SEICHE 2022

Introduction to Arduino

FINAL Lesson 10/11/12 – Smart LED Strips, Flashing software binaries, WLED

Instructor: Paul Frommeyer

www.paulfrommeyer.com

Corporate Sponsor: DXC Technology

GUESS WHAT?

YOU DON'T HAVE TO COPY ANYTHING TODAY!

Y'all are keeping your flash drives, and the only thing that has been updated is this presentation.

Updated files are in the Lesson 10-11-12 folder on your flash drives. *I have included the BOM for the lamp I made*.

Enjoy!

Lesson Plan Overview

Lesson 1 – Intro and Setup
[may require 2 classes]

- Introduction to class format
- Overview of lesson plan
- Presentation format (monitor, camera, screen, whiteboard)
- Inventory of Arduino kits
- Inventory of USB drives
- Installation of ESP32 serial port drivers (Windows only)
- Installation of Arduino software
- History of Computing

Lesson 2 and 3 – Laptop operation review

- Control panel/Settings location
- Home directories and folder hierarchy
- Arduino file locations
- Search functions
- Open questions and issues
- History of Computing redux

Lesson 4 and 5 – IDE essentials

- The boards manager
- Setting board type
- Installing Libraries
- Loading example sketches
- Finding and selecting the Arduino USB port
- Uploading a sketch to the microcontroller [Example Sketch#1]
- Basic Arduino sketch (program) structure
- Uploading your own sketch [Class Sketch #2]
- History of Telecommunications and Networking – Part 1

Lesson 6 – DC Electricity Basics and DMM Intro

- Quick review: DC vs AC electric currents
- DC current operation
- Voltage vs Amperage (current)
- Ohm's Law
- Series vs parallel circuits
- Introduction to basic DMM functions and usage
- Measuring voltage [DMM usage #1]
- History of Telecommunications and Networking – Part 2

Lesson 7 and 8 – Grounding, ESD, and Connecting LEDs

- Grounding
- The Phantom Menace of ESD
- Resistors!
- LED's [DMM usage #2, maybe]
- Powering LEDs – Ballast and dropping resistors
- Connecting LEDs to microcontrollers
- Parts of an Arduino Sketch
- ESP32 Specifications – Ampacity
- Let there be Light! – Class Sketch #2
- History of Telecommunications and Networking – Part 3

Lesson 9 – ESP32 External Power Sources [may require 2 classes]

- Capacitors!
- Microcontroller board power requirements
- Board power input options and grounding
- Components with different voltage requirements – level shifting
- External power circuit design

Lesson 10-11-12 – Smart LED Strips, Flashing Software

- External device power circuit design recap
- Level shifting recap
- WS2812B and Neopixel connections – Class Sketch #3
- Flashing software binaries
- Flashing WLED software to your board

AND THAT'S A WRAP!

Lesson 10/11/12 – Smart LED Strips

- External device power circuit design recap
- Smart LED recap
- Level shifting recap
- WS2812B and Neopixel connections
- Class Sketch #3
- Flashing software binaries
- Flashing software on Linux
- Flashing software on Windows
- Flashing WLED software
- Appendices: ESP8266 reference: Pinouts, flashing procedures

External and USB Power

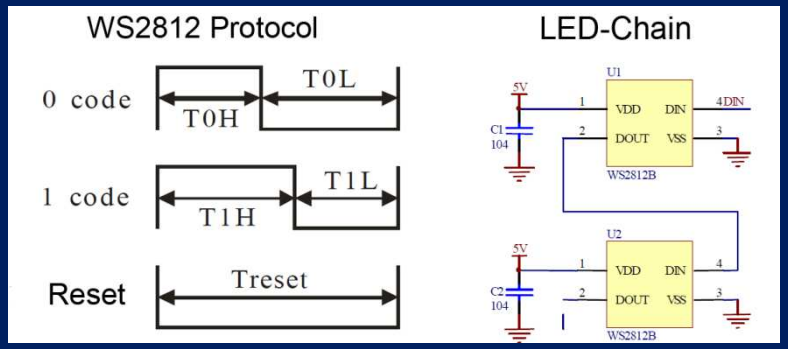
- Connecting USB and external power at the same time will frequently fry a board. This is especially true for the ESP-32, which uses lower capacity power regulation as it uses 3VDC.
- The easiest solution is (obviously) to only have one plugged in at a time.
- BEWARE!: As previously noted, when on USB power many boards will still supply 4.2VDC from the USB connector to the 5 volt rail. This allows peripherals to operate on USB power.
- That's not a problem for low current devices, but quantities of Neopixels will burn out the regulator, the laptop's USB chip, *or both*.
- The solution to this problem is to place a diode in-line between the 5 volt rail and the board's input power pin.
- The diode prevents power from "flowing back" from USB power to the peripherals.

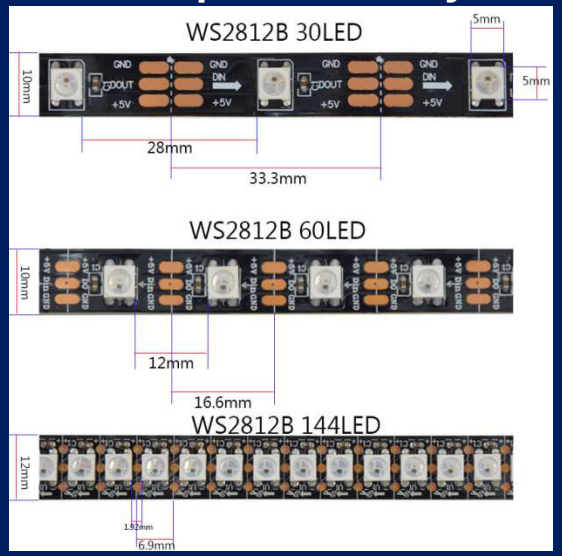
The WS2812B LED aka Neopixel

- The WS2812B is an (integrated) chip and 5050 package (flat, square) RGB LED
- An RGB LED can display 24 bits of color (0-255 red, 0-255 green, 0-255 blue)
- There are RGB LED controllers other than the WS2812B; discussion of those is outside the scope of our class
- “Neopixel” is a proprietary marketing term used by Adafruit Industries for their line of products based on the WS2812B
- RGB LEDs, and their chipsets, can be designed to run at 5VDC and 12VDC
- The chips of “Smart” LEDs make them individually controllable
- Control is by means of a one-wire transmit-only serial data protocol; the signal is directional with an input and an output for each pixel. Usually daisy-chained.

This is the WS2812B chip

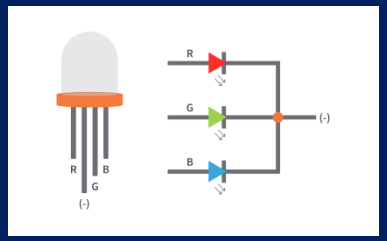
5050 RGB LED package





Types of WS2812B Strips

PIXELS ARE DIRECTIONAL!!!



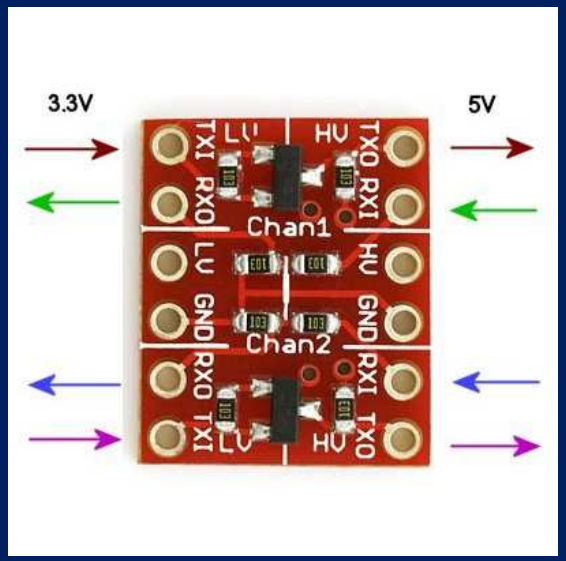
“Traditional” RGB LED (5mm package)

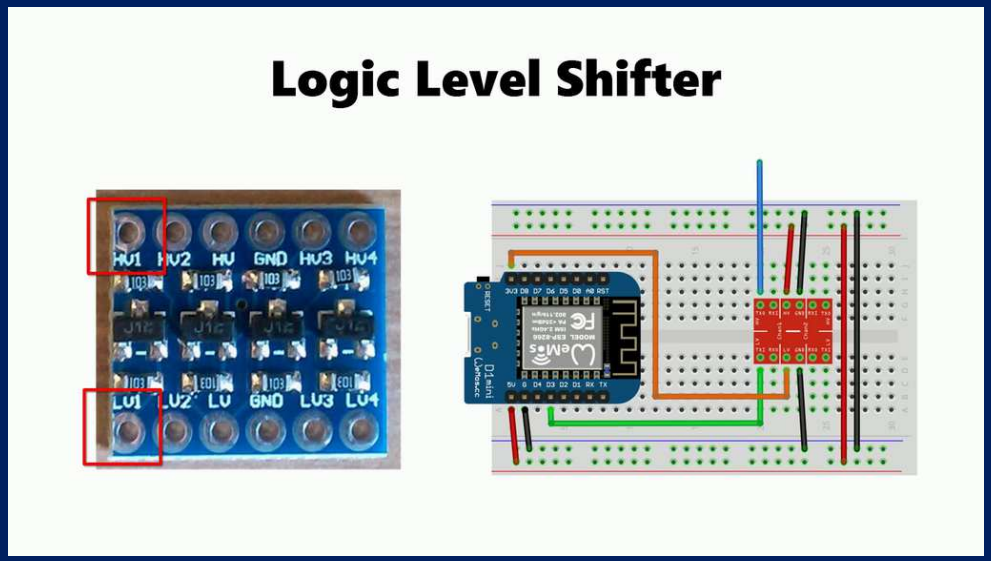
Different Voltage Requirements

- Not all peripherals will run at the same voltage as the microprocessor board you are using
- This results in different signal level tolerances. Not the power rails, *but the control signals themselves* operate at different voltages.
- Transmitting a 5 volt signal into a 3.3 volt device will damage or destroy the receiving device. This is as true for SPI as it is for traditional RS-232C TTL serial connections.
- Transmitting a 3 volt signal into a 5 volt device *sometimes* works. It depends on how well designed the receiving device is.
- The best-practice solution to this problem is to use devices called level shifters. Level shifters provide voltage translation between the two different signal levels.

