PHYSICAL SENSORS FOR ENVIRONMENTAL SIGNALS

Irene Nutini

Master Degree in Artificial Intelligence for Science and Technology (AI4ST)

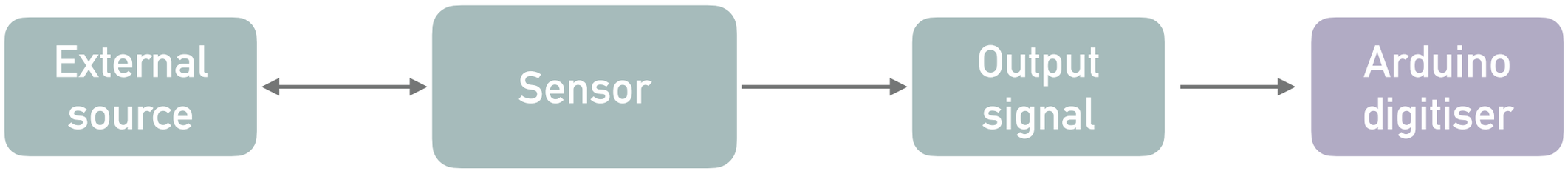
A.y. 2023-2024

OUTLINE OF THE COURSE

- Lecture 1: Introduction to environmental signals and physical sensors
- Lab 1: Introduction to instruments for measurements
- Lecture 2: Vibrations: sources and detection
- Lab 2: Characterisation of an acoustic system
- Lecture 3: Distance, position and speed measurement
- Lab 3: Measuring distance with ultrasounds and speed with an accelerometer
- Lecture 4: Electromagnetic radiation: sources and detection
- Lab 4: Detecting and generating light

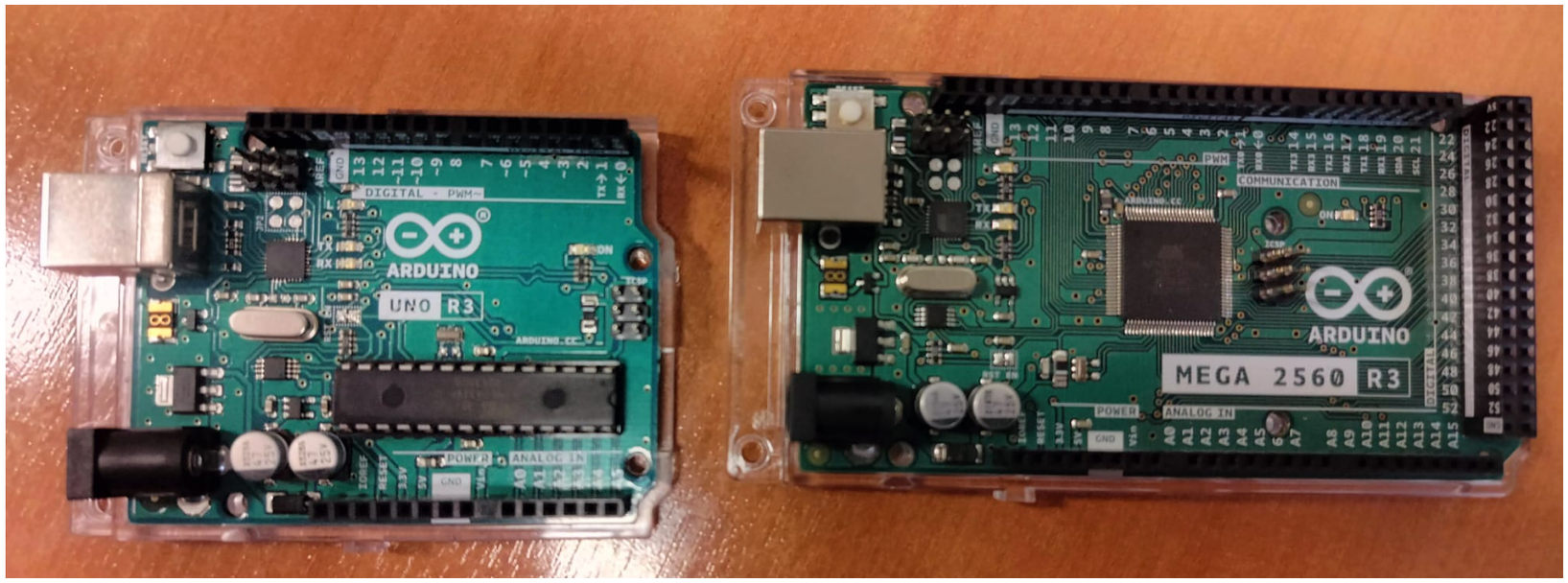
SENSING THE ENVIRONMENT

EXAMPLE: ULTRASOUND DETECTOR/ACCELEROMETER READOUT CHAIN



- Source: element in space (static / in motion)
- Sensor: ultrasounds detector / accelerometer → ***Lab.3 (today)***
- Read the signal output: Arduino digitiser

