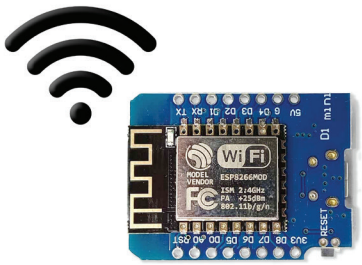
Spread ideas with a pocket wi-fi portal



a how-to guide from

319 N. 11th St. #3D
Philadelphia, PA 19107

Join our email list
at iffybooks.net

Follow @iffybooks
on social media

Send corrections to
iffybooks@protonmail.com

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 "Free Reading Wi-fi." When you connect, you get a popup 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, and it'll easily run for a couple days.

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

- ➔ Share an article/essay/political slogan with anyone who happens to be at the coffee shop.
- ➔ Promote an upcoming event without using social media.
- ➔ Share a poetry anthology with other commuters on your train.
- ➔ Share maps and information on a hiking trip without cell reception.
- ➔ Use several wi-fi boards to send a message using SSID names alone.

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.

You don’t need to know how to program to complete this tutorial! Knowing some HTML will help, but it isn’t necessary. You'll start by installing the **Arduino IDE** (integrated development environment), which is the application you'll use to program your Wemos D1 mini.

➡️ Install the Arduino IDE (macOS)

For this project you'll need to use **version 1.8.x 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 and click **Mac OS X**:

https://www.arduino.cc/en/software

Open the .dmg file you just downloaded and drag the **Arduino IDE** application file to your Applications folder.

➡️ Install the Arduino IDE (Windows)

Go to the following URL and download the Arduino IDE for WIndows. Follow the instructions to install it on your machine.

https://www.arduino.cc/en/software

➡️ Install the Arduino IDE on Debian-based Linux (e.g., Ubuntu)

Installation method 1: Snap package manager

Open a terminal window and run the following command to install the Arduino IDE using the Snap package manager. You'll need to enter your password, and it will take a few minutes to install.

```
sudo snap install arduino
```

Now run the following command to add your username to the dialout group. This will let you program the board via USB.

```
sudo usermod -a -G dialout $USER
```

Restart your computer.

Open a terminal window and run the command **`arduino`** to start the Arduino IDE.

Installation method 2: APT package manager

Depending on the version of Ubuntu you're using, you may be able to install Arduino using the APT package manager. Simply type the following command in a terminal window and press enter. You'll need to enter your password, and it will take a few minutes to install.

```
sudo apt-get install arduino
```

Now run the following command to add your username to the dialout group. This will let you program the board via USB.

```
sudo usermod -a -G dialout $USER
```

Run the command **`arduino`** to start the Arduino IDE.

Installation method 3: Installer Script

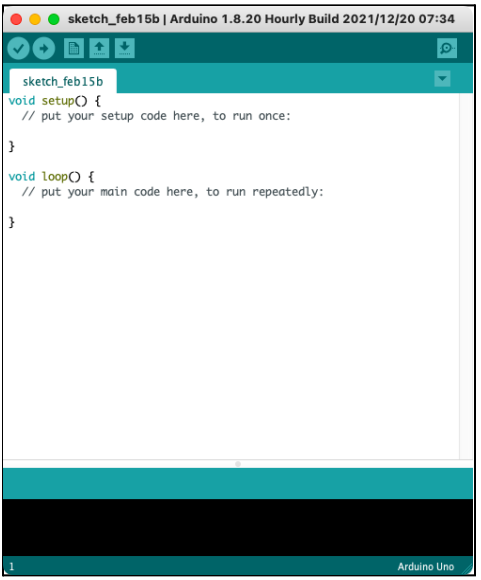
Go to the following URL and follow the steps to install the Arduino IDE using the installer script from **https://arduino.cc**:

https://linoxide.com/how-to-install-arduino-ide-on-ubuntu-20-04

Don't use sudo with this installer script!

