Apio Installation and Use Guide for macOS and Windows

Made By: Isaac Medina and Duncan DeFonce for Tufts ES4

Tufts University Department of Electrical and Computer Engineering

October 1, 2025

We are beginning with macOS. Skip to page 5, or press **here**, for the Windows installation guide.

macOS

Before you can use Apio, there are a few preliminary steps to take. This may include installing or updating Python via Homebrew, and subsequently downloading Apio and adding it to your $PATH.

1) Installing Python and Homebrew

To install Apio, you will need to have **Python 3.11** or higher installed–The recommended version is **Python 3.13**. If you are certain that you have Python 3.11 or higher and Homebrew installed, skip to page 4, or press **here** for Apio installation steps. If you do not have these, follow the steps for installing Homebrew and then Python.

1. To check whether you have Python installed, open a new terminal and run the following commands.

```
which python
python3 --version
```

⟶If you have it installed, the output should be something like

```
/opt/homebrew/opt/python@3.13/libexec/bin/python
python 3.13.7
```

⟶If you do not have it installed, you'll see something like,

```
python not found
command not found: python
```

If you do not have the required version of Python 3, please proceed with the following steps.

1.1) Installing Homebrew

Homebrew is a package manager for macOS that makes it easy to install, update, and manage software from the command line. In this case, we are using it to install or update Python.

1. Check to see if you have Homebrew installed by running the following commands in terminal:

   ```
   which brew
   brew --version
   ```

   ⟶Again, if you have it installed, you'll see

   ```
   /opt/homebrew/bin/brew
   Homebrew 4.6.11
   ```

   ⟶If not,

   ```
   brew not found
   command not found: brew
   ```

   ⟶ If you have it installed, update it to the latest version with:

   ```
   brew update
   ```

2. If you **do not** have it installed, you'll first need to check to make sure you have Apple's Xcode toolset installed.

   ```
    xcode-select --install
   ```

   ⟶ You'll likely see this message, as most have it pre installed,

   ```
   xcode-select: note: Command line tools are already installed. Use "
      Software
   Update" in System Settings or the softwareupdate command line interface
      to
   install updates
   ```

   ⟶ If you don't have it installed, a pop-up install will appear. Install the packages.

3. To officially install Homebrew run the following command:

   ```
   /bin/bash -c "$(curl -fsSLhttps://raw.githubusercontent.com/Homebrew/
      install
   /HEAD/install.sh)"
   ```

   ⟶ The above command installs Homebrew, but you need to add it to your $PATH. To do this, run the following two commands.

```
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> ~/.zprofile
eval "$(/opt/homebrew/bin/brew shellenv)"
```

4. You should now have Homebrew installed and connected to your $PATH. Check this by running:

   ```
   brew --version
   ```

   ⟶ You should see:

   ```
   Homebrew 4.x.x
   ```

   ⟶ Before moving on, use the following command to make sure Homebrew is up to date.

   ```
   brew update
   ```

   ⟶ It should say:

   ```
   Homebrew 4.6.11
   ```

1.2) Installing and/or Updating Python

1. Now that you have Homebrew installed, we can move on to installing Python. If you have Python installed, you can update to the latest version with

   ```
   brew upgrade python
   ```

2. To install Python, use the following command.

   ```
   brew install python
   ```

3. Homebrew should automatically install it to $PATH. Check to make sure you have it with these commands:

   ```
   echo $PATH
   which brew
   which python3
   which pip3
   ```

4. You should see something like the following:

   ```
   /opt/homebrew/bin/brew
   /opt/homebrew/bin/python3
   /opt/homebrew/bin/pip3
   ```

5. Homebrew should have installed the latest version of Python, but if you want to check run:

```
python --version
```

⟶ It should be:

```
Python 3.13.7
```

Once you have successfully installed Python, move on to installing Apio.

2) Installing Apio

1. Install pipx:

```
brew install pipx
pipx ensurepath
```

⟶ This ensures that what we install with pipx is automatically connected to $PATH

2. Run the following command:

```
pipx install git+https://github.com/fpgawars/apio.git@develop
```

⟶ This may take a few minutes. Pipx will install Apio and all required dependencies.

3. Verify installation:

```
apio --version
```

⟶ It should say:

```
apio, version 1.0.0
```

4. **If you are having issues and the command does not come up, most likely you need to add apio to your $PATH:**

```
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

To verify it worked, run:

