Introduction to Arduino

Joe Wilkes, PhD

Week 1

Goal

- The class is designed to introduce a new user to the Arduino
  - Software for Arduino
  - Hardware for Arduino
  - Example projects

- At the end of the class, you will
  - Be able to use the Arduino,
  - Understand its hardware
  - Understand its software
  - Build your own projects
  - Expand your knowledge of C and the arduino

The Instructors

- Joe
  - Electrical Engineer
  - Co-author of Three books on cellular phone technology
  - Been building electronic and computer projects since teenager
  - Extra Class Ham- WA2SFF

- Neil
  - CDL board Member
  - Programming since 1978
  - Author: Linux Home Automation
  - General Class Ham – KD2ZRQ

The Class

- Week 1
  - Why the Arduino
  - Arduino Models
  - Arduino Specifications
  - Into to Development Environment
  - Ham radio and the Arduino
  - Simple Example
- Week 2
  - More Into to C
  - More Arduino Capabilities
  - Examples
- Week 3
  - Into to Hardware
  - GPIO
  - More Hardware
- Week 4
  - Getting to Know your Arduino in more detail
  - Experiments with Hardware and Software

Why the Arduino?

- Low cost
- Plenty of project ideas
- Plenty of free code to download and modify
- Easy to design own add on board called a "Shield"
- Multiple "Shields" available to increase utility

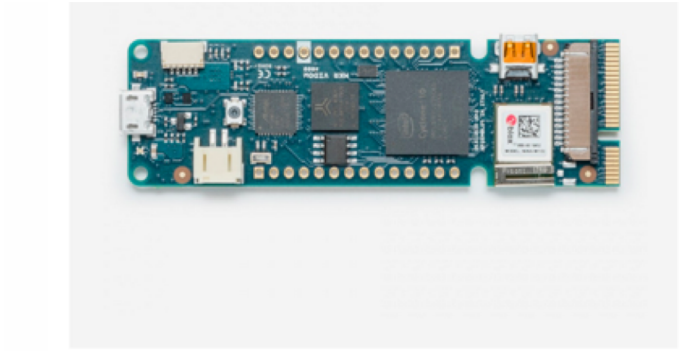
Setting up the Arduino

- Before the class started you should have installed and configured the Arduino Development System on your laptop

- For details on how to do this see:
  - https://github.com/drjoew/Computer_Deconstruction_Lab_Arduino_class
  - Download pre class instructions pdf for your operating system:
    - Windows
    - Mac
    - Linux

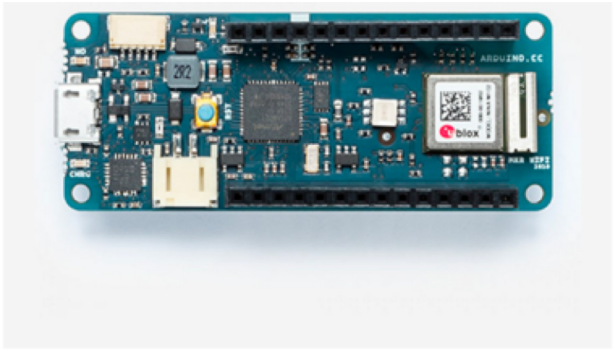
Arduino Family has multiple Models and many clones

We will Focus on the Uno



~~$74.90~~ $60.00

Arduino MKR Vidor 4000

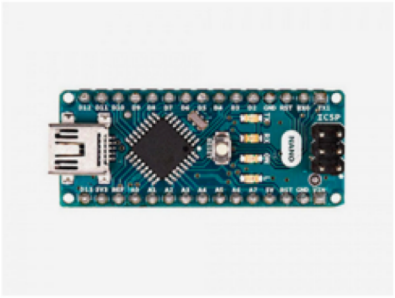


~~$33.90~~ $29.90

Arduino MKR WiFi 1010

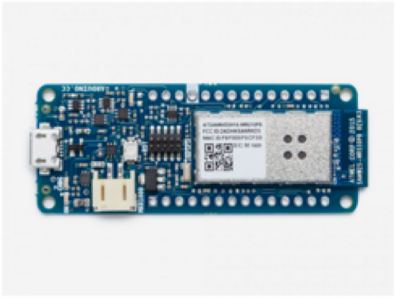
$22.00

Arduino Uno Rev3



$22.00

Arduino Nano



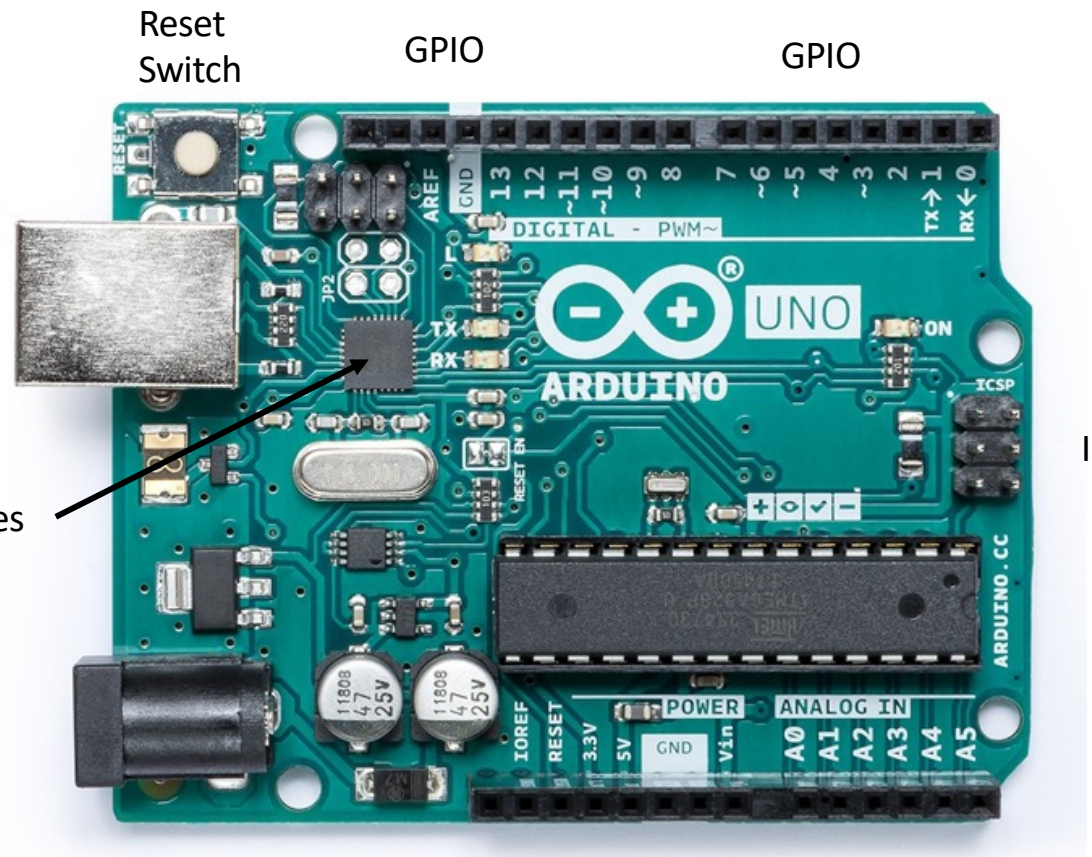
$34.99

Arduino MKR1000 WIFI

Arduino Uno Specs

	Microcontroller	ATmega328P	
	Operating Voltage	5V	
	Input Voltage (recommended)	7-12V	
	Input Voltage (limit)	6-20V	
	Digital I/O Pins	14 (of which 6 provide PWM output)	
	PWM Digital I/O Pins	6	
	Analog Input Pins	6	
	DC Current per I/O Pin	20 mA	
	DC Current for 3.3V Pin	50 mA	
	Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader	
	SRAM	2 KB (ATmega328P)	
	EEPROM	1 KB (ATmega328P)	
	Clock Speed	16 MHz	
	LED_BUILTIN	13	
	Length	68.6 mm	
	Width	53.4 mm	
	Weight	25 g	

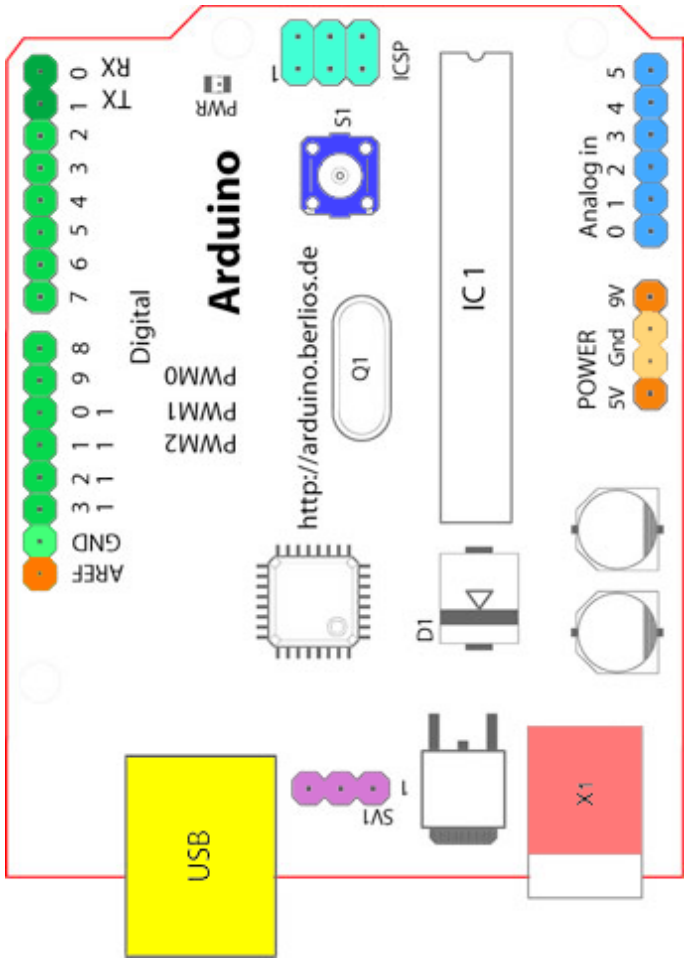
Arduino



Projects I have built with Arduino

- Variety of Robots
- HO train controller
  - Digital control of each engine
  - Trains controlled by laptop sending data over Bluetooth to controller
- Track side Bill Board
- Digital VFO for receivers
- SDR transmitter from QST
- Mixed Media Model Car with blinking wheels
- Scrolling "Merry Christmas" Display
- Blinking Christmas Tree
- Menorah with LED emulating flickering Candles