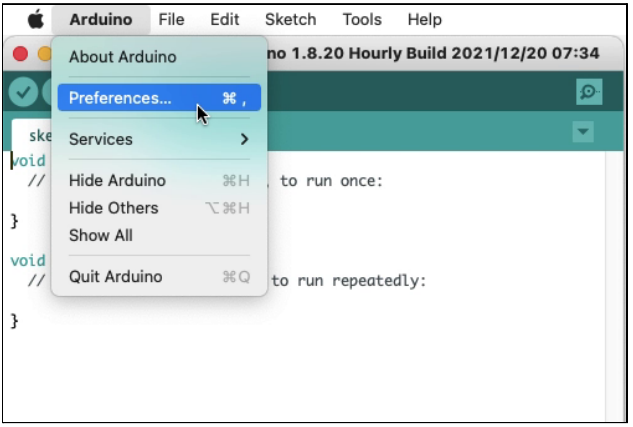
➡️ Install the ESP8266 software package

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



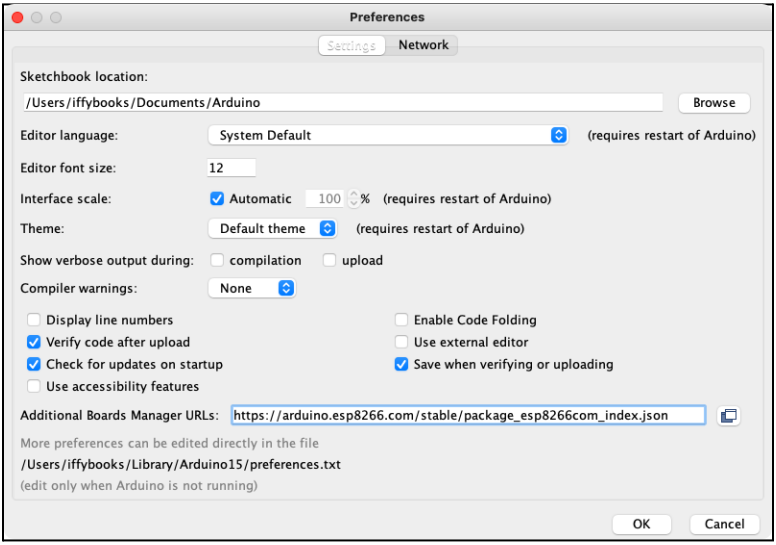
Before you write any code, you'll need to install a package with the software required to program the ESP8266 chip.

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

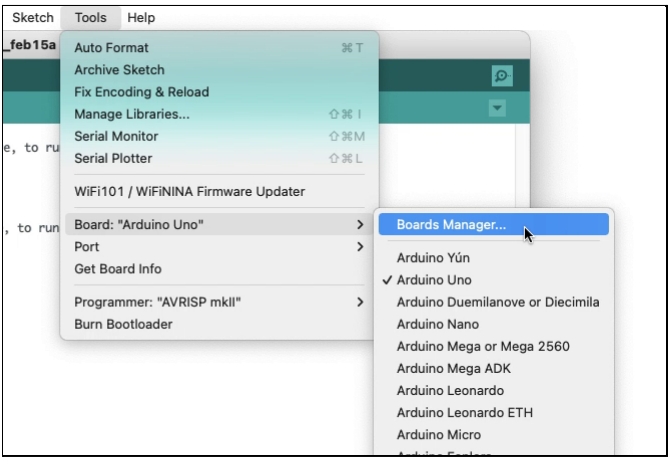


Add the following URL (all one line) to the list of **Additional Boards Manager URLs**, then click **OK**.

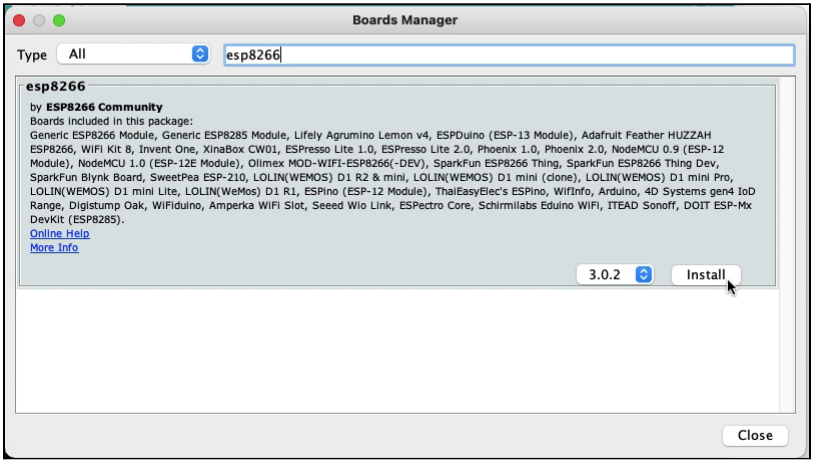
`https://arduino.esp8266.com/stable/package_esp8266com_index.json`



Next you'll install the software package you need to program the ESP8266 chip. In the menu bar, go to **Tools > Board > Boards Manager...**.