```
which apio
apio --version
```

You should now see:

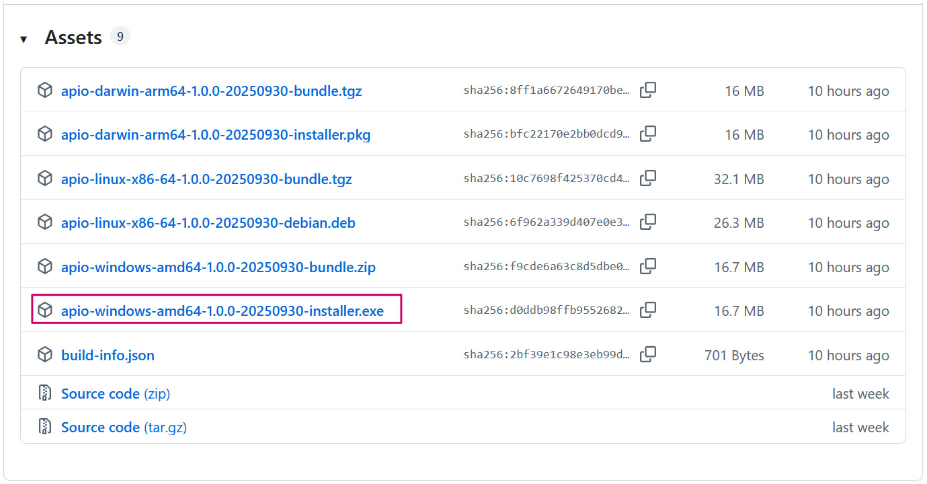
```
/Users/xxUSERNAMExx/.local/bin/apio
apio, version 1.0.0
```

Congratulations, you have successfully installed Apio in your macOS machine. Move on to page 8, or press here for the Apio user guide.

Windows

1) Installing Apio

1. Begin by downloading the latest Windows release from this website. Scroll down to "Assets" and download the file called "apio-windows-amd64-1.0.0-XXXXX-installer.exe".



Assets 9

apio-darwin-arm64-1.0.0-20250930-bundle.tgz	sha256:8ff1a6672649170be…	16 MB	10 hours ago
apio-darwin-arm64-1.0.0-20250930-installer.pkg	sha256:bfc22170e2bb0dcd9…	16 MB	10 hours ago
apio-linux-x86-64-1.0.0-20250930-bundle.tgz	sha256:10c7698f425370cd4…	32.1 MB	10 hours ago
apio-linux-x86-64-1.0.0-20250930-debian.deb	sha256:6f962a339d407e0e3…	26.3 MB	10 hours ago
apio-windows-amd64-1.0.0-20250930-bundle.zip	sha256:f9cde6a63c8d5dbe0…	16.7 MB	10 hours ago
apio-windows-amd64-1.0.0-20250930-installer.exe	sha256:d0ddb98ffb9552682…	16.7 MB	10 hours ago
build-info.json	sha256:2bf39e1c98e3eb99d…	701 Bytes	10 hours ago
Source code (zip)			last week
Source code (tar.gz)			last week

2. After the download has completed, run the .exe file on your machine and allow it to make changes to your device (you may get a security message because this is open source software, but it's perfectly safe).
   (a) Check the "Add application directory to your environmental path" box
   (b) Hit next and then install.
3. Congratulations! Apio is now installed (with a lot less effort than it would take on Mac).

2) Using Windows Command Prompt

Unfortunately, Apio has no user interface, so we have to use the Windows Command Line. To warm up, we'll make a directory to keep all your code neatly on your machine.

If you're familiar with the command line, skip to step 3 to start with APIO setup

1. Hit the Windows Key, type "Command Prompt", and hit enter. This should open a window that looks something like:

```
C:\Users\dunca>
```

   This is your current directory (or folder). You can see a list of all the current subdirectories with the command

```
dir
```

   The output looks like this, with directory names on the right side.

```
        09/22/2025 03:03 PM <DIR> .
        01/20/2025 08:24 PM <DIR> ..
        09/24/2025 12:43 PM <DIR> .apio
        01/21/2025 12:21 PM <DIR> Contacts
        12/26/2022 04:53 AM <DIR> Documents
        09/30/2025 04:49 PM <DIR> Downloads
        01/21/2025 12:21 PM <DIR> Favorites
        09/23/2025 12:48 PM <DIR> OneDrive
```

We can enter any of the directories by using the *change directory* command. If there's a space in the directory's name, you have to include it in double quotes.

```
        cd "directory name"
```

If you enter the wrong directory by accident, you can back up a step with

```
        cd ..
```

You can repeat these two commands to enter any directory just like we would in file explorer.

2. Let's navigate to a good place to store our data. If you already have a folder with class materials, feel free to navigate with the *cd* and *dir* commands as described above. If not, we'll make a new one under "Documents". Use the following commands:

```
        cd Documents
        mkdir ES4
        cd ES4
        mkdir TestProject
        cd TestProject
```

Now the filepath should look something like the following. You'll have to enter this directory using the *cd* command every time you open a new command prompt window.

