

# Pocket Portal Power Play



Wi-Fi Access Points  
with a Twist

# Pocket Portal Power Play

Crafting Wi-Fi Access Points with a Twist

A how-to guide by Candide Uyanze  
based on "Spread ideas with a pocket wifi portal"  
by Iffy Books

License: CC0 1.0 Universal



Download the zine's source files:  
[github.com/pocket-portal/zine](https://github.com/pocket-portal/zine)

**The Wemos D1 mini is a low-cost, low-power development board based on the ESP8266 Wi-Fi chip. You can use it to host a web page that anyone nearby can access.**

Imagine you're on a train, in an area with poor cell reception. You look for available Wi-Fi networks, and there's one called "Come Get This Wi-Fi". When you connect, you get a pop-up window that

says, "Sorry, this network doesn't connect to the internet! Enjoy this essay!" And what follows is something you never would've read otherwise!

This zine will show you how to make your own wi-fi access point and web server using a Wemos D1 mini board. The Wemos D1 mini is small, cheap, and easy to program using the Arduino IDE. It's also

incredibly power efficient. Plug one into a medium-sized USB power bank to create a portable server that can run for hours to a couple of days.

Here are things you could do with your wi-fi captive portal using a Wemos D1 mini:

- Welcome guests to your house party

- Provide a digital colouring page for commuters
- Plug your mixtape / jukebox / playlist
- Promote your clothing swap / garage sale / repair cafe to neighbours without using social media
- Share an article / essay / political slogan / poetry anthology with anyone who happens to be at the coffee shop.

- Share maps and information on a hiking trip without cell reception.
- Use several wi-fi boards to send a message using SSID names alone.

## **What's a captive portal?**

A captive portal is a page displayed to the user when they connect to a Wi-Fi access point, often used to let the user

authenticate before connecting to the internet. You may have seen Wi-Fi captive portals in coffee shops and hotels.

## What you'll need:

- Computer
- Wemos D1 Mini (4MB) or Wemos D1 Mini Pro (16 MB)
- MicroUSB cable
- Arduino IDE (integrated development environment),

which is the application you'll use to program your Wemos D1 mini.

- Source-code editor of your choice, which you'll use to edit your site files. (I like VS Codium)

You don't need to know how to program to complete this tutorial!

Knowing some HTML will help, but it isn't necessary.

You will find all the code and example projects at:

[github.com/pocket-portal/code](https://github.com/pocket-portal/code)



# Install the Arduino IDE

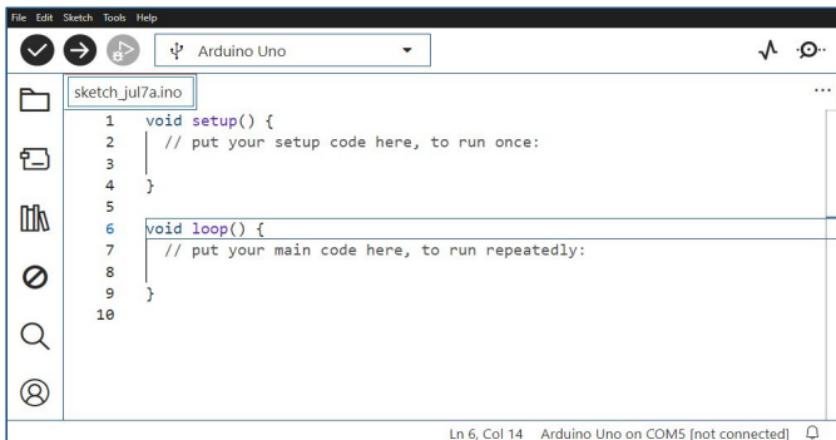
For this project you'll need to use version **2.0x of the Arduino IDE**. If you aren't sure what version you're using, go to the menu bar and select **Arduino > About Arduino**.