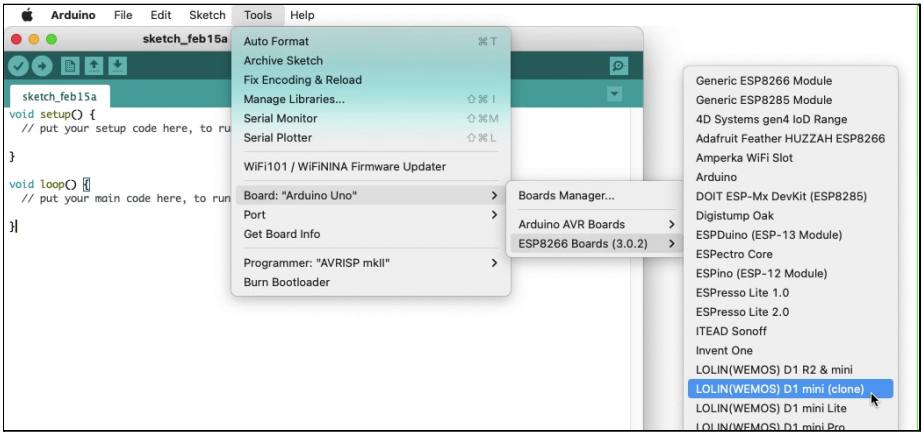


Type **"esp8266"** to find the package, then click **Install**.



When installation is finished, click **Close**.

Now go to **Tools > Board > ESP8266 Boards** and select **LOLIN(WEMOS) D1 mini (clone)**.



➡️ Select a serial port

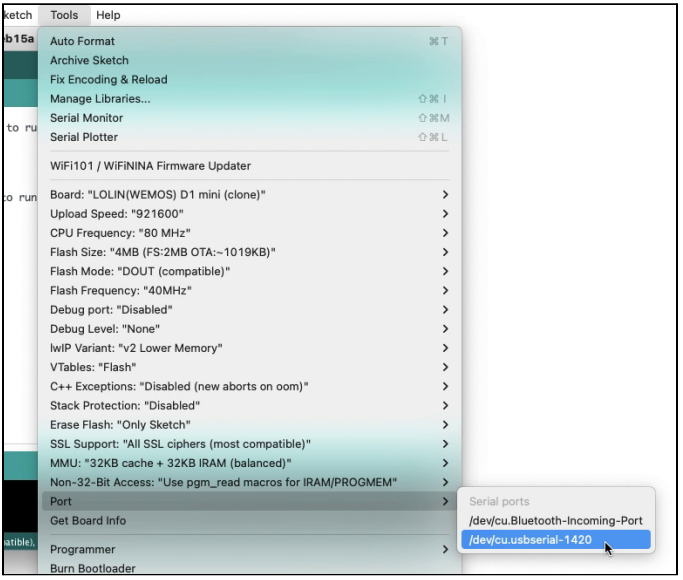
Next you'll use the Arduino IDE to select a serial port, which your computer will use to transfer data to your Wemos D1 mini. Start by connecting your Wemos D1 mini to your computer with a microUSB cable.

Some micro USB 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 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.



Steps for Windows:

Go to **Tools > Port** and you'll see a menu titled **Serial Ports** with 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 and Driver Installation** menu. Look for "unknown devices" and update your USB driver.

➡️ 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.

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.

Visit the following URL to download the CH341 driver:

https://www.wemos.cc/en/latest/ch340_driver.html

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:

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.

Now you're ready to program your Wemos D1 mini clone board. Nice work!

Test 1: Make the LED blink

Let's start with a super simple program, just to make sure you're able to connect to the board and program it. You can find the example code for this project at one of the following URLs:

https://iffybooks.net/pocket-wifi-portal
https://github.com/iffybooks/pocket-wifi-portal

Copy the code below into the Arduino IDE:

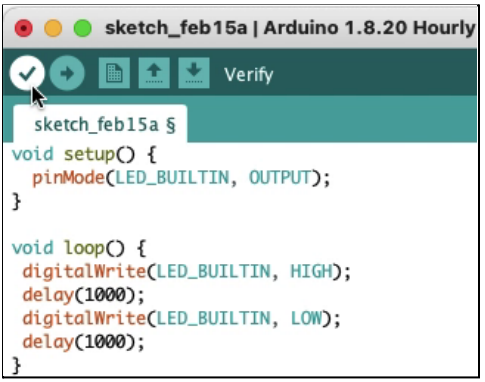
```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

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

The code above defines two functions: **`setup()`** and **`loop()`**. The **`setup()`** function runs first, then the **`loop()`** function runs repeatedly as long as the board is plugged in.

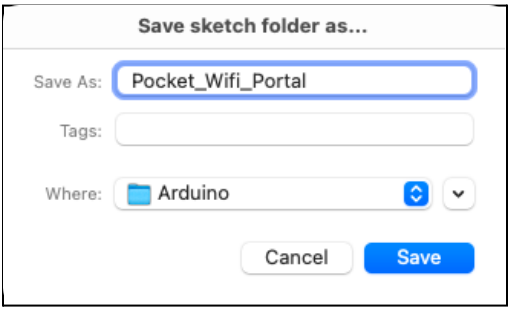
In the **`setup()`** function, **`pinMode(LED_BUILTIN, OUTPUT)`** prepares the board to use the built-in LED.

Next, the **`loop()`** function makes the LED blink repeatedly. It first uses **`digitalWrite(LED_BUILTIN, HIGH)`** to turn the LED **on**. Next, the function **`delay(1000)`** pauses your code for 1000 milliseconds, or 1 second. The function **`digitalWrite(LED_BUILTIN, LOW)`** turns off the LED, then **`delay(1000)`** pauses for one second.

To compile your code, click the **Verify** button in the top left corner.

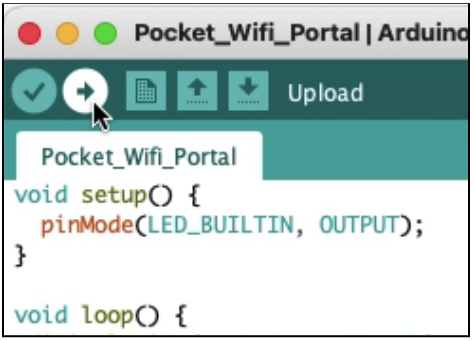


Choose a name for your project, then click **Save**. Don't change the save location.

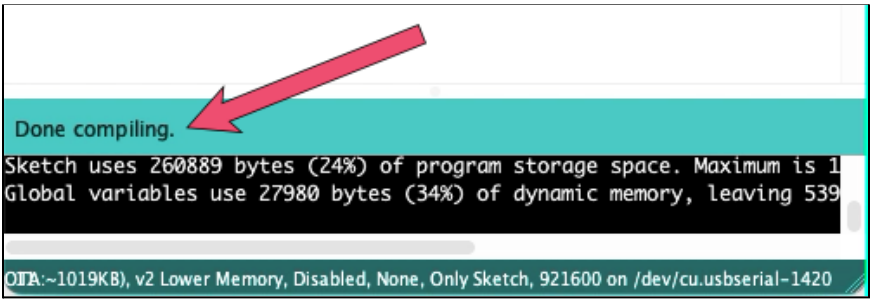


Note: The Arduino IDE may be unresponsive for 10–20 seconds while it compiles your code.

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.



The LED on your board should start blinking on and off. Pretty cool!

Note: If you're using Linux or an older Mac and you weren't able to upload your code, go back to the previous section and install the CH341 driver.

Test 2: Create a wi-fi access point with an SSID name

Next you'll run a test to make sure you can create a wi-fi access point. The code below 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.

In the Arduino IDE, update your code to match the example below. You'll need to add the three lines in bold.

```
#include <ESP8266WiFi.h>

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  WiFi.mode(WIFI_AP);
  WiFi.softAP("Free Reading Wifi");
}

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

To change the SSID name, edit the text between double quotes in the **WiFi.softAP()** function. The example above will create a network called "**Free Reading Wifi**." You can change the SSID if you want.

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.

Click the **Upload** icon to upload the program to your Wemos D1 mini. When it's finished, look at the available wi-fi networks and you should see the one you just created. (If you try to connect, nothing will happen.)

Captive portal approach 1: Write HTML in the Arduino IDE

Now let's make a captive portal! With this approach, you'll store the complete HTML code for your web page as a string variable in your program. The advantages are that all your code will be in one place, and you won't need any extra plugins.

However, this approach has some limitations. You'll only be able to create a single page, without any links. You can use images, but only ones that are 20 KB or smaller. And you'll only be able to post short documents, a few thousand words or less.

If you know what you're doing and you just want to build a more complex site, it's OK to skip this section!

➡️ Create a minimal example page

Here's the code for a minimal captive portal page, which simply displays the word "Hello!" The important parts, for the purposes of this tutorial, are in bold.

```
#include <ESP8266WiFi.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>

const byte DNS_PORT = 53;
IPAddress apIP(172, 217, 28, 1);
DNSServer dnsServer;
ESP8266WebServer webServer(80);

String responseHTML = "<!DOCTYPE html>"
"<html>"
"<head>"
"<title>A Minimal Example Page</title>"
"</head>"
"<body>"
"<p>Hello!</p>"
"</body>"
"</html>";

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  WiFi.mode(WIFI_AP);
  WiFi.softAPConfig(apIP, apIP, IPAddress(255,255,255,0));
  WiFi.softAP("Minimal Example Wifi");
  dnsServer.start(DNS_PORT, "*", apIP);
  webServer.onNotFound([]() {
    webServer.send(200, "text/html", responseHTML);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
    digitalWrite(LED_BUILTIN, HIGH);
  });
  webServer.begin();
}

void loop() {
 if (WiFi.softAPgetStationNum() == 0)
 {
    delay(100);
 } else {
    delay(100);
    dnsServer.processNextRequest();
    webServer.handleClient();
 }
}
```

Go to the following URL, then highlight and copy the code:

https://github.com/iffybooks/pocket-wifi-portal/blob/main/03_Captive_portal_approach_1/Minimal_Portal.ino

Paste the code into the Arduino IDE window, replacing your previous code. Click the **Verify** button at the top left to make sure it compiles.

