PHYSICAL SENSORS FOR ENVIRONMENTAL SIGNALS

Irene Nutini

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

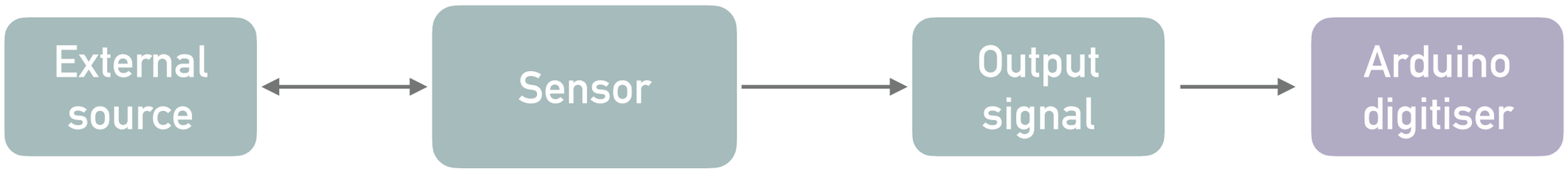
A.y. 2024-2025

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

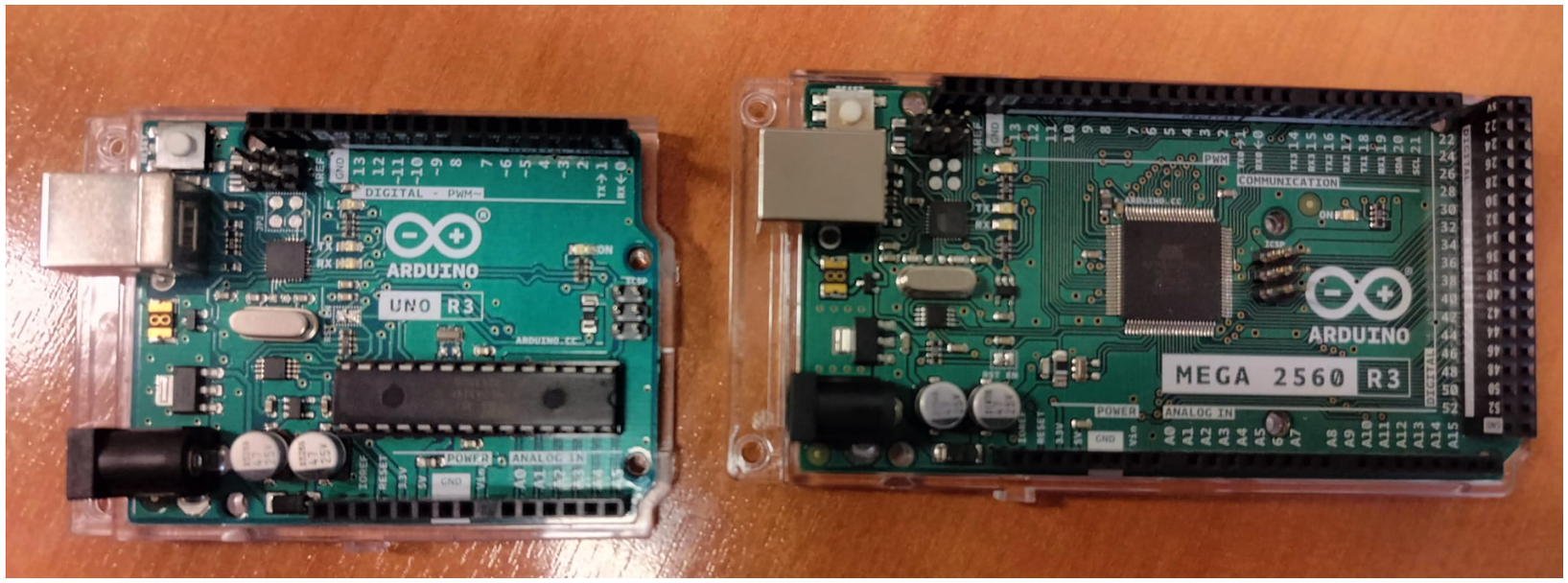
EXAMPLE: MICROPHONE READOUT CHAIN



- Source: acoustic sound (voice/background/audio)
- Sensor: microphone
- Read the signal output: Arduino digitiser

Lab.2 (today)

