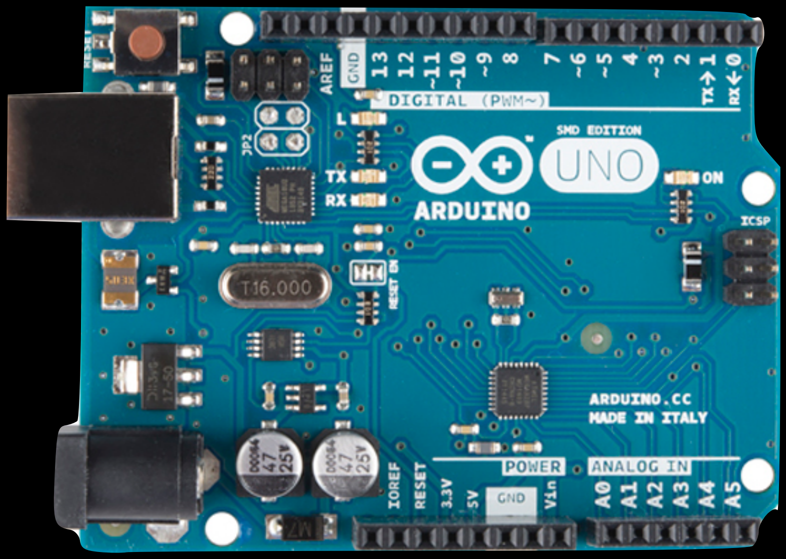
Prototyping with Arduino

Celine.Coutrix@imag.fr

Aim

- Learn how to prototype interaction techniques with Arduino

How?



- Arduino
    - open-source electronics prototyping platform (http://www.arduino.cc/)

How?

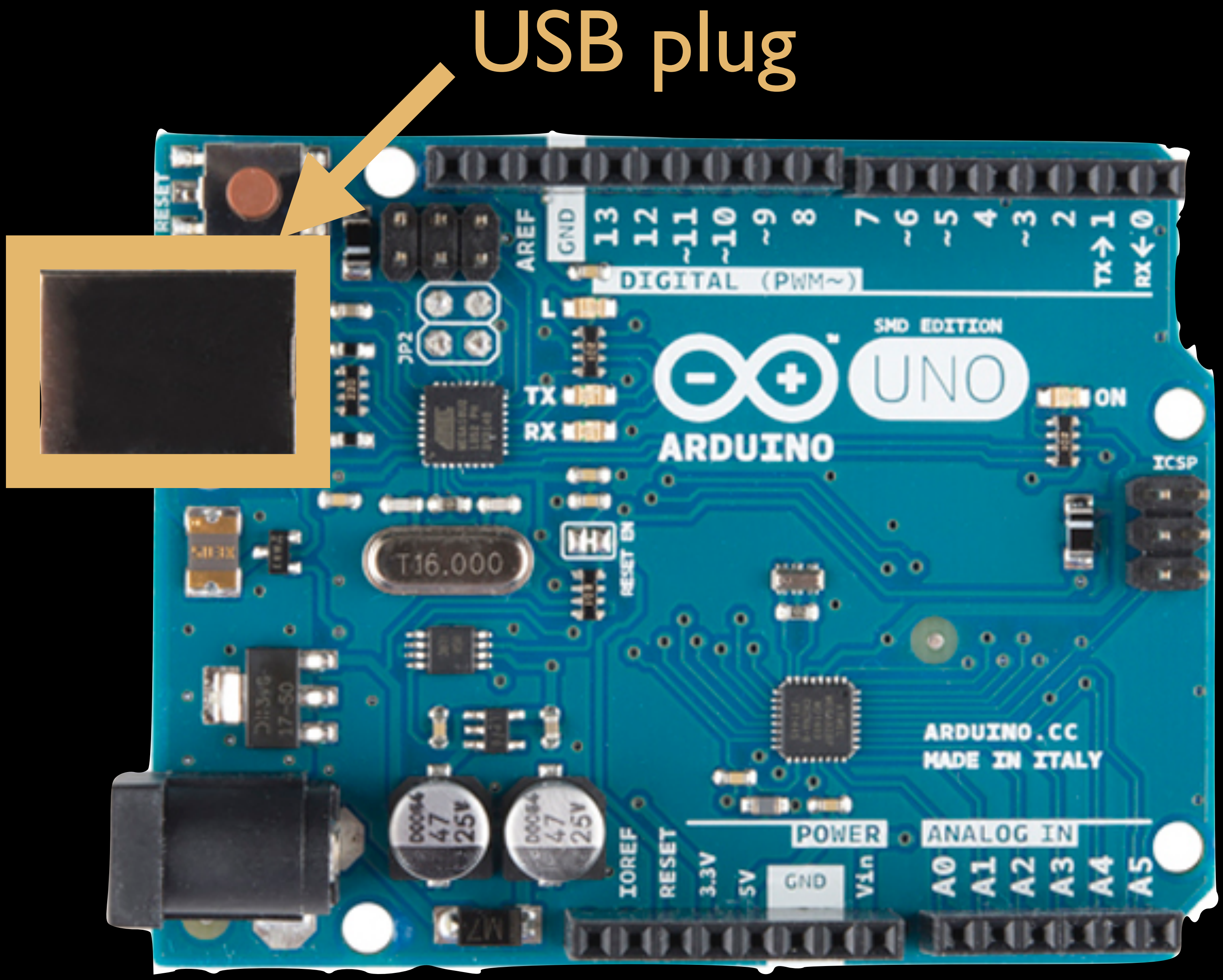
- + Sensors, effectors and other components like resistors

Arduino Principle

- Make your simple electronic prototype
- Program on computer
- Upload program to the Arduino board
- Run on the Arduino board

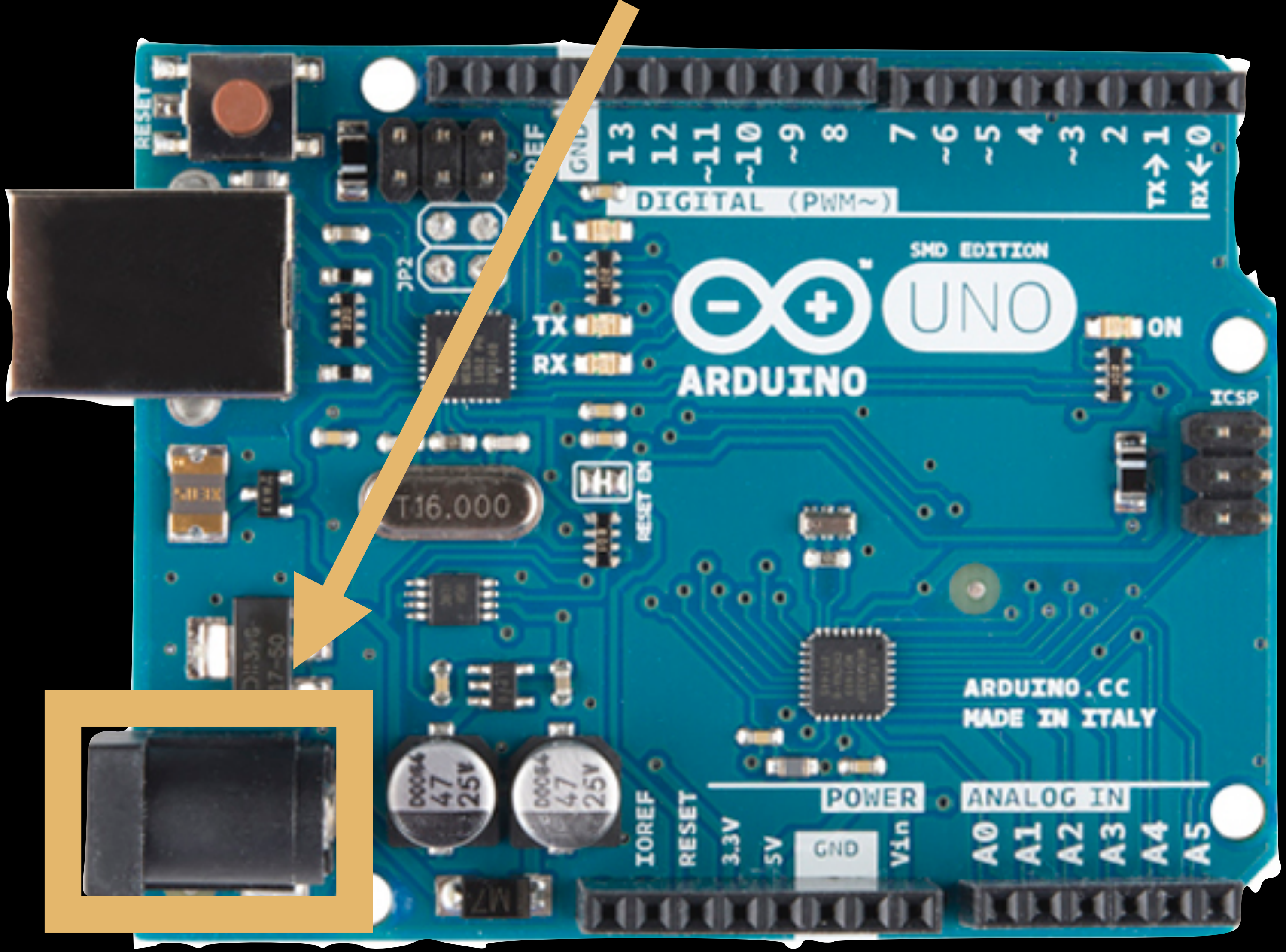
→ you can disconnect the Arduino board from the computer, if plugged to power

