Submitting Arduino Code to the Adafruit Learning System

Created by Anne Barela

https://learn.adafruit.com/submitting-arduino-code-to-the-adafruit-learning-system

Last updated on 2024-06-03 03:29:33 PM EDT

Table of Contents

Overview

The Adafruit Learning System has a great number of Arduino projects for good reason - Arduino is the oldest and most ubiquitous microcontroller development framework for makers, artists, and hobbyists.

This guide is to help submit Arduino projects to the Adafruit Learning System. There are some specifics that need to be followed to ensure the Continuous Integration (CI) system is able to verify the code compiles and is ready.

Two Methods

There are two methods Arduino code may be added to the Adafruit Learning System.

1. You may follow this guide and make your own GitHub Pull Request (PR). This is the preferred method if you are familiar with GitHub (and git, depending on your setup).

2. You may discuss with your Adafruit Learning System facilitator and see if they will add the code for you. This is fine for beginner programmers and those unfamiliar with GitHub. Experienced coders may be expected to learn to submit their own PR.

All repositories have a special file in them to let the CI determine the board being used in the project to ensure the Arduino IDE uses the correct board definition. This file is explained on the next page.

Add Author and License Information

Adafruit is standardizing on adding author and license information to all code files. For Arduino, this would be all .ino, .h, and .cpp files. For CircuitPython, it would be all .py files.

The information goes at the top of each code file before any other comments or code.

Please put the current year and the author(s) first and last name in the first line. In the third line, generally Adafruit code is MIT licensed. If you code derives from code under CC or other licenses, please list them, although **MIT is the preferred license**.

The phrase "for Adafruit Industries" is for Adafruit authors and is not required for general authors.

Arduino

```
// SPDX-FileCopyrightText: YYYY Your Name for Adafruit Industries
//
// SPDX-License-Identifier: MIT
```

An example:

```
// SPDX-FileCopyrightText: 2021 Anne Barela for Adafruit Industries
//
// SPDX-License-Identifier: MIT

First line of code or additional comments
```

or

```
// SPDX-FileCopyrightText: 2022 Jane Doe
//
// SPDX-License-Identifier: MIT
```

If there is more than one author, you can have multiple lines for SPDX-FileCopyrightText, one below the other each listing the date and author as appropriate:

```
// SPDX-FileCopyrightText: 2017 Limor Fried/ladyada for Adafruit Industries
// SPDX-FileCopyrightText: 2017 Phillip Burgess for Adafruit
```

`Industries`
`//`
`// SPDX-License-Identifier: MIT`

CircuitPython

CircuitPython uses the Python comment `#` instead of the C comment `//`. ALso the latest versions of the Lint/CI checker want a description delimited by three single quotes beginning and end:

```
# SPDX-FileCopyrightText: YYYY Your Name for Adafruit Industries
#
# SPDX-License-Identifier: MIT
'''MyExample is a program to play MP3 sound files'''
```

An example:

`# SPDX-FileCopyrightText: 2021 Anne Barela for Adafruit Industries`
`#`
`# SPDX-License-Identifier: MIT`
`'''MyExample is a program to draw pictures'''`

`First line of code or additional comments`

SPDX

Documenting information in this fashion is consistent with The Software Package Data Exchange® (SPDX®). SPDX is an open standard for communicating software bill of material information, including components, licenses, copyrights, and security references. SPDX reduces redundant work by providing a common format for companies and communities to share important data, thereby streamlining and improving compliance.

The SPDX specification is an international open standard ISO/IEC 5962:2021 (https://adafru.it/VrB).

The list of valid license types are on the SPDX website here (https://adafru.it/VrC).

Please keep to the formats listed above unless instructed by Adafruit staff.

Troubleshooting

- Be sure the comments are left justified and worded close to the example text including spaces.
- Be sure you used the right comment mark for the language you are coding with.
- Check that the license given is valid in the list here (https://adafru.it/VrC)
- If you are not writing for Adafruit, the phrase "for Adafruit Industries" is not required.

Adding a .test.only file

How does the CI know what board it is compiling against? That is usually selected in the Arduino IDE but the CI doesn't know how to do this unless you provide it some information.

The Adafruit Learning System uses a zero length file to test and see which microcontroller board it should run the tests on. This takes the form .**xxxxx.test.only** where xxxxx is a code for the board to test against.

The most current list of test boards is found in the **arduino_cron.yml** file under `arduino-platform` https://github.com/adafruit/Adafruit_Learning_System_Guides/blob/main/.github/workflows/arduino_cron.yml (https://adafru.it/19dT)

The detailed platform list is in https://github.com/adafruit/ci-arduino/blob/master/all_platforms.py (https://adafru.it/19dU)

If a microcontroller is not listed, then you should contact your Adafruit Learning System facilitator for assistance.