To download the Arduino IDE, go to the following URL:

[arduino.cc/en/software](https://www.arduino.cc/en/software)

# Install the ESP2866 software package

When you open the Arduino IDE, you'll see a window that looks like the one below:



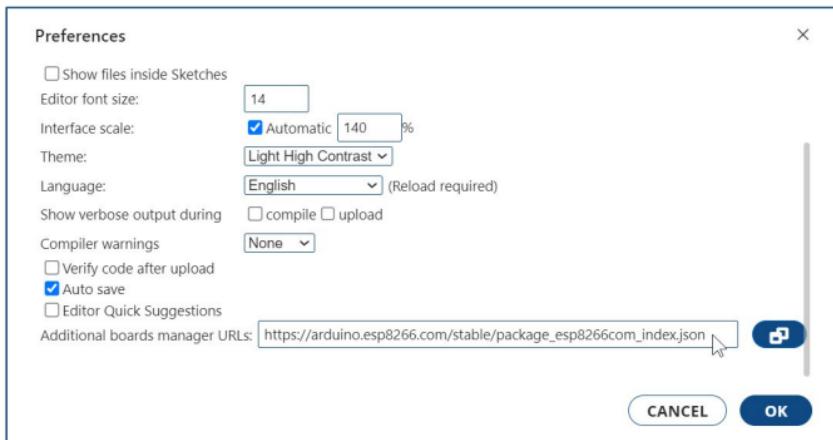
Before you write any code, you'll need to install a package with the software required to program the ESP8266 chip.

In the menu bar, select **File > Preferences**.



Add the following URL (all one line) to the list of **Additional Boards Manager URLs**, then click **OK**:

[https://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](https://arduino.esp8266.com/stable/package_esp8266com_index.json)

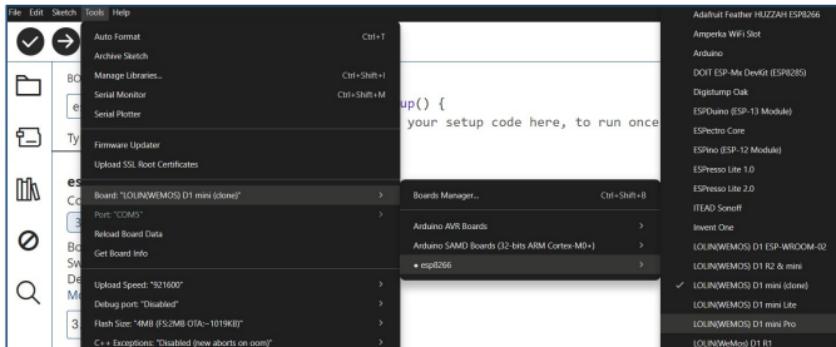


Next you'll install the software package you need to program the ESP8266 chip. To access the Boards Manager, select the second icon on the left-hand vertical menu, or go to **Tools > Board > Boards Manager...**

Type "**esp8266**" to find the package, then click **Install**.



Now, go to **Tools > Board > ESP8266 Boards** and select **LOLIN(WEMOS) D1 mini (clone)** OR **LOLIN(WEMOS) D1 mini Pro**, depending on your board.



# Select a serial port

Next you'll use the Arduino IDE to select a serial port, which your computer will use to transfer data to your Wemos D1 mini. Start by connecting your Wemos D1 mini

to your computer with a microUSB cable.

Some microUSB cables have a data connection, while others supply power but can't transfer data. If you have trouble with this step, try using a different micro USB cable.

*Note: If you plug your board into a different USB port, you'll need to repeat this step.*

## **Steps for Windows:**

In **Tools > Port**, you may see a list of options such as COM3, COM4, etc. Make a mental note of the ports you see, or write them down.

