A page about a specific "DOFBOT" component or concept. Check content for specifics.
- style.css
  - Defines site-wide visual rules—fonts, layout, colors, and responsiveness.

- script.js
    - Adds interactivity, such as dynamic content loading, navigation behaviors, or DOM manipulations.
- assets/images/
    - Insert any images used for the website into this folder

Code Documentation & Important Content Specific

Please follow this guide for understanding about the structure of the html file, as well as understanding on how to add/update content for the dofbot or future repository html file.

High-Level Structure Summary for Repository Content

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <t-header></t-header>                    <!-- Custom site header -->
    <t-hero ... ></t-hero>                   <!-- Intro section with hero image -->

    <section class="content-body">
      <aside id="sidebar"> ... </aside>      <!-- Navigation menu -->
      <main> ... </main>                     <!-- All lesson content -->
    </section>
  </body>
</html>
```

Key UI section:

1. Header (<t-header>):

Custom reusable header component (likely defined in external JS).

2. Hero Section (<t-hero>):

Title, intro text, warnings, and hero image for DOFBOT.

3. Sidebar (<aside id="sidebar">):

Accordion-style navigation with expandable/collapsible menus (toggleSidebar() & toggleSubMenu() functions in JS).
Links use data-target attributes to scroll to content in <main>.

4. Main Content (<main>):

Each lesson or section is in a with a unique id matching data-target in the sidebar.
Includes explanations, images, code examples, and API documentation.

Header

This is all you need for the header. Just paste this line of code in any page of the html as it will display the header content in any page.

```
<!-- header -->
        <t-header></t-header>
```

For more in-depth, look at the Javascript file as it shows the html code for both mobile and other displays.

```
// header
class THeader extends HTMLElement {
    connectedCallback() {
        this.innerHTML = `
        <header>
            <div class="nav-div">
                <ul id="nav-sidebar" class="nav-links">
                    <li onclick="hideSidebar()">
                        <a href="#"
                            ><span class="material-symbols-outlined">
                                close
                            </span></a
                        >
                    </li>
                    <li><a href="index.html">Home</a></li>
                    <li><a href="index.html#Robot">Robotics</a></li>
                    <li><a href="about.html">About</a></li>
                </ul>
                <ul class="nav-links">
                    <li>
                        <a href="index.html"
                            ><img src="assets/images/logo-1.png"
alt="logo" class="logo-img"
                        /></a>
                    </li>
                    <li class="hideOnMobile"><a
href="index.html">Home</a></li>
                    <li class="hideOnMobile"><a
href="index.html#Robot">Robotics</a></li>
```

```
                    <li class="hideOnMobile"><a
href="about.html">About</a></li>
                    <li class="menu-button" onclick="showSidebar()">
                        <a href="#"
                            ><span class="material-symbols-outlined">
                                menu
                            </span></a
                        >
                    </li>
                </ul>
            </div>
        </header>
        `;
    }
}
customElements.define("t-header", THeader);
```

Hero Section

The hero section takes in 4 arguments:

- **Heading**: As the name suggested, it's a heading
- **Details**: Any paragraph to display
- **Subtext**: **optional**; a caption for extra information separate from Details.
- **Img**: insert a img for the background

```
<!-- hero -->
        <t-hero
            heading="Welcome to DOFBOT-JetsonNANO repository"
            details="Be sure to read the initial warnings for
utilizing the Dofbots. These platforms will be utilized by many
students this semester and in the future. To ensure their integrity,
any deliberate misuse of the arms will result in revocation of use
privileges and possible disciplinary action."
            subtext="Many examples are adapted from: <a
href='http://www.yahboom.net/study/Dofbot-Jetson_nano'
target='_blank'>http://www.yahboom.net/study/Dofbot-Jetson_nano</a>.
Note that this website may have changed so students should talk to
the instructor before running anything found on this site."
```

```
            img="assets/images/dofbot1.jpg"
        ></t-hero>
```

Check the JavaScript file for the actual html code!

Footer

Similar as the Header, insert this one code to display the footer in any page of html.
Note: For in the future, you can change the footer (in JavaScript file) to make it a more RPI department-like structure, since this is used for RPI student purposes.
DO site the original creator: Pritom Paul