Level Shifters

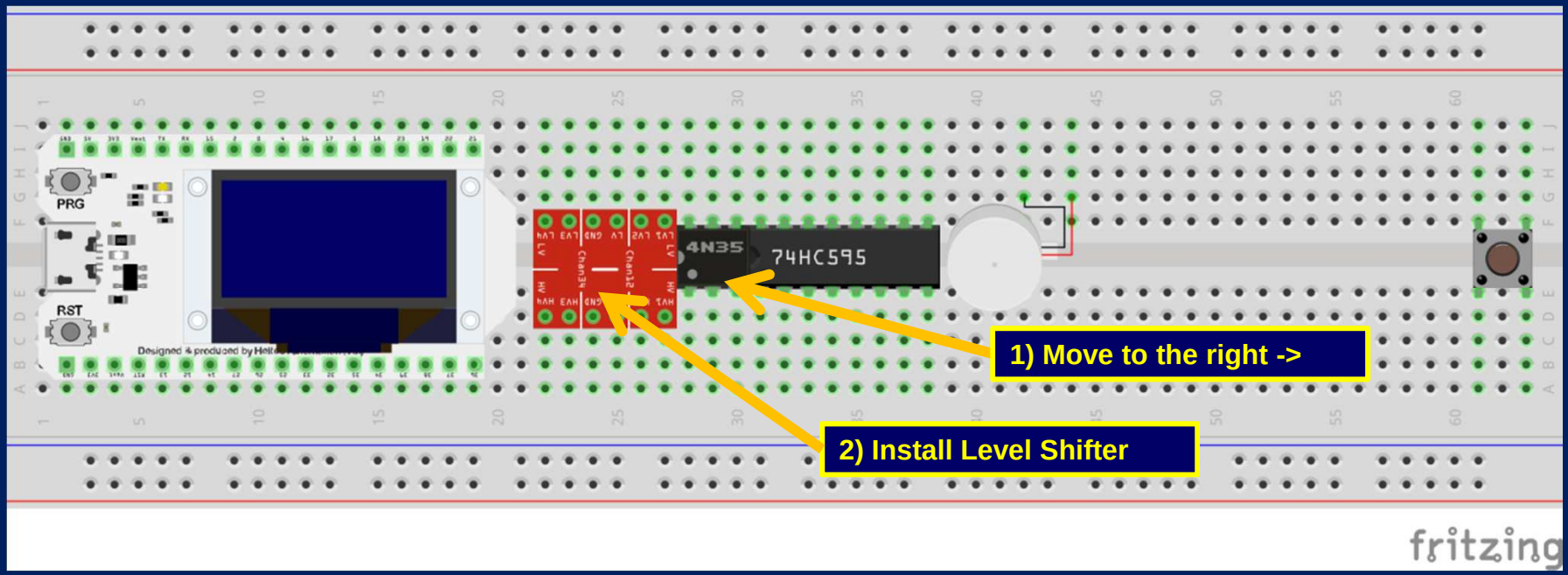
- Level shifters provide bidirectional translation of logical signal voltages between different digital peripherals and devices
- For instance, if you needed to connect a 3.3V microcontroller to a 5V real time clock (RTC), you would use a level shifter on the SPI or I2C serial connections
- Nearly all level shifters are for 5 VDC and 3.3VDC translation
- Level shifters are not self powered; they must have external power connections to the higher voltage (5V) rail





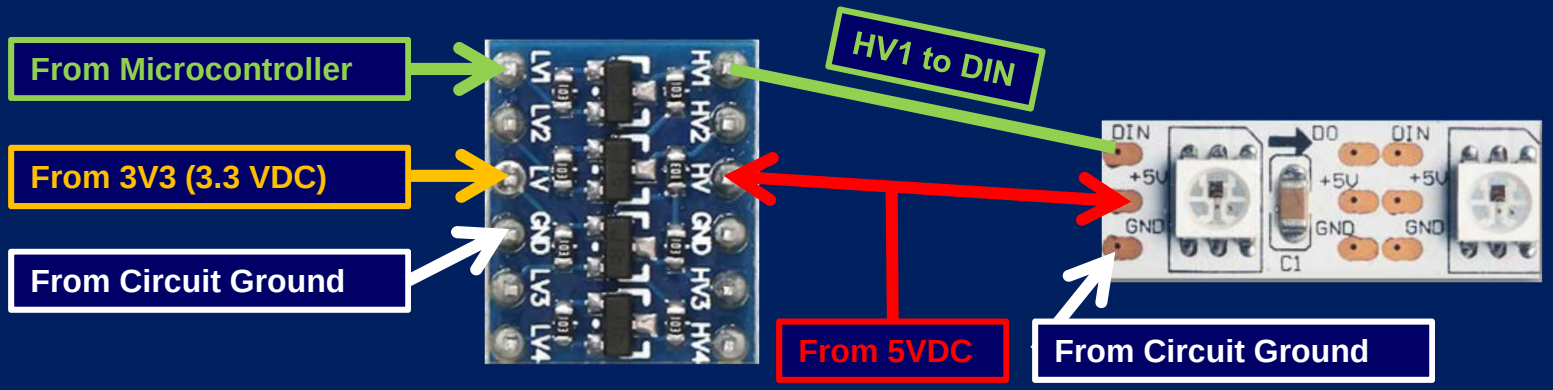
Adding A Level Shifter

- Level shifters provide bidirectional translation of logical signal voltages between different digital peripherals and devices
- Update your breadboard to look like the below image
- Shift the 4N35 opto-isolator to the right if needed to make room for the level shifter next to the Heltec board



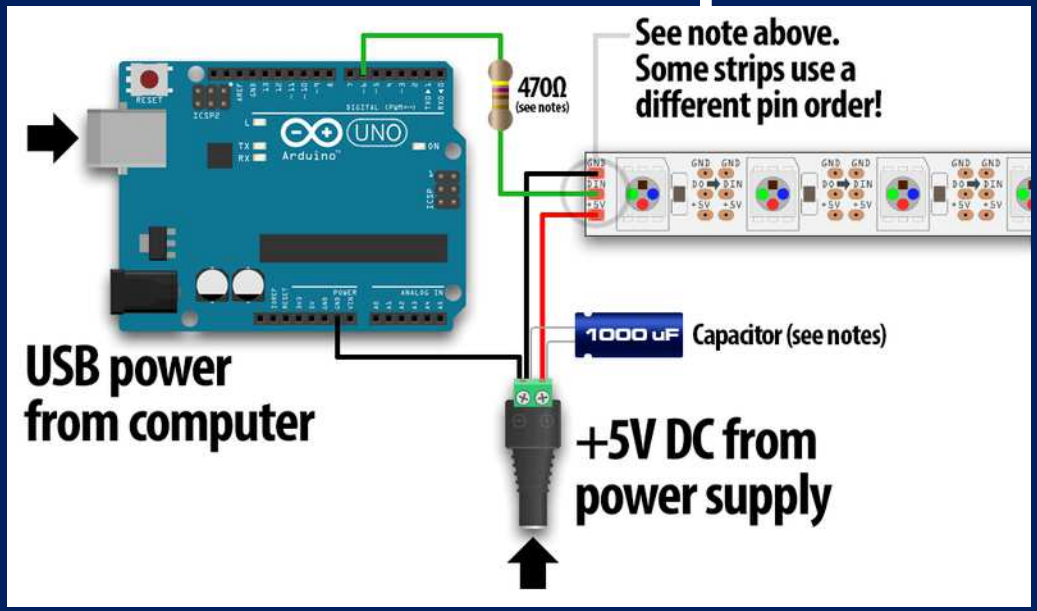
Using a Level Shifter

- Level shifters require connections from both 5VDC (“high voltage”) and 3V3 (“low voltage”)
- 3V3 is often taken direct from the microcontroller; we will do this
- 5VDC should come from the main 5V rail
- Only a single ground connection to the level shifter is required
- **The microcontroller data out pin connects to an “LV” input of the level shifter**
- **The “HV” output of the level shifter connects to the LED strip (or other 5VDC peripheral)**



External Power and WS2812B Neopixels

- RGB LEDs of any kind draw *LOTS* of current. 20mA per LED x 3 LEDs = 60 mA of current as a general power budget. *THAT'S PER LED.*
- More than 5 Neopixels will overload most 500mA board power regulators.
- The solution to this problem is to use external power for the LED strip.
- When the external power is 12 or 5 volts and the microcontroller is 3 volts things get very tricky indeed
- It's OK to send 5V to the board input power pin and the LEDs simultaneously
- It's generally NOT OK to connect USB power *at the same time*
- *Level shifting may be required for the serial data line (Data IN) to the Neopixels. This is entirely dependent on the pixel type. YMMV!*



The ability to SAFELY connect USB power and external power at the same time depends on the board!!!

