IoT Blink

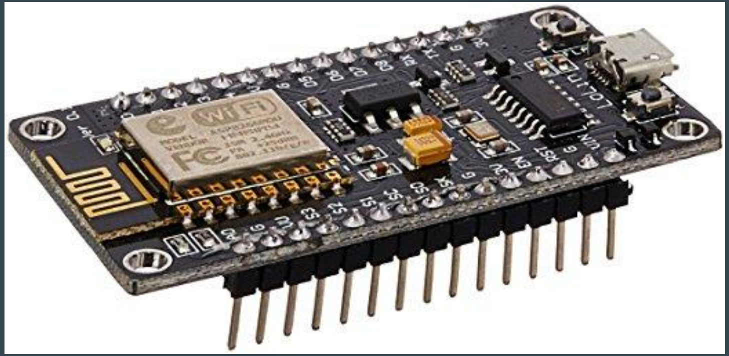
Making physical things controlled by the web

Arduino

- Low powered microcontroller
  - A bit like a Raspberry Pi but only runs one thing
- Used to add computing to electronics
- Open source

ESP8266

- Cheap and powerful chip that provides IoT functionality
  - Connect to WiFi
  - Create access points
  - Manage over the air updates
  - Host a (small) server
  - Make HTTP requests



Setup

Getting the IDE

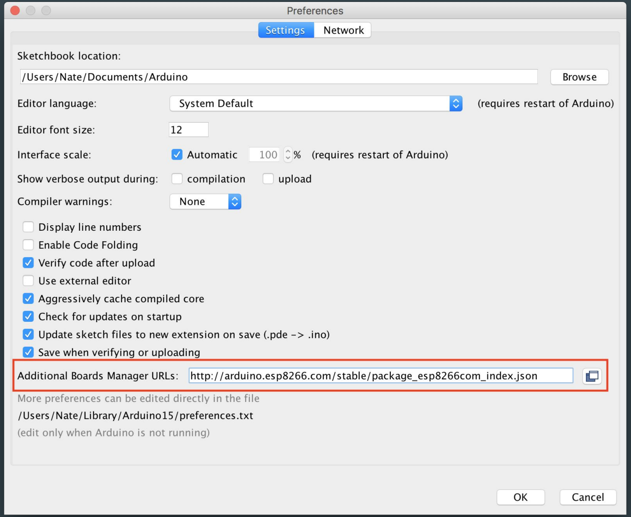
- Available from Arduino website
    - Online version exists
- Comes with tools needed to build and upload code

https://www.arduino.cc/en/Main/Software

Adding Boards

- In preferences, enter the URL to ESP8266 boards

http://arduino.esp8266.com/stable/package_esp8266com_index.json



Sketchbook location: /Users/Nate/Documents/Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

Compiler warnings: None

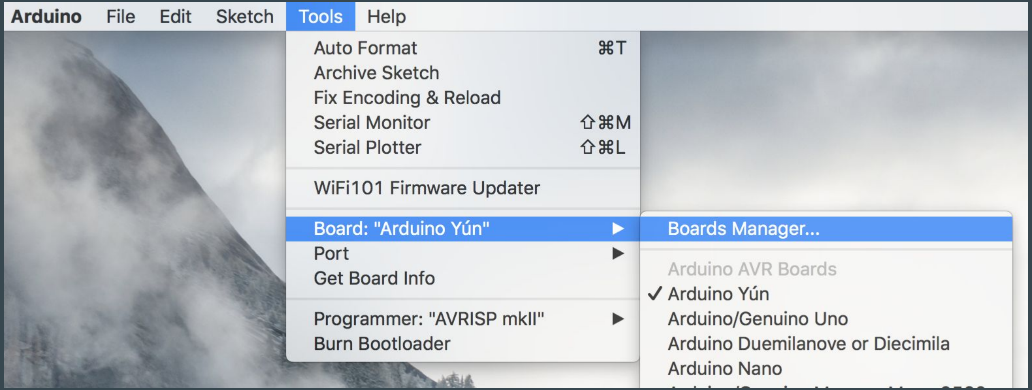
- [ ] Display line numbers
- [ ] Enable Code Folding
- [✓] Verify code after upload
- [ ] Use external editor
- [✓] Aggressively cache compiled core
- [✓] Check for updates on startup
- [✓] Update sketch files to new extension on save (.pde -> .ino)
- [✓] Save when verifying or uploading

