Arduino hands-on session

Paolo Burgio
paolo.burgio@unimore.it

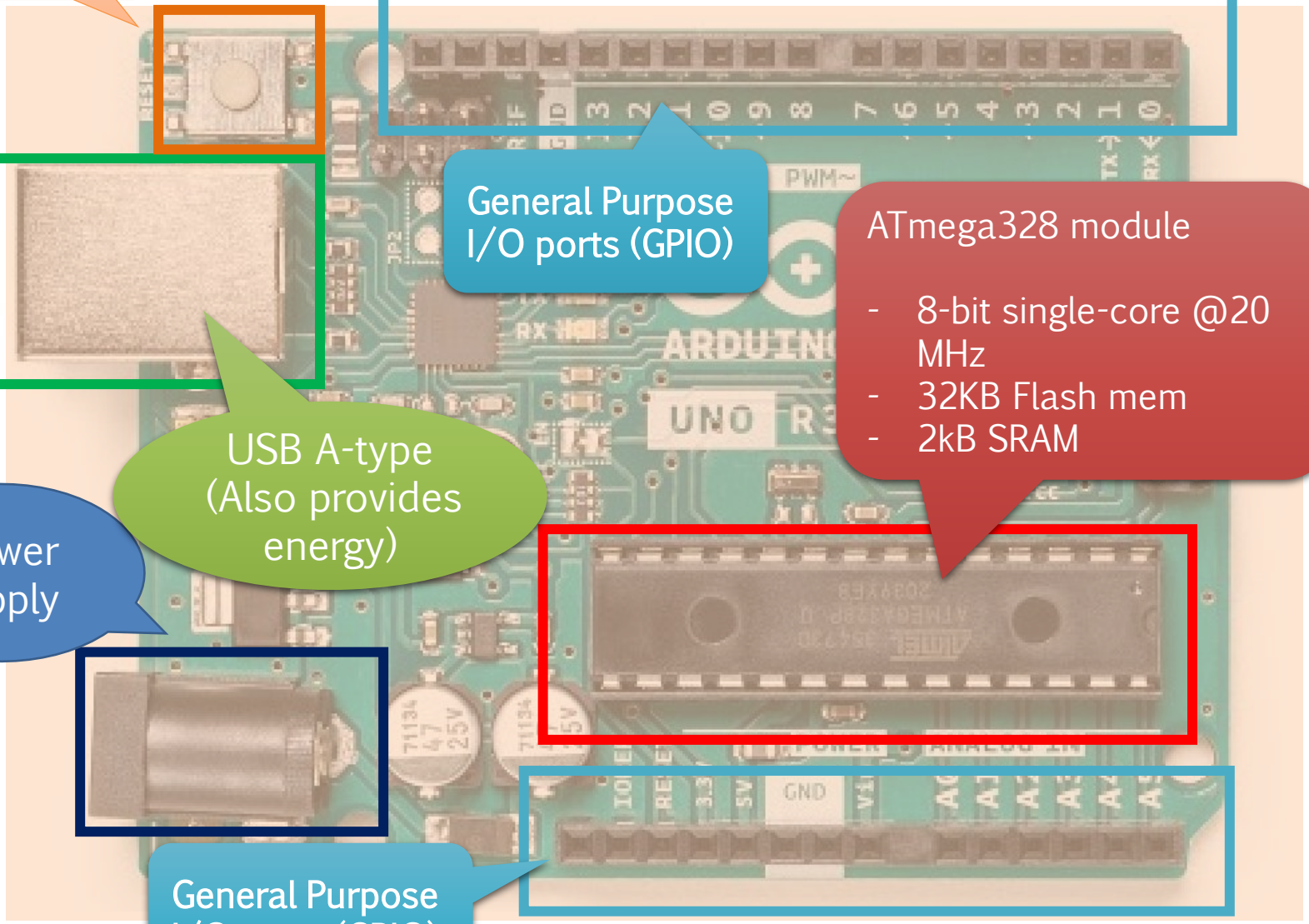
UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

High Performance
Real Time Lab

Programming is a skill best acquired by practice and example rather than from books.

ALAN TURING

Our guy (Arduino Uno R3)



Software

Micro-kernel

- No OS, need to flash al memory regions

Arduino IDE

- Integrated Development Environment, you do (nearly) everything through it
- Debug via USB (won't see this)

