Arduino for Musicians

Alex Grant

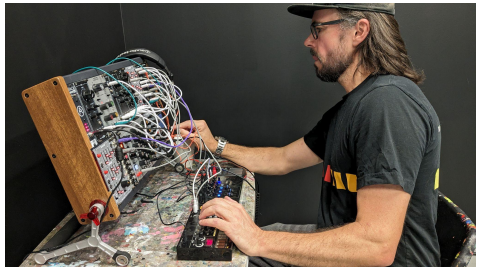
Workshop Plan

Week 1 [Tues, May 13th]
Introduction to Arduino: Focusing on the Arduino microcontroller, participants will be introduced to basics of electronics.

Week 2 [Tues, May 20th]
MIDI: An introduction to the MIDI standard for interconnecting music equipment. Participants will learn how MIDI sends messages for notes and control values. We will build a simple system using the Arduino to send MIDI commands to a digital audio workstation.

Week 3 [Tues, May 27th]
Modular Synthesis: Using VCV Rack participants will explore sound design and music production using modular synthesis.

Week 4 [Tues, June 3rd]
Homebrew MIDI Controllers: Bringing together the previous three workshops, in this session, participants will build custom MIDI controllers using Arduino and various sensors.



Prerequisites

Week 1 [Tues, May 13th]

- Laptop with Arduino IDE installed
- Optional: fritzing
- USB-A to USB-C converter or hub if needed
- Arduino starter kit (provided)

Week 2 [Tues, May 20th]

- Laptop with Ableton Live and MIDIView installed
- Alternatively Garageband, Tracktion or whatever DAW
- Headphones
- Arduino starter kit

Week 3 [Tues, May 27th]

- Laptop with VCV Rack Installed
- Headphones

Week 4 [Tues, June 3rd]

- Laptop with Arduino IDE, Ableton Live, VCVRack and MIDIView installed
- Headphones
- Arduino starter kit

Lab Guest

TheLab2024?

Download the Code

https://github.com/grantaj/labrats-arduino

git is a tool for tracking changes in files

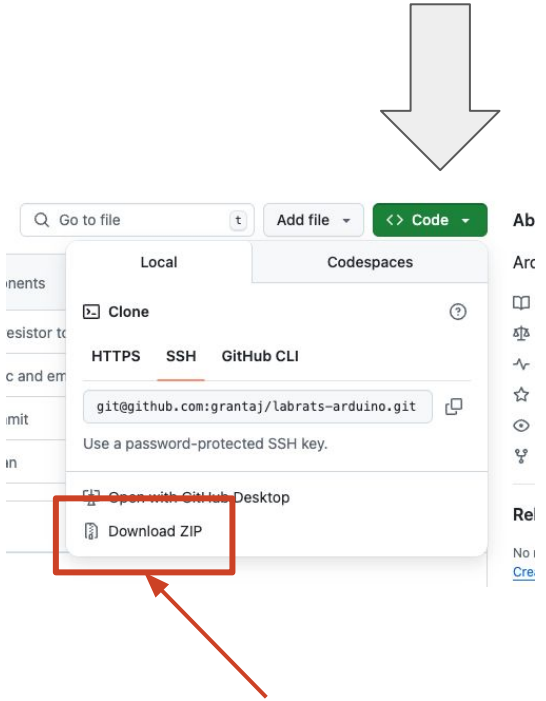
Saves versions of work so you can go back to earlier stages

You can work on different features or ideas in parallel using "branches"

Git works locally, but you can also use websites like **GitHub** to share your work and collaborate

Helps everyone stay organized and avoid overwriting each other's changes

Every change you save with Git is called a "commit", each commit has a message describing the change.



download the code as a .zip archive to your computer

Introduction to Arduino

Outline

What is Arduino and what is it for?

The Arduino IDE, uploading a basic sketch

Breadboards

LED

Controlling the rate

Switching on and off

RGB LED

- **What is Arduino?**
  - An open-source electronics platform for building **interactive objects**
  - Combines a **microcontroller** board with a simple **programming environment**

- **What can it do?**
  - **Sense** the environment using sensors (light, temperature, sound etc)
  - **Control** things (lights, motors, etc)
  - **Communicate** with other devices