```
        <!-- footer -->
        <t-footer></t-footer>
```

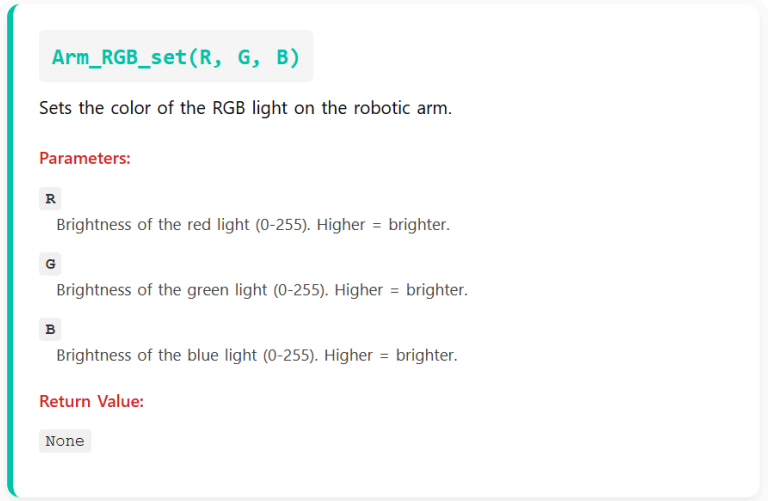
Any changes to the footer, please look at the JavaScript file for the Footer HTML code.

Navigation Logic

- JavaScript Hooks:
    - toggleSidebar() — expands/collapses the entire sidebar.
    - toggleSubMenu(this) — toggles visibility of a submenu for a topic.
- Data Attributes:
    - <a href="#" data-target="intro-notes"> links match in main content.
    - It uses JS to show the selected section.

How to use Function Card and Code Block?

Wondering on how to use this to display what a function does?



`Arm_RGB_set(R, G, B)`

Sets the color of the RGB light on the robotic arm.

Parameters:

`R`
Brightness of the red light (0-255). Higher = brighter.

`G`
Brightness of the green light (0-255). Higher = brighter.

`B`
Brightness of the blue light (0-255). Higher = brighter.

Return Value:

`None`

Here's the Code on how the **function card** works:

All class can be viewed in the .css file for how it works.

```
<div class="func-wrapper">
    <div class="function-card">
        <div class="function-header">
            <div class="function-name">
                <code>Arm_RGB_set(R, G, B)</code>
            </div>
            <p class="function-desc">
                Sets the color of the RGB light on the
                robotic arm.
            </p>
        </div>

        <div class="section">
            <h4 class="section-title">Parameters:</h4>
            <dl>
                <dt><code>R</code></dt>
                <dd>
                    Brightness of the red light (0-255).
                    Higher = brighter.
                </dd>
                <dt><code>G</code></dt>
                <dd>
                    Brightness of the green light (0-255).
                    Higher = brighter.
                </dd>
                <dt><code>B</code></dt>
                <dd>
                    Brightness of the blue light (0-255).
                    Higher = brighter.
                </dd>
            </dl>
        </div>

        <div class="section">
            <h4 class="section-title">Return Value:</h4>
            <p><code>None</code></p>
        </div>
    </div>
</div>
```

 is the container.
 is the card outline.

 is the header div that contains the function-name and function-desc.

Each will contain a <h4> tag with a class=”section-title”

From there for the **Parameters**: followup with each argument with its respectable <dt><code><code></dt> with a <dd> of the description.

From there for the **Return Value**: followup with its respectable <code><code> .

How to do a **single-code line** for base command?

Code path:

```
/home/jetson/Dofbot/3.ctrl_Arm/1.rgb.ipynb
```

Explanation:

```
<p>Code path:</p>
<div class="code-path">
    /home/jetson/Dofbot/3.ctrl_Arm/1.rgb.ipynb
</div>
```

 bash_command goes here

- The class name “code-path” is defined in the .css file for how it works.

How to do a Block Code for python?

```python
# Cycle through the RGB lights on the robot arm expansion board to illuminate red, green,
#!/usr/bin/env python3
#coding=utf-8
import time
from Arm_Lib import Arm_Device
# Get the object of the robotic arm
Arm = Arm_Device()
time.sleep(.1)
def main():
    while True:
    Arm.Arm_RGB_set(50, 0, 0) #RGB Red light on
    time.sleep(.5)
    Arm.Arm_RGB_set(0, 50, 0) #RGB Green light on
    time.sleep(.5)
    Arm.Arm_RGB_set(0, 0, 50) #RGB blue light on
    time.sleep(.5)
    print(" END OF LINE! ")
try :
    main()
except KeyboardInterrupt:
    # Release the Arm object
    del Arm
    print(" Program closed! ")
    pass
```

Note: this uses the prism folder that is a plugin for the language to be shown as code.

Explanation:

-
    - .css file for outlining how the code-box to look like

- <pre class="line-numbers">
  - Refer to prism.js documentation on how to use it
  - **TLDR**: this is how to get the line number shown on the side.
- <code class="language-python">
  - Refer to prism.js documentation on how to use it
  - **TLDR**: once you download your preferred language from the prism website into the prism folder, add class="lanauge-*code_lang*" for the syntax highlight to apply.

