Studica App - Documentation

Table of Contents

Adding a Device:

Refer to the following steps to add support for a new device on the Studica App:
Note: To add support for multiple devices of the same type, you would need to repeat these steps for each additional device you would like to support.

1. Open “index.html” and search for “Adding Device - 1”. This is where you will create a button for the device you would like to add. Use the button template provided at the end of that div (“Add new device button here”).
    a. Save an image/icon of the device in the “img” folder located in the project directory.
    b. In the add new button template, inside the parentheses of “url()”, add the path of the image you added to the “img” folder (e.g. url(./img/image.png)).
    c. Then, for “class”, add a class name similar to: [deviceName]Image, this will be used later in CSS.
    d. Then, for “id”, add an ID similar to: [deviceName]Btn, this will be used later in renderer.js.
    e. Finally, inside the span tag (HERE), add the device name as you would want it to be displayed on the front end.

2. Open “index.css” and search for “Adding Device - 2”. This is where you will be adding the class name you created in (1c). In the CSS styling declaration under that comment, add the class name you created ahead of the older ones, separated by a comma.
   (e.g.
   .VMX2Image,
   .NavX2MicroImage,
   .NavX2MXPImage,
   .[deviceName]Image { …
)

3. Open “renderer.js” and search for “Adding Device - 3”. This is where you will declare a variable for the new device to store its corresponding index value. Add the variable declaration under “Add additional devices here”.

4. Search for “Adding Device - 4”. This is where you will toggle off the visibility of the new device button on the application load. Use the template provided under “Template”. Inside the quotations in parentheses, add the ID you created in (1d).

5. Search for “Adding Device - 5”. This is where you will declare a boolean variable that stores whether multiple of the same type device was detected. Under “Add additional devices here”, declare two variables corresponding to your device name:
   (e.g.
   let connected1_[deviceName] = false;
   let connected2_[deviceName] = false;

)

6. Search for “Adding Device - 6”. This is where you would add code to remove styling from the new device button. In the template, under the “Template” section, add the ID that you created in (1d) inside the quotations in parentheses.

7. Open “index.html” and search for “Adding Device - 7”. This is where you add your device for the auto DFU Mode detect functionality. Replace “device4” with the respective device name as follows: “[deviceName]Device”. Additionally, inside the p tag (HERE), add the device name as you want it to be displayed on the frontend

8. Open “renderer.js” and search for “Adding Device - 8”. This is where you add a variable to be used for event handling. Select the appropriate template from “Template” and replace “[ID]” with the device name you created in (7).

9. Search for “Adding Device - 9”. This is where you add the event handler that responds when the new device button is clicked. Use the template under “Template” and replace “[ID]” with the id created in (7) and replace “[deviceName]” with the new device name.

10. Search for “Adding Device - 10”. This is where you add a variable to be used for event handling. Select the appropriate template from “Template” and replace [ID] with the device name you created in (1d).

11. Search for “Adding Device - 11”. This is where you add the event handler that responds when the new device button is clicked. Use the template under “Template” and replace “[ID]” with the id created in (1d) and replace “[deviceName]” with the new device name.

12. Search for “Adding Device - 12”. This is where you add the event handler that responds when the new device button is clicked. It then fills out an area of the sidebar bar with the device information. Use the template under “Template” and replace “[ID]” with the id created in (1d) and replace “[deviceName]” with the new device name.

13. Search for “Adding Device - 13”. This is where you add the if condition that parses the JSON file for update data for the new device. Use the template and replace “[deviceName]” with the device name.

14. Search for “Adding Device - 14”. This is where you add an if condition to download new updates for the new device to its respective local folder. Replace “[deviceName]” with the new device name. Replace “[folder]” with the local folder made to store updates for the new device.

15. Search for “Adding Device - 15”. This is where you add an if condition to download multiple new updates for the new device to its respective local folder. Replace [deviceName] with the new device name. Replace “[folder]” with the local folder made to store updates for the new device.

16. Search for “Adding Device - 16”. This is where you toggle cursor styling for the new device. Replace “[deviceName]” with the new device name.

17. Search for “Adding Device - 17”. This is where you add any new global variables you declared at the beginning. Make sure to set them back to their initial declaration values.

18. This step may or may not be required. Search for “Adding Device - 18”. This is where you would configure pitch, roll, and yaw movement calculations if the board rotations don’t match the device. Use the code there as a reference and change the calculations as needed to make the rotations work correctly.

19. Open “Program.cs” located in the “CSHARP” folder. Search for “Adding Device - 19”. This is where you add an if condition to check for new updates for the new device. Replace “[deviceName]” with the device name exactly as found in the board info. Replace “[folder]” with the local folder name that stores update files.