Connect your Wemos D1 mini to your computer, then click **Upload**. When the upload is complete, you'll see the message "Done uploading" at the bottom of the code editor.

On a laptop or phone, open your wi-fi menu. You should see a network called **Minimal Example Wifi**.

Click/tap **Minimal Example Wifi** to connect to the network. If you wait a moment, a pop-up window should appear with the word "Hello!"

Here's the code snippet (repeated from above) that contains HTML code for the captive portal page. It creates a string variable called **responseHTML** that begins with "**<!DOCTYPE html>**" and ends with "**</html>**".

```
String responseHTML = "<!DOCTYPE html>"
"<html>"
"<head>"
"<title>A Minimal Example Page</title>"
"</head>"
"<body>"
"<p>Hello!</p>"
"</body>"
"</html>";
```

Note that each line has double quotes at the beginning and end, and that there's a semicolon after the final line. That's the C/C++/Arduino syntax for defining a string over multiple lines. The double quotes aren't part of the HTML code.

Trying making a few edits in the HTML! To change the page title, update the text between **<title>** and **</title>**. And add some new text between the paragraph tags ** ** and ** **.

```
String responseHTML = "<!DOCTYPE html>"
"<html>"
"<head>"
"<title>Bashō Wifi</title>"
"</head>"
"<body>"
"<p>old pond<br />frog leaping<br />splash</p>"
"</body>"
"</html>";
```

The example above uses the **
** tag to indicate line breaks.

Connect a Wemos D1 mini to your computer and click **Upload** in the top left corner of the window. When the upload is finished, use your phone or computer to connect to the wi-fi network you just created.

➡️ Make a page with CSS formatting

So far, you've created a web page that looks pretty plain. Here's some example code that will make a more modern-looking page. (For the moment, don't worry about copying it into your IDE.)

```
String responseHTML = "<!DOCTYPE html>"
"<html lang='en'>"
"<head>"
"<meta charset='utf-8'>"
"<meta name='viewport'
   content='width=device-width,
   initial-scale=1'>"
"<title>Free Reading Wi-fi Portal</title>"
"<style>"
"p {font-family: Georgia, serif; font-size:
   200%; text-align: center; padding-top:
   50px;}"
"div {width: 75%; margin: 0 auto;}"
"</style>"
"</head>"
"<body bgcolor='lightpink'>"
"<div>"
"<p>\"What we do is more important than what
   we say or what we say we believe.\" —bell
   hooks</p>"
"</div>"
"</body>"
"</html>";
```

Because double quotes have a special meaning to the C/C++ compiler, you can't use ordinary double quotes in your HTML code. For HTML attributes you can use single quotes, as in the following line:

```
"<html lang='en'>"
```

Or you can insert a backslash (\) before double quotes to escape them:

```
"<p>\"What we do is more important than
   what we say or what we say we believe.\"
   –bell hooks</p>"
```

➡️ Add a Base64-encoded image

To include an image in a web page, you'd ordinarily use an **<img>** tag that points to an image file. The **src** attribute specifies the images's location using either a URL or a relative path to a file on the same server. Here's an example of the latter:

```
<img src='images/bell_hooks.png'>
```

Instead of pointing to an external file, you can encode an image as a Base64 string and include it in your HTML file. Here's an example of an <img> tag using a Base64 string for a very small image file (a single green pixel):

```
<img src='
KGgoAAAANSUhEUgAAAAEAAAABCAMAAAAoyzS7AA
AABlBMVEV/01gAAACf/yQfAAAADElEQVR42mJgA
AgwAAACAAFPbVnhAAAAAElFTkSuQmCC'
alt='one green pixel' />
```

Now we'll add the following image below the quote. It's a 948-byte PNG image of bell hooks, based on a photo from her Flickr page that she marked as public domain. You can find the uncropped original here:

https://www.flickr.com/photos/bellhooksphilosophy/36335440686

And here's a link to the file above:

https://github.com/iffybooks/pocket-wifi-portal/blob/main/03_Captive_portal_approach_1/bell_hooks.png