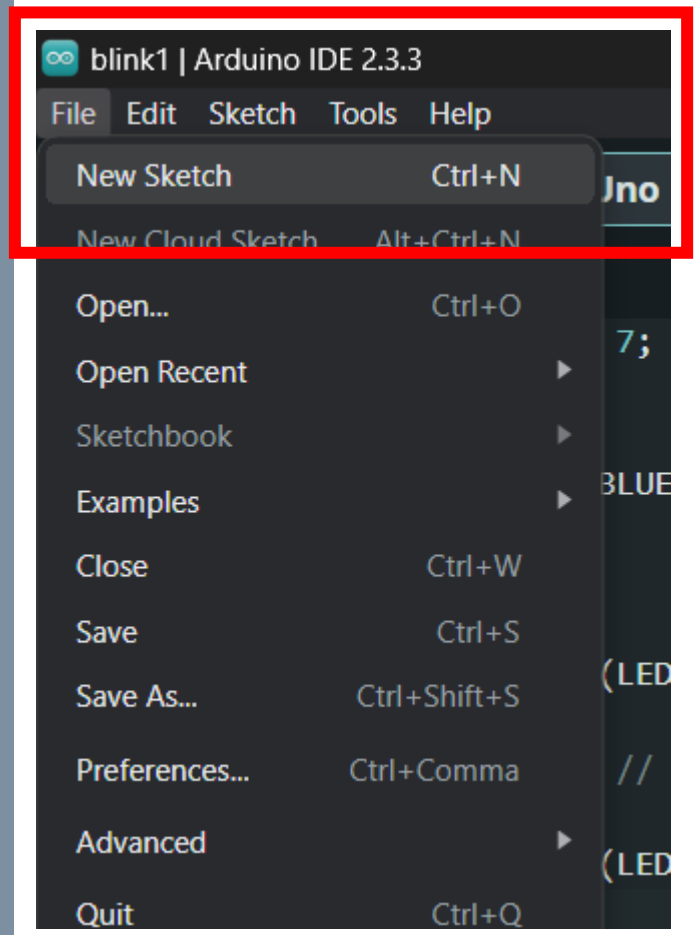
How to work

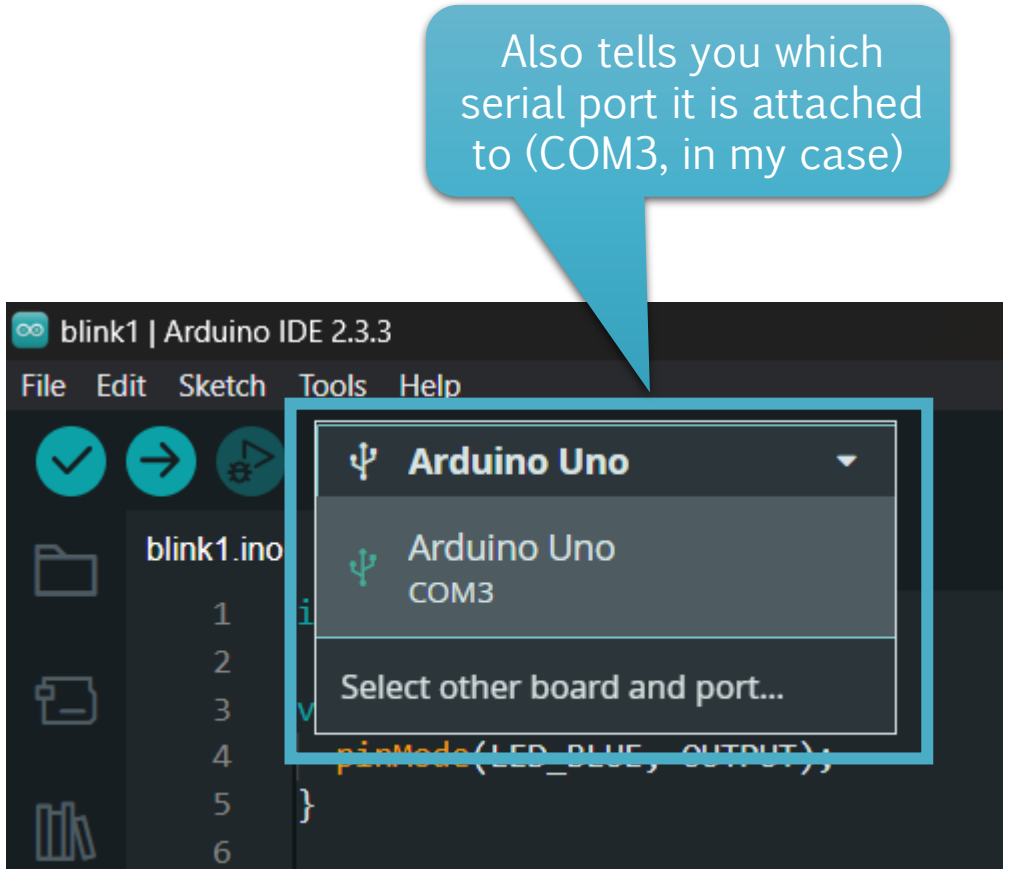
- No way is to compile our code directly on Arduino
- Cross-compilation *via* the CubeIDE
- Flash the whole OS+program via USB