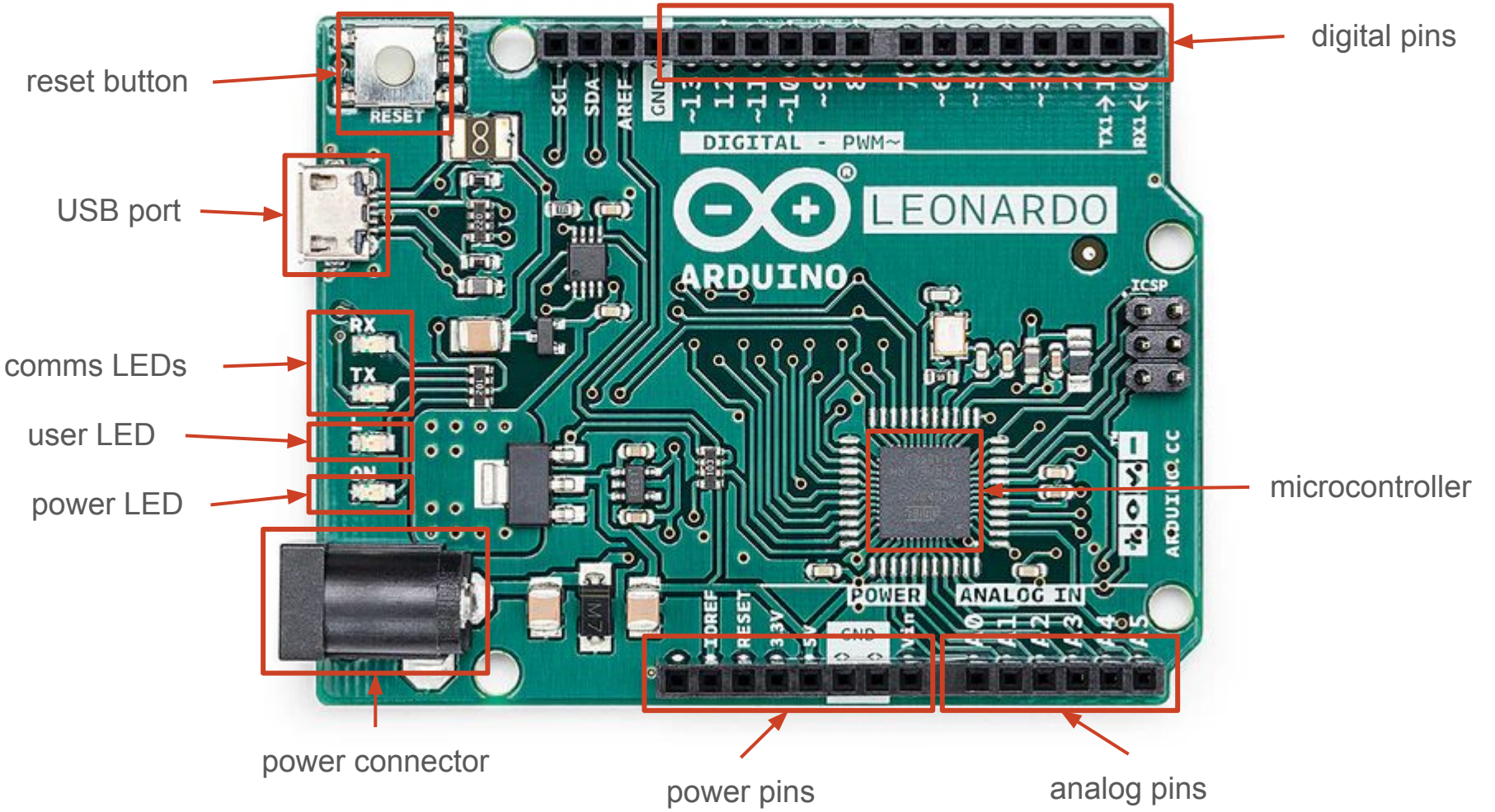
Feature	Nano	Uno	Mega 2560	Due	Leonardo
Microcontroller	ATmega328P	ATmega328P	ATmega2560	ATSAM3X8E	ATmega32u4
Architecture	8-bit AVR	8-bit AVR	8-bit AVR	**32-bit ARM**	8-bit AVR
Operating Voltage	5V	5V	5V	**3.3V**	5V
Digital I/O	14	14	54	54	20
PWM Pins	6	6	15	12	7
Analog Inputs	8	6	16	12	12
Flash Memory	32 KB	32 KB	256 KB	512 KB	32 KB
SRAM	2 KB	2 KB	8 KB	96 KB	2.5 KB
USB Type	Serial chip	Serial chip	Serial chip	Native + Prog	**Native**
Special Feature	Small		Lots of I/O	Fast CPU	**HID emulation**





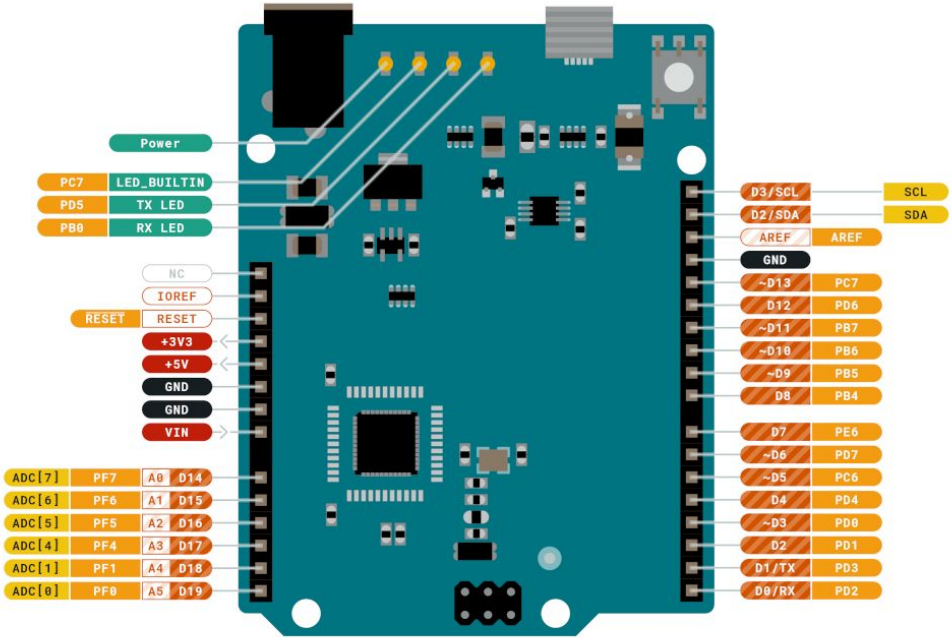
What does your Arduino Leonardo include?

- **Microcontroller**: the "brain" that runs your code (ATmega32u4)
- **Flash Memory**: 32 KB (non volatile memory for programs)
- **SRAM**: 2.5 KB (volatile memory for variables while your program is running)
- **EEPROM**: 1 KB (non-volatile memory for storing data, settings etc)
- **Digital & Analog I/O Pins** – for connecting sensors, LEDs, motors, etc.
- **USB Port** – for programming and power (supports HID & MIDI)
- **Power Supply Input** – to run independently of a computer
- **Reset Button** – to restart the program


reset button
USB port
comms LEDs
user LED
power LED
power connector
digital pins
microcontroller
power pins
analog pins
RESET
DIGITAL - PWM~
LEONARDO
ARDUINO
ICSP
RX
TX
POWER
ANALOG IN

ARDUINO

ARDUINO LEONARDO



Ground
Power
LED
Internal Pin
SWD Pin
Other Pin
Digital Pin
Analog Pin
Default
Microcontroller's Port