GPIO Pins



- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) –
  - *These pins cannot be used for digital i/o (**digitalRead** and **digitalWrite**)*
  - *if you are also using serial communication (e.g. **Serial.begin**).*
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)
- Analog In Pins 0-5 (light blue)
- Power and Ground Pins
  - (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC)
  - - X1 (pink)

Arduino Uno Pin Descriptions

	Pin Category	**Pin Name**	**Details**	
	power	**Vin**	**Vin: Input voltage to Arduino when using an external power source.**	
		3.3 volts	**3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.**	
		5 volts	**5V: Regulated power supply used to power microcontroller and other components on the board.**	
		GND	**GND: ground pins.**	
	Reset	**Reset**	**Resets the microcontroller.**	
	Analog Pins	**A0 – A5**	**Used to provide analog input in the range of 0-5V**	
	Input/Output Pins	**Digital Pins 0 - 13**	**Can be used as input or output pins.**	
	Serial	**0(Rx), 1(Tx)**	**Used to receive and transmit TTL serial data.**	
	External Interrupts	**2, 3**	**To trigger an interrupt.**	
	PWM	**3, 5, 6, 9, 11**	**Provides 8-bit PWM output.**	
	SPI	**10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)**	**Used for SPI communication.**	
	Inbuilt LED	**13**	**To turn on the inbuilt LED.**	
	TWI	**A4 (SDA), A5 (SCA)**	**Used for TWI communication.**	
	AREF	**AREF**	**To provide reference voltage for input voltage.**	

Some Basics to Get Started

Programming Environment

- The Arduino is programed in the C and C++ languages
- If you know C but not C++, you can do most things in C
- This class will provide a basic knowledge of C to get started
- In addition, the Arduino uses extensions to the language
  - to make Input and Output easier
  - Hide specifics of the ATmega328P chip from the programmer
- See https://www.arduino.cc/reference/en/

Syntax of C

- Almost all statement end with a semicolon ;
- Statements can be grouped together using curly braces {}
- Comments are preceded by two backslashes together //
- A group of comments is started by /* and ended by */

Input and Output Commands - Digital

- Used on pins 0-13 and A0-A5
- pinMode() Identifies a pin as input or output
- Syntax:
  - pinMode(pin, mode); where mode: INPUT, OUTPUT, or INPUT_PULLUP
  - Example: pinMode(13, OUTPUT); // sets the digital pin 13 as output

- digitalRead() reads input from a pin (HIGH or LOW)

- digitalWrite() writes output to a pin (HIGH or LOW)

Input and Output Commands - Analog

- Used on pins A0-A5
- analogRead()
- analogReference()
- analogWrite()

Delay()

- Used when we want the Arduino to not doing anything for a fixed amount of time
- Syntax
  - delay(x), where x is the time in thousands of a second (milliseconds)

- delay (1000) means do nothing for 1 second

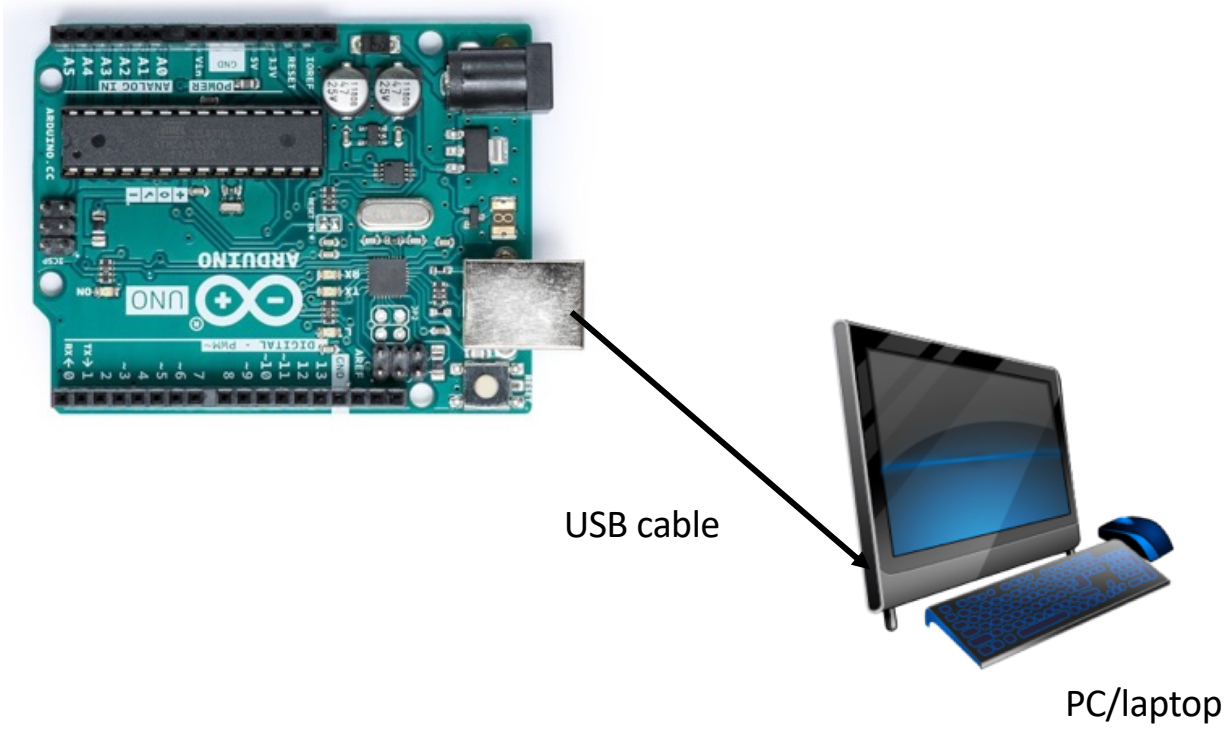
Errors in coding

- Typical errors in writing code
- Missing { or }
- Missing ;
- Misuse of Capital or lower case in a name

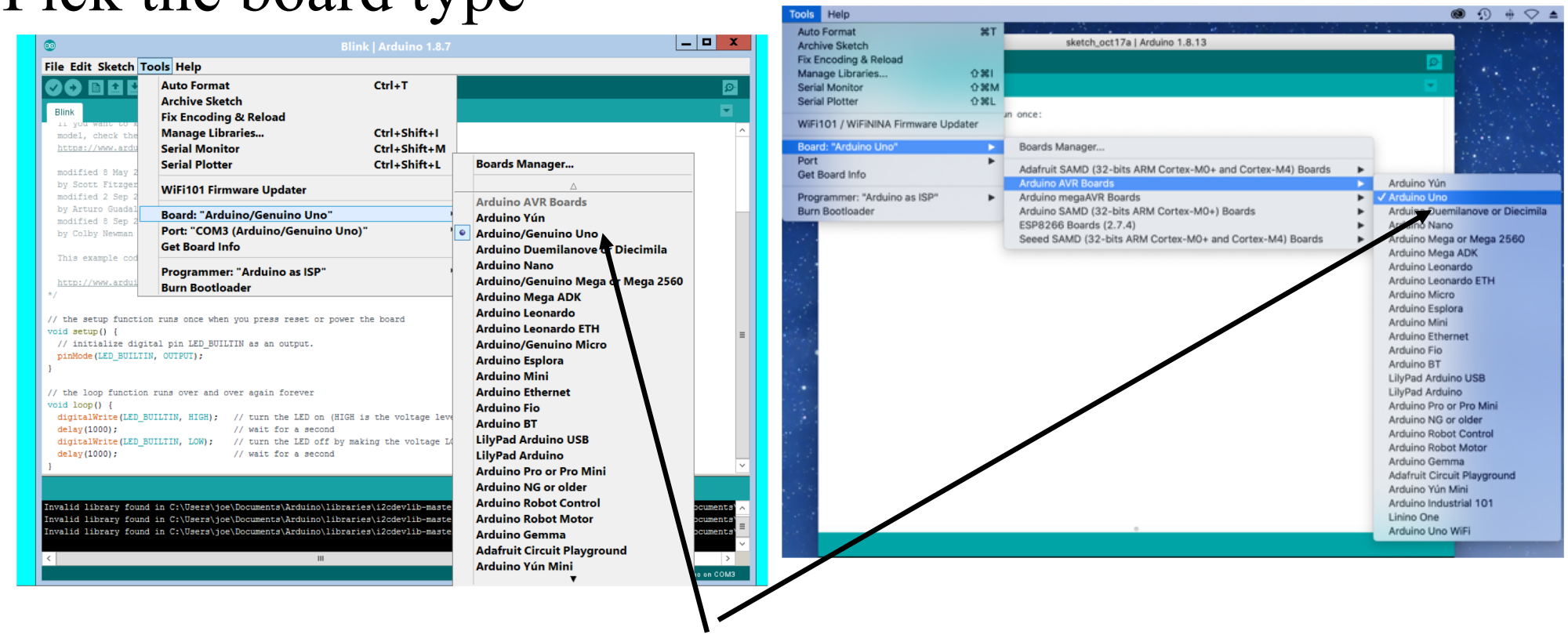
We now have enough info to do our first example