Connect your Wemos D1 mini to your computer, then go back to **Tools > Port**. Look for a COM port

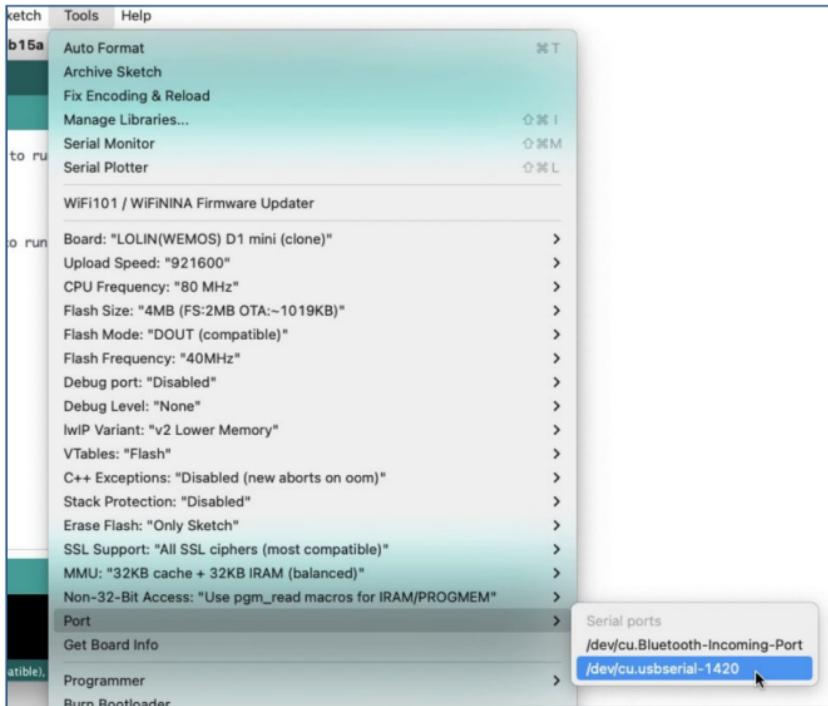
that wasn't there before, and select that one.

If that doesn't work, open the **Windows Device Manager**. Look for "unknown devices" and update your USB driver.

### **Steps for Mac & Linux:**

Connect your Wemos D1 mini to your computer using a micro USB cable. Next you'll select the serial port your computer will use to

communicate with your board. In the menu bar, go to **Tools > Port** and select **/dev/cu.usbserial-1420**. The number at the end will probably be different for you.



# Install the CH341 driver

If you're using Windows or a Mac from mid-2018 or earlier, you'll need to install a driver for the **CH341 chip** on your Wemos D1 mini. The CH341 is the chip that translates data from your computer's USB connection to the serial connection on the ESP8266.

Visit the following URL to download the CH341 driver:

[www.wemos.cc/en/latest/ch340\\_driver.html](http://www.wemos.cc/en/latest/ch340_driver.html)

If you're using a Mac and you aren't sure whether to install the CH341 driver, it's better to wait and see if you need it. Move on to **Test 1 (below)** and try to upload some code to your board. If it doesn't work, you can come back to this step and install the driver.

If you're using Linux, try the next step without installing the driver and see if it works. If you get an error, come back to this step and install the following driver:

[github.com/raashidmuhammed/  
esp8266](https://github.com/raashidmuhammed/esp8266)

Once the driver is installed, quit the Arduino IDE and reopen it. This may or may not be necessary, but it can't hurt.



## Create a wi-fi access point with an SSID name



You're now ready to program your Wemos D1 mini clone board!

Let's start with a super simple program to create a wi-fi access point. The following code will broadcast a network name, or

SSID (service set identifier), that you'll be able to see when you look for Wi-Fi networks on your laptop or phone.

You can also find the example code here:

[github.com/pocket-portal/code](https://github.com/pocket-portal/code)

Paste the code below into the Arduino IDE:

```
#include<ESP8266WiFi.h>

void setup()
{ pinMode(LED_BUILTIN,OUTPUT);
WiFi.mode(WIFI_AP); WiFi.softAP(
"Pocket Portal Power Play");
}

void loop()
{ digitalWrite(LED_BUILTIN,HIGH);
delay(1000);digitalWrite(LED_BUIL
TIN, LOW);delay(1000);
}
```