```
                    <div class="code-box">
                        <pre class="line-numbers">
                            <code class="language-python">
                            # Cycle through the RGB lights on the
robot arm expansion board to illuminate red, green, and blue.
                            #!/usr/bin/env python3
                            #coding=utf-8
                            import time
                            from Arm_Lib import Arm_Device
                            # Get the object of the robotic arm
                            Arm = Arm_Device()
                            time.sleep(.1)
                            def main():
                                while True:
                                Arm.Arm_RGB_set(50, 0, 0) #RGB Red
light on
                                time.sleep(.5)
                                Arm.Arm_RGB_set(0, 50, 0) #RGB Green
light on
                                time.sleep(.5)
                                Arm.Arm_RGB_set(0, 0, 50) #RGB blue
light on
                                time.sleep(.5)
                                print(" END OF LINE! ")
                            try :
                                main()
                            except KeyboardInterrupt:
                                # Release the Arm object
                                del Arm
```

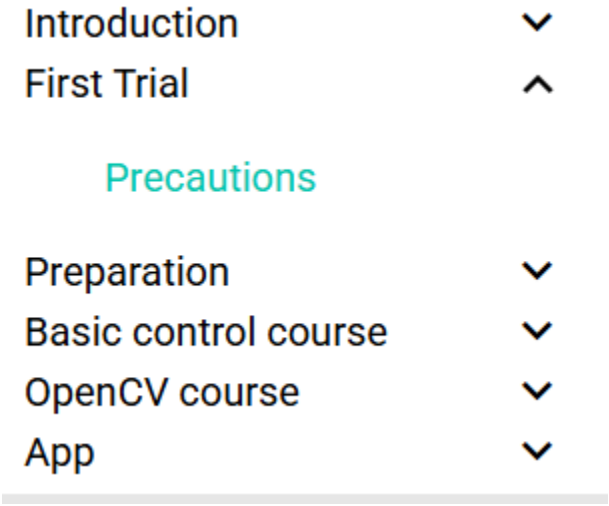
```
                print(" Program closed! ")
                pass
            </code>
        </pre>
    </div>
```

How to Add/Update Content

1. Add new sidebar link:

In the dofbot.html file, there is a code related to the sidebar where the user can click the section and it will refresh the page to showcase the content within the <li> tag.

```
<li>
    <button
        onclick="toggleSubMenu(this)"
        class="dropbutton-btn"
    >
        <span class="dropbutton-label">First Trial</span>
        <span
            class="material-symbols-outlined dropbutton-icon"
        >
            keyboard_arrow_down
        </span>
    </button>
    <ul class="sub-menu">
        <div>
            <li>
                <a href="#" data-target="precautions"
                    >Precautions</a
                >
            </li>
        </div>
    </ul>
</li>
```



The name “First Trial” is within the <button> tag

- To add a new sidebar dropdown name comes from:
    - First Trial
- keyboard_arrow_down is the “^” icon

To have all of the dropdown submenu such as “Precautions” revolves on the <ul> tag with the class=”sub-menu”.

- So to add a new sidebar link you have to

```
<li>
  <a href="#" data-target="new-section">New Lesson</a>
</li>
```

2. Add new content section:

This huge div tag has the same id tag as the data-target, which any content you wanna display will show in that First Trial content page.

```
<div id="precautions" class="content-container">
    <h2>Precautions for DIY Robotic Arm</h2>
    <p>
        <b
            >Note: When the robot arm is gripping objects, it is
            necessary to control the angle of the gripper.
            Improper angle setting may cause the servo to stall
            and burn.</b
        >
    </p>
    <p>
        Here is a table of gripper angles, which records the
        angle that the servo needs to be set to for every 0.5 cm
        object.
    </p>
    <p>
        You can adjust the angle you set when gripping according
        to this table to avoid stalling the servo.
    </p>
    <div class="img-div">
        <img
            src="assets/images/motor6_length.jpg"
            alt="Showing the Object length from the gripper."
        />
    </div>
```

- The class=”content-container” is just a class tag where in the css I can format

So to add new content that link with the data-target of “new-section”:

```
<div id="new-section" class="content-container">
  <h2>New Lesson Title</h2>
  <p>Explanation...</p>
  <div class="code-box">
    <pre><code class="language-python"># Example code here</code></pre>
  </div>
</div>
```