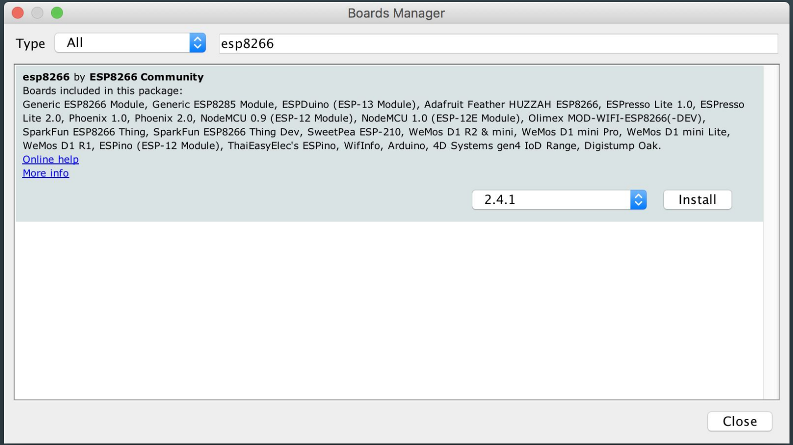
Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file
/Users/Nate/Library/Arduino15/preferences.txt
(edit only when Arduino is not running)

Adding Boards

- Open boards manager
  - Tools > Board > Boards Manager
- Search for ESP8266
- Click to install





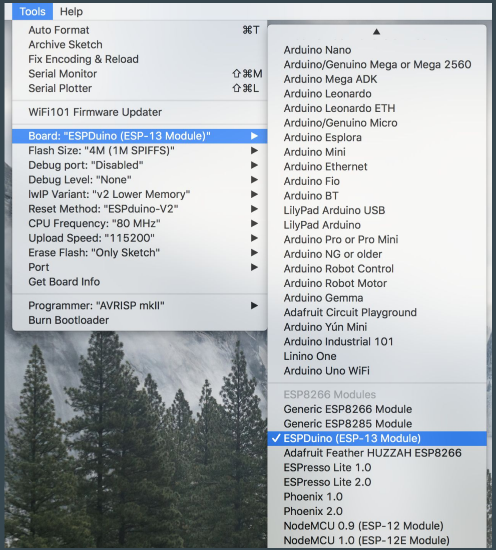
Type: All — esp8266

esp8266 by **ESP8266 Community**
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SweetPea ESP-210, WeMos D1 R2 & mini, WeMos D1 mini Pro, WeMos D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak.

Version: 2.4.1

Selecting Board

- Select “ESPDuino (ESP-13 Module)” from the boards list
  - Tools > Board
- **If the docs for your ESP8266 says to choose a different, do that**
- Will add other menu items under tools



Tools Help

Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Serial Plotter ⇧⌘L

WiFi101 Firmware Updater

Board: "ESPDuino (ESP-13 Module)"
Flash Size: "4M (1M SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
Reset Method: "ESPduino-V2"
CPU Frequency: "80 MHz"
Upload Speed: "115200"
Erase Flash: "Only Sketch"
Port
Get Board Info

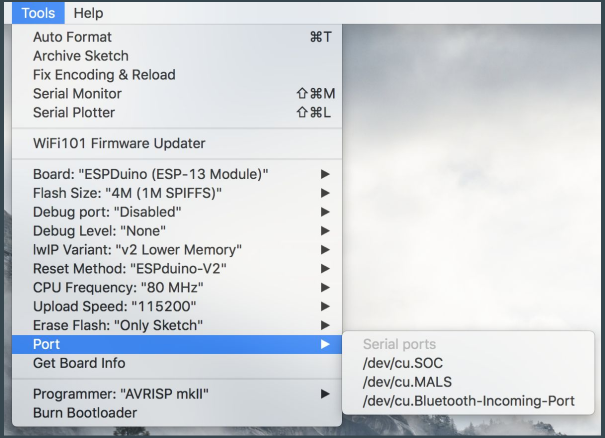
Programmer: "AVRISP mkII"
Burn Bootloader