Arduino Board

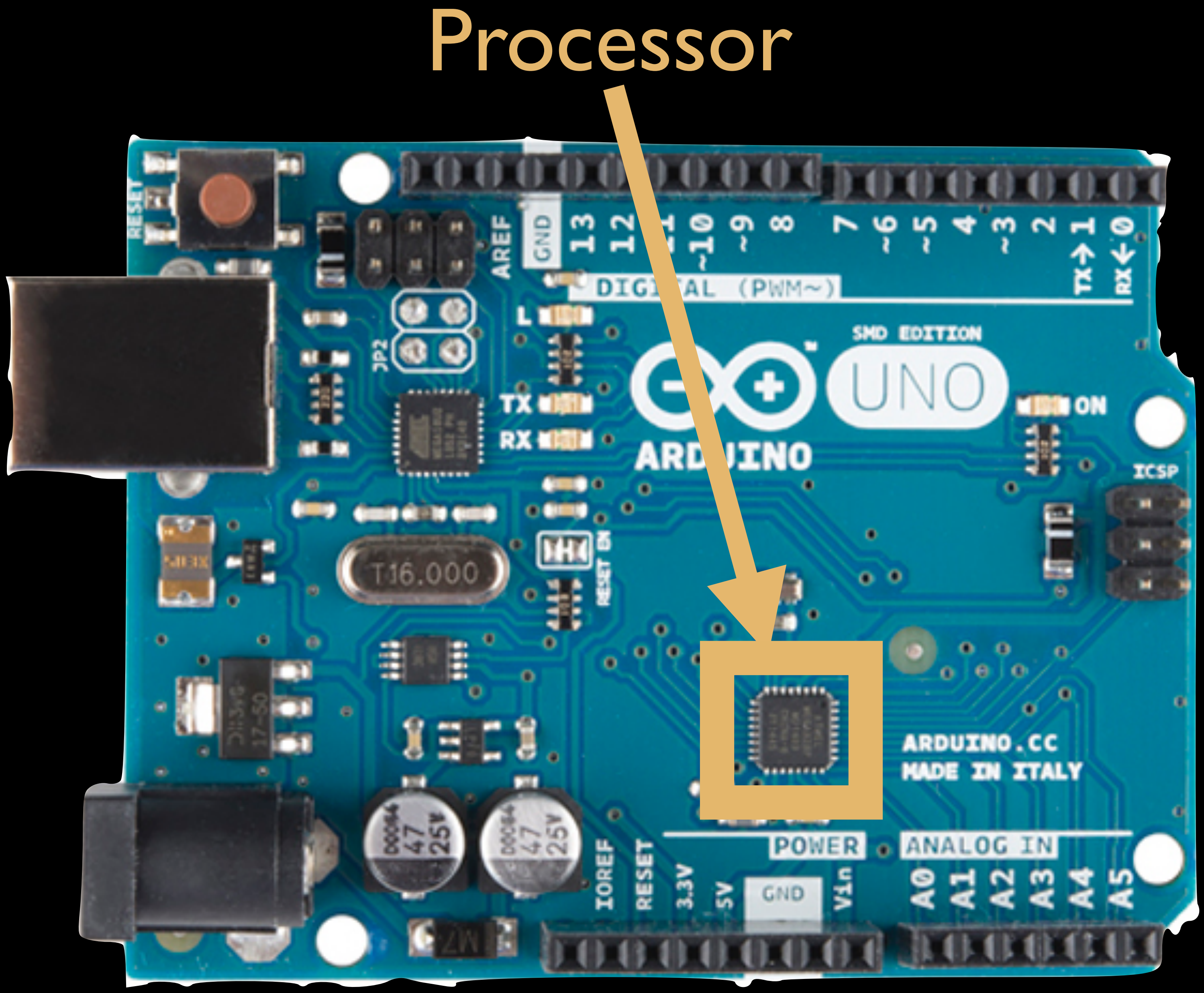


Arduino Board

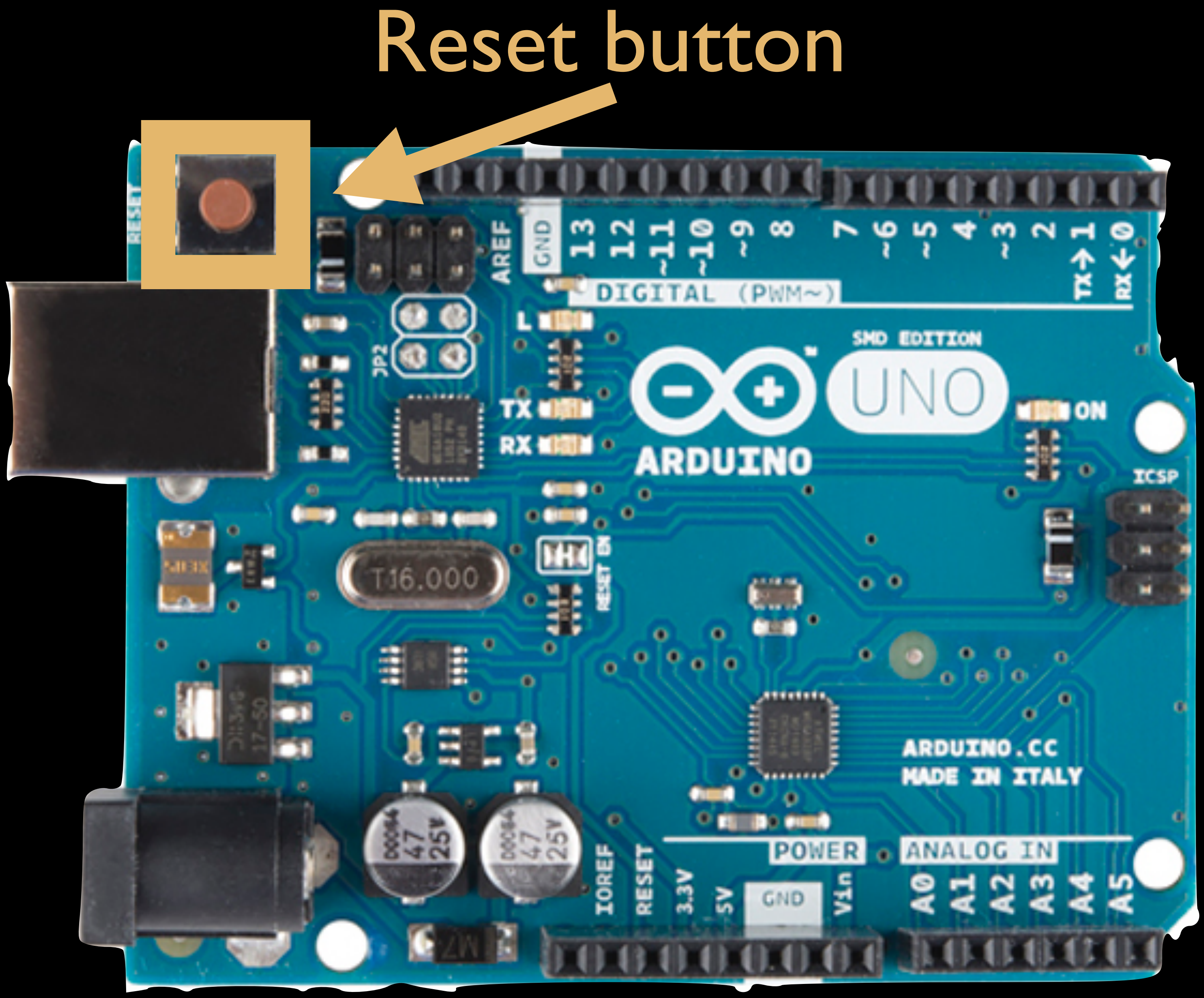
External power
(use if not plugged in USB)