A simple application

Create a new “Blink” project

- File -> New Sketch
- Select the target board



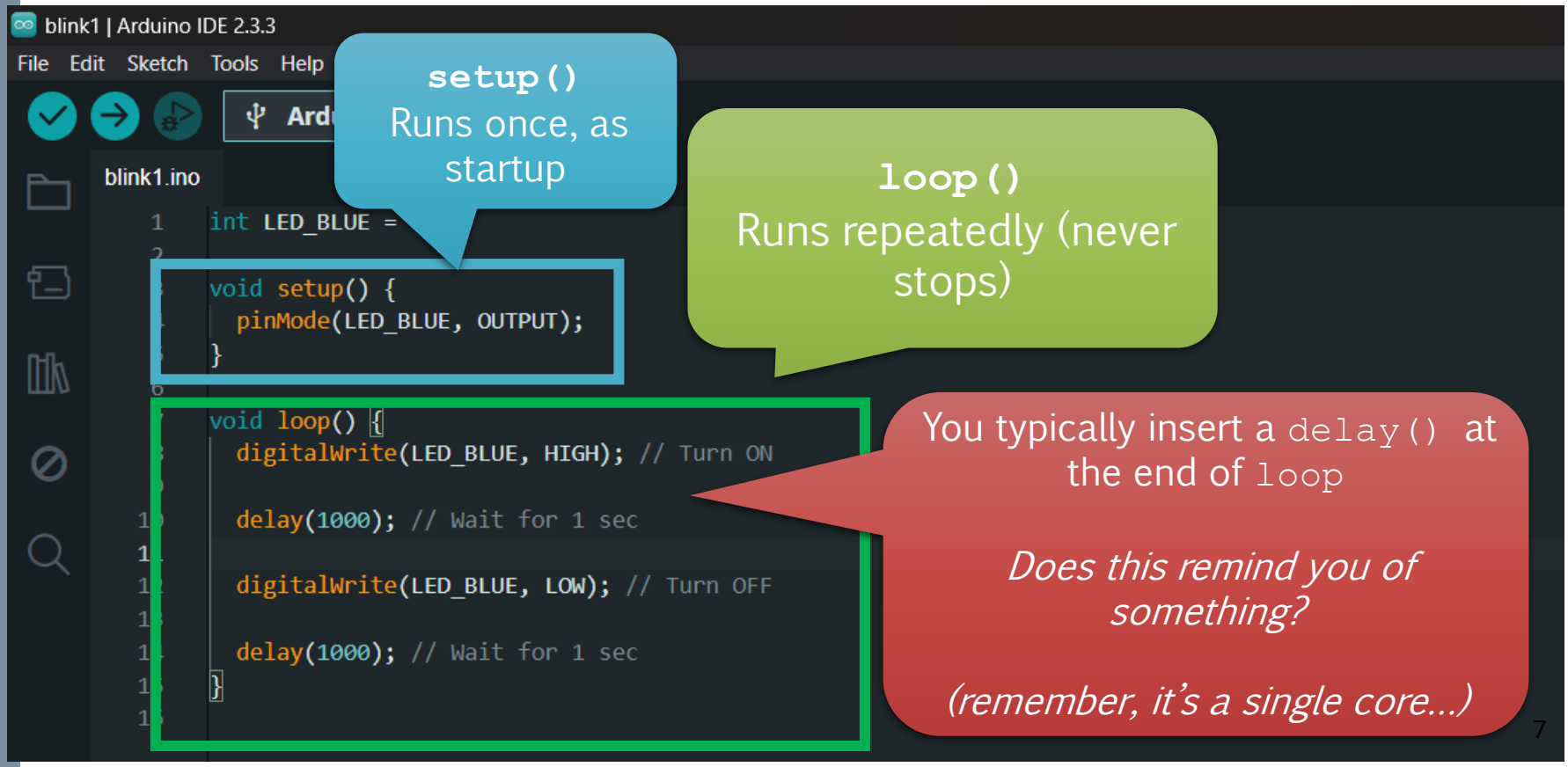


Working environment

One source file (`.ino`)

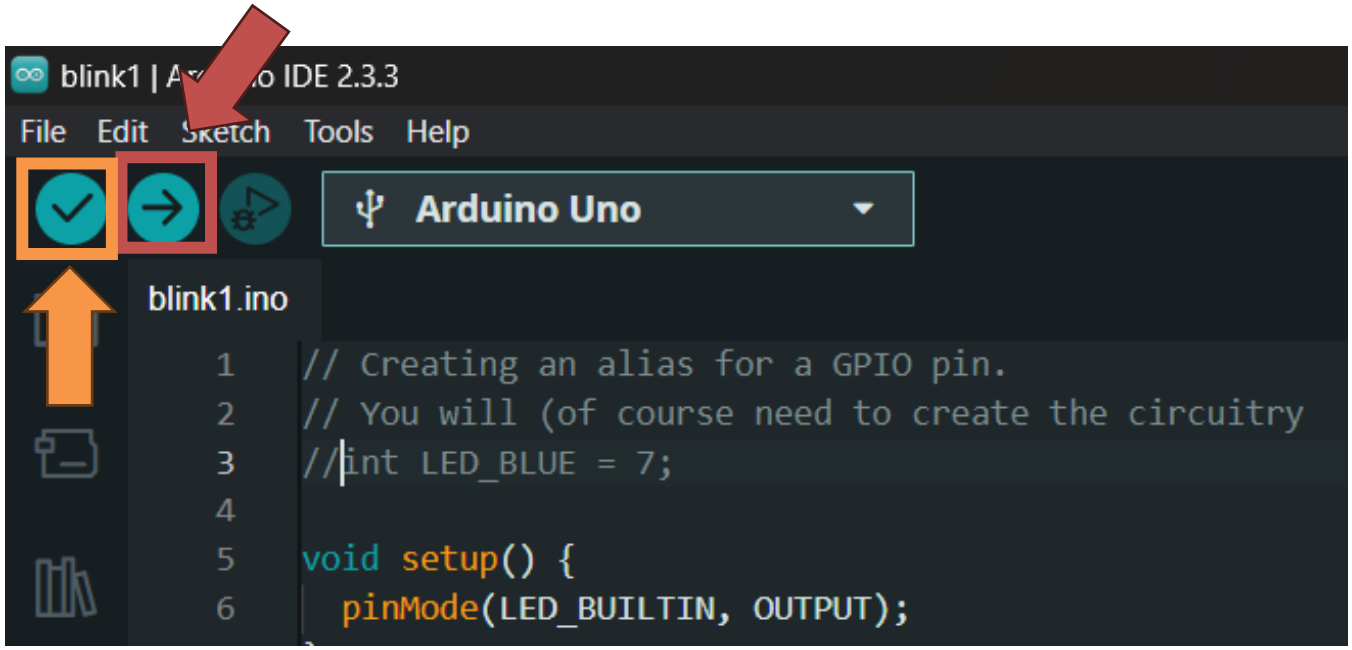
› Include other files with `#include`

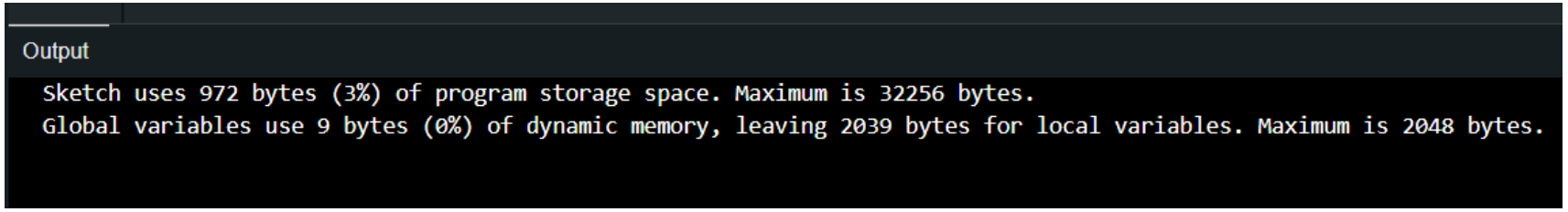
› WARNING: the only language you can use has pseudo-C syntax plus a lot of builtins!