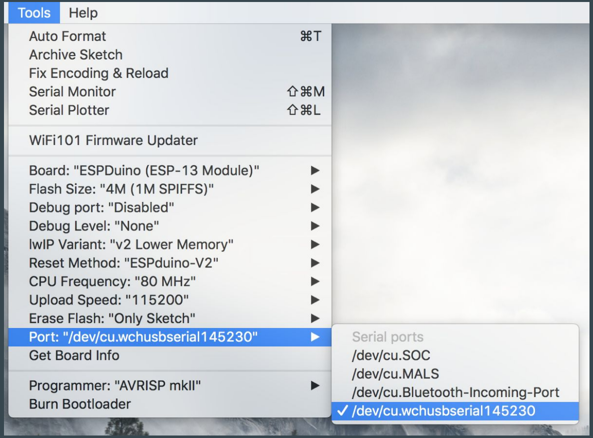
Arduino Nano
Arduino/Genuino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino/Genuino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma
Adafruit Circuit Playground
Arduino Yún Mini
Arduino Industrial 101
Linino One
Arduino Uno WiFi

ESP8266 Modules
Generic ESP8266 Module
Generic ESP8285 Module
✓ ESPDuino (ESP-13 Module)
Adafruit Feather HUZZAH ESP8266
ESPresso Lite 1.0
ESPresso Lite 2.0
Phoenix 1.0
Phoenix 2.0
NodeMCU 0.9 (ESP-12 Module)
NodeMCU 1.0 (ESP-12E Module)

Selecting Port

- Note items listed under port with board unplugged
  - Tools > Port
- Plug in, then select the new one
- May need to run as sudo to access under Linux





Basic Blink

- Blinking is a physical hello world
- Most Arduino boards have an onboard LED
- Examples under File > Example > ESP8266

```
const int led = 2;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, LOW);
  delay(300);
  digitalWrite(led, HIGH);
  delay(300);
}
```

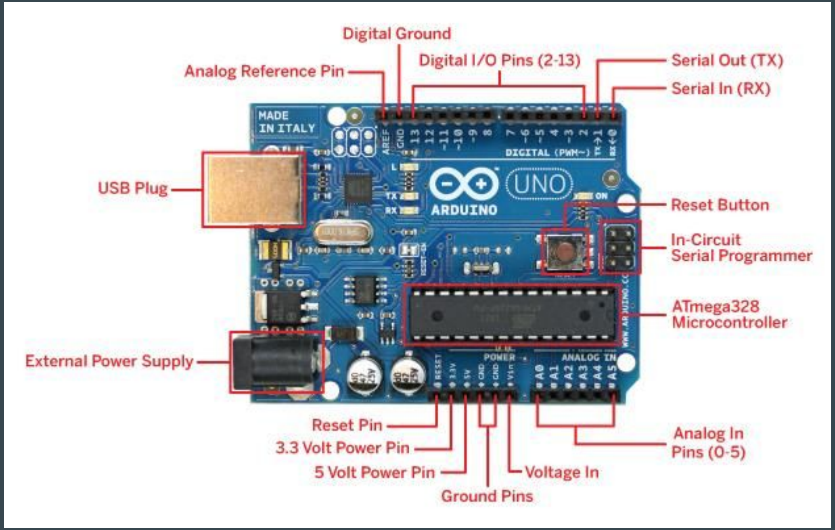
Basic Blink

- `setup` and `loop` run automatically
- `OUTPUT`, `LOW` and `HIGH` are defined for us
- `LED_BUILTIN` should map to the onboard LED pin, but mine was different
  - Hence the variable

```
const int led = 2;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, LOW);
  delay(300);
  digitalWrite(led, HIGH);
  delay(300);
}
```

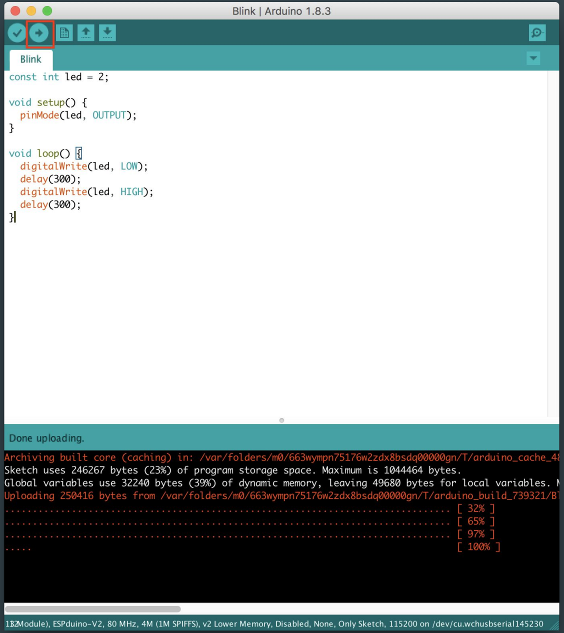
Pins

- Plugs for electronic bits
- Can be input, output, power or ground
  - Input: button, sensor, switch
  - Output: LED, motor, speaker
- Digital
  - On or off
- Analog
  - Amount of current (mostly)
- Can also transfer data