Arduino Board



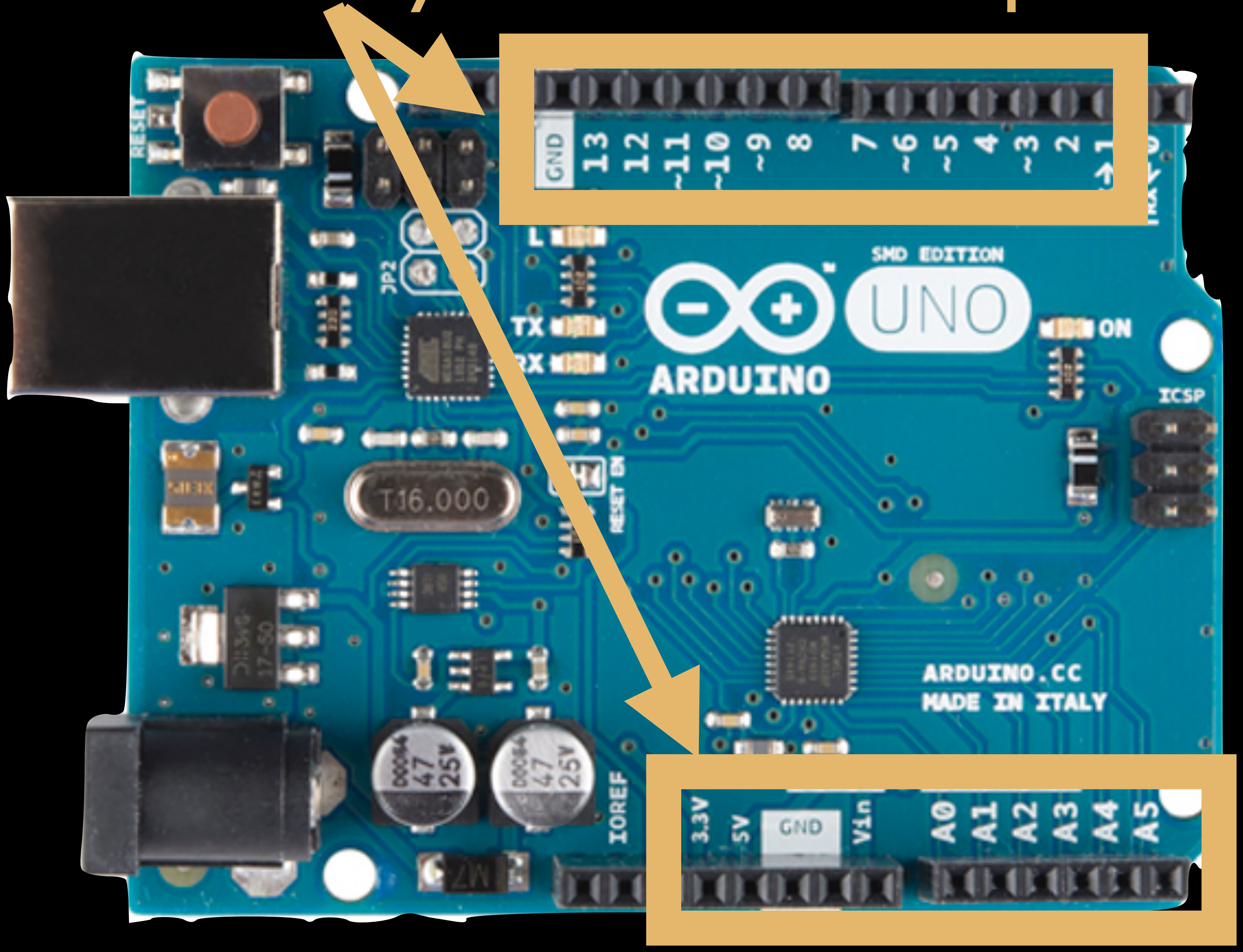
Arduino Board



Arduino Board

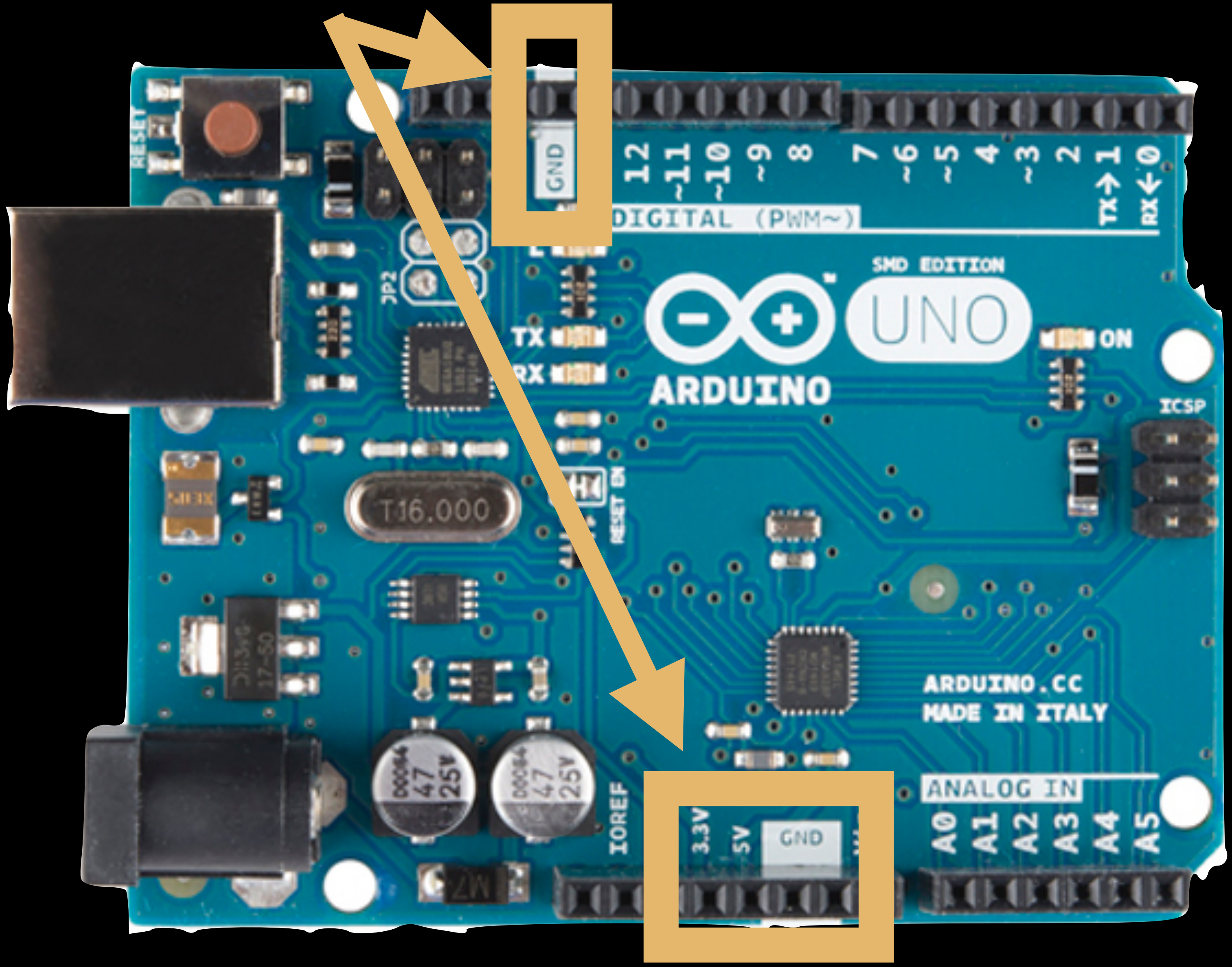
Pins

(to connect to your electronic prototype)