Verify & Upload

- Compile => Verify
- Upload on board (connected via USB)
  - This step also recompiles

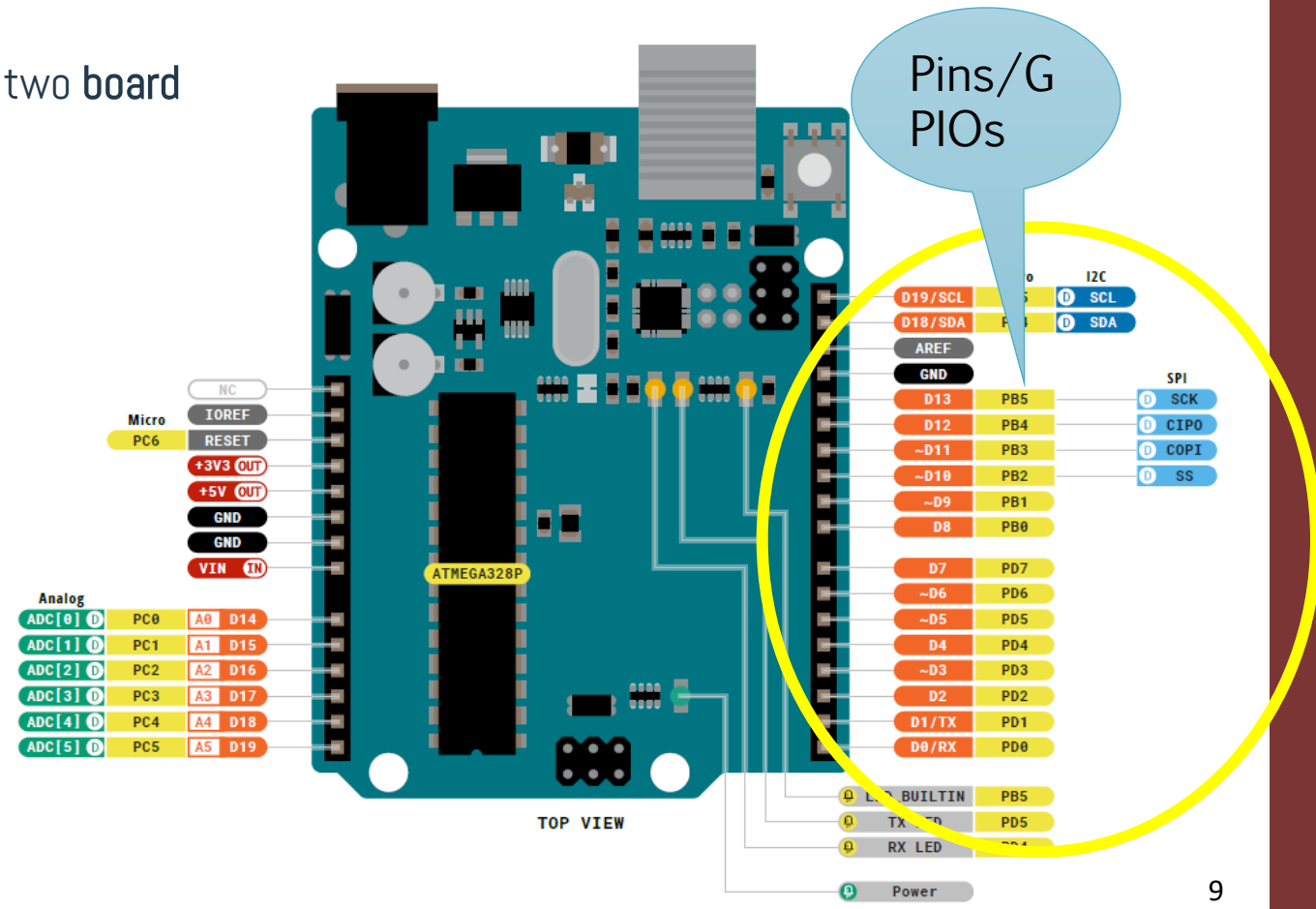




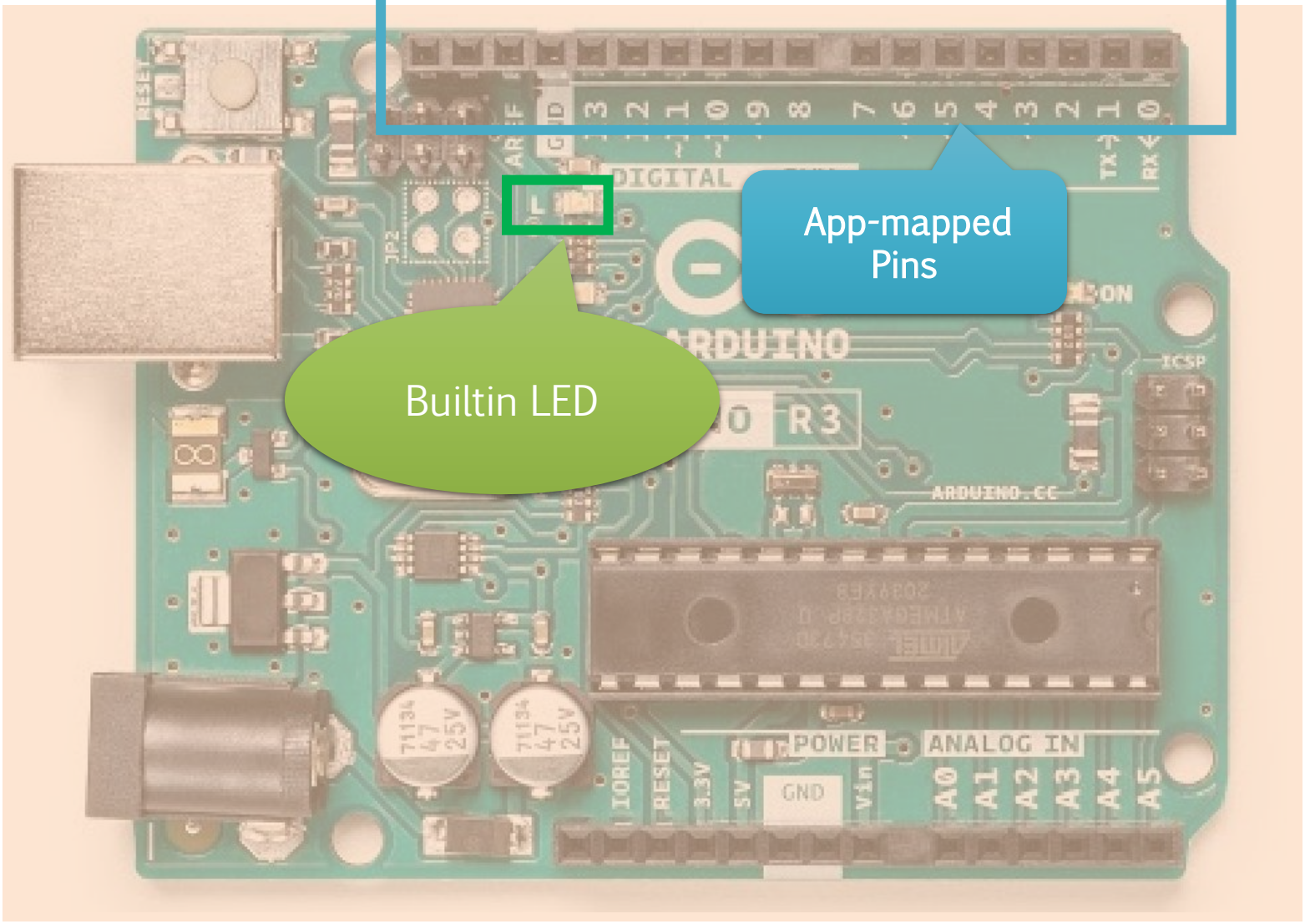
Arduino Uno R3 - Pinout

Our interface towards the external world

- Pins are directly mapped onto R/W primitives
- Pins are divided into two **board blocks**



Arduino Uno R3 - Pins and LEDs



Use LEDs and GPIO pins

We can address pins/LEDs using their number, or alias macros

- `LED_BUILTIN` is language-defined
- Other pins are mapped and can be addressed with their numbers, as `int`

blink.ino

```
// Creating an alias for a GPIO pin.
// You must (of course) create the circuitry