ARDUINO

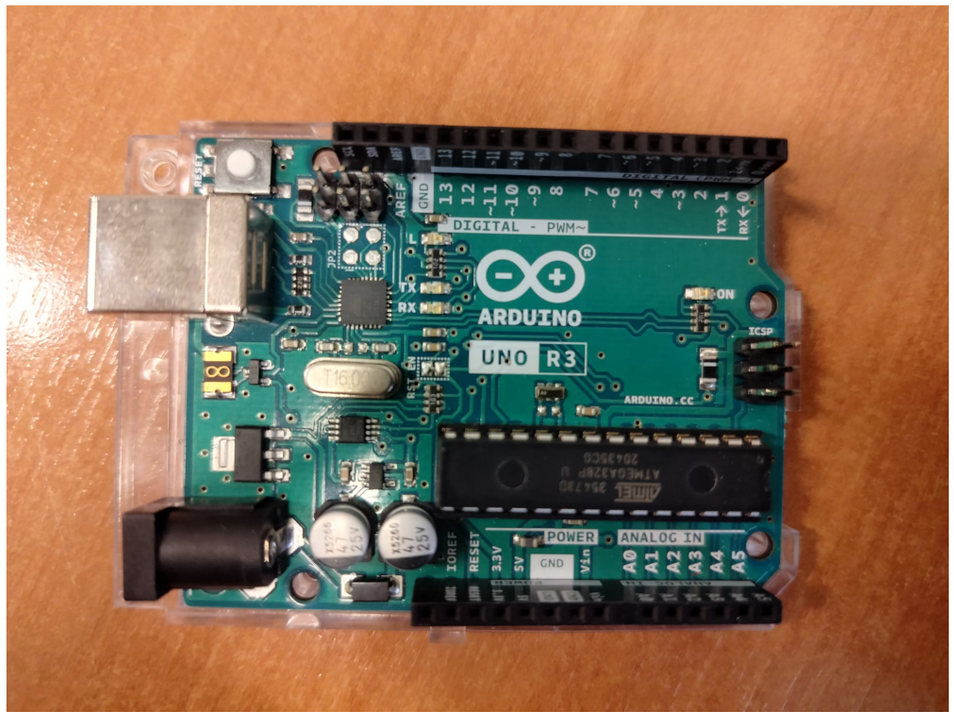


Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It'svintended for artists, designers, hobbyists,
and anyone interested in creating interactive objects or environments.
From Arduino.cc

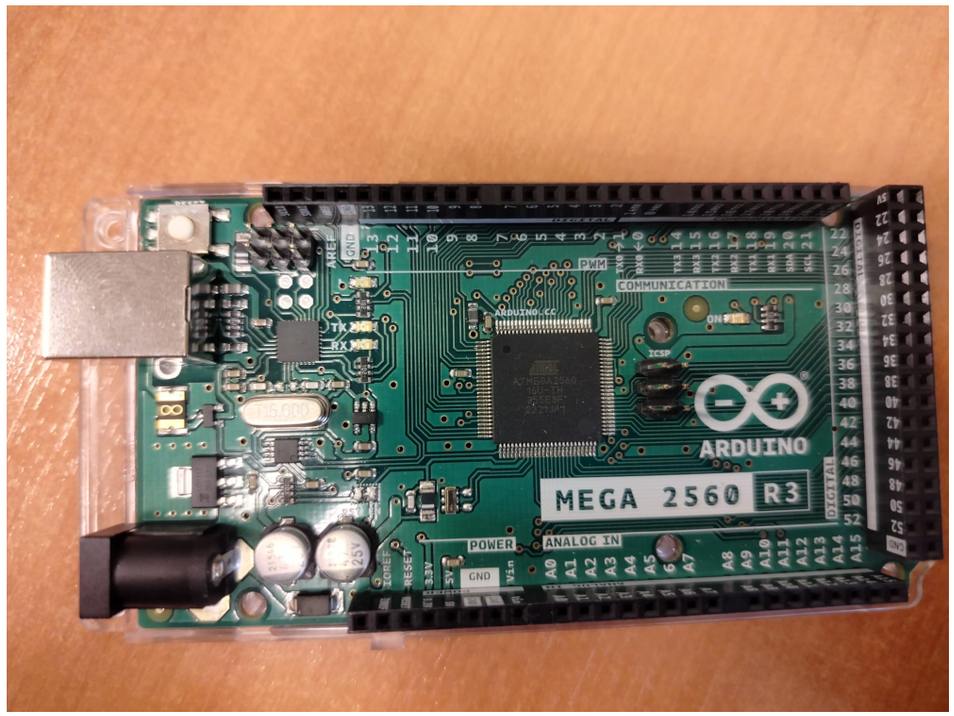
Main components
- Microcontroller
- Analogic and digital I/O pins
- Flash memory
- USB port for serial communication