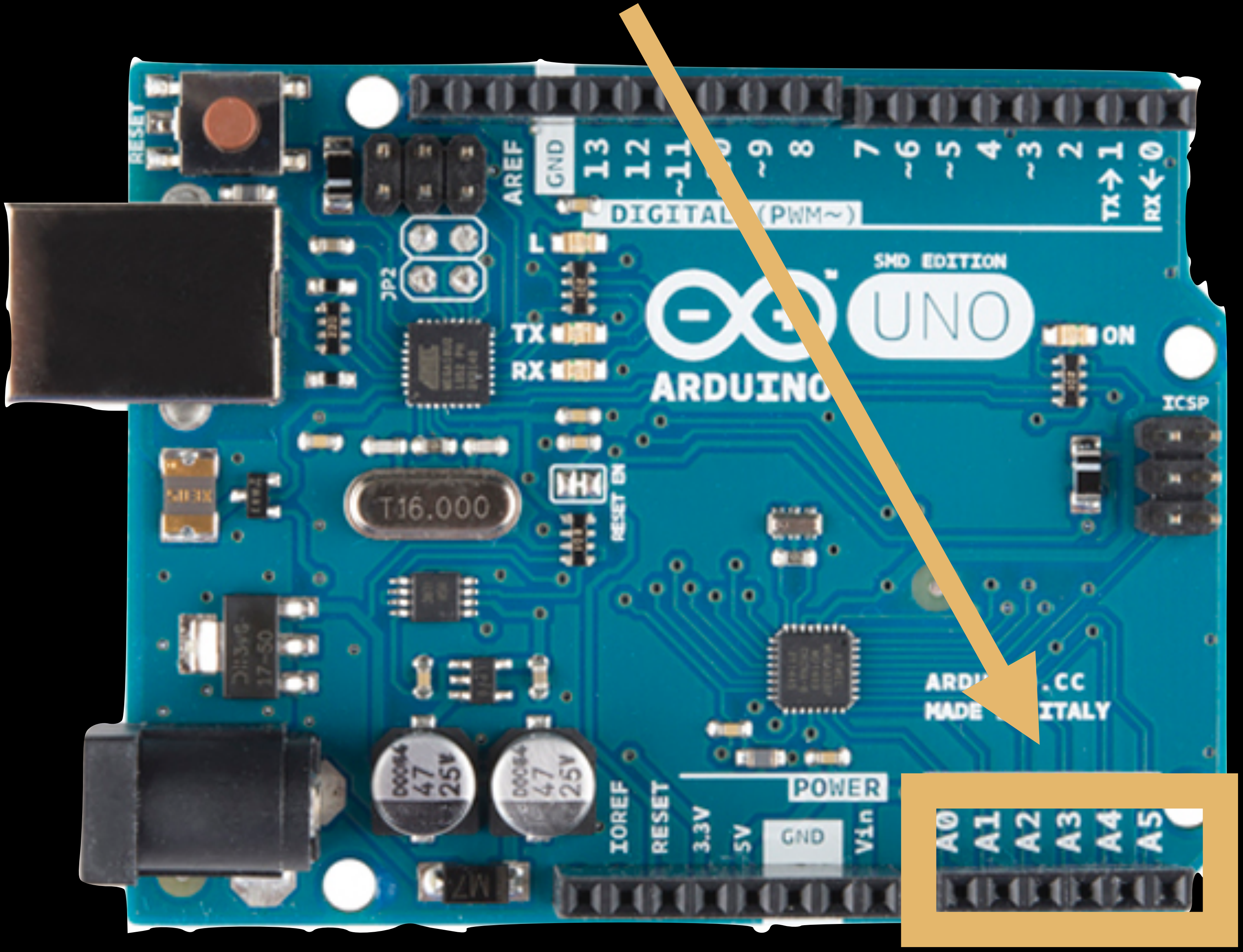
Arduino Board

Power and Ground pins



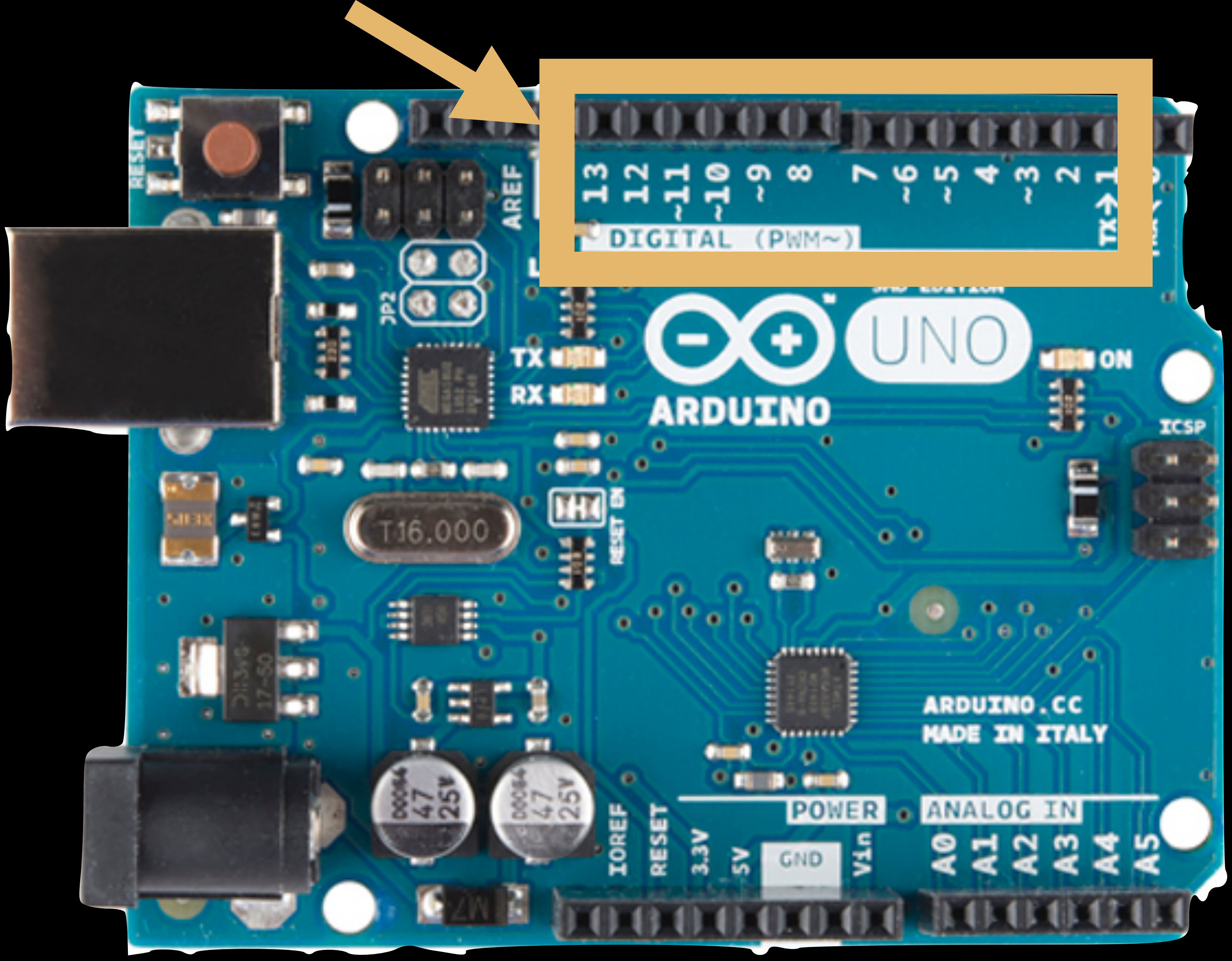
Arduino Board

Analog (=continuous) input pins



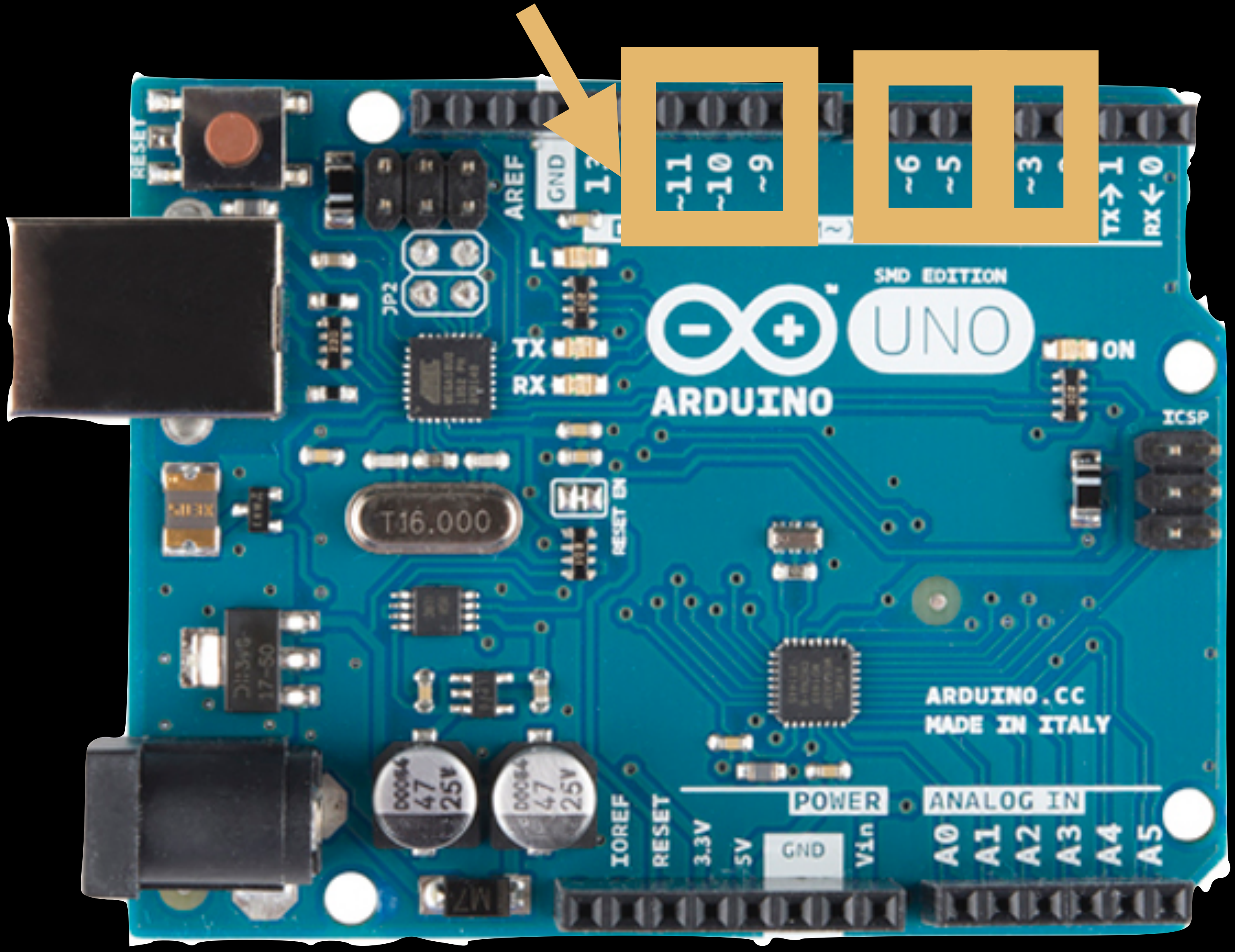
Arduino Board

Digital (=0 or 1) input/output pins



Arduino Board

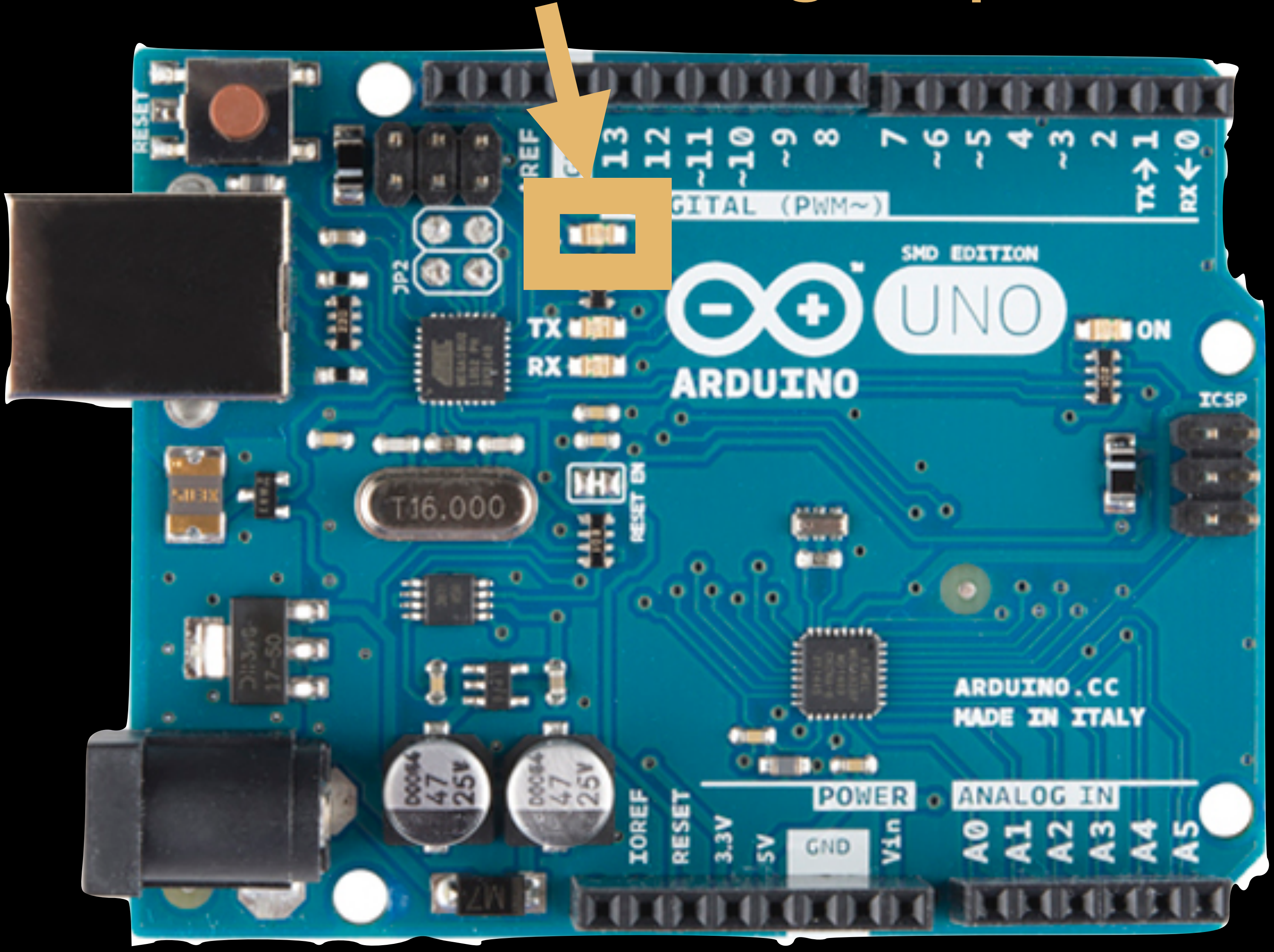
PWM (≈ continuous) output enabled digital pins



Arduino Board

Built-in LED

(⇔ LED connected between digital pin 13 and ground)



Arduino First Use

- Download the Arduino Environment:

  in particular for Linux users — not from the software center, but rather from the webpage

  http://arduino.cc/en/Guide/HomePage

- Install the Arduino Environment
- Launch the Arduino Environment

  Linux users might need to do ‘sudo ./<name of arduino app>’ in the application folder