int LED_BLUE = 7;

void setup() {
  pinMode(LED_BLUE, OUTPUT);
}

void loop() {
  digitalWrite(LED_BLUE, HIGH); // Turn ON

  delay(1000);
}
```

<<LANGUAGE BUIILTIN>>

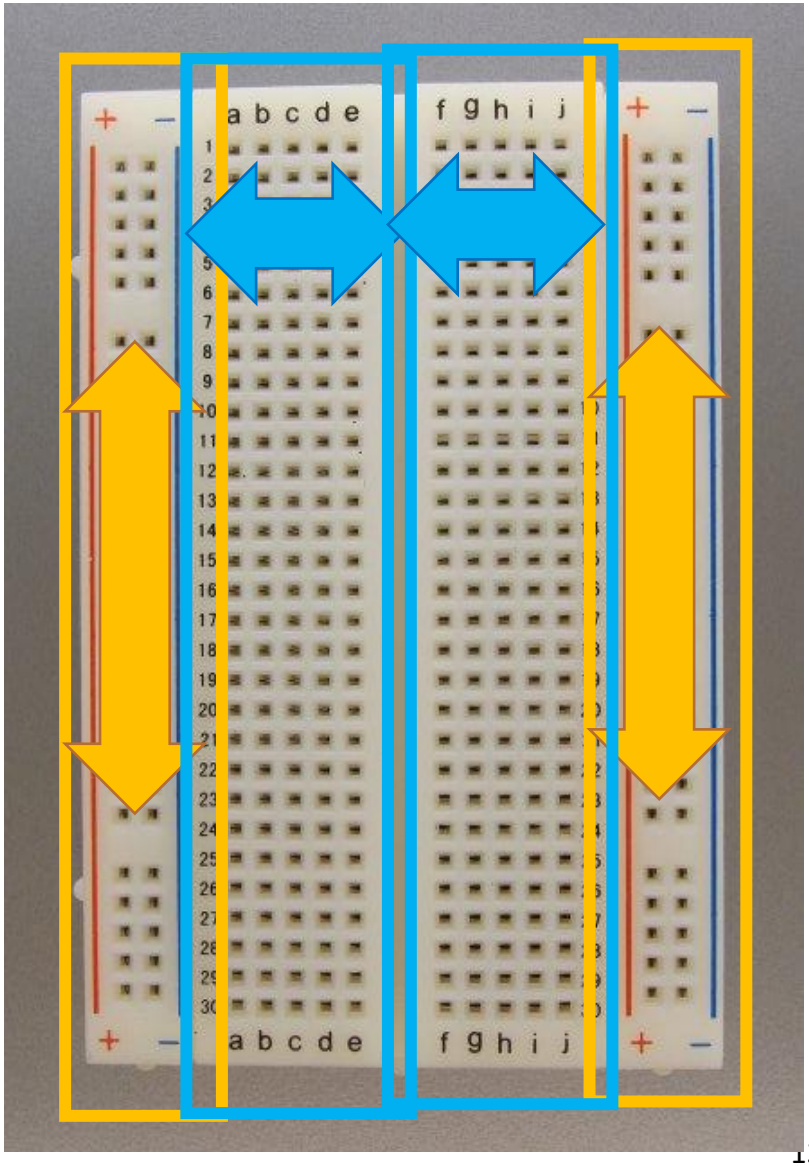
```
/*
 * Params
 *   pin: the Arduino pin number
 *   value: HIGH or LOW
 *
 * Returns:
 *   Nothing
 */
 void digitalWrite (pin, value);

/*
 * Params
 *   pin: the Arduino pin number
 *   mode: INPUT, OUTPUT, or INPUT_PULLUP
 *
 * Returns:
 *   Nothing
 */
 void pinMode (pin, mode);
```

Let's play!

Breadboard

Provides electrical connectivity

- Vertical vs. horizontal rails
- (Typically, power vs other)
- Can use jumper wires