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

ARDUINO



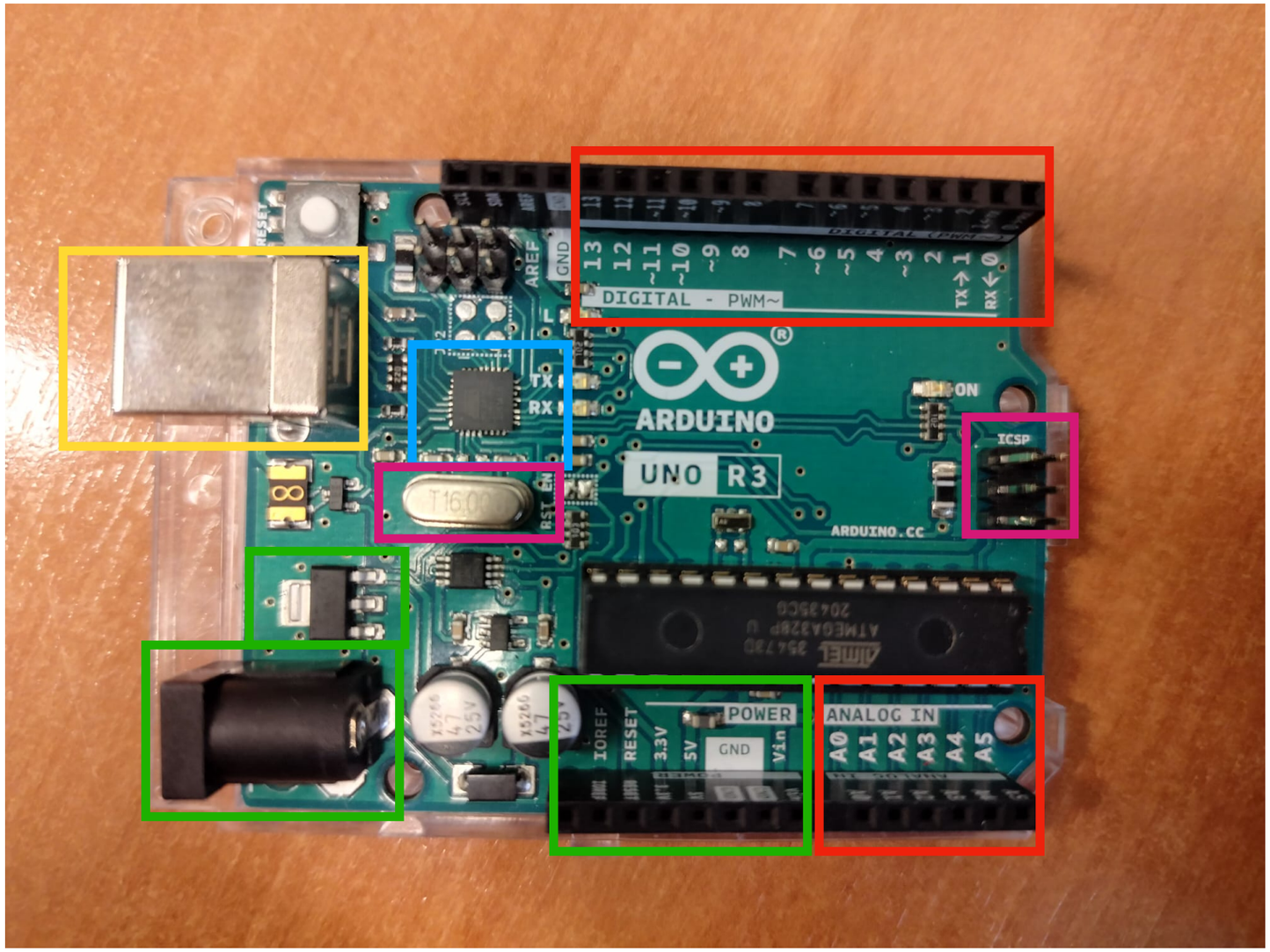
The **Arduino Uno** is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button.