ARDUINO

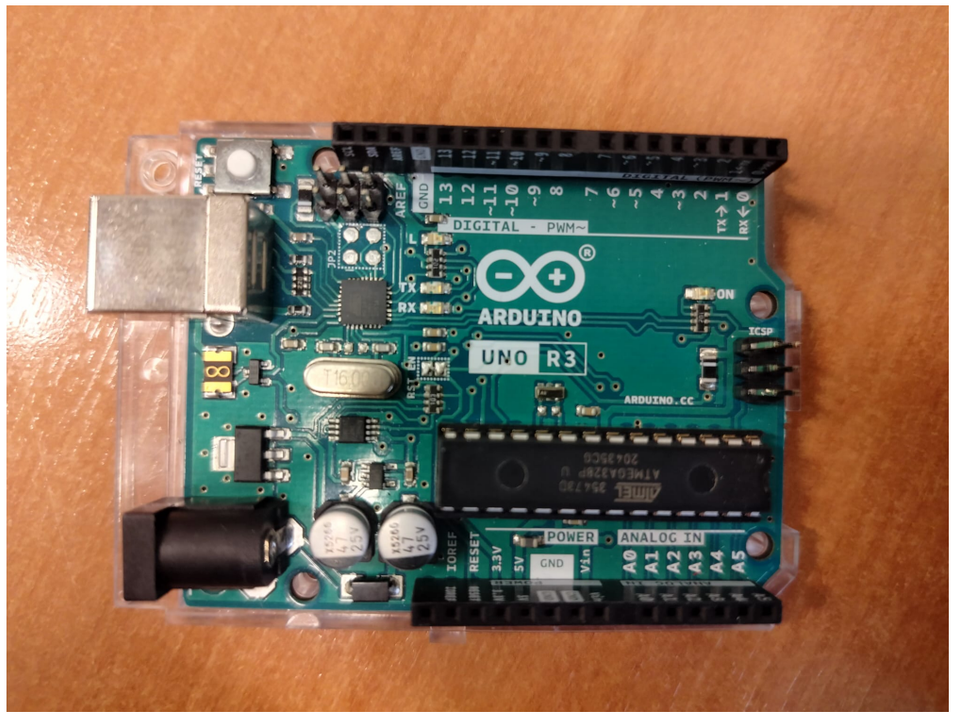


Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It'svintended for artists, designers, hobbyists,
and anyone interested in creating interactive objects or environments.
From Arduino.cc

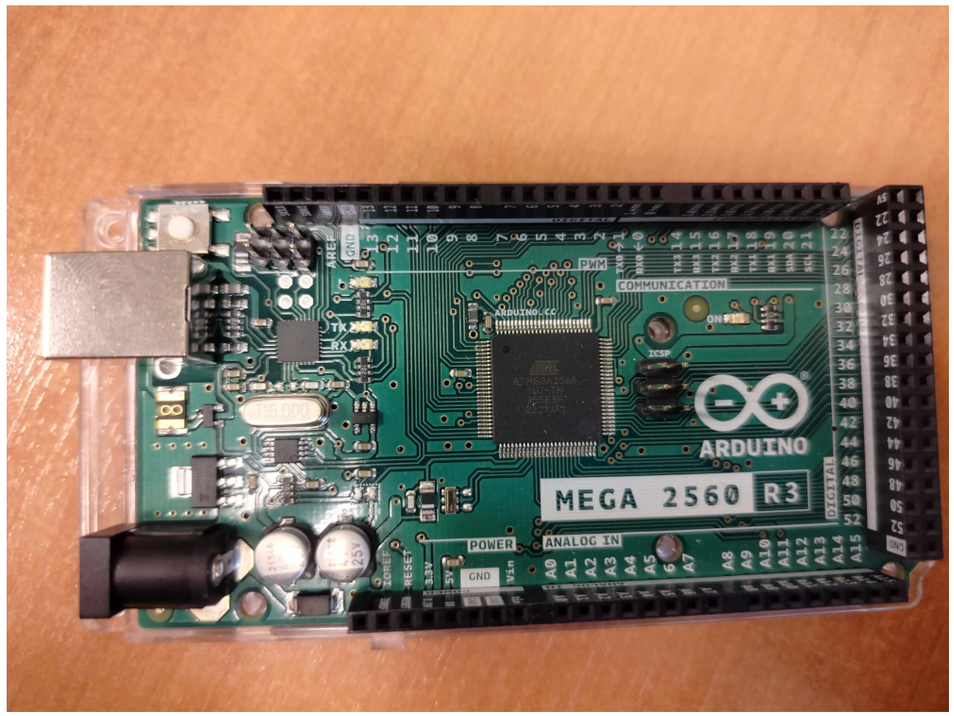
Main components
- Microcontroller
- Analogic and digital I/O pins
- Flash memory
- USB port for serial communication

What Arduino can do:
- Read sensors
- Control peripheral devices
- Communicate via serial port
- Remotely programmable

ARDUINO



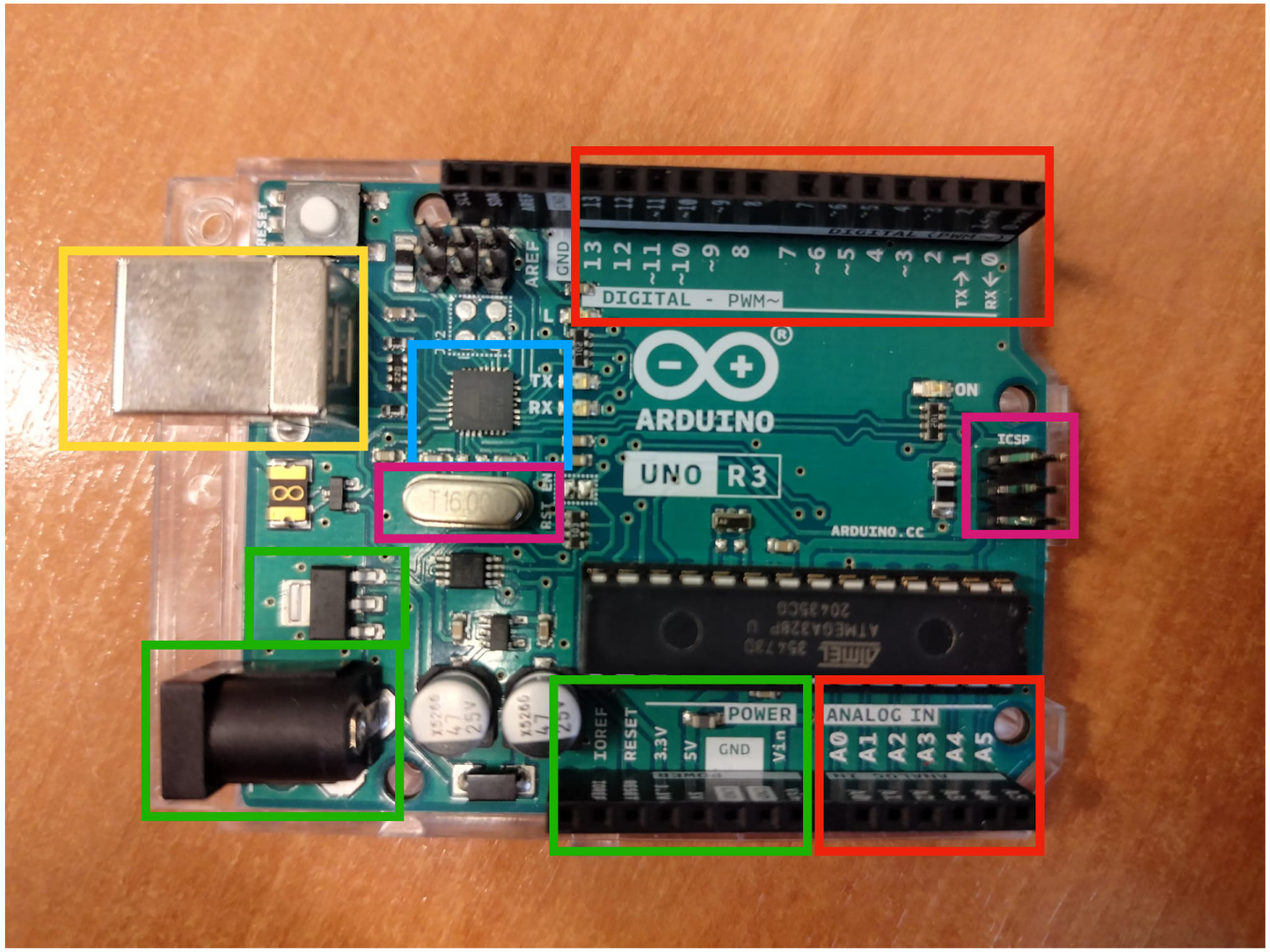
The **Arduino Uno** is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button.