If you are an Adafruit team member, you will need to add the new board entry. The repo for the Arduino CI is https://github.com/adafruit/ci-arduino (https://adafru.it/ZXD) and it describes various processes in the CI pipeline. It is suggested you get assistance from the CI maintainers with such changes (see the repo for current active maintainers).

Example:

If you have an Adafruit Circuit Playground Express, you look in the file and there is a listing for "cpx_ada". Use that one.

Using your editor, make a file (contents irrelevant, zero length or just a space are fine). Name the file **.cpx_ada.test.only** and include it in the directory with the .ino file for the project.

If you have examples for multiple microcontroller boards, put each example in it's own subdirectory with an appropriate .test.only file.

An extra step for using TinyUSB

If your sketch uses the TinyUSB USB functionality of compatible boards, you need to make one more check. In **githubci.yml**, some boards exist in two forms - one with the board name and one with _TinyUSB appended, re.

metro_m4 and **metro_m4_tinyusb**

The second version with **_tinyusb** will select compiling the Metro M4 with TinyUSB turned on. Only boards in **githubci.yml** that have the **_tinyusb** entry also work with compiling with TinyUSB.

Advanced: for a list of TinyUSB compatible boards, you can also look in CI file https://github.com/adafruit/ci-arduino/blob/master/build_platform.py (https://adafru.it/Y3F) (note this may have changed in 2023).

What if I don't do this?

The CI will look at the files and not know what board to use in the Arduino IDE run and will give errors as it tries everything. Your submission will not pass and your facilitator will ask you to include the correct .test.only file. You may ask the facilitator to do it for you but you know your project, it's expected you do this yourself.

Advanced: Skip the CI

In very rare cases, like when code is only a template and will not compile, the keyword file **.none.test.only** might be considered.

Please discuss with the Guide Moderation Coordinator is you have this edge case. Normal code using this will not be accepted.

Making Your Own Pull Request

Those proficient in GitHub and Git may look to submit their own pull request for their project code and files.

Please follow the steps below:

1. Please ensure your code compiles clean for the target microcontroller board in the latest downloadable IDE (re. not Arduino Create/web).

2. Gather all files the project needs. Source files, data files, etc.

3. Add appropriate author and license information per instructions in this guide.

4. Make an appropriate **.xxxx.test.only** file.

Once these steps are done, please follow the Adafruit Learning System guide

Contribute to the Adafruit Learning System with Git and GitHub (https://adafru.it/VrD)

This guide goes step by step into using GitHub and Git - in getting a fork of the Adafruit Learning System, Branching, Add/Commit/Push, and creating a Pull Request.

Follow These Guidelines

Be sure the directory structure of the Pull Request matches what you want in the Learning System.

Name the main directory after your board and project. Look at other submissions in the repository on how others should do it. The format is

`Boardname_Descriptive_Text`

With boardname the microcontroller board (At least the first letter capitalized, not in all caps, please. And the descriptive text as necessary. Example - a project using the Adafruit Feather M0 board to do servo rotation might be:

`Feather_M0_Servo_Rotation`

If you have code for multiple projects, please put it in directories with the same name as the code file, re:

- Feather_M0_Servo_Rotation
  .feather_m0_test.only
  - servo_rotate_right
    servo_rotate_right.ino
  - servo_rotate_left
    servo_rotate_left.ino

Please be sure you credit yourself as author and note the software license of the code per Contribute to the Adafruit Learning System with Git and GitHub (https://adafru.it/VrD).

Bonus: Include a README.md file in the main directory with information on your code and project.

Submitting a Pull Request

When the PR window is displayed in GitHub, the description text has been prepopulated with instructions. Please review them. Add your own comments on the title of your learning guide.

Submit your PR.

As for reviewers, please tag your facilitator for review.

The Continuous Integration (CI) process will start. Please keep an eye on the status. If the code fails to compile on your target board(s), please make corrections and resubmit.

If you find a library is missing, see the next page "Missing Libraries".

Once the code has been certified by the CI, it'll be ready for facilitator review and possible merge. Please work with your facilitator if any issues arise.

If you seem to have issues with the CircuitPython linting process in the CI, place a file named .circuitpython.skip in your folder to tell the CI that it shouldn't bother using CircuitPython with this project.

When Your PR is Merged

When your code has completed CI and Adafruit has merged your PR, you can go to GitHub and get links to your main source files for inclusion in your guide. Skip to the page Placing Your Code Into Your Guide for instructions.

Missing Libraries

Sometimes your code is compiling in the CI and it gives an indication that a library file is missing. This is definitely possible. Just as you need to use a .test.only file to select the board (as the compiler doesn't know which one you used), the compiler also doesn't know all the libraries you used in the Arduino IDE.