Blink an on-board LED

Demo Setup

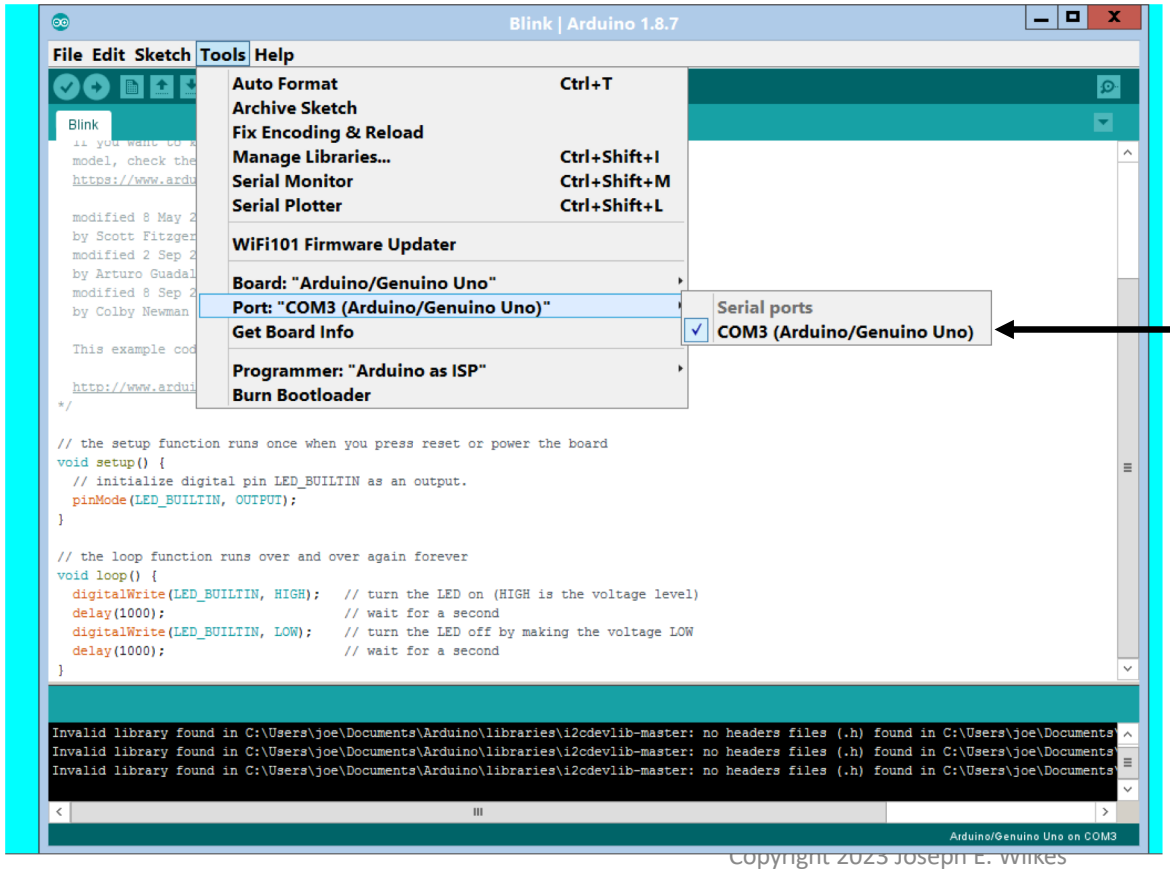


Setup for Development Environment: Pick the board type



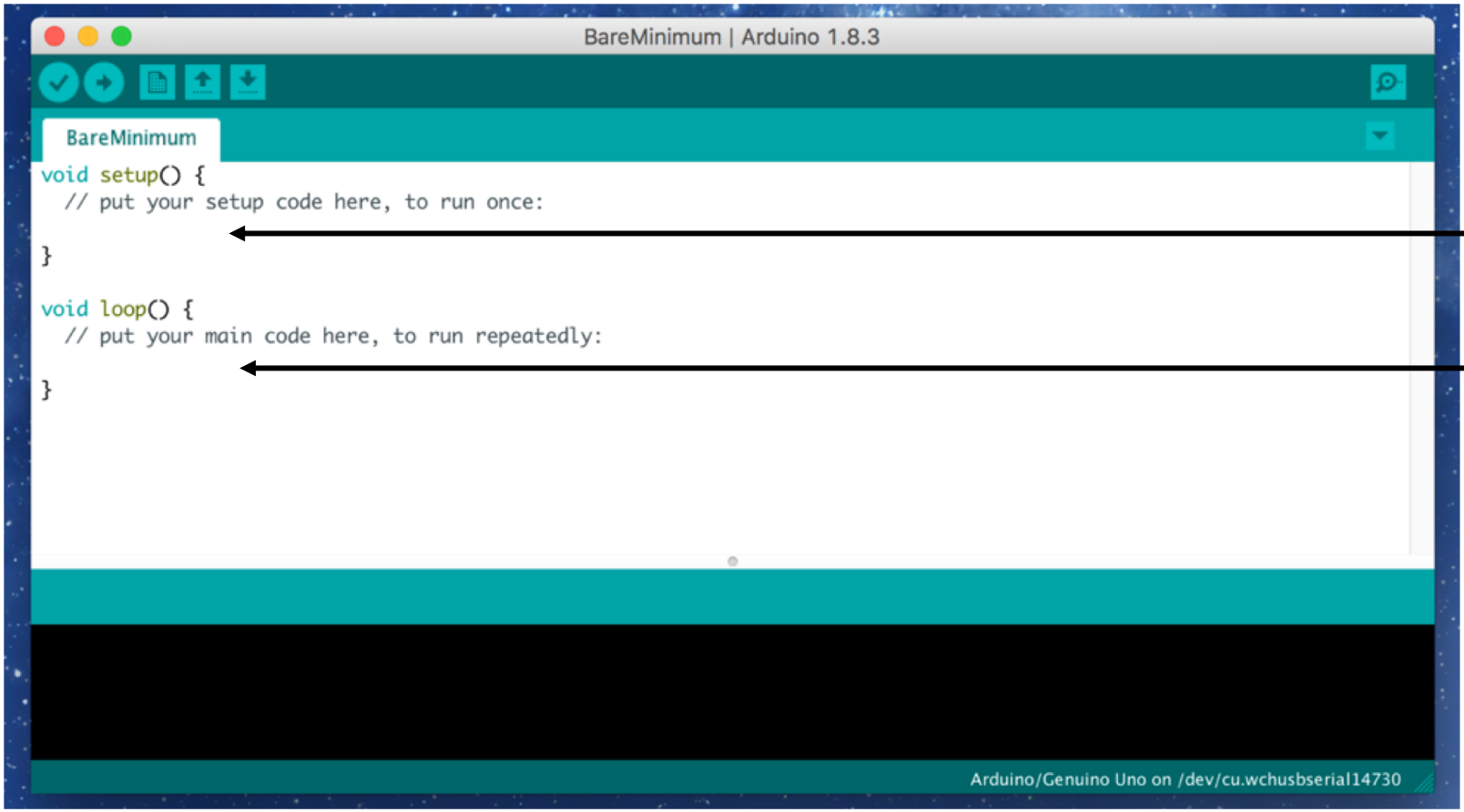
Select Uno

Setup for Development Environment: Pick the correct Serial Port



Select the serial port
For the Uno
Each laptop
will show a different
serial port

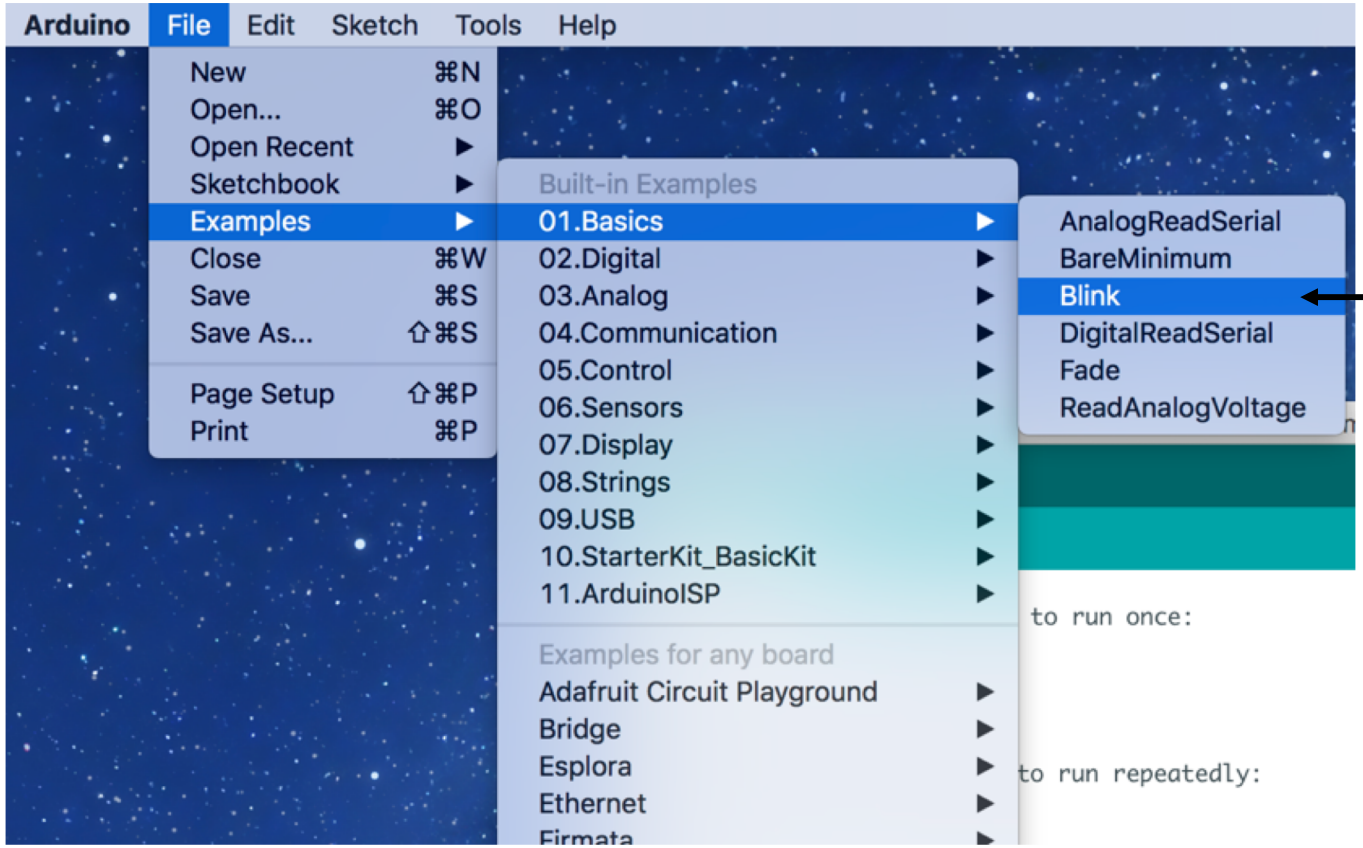
Development Environment



Startup code Goes here

Main code Goes here Runs in a Continuous loop

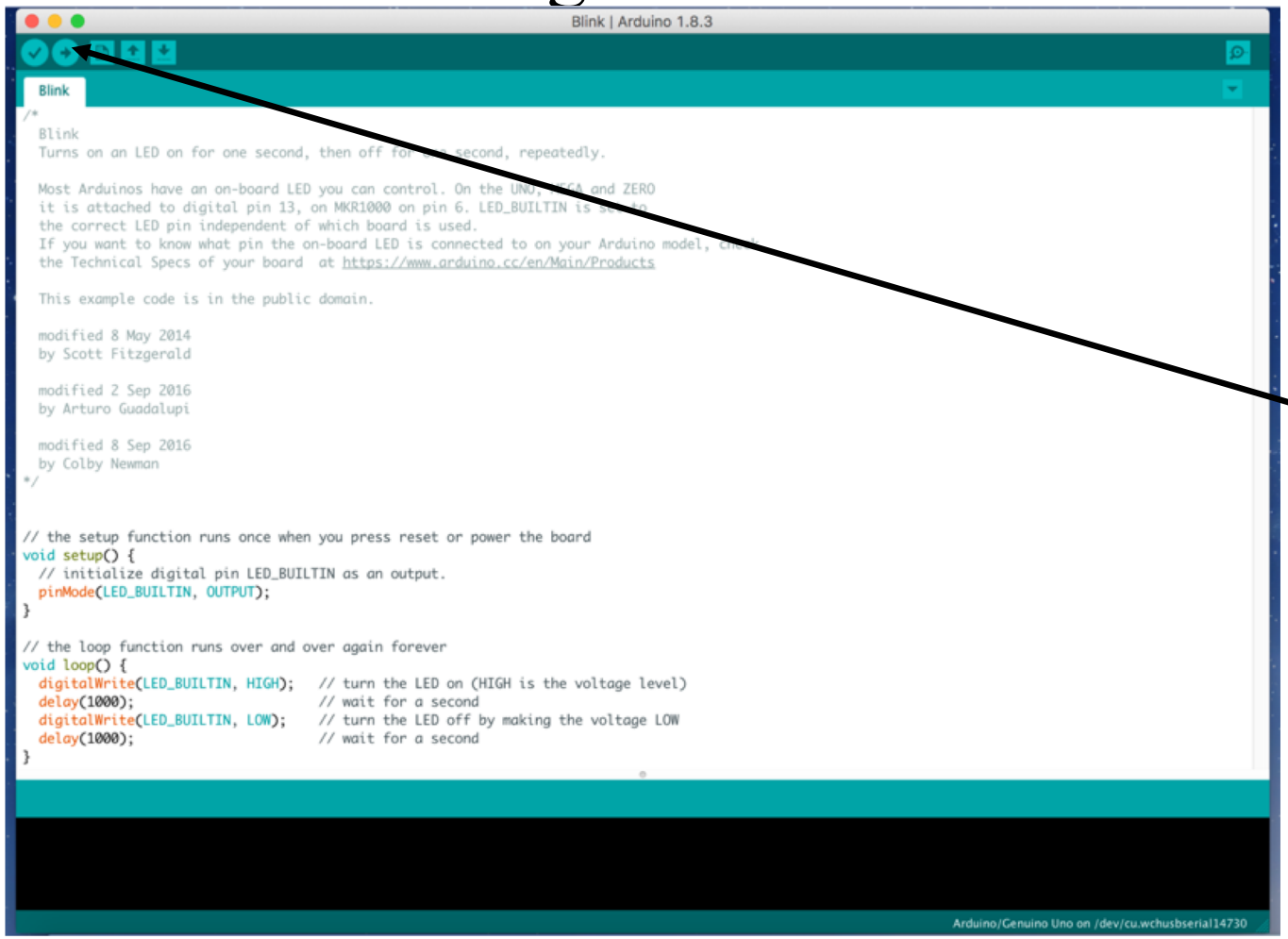
The Blink Program



There are a large number of examples in the development environment

Here we are opening the Blink Program

The Blink Program



Click here
To upload the
program in your arduino

The Blink Program: closer look

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Let's try blink on your Arduino

- Connect up your board to your laptop
- Select the Uno board and correct serial port
- Open the blink program
- Click the upload button
- After upload completes look at on board LED to see it blink
- Change the code to have the LED on for 2 seconds and off for ½ second
- Upload again
- See the blink rate change

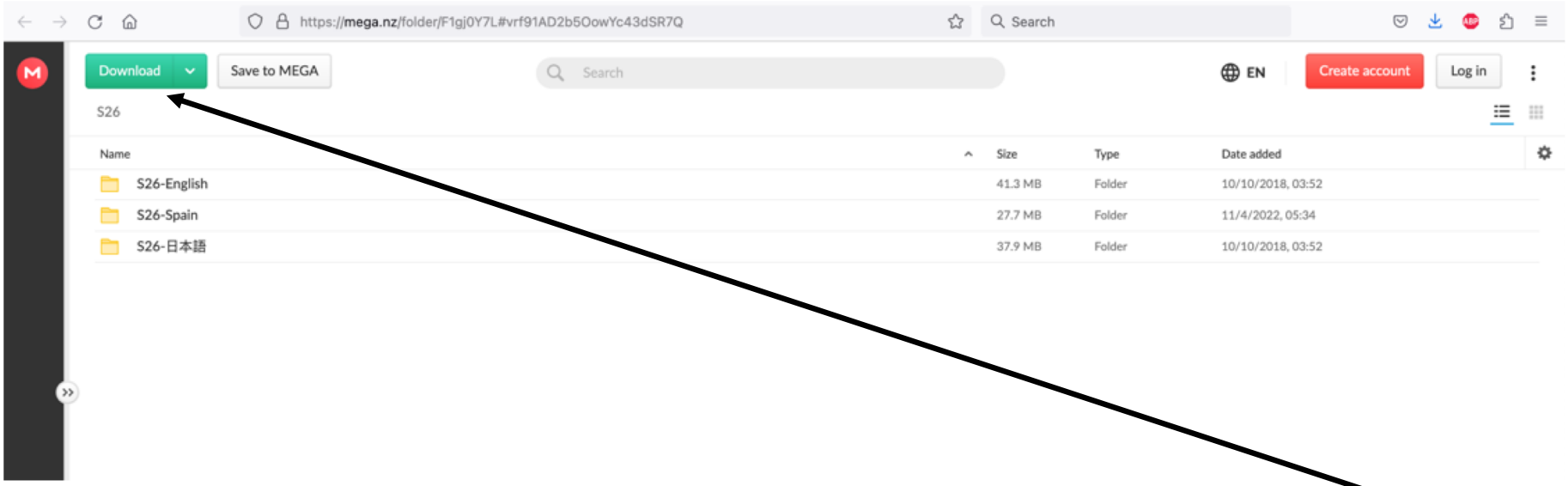
First Experiment

