Lab 1 - Blink an LED

Adapted tutorial from randomnerdtutorials.com https://randomnerdtutorials.com/getting-started-with-esp32/

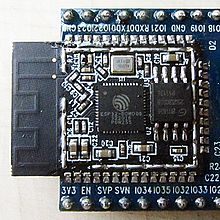
## Learning Outcomes:

* Learn about esp2
* Blink an LED

## Background:

**ESP32:**

ESP32 is a series of low-cost microcontrollers with a built-in wifi and bluetooth module. A microcontroller is an integrated circuit with a small amount of memory and other resources that executes a program over and over again. It is different from your computer in that your computer runs an operating system that can change which task it’s working on. A microcontroller doesn’t run an operating system so it will execute the task you gave it until it's powered off or it’s memory is changed.



*Inside an ESP-WROOM-32 Module. Image Courtesy of Wikipedia.*

We will be using the DOIT DEVIT V1 ESP32-WROOM-32 Development Board to run our RC/AutonoMouse car. Before we build our car we will learn how each subsystem works and test each to ensure that they work.

Video about ESP32: [Introduction to ESP32 - Getting Started](https://www.youtube.com/watch?v=xPlN_Tk3VLQ) (0:00-10:25)

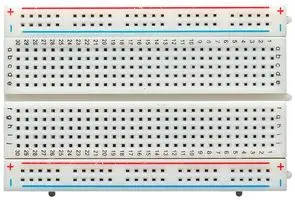
Video about microcontrollers: [What is a Microcontroller?](https://www.youtube.com/watch?v=jKT4H0bstH8)

**Breadboards:**

When working with electronics we usually use a breadboard to prototype our circuit before permanently creating it on a printed circuit board. To finish this lab you need to show that you can blink an LED with your esp32.

Before we start creating our circuit please pay attention to a few things:

First, notice that the breadboard is organized a certain way. The - and + rails are connected horizontally, but all other rails are connected vertically. This is important to remember and take advantage of while prototyping.



*Breadboard. Image courtesy of Newark.*

Second, notice that in the diagram below one side of the LED is bent and the other is straight. This is because an LED is a polarized component (meaning that current only flows one way, if you put it in backwards the LED won’t turn on). The positive side (bent side) is called the anode and the negative side (straight side) is called the cathode. Make sure the anode is connected to power and the cathode is connected to ground.

Third, notice that the LED is in series with the 330 ohm resistor. The LED has almost no resistance and if we put it directly between power and ground too much current would flow and break the LED. The resistor constricts the current to a much smaller value that the LED can handle.

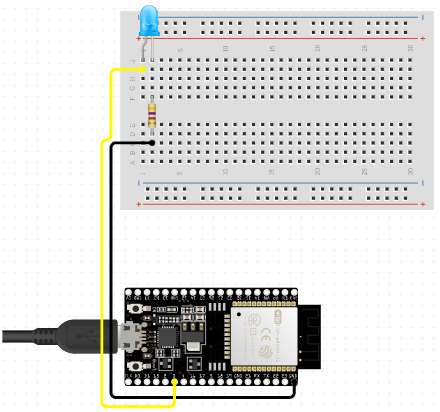
Fourth, notice that there is a connection between ground on the breadboard and the ground (gnd) pin on the esp32. Without this connection there would not be a common ground between the board and the arduino, and the circuit would not work.

The objectives for today’s lab is to 1) build a circuit on a breadboard to blink an LED, 2) install Arduino IDE, and 3) code our esp32 to blink our LED.

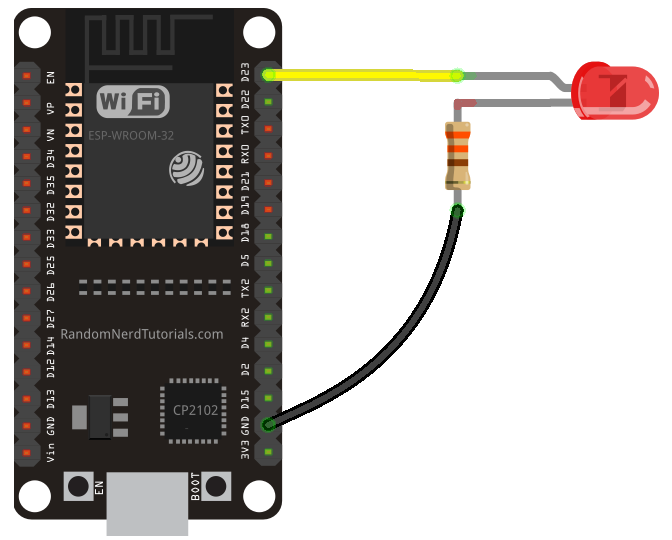
## Step 1 Blink LED Circuit Diagram:

Create the circuit shown in the picture below. (the board used in the picture here is an Arduino Nano, use your esp32 instead and make sure to use the correct pins) Push the esp32 far enough into the board to make a connection but be careful not to break any pins).