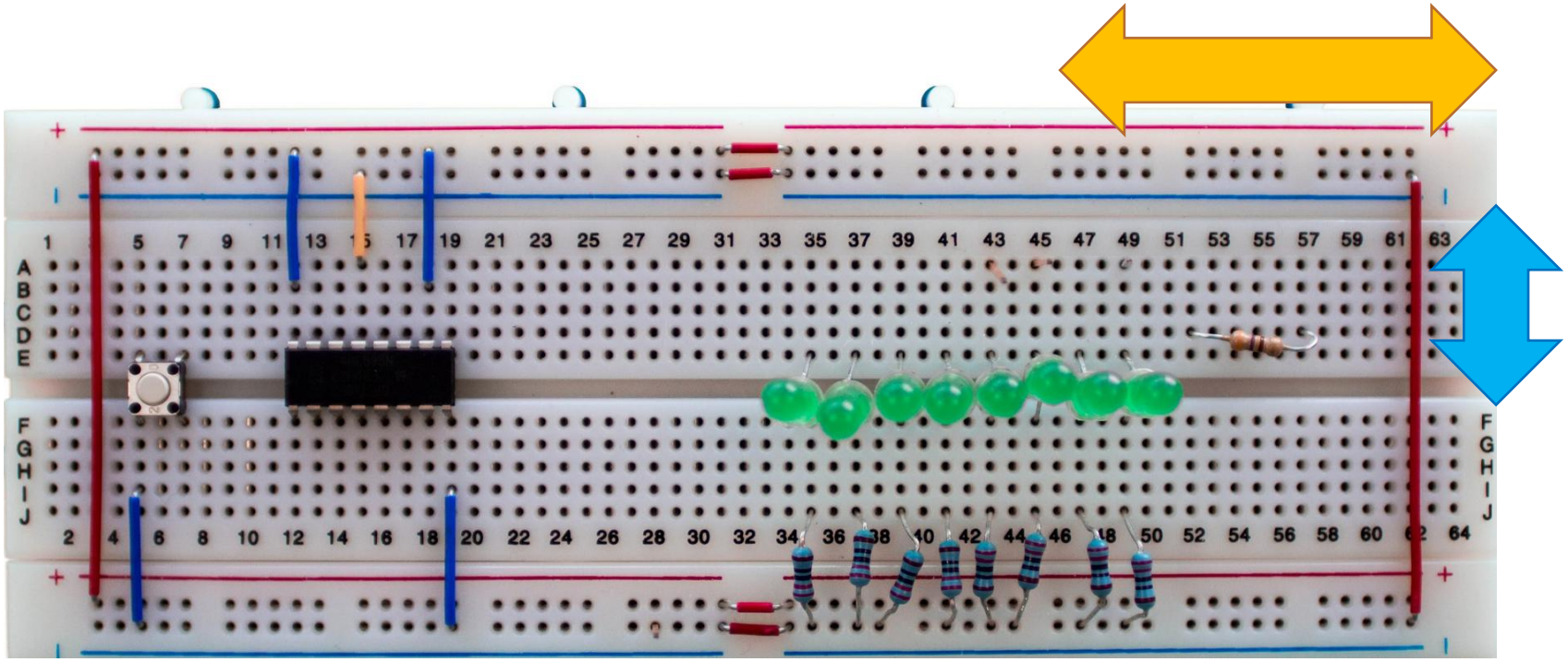
Breadboard

The two sides of the + and - rails are wired together

› Typically, used for power/GND

Brought to the internal rails with jumper wires

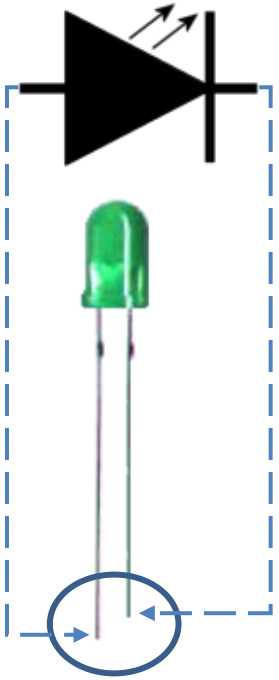
› Where core/chip and other stuff reside