- We used the simple blink example to blink the built-in LED of the Arduino
- Now we will start using the starter kit to make a more complicated blink program

Some parts you will be using

- We will be using a kit a parts that is suggested for the class
- You can use your own parts but every experiment may not work the same
- If you want the kit of parts see:
  - https://www.amazon.com/gp/product/B01L0ZL8N6
- All the example code for the kit can be found on the enclosed CD
- It also can be found at https://mega.nz/#F!F1gj0Y7L!vrf91AD2b5OowYc43dSR7Q
- Click on Download as Zip in the upper right hand side of screen
- Some of the examples are written by the instructor
  - All examples available on github
  - https://github.com/drjoew/Computer_Deconstruction_Lab_Arduino_class
  - Download the example.zip file and un pack it to your computer

Get Code for Kit Here

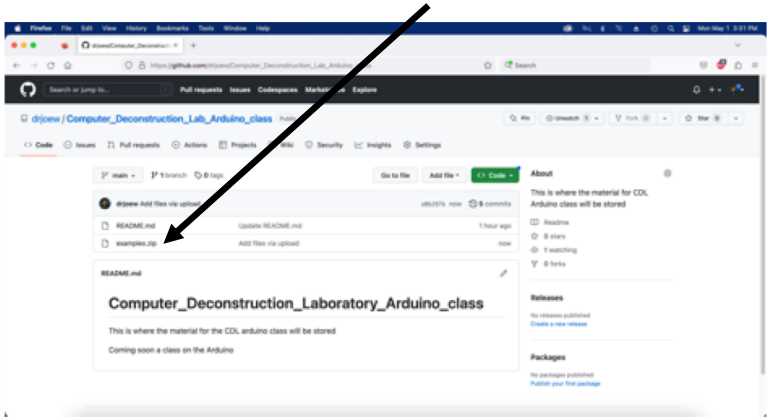


Click here
To download
Zip file

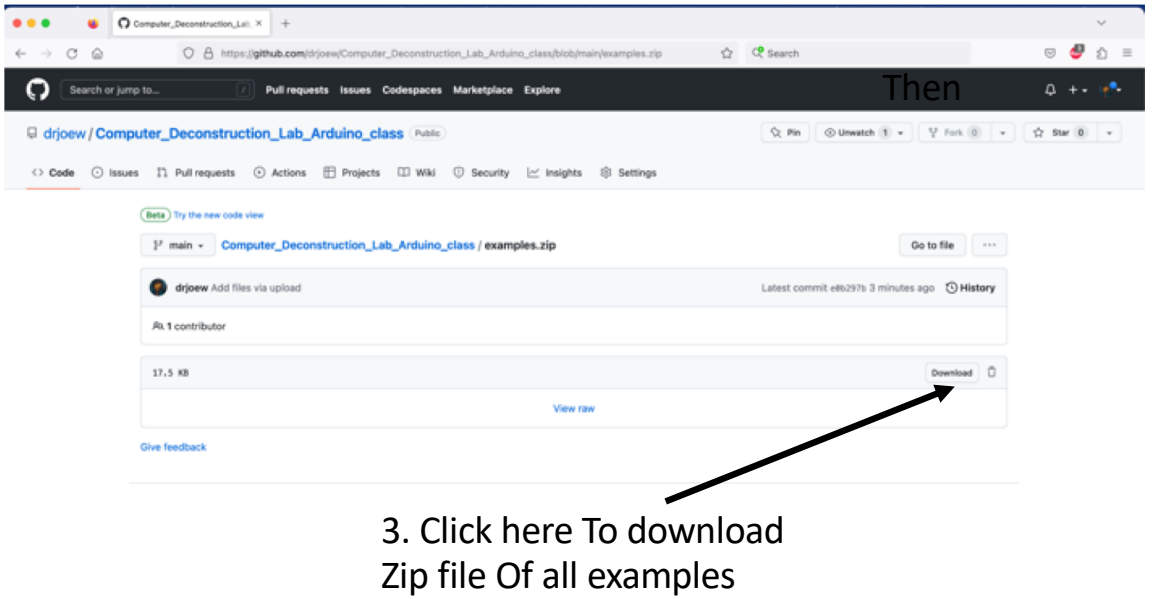
Github

- We have stored all the files for the class on the free site github
- It is setup so you do not need a login to get the files
- https://github.com/drjoew/Computer_Deconstruction_Lab_Arduino_class