Arduino

- Single-chip: CPU + memory + I/O in one package
- Designed for sensing and controlling devices
- Limited resources (storage, processing power)
- No operating system (instant boot)
- Runs one program at a time
- Small (integrate into interactive objects)
- Low power (e.g. 50mW - can run off cheap batteries)
- $10s

Raspberry Pi

- Single-board computer running Linux
- Large memory, many interfaces
- Has digital I/O pins
- Can run multiple programs simultaneously
- Slower startup due to OS boot
- Ideal for complex tasks (e.g. web, media, AI)
- Supports multiple programming languages (Python, C++, etc.)
- Has HDMI, USB, audio, Wi-Fi, Ethernet
- Higher power consumption
- $100+

General-Purpose Computer

- Powerful processor
- External RAM, storage, and peripherals
- Designed for complex tasks (browsing, gaming, content creation)
- Can run large operating systems (e.g., Windows, Linux, macOS)
- Run many different programs (at the same time)
- High power (e.g. 15W = 300X arduino)
- $1000s

Outline

What is Arduino and what is it for?

The Arduino IDE, uploading a basic sketch

Breadboards

LED

Controlling the rate

Switching on and off

RGB LED

Connect Arduino & Start IDE

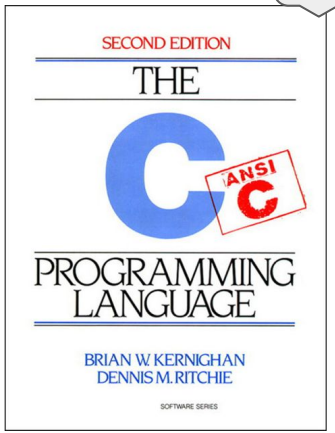
Connect Arduino via USB

Start Arduino IDE

Select Arduino Leonardo

The C Programming Language

read me



What is C?

- A human readable **language** that tells the Arduino what to do
- C is a mid-level language with low-level access to memory and hardware. It lets you write programs that directly control data and devices while still using structured code.
- The IDE **compiles** this to machine code that is uploaded to the Arduino

We will just jump in and use it, **learning C is not the focus of this workshop**

Basic Parts:

- **Functions**: Mini-programs that do things (e.g., `setup()`, `loop()`) or tell the Arduino to do something (`digitalWrite()`, `delay()`)
- **Variables**: Named boxes that store numbers or values. Variables have a *type* e.g. `int`, `bool`, `float`, `char`
- **Control Structures**: Let the program make decisions, optionally execute and repeat things `if`, `else`, `for`, `while`, `switch`

C syntax basics

- **Comments:** Notes in the code for humans, ignored by the Arduino
  Single line: `// this is a comment`
  Multi-line:
  ```
  /* This is
     a multi-line comment */
  ```
- **Semicolons:** Ends most lines of code — like a full stop in a sentence
  ```
  digitalWrite(13, HIGH);
  ```
- **Braces:** Group blocks of code together
  ```
  void loop() {
    // code goes here
  }
  ```
- **Parentheses:** Used for functions and conditions
  ```
  if (x > 10) {
      // do something
  }
  ```
- **Indentation:** is ignored, but useful for humans
- **Case-sensitive:** `digitalWrite` ≠ `DigitalWrite`

Category	Operator(s)	Example	Meaning
Assignment	`=`	`x = 5;`	Assign value
Arithmetic	`+ - * / %`	`x = a + b;`	Add, subtract, multiply, divide, remainder (modulo)
Comparison	`== != < > <= >=`	`if (x != y)`	Equal, not equal, less than, greater than, less than or equal, greater than or equal
Logical	`&& || !`	a && b	Logical AND, OR, NOT (different to bitwise)
Increment/Decrement	`++ --`	`x++; y--;`	Increase or decrease by 1
Compound Assignment	`+= -= *= /= %=`	`x += 2;`	Shortcut: add and assign, etc.

File > Examples > 01.Basics > Blink

```
// 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
}
```

Modifications

- Change the on and off times (delay time is milliseconds)
- Define our own int **variable** to store the LED pin number
- Use a **for loop** to blink the LED a fixed number of times
- Use a **while loop** to halt execution

Blink 10 times then stop

2-Blink-internal-loop

```
int ledPin = LED_BUILTIN;

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

// the loop function runs over and over again forever
void loop() {
 for (int i=0; i < 10; i++) {
   digitalWrite(ledPin, HIGH);  // turn the LED on (HIGH is the voltage level)
   delay(500);                      // wait for a second
   digitalWrite(ledPin, LOW);   // turn the LED off by making the voltage LOW
   delay(500);                      // wait for a second
 }

 while (true) {}
}
```

Outline

What is Arduino and what is it for?

The Arduino IDE, uploading a basic sketch

Breadboards

LED

Controlling the rate

Switching on and off

RGB LED

Breadboards

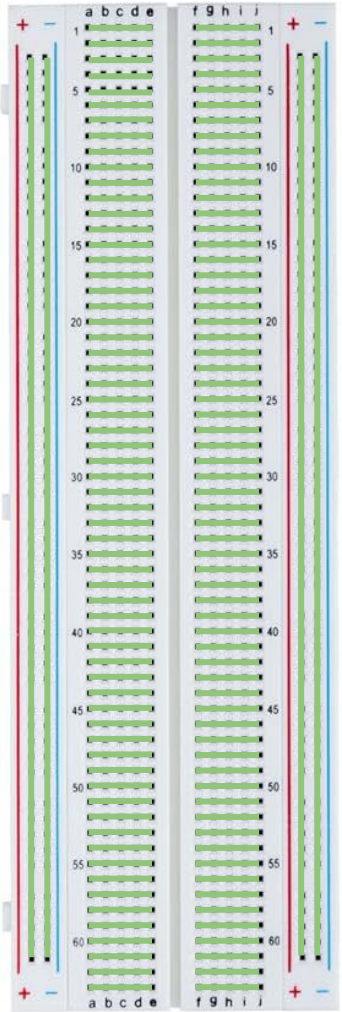
Quickly prototype circuits

Connect components using jumper wires

Each row of five pins is already connected

The + column and - column on each side is connected

Breadboard oriented design software (also handles schematics and code): `https://fritzing.org/`



Outline

What is Arduino and what is it for?