Upload Code

- Click upload button to build and upload code



```
const int led = 2;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, LOW);
  delay(300);
  digitalWrite(led, HIGH);
  delay(300);
}
```

Done uploading.

```
Archiving built core (caching) in: /var/folders/m0/663wympn75176w2zdx8bsdq00000gn/T/arduino_cache_48
Sketch uses 246267 bytes (23%) of program storage space. Maximum is 1044464 bytes.
Global variables use 32240 bytes (39%) of dynamic memory, leaving 49680 bytes for local variables. M
Uploading 250416 bytes from /var/folders/m0/663wympn75176w2zdx8bsdq00000gn/T/arduino_build_739321/Bl
................................................................................ [ 32% ]
................................................................................ [ 65% ]
................................................................................ [ 97% ]
.....                                                                            [ 100% ]
```

13Module), ESPduino-V2, 80 MHz, 4M (1M SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on /dev/cu.wchusbserial145230

IoT Code

Blink Without Delay

- Using delay is blocking
  - Will prevent doing anything else while running
- We need to change that before handling requests
- This example is from File > Examples > ESP8266 > BlinkWithoutDelay

Blink Without Delay

```
int ledState = LOW;
unsigned long previousMillis = 0;
long blinkInterval = 1000;

void blinkLed() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= blinkInterval) {
    previousMillis = currentMillis;

    if (ledState == LOW)
      ledState = HIGH;
    else
      ledState = LOW;

    digitalWrite(led, ledState);
  }
}
```

Blink Without Delay

```
void loop() {
  blinkLed();
}
```

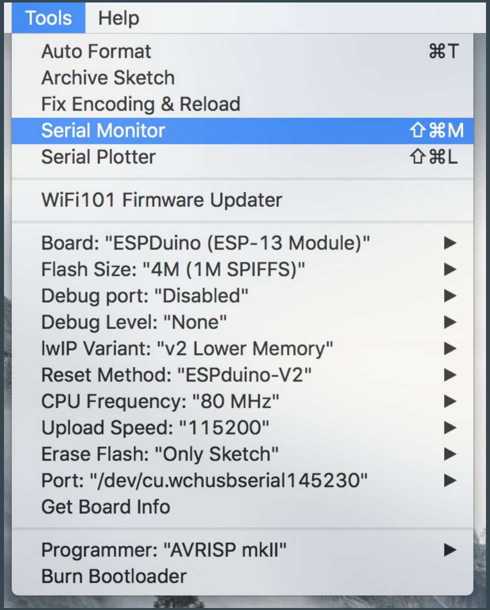
Connecting to WiFi

```
#include <ESP8266WiFi.h>

void connectToWiFi() {
  WiFi.begin("SSID", "PASSWORD");

  Serial.println("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  Serial.print("Local IP Address: ");
  Serial.println(WiFi.localIP());
}
```

Connecting to WiFi

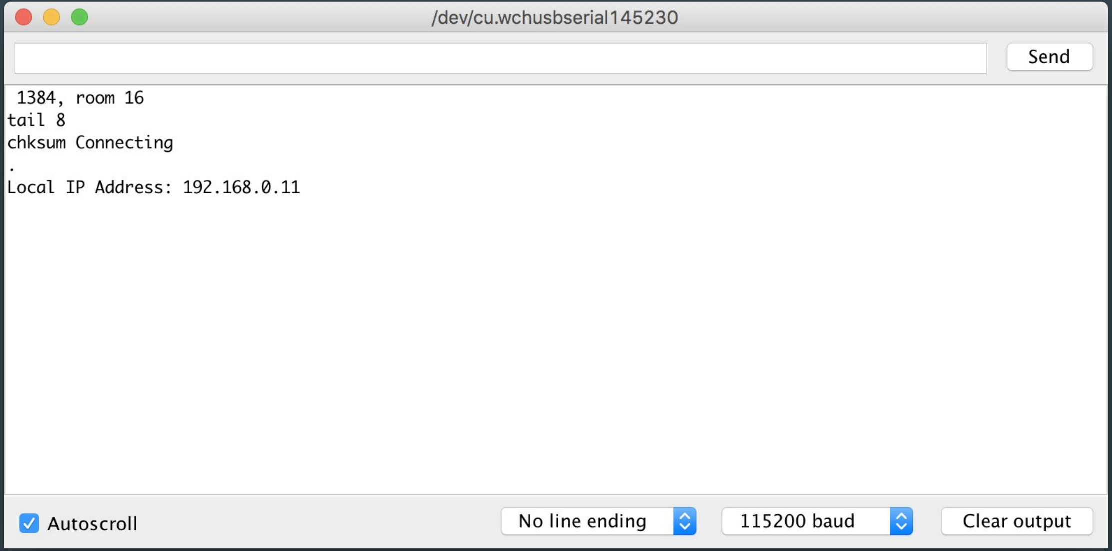
```
void setup() {
  Serial.begin(115200);
  connectToWiFi();
  // ...
}
```