- Connect the Arduino board to the USB

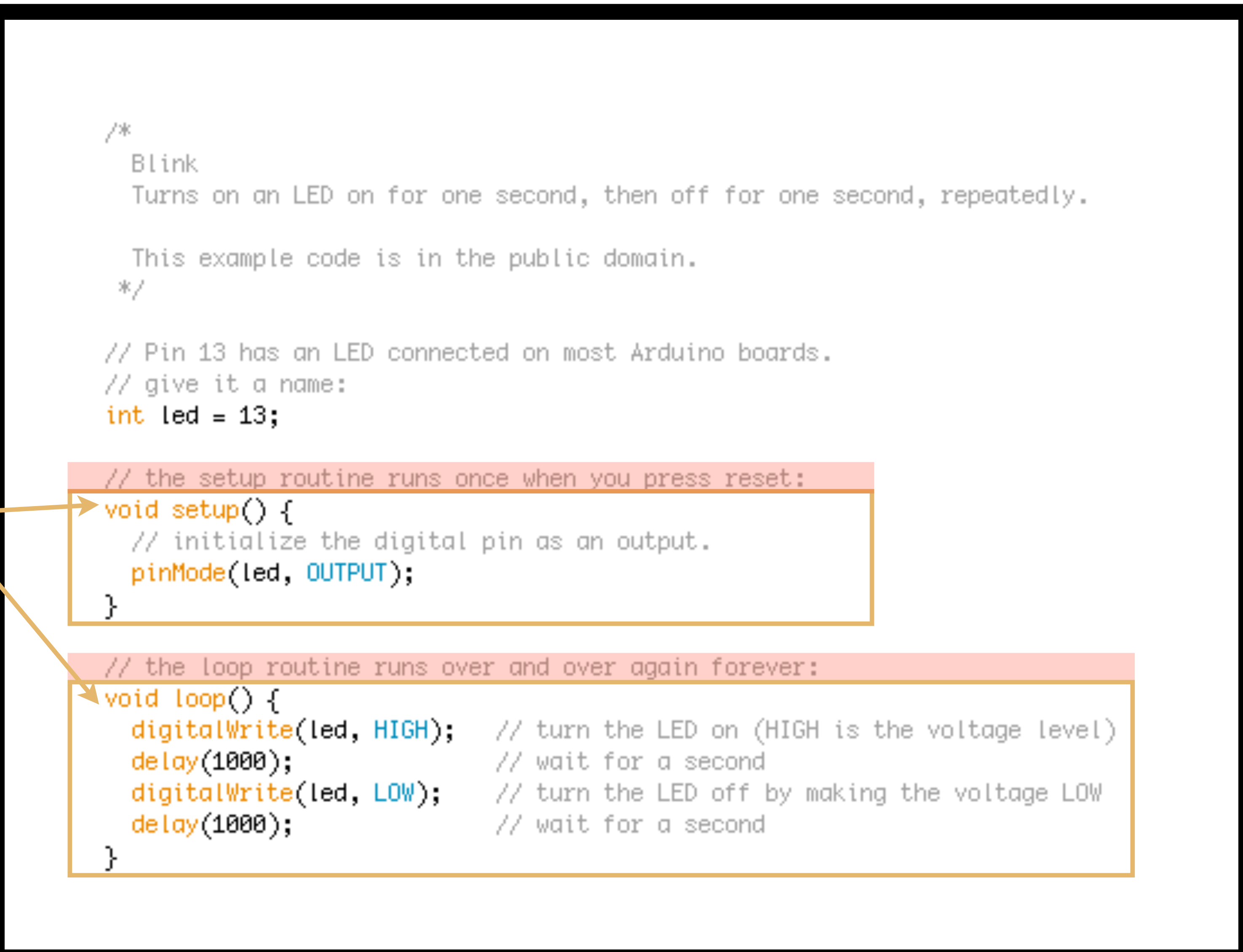
Arduino First Use

- Select your board: Tools>Board> Uno
- Select your serial port: Tools>SerialPort> /dev/tty.usbserial-XXXXXXXX or /dev/tty.usbmodemXXXXXX

For linux users: sudo chmod a+rw <serial port>
with arduino plugged in and from the application folder

Arduino First Use

- Open the *blink* example



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

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

Structure of an arduino program

Like other languages, you can (+should) write your own custom functions in addition

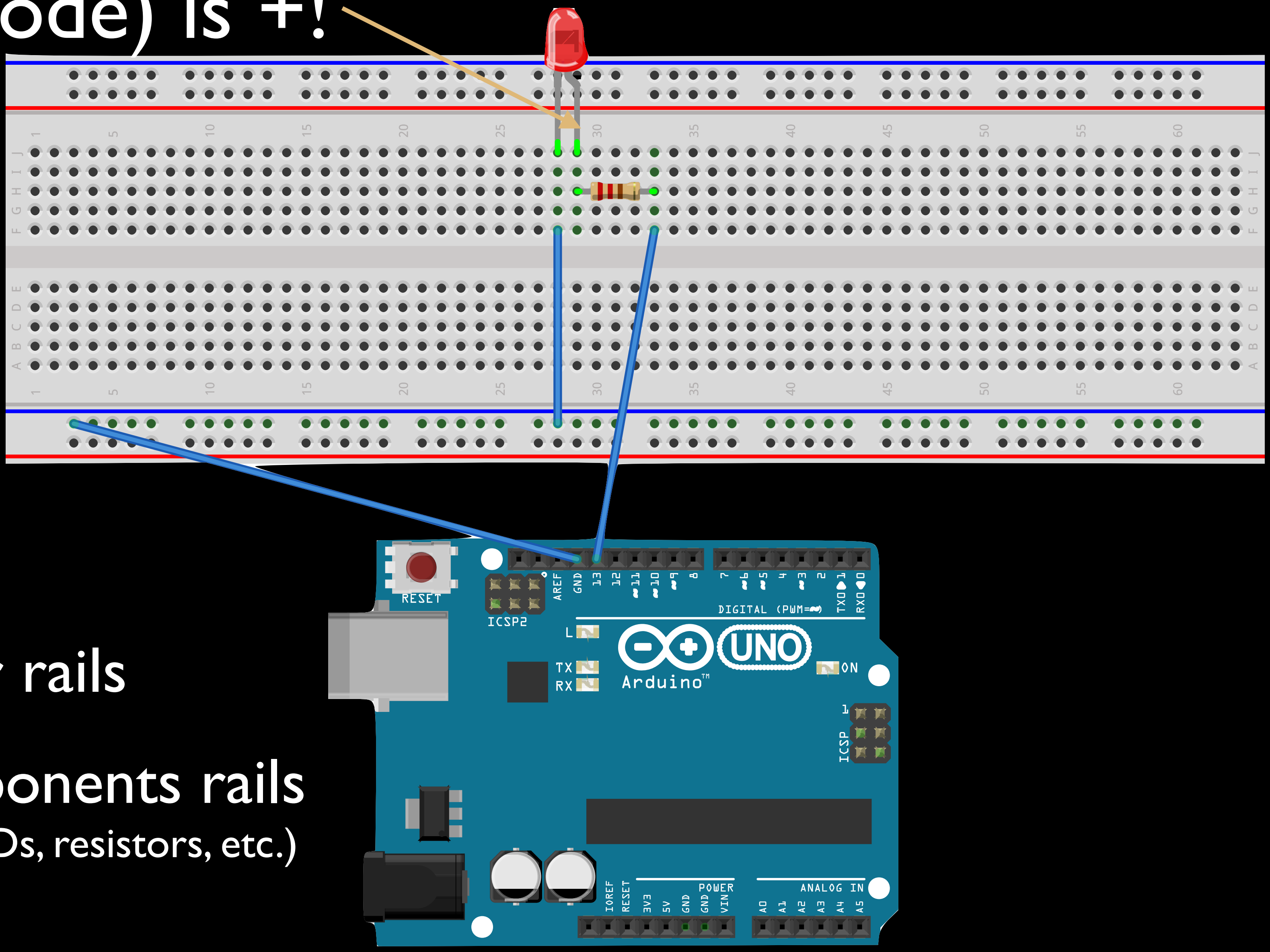
Arduino First Use

- setup() executed
  - Once, first
  - When reset button is pressed, too
- loop() executed
  - In loop after setup()

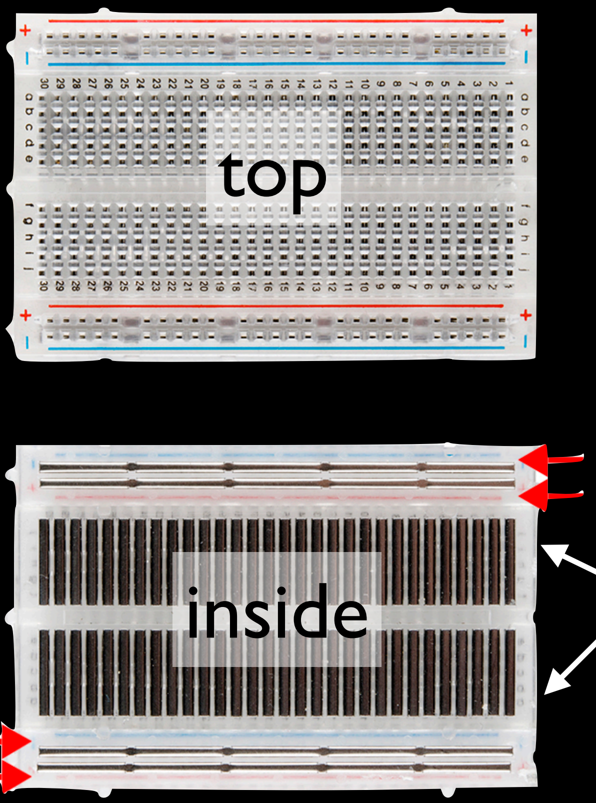
Arduino First Use

- Make the electronic blink prototype

Beware! Longest leg (anode) is +!

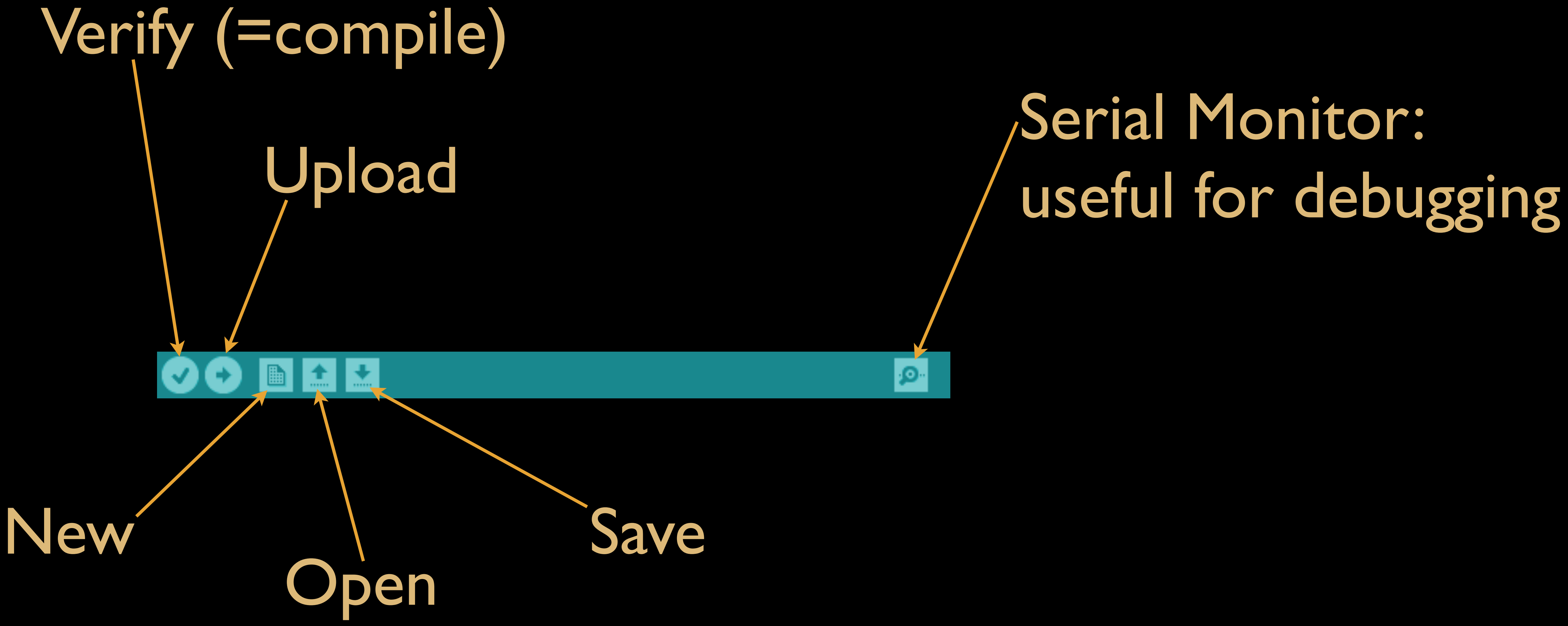


Breadboard principle:



Arduino First Use

- Upload the program and watch!