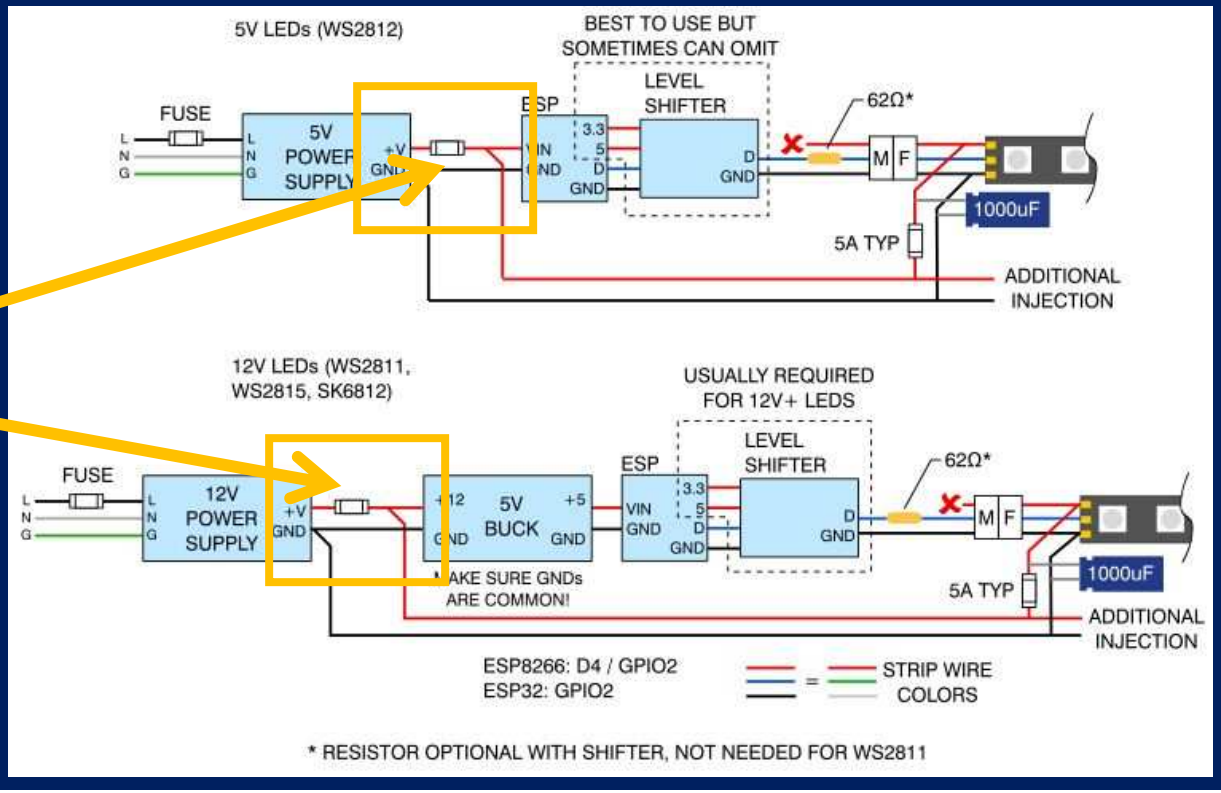
Most boards will attempt to power peripheral devices *using USB power* There is no sanity checking of this action

Trying to power numbers of Neopixels from *any* board will overload the onboard regulator

AN IMPORTANT NOTE ON FUSES!

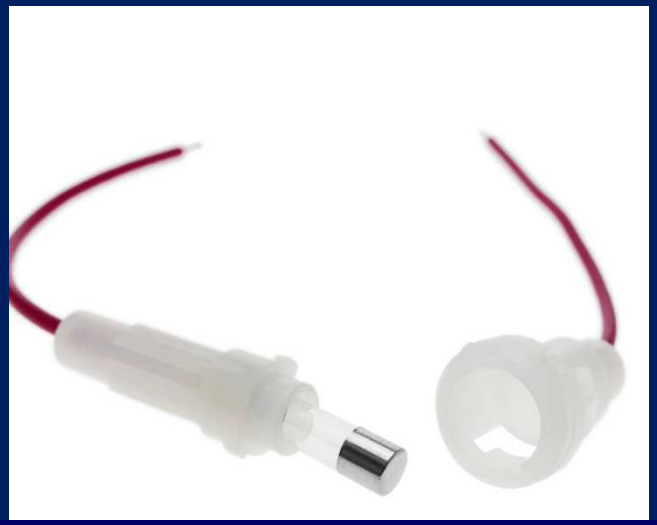
- **Remember that lots of LEDs draw huge amounts of power!**
- **For 2A or less, it's OK to rely on your power supply to provide overcurrent and fault protection**
- **For 3A or larger power supplies, *always use a fuse!***
- ***Fuses keep magic smoke from becoming real fire!!***

<u>Place fuses in-line (in series) with positive rail</u> *immediately following your power supply positive output.* Do this for all power supplies Three (3) Amps or larger!!!

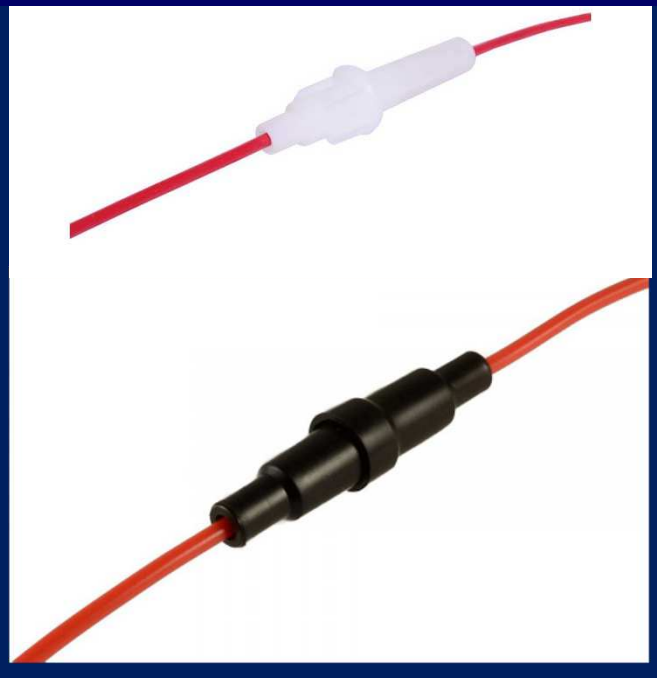


Typical in-line fuses for electronics

MINI BLADE FUSE HOLDER



SCREW TYPE BUSS HOLDER



Use mini, not full size, fuses for your projects. Both blade and glass/Buss fuses and fuse holders come in mini sizes.
Use a slightly larger fuse than your power supply rating, e.g., 5A fuse for 3A or 4A supply, 15A fuse for 10A supply, etc. Don't massively oversize the fuse though (20A fuse on a 5A supply) as that defeats the purpose.

Wiring an ESP32 and Neopixels: 3.3 volt

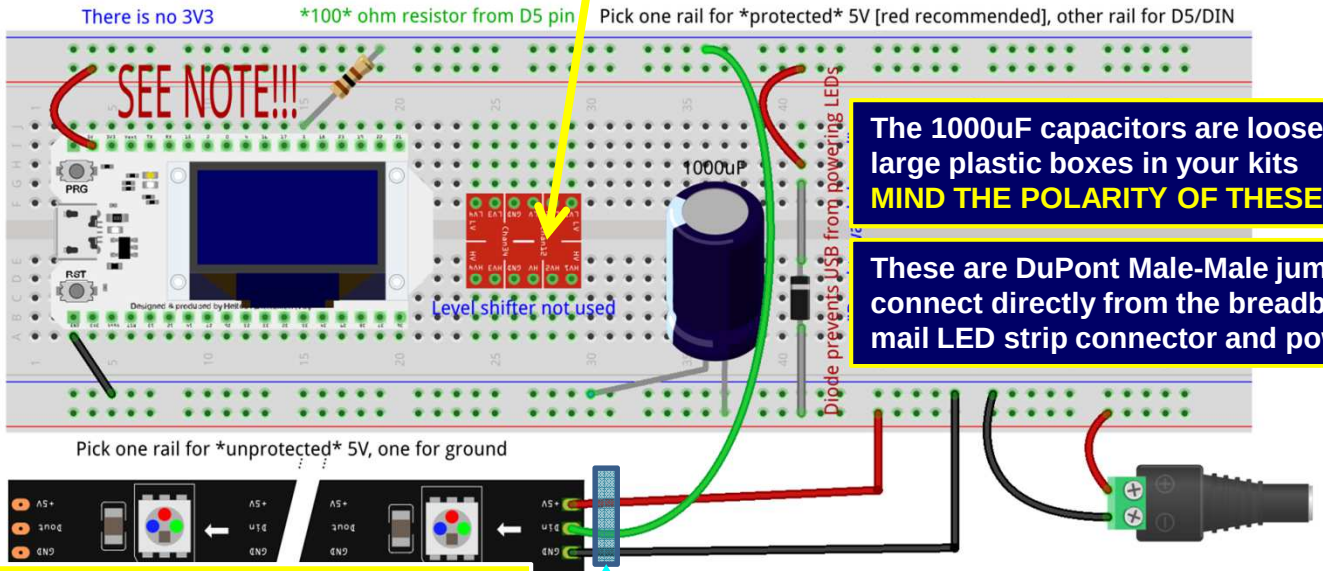
From your Arduino kits:

- Green power plug adapter
- 1000uF capacitor
- 100 ohm (brown-black-brown) resistor
- WS2812B LED strip
- One (1) general purpose diode
- Male-Male DuPont jumpers as required

Further research shows that in many cases a 100 ohm resistor is sufficient for coupling the microcontroller to the LED strip. The level shifter is not required. Unfortunately, this depends on microcontroller quality which varies greatly with our inexpensive units.