Connect to WiFi



Board: "ESPDuino (ESP-13 Module)"
Flash Size: "4M (1M SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
Reset Method: "ESPduino-V2"
CPU Frequency: "80 MHz"
Upload Speed: "115200"
Erase Flash: "Only Sketch"
Port: "/dev/cu.wchusbserial145230"

Programmer: "AVRISP mkII"



```
 1384, room 16
tail 8
chksum Connecting
.
Local IP Address: 192.168.0.11
```

Autoscroll ✓
No line ending
115200 baud

Starting a Server

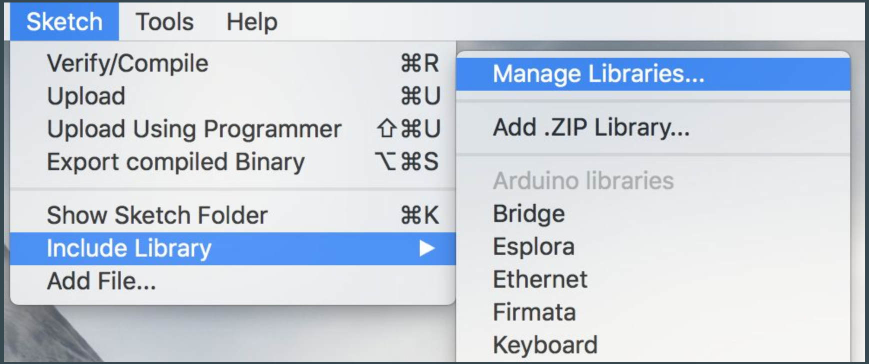
```
#include <ESP8266WebServer.h>

ESP8266WebServer server(80);

void setupServer() {
  server.on("/blink", HTTP_PATCH, handleBlink);
  server.begin();
}
```

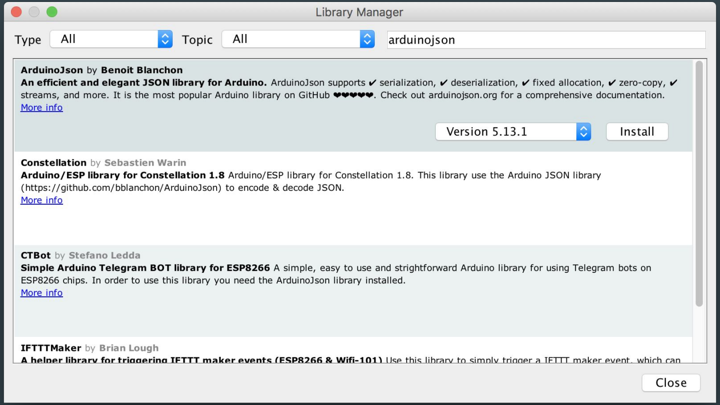
Starting a Server

```
void setup() {
  // ...
  setupServer();
}

void loop() {
  server.handleClient();
  // ...
}
```

Adding Libraries

- Need a library to use JSON
- Open Manage Libraries
  - Sketch > Add Library > Manage Libraries
- Search for "ArduinoJson" by Benoit Blanchon





ArduinoJson by **Benoit Blanchon**
An efficient and elegant JSON library for Arduino. ArduinoJson supports ✔ serialization, ✔ deserialization, ✔ fixed allocation, ✔ zero-copy, ✔ streams, and more. It is the most popular Arduino library on GitHub ❤❤❤❤❤. Check out arduinojson.org for a comprehensive documentation.
More info