Arduino Language

- Language reference: http://arduino.cc/en/Reference/HomePage
- Based on C

➡ Now tour of useful & Arduino-specific functions

Arduino Language

- To read from sensors

➡ *digitalRead(*<pinNumber>*);*

  - returns the value (*HIGH* or *LOW*)
  - read from the digital pin <pinNumber>

➡ *analogRead(*<pinNumber>*);*

  - returns the value (*0* to *1023*)
  - read from the analog pin <pinNumber>

Arduino Language

- To write on actuators

➡ *digitalWrite(*<pinNumber>, <value>*);*

- writes <value> (*HIGH* or *LOW*)
- on digital pin <pinNumber>

➡ *analogWrite(*<pinNumber>, <value>*);*

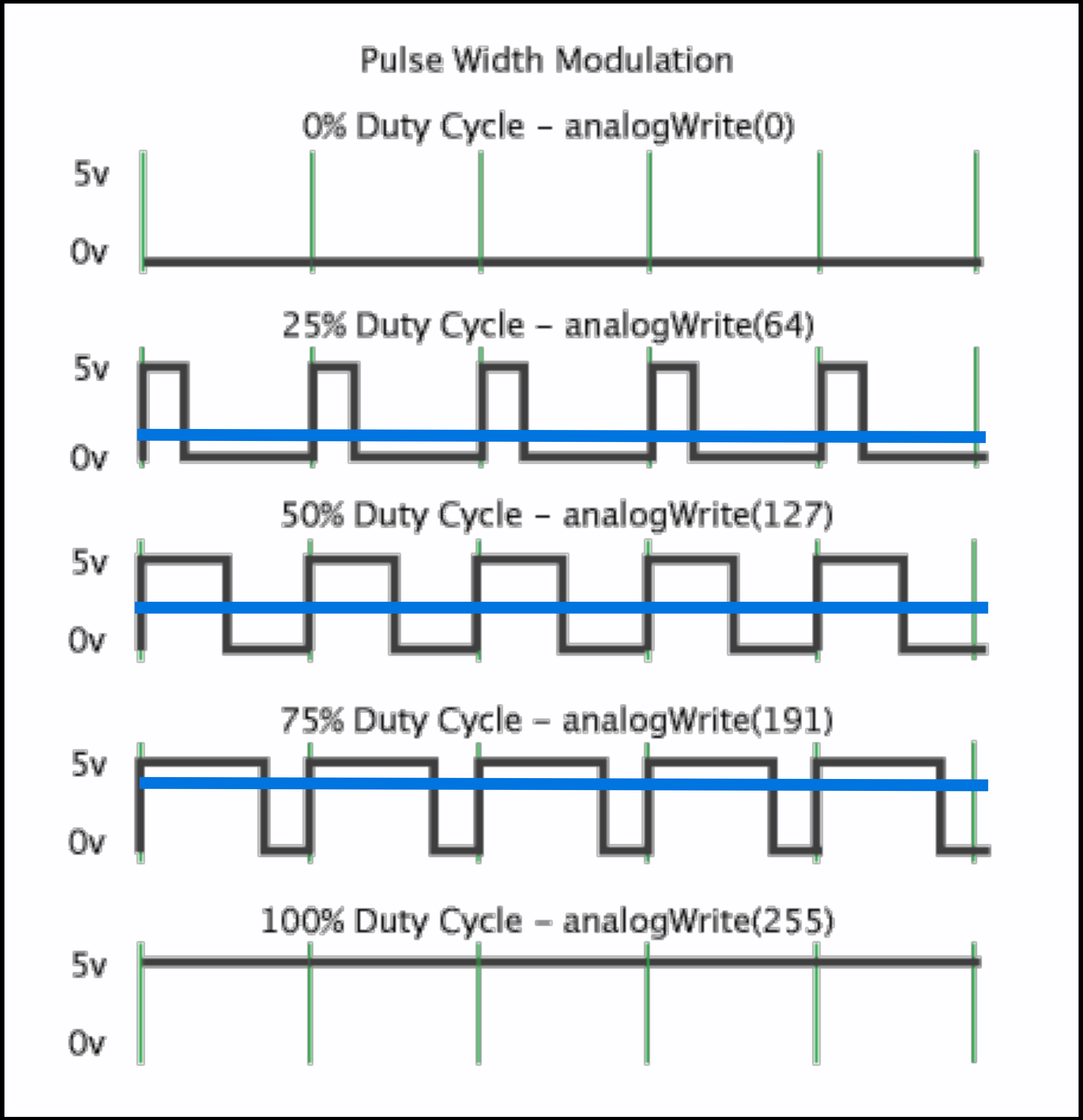
- writes <value> (*0* to *255*)
- on PWM digital pin <pinNumber>

Arduino Language

🤔 *analog*Write for PWM digital pins?!

- Writes a wave: <value>/255*100 % of the time on *HIGH* and the rest on *LOW*
- Gives illusion of continuous intensity of <value>/255*100 %

PWM = Pulse With Modulation



Arduino Language

➡ You can do the same (by hand) on non-PWM digital pins!

- Can be useful when not enough PWM pins for your needs
- Try it on the blink example!
  - Program the wave to have 25% intensity
  - Program the wave to have 75% intensity

Arduino: Get started!

- Modify the program (write functions!) in order to make the LED fade in and out, i.e. continuously light up and down
    - *Use 'analogWrite'* this time
    - Beware! Remember execution is fast for human sight!

Arduino Language

- Beware! Digital pins can be used as input or output!

➡ *pinMode(*<pinNumber>, <mode>*);*

  - Sets the pin <pinNumber> as <mode> (*INPUT* or *OUTPUT*)
  - Default is *INPUT*
  - Use, e.g., in setup() function

Arduino Language

- Useful for debugging

➡ *Serial.begin(<speed>)*

  - Sets the data rate at <speed> bits per second (= <speed> baud)
  - For communicating with the computer, use one of these rates: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200
  - Use, e.g., in setup() function

Arduino Language

- Useful for debugging

➡ *Serial.print(<value>)*

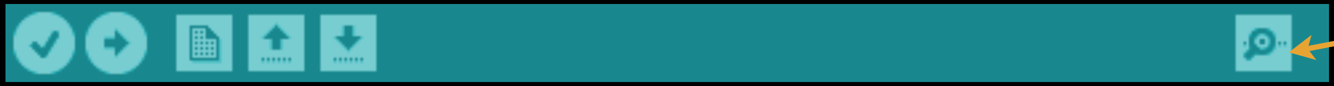
  - prints <value> (any type) to the serial port

➡ *Serial.println(<value>)*

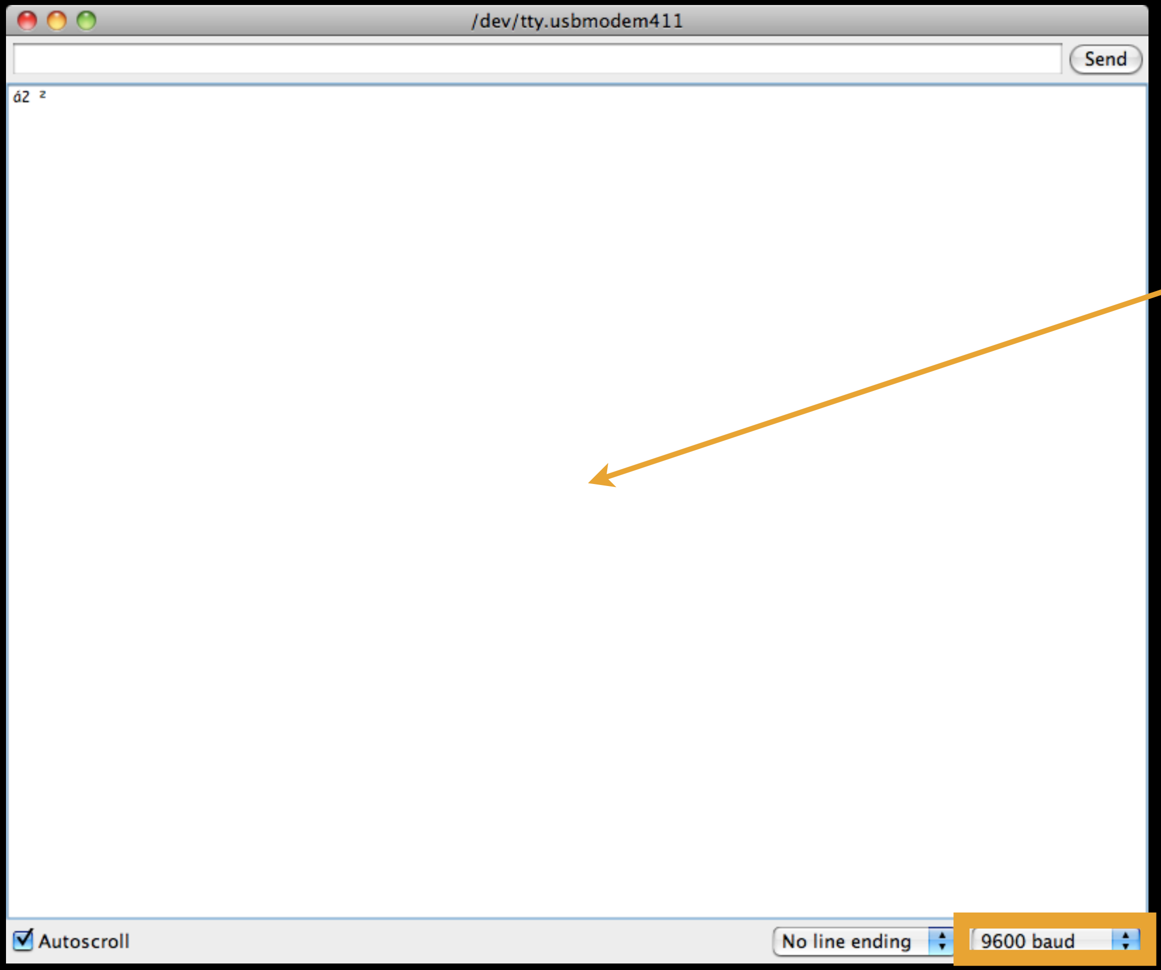
  - same + new line

Arduino Language

- To see the output of printing on Serial monitor



Display Serial Monitor Window



No line ending

9600 baud

Serial Monitor Window

Choose corresponding data rate <speed>

Arduino: Get started!

