**Portal Box ESP32 Guide**

# Device Specifications

# 

Figure 1: [ESP32-C6-DevKitC-1 v1.1](https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32c6/esp32-c6-devkitc-1/user_guide_v1.1.html)

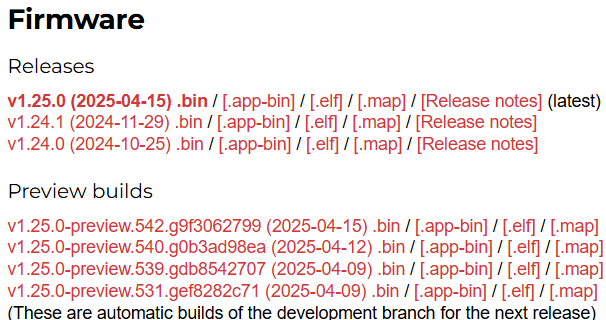
In the previous version of the Portal Box, a Raspberry Pi Zero was used, however, the Raspberry Pi was expensive and had much more processing power than what was required to operate the function of the Portal Box. We have decided that an ESP32-C6 would be the better option because it was 1/10 of the cost of the Raspberry Pi while also being more than capable enough to run the software. Currently, we have transitioned from CircuitPython to MicroPython due to the lack of tools which enable CircuitPython to be coded onto the C6. The C6 has a built-in Wifi Module as well as two ports for UART and USB connections. Comparing the C6 to other ESP32 models such as the S3 or the C3, the C6 provided the best specifications of everything we wanted while also maintaining functionality and cost. For more information, a link has been provided to the ESP32-C6 above in Figure 1.

# Software and Firmware Installation

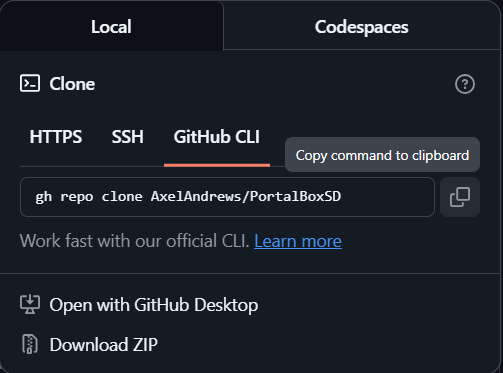
Step 1: Installing the Integrated Development Environment (IDE)

* First, we chose [Virtual Studio Code (VSCode)](https://code.visualstudio.com/download) as our IDE of choice.
* Second, we need to install [GitBash](https://www.geeksforgeeks.org/install-git-on-windows/) using the tutorial provided.
* Third, we need to install 2 important modules using the terminal (ctrl + `) while in VSCode. While in the terminal, type:
  + **esptool**: (pip install esptool)
  + **mpremote**: (pip install mpremote)

Step 2: Installing the Firmware and Flashing the ESP32

* First, we must first install the Firmware for Micropython to put onto the ESP32 from the link below:
* <https://micropython.org/download/ESP32_GENERIC_C6/>
* Scroll down to the bottom of the page and click on the first link in the **Firmware** section as shown in bold below:
  + [**v1.25.0 (2025-04-15) .bin**](https://micropython.org/resources/firmware/ESP32_GENERIC_C6-20250415-v1.25.0.bin) (or a future version of the Firmware).
* 
* Second, connect to the ***USB*** port on the ESP32 using a cord that is **able to** transfer data and not just power.
* Third, we need to find the proper Port that the device is located at. This can be done a number of ways when the device is connected.
  + Method 1 (**Windows**):
    - Click on the Windows button on your keyboard OR click on the Windows icon in the bottom left corner of your screen and look up “Device Manager”.
    - Scroll down to **Ports (COM & LPT)** and make note of which COM port is assigned to your ESP32. They may require you to reconnect. ***THE COM PORT IS DIFFERENT PER MICROCONTROLLER.***
    - 
  + Method 2 (**Mac):**
    - Open Terminal (you can find it in Applications > Utilities, or by searching for "Terminal" in Spotlight)
    - Type ls /dev/tty.\* and press Enter
    - Look for something like /dev/tty.SLAB\_USBtoUART or /dev/tty.usbserial-\* in the results - this is likely your ESP32 port
    - If you don't see a matching device, try disconnecting and reconnecting your ESP32, then run the command again
    - **Remember that the exact port name might vary depending on the USB-to-serial chip used in your ESP32 development board**
* Fourth, open the terminal on VSCode and type and enter the lines:
  + (esptool --port <COMX> erase\_flash)
  + (esptool --port <COMX> --baud 460800 write\_flash 0 <PATH TO THE DOWNLOADED .bin FILE>)
    - **NOTE:** Getting the path can be done by finding the file, right clicking and clicking on “Copy as path”
    - The COMX is the name of the port like the one shown above where X is the number assigned to the COM port.
* Finally, the ESP32 is flashed with MicroPython and ready to be used
  + The ESP32 can be modified using [mpremote](https://docs.micropython.org/en/latest/reference/mpremote.html).
  + The commonly used mpremote commands are:
    - (mpremote)
      * Check to see if a device is connected to the computer.
    - (mpremote run <fileName.py>)
      * Runs the file from your computer
      * Note: Imported Files must be stored onto the ESP32.
    - (mpremote fs cp <source file> :<destination file>)
      * Note: There is a space between the source file and the colon
      * The destination file, will be the name of the file and overwritten if it already exists.
    - (mpremote fs rm <fileName.py stored on the ESP32>)
      * Removes a file from the ESP32
  + Full list of commands can be found in the link provided above.