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.

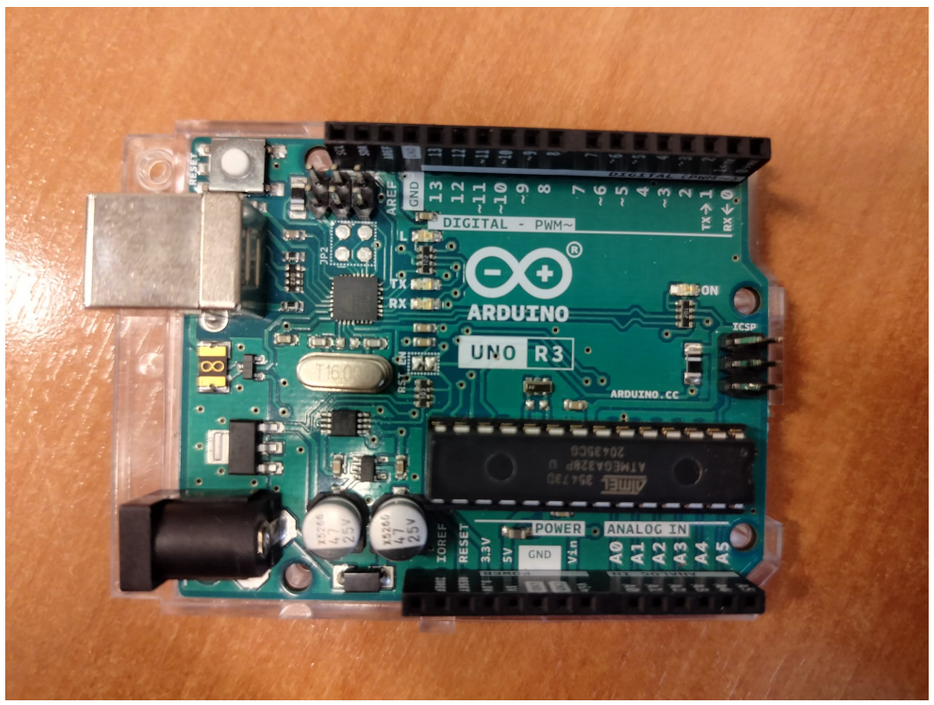
ARDUINO

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++) { ... }

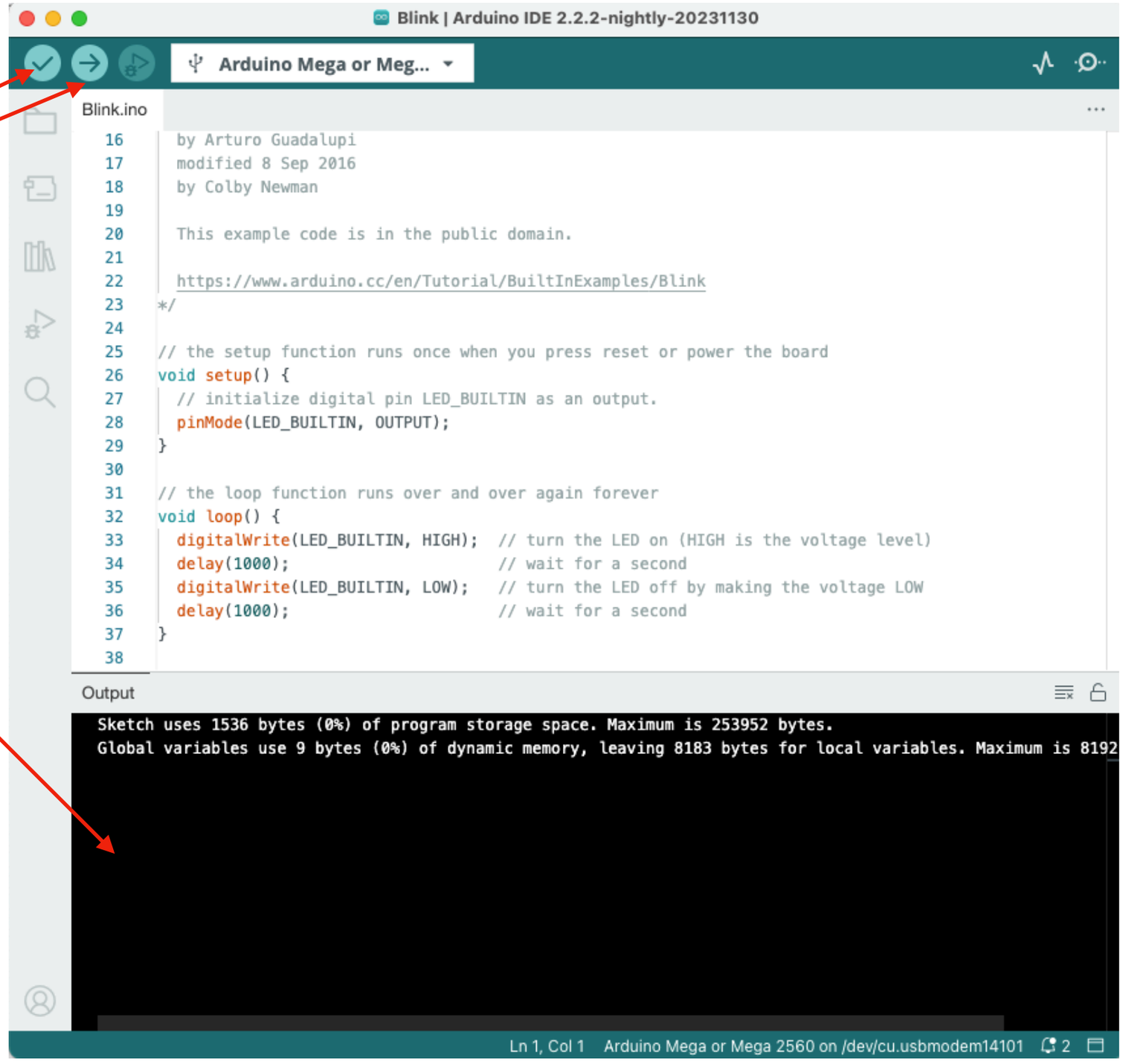
Printing

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

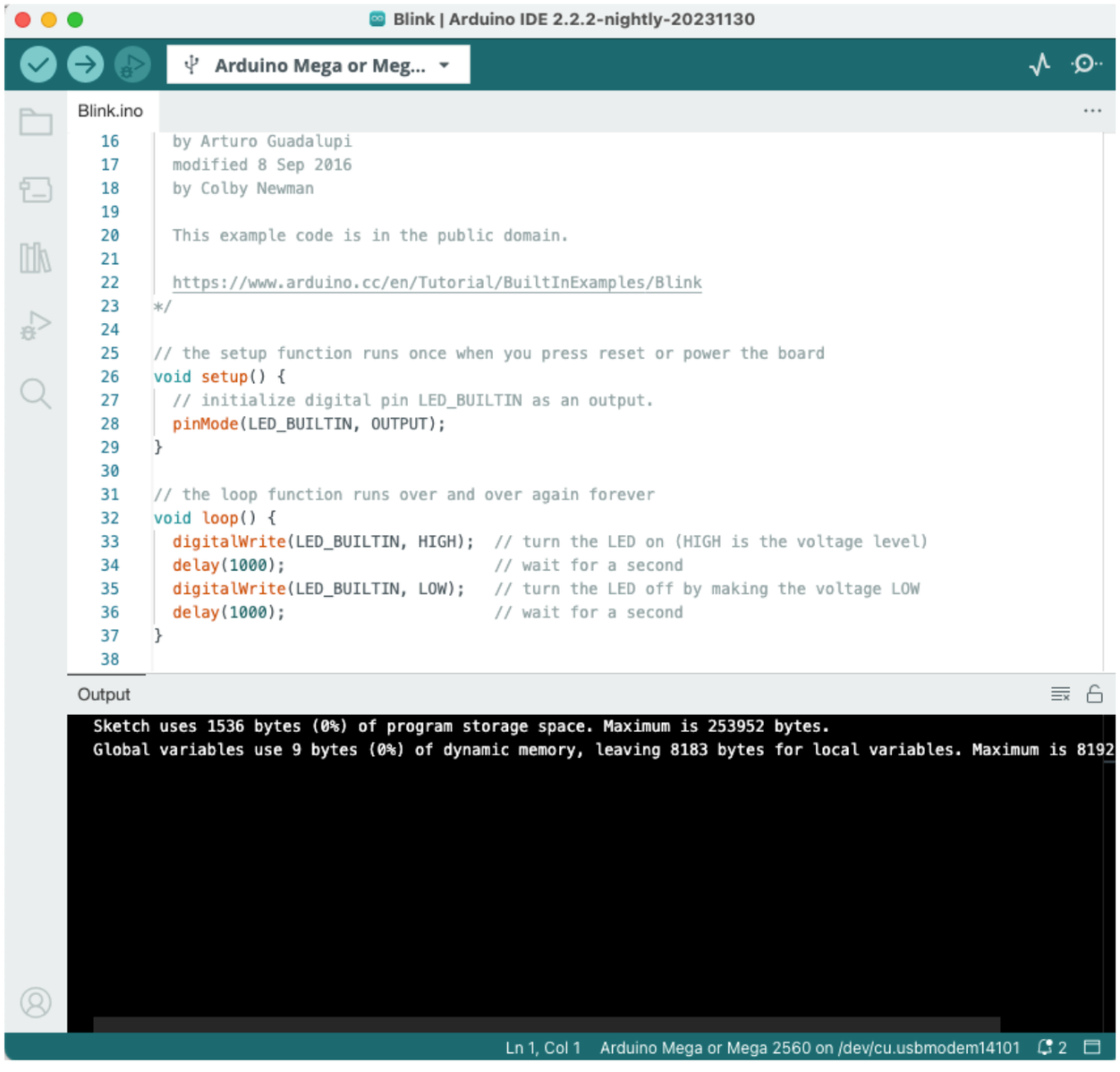
```
24