- Modify the program in order to
  - Write the intensity of the LED on the serial monitor

Arduino Language

- To map sensors output values to actuators input values

➡ *map(<value>, <fromLow>, <fromHigh>, <toLow>, <toHigh>);*

  - maps <value>
    - from one range [<fromLow>, <fromHigh>]
    - to a new range [<toLow>, <toHigh>]
  - returns the mapped value in the new range

Arduino: Get started!

- Use the function map in the program in order to
  - Write the *percentage* of LED intensity on the serial monitor

Basic Needs in Electronics

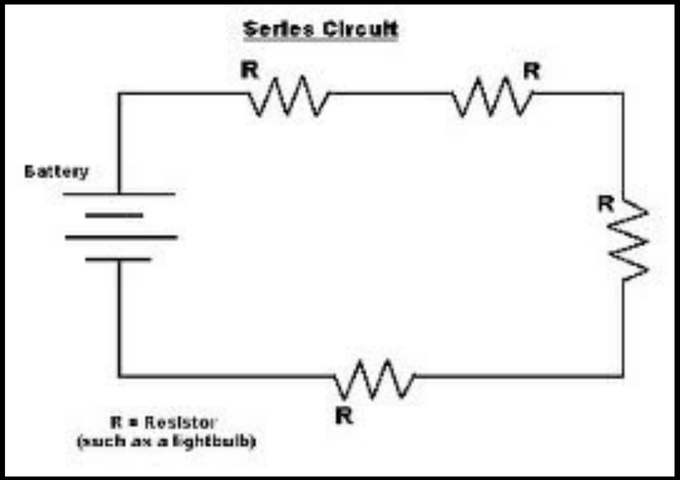
- Before you make your own circuits
  1. Arduino's voltage and current
  2. Basic laws
  3. Protect fragile components
  4. Ensure that Arduino handles reliable information

Arduino's voltage and current

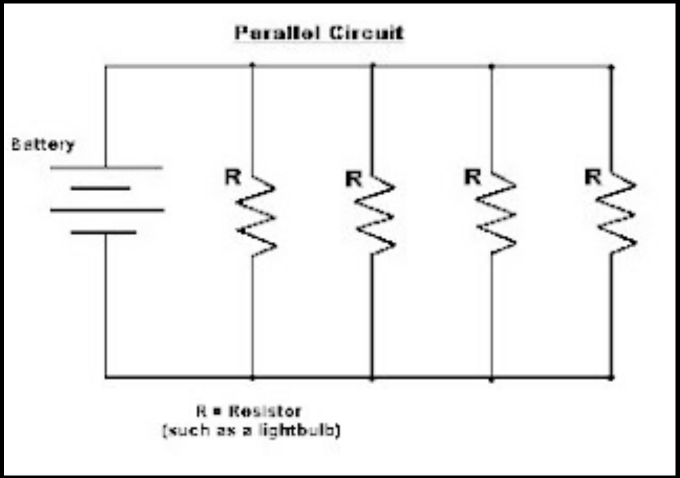
- Arduino's voltage on pins: 0V < u < 5V
  - LOW read when 0V < u < 2V and written with 0V
  - HIGH read when 3V < u < 5V and written with 5V
- Arduino's current on pins: 0A < i < 0.04A

Basic Laws

- For components connected in series
  - Voltages add up
  - Currents are the same



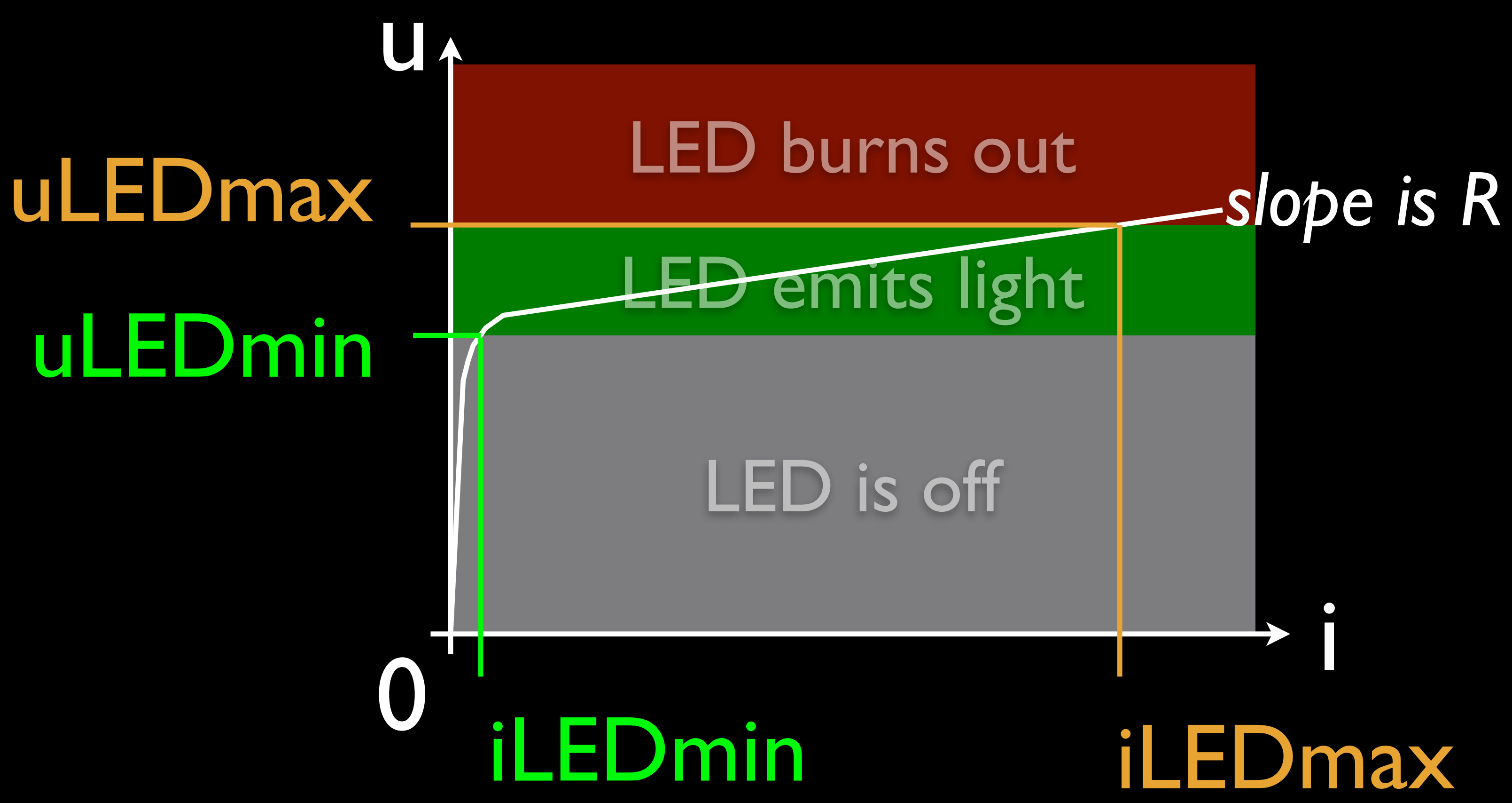
- For components connected in parallel
  - Voltages are the same
  - Currents add up



- Ohm's Law: $\underset{\text{Volt (V)}}{u} = \underset{\text{Ohm }(\Omega)}{R} \times i \longleftarrow$ Ampere (A)

Protect fragile components

- Protect fragile components, e.g. LED, from too high current



Color	**uLEDmin**	**uLEDmax**
Red	1.7	2.2
Orange	2	2.2
Yellow	2.1	2.4
Green	2	2.3
Blue	3.2	4
White	3.3	3.6

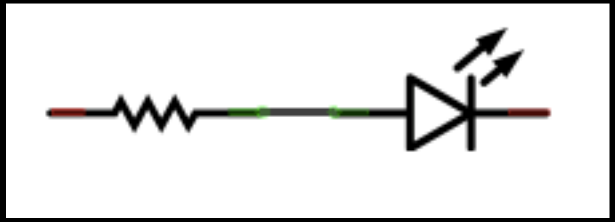
iLEDmax = 20mA

Protect fragile components

- Protect fragile components, e.g. LED, from too high current

➡ Use resistor in serie

R LED

Arduino pin (max 5V)  GND (0V)

➡ R = (5 - uLEDMax)/0.02

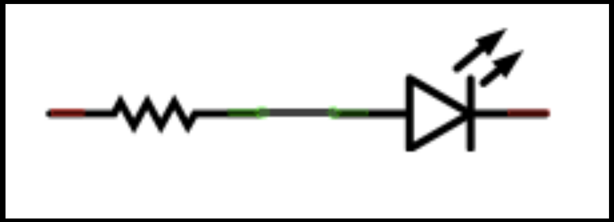
worst case total voltage

iRMax = iLEDMax = 0.02A

Protect fragile components

- Protect fragile components, e.g. LED, from too high current

Minimum resistance not to burn out the LED



LED Color	R (Ω)
Red	140
Orange	140
Yellow	130
Green	135
Blue	50
White	70

Beware! If circuit is different, make the calculation again!