The example above will create a network called "**Pocket Portal**

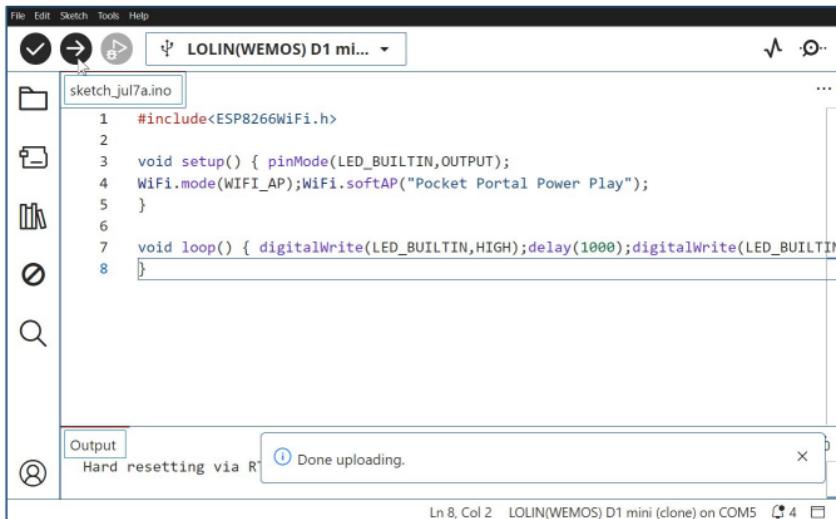
**Power Play".** To change the SSID name, edit the text between double quotes in the **WiFi.softAP()** function.

Your network's SSID name can be up to 32 characters long, but in practice you should stick to 31 characters. That's because some devices use null-terminated strings for SSID names.

Save your sketch by going to **File > Save**, or hitting **[Ctrl] + [S]**. Choose a name for your project, and leave the save location as it is.

Next, click **Upload** to start uploading the program to your board.

When the upload is complete, you'll see "**Done uploading**" below the code window.



```
sketch_jul7a.ino
1 #include<ESP8266WiFi.h>
2
3 void setup() { pinMode(LED_BUILTIN,OUTPUT); WiFi.mode(WIFI_AP);WiFi.softAP("Pocket Portal Power Play"); }
4
5
6 void loop() { digitalWrite(LED_BUILTIN,HIGH);delay(1000);digitalWrite(LED_BUILTIN
7
8 }
```

Output  
Hard resetting via R Done uploading.

Ln 8, Col 2 LOLIN(WEMOS) D1 mini (clone) on COM5 4

Now, look at the available Wi-Fi networks on your devices. You should see the one you just created! (If you try to connect, nothing will happen.)



## **Captive portal approach with LittleFS storage**



Time for the fun stuff: hosting a captive portal on your board!

## **Install the LittleFS Upload Plugin**

In order to upload HTML files, images, etc. to the ESP8266 chip, you'll first need to install a plugin. Go to the releases page and click **arduino-littlefs-upload.vsix** to download the latest release:

[github.com/earlephilhower  
/arduino-littlefs-upload/releases](https://github.com/earlephilhower/arduino-littlefs-upload/releases)

Save the file to

`~/ .arduinoIDE/plugins/` on Mac

and Linux or `C: \Users \<user>`

\.arduinoIDE\plugins\ on

Windows (you may need to create the plugins directory yourself).

Restart the Arduino IDE. To see if the plugin is working, press [**Ctrl**] + [**Shift**] + [**P**] (Windows) / [**⌘**] + [**Shift**] + [**P**] (Mac) to open the Command Palette, then type "**Upload LittleFS to Pico / ESP8266 / ESP32**":