Here's the new HTML string, including the image:

```
String responseHTML = "<!DOCTYPE html>"
"<html lang='en'>"
"<head>"
"<meta charset='utf-8'>"
"<meta name='viewport'
content='width=device-width,
initial-scale=1'>"
"<title>Free Reading Wi-fi Portal</title>"
"<style>"
"p {font-family: Georgia, serif; font-size:
200%; text-align: center; padding-top: 50px;}"
"div {width: 75%; margin: 0 auto;}"
"</style>"
"</head>"
"<body bgcolor='lightpink'>"
"<div>"
"<p>\"What we do is more important than what
we say or what we say we believe.\" -bell
hooks</p>"
"<p><img
src='
EUgAAAHIAAAB/CAMAAAAEnWaRAAAABlBMVEX///8AAABVw
tN+AAADaUlEQVR42uyb647VMAyEx+//0ggdTjdNfRlfUoR
EhcSPXfLFjuOMnQB5/cP7Q6HJWP49cBCJz5/vlxysgPwNw
fpds8BJx67Q7Bio/OJmJXKrmUBidS12KOjhkF5IBbhF0Sg
```

```
S/se5NrV/gQkmylujzMSYV+9x1EWCJmLSsTwSI1aCJ8ZMj
BPD5cSsV/+YideR0kQiSwRGwgeDnqWsBCbNxAG/oh2xKHz
icBHNBrXvq8nesxLO2NEOQhmJErJhJV4Pn09yL63l7ldlC
sak7MEJB2h61Z/FNZViKlDXx51EsJhg/CqWnc5qF8MH2/m
8W+nGVxGJy9LvyLec5AZ0J61jK3BWs2GrXLMuouWWtnMeY
Sa7lVnk5Tlduz9/su2YRvaxdo6ChLaDMlYqc7/5VbNnSUd
FK6HJ2AdyOT3W6C4hzSPfLrw2o9OODQjxnsSMlTHTHR0nz
soLWkkFiXWkD0wemS3AzCr+lGOdcqiA7PYNUqrg4adSRc0
cXs4hWJGWSVGZXG01gNAk+nJeBaBJNPuI9tAEUiKkeIFTc
eyP0oPhBMkJrkTX/6F1rr+G9yURoUn1jD7R7CYYNSb6RNO
1HR0b7XvL4egjFRG51An79YlgAGnp2ZuankMitFxb1EyOV
VxIOFuthRrN0YpIGkZqhTyTgJDZHKZOmD6iH4W6cJmgi7y
7Nl8oVJF7Ne12tkz1jGyuI4nOyJk7r/sxfbadf5ckoaqbv
TL9cNUcs58eE44Nt8LWEMV55IkHG8zyYRbZvmsvChF5GXn
rJNWADSFSJk4+MfqPPIjEX7CSeIPzz1tJJT8ME9927O0gM
9/hYJgIUyqdQG6Vj5n7DyB/yj4R9SpyzPL7PcKlkxi1iR7
xUs22jrcjoIkUs40wcAA+kEuXjevP94iblUNPzCImck/aq
kTxWkFyGrlkdzBFDJDagpSd0nasVRpU7zZCpNalFHugeSK
cC8chZPrO64CRfjuEuwksNJ27zFHHUsguUfKLGb1TwXQAE
aoX4j4ssSieDWBIErSYtNRHPysYerQn7jG/w/apQ7/n54wv
E8/lwciW51Q2f5CGd2qRDugBjVLLaifZ6KtBAlK1xeomRS
4048f8vvp1SMqLZu3/xb/iJmP2Gg2AASSz2/SWYUFdcUce
U3privW59ML3IUH/jlwADAFRWI+4MU5SBAAAAAElFTkSuQ
mCC' alt='high-contrast photo of bell hooks'
/></p>"
"</div>"
"</body>"
"</html>";
```

You can find the complete code at the following link:

https://github.com/iffybooks/pocket-wifi-portal/blob/main/03_Captive_portal_approach_1/Quote_Portal.ino

FInd an image that's 20KB or less, convert it to a Base64 string, and try adding it to your web page.

If you do a web search for the phrase "convert image to base64," you'll find lots of sites that can help.

Or open a terminal window and use a **base64** command. Here's an example, using a PNG file in the **Desktop** folder.

```
base64 ~/Desktop/bell_hooks.png
```

✤⦿❑▸❖◯▿▫✲◎▲

Captive portal approach 2: Use SPIFFS storage

▲◎✲▫▿◯❖▸❑⦿✤

➡️ Install SPIFFS uploader 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 **ESP8266FS-0.5.0.zip** to download the latest release:

`https://github.com/esp8266/arduino-esp8266fs-plugin/releases`

When the download is finished, unzip the file to create a directory called **ESP8266FS**.

Go to **Arduino > Preferences** and make a note of 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:

macOS: **`/Users/your_username/Documents/Arduino/`**
Windows: **`C:\Users\your_username\Documents\Arduino\`**
Linux: **`/Users/your_username/Arduino/`**

In the sketchbook directory, look for a directory called **tools**. If you don't see one, you can create it yourself.

Move the **ESP8266FS** directory from your Downloads folder to the **tools** directory.