```
        C:\Users\dunca\Documents\ES4\TestProject>
```

3) Setting up Drivers

Lastly, we need to download the drivers to get our machine to recognize the board when we plug it in. Plug your board in and use the following command:

```
        apio drivers install ftdi
```

This will open a new program called zadig.exe. Allow it to make changes to your device.

1. Plug your upduino into your computer
2. Find and open the Zadig window on your screen.

3. Select your FPGA board from the drop down list.(For us it's the upduino31).

   VERY IMPORTANT - If your board appears multiple times, make sure to select its 'interface 0' entry.

4. Select the 'WinUSB' driver as the target driver. For example 'WinUSB (v6.1.7600.16385)'.
5. Click 'Replace Driver' and wait for a successful completion, this can take a minute or two.
6. Close the Zadig window.
7. Disconnect and reconnect your FPGA board for the new driver to take affect.
8. Run the command 'apio devices usb' and verify that your board (the upduino) is listed.

Congratulations! Apio is all set to go on your device. Continue on to get started.

Using Apio

First, make sure you're in the filepath that you want to store your project in. If you're on Windows, you can use the *cd* and *dir* commands as described in the windows section. macOS is the same: cd and ls work fine for managing directories. We highly reccomend using VScode for your apio projects as you can use the internal VScode terminal with apio.

To write our code, we'll need a code editor like VSCode. If you don't have it, you can download it here

To see the full list of available apio commands, type:

```
    apio -h
```

Or to see more info about a specific command, type:

```
        apio [insert command here] -h
```

These are the important commands that we will use the most:

```
Build commands:
  apio build (Synthesize the bitstream.)
  apio upload (Upload the bitstream to the FPGA.)
  apio clean (Delete the apio generated files)

Verification commands:
  apio lint (Lint the source code.)
  apio sim (Simulate a testbench with graphic results.)

Setup commands:
  apio create (Create an apio.ini project file.)

Utility commands:
  apio boards (List available board definitions.)
```

1)Creating a new project

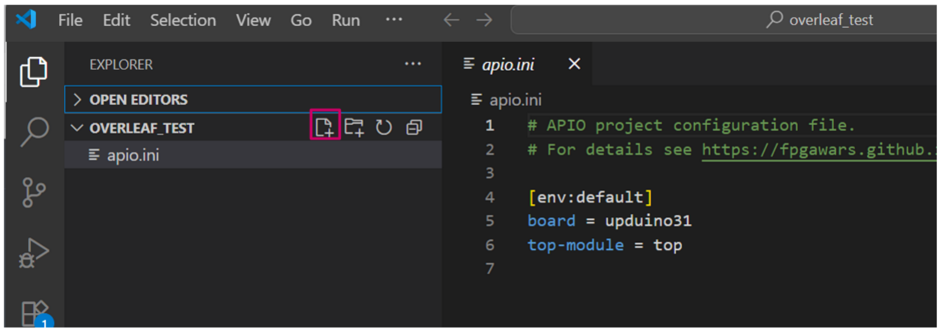
1. Once we're in the right filepath (just for this test project), we want to use the *apio create* command to make our project. Check your board to see if you have the **upduino31** or **upduino30**.

```
        apio create -b [insert the board you have] -t top
```

2. Now we need to make the top module which will house our code. Either type the following command into your terminal, or run VSCode and open your current folder.

```
        code .
```

3. We can add a file in VSCode by clicking the newfile icon in the top left. All our SystemVerilog files end in .sv, name this one **top.sv** and hit enter.



```
# APIO project configuration file.
# For details see https://fpgawars.github.

[env:default]
board = upduino31
top-module = top
```

4. Create a top.sv file, and other dependencies thereof.

2) apio lint

1. Once you have some code you're happy with, you can test it with the command:

```
apio lint
```

If sucessful, move on to the next test to flash your code onto the FPGA. If not, debug the error it gives you.

3) Flashing to the UPduino 3.0/1 FPGA

1. First, you will need a .pcf file – typically called top.pcf. This is a file that holds your pinmappings. It looks something like this:

```
# Inputs from DIP switches
set_io V[0] 44
set_io V[1] 4
set_io V[2] 3
set_io V[3] 48
set_io V[4] 45
set_io V[5] 47

# Common anodes for 7-segment display digits
set_io digitone 28
set_io digittwo 6
```

2. Once you have mapped your pins, you run the following commands to flash your FPGA

```
apio build
apio upload
```

If the build is successful, you should see the following output

```
isaacmedina12@Isaacs-MacBook-Pro Lab 6 Verilog % apio build
Using env default (upduino31)
Setting shell vars.
------------------------------------------------------------------------------

scons: ‘build’ is up to date.
====================== [SUCCESS] Took 0.21 seconds ====================
isaacmedina12@Isaacs-MacBook-Pro Lab 6 Verilog % apio upload
Using env default (upduino31)
Setting shell vars.
Scanning for a USB device:
- FILTER [VID=0403, PID=6014, REGEX="^UPduino v3\.1"]
- DEVICE [0403:6014] [0:4] [tinyVision.ai] [UPduino v3.1] []
------------------------------------------------------------------------------

iceprog -d d:0/4 _build/default/hardware.bin
init..
cdone: high
reset..
cdone: low
flash ID: 0xEF 0x40 0x16 0x00
file size: 104090
erase 64kB sector at 0x000000..
erase 64kB sector at 0x010000..
programming..
done.
reading..
VERIFY OK
cdone: high
Bye.
============== [SUCCESS] Took 6.11 seconds ===================
```