The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

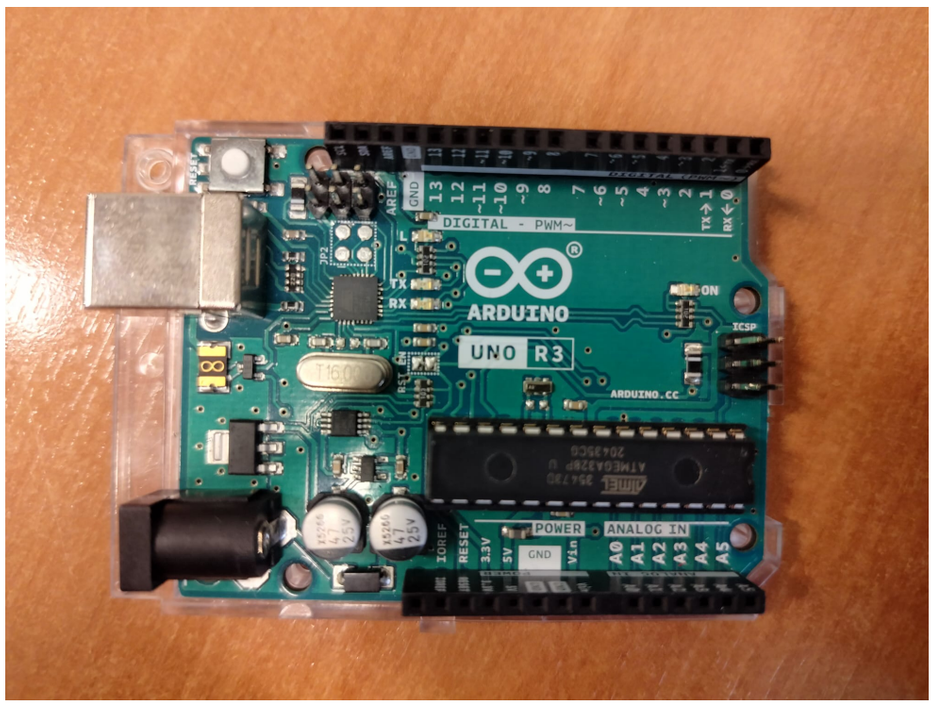
ARDUINO

Components:
- Microcontroller
- Pins I/O
- Power: power jack, voltage regulator
- ICSP and clock
- USB port



ARDUINO: SOFTWARE

- Arduino is a programmable device
- Dedicated software development environment (Arduino IDE)
- Programming language similar to C/C++
- Dedicated libraries for handling input/output



Arrays

Arduino	*Processing*
int bar[8]; bar[0] = 1;	int[] bar = new int[8]; bar[0] = 1;
int foo[] = { 0, 1, 2 };	int foo[] = { 0, 1, 2 }; *or* int[] foo = { 0, 1, 2 };

Loops

Arduino	*Processing*
int i; for (i = 0; i < 5; i++) { ... }	for (int i = 0; i < 5; i++) { ... }