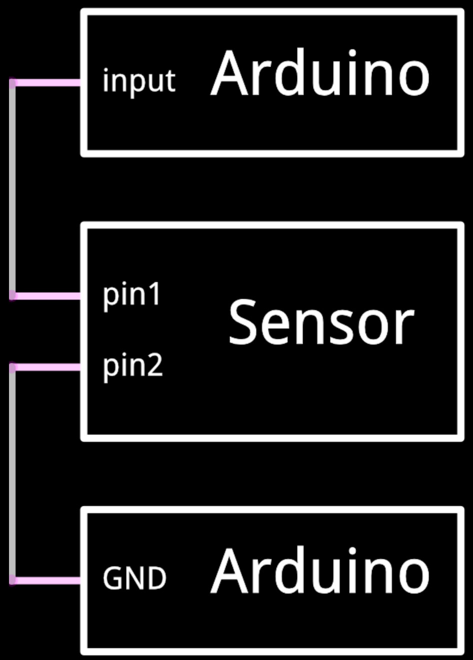
Reliability

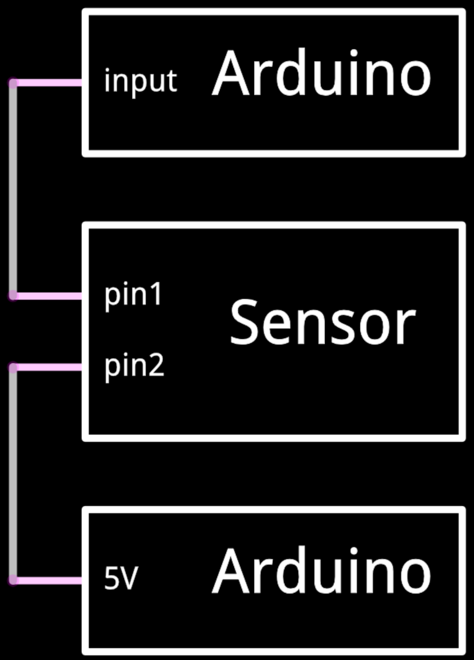
- When you use components, ensure that the Arduino pin gives reliable information
- Because, e.g., sensor is not activated

  ⇒ the pin is not connected

  ⇒ the pin is susceptible to interference

  ⇒ the pin reads random values

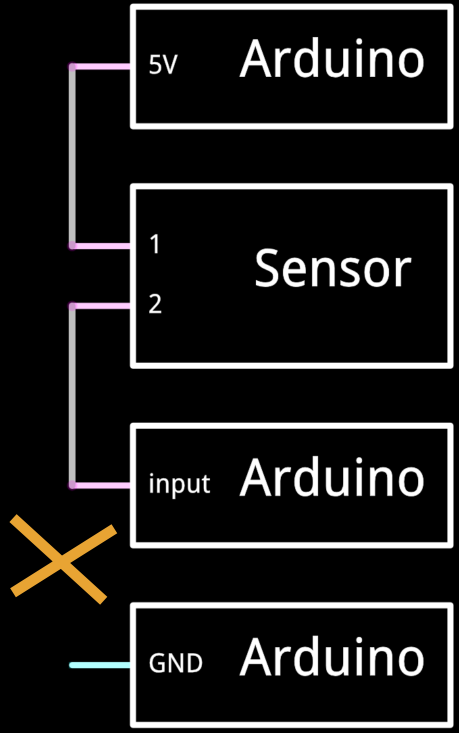




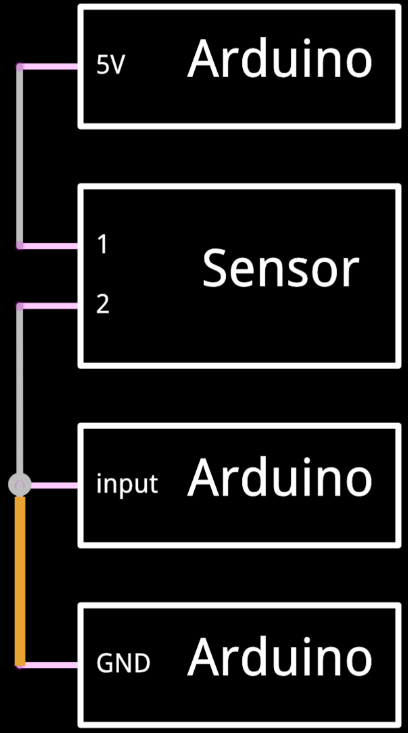
Reliability

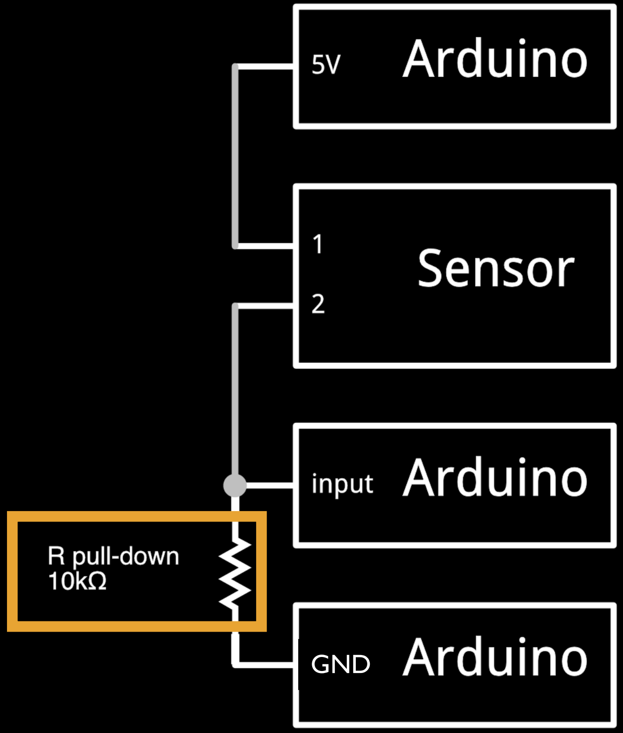
- When you use sensors, ensure that the Arduino pin gives reliable results
  - If random values appear when it should be *LOW*, use 10kΩ pull-down resistor: between pin and GND
  - Pulls the voltage of the pin down to 0V

if no connection, pin not connected when sensor not activated



if direct connection, short circuit between 5V and GND when sensor activated

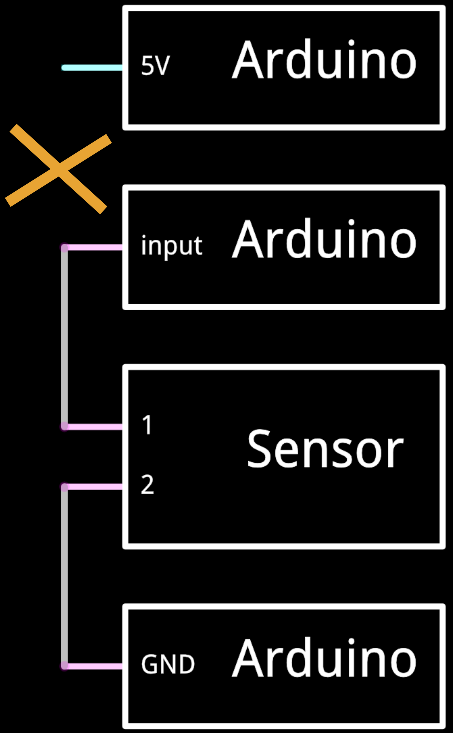




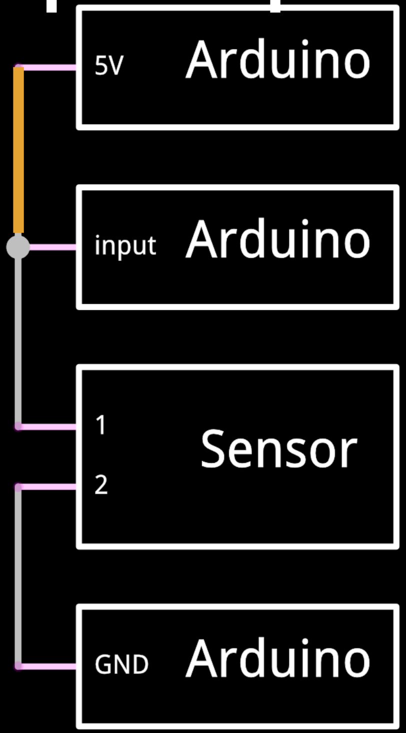
Reliability

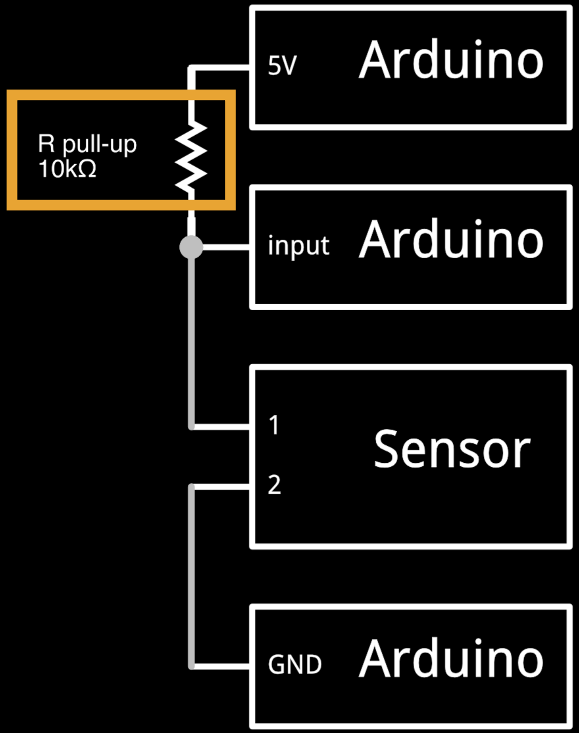
- When you use sensors, ensure that the Arduino pin gives reliable results
    - If random values appear when it should be *HIGH*, use 10kΩ pull-up resistor: between pin and 5V
        - Pulls the voltage of the pin up to 5V

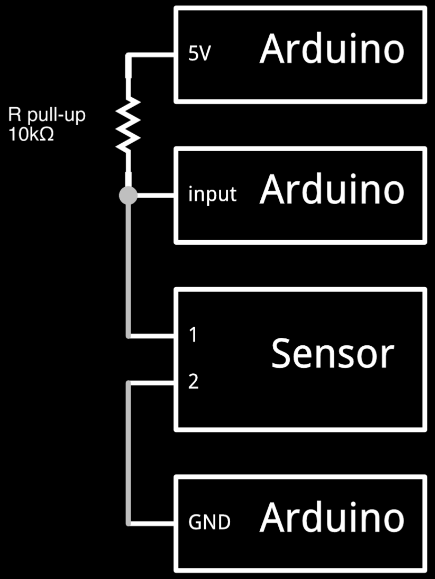
if no connection, pin not connected when sensor not activated



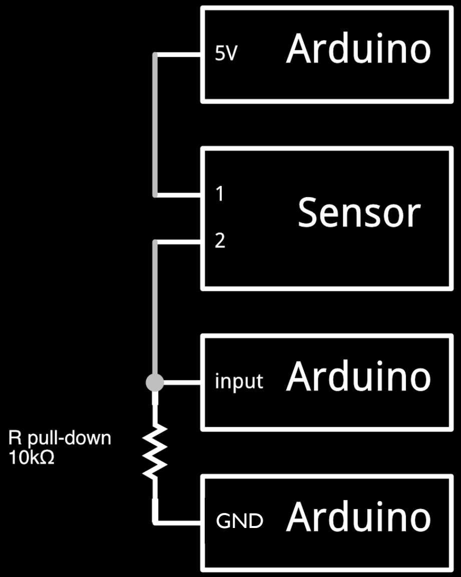
if direct connection, short circuit between 5V and GND when sensor activated







Arduino: Sensors



- Make the LED lights up proportionally to the pressure on a 2-legged sensor

1. Draw and show the circuit with resistors!
2. Wire the circuit and write the program

Arduino: Actuators

- Make a (servo)motor move from 0 to 180 degrees

  → use the servo library

Arduino: Go further!

Let’s make CatchCats with what we have
https://youtu.be/6bWu52S6uWs