1. Click here



2. the next screen will show



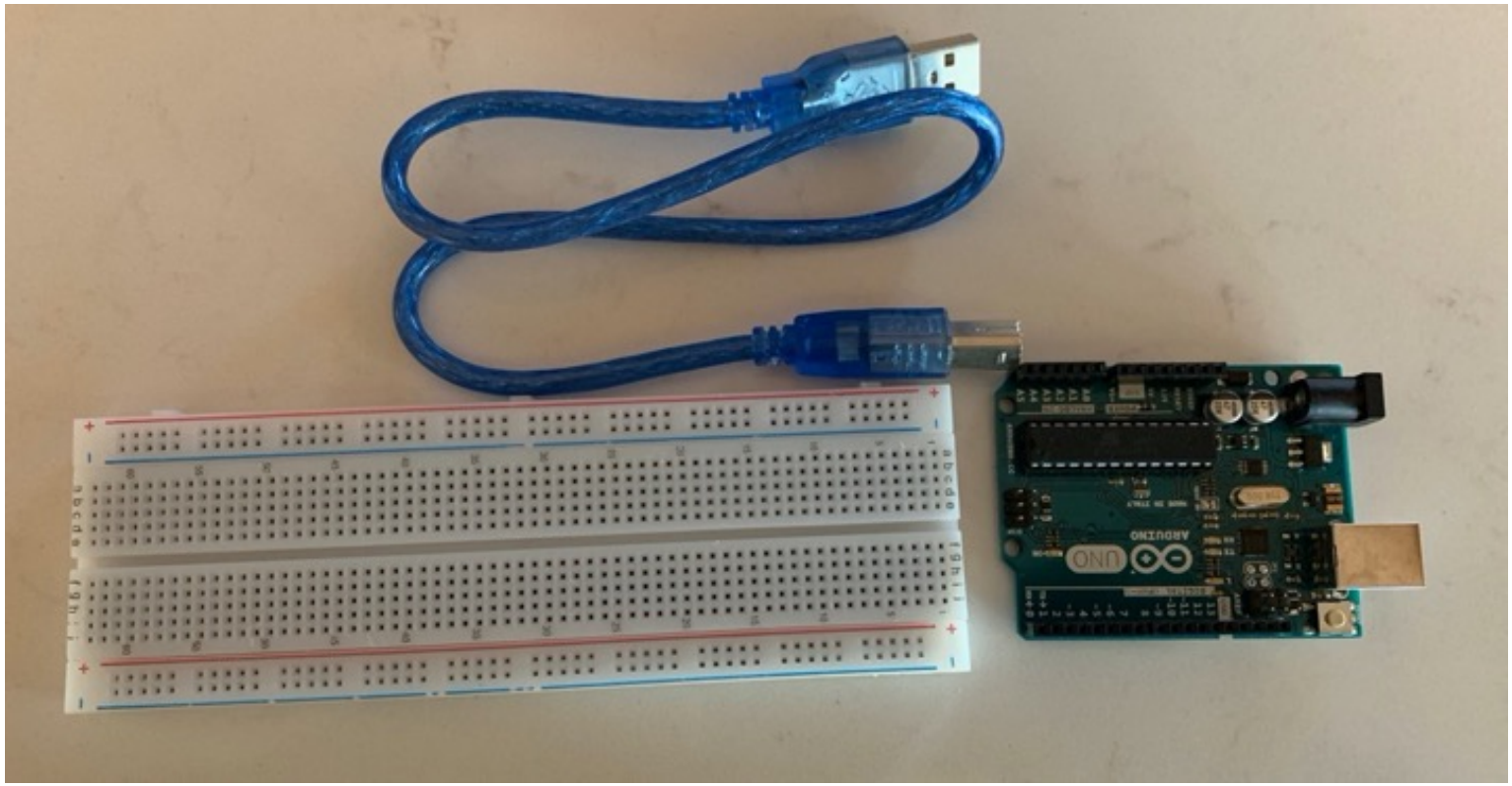
3. Click here To download
Zip file Of all examples

Approach for Each of the Experiments

- Describe the setup
- Describe the code
- Spend some class time building and testing the experiment

Experimental Setup

Cable to laptop



Prototype board

Arduino

Experimental Setup

- Make sure that the Arduino is unplugged from power
- Look in your kit and find:
  - Package of wires
  - The 830 Tie points breadboard
- Connect them together like the picture on the next few slides

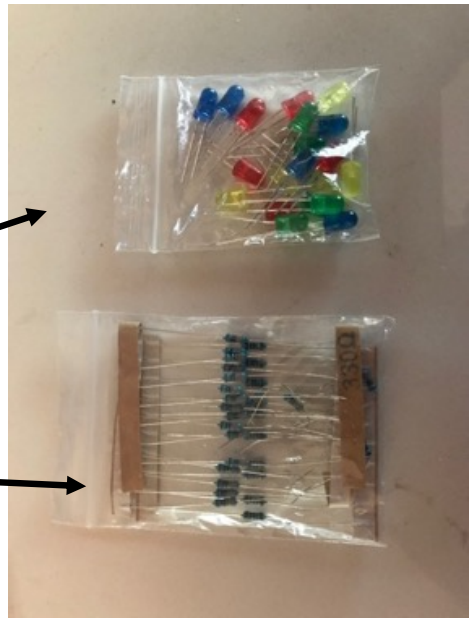
Wiring up Electronics

- In the US, the custom for electronic wiring is:
  - Black---Ground
  - Red ---- Power
  - Other colors – your choice but try to be consistent
- Minor Problem
  - Most starter kits do not include black and red wires (or only a few of them)
- Joe's Solution
  - Use Blue for Ground
  - Use Orange for Power
  - Do not use them in any other place

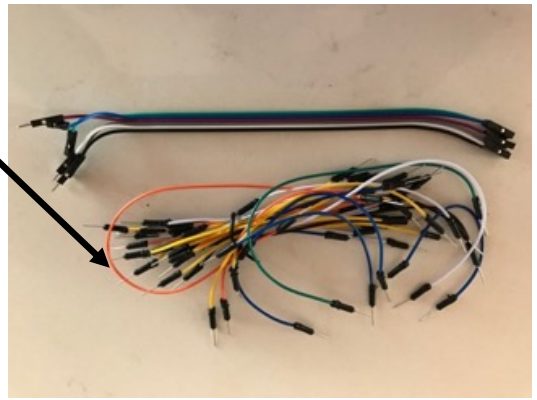
- Important Note! In the US, the custom for House Wiring is
  - Black --- Power (120 volts)
  - White --- Neutral (not ground but return for power)
  - Copper --- Ground
- Touching Power Wires can be deadly

First experiment:
Test wiring with LED

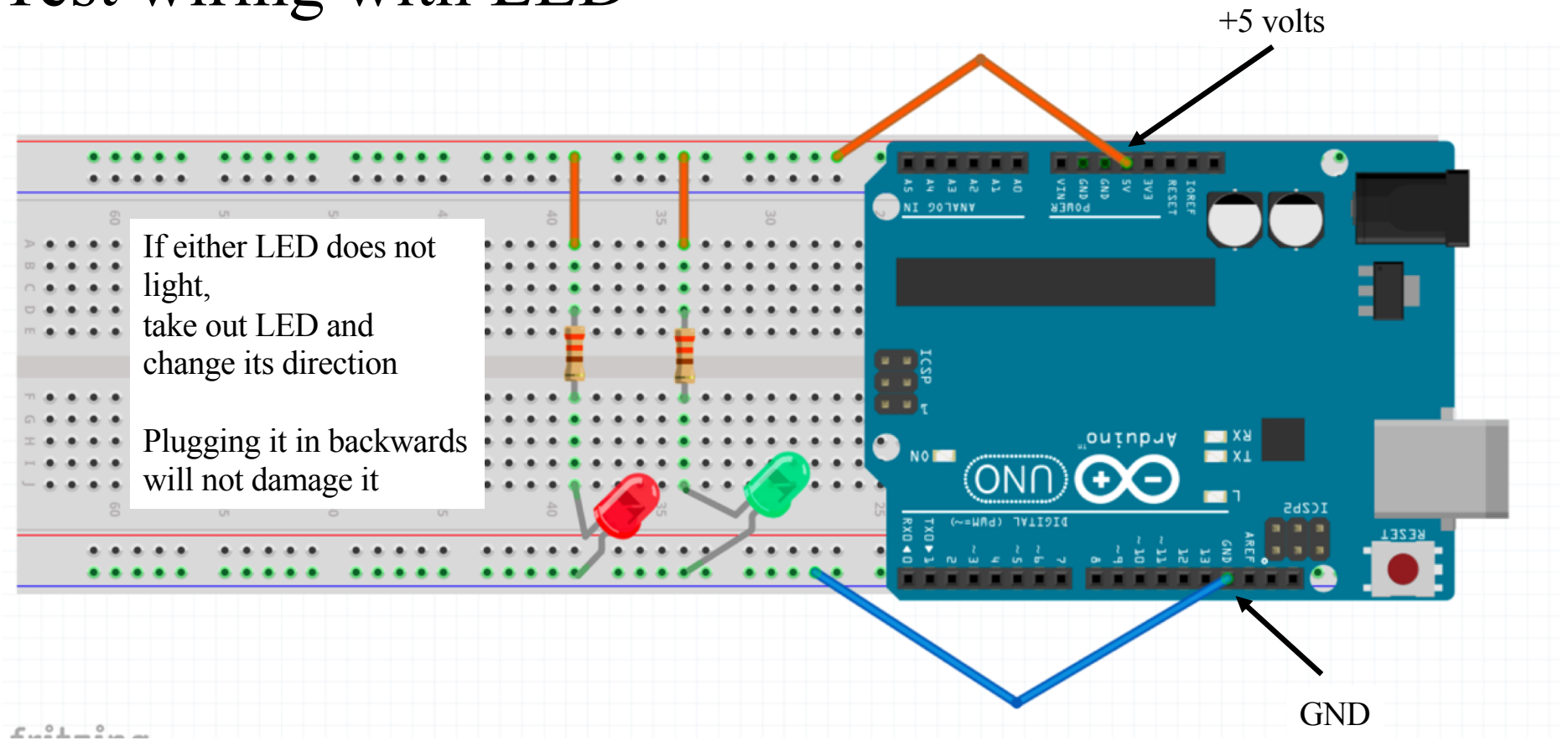
- Before we do anything, we need to make sure that the cables are connected correctly,
  - Otherwise we may destroy the Arduinoi
- Get an LED (green) and a 330 ohm resistor from your kit
- Get two long jumper wires from your kit
- Connect everything according the picture on the next page
- Apply power
- The LED should light up
- If not remove power and get help from Joe or Neil