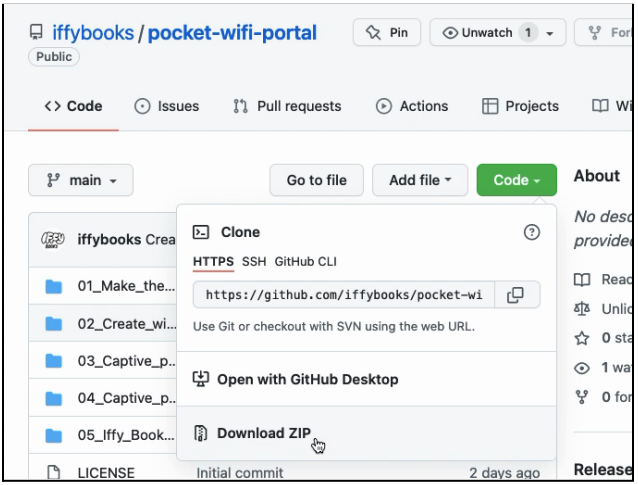
Quit the Arduino IDE and reopen it. In the Tools menu, you should now see **ESP8266 Sketch Data Upload** as one of the options.

➡️ Upload files with plugin

To download the code you'll need, go to the following URL:

`https://github.com/iffybooks/pocket-wifi-portal`

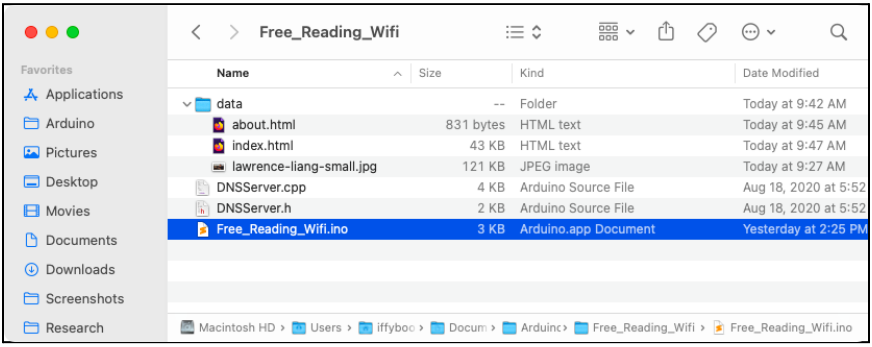
Click the green **Code** button, then click **Download ZIP**.



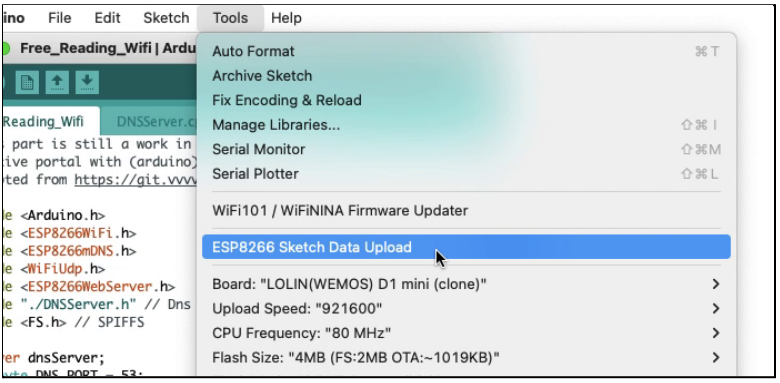
Find the file you just downloaded in your Finder/File Explorer (**pocket-wifi-portal-main.zip**) and unzip it. Open the directory.

In a new Finder/File Explorer window, open your Arduino sketchbook directory. In the directory **pocket-wifi-portal-main/04_Captive_portal_approach_2**, you'll find a directory called **Free_Reading_Wifi**. Drag the directory **Free_Reading_Wifi** into your Arduino sketchbook folder.

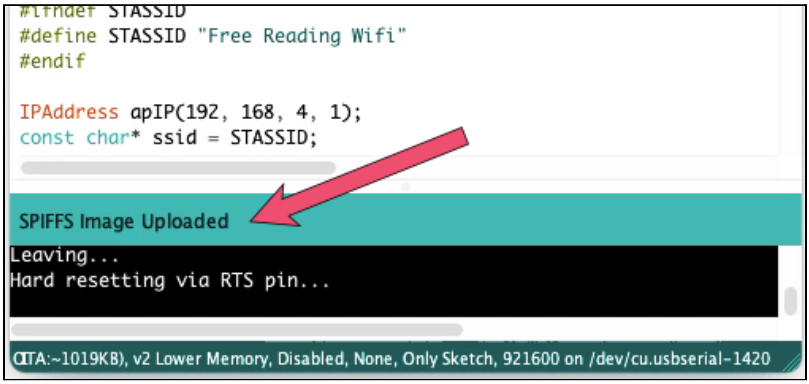
Open the directory **Free_Reading_Wifi** and find the file **Free_Reading_Wifi.ino**. Double click it to open the project in the Arduino IDE.