Constellation by **Sebastien Warin**
Arduino/ESP library for Constellation 1.8 Arduino/ESP library for Constellation 1.8. This library use the Arduino JSON library (https://github.com/bblanchon/ArduinoJson) to encode & decode JSON.
More info

CTBot by **Stefano Ledda**
Simple Arduino Telegram BOT library for ESP8266 A simple, easy to use and strightforward Arduino library for using Telegram bots on ESP8266 chips. In order to use this library you need the ArduinoJson library installed.
More info

IFTTTMaker by **Brian Lough**
A helper library for triggering IFTTT maker events (ESP8266 & Wifi-101) Use this library to simply trigger a IFTTT maker event, which can

Blink Route

```
#include <ArduinoJson.h>

bool ledBlinking = false;

void handleBlink() {
  StaticJsonBuffer<200> jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(server.arg("plain"));

  if (root.containsKey("blink")) {
    ledBlinking = root["blink"];
  }

  if (root.containsKey("blinkRate")) {
    blinkInterval = root["blinkRate"];
  }

  server.send(200, "text/plain", "I have blinked\n");
}
```

Endpoint Functionality

```
void setup() {
  // ...
  if (ledBlinking) {
    blinkLed();
  }
}
```

Trying Out

- Compile and upload code
- Replace `$ESPIP` with the IP address

curl -H "Content-Type: application/json" -X PATCH -d '{ "blinkRate": 500, "blink": true }' $ESPIP/blink

Index Page

Sending HTML

- String
  - Easier for just HTML
  - `server.send(200, "text/html", "<h1>Hello<h1>");`
- SPIFFS
  - Nicer dev experience
  - Better memory management

Setup SPIFFS

- Download upload tool
    - https://github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.1.3/ESP8266FS-0.1.3.zip
- Extract Arduino tools directory
    - ~/Arduino/tools/ESP8266FS/tool/esp8266fs.jar
    - Arduino path may be in ~/Documents
    - Create `tools` dir if it doesn't exist
- Restart IDE
- Upload data item should be added to tools menu

http://esp8266.github.io/Arduino/versions/2.0.0/doc/filesystem.html

Create Index Route

```
void setupServer() {
  server.on("/", HTTP_GET, handleIndex);
  // ...
}
```

Sending Index

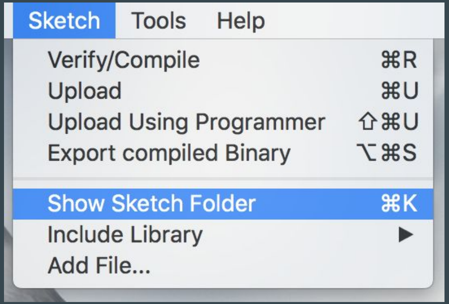
```
#include <FS.h>

void handleIndex() {
  File index = SPIFFS.open("/index.html", "r");
  if (!index) {
    server.send(500, "text/html", "Unable to serve page\n");
    return;
  }

  server.streamFile(index, "text/html");
  index.close();
}
```

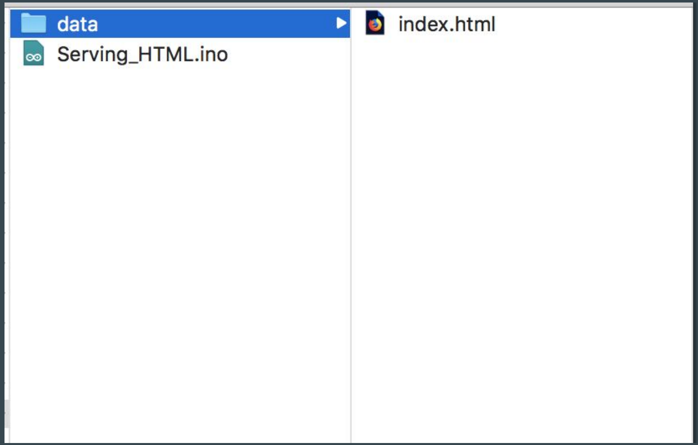
Start SPIFFS

```
void setup() {
  SPIFFS.begin();
  // ...
}
```

Creating the Index File

- Create `data` directory in source directory
  - Sketch > Show Sketch Folder
- This is where to save files to upload