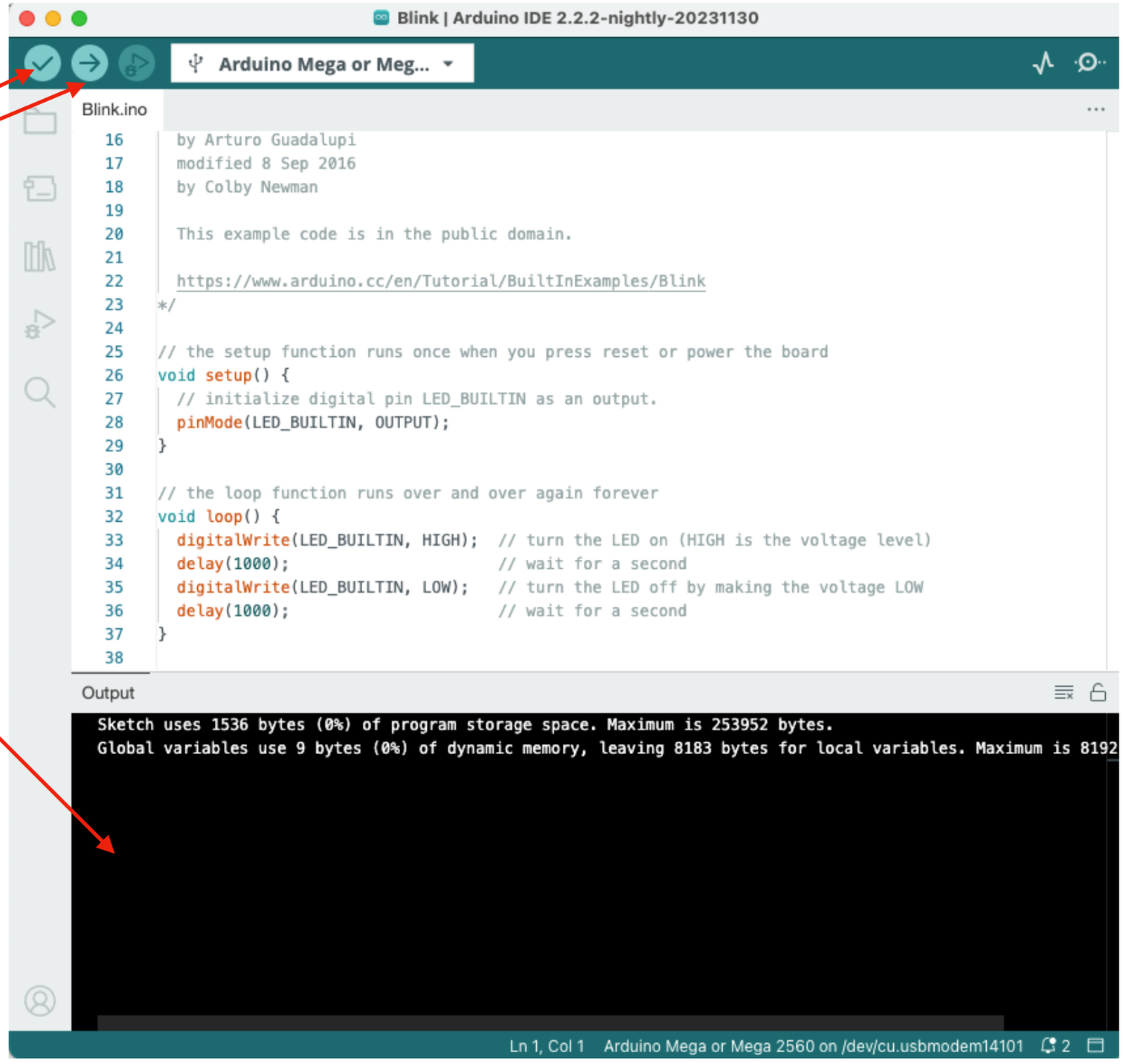
Printing

Arduino	*Processing*
Serial.println("hello world");	println("hello world");
int i = 5; Serial.println(i);	int i = 5; println(i);
int i = 5; Serial.print("i = "); Serial.print(i); Serial.println();	int i = 5; println("i = " + i);

ARDUINO: SOFTWARE

Dedicated software development environment (Arduino IDE)

1. Compile/Verify
2. Upload/Run
3. Status



ARDUINO: SOFTWARE

Dedicated software development environment (Arduino IDE)

Main functions:
- *setup()*: called once when the program starts. Initialization of variables and pins status
- *loop()*: loop inside which the code has to be implemented

Presence of dedicated libraries for accessing the pins and set/read their status

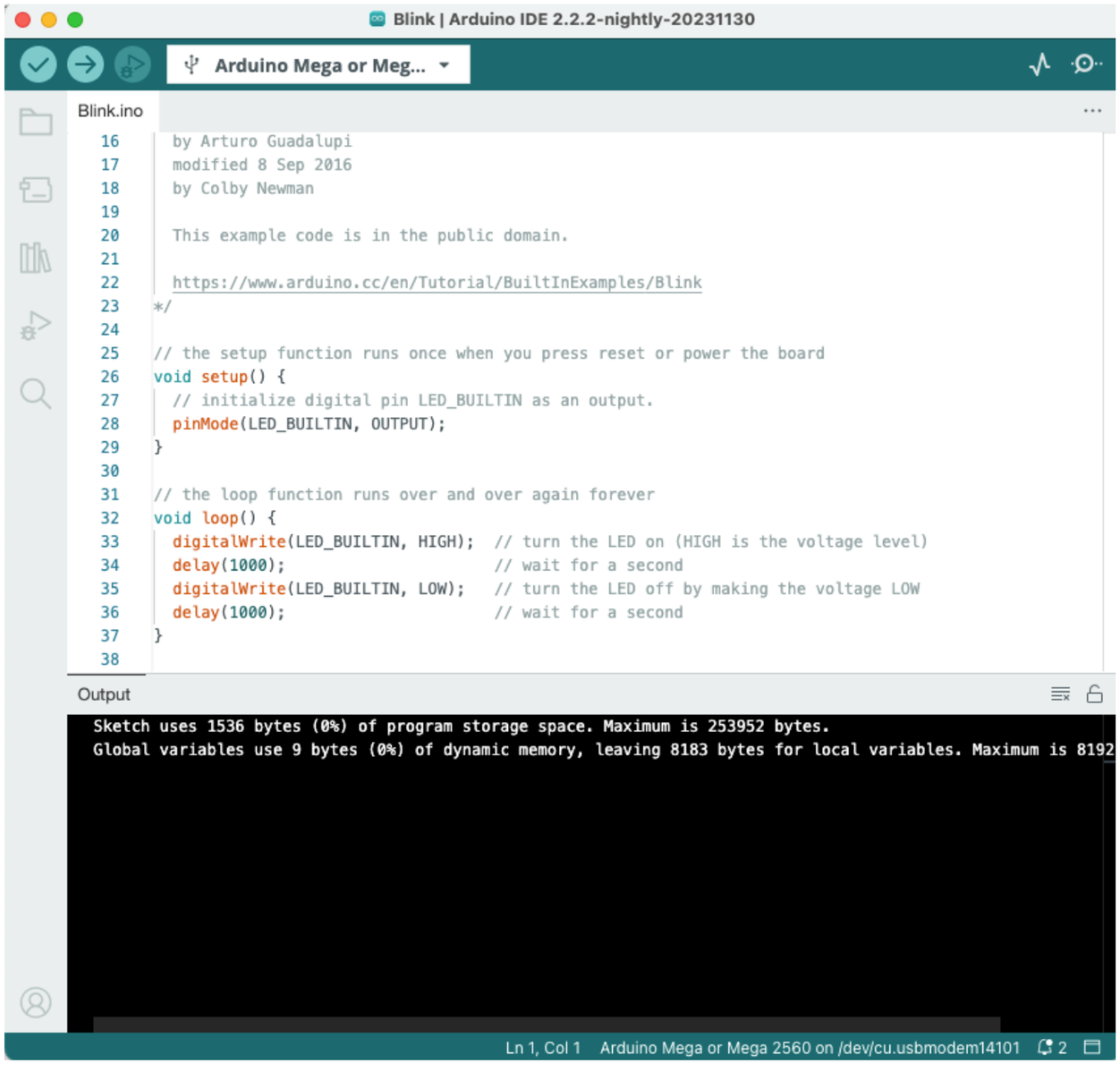
```
24
25  // the setup function runs once when you press reset or power the board
26  void setup() {
27    // initialize digital pin LED_BUILTIN as an output.
28    pinMode(LED_BUILTIN, OUTPUT);
29  }
30
31  // the loop function runs over and over again forever
32  void loop() {
33    digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
34    delay(1000);                      // wait for a second
35    digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
36    delay(1000);                      // wait for a second
37  }
38
```