The Adafruit CI includes many libraries in it's compilation process. But the libraries are changing all the time and some relatively new library may have been added and used by a guide developer and it isn't in Adafruit's list to include.

Generally only Adafruit developers will add new libraries to the CI list. All PRs on the file below should be well documented as to the intent on changes.

Where Libraries are Listed for the Learning System

In the main Adafruit Learning System Guides repository (https://adafru.it/Clx) is a file named library.deps **(https://adafru.it/VrE)**. That file has one line listing the library names automatically loaded by the CI.

Getting the Library's Name

The **library.deps** file uses the specific library names separated by a comma and space,

A specific library name if found as follows:

Go to the directory housing the library in GitHub.

There should be a file named **library.properties** in the source files for the library.

Look in the **library.properties** file for a line with "name="

All the next after the equals sign is the library name you'll be adding to **library.deps**.

Note some library names include the word Adafruit, some include the word Library, but it may not. This is why you must use the text exactly as listed after the equals sign.

Example: Here is the the contents of **library.properties** for the Adafruit BMP280 Library (https://adafru.it/fIK):

```
name=Adafruit BMP280 Library
version=2.4.2
author=Adafruit
maintainer=Adafruit &lt;info@adafruit.com&gt;
sentence=Arduino library for BMP280 sensors.
paragraph=Arduino library for BMP280 pressure and altitude sensors.
category=Sensors
url=https://github.com/adafruit/Adafruit_BMP280_Library
architectures=*
depends=Adafruit Unified Sensor, Adafruit BusIO
```

The name of the library is all the text on line with name=. So this library name for our use is "Adafruit BMP280 Library"

Putting in a PR for a Library Addition

Do a Pull Request on an edited version of **library.deps**. The change consists of editing the file to add a comma and space after the last library listed. Then put the library name you found in the step above to the list of libraries.

The Pull Request should be rather descriptive as to why you're changing the file - generally including a new library defined by Adafruit.

When your PR is approved, you can rerun the CI on your Adafruit Learning System guide code.

Is there an error about needing TinyUSB?

See the "Adding a .test.only file" page to correct this.

Send Code to Your Facilitator

If you are not comfortable using git or GitHub, it is perfectly ok to coordinate with your Adafruit guide facilitator and provide your code to them. There are some guidelines we'd like to have with your code:

1. Ensure it compiles with the latest downloadable Arduino IDE (not the online Create site).

2. At the top of each source file (.ino, possibly .h and/or .cpp files), you need a header with the author and license information (as noted previously in this guide).

3. Note to your facilitator what microcontroller board your code runs on. Most often this is an Adafruit board but it could be others.

4. If you have code that runs on different microcontrollers, re. one for Circuit Playground Express and one for Feather SAMD21, please note which is which as the code will be placed in separate directories within the main repository for the guide.

What Will Happen

Your facilitator will copy your code into the Adafruit Learning System GitHub repository at https://github.com/adafruit/Adafruit_Learning_System_Guides (https://adafru.it/CIx).

The code will be set up in one directory, with separate subdirectories as needed for data, etc. If there is code for more than one microcontroller type or multiple examples, these may be put in subdirectories.

The code will be processed by the Continuous Integration (CI) process which essentially uses a command line version of the Arduino IDE to compile your code along with necessary libraries.

If your code does not compile:

Your facilitator will let you know what errors are encountered and the author is expected to fix issues and send corrections back to the facilitator.

When Your PR is Merged

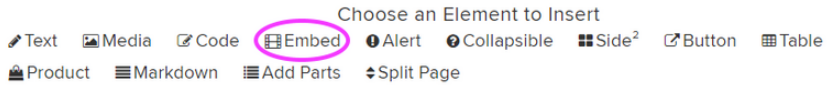
When your code has completed CI and Adafruit has merged your PR, you can go to GitHub and get links to your main source files for inclusion in your guide. See the next page for instructions.

Placing Your Code Into Your Guide

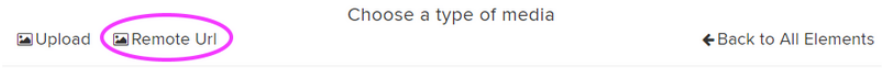
When the code compiles and completes CI:

Embed links to your programs into your guides. Only the .ino file for the main program need be embedded.

Where you want the code, choose the Embed Element:



In the the dialog box that follows, choose URL, in which you will use the web address for the .ino file in GitHub:



Use the URL provided by the facilitator. When done, you can use the guide preview function to see how it looks. If you get an error "Unable to display", use the browser refresh button.