HTML Contents

```html
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title>Blink</title>
</head>
<body>
  <h1>ESP8266 Blink</h1>
</body>
</html>
```

HTML Form

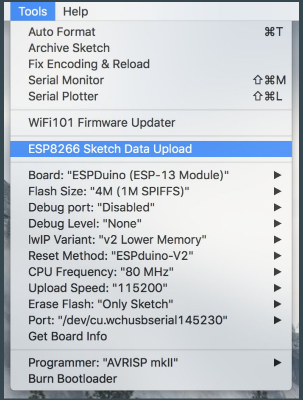
```html
<form id="blinkForm">
  <div>
    <label for="blinkRate">Blink Rate</label>
    <input type="number" name="blinkRate" id="blinkRate">
  </div>
  <div>
    <label for="shouldBlink">Should Blink</label>
    <input type="checkbox" name="shouldBlink" id="shouldBlink">
  </div>
  <div>
    <button type="submit">Update</button>
  </div>
</form>
```

Making Blink Request

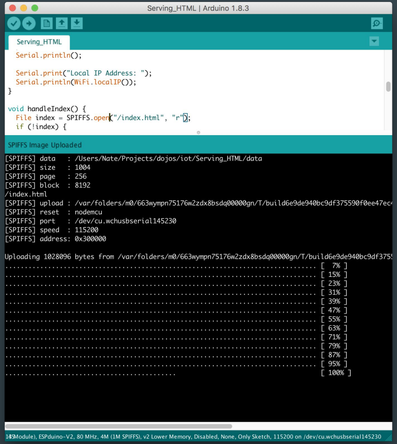
```
<script>
  document.querySelector('#blinkForm').addEventListener('submit', (e) => {
    e.preventDefault();

    const postData = {
      blink: document.querySelector('#shouldBlink').checked,
      blinkRate: document.querySelector('#blinkRate').value,
    };
    const headers = {
      method: 'PATCH',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(postData),
    };

    fetch('/blink', headers).then(res => console.log(res));
  });
</script>
```

Upload Index File

- In the IDE run the data uploader we installed
- Be sure to close the serial monitor
- You don't need to upload every time you upload your sketch
  - You also don't need to restart your board when you upload



Board: "ESPDuino (ESP-13 Module)"
Flash Size: "4M (1M SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
Reset Method: "ESPduino-V2"
CPU Frequency: "80 MHz"
Upload Speed: "115200"
Erase Flash: "Only Sketch"
Port: "/dev/cu.wchusbserial145230"
Programmer: "AVRISP mkII"



```
  Serial.println();

  Serial.print("Local IP Address: ");
  Serial.println(WiFi.localIP());
}

void handleIndex() {
  File index = SPIFFS.open("/index.html", "r");
  if (!index) {
```

```
[SPIFFS] data   : /Users/Nate/Projects/dojos/iot/Serving_HTML/data
[SPIFFS] size   : 1004
[SPIFFS] page   : 256
[SPIFFS] block  : 8192
/index.html
[SPIFFS] upload : /var/folders/m0/663wympn75176w2zdx8bsdq00000gn/T/build6e9de940bc9df375590f0ee47ec
[SPIFFS] reset  : nodemcu
[SPIFFS] port   : /dev/cu.wchusbserial145230
[SPIFFS] speed  : 115200
[SPIFFS] address: 0x300000

Uploading 1028096 bytes from /var/folders/m0/663wympn75176w2zdx8bsdq00000gn/T/build6e9de940bc9df3755
................................................................................ [  7% ]
................................................................................ [ 15% ]
................................................................................ [ 23% ]
................................................................................ [ 31% ]
................................................................................ [ 39% ]
................................................................................ [ 47% ]
................................................................................ [ 55% ]
................................................................................ [ 63% ]
................................................................................ [ 71% ]
................................................................................ [ 79% ]
................................................................................ [ 87% ]
................................................................................ [ 95% ]
............................................                                     [ 100% ]
```

14S Module), ESPduino-V2, 80 MHz, 4M (1M SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on /dev/cu.wchusbserial145230

Upload Sketch

- Upload sketch as usual

Useful Tips

- Examples come bundled
- Can create a .local domain
- Access point with a page to connect to WiFi
- You don’t have much storage or memory
- VSCode plugin

https://tttapa.github.io/ESP8266/Chap11%20-%20SPIFFS.html https://github.com/NRauh/IoT-Blink