The Arduino IDE, uploading a basic sketch

Breadboards

LED

Controlling the rate

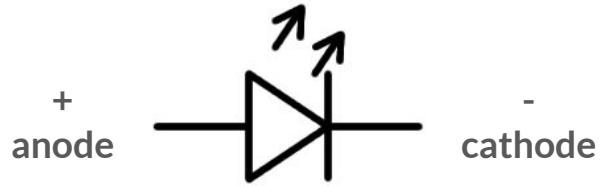
Switching on and off

RGB LED

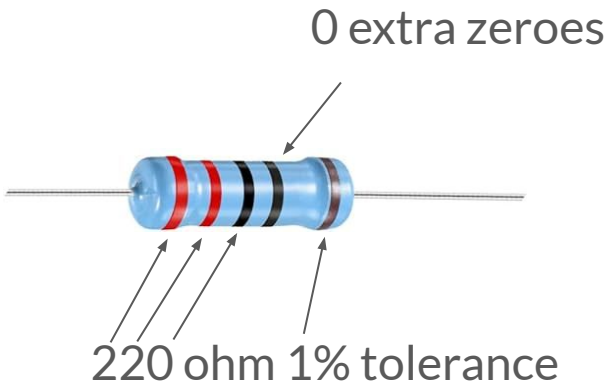
Light Emitting Diode

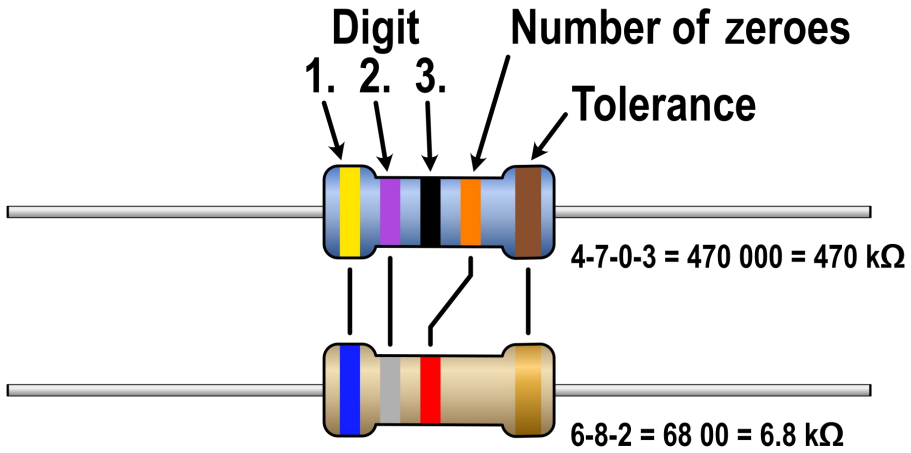
flat side
short leg
cathode
NEGATIVE

round side
long leg
anode
POSITIVE



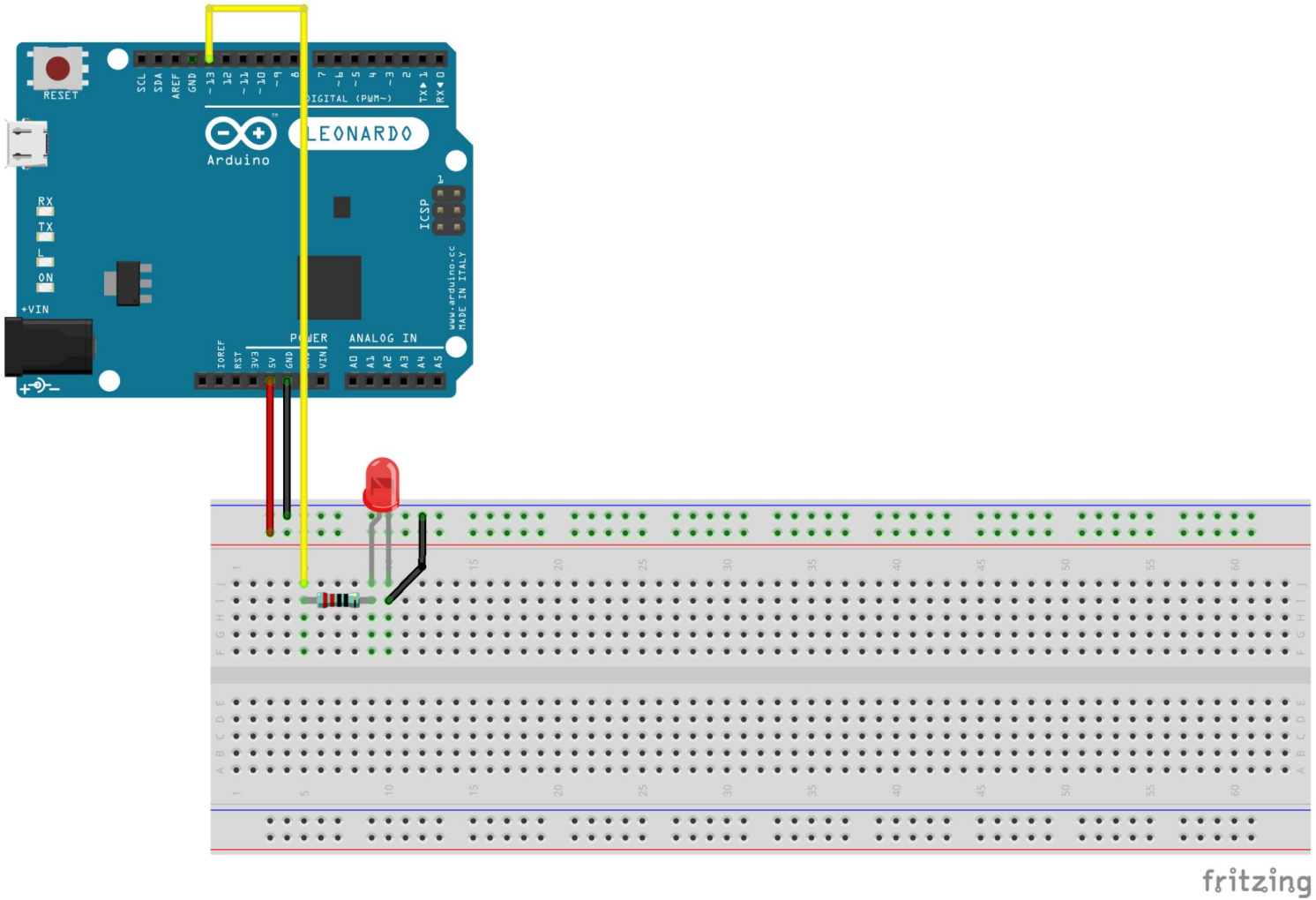
Resistor





Digit	0	1	2	3	4	5	6	7	8	9

Tolerance	Silver ±10 %	Gold ±5 %	±1 %	±0.5 %	±0.1 %


RESET
Arduino
LEONARDO
ICSP
RX
TX
L
ON
+VIN
POWER
ANALOG IN
www.arduino.cc
MADE IN ITALY
fritzing

Blink LED

1a-Blink

```
int ledPin = 13;

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

// the loop function runs over and over again forever
void loop() {
 digitalWrite(ledPin, HIGH);  // turn the LED on (HIGH is the voltage level)
 delay(500);                      // wait for a second
 digitalWrite(ledPin, LOW);   // turn the LED off by making the voltage LOW
 delay(500);                      // wait for a second
}
```

Modifications

- Change the LED to pin 12
- Alternate between blinking onboard and external LED
- Blink onboard and external LEDs at different rates:
    - Use **millis()** to keep track of time (in milliseconds)
    - Control program flow using **if** statements
    - Invert a **bool** variable using **!** (not) operator

Blink both LEDs

3-Blink-internal-external

```
const int led1 = 13;       // Built-in LED
const int led2 = 12;       // External LED

unsigned long previousMillis1 = 0;
unsigned long previousMillis2 = 0;

const unsigned long interval1 = 1000; // 1 second
const unsigned long interval2 = 100;  // 0.1 seconds

bool led1State = false;
bool led2State = false;

void setup() {
 pinMode(led1, OUTPUT);
 pinMode(led2, OUTPUT);
}

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

 // Blink LED1
 if (currentMillis - previousMillis1 >= interval1) {
   previousMillis1 = currentMillis;
   led1State = !led1State;
   digitalWrite(led1, led1State);
 }

 // Blink LED2
 if (currentMillis - previousMillis2 >= interval2) {
   previousMillis2 = currentMillis;
   led2State = !led2State;
   digitalWrite(led2, led2State);
 }
}
```

Outline

What is Arduino and what is it for?