25  // the setup function runs once when you press reset or power the board
26  void setup() {
27    // initialize digital pin LED_BUILTIN as an output.
28    pinMode(LED_BUILTIN, OUTPUT);
29  }
30
31  // the loop function runs over and over again forever
32  void loop() {
33    digitalWrite(LED_BUILTIN, 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

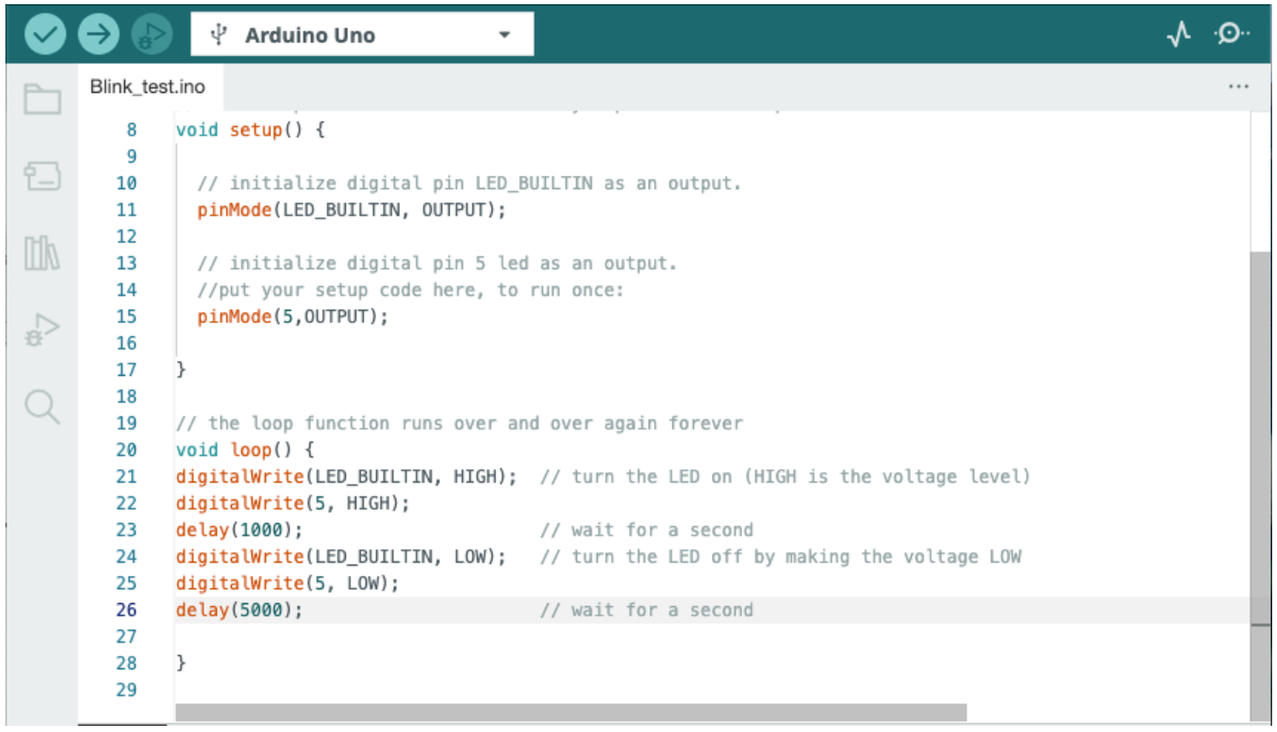
10 ways to destroy Arduino (beware!):

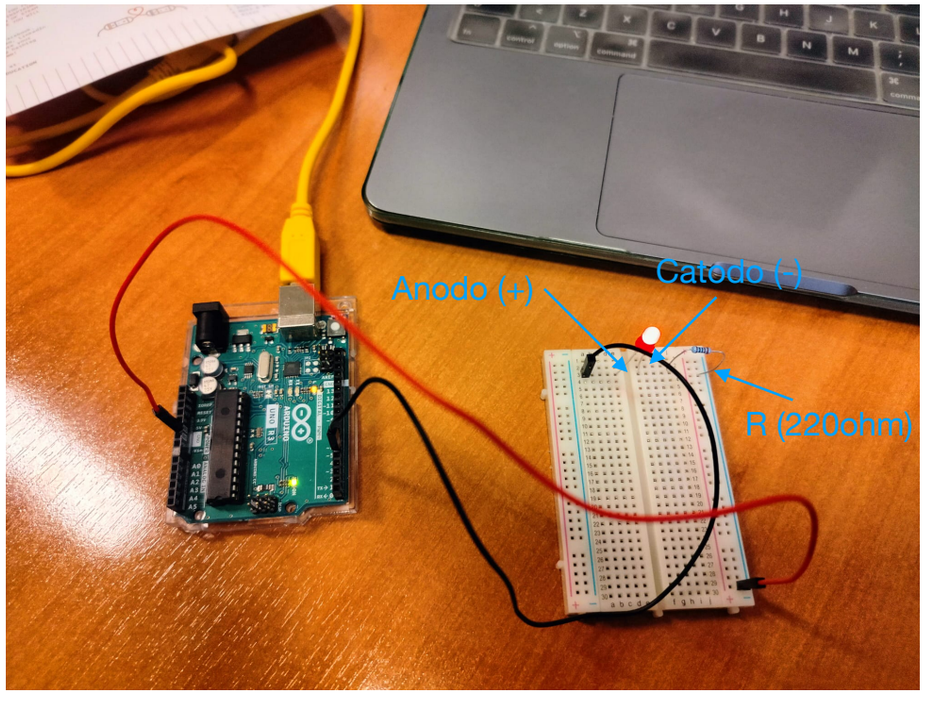
1. Set I/O pins to ground
2. Connect I/O pins together
3. Apply too much voltage on I/O pins
4. Apply voltage on Vin with inverted polarity
5. Apply > 5V on the ‘5V’ pin
6. Apply > 3.3V on the ‘3.3V’ pin
7. Set Vin to ground
8. Apply >13V to the reset