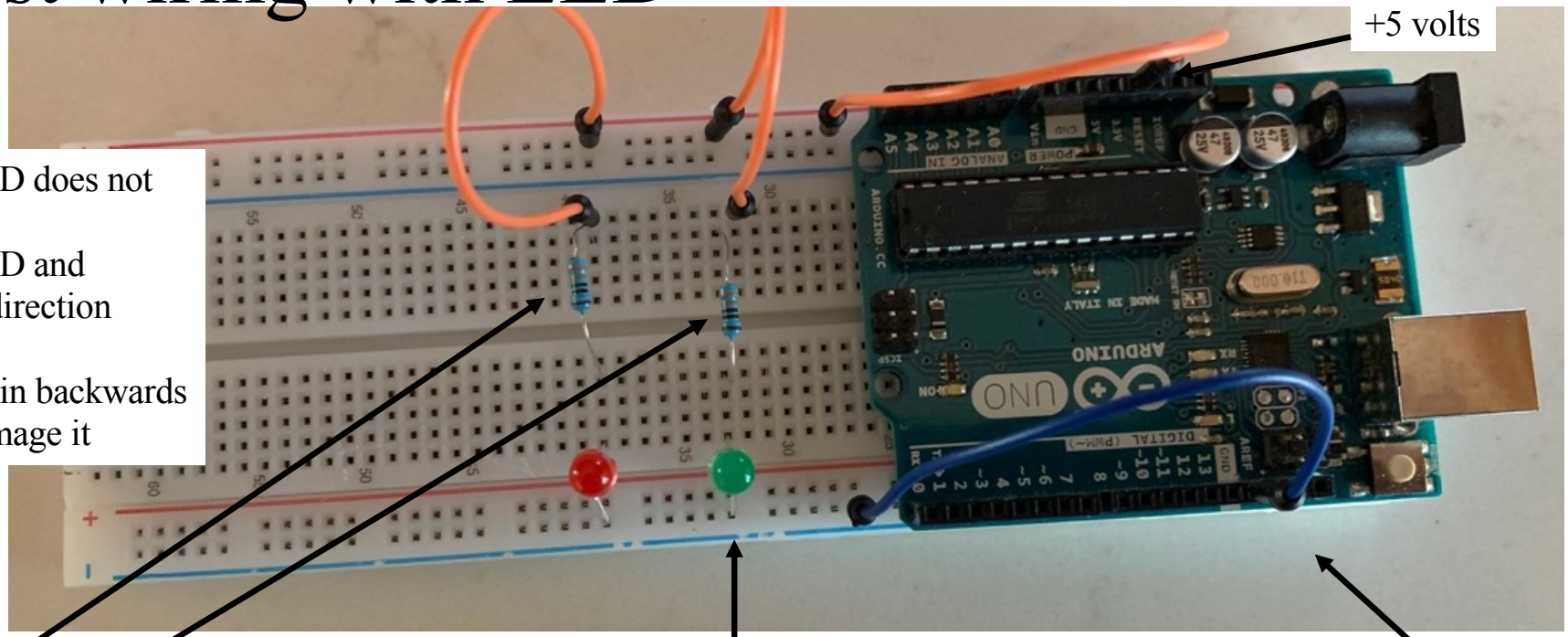
Resistor color code: band of orange, orange, brown, any color