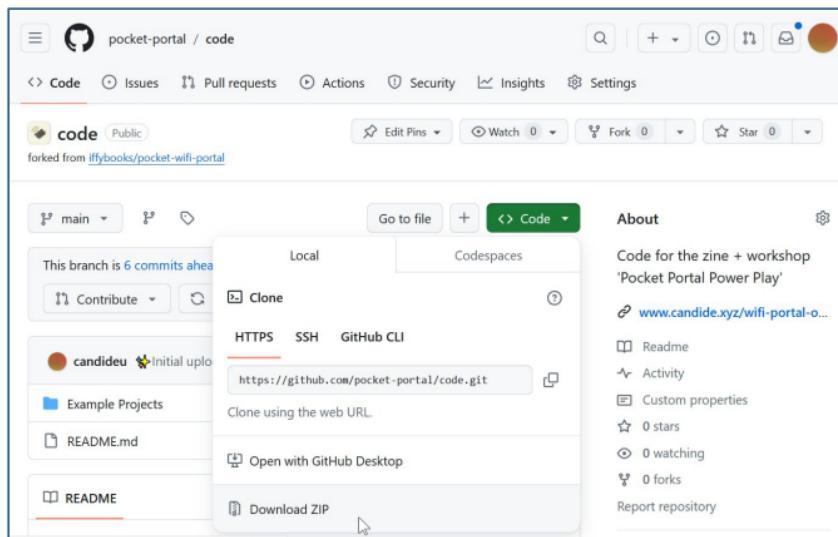


## Upload files with plugin

It's now time to create a page and upload it to your board. To help, I've provided some example projects and samples to get you started. To download them, visit:

[github.com/pocket-portal/code](https://github.com/pocket-portal/code)

# Click the green **Code** button, then click **Download ZIP**.



# Find the download in your Finder/File Explorer (**code-main.zip**) and unzip it.

You'll find several project folders, each containing the Arduino code for the server (`.ino` sketch file, `.DNSServer.cpp`, and `.DNSServer.h`) as well as a data folder containing the static site files (HTML, CSS, JS, etc.).

Open the “Example Projects” folder, and move its contents to your Arduino IDE’s sketchbook folder. To find it, go to **Arduino >**

**Preferences** and make a note of or copy the path under "**Sketchbook location.**" Go to the Finder/File Explorer and locate the sketchbook directory. Here's where you can expect to find it:

**Linux:** /Users/<user>/Arduino/

**macOS:**

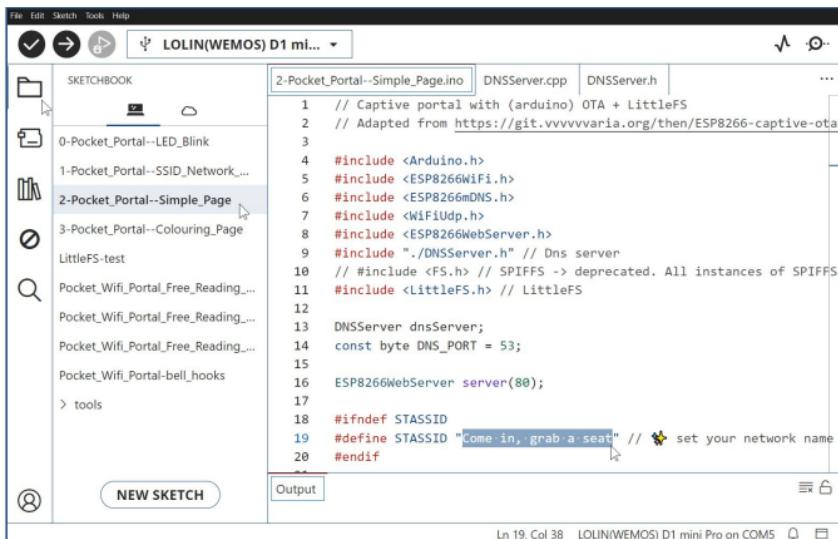
/Users/<user>/Documents  
/Arduino/

# Windows:

C:\Users\<user>\Documents

\Arduino\

The projects will now appear in the IDE. Double-click to open one.