Step 3: Cloning the Repository

* Go to our GitHub repository: <https://github.com/AxelAndrews/PortalBoxSD>
* Click on the blue “< > Code” button and click on the HTTPS tab within “Local” and copy the link.
* Open VSCode and open the terminal and change your directory (cd <path to file you want this to be stored at>) to the location you want to store the files at via the terminal. Then to clone the repository, type (git clone https://github.com/AxelAndrews/PortalBoxSD.git). This will copy the files from the repository to the file location that you are currently at.
  + 
  + 

Step 4: Updating config.json

* ***At this point, the Portal Box SHOULD be assembled. Refer to*** [***Portal Box Assembly Guide***](https://docs.google.com/document/d/1g384_tL9M6RhhY-4YpHSNVq1-W_CzYQTKuRvIzMxAE0/edit?usp=sharing) ***for the construction instructions. Future instructions will be under the assumption that the box has been built without hardware issues.***
* After cloning the files, it is crucial that you update the config.json file with your credentials. In particular, it is **critical** that you update the db and wifi section of the json file with the credentials to your network and to your respective database such that the Portal Box can connect to the internet and to the database where user information is stored.
* Additional options:
  + "enable\_buzzer": Enables/Disables the buzzer while in operation
  + "buzzer\_pwm": Enables modulation of the buzzer frequency for various tones
  + "enable\_keypad": Enables/Disables the keypad while in operation. A pushbutton replaces its position if set to “*False*”
  + "enable\_LCDScreen": Disables the LCD Screen if set to “*False*”

Step 5: Uploading files onto the ESP32

* While in the root of the newly cloned repository and while connected to the ESP32. Reset the device and then type: (py copyFiles.py Firmware/) into the terminal and click enter. This will attempt to clone all files from the Firmware folder onto the ESP32 which are all the necessary files required to use the Portal Box.
  + **NOTE 1: For your specific device may not need py to run the file. It may be python or python3 instead.**
  + **NOTE 2: It may crash the first time and may require the code to be ran several times until all files are copied. If the code were to crash then either kill the terminal, click the reset button on the ESP32, or click (ctrl+c) while in the terminal and try again.**

Step 6a: Running the Portal Box while connected via USB-C (For Debugging/Testing)(RECOMMENDED for the first use of the device)

* While in the root of the repository, you can use (mpremote run Firmware/Service.py) to initialize the Portal Box. After a series of changing colors for starting up the box and getting connected to the database and internet, we are able to start using the Portal Box!
* **Note: The box may require you to press the reset button before running any mpremote commands**

Step 6b: Running the Portal Box for full time operation

* In order to have the box running full time. We need to copy boot.py (mpremote fs cp boot.py :boot.py) onto the ESP32 which will run Service.py **while the device has power.** The next time the device is reset, the software should operate with just power.

# Debugging

While boot.py is on the ESP32, users cannot communicate with the ESP32. Under that circumstance, users should refer to the reset\_boot.py stored in the BootingHelp folder on the repo. The directions are stored both on the file as well as below.

* Removing Boot.py
* While connected to the ESP32: Reset the Device
* Enter “mpremote” (Recommended that you pre-type this). This will start a 3 second countdown
* Before the countdown expires. Press (ctrl+c) on the keyboard to enter REPL.
* Copy the entire script of reset\_boot.py (ctrl+a → ctrl+c)
* Enter the entire script onto the REPL (ctrl + v)
* Press the enter button twice and boot.py should be removed from the device.
* Press (ctrl+x) to exit the REPL.
* Debugging the Software
  + While the Service.py is running, the terminal will have an abundance of outputs displaying anything that happens. Pay attention to anytime the software may crash and what state it was in prior to crashing. Once finished, you can load boot.py onto the ESP32 and continue operation. Refer to Step 6a to operate the Portal Box whilst connected via USB-C.

# 