1. Place the esp32 in the middle of the breadboard bridging both sides
2. Place a wire from the gnd pin on the esp32 to the (-) rail on the breadboard
3. Place the LED somewhere to the side of the esp32 (don’t place it on any of the same rails that esp32 is on)
4. Place a wire from Pin25 on the esp32 to the anode of the LED
5. Place a 330 ohm resistor (look up resistor band chart) between the (-) rail and the cathode of the LED



*Circuit diagram from circuito.io*



*Image courtesy of randomnerdtutorials.com*

[*59 thoughts on “Getting Started with the ESP32 Development Board”*](https://randomnerdtutorials.com/getting-started-with-esp32/)

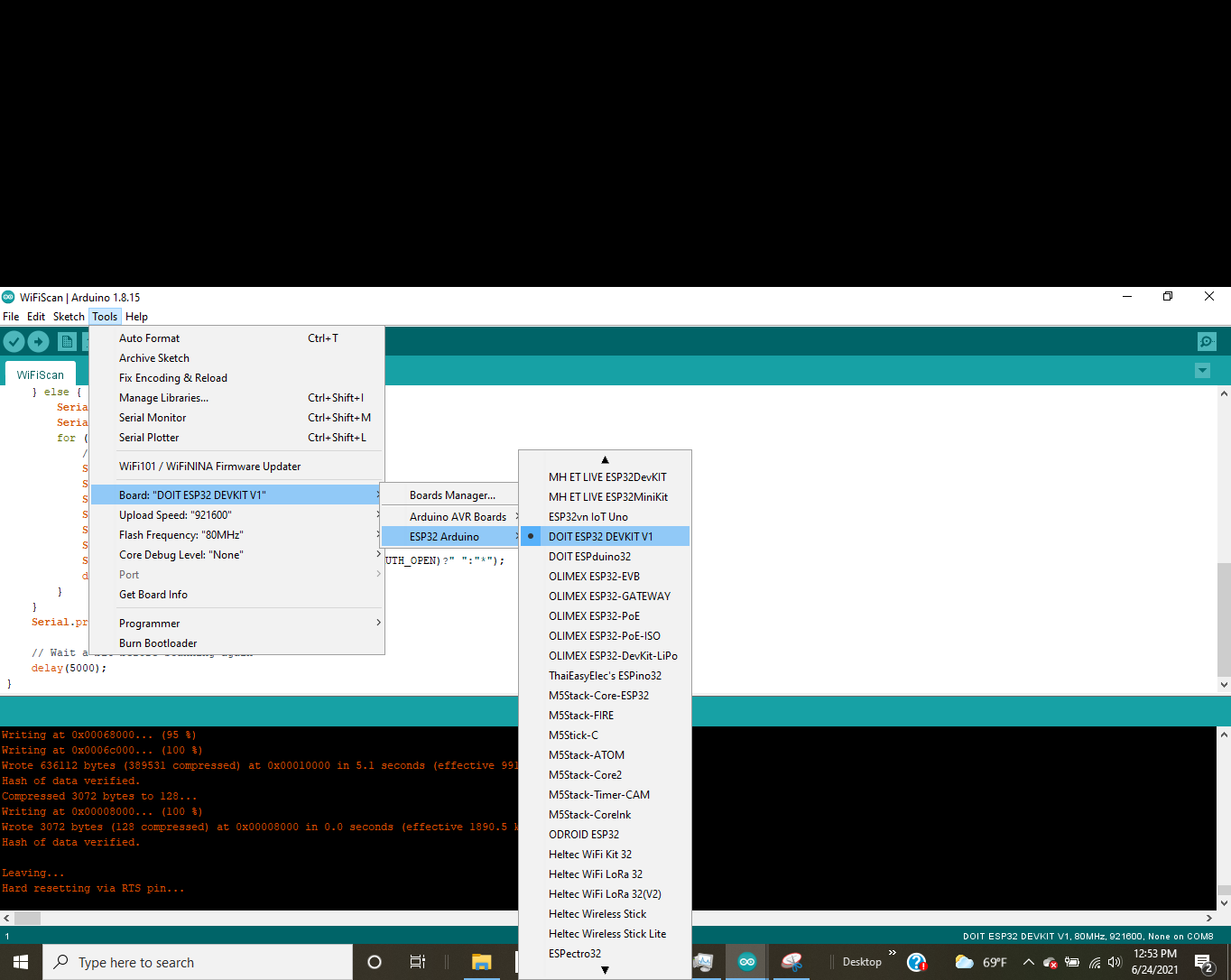
## Step 2 Downloading Arduino Software:

You are not required to learn much about coding in this course, however, you will still need to learn to download and use the Arduino integrated development environment (Arduino IDE). The esp32 is coded the same way you would code an Arduino. An IDE is software that helps you as the programmer program the specific microcontroller/microprocessor you’re using. The Arduino IDE is small and simple to use. Choose your operating system and follow the instructions to download Arduino IDE at [Getting Started with Arduino products](https://www.arduino.cc/en/Guide).