NEVER connect USB and external power *without* the in-line diode. Or else something will go *terribly* wrong.

Level shifter is NOT used *or wired* with this build!



The 1000uF capacitors are loose in the large plastic boxes in your kits
MIND THE POLARITY OF THESE!!!!

These are DuPont Male-Male jumper wires that connect directly from the breadboard rails to the mail LED strip connector and power plug.

Be 100% certain that you have not reversed the 5V and ground connections to the LED strip!
Ground wire on strips is WHITE, not BLACK.

JST connector not fully shown

Have your instructor check all work before applying power to anything!

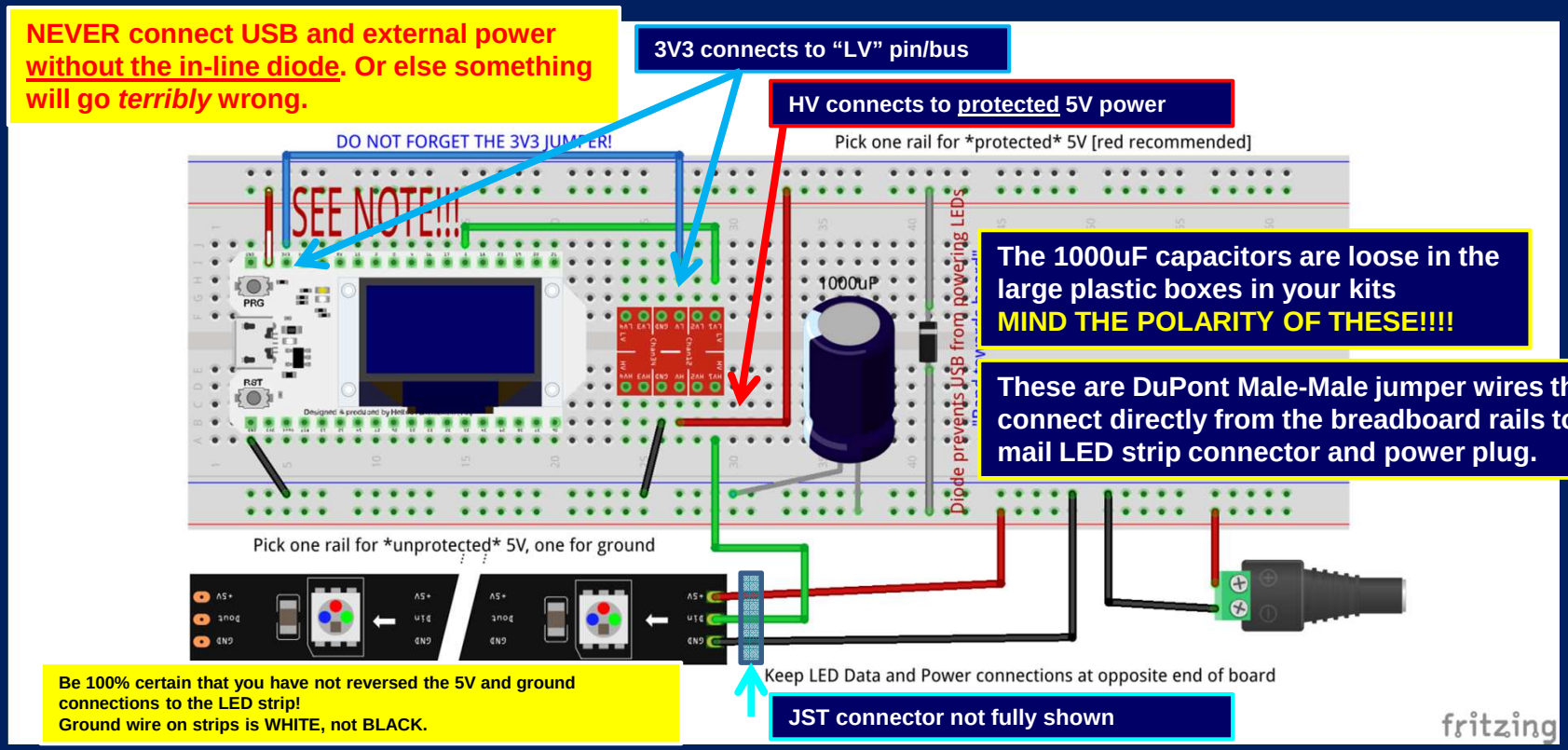
Wiring an ESP32 and Neopixels: Level Shift

From your Arduino kits:

- Green power plug adapter
- 1000uF capacitor
- WS2812B LED strip
- One (1) general purpose diode
- Male-Male DuPont jumpers as required

In spite of extensive pre-testing of the *design*, it turns out that our LED strips are an off-brand that will ~~probably~~ not operate without level shifters. These will be preinstalled in kits or supplied in class.

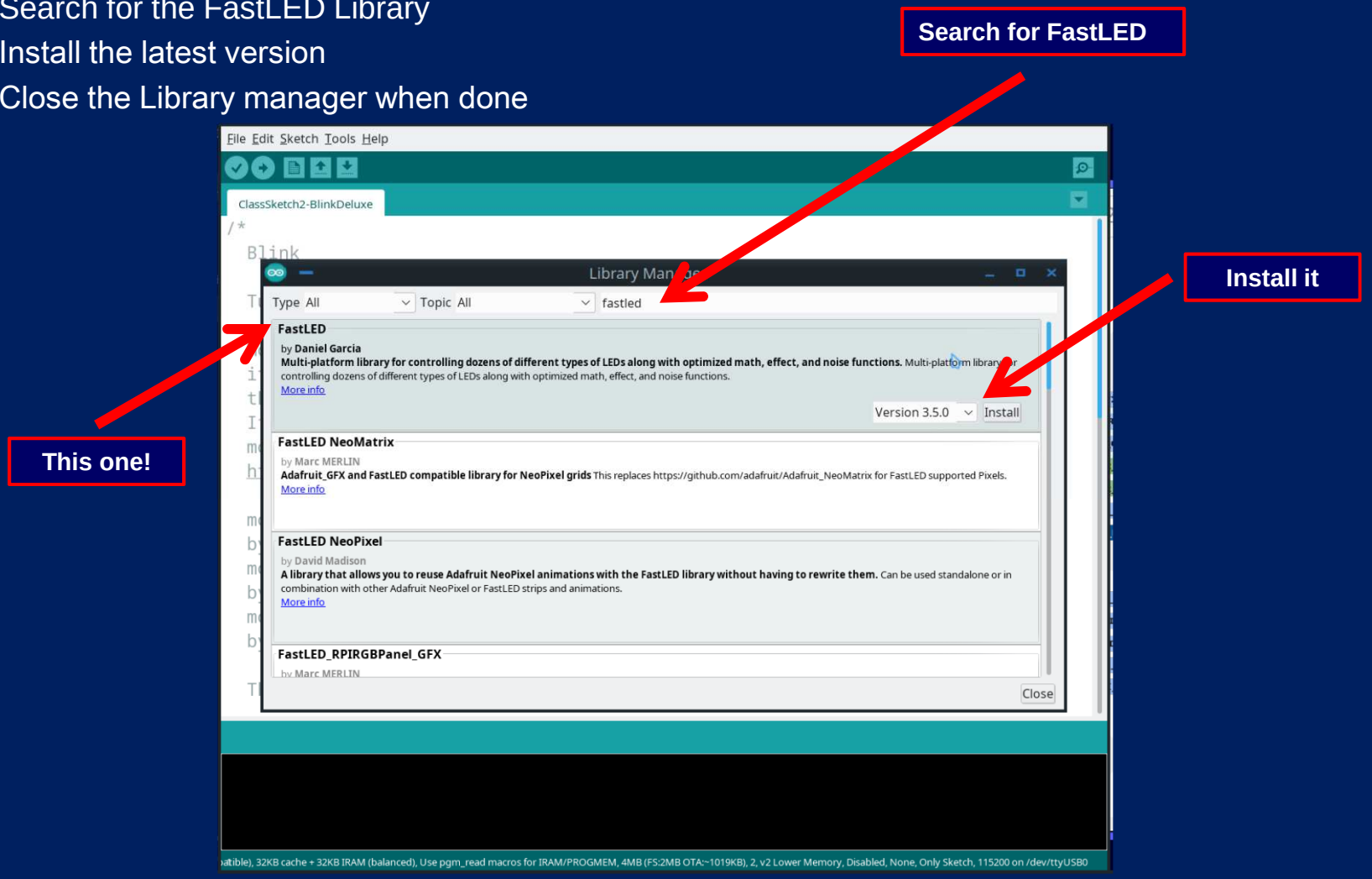
Wire your breadboard up to look like this:



Have your instructor check all work before applying power to anything!

The FastLED Library

- Find the Arduino sketch in your Lesson 9 folder (which you copied from your USB drives, right?) named “ClassSketch3-V2” and open it in the Arduino IDE (double clicking the [.ino] file is easiest)
- Open the Library manager (Tools-Manage Libraries)
- Search for the FastLED Library
- Install the latest version
- Close the Library manager when done



Let There Be Light Redux

- https://install.wled.me/
- Connect your board to your laptop using USB cable (board end first, then the laptop end!) LEAVE EXTERNAL 5V POWER DISCONNECTED!!!
- Remember to set your board type and port speed! (Change speed if initial upload fails!)
- Upload the sketch! ➔ (this also validates it first)
- **Disconnect USB!!!**
- **Connect the external 5VDC power supply**
- Did it work? Enjoy the light show!☺
- It didn't work? Now we get to troubleshoot!🛠😎

See what happens if you change "RAINBOW true" to "RAINBOW false" in your sketch. Remember to reupload after modifications!