ARDUINO: SOFTWARE

Dedicated software development environment (Arduino IDE)

Install and test:
1. Download the software from: http://arduino.cc/en/Main/Software
2. Connect the board to the PC via USB
3. Launch ArduinoIDE
4. Open the Basics Example: ‘Blink.ino’
5. Select the Arduino board type
6. Verify and upload the program.

If the setup worked, after few s from the upload, the orange LED should start blinking



ARDUINO: WARNINGS

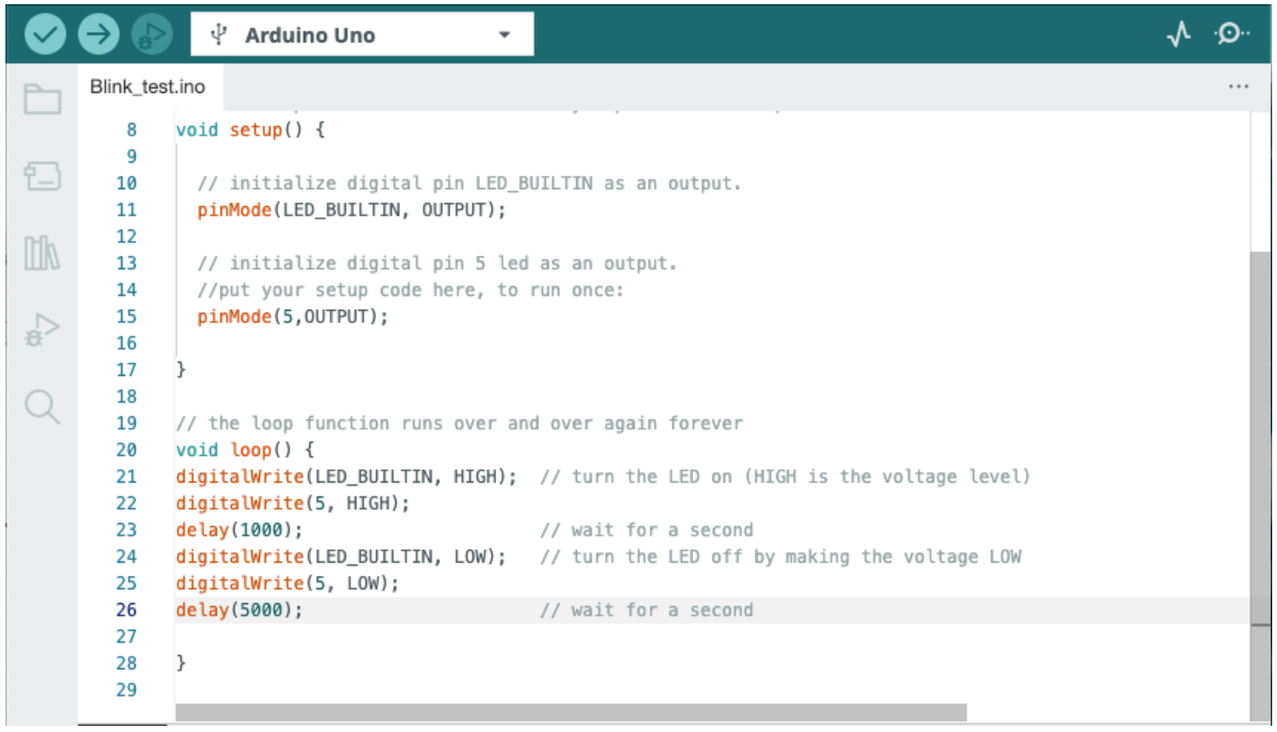
10 ways to destroy Arduino (beware!):

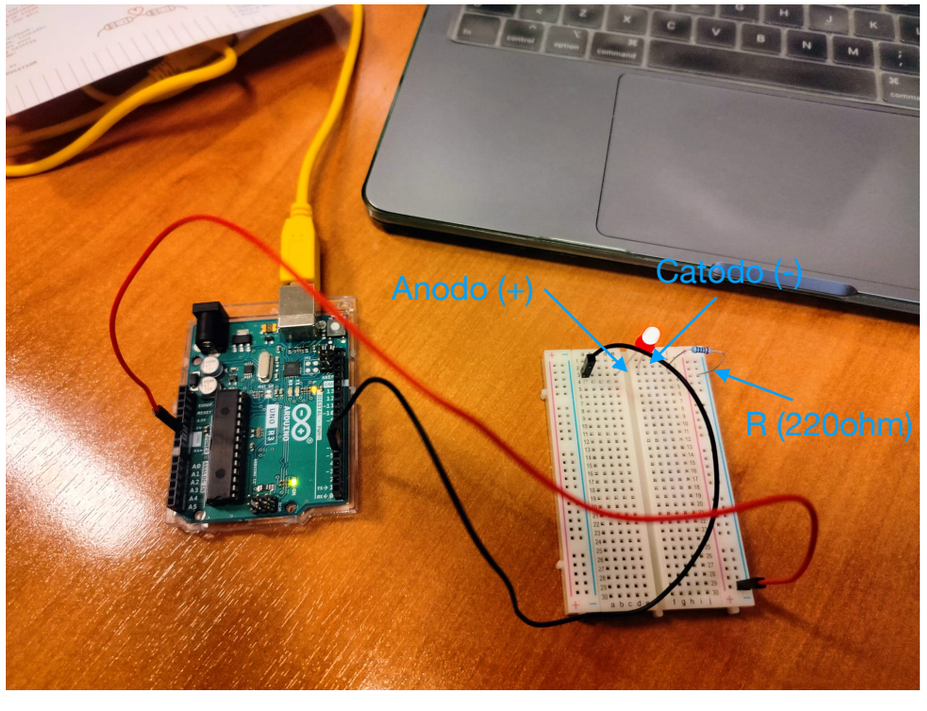
1. Set I/O pins to ground
2. Connect I/O pins together
3. Apply too much voltage on I/O pins
4. Apply voltage on Vin with inverted polarity
5. Apply > 5V on the ‘5V’ pin
6. Apply > 3.3V on the ‘3.3V’ pin
7. Set Vin to ground
8. Apply >13V to the reset
9. Apply voltage to the ‘5V’ pin and charge Vin
10. Exceed the max current for the microcontroller (200mA)