The Arduino IDE, uploading a basic sketch

Breadboards

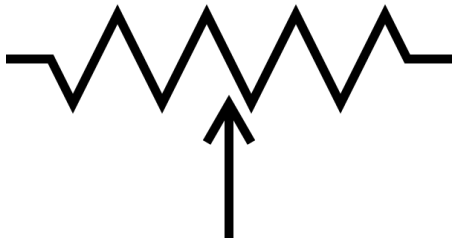
LED

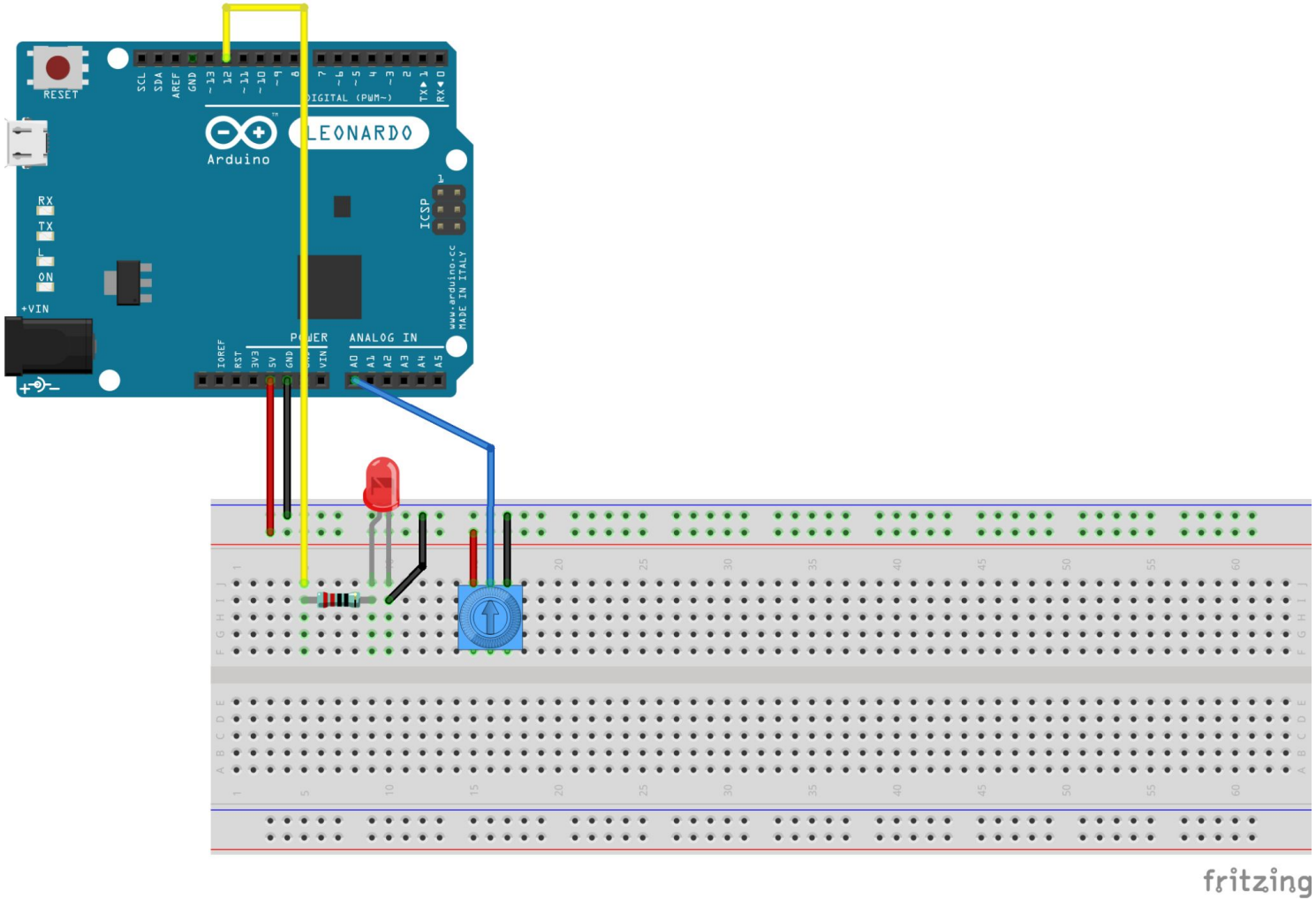
Controlling the rate

Switching on and off

RGB LED

Trimpot (variable resistor)




RESET
LEONARDO
Arduino
RX
TX
ON
+VIN
ICSP
POWER
ANALOG IN
fritzing

Read an analog value

4-Analog-read

```
const int potPin = A0;  // Potentiometer connected to analog pin A0

void setup() {
 Serial.begin(9600);   // Start the Serial communication at 9600 baud
}

void loop() {
 int potValue = analogRead(potPin);  // Read analog value (0 to 1023)
 Serial.println(potValue);           // Print the value to the Serial Monitor
 delay(200);                         // Wait a bit between readings
}
```

Tools > Serial Monitor

Modifications

- Change the analog pin
- Analog pins return 0 … 1023. Use **map** to rescale to a different range
- Use this value to control the LED blink time

Variable rate blinking

5-Blink-variable-rate

```
// Pin definitions
const int ledPin = 12;     // LED connected to digital pin 12
const int potPin = A0;     // Potentiometer connected to analog pin A0

void setup() {
 pinMode(ledPin, OUTPUT);  // Set LED pin as an output
}

void loop() {
 int potValue = analogRead(potPin);          // Read the potentiometer (0 to 1023)
 int delayTime = map(potValue, 0, 1023, 100, 1000); // Map to 100-1000 ms delay

 digitalWrite(ledPin, HIGH);   // Turn the LED on
 delay(delayTime);             // Wait
 digitalWrite(ledPin, LOW);    // Turn the LED off
 delay(delayTime);             // Wait again
}
```

Outline

What is Arduino and what is it for?