The screenshot shows the Arduino IDE interface. The left sidebar lists projects in the SKETCHBOOK folder, including "0-Pocket\_Portal--LED\_Blink", "1-Pocket\_Portal--SSID\_Network\_...", "2-Pocket\_Portal--Simple\_Page" (which is currently selected), "3-Pocket\_Portal--Colouring\_Page", "LittleFS-test", "Pocket\_Wifi\_Portal\_Free\_Reading\_...", "Pocket\_Wifi\_Portal\_Free\_Reading\_...", "Pocket\_Wifi\_Portal\_Free\_Reading\_...", "Pocket\_Wifi\_Portal-bell\_hooks", and a "tools" section. At the bottom left is a "NEW SKETCH" button. The main area is a code editor with the following content:

```
1 // Captive portal with (arduino) OTA + LittleFS
2 // Adapted from https://git.vvvvvvaria.org/then/ESP8266-captive-ota
3
4 #include <Arduino.h>
5 #include <ESP8266WiFi.h>
6 #include <ESP8266DNS.h>
7 #include <WiFiUdp.h>
8 #include <ESP8266WebServer.h>
9 #include "./DNSServer.h" // Dns server
10 // #include <FS.h> // SPIFFS -> deprecated. All instances of SPIFFS
11 #include <LittleFS.h> // LittleFS
12
13 DNSServer dnsServer;
14 const byte DNS_PORT = 53;
15
16 ESP8266WebServer server(80);
17
18 #ifndef STASSID
19 #define STASSID "Come in, grab a seat" // 🌟 set your network name
20 #endif
```

The status bar at the bottom right shows "Ln 19, Col 38 LOLIN(WEMOS) D1 mini Pro on COM5".

To set the SSID name, open the `.ino` file in the IDE, find the `STASSID` macro and edit its value between double quotes:

```
#define STASSID "Your Network Name"  
// ✨ set your network name here ✨
```

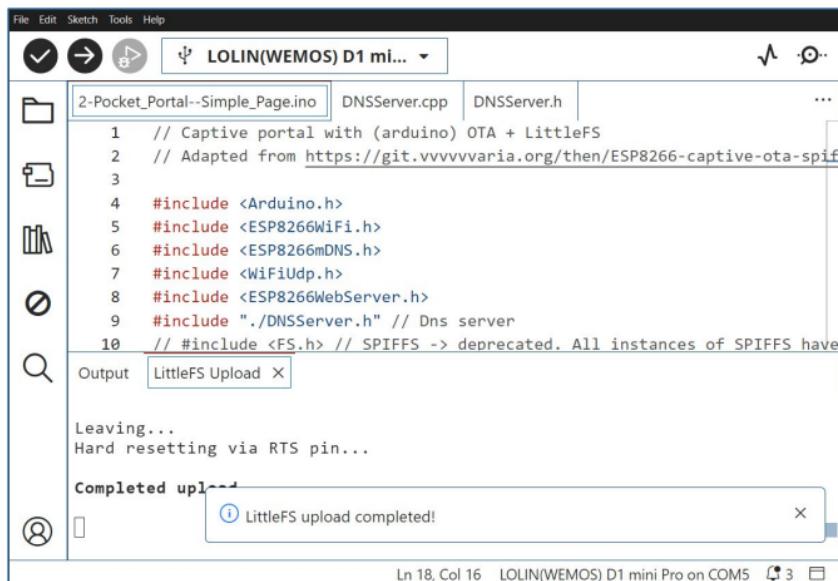
It's now time to upload the example to your board using the plugin! Press **[Ctrl] + [Shift] + [P]** (Windows) / **[⌘] + [Shift] + [P]** (Mac) to open the Command

# Palette, then type "Upload LittleFS to Pico / ESP8266 / ESP32":



This plugin will look for a directory called **data** in the project directory and upload its contents to LittleFS storage on the ESP8266.

When the upload is finished, you'll see the message "**LittleFS upload completed!**" beneath the code editor.



```
File Edit Sketch Tools Help
LOLIN(WEMOS) D1 mi... ▾
2-Pocket_Portal--Simple_Page.ino DNSServer.cpp DNSServer.h ...
1 // Captive portal with (arduino) OTA + LittleFS
2 // Adapted from https://git.vvvvvaria.org/then/ESP8266-captive-ota-spif
3
4 #include <Arduino.h>
5 #include <ESP8266WiFi.h>
6 #include <ESP8266mDNS.h>
7 #include <WiFiUdp.h>
8 #include <ESP8266WebServer.h>
9 #include "./DNSServer.h" // Dns server
10 // #include <FS.h> // SPIFFS -> deprecated. All instances of SPIFFS have
Output LittleFS Upload X
Leaving...
Hard resetting via RTS pin...
Completed upl
LittleFS upload completed!
Ln 18, Col 16 LOLIN(WEMOS) D1 mini Pro on COM5 3
```

Click the **Upload** button in the top left to compile and upload the project code to your Wemos D1 mini.