Anatomy of a sketch

```
#include <FastLED.h>
```

FastLED library must be installed with Library Manager before it can be included in the sketch

```
// How many leds in your strip?
#define NUM_LEDS 30
// Which pin is the pixel DIN line connected to?
#define DATA_PIN 5
#define LED_TYPE WS2812B
#define COLOR_ORDER GRB
#define BRIGHTNESS 30

#define RAINBOW true

Setup()
{
  delay(3000);
  FastLED.addLeds<LED_TYPE, DATA_PIN, COLOR_ORDER>(leds, NUM_LEDS);
  FastLED.setBrightness(BRIGHTNESS);
}
```

These #define's are actually *macro* definitions, and not *variable* declarations. Macros are substituted wherever they appear. More on that next semester– or see The Arduino Cookbook.

Setup section excerpt

```
void loop()
{

  if (RAINBOW)
  {
    fill_rainbow(leds,NUM_LEDS,i++,1);
    FastLED.show();
  }
...
```

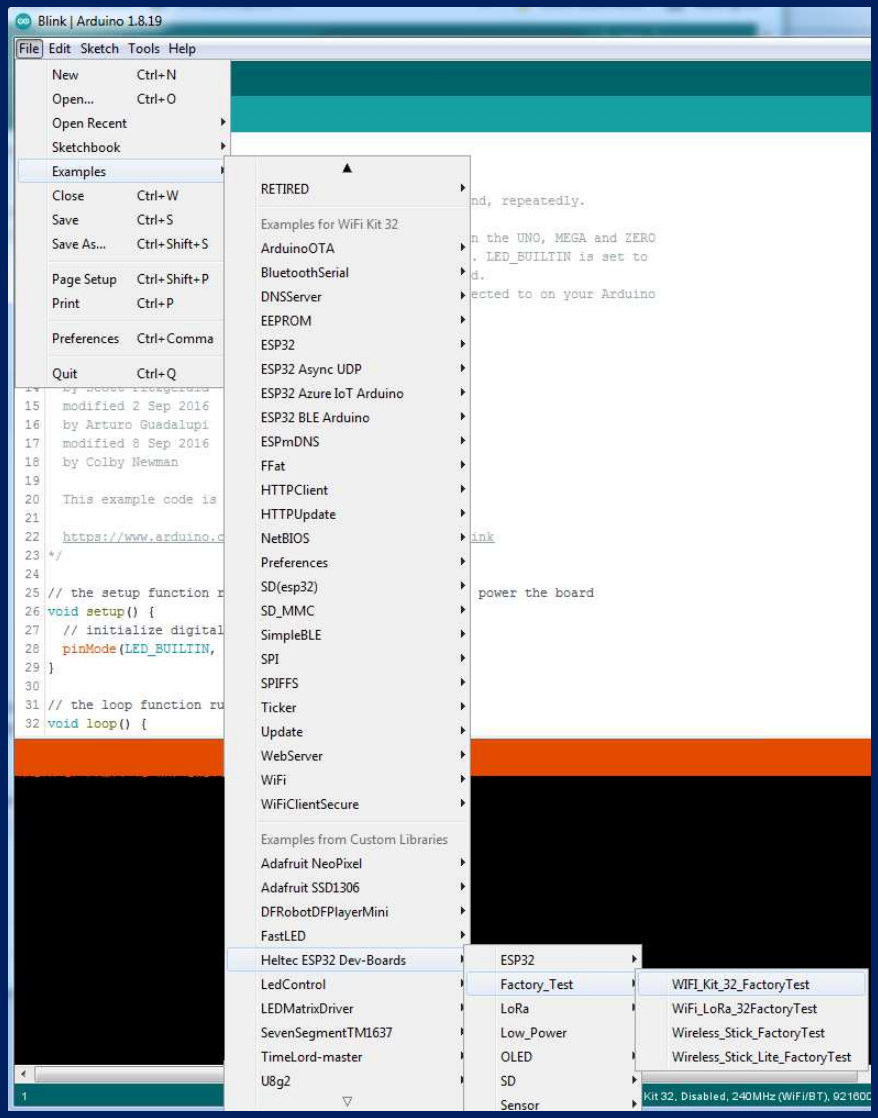
Loop section excerpt

What happens if you change the RAINBOW def to false?

WiFi Operation Test

Due to the numerous manufacturer defects in our off-brand Heltec boards, we are going to validate WiFi operation before moving forwards. Anyone with a board having defective WiFi will receive a Wemos D1 mini controller to complete the class with.

- Connect your Heltec board
- Find the WiFi_Kit_32_FactoryTest example sketch under Heltec ESP32 DevBoards sub-menu
- After checking your port and port speed settings, click the upload button
- You should see at least one wifi network listed once the board reboots and fails its initial connection attempt.



Flashing Software Binaries

- The Arduino IDE takes Processing/C++ source code and compiles it into a single binary file which is then transmitted to the microcontroller where it is stored in flash memory
- Uploading of binary executable files to a microcontroller is called “flashing”
- It is possible to obtain precompiled binaries and flash them without having source code or using the Arduino IDE
- This is very, very similar to installing an app on a phone (and is identical to initially installing the *operating system* onto a phone, which is done at the factory)
- There is extremely sophisticated smart LED strip control software called WLED which is specially made to be flashed to ESP8266 and ESP32 chipsets. And we will do this.

Flashing WLED Software on ESP32
3 Different methods based on OS

- Kubuntu Linux
  Install and use esptool (uses Konsole CLI)
- Windows (with current version of Chrome)
  https://install.wled.me/
- Windows (if online browser install doesn’t work)
  Install Python3 and use esptool.py

Flashing WLED Software on Linux

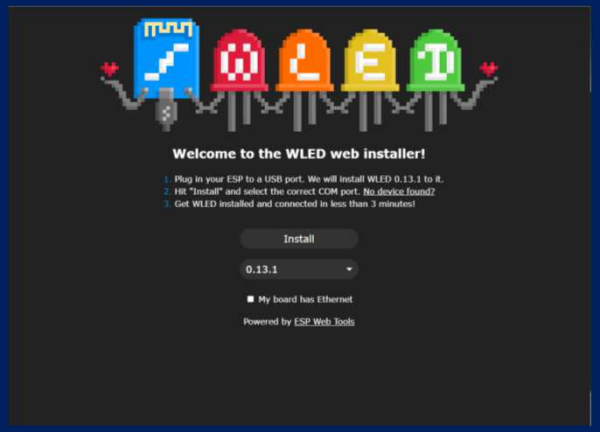
- Copy these software binary files from your flash drive to your Documents folder (use the Dolphin GUI file explorer)
  `esp32_bootloader_v4`
  `WLED_0.13.1_ESP32`

- Open a Konsole window, and enter the following commands (enter your account password when/if prompted)

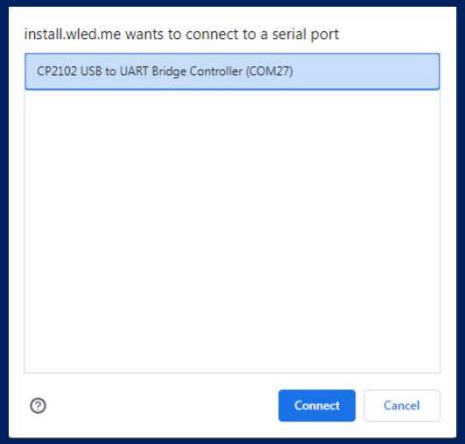
```
sudo apt install esptool
cd Documents
esptool --port /dev/ttyUSB0 write_flash 0x0
./esp32_bootloader_v4.bin
esptool --port /dev/ttyUSB0 write_flash 0x10000
./WLED_0.13.1_ESP32.bin
```

WLED: Windows with Chrome

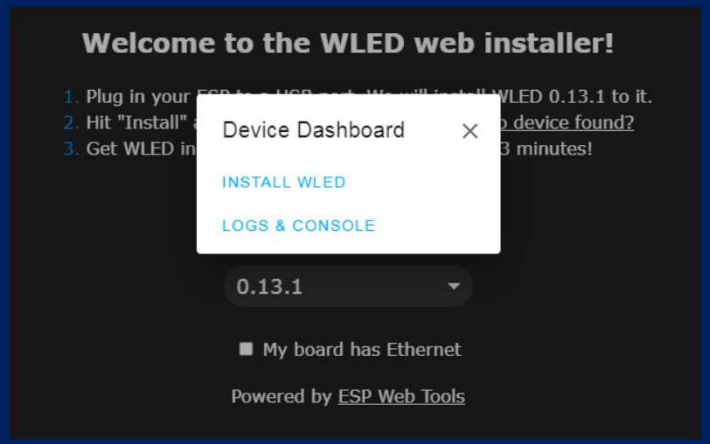
- Go to https://install.wled.me/ in Chrome
- Click install and do the logical thing at the prompts. See below for details