ARDUINO: YOUR FIRST TEST

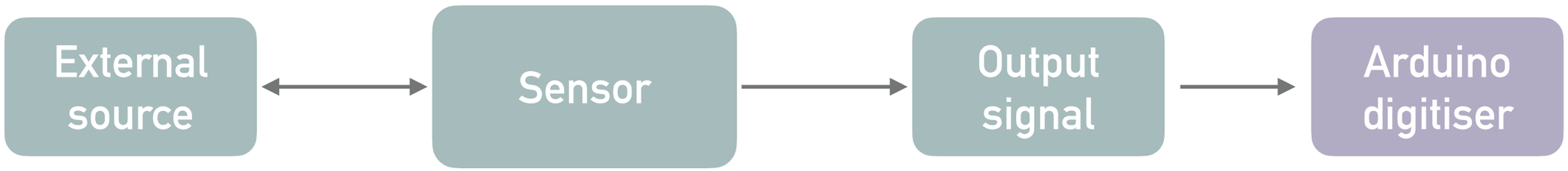
Playing with the blinking LED
Start with the Basics
Example: ‘Blink.ino’

Modify the program to:
1. Change the blinking frequency
2. Add other LEDs on digital pins
3. Turn on multiple LEDs sequentially
4. Change the LED brightness





EXAMPLE: ULTRASOUND DETECTOR/ACCELEROMETER READOUT CHAIN



- Source: element in space (static / in motion)
- Sensor: ultrasounds detector / accelerometer → ***Lab.3 (today)***
- Read the signal output: Arduino digitiser

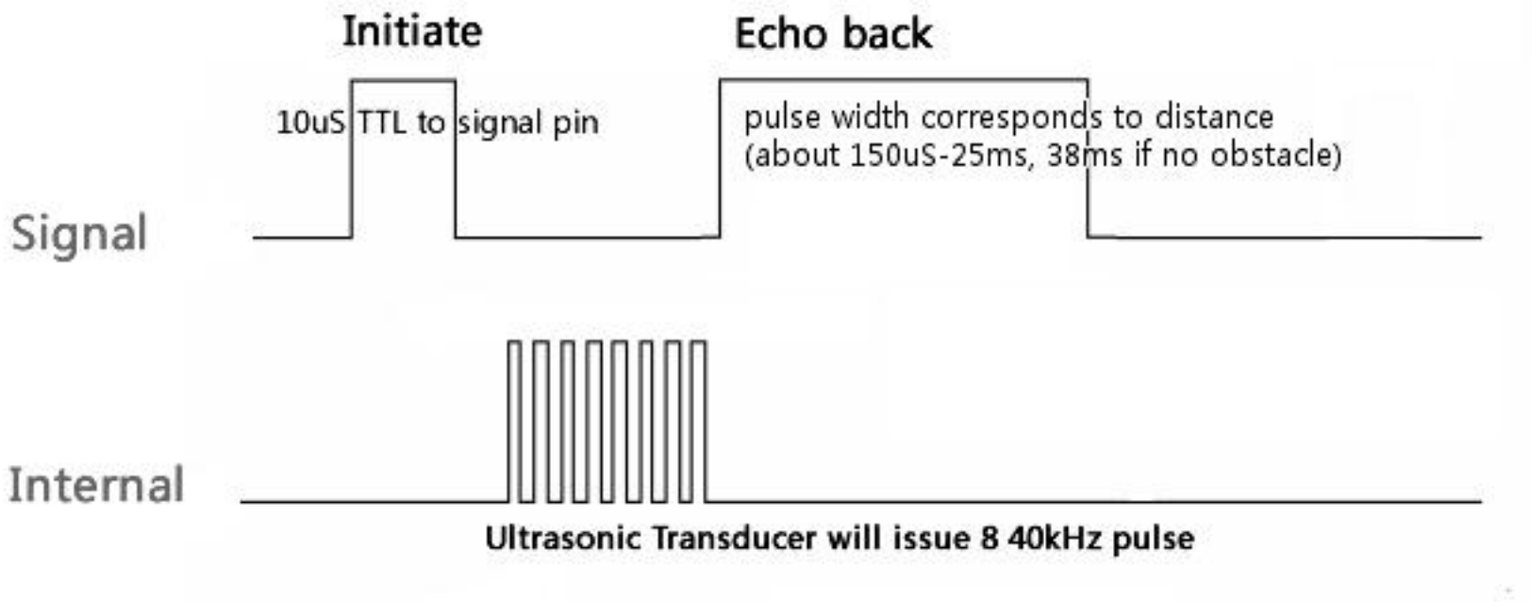
ULTRASOUND DETECTOR

- Source: element in space (static/dynamic)
- Sensor: ultrasounds detector
- Read the signal output: Arduino digitiser to serial port

Reference example: https://win.adrirobot.it/sonar/HC-SR04/Sensore_sonar_HC-SR04.htm