The Arduino IDE, uploading a basic sketch

Breadboards

LED

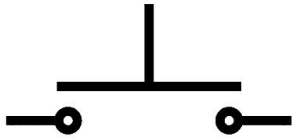
Controlling the rate

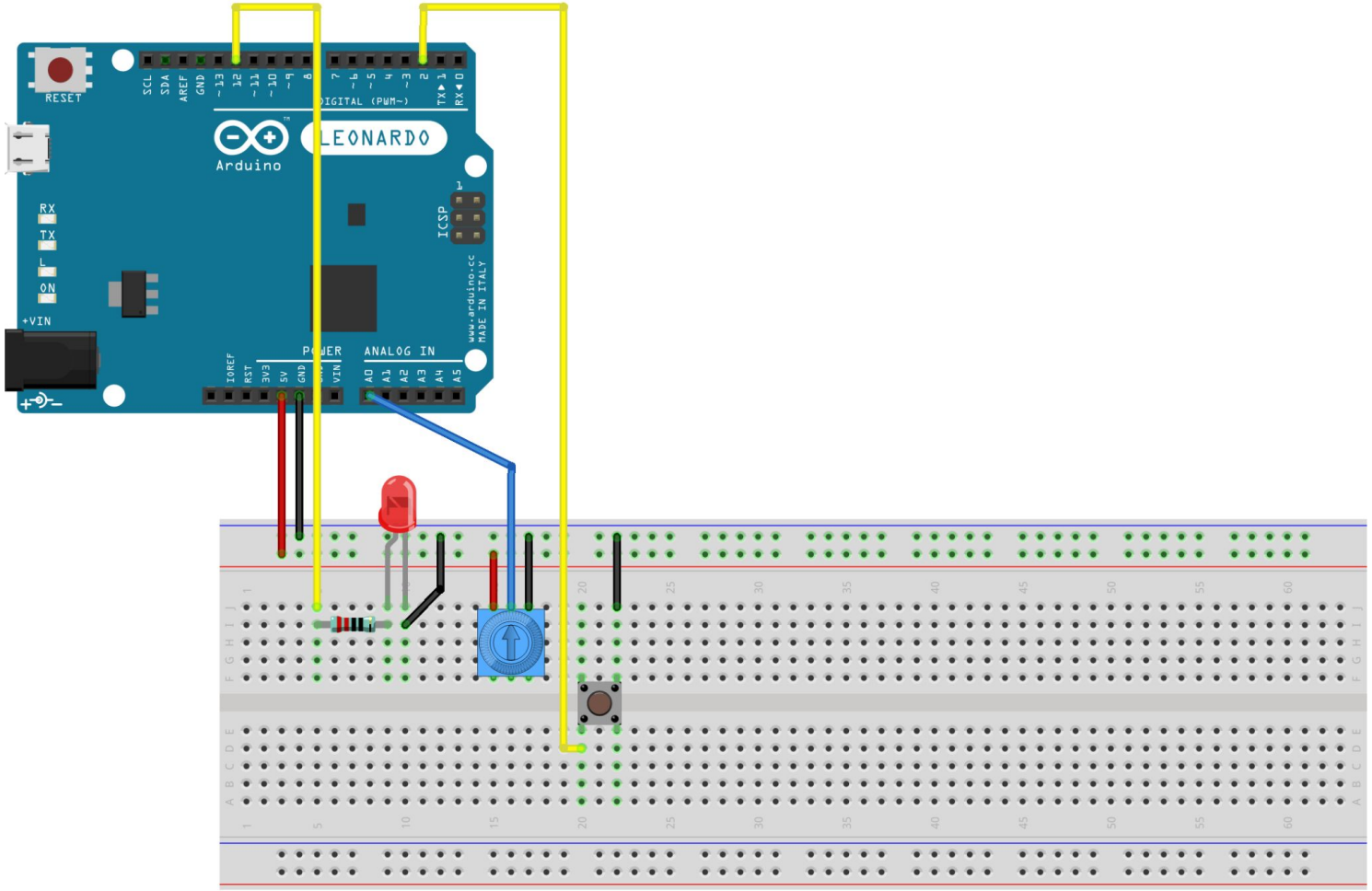
Switching on and off

RGB LED

Tactile switch (momentary)

pairs of pins already connected - use opposite corners if unsure




RESET
Arduino
LEONARDO
RX
TX
ON
+VIN
ICSP
POWER
ANALOG IN
www.arduino.cc
MADE IN ITALY
fritzing

Toggle blinking state using interrupt

6-Blink-variable-rate-switch

```
const int ledPin = 12;
const int potPin = A0;
const int buttonPin = 2; // Must be interrupt-capable pin (2 or 3 on Uno)

volatile bool blinking = true;  // Must be volatile because it's modified in ISR

void setup() {
 pinMode(ledPin, OUTPUT);
 pinMode(buttonPin, INPUT_PULLUP);  // Use internal pull-up

 // Attach interrupt to the button pin (falling edge = button press)
 attachInterrupt(digitalPinToInterrupt(buttonPin), toggleBlinking, FALLING);

 Serial.begin(9600);
}

void loop() {
 if (blinking) {
   int potValue = analogRead(potPin);
   int delayTime = map(potValue, 0, 1023, 100, 1000);

   digitalWrite(ledPin, HIGH);
   delay(delayTime);
   digitalWrite(ledPin, LOW);
   delay(delayTime);
 }
}

void toggleBlinking() {
 blinking = !blinking;
}
```

ISRs must be short and simple:

- They hold up the main loop
- Timers paused
- Interrupts disabled - no delay, no serial

Best practice: set a variable then act on it in loop()

Modifications

- Use **else** to illuminate the onboard LED when not blinking

Outline

What is Arduino and what is it for?