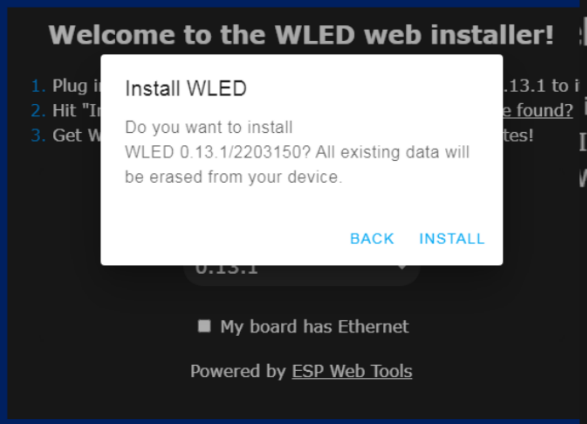
Click on Install



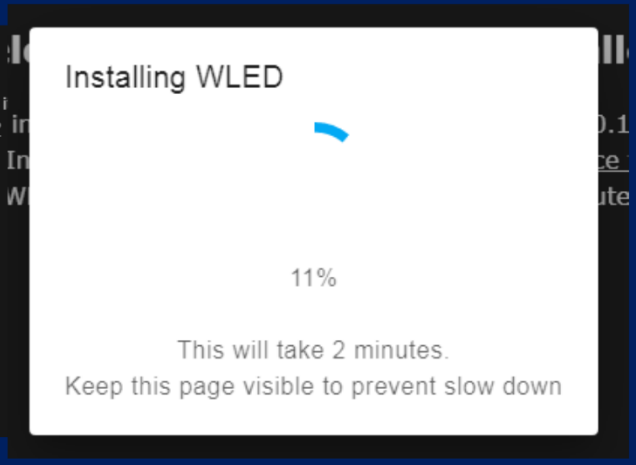
Click on Connect



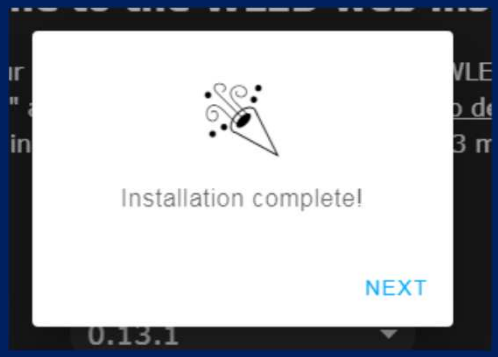
Click on INSTALL WLED



Click on INSTALL (once more)



Wait a while…



All done!

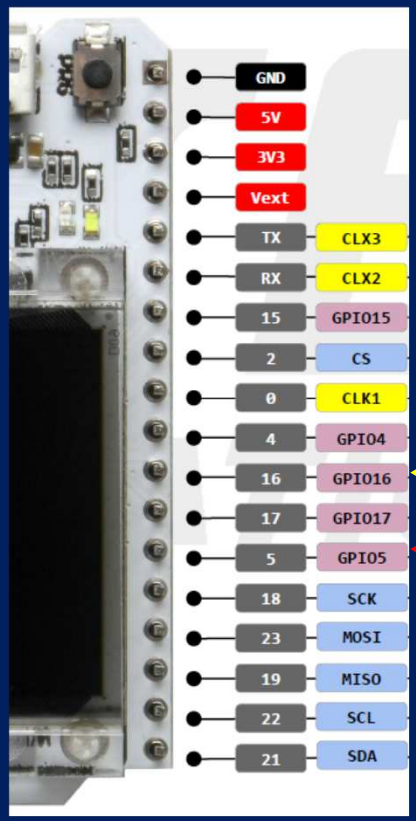
WLED: Windows with Python3

- If the browser install doesn't work on your Windows workstation, you'll need to install Python3 and then install esptool.py
- Copy these software binary files from your flash drive to your Documents folder
  `esp32_bootloader_v4`
  `WLED_0.13.1_ESP32`
- For Windows 10/11, run the `python-3.10.4-amd64` installer (copy it from your flash drive FIRST!)
- For Windows 7/8, run the `python-3.8.10-amd64` installer (copy it from your flash drive FIRST!)
- Once Python is installed, open a CMD (DOS CLI) window
- Enter the following commands into the DOS CLI prompt (you can skip the REM comments):

```
pip install esptool
REM Change to the directory you copied the WLED binaries to
cd %userprofile%\Documents
REM Connect your board and find the port it's on with Device Mgr (dnu "??")
REM Then flash the bootloader first
esptool.py --port COM?? write_flash 0x0 ./esp32_bootloader_v4.bin
REM Then flash the WLED software itself
esptool.py --port COM?? write_flash 0x10000 ./WLED_0.13.1_ESP32.bin
```

WLED – LED PINS

- WLED is configured to use certain hardware defaults for the ESP-32 processor
- You will need to either change the GPIO pin used to DIN to your LED strip, or update the GPIO pin in the settings dialog on the web UI
- Either will work!



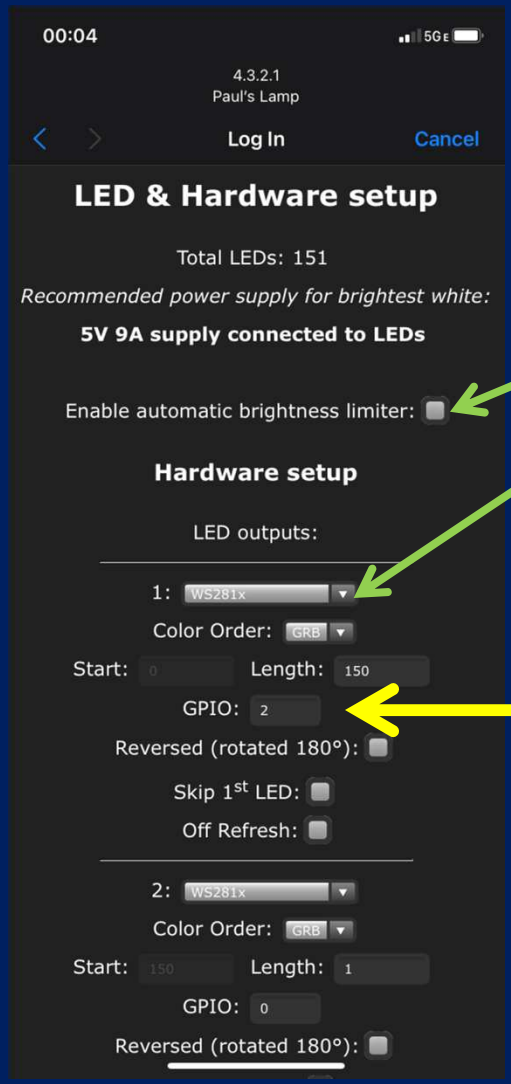
WLED IS SET TO USE GPIO16 BY DEFAULT. MOVE LED DATA DIN JUMPER TO THIS PIN TO RECONFIGURE BOARD

WE USE GPIO5 FOR SKETCH#3; THIS IS HOW YOUR BOARD IS WIRED NOW

YOU MUST EITHER MOVE THE DIN JUMPER TO GPIO16 OR UPDATE THE SOFTWARE SETTING!

WLED – GPIO SETTING

- WLED is configured to use certain hardware defaults for the ESP-32 processor
- You can change the GPIO pin used for the LED DIN data line in the LED settings screen (Configuration->LED Settings)
- Beware that GPIO pins may be *named* differently on different boards! Always verify using the pinout diagram for your board!



Enabling automatic brightness limiting is highly recommended with your 2A power supplies!

Note: Nearly all strips are WS281x

This setting must match the GPIO pin that the DIN data line to the LED strip is connected to!

Modify your LED settings screen to match the above if you want to use the existing class configuration of GPIO5

WLED – Getting Started

- Introductory instructions are at https://kno.wled.ge/basics/getting-started/
- Default WiFi SSID on startup is WLED-AP
- Default WiFi password is wled1234
- Once you connect, you should automagically be taken to the main WLED page on your ESP32
- Click on “WIFI SETTINGS”; Change “AP SSID” and “AP password” to personalize your WLED controller; “Save and Connect” **DO NOT connect your controller to a local wifi net at this time!**
- Reconnect to your new ESP32 lamp controller, and check out the zillion and one options!

WLED – Going Further

- WLED is designed to be connected to a WiFi network. **Once you configure the software to connect to WiFi, you will need the Android or iOS phone application WLED in order to reach the controller!**
- You can also reach the controller at the IP address assigned to it by your WiFi network, but figuring out what that assigned IP address is can be tricky. Best use the phone app instead.
- Once connected to WiFi, you can enable Internet time sync which lets you configure schedules etc.
- There is a wealth of awesome information and ideas at https://kno.wled.ge/basics/tutorials/

ESP8266 Addenda

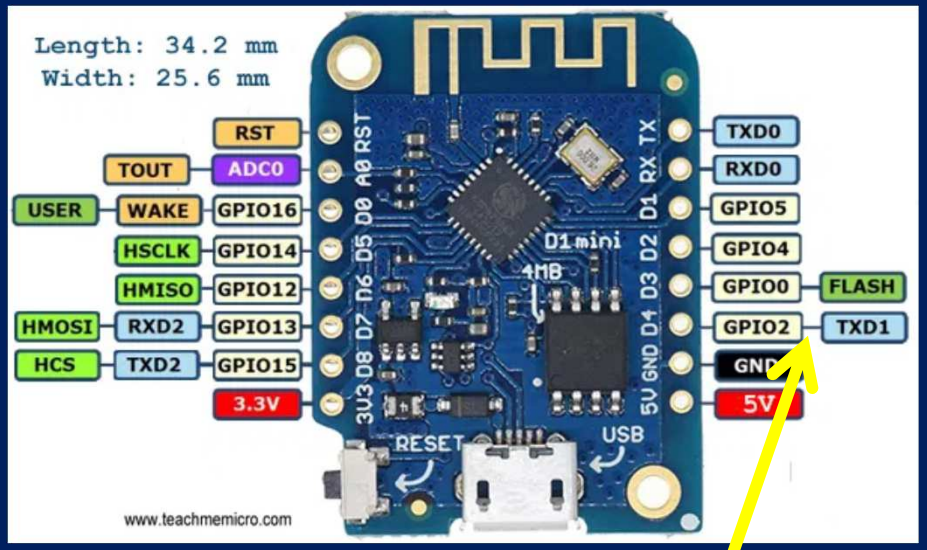
Since we have just used up all the spare microcontrollers due to recent failures and factory defects, any boards which fail the WiFi check will be replaced with inexpensive ESP8266 boards. Here is information on that board for your reference, duplicated to your flash drives.