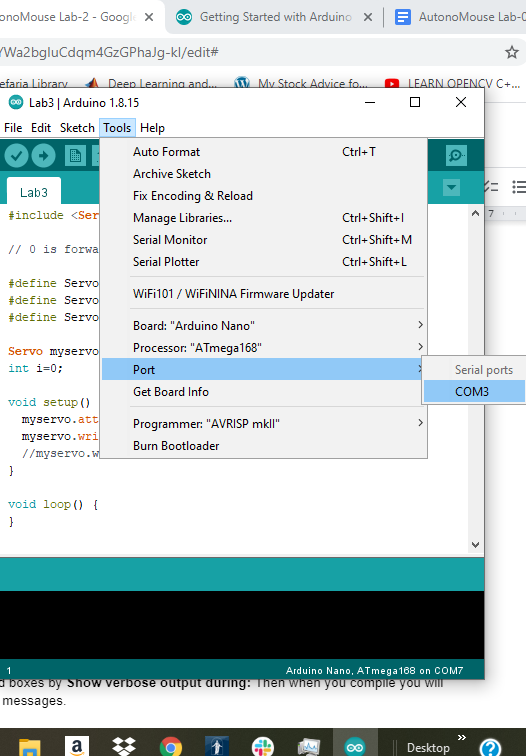
Once you’ve installed the Arduino IDE we need to download one last package to allow it to work with esp32. Follow the download instructions in this video: [Introduction to ESP32 - Getting Started](https://www.youtube.com/watch?v=xPlN_Tk3VLQ) (10:25-12:30)

Once you're done we will now try to get it to interface with your esp32.

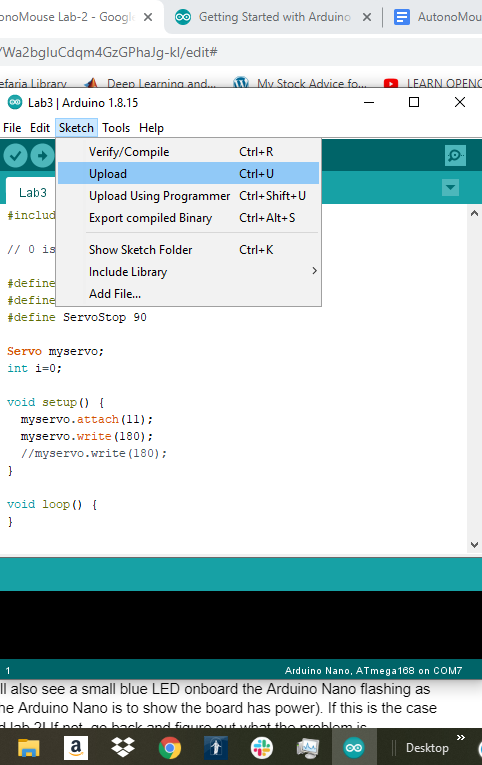
1. Open the IDE
2. Plug in your esp32 via usb (you will need a usb-A to micro-usb)
3. In **Tools/Board:** select **“DOIT ESP32 DEVKIT V1”** (this tells the IDE what architecture to expect)



1. Make sure **Tools/Upload Speed** is 921600, **Tools/Flash Frequency** is 80MHz, and **Tools/Core Debug Level** is set to “None”.
2. Under **Tools/Port** select which port on your computer the esp32 is plugged into. Note: this might be tricky. My computer’s usb driver wasn’t compatible with arduino and it wouldn’t install the arduino drivers which came with arduino IDE so I found a CH340 Driver at [CH340 Drivers for Windows, Mac and Linux](https://sparks.gogo.co.nz/ch340.html) that worked. First go to device manager (on windows) and look under ports for the port your arduino is connected to. (an easy way to do this is to plug in your arduino nano and see if a new port will pop up) If it does then you should be able to select which port in arduino IDE. If not, check if an unidentified usb device came up instead. If that’s the case then you need to install a compatible driver. Need to figure out how it works with MAC and Linux



1. Select **Sketch/Upload** to upload the software onto the device. If it worked correctly your esp32 should start doing what you told it to do. If not, you will see error messages saying that your code compiled incorrectly. You will need to fix whatever is causing the problem. Note: it’s helpful sometimes to get the whole error message when you’re debugging what went wrong. To change these settings go to **File/Preferences** and select compilation and upload boxes by **Show verbose output during:**. Then when you compile you will get the complete error messages.



## Step 3 Blinking the LED:

If you did steps 1 and 2 correctly then you should see the LED turn on for a second and then turn off for a second. If this is the case congratulations!

Troubleshooting Resources:

Try pushing the onboard enable button if the code doesn’t immediately run.

Look at the troubleshooting section of [Installing ESP32 in Arduino IDE (Windows, Mac OS X, Linux)](https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/) if the code doesn’t upload correctly.

### Extra Credit:

Change the time the LED is on by adjusting the values passed to the delay function in the code. The delay function takes a number as an argument and waits for that amount of time in milliseconds (eg 1000 milliseconds = 1 second)