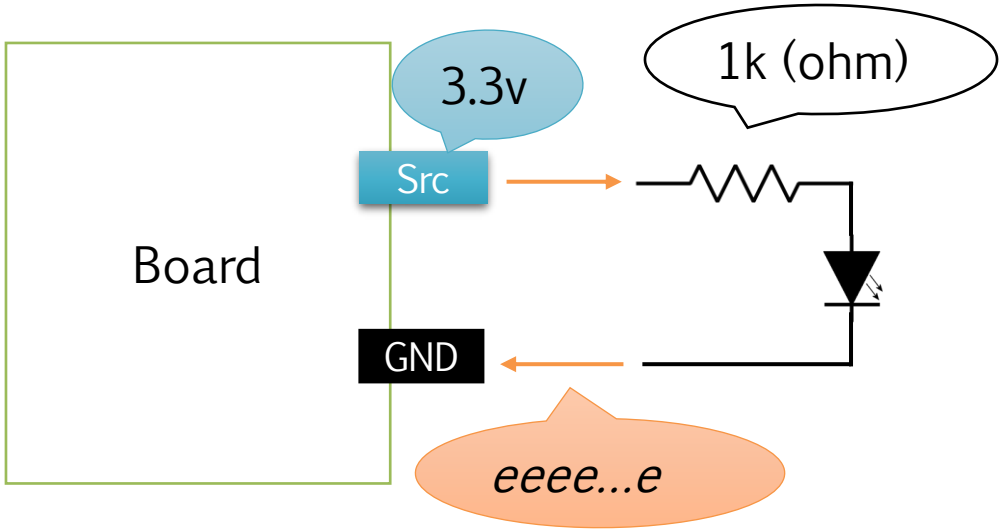
Finally...LEDs

Light Emitting Diodes

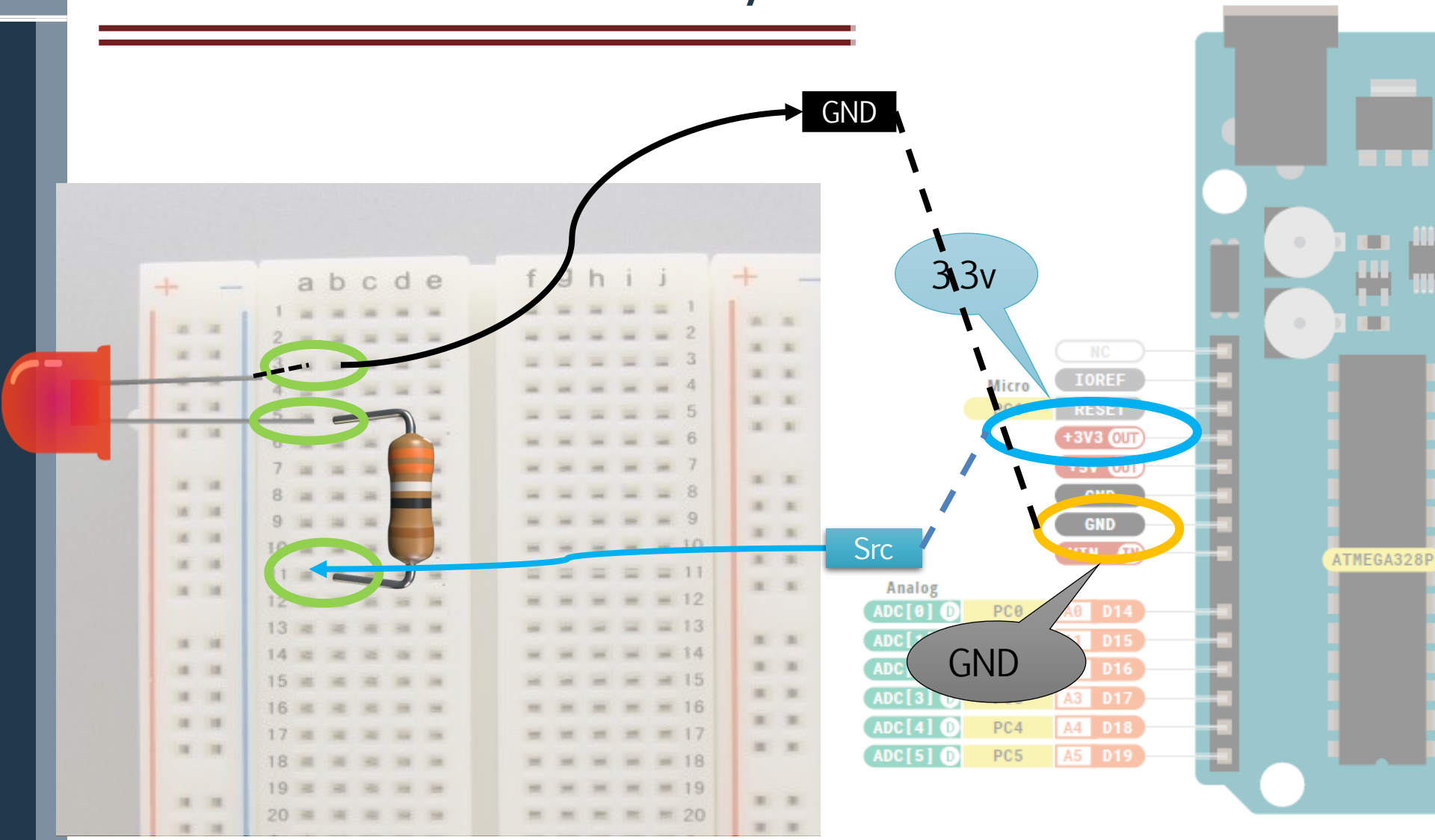
- You feed with electrons; they light up
- They have a side!!!!
- They need a resistance to lower the charge



Wrong wiring => you burn them...



E/E system


GND
3.3v
Src
GND
Micro
NC
IOREF
RESET
+3V3 OUT
GND
Analog
ADC[0] PC0 A0 D14
ADC[3] A3 D17
ADC[4] PC4 A4 D18
ADC[5] PC5 A5 D19
ATMEGA328P

Exercise

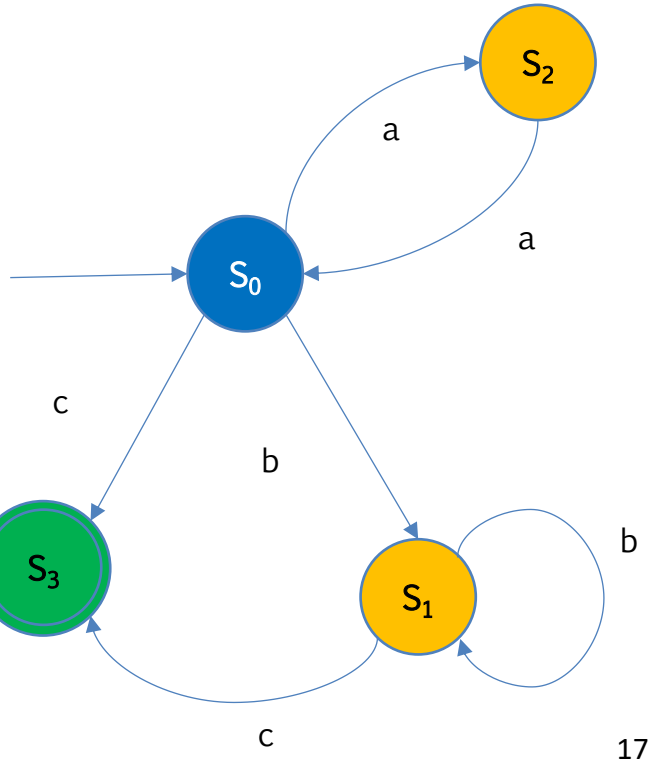
Let's code!

› Implement the Moore machine of the FSM that understands whether a words is from L

"Identify even sequences of a (even empty),

followed by one, or more, or no, b, ended by c"