- **wemos D1 Mini Pinout**
- **WS2812 Breadboard diagram for wemos D1**
- **Flashing WLED software on ESP8266**

wemos D1 Mini and Pinout

Hilights

- **11 digital IO, interrupt/pwm/I2C/one-wire supported(except D0)**
- **1 analog input(3.2V max input)**
- **Micro USB or Type-C USB Port (clones usually have micro USB)**
- **LOLIN I2C Port**
- **Compatible with MicroPython, Arduino, nodemcu**
- **Uses the CH340 USB-to-serial driver (installation usually needed on Windows)**



WLED default button is GPIO0 (pin D2)

WLED default LED data is GPIO2 (pin D4)

Microcontroller	ESP8266
Processor	Tensilica Xtensa Diamond 32-bit
Operating Voltage	3.3V
Minimum Operating Voltage	2.58V
Maximum Operating Voltage	3.6V
Arduino IDE Board	LOLIN(WEMOS) D1 R2 & mini
Power Supply via VIN,VCC	4V...6V
VIN,VCC output USB power supply	4.67V
Digital I/O Pins (with PWM)	11 (11)
Analog Input Pins	1
Resolution ADC	10 bit (0...1023)
Analog Output Pins	0
SPI/I2C/I2S/UART	1/1/1/1
Max DC Current per I/O Pin	12 mA
Flash Memory	4 MB
SRAM	64 KB
EEPROM	512 bytes
Clock Speed	80/160 MHz
Length x Width	34mm x 26mm
Fits on standard breadboard	yes
WIFI	IEEE 802.11 b/g/n
Bluetooth	no
Touch sensor	no
CAN	no
Ethernet MAC Interface	no
Temperature Sensor	no
Hall effect sensor	no
Power jack	no
USB connection	yes
Battery Connection	no
Programmable	Arduino IDE, LuaNode SDK, Micropython
3.3V Voltage Regulator	ME6211
Output Voltage	3.3V
Maximum Input Voltage	**6V**
Minimum Input Voltage	4.3V
Maximum Output Current	**500mA**

Wiring a D1 Mini and Neopixels – 3.3 Volt

From your Arduino kits:

- Green power plug adapter
- 1000uF capacitor
- 100 ohm resistor (brown-black-brown)
- WS2812B LED strip
- One (1) general purpose diode
- Male-Male DuPont jumpers as required

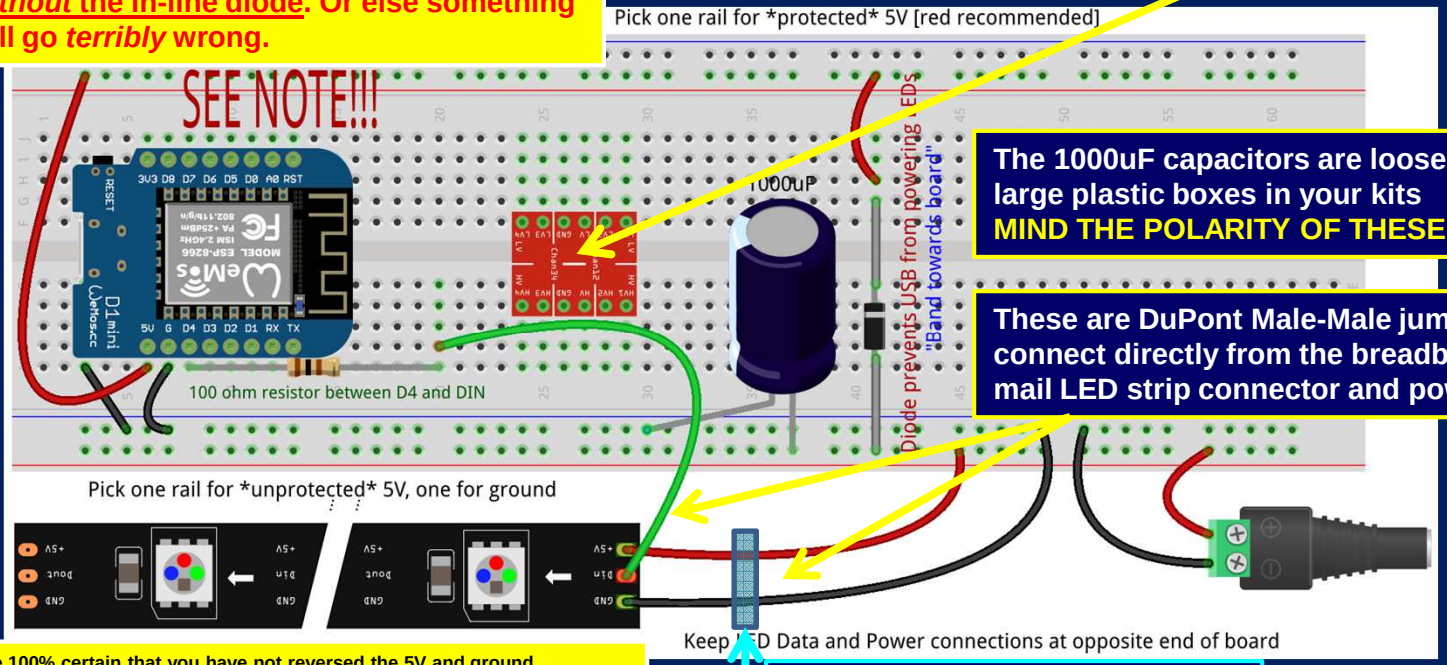
Wire your breadboard up to look like this:

Use this diagram if you have been given a D1 Mini ESP8266 board instead of a Heltec ESP32 board

In most cases, level shifters are not required, only 100 ohm resistors to couple the D4 pin in on the ESP8266 to the DIN pin on the LED strip

NEVER connect USB and external power *without* the in-line diode. Or else something will go *terribly* wrong.

Level shifter is NOT used *or wired* with this build!



The 1000uF capacitors are loose in the large plastic boxes in your kits
MIND THE POLARITY OF THESE!!!!

These are DuPont Male-Male jumper wires that connect directly from the breadboard rails to the mail LED strip connector and power plug.

Be 100% certain that you have not reversed the 5V and ground connections to the LED strip!
Ground wire on strip JST connectors is WHITE, not BLACK.

JST connector not fully shown

Have your instructor check all work before applying power to anything!

Wiring a D1 Mini and Neopixels – Level Shift

From your Arduino kits:

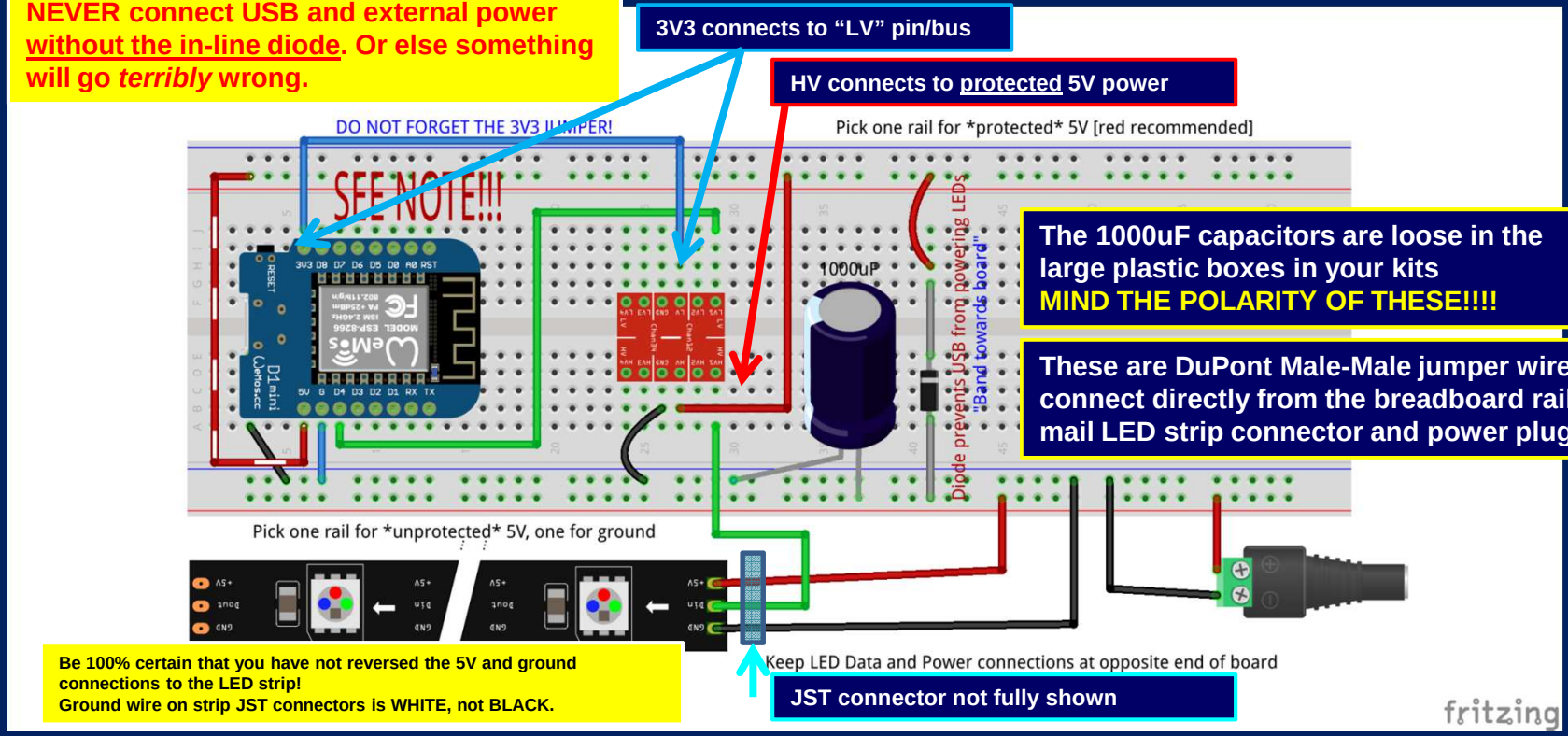
- Green power plug adapter
- 1000uF capacitor
- WS2812B LED strip
- One (1) general purpose diode
- Male-Male DuPont jumpers as required