Test wiring with LED



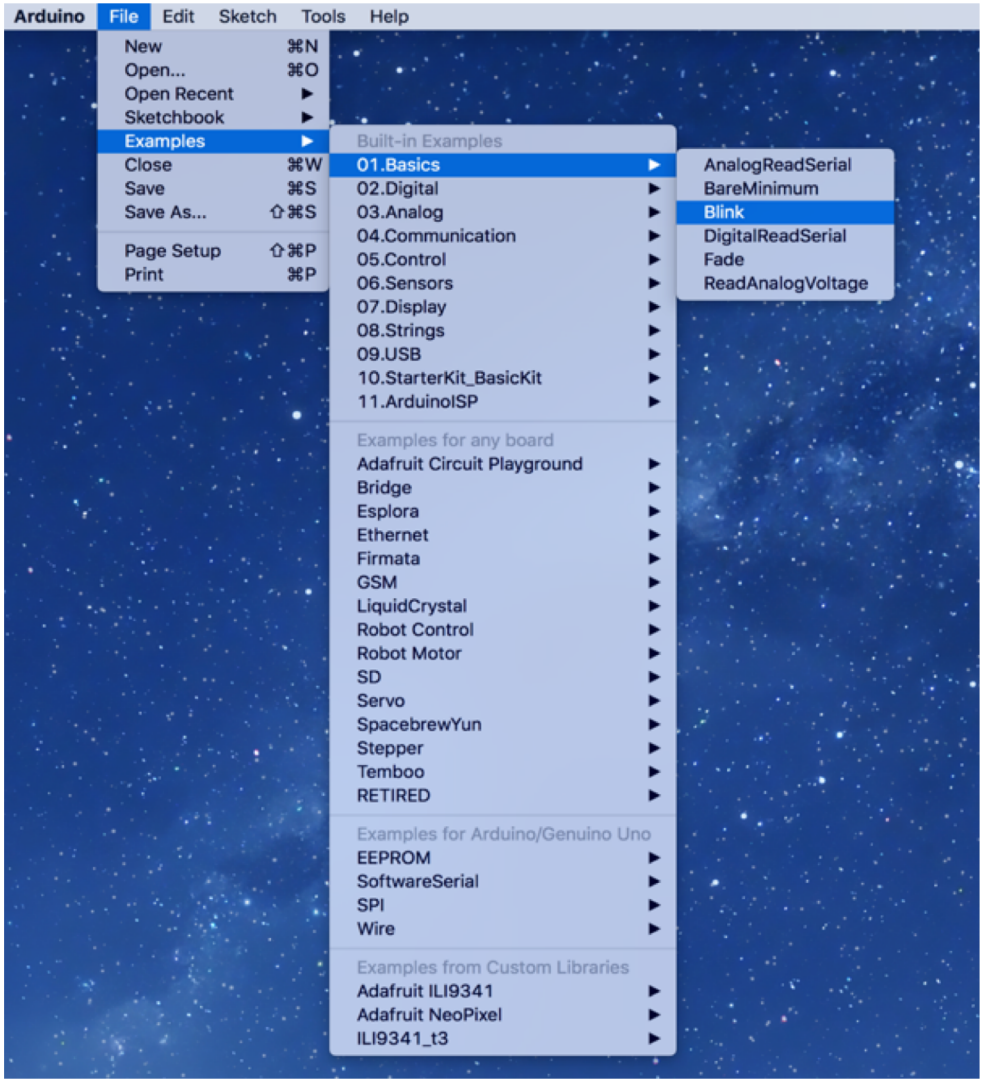
Test wiring with LED



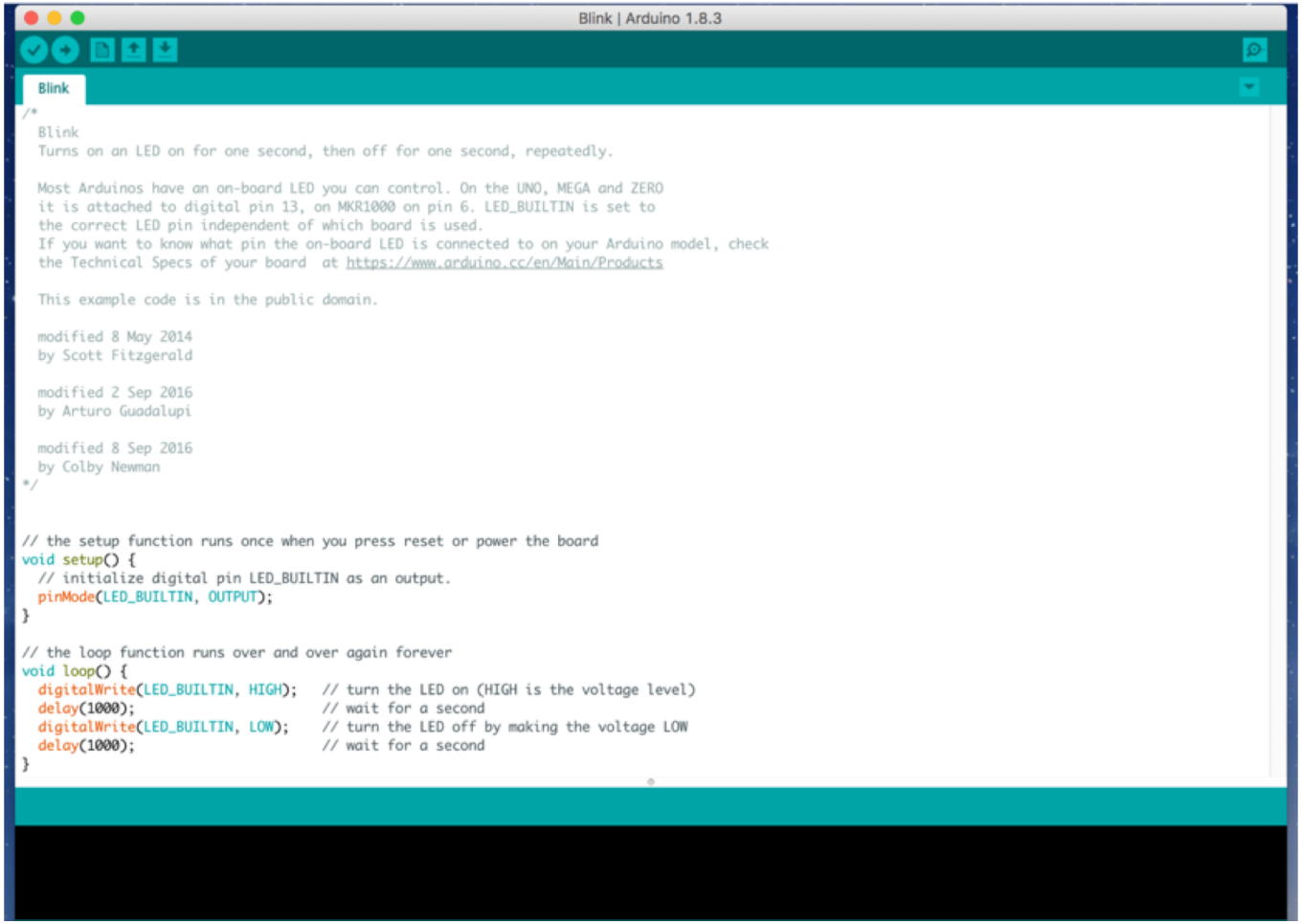
Blinking LED

- We are now going to move the positive wire of the LED to pin 13
- We will then use a C program to blink the LED
- We will modify the C program to change the blinking rate
- Load example program
  - Basic: blink

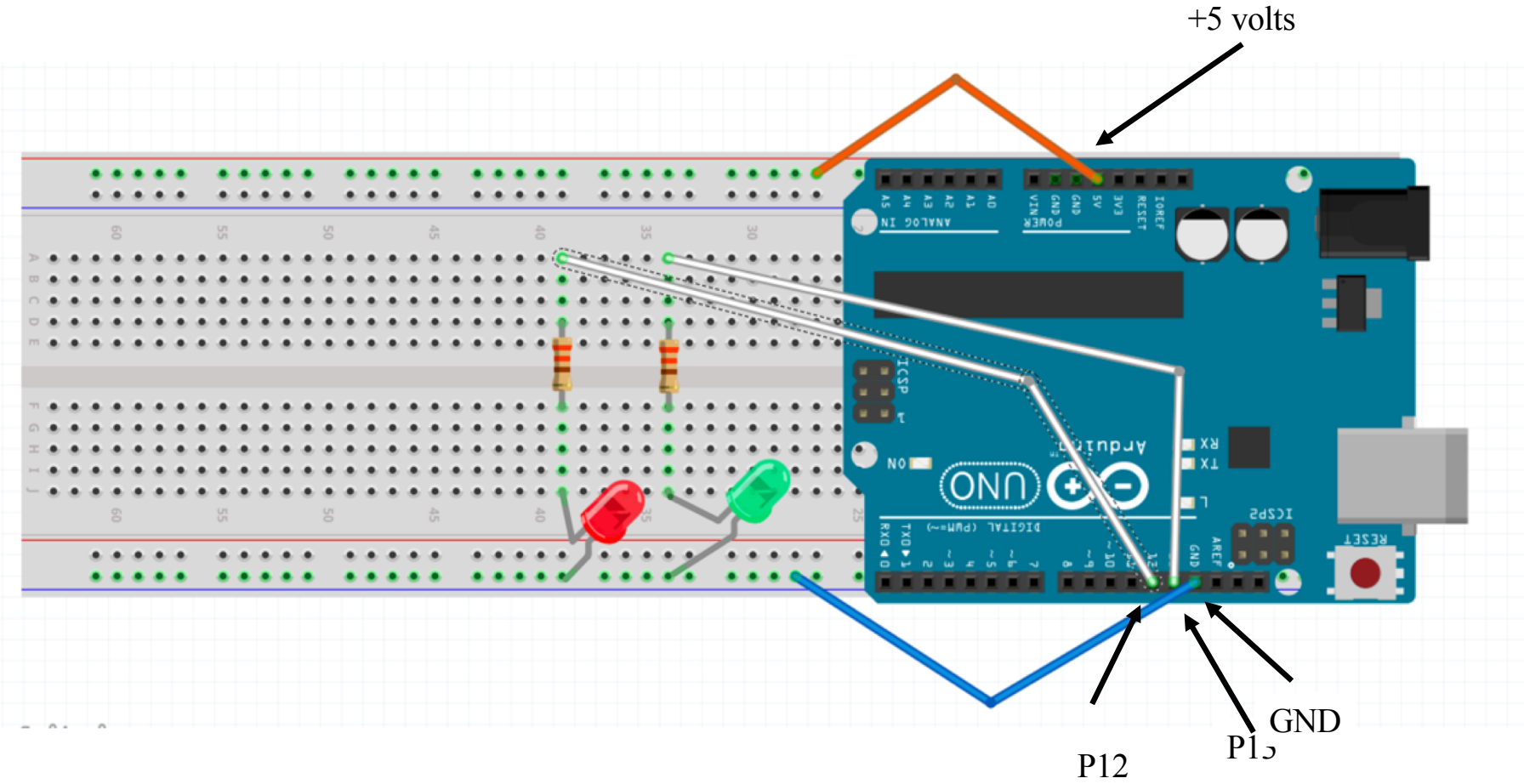
The blink program is part of the examples



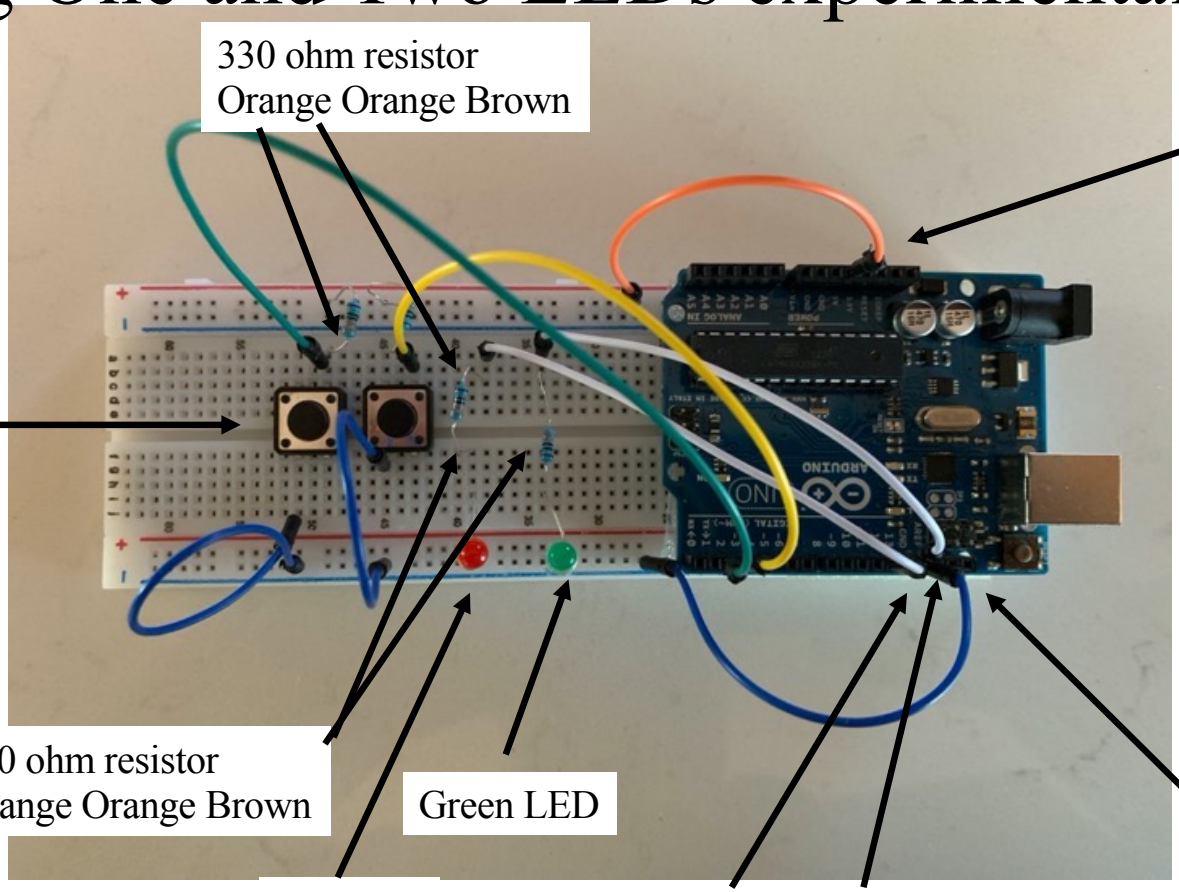
The blink Program (again)