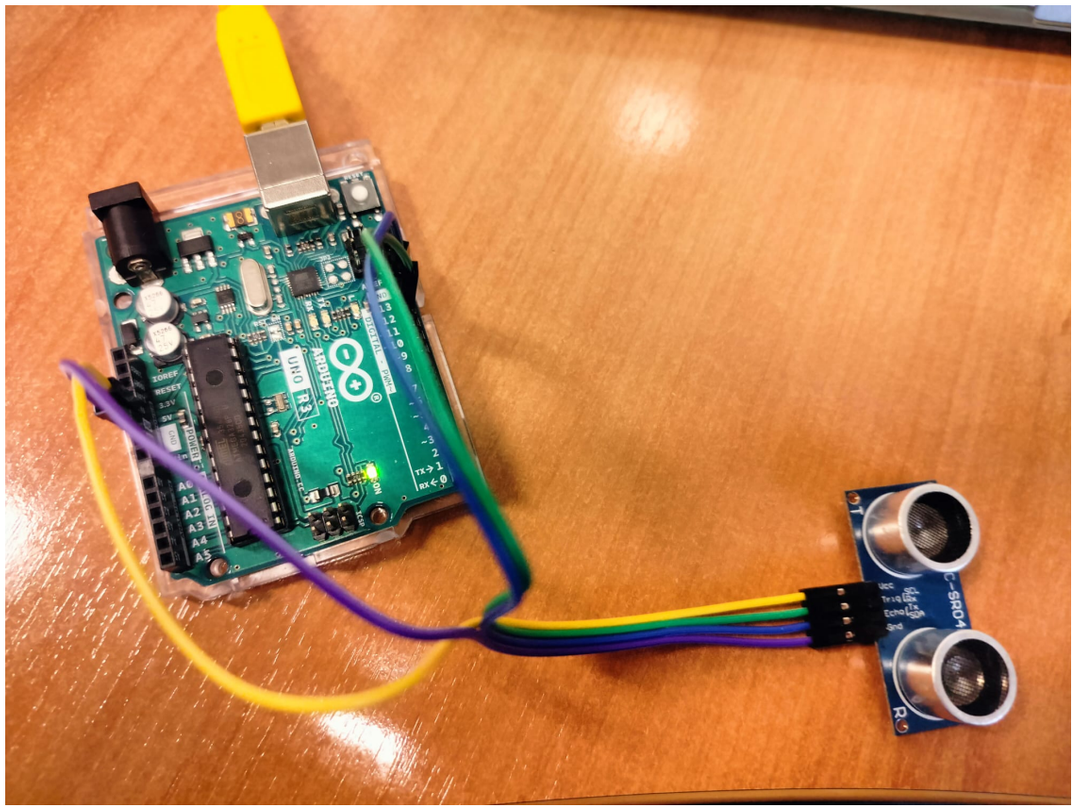
Other examples: https://docs.arduino.cc/built-in-examples/sensors/Ping

- Transmitter
- Receiver



ULTRASOUND DETECTOR

- Source: element in space (static/dynamic)
- Sensor: ultrasounds detector
- Read the signal output: Arduino digitiser to serial port



Ultrasound sensor (HC-SR04) to Arduino
- Vcc to pin 5V
- Trig to pin12
- Echo to pin 23
- Gnd to pin Gnd

ULTRASOUND DETECTOR

- Reading the Signal output with Arduino

Onde_sonore.ino

```
#define TRIG 12
#define ECHO 13

#include <Ultrasonic.h>

Ultrasonic ultrasonic(12, 13); //(trig, echo)

void setup() {

Serial.begin(9600);

}

void loop() {

Serial.print("New Measurement. \n");

int i =0;
Serial.print("Time[ms] and Distance[cm]: \n");
while(i< 20){
  Serial.println(String(micros()*1e-3) + String('\t') + ultrasonic.read());
  //Serial.println(micros()*1e-3);
  delay(1000);
  i++;
}

exit(0);

}
```

Output Serial Monitor ×

Message (Enter to send message to 'Arduino Uno' on '/dev/cu.usbmodem14101') | No Line Ending | 9600 baud

```
New Measurement.
Time[ms] and Distance[cm]:
18.67   293
1037.70 293
2056.76 293
3075.84 293
4094.94 293
5114.03 293
6133.08 293
```

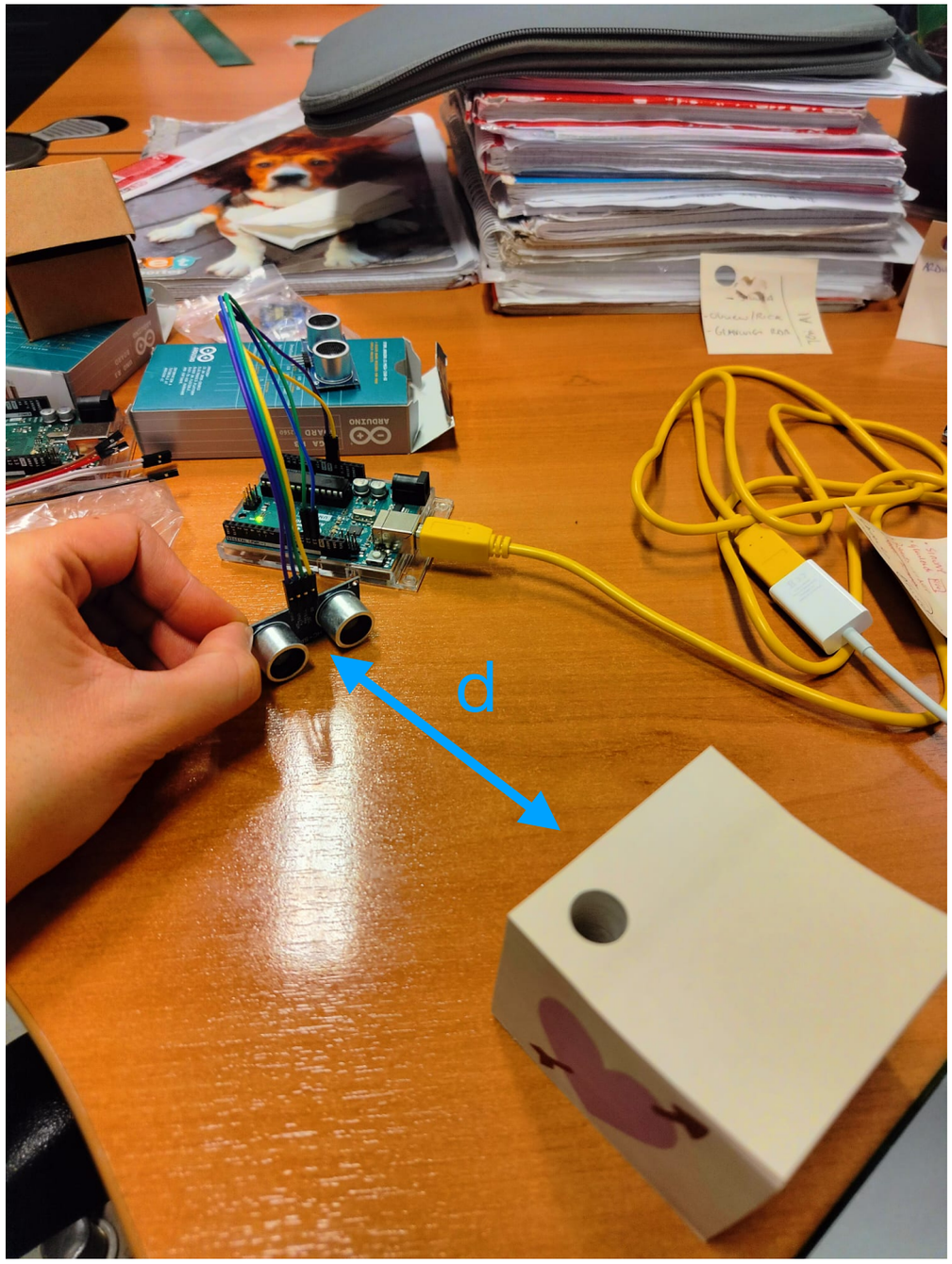
Ln 2, Col 16 Arduino Uno on /dev/cu.usbmodem14101 3