The Arduino IDE, uploading a basic sketch

Breadboards

LED

Controlling the rate

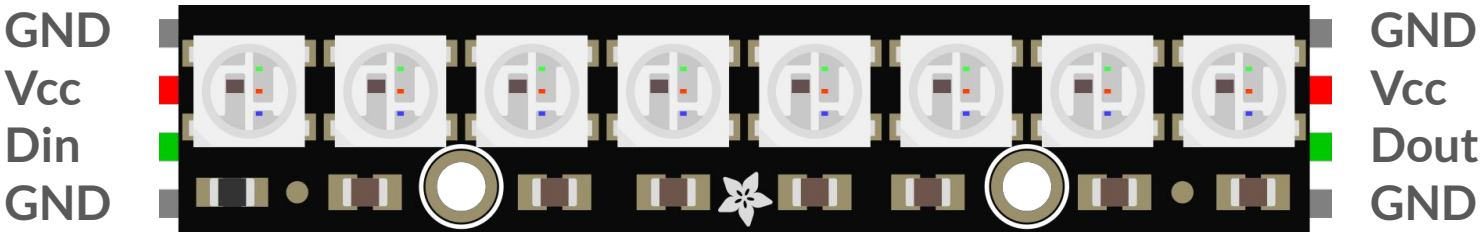
Switching on and off

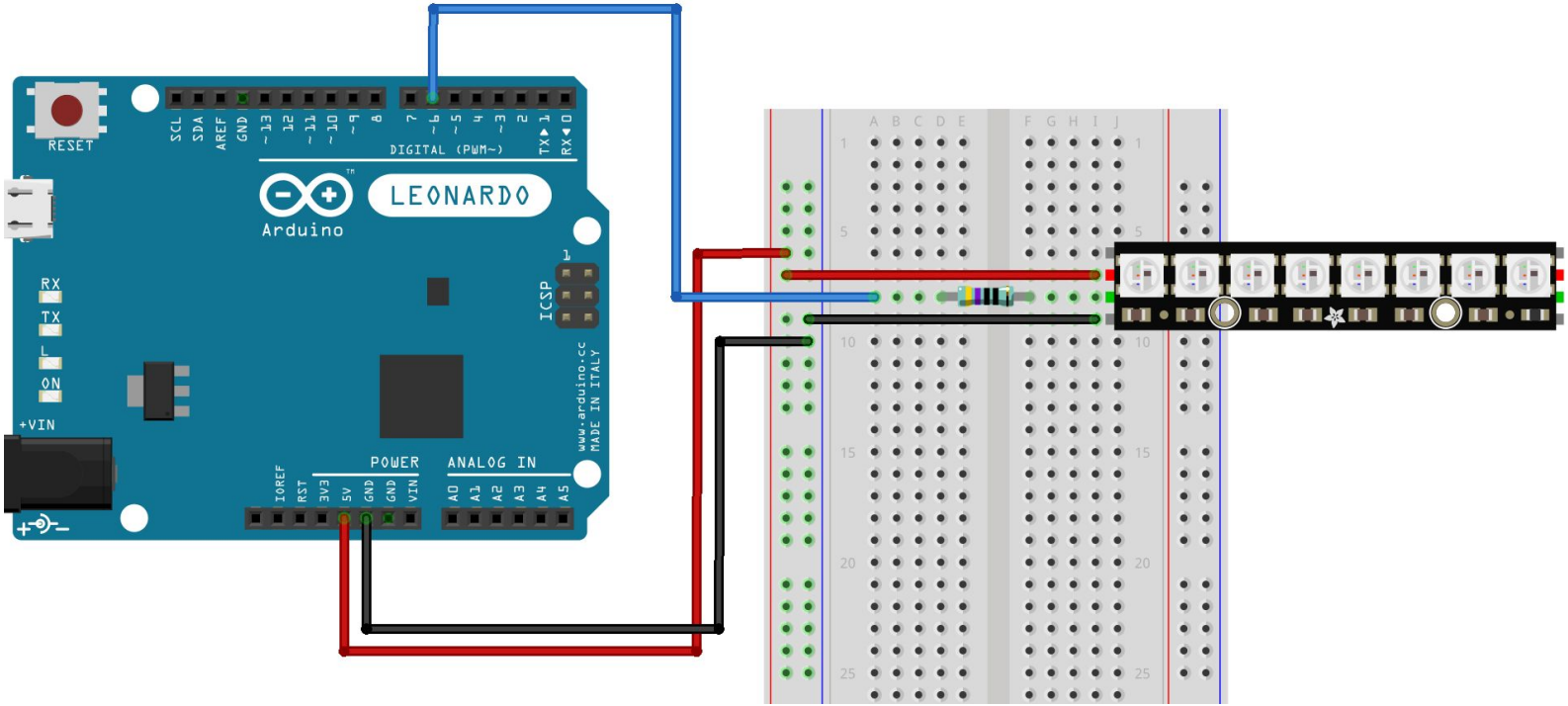
RGB LED

NeoPixel Strip

- NeoPixels use the **WS2812B** protocol, combining red, green, and blue LEDs with a controller chip in a single package
- Communication is over a single data line, where each LED receives data and passes the rest down the chain to the next LED
- Strips can be “daisy-chained” connecting **Dout** to **Din**

Arduino NeoPixel Library code and documentation:
https://github.com/adafruit/Adafruit_NeoPixel




RESET
SCL
SDA
AREF
GND
~13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
LEONARDO
Arduino
RX
TX
L
ON
+VIN
ICSP
www.arduino.cc
MADE IN ITALY
POWER
ANALOG IN
IOREF
RST
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
A B C D E
F G H I J
1
5
10
15
20
25

Install NeoPixel Library

Sketch → Include Library → Manage Libraries…

Search for "**Adafruit NeoPixel**"

Install

File > Examples > Adafruit NeoPixel > **simple**

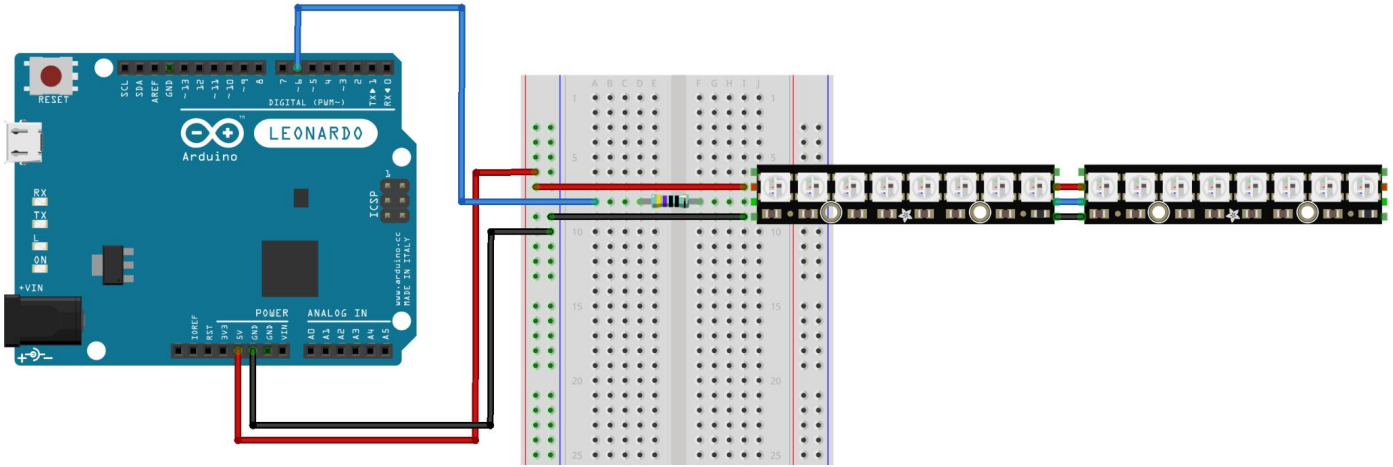
File > Examples > Adafruit NeoPixel > **RGBWstrandtest**

File > Examples > Adafruit NeoPixel > **buttoncycler** (needs switch on pin 2)