20. Search for “Adding Device - 20”. This is where you add the backend selection for a DFU detection for the new device. Use the template and replace [deviceName] with the new device name.

21. Search for “Adding Device - 21”. This is where you add an if condition to get paths for locally saved updates for the new device. Replace “[deviceName]” with the device name exactly as found in the board info. Replace “[folder]” with the local folder name that stores update files.

Increasing Download List Support:

As of app version (development) 5.7.4, the maximum number of download files supported by the list is 14. If you wish to increase this value to support more downloads and display them on the list, refer to the following steps:

1. Open “index.html” and search for “Adding Downloads - 1”. You will see a list of 14 downloads, each declared by a <p> tag. This list is used in Firmware Updater and appears when “Load Available Downloads” is clicked. Use the template located at the bottom of the list, and replace “[#]” with a numerical value following the sequence of the list above the template.

2. Search for “Adding Downloads - 2”. Similar to step 1, you will see a list of 14 downloads, this list is for the pop-up that appears when new downloads are available on the C.D.N. Use the template located at the bottom of the list, and replace “[#]” with a numerical value following the sequence of the list above the template.

3. Open “renderer.js” and search for “Adding Downloads - 3”. Here you will add an event detection using the template. Add this line for each new download you added in index.html. Replace “[#]” with the number you added in steps 1 and 2.

4. Search for “Adding Downloads - 4”. Here you will add the event handler for a click on the new download. Add this for each new download you added in steps 1 and 2. Copy the template and replace “[#]” with the value from the previous steps. Replace “[# - 1]” with 1 less than the previous values, because it is indexing an array.

5. Search for “Adding Downloads - 5”. Here you will declare a boolean variable that stores whether the download has been selected. Declare a new variable for each new download support added. Use the template and replace “[#]” with the value from the previous steps.

6. Search for “Adding Downloads - 6”. Here you will add an event detection using the template. Add this line for each new download you added in index.html. Replace “[#]” with the number you added in steps 1 and 2.

7. Search for “Adding Downloads - 7”. Here you will add the event handler for a click on the new download. Add this for each new download you added in steps 1 and 2. Copy the template and replace “[#]” with the value from the previous steps.

8. Search for “Adding Downloads - 8”. Here you will change the arguments for the two for-loops below the comment. Change the conditions to the maximum number of downloads supported on the list (e.g. if the maximum is 14: for (let i = 0; i < 14; i++)).

9. Search for “Adding Downloads - 9”. Similar to before, here you will change the argument of the first for-loop below the comment. Change the condition to the maximum number of downloads supported on the list (e.g. if the maximum is 14: for (let i = 0; i < 14; i++)).

10. Search for “Adding Downloads - 10”. Exactly as the step before, here you will change the argument of the first for-loop below the comment. Change the condition to the maximum number of downloads supported on the list (e.g. if the maximum is 14: for (let i = 0; i < 14; i++)).

Packaging the App:

By default, the app/code should be set up for development mode. In order to switch the app for the release build, refer to the following steps:

1. Open “renderer.js” and search for “Debugging & Build Config”. Refer to the comments in that section and comment in/out a line as necessary. The first line (“For Dev Mode”) should be commented out when building the app for release. The second line (“For Release Build”) is for release with the backend (c#) not compiled. The final line (“For Release Build (.exe)” is for release with the backend (c#) compiled. There should only be one line active at a time.

2. Open “program.cs” and search for “Release Path”. Remove the comment from each line that includes “Release Path”, and comment out the line underneath, which says “Dev Path”. Note, when switching back to dev mode, repeat this step and do the opposite, comment out lines with “Release Path” and remove the comment from lines with “Dev Path”.

3. Now you can package the app using the line: “npm run make”. For more information on this command and how to set a target architecture, refer to the readme in the FInal APP project folder.

Note: If you want to use a compiled version of the backend, run the following command in the “CSHARP” directory: “dotnet publish”. This will create an executable file in the directory specified in the final line of “Debugging & Build Config” (see step 1).

Changing Directories:

Depending on where you download the project folder or how you set up the app for deployment, changing the specific directories referenced in the app may be necessary. Refer to the following steps on how to do so:

1. Open “renderer.js” and search for “/Directory”, every line in renderer.js with this comment references a specific directory. “${__dirname}” means current directory. You can modify these lines according to your needs.

2. Open “program.cs” and search for “/Directory”, every line in program.cs with this comment references a specific directory. You can modify these lines according to your needs. For more information: https://learn.microsoft.com/en-us/dotnet/api/system.io.directory.getdirectories?view=net-7.0