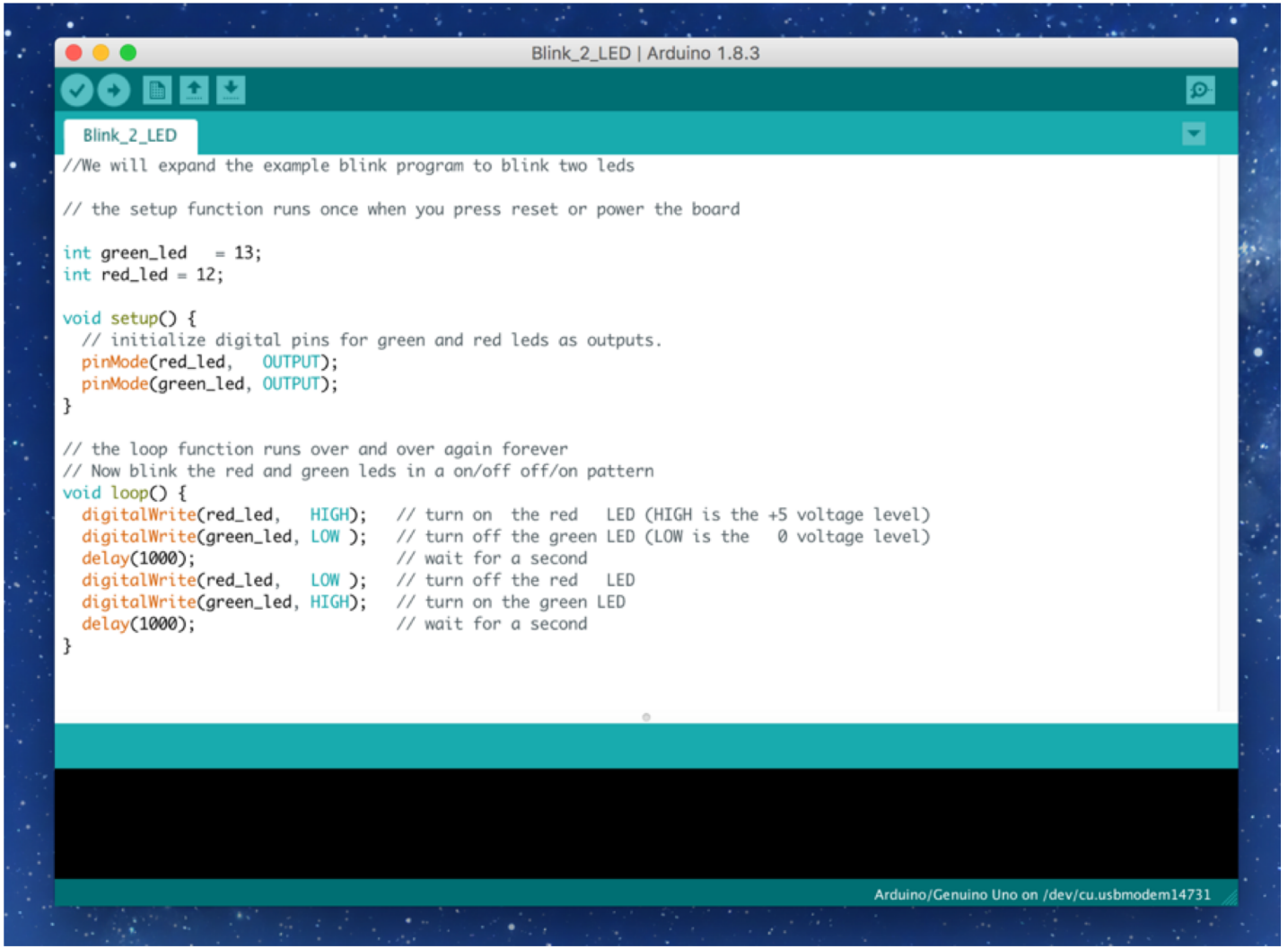
Blinking One and Two LEDs experimental setup



Blinking One and Two LEDs experimental setup



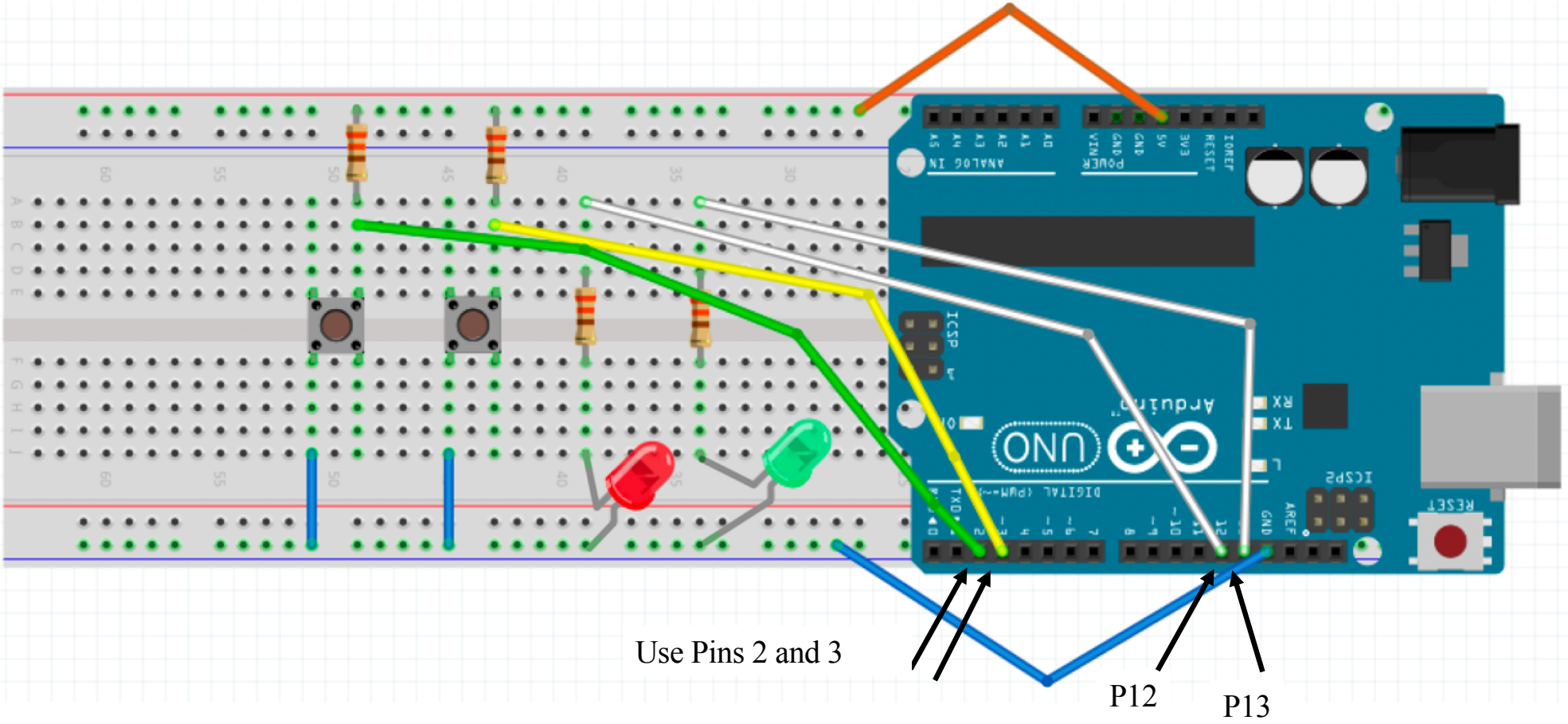
Blink_two_LED Program



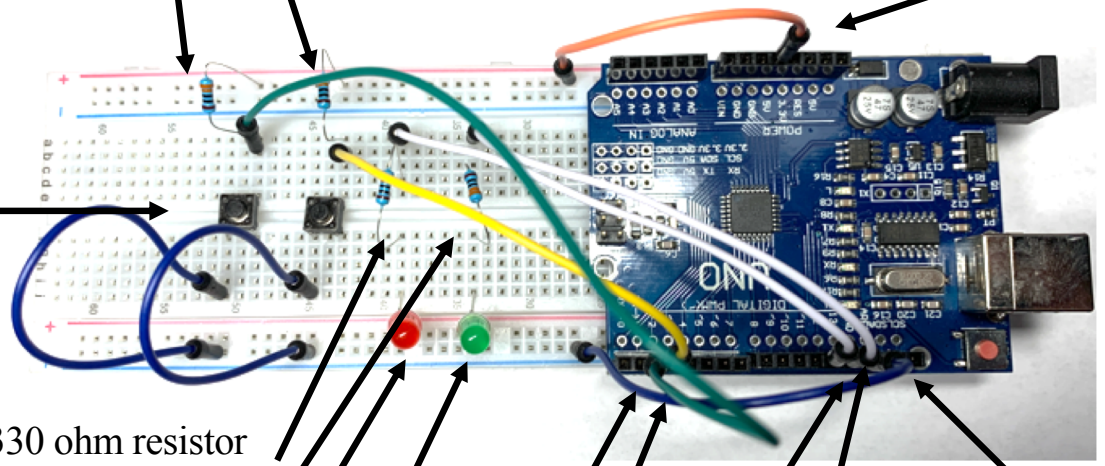
Turning on LEDs with buttons

- We are now going to add buttons to our LED project
- Many times we need buttons to do things
- This example will show how to use buttons

Turning on LED with buttons experimental setup



Turning on LED with buttons experimental setup



330 ohm resistor
Orange Orange Brown

+5 volts

Buttons

Install any parts
that were not installed
for the blink LEDs
experiment

330 ohm resistor
Orange Orange Brown

Green LED

Red LED

Use Pins 2 and 3

P12

P13

GND

Turning on LEDs with buttons

light_one_LED_with_button | Arduino 1.8.3

light_one_LED_with_button

```
// This code uses a button to light the green LED
  int green_led    = 13;
  int green_button = 3;
  int button_green_status = 0;

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin for the green LED as an output.
  pinMode(green_led, OUTPUT);
  // initialize the digital pin for the green button as input
  pinMode(green_button, INPUT);

}

// the loop function runs over and over again forever
void loop() {
  button_green_status = digitalRead(green_button);
  if (button_green_status == 0)
  {
    digitalWrite(green_led, HIGH);   // turn the green LED on (HIGH is +5 volts)
  }
  else
  {
    digitalWrite(green_led, LOW);   // turn the green LED off (LOW is 0 volts)
  }
}
```

Arduino/Genuino Uno on /dev/cu.usbmodem14731

The C language

- We have just given a quick overview of C
- C is very powerful and there is a rich set of capabilities
- As you get into coding for the Arduino you will need to make reference to more capabilities
- Always remember "google is your friend"
- For some free books see:
  - https://www.ossblog.org/learn-c-programming-with-9-excellent-open-source-books/
- It is legal to download and use these books
- The authors have given everyone copyright permission

Questions?

Short Lab time

- Play with the blink program and change the on and off times for the LED
- For example make the LED be on for 2 seconds and off for ¼ second.

Later exploration

- Explore the www.arduino.cc web site
- Examine https://www.arduino.cc/reference/en/ to learn more about the commands used by the Arduino
- Look at the extras for week 1 to learn more about C