- *https://github.com/ErickSimoes/Ultrasonic/tree/master/src*

ULTRASOUND DETECTOR: MEAS (1)

Position a static object in front of the sensor (theta=0)

- Goal: Check the range capabilities of the us-sensor, varying the object distance d. Compare reconstructed d (us-sensor) vs measured d (tape)
- Goal: for a fixed d, check the capabilities of the sensor to reconstruct the distance for different size/shape of the object

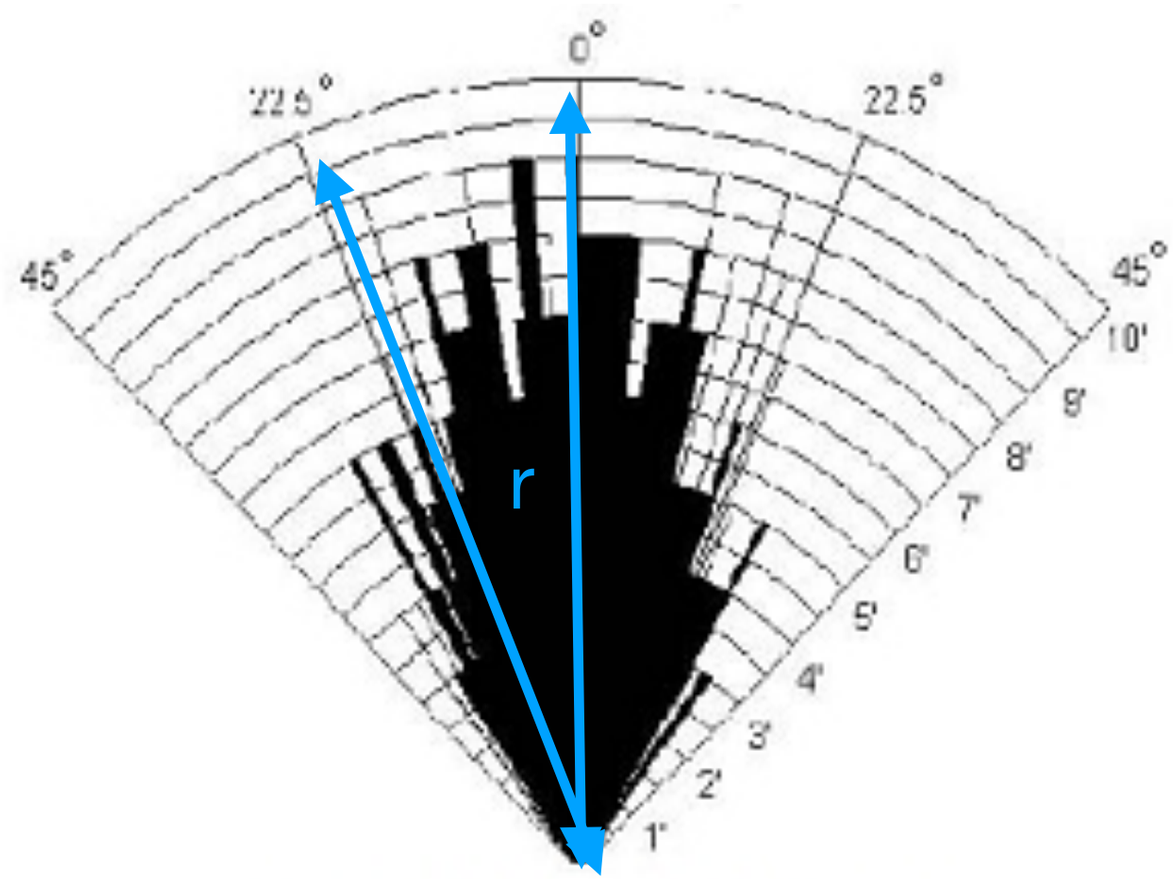


Additional: Use 2 us-sensors to sample the position variation in a plane of an object

ULTRASOUND DETECTOR: MEAS (2)

Position a static object at a given distance and angle from the sensor (r, theta)

- Goal: Check the capabilities to measure r vs theta. Compare reconstructed r (us-sensor) vs measured r (tape) at different angles
- Goal: Check the effect of possible interfering objects in the quality of the measurement



Practical test of performance,

Best in 30 degree angle

ULTRASOUND DETECTOR: MEAS (3)

Position an object at a given distance d in front of the sensor and, while starting the digitization, move it along the vertical direction on the plane

- Goal: Plot distance vs time from the us-sensor. Check the capabilities to reconstruct the average speed
- Goal(+): free fall (from height h (tight the object with a wire) to ground (where the us-sensor is positioned)