Use this diagram if you have been given a D1 Mini ESP8266 board instead of a Heltec ESP32 board

In spite of extensive pre-testing of the *design*, it turns out that our LED strips are an off-brand that will not operate without level shifters. These will be preinstalled in kits or supplied in class.

Wire your breadboard up to look like this:

NEVER connect USB and external power without the in-line diode. Or else something will go *terribly* wrong.



Be 100% certain that you have not reversed the 5V and ground connections to the LED strip!
Ground wire on strip JST connectors is WHITE, not BLACK.

Have your instructor check all work before applying power to anything!

Flashing WLED Software on ESP8266
3 Different Methods based on OS

- Kubuntu Linux
  Install and use esptool (uses Konsole CLI)
- Windows (with current version of Chrome)
  https://install.wled.me/
- Windows (if online browser install doesn’t work)
  Install Python3 and use esptool.py

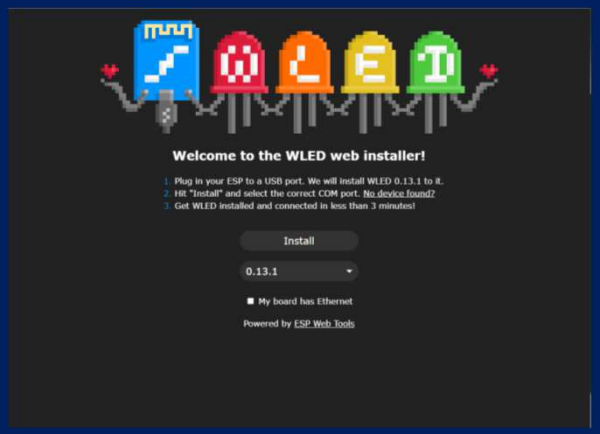
Flashing ESP8266 Software on Linux

- Copy these software binary files from your flash drive to your Documents folder (use the Dolphin GUI file explorer)
  `WLED_0.13.1_ESP8266`
- Open a Konsole window, and enter the following commands (enter your account password when/if prompted)

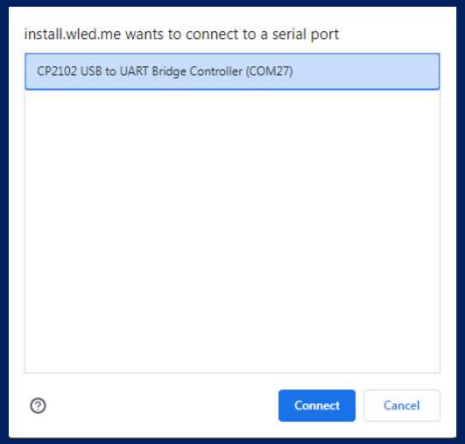
```
sudo apt install esptool
cd Documents
esptool --port /dev/ttyUSB0 write_flash 0x0 WLED_0.13.1_ESP8266
```

WLED ESP8266: Windows with Chrome

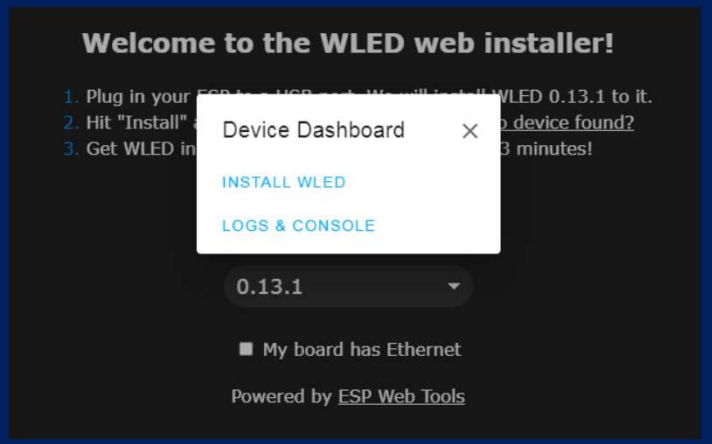
- Go to https://install.wled.me/ in Chrome
- Click install and do the logical thing at the prompts. See below for details



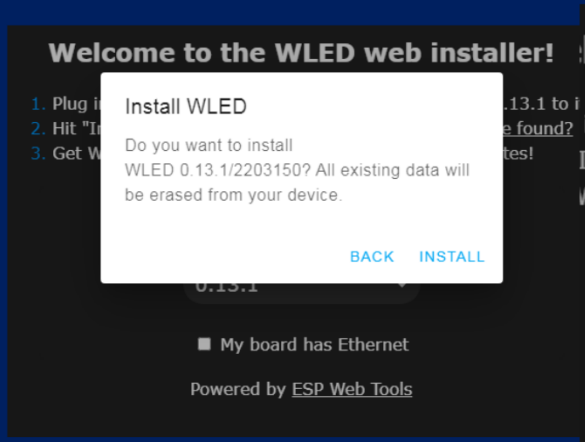
Click on Install



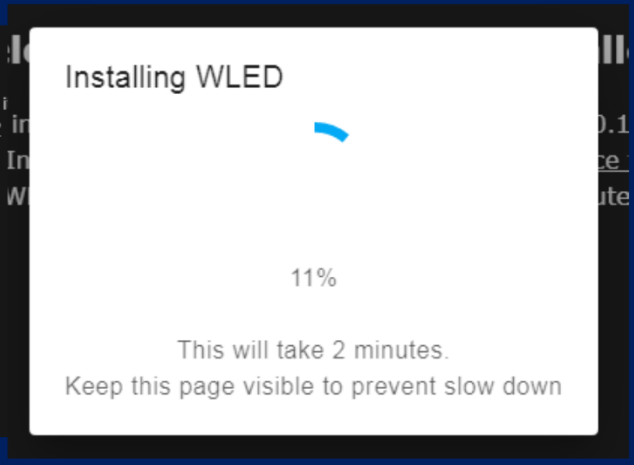
Click on Connect



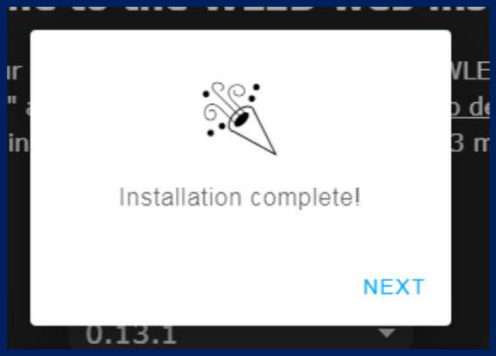
Click on INSTALL WLED



Click on INSTALL (once more)



Wait a while…



All done!

WLED ESP8266: Windows with Python3

- If the browser install doesn't work on your Windows workstation, you'll need to install Python3 and then install esptool.py
- Copy these software binary files from your flash drive to your Documents folder `WLED_0.13.1_ESP8266`
- For Windows 10/11, run the `python-3.10.4-amd64` installer (copy it from your flash drive FIRST!)
- For Windows 7/8, run the `python-3.8.10-amd64` installer (copy it from your flash drive FIRST!)
- Once Python is installed, open a CMD (DOS CLI) window
- Enter the following commands into the DOS CLI prompt (you can skip the REM comments):

```
pip install esptool
REM Change to the directory you copied the WLED binaries to
cd %userprofile%\Documents
REM Connect your board and find the port it's on with Device Mgr
REM Then flash the WLED software
esptool.py --port COM?? write_flash 0x0 ./WLED_0.13.1_ESP8266
```

WLED – Getting Started

- Introductory instructions are at https://kno.wled.ge/basics/getting-started/
- Default WiFi SSID on startup is WLED-AP
- Default WiFi password is wled1234
- Once you connect, you should automagically be taken to the main WLED page on your ESP8266
- Click on “WIFI SETTINGS”; Change “AP SSID” and “AP password” to personalize your WLED controller; “Save and Connect” [DO NOT connect your controller to a local wifi net at this time!]
- Reconnect to your new ESP8266 lamp controller, and check out the zillion and one options!

Formal End of Lesson 10-11-12

AND THAT'S THE END OF THIS CLASS!

Arduino Programming will be offered in Fall of 2022. Hope to see you all there!

Note: There are no prerequisite classes for Arduino Programming. All students will be supplied with a prewired ESP8266 microcontroller and LED matrix display for use during the class.

Remember to keep your kits and USB drives!