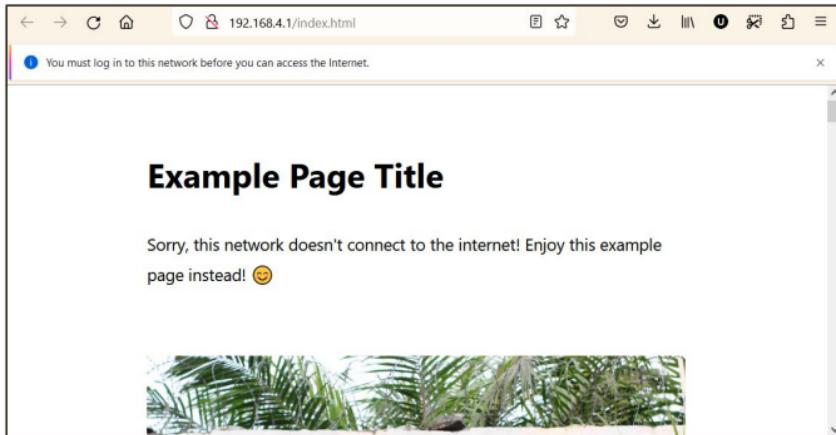
Once complete, you'll see **Done uploading** beneath the code editor.

Now, open your device's list of available networks and look for the

SSID you set earlier. When you connect, you should see a pop-up window with the web page content from the **data** folder.

In other words, you now have yourself a captive portal!

Here's an example of what the Simple Page looks like:



To make your own captive portal content, edit the **data** folder files in the code editor of your choice. The **HTML** file `index.html` will always load first.

# Tips & Tricks

1. 4 devices can be connected to the portal at the same time, and the limit can be increased up to 8 (see code examples).
2. Optimize your media: Scale it down. Compress it. Transcode it. Use an efficient file format.
3. Make sure to remove all spaces in the file names of your media.

4. Remember the twist: your portal doesn't actually connect to Wi-Fi! Make sure all your media (stylesheets, media, fonts, JavaScript, etc.) is hosted on the board, and that you're pointing to a local/relative file.
5. Use your pocket portal for good! Consider the impact of the message you're broadcasting on your local community.

# Further Ideas

1. Not comfortable using a code editor to create web pages?  
Find HTML / CSS / JS confusing? Not to worry!  
There are many ways of generating static web pages, and many tools offer an export to HTML function:
  - a. Word Processors and note-taking apps (MS Word, LibreOffice Writer, OnlyOffice, Notion,

Obsidian + “Export Vault to HTML” plugin, etc.)

b. WYSIWYG Prototyping apps like Penpot’s code export, GrapeJS

2. Don’t want to use a plugin?  
No worries! It’s also possible to write HTML in the Arduino IDE sketch, although there are limitations. Refer to page 16 of Iffy Books’ zine for more info on this approach:  
[iffybooks.net/pocket-wifi-portal/](http://iffybooks.net/pocket-wifi-portal/)

3. Want to host your web pages online instead? No problem! You can host the static site files in the **data** folder on platforms such as Netlify Drop, Gitlab Pages, GitHub Pages, etc.

4. Consider ways of advertising and prompting people to connect to your captive portal:

a. **Wi-Fi QR codes** are readable by iOS 11+ and Android 10+ devices.

You can create Wi-Fi QR codes on sites such as this one: [ianharris.io/wifi-qr-web/](https://ianharris.io/wifi-qr-web/)

- b. **NFC tags** are able to quickly connect tap-enabled devices to a network. Apps like NFC Tools (Android) can write data to tags.

You've now made your own

Pocket Portal Power Play! 

Version 2

Published May 4th 2024

License: CC0 1.0 Universal