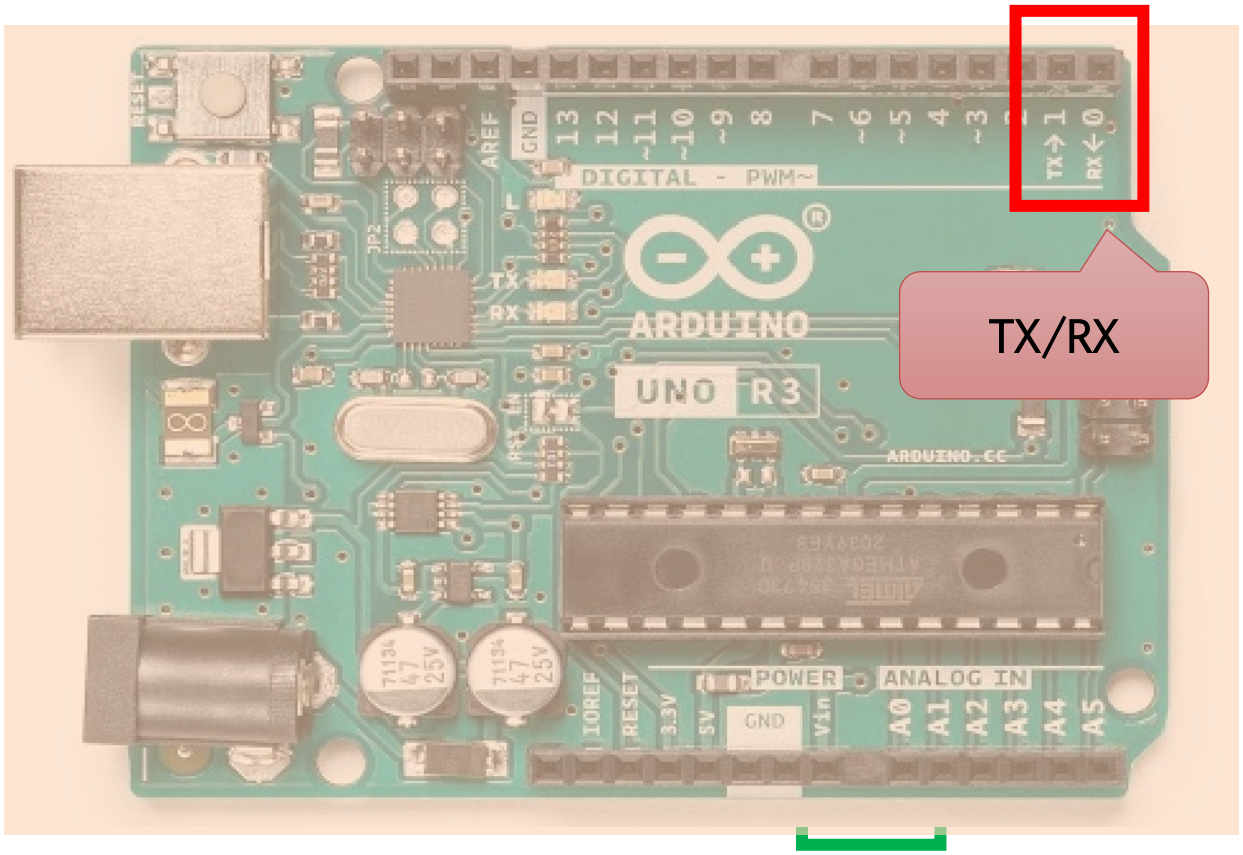
› ..and turns on the corresponding led color
  - Blue => GPIO 0
  - Red (error state) => GPIO 1
  - Yellow => GPIO 2
  - Green => GPIO 3



Serial communication

Serial port(s) are available through multiple objects, that abstract them

- `Serial`, `Serial1`, `Serial2`, etc
- Arduino Uno R3 only has `Serial`, mapped on (RX-TX) pins 0 and 1



Serial: system header

<<LANGUAGE BUIILTIN>>

```
/*
 * Serial: serial port object.
 * speed: in bits per second (baud). Allowed data types: long.
 */
Serial.begin(speed);

/*
 * Serial: serial port object.
 * val: a value to send as a single byte.
 * str: a string to send as a series of bytes.
 * buf: an array to send as a series of bytes.
 * len: the number of bytes to be sent from the array.
 */
Serial.write(val);
Serial.write(str);
Serial.write(buf, len);
```

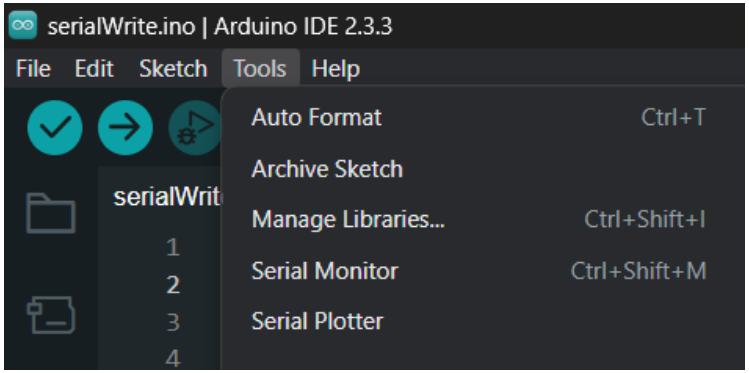
blink.ino

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.write("hello\n");

}
```

On your machine… (1)

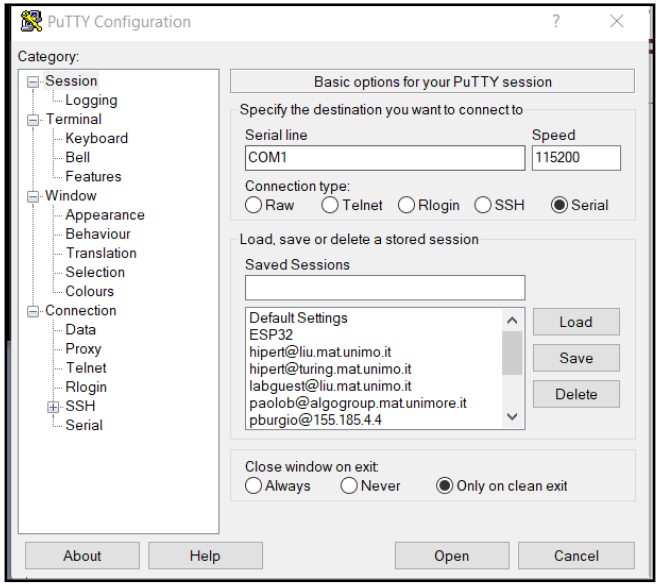
First, test with a “standard” serial Monitor

- Arduino IDE has a builtin monitor port



- Linux
  - `sudo apt install minicom`
  - Serial/USB ports are typically `/dev/ttySOMETHING`

- Windows
  - Putty
  - Serial/USB ports are COMx



On your machine… (2)

Programmatically read from serial/USB

- C++
  - https://github.com/imabot2/serialib
- Python
  - pySerial

References

Course website

- http://hipert.unimore.it/people/paolob/pub/Industrial_Informatics/index.html

My contacts

- paolo.burgio@unimore.it
- http://hipert.mat.unimore.it/people/paolob/

Resources

- A "small blog -> http://www.google.com
- Everything you need on arduino
  - https://www.arduino.cc
  - https://docs.arduino.cc