In the menu bar, select **Tools > ESP8266 Sketch Data Upload**. This plugin will look for a directory called **data** in the project directory and upload its contents to SPIFFS storage on the ESP8266.



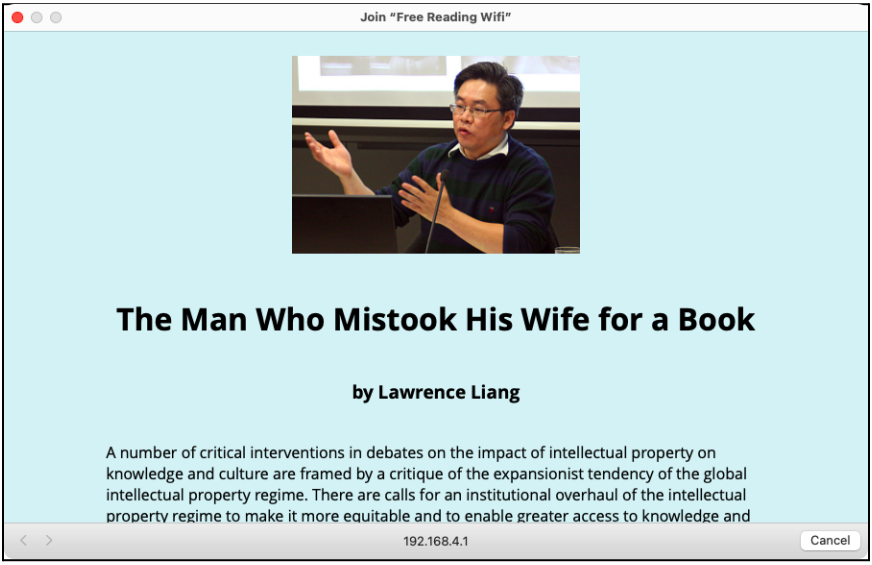
When the upload is finished, you'll see the message **SPIFFS Image Uploaded** beneath the code editor.



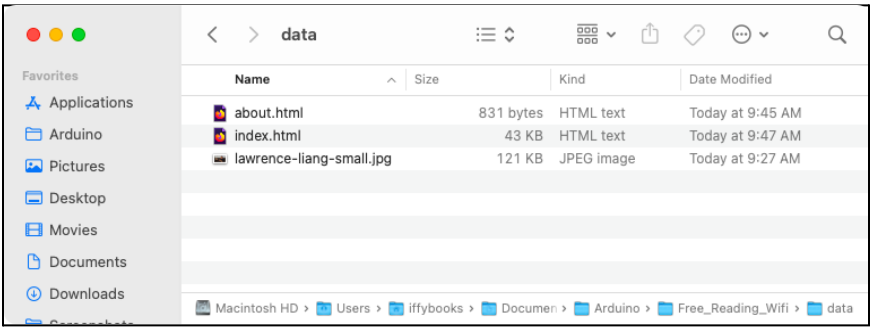
Click the **Upload** button in the top left to compile and upload the project code to your Wemos D1 mini.

When the upload is done, you'll see **Done uploading** beneath the code editor.

Look for the wi-fi network **Free Reading Wifi**. When you connect, you should see a popup window that looks like this:



To make your own captive portal, you can edit the files in the **data** directory. The HTML file **index.html** will always load first.



Here's what an **<img>** tag looks like. Note that the path begins with a slash (/).

```
<img src='/lawrence-liang-small.jpg'
alt='photo of Lawrence Liang speaking at a
podium' width='300' height='206'/>
```

And here's a link to another page:

```
<a href='/about.html'>About Lawrence Liang</a>
```

Have fun!

➡️ General tips

Don't make a page that says "You've been pwned!" or "Your computer is now infected with a virus." You'll scare people for no reason.

Don't use an SSID that's the same or similar to one from a specific business or institution. Calling your network "Starbucks wifi," for example, could potentially get you in legal trouble.

➡️ Credits

Method 1 is based on the following 2 tutorials:

- https://gist.github.com/Cyclenerd/7c9cba13360ec1ec9d2ea36e50c7ff77
- http://melissamerritt.epizy.com/wifi-joke/wifi-joke.html

Method 2 is based on this project:

- https://git.vvvvvvaria.org/then/ESP8266-captive-ota-spiffs

Published February 2022

IFFY BOOKS

Philadelphia, Pennsylvania

Version 0.6

Download this zine as a PDF:
https://iffybooks.net/pocket-wifi-portal