For all examples, change number of pixels to 8

Modifications

- Modify example code to change colours
- Change the speed of the effects
- See if you can write your own effect
- Work with your neighbours to daisy chain two or more NeoPixel sticks (remember to update the code to let Arduino know how many LEDs you have in the chain)



vibe code…

Use ChatGPT or your favourite AI to develop an effect for NeoPixel

“make a contemplative effect for an 8 element arduino neopixel strip that is reminiscent of being underwater”

Next Week...

Outline

What is MIDI and what is it for?

MIDI messages

Arduino USB MIDI library

A basic MIDI sequencer

Controlling tempo

Basic MIDI “keyboard”

Adding pitch bend

Prerequisites

Week 1 [Tues, May 13th]

- Laptop with Arduino IDE installed
- Optional: fritzing
- USB-A to USB-C converter or hub if needed
- Arduino starter kit (provided)

Week 2 [Tues, May 20th]

- Laptop with Ableton Live and MIDIView installed
- Alternatively Garageband, Tracktion or whatever DAW
- Headphones
- Arduino starter kit

Week 3 [Tues, May 27th]

- Laptop with VCV Rack Installed
- Headphones

Week 4 [Tues, June 3rd]

- Laptop with Arduino IDE, Ableton Live, VCVRack and MIDIView installed
- Headphones
- Arduino starter kit