9. Apply voltage to the ‘5V’ pin and charge Vin
10.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

ARDUINO: YOUR FIRST TEST

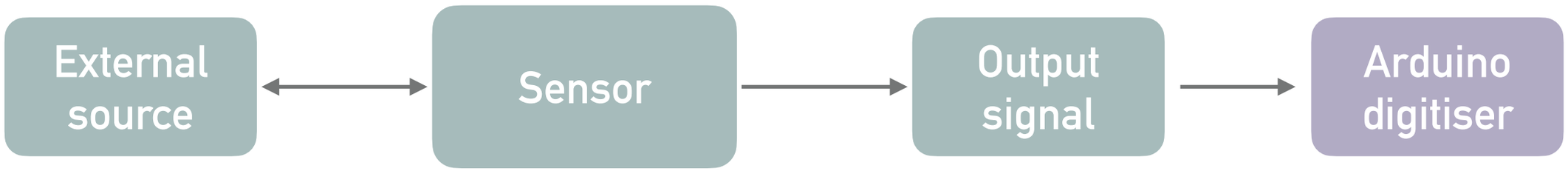
Playing with the blinking LED
Start with the Basics
Example: ‘Blink.ino’

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: MICROPHONE READOUT CHAIN



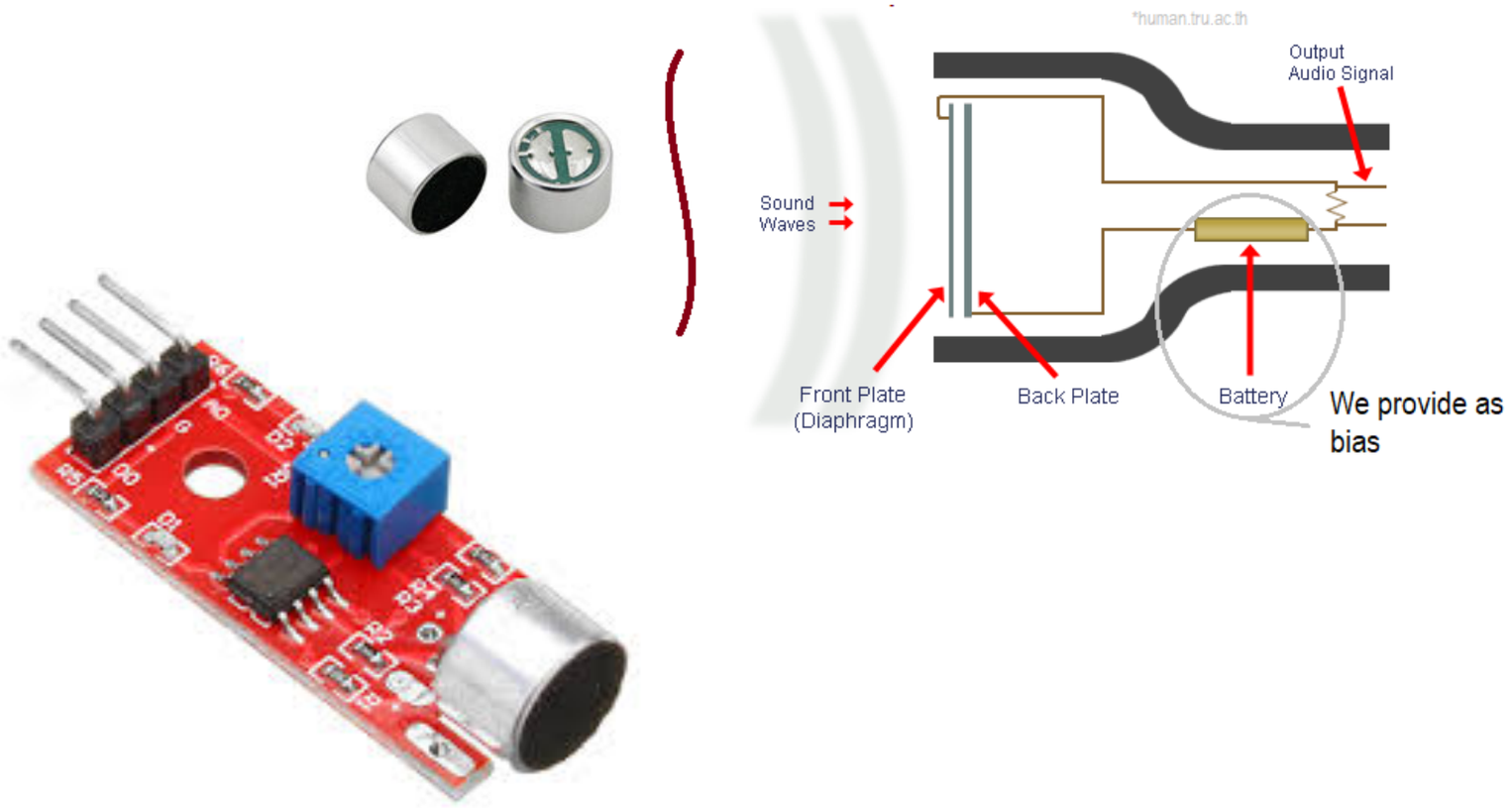
- Source: acoustic sound (voice/background/audio)
- Sensor: microphone
- Read the signal output: Arduino digitiser

Lab.2 (today)

MICROPHONE SENSOR

- Source: acoustic sound (voice/background/audio)
- Sensor: microphone
- Read the signal output: Arduino digitiser to serial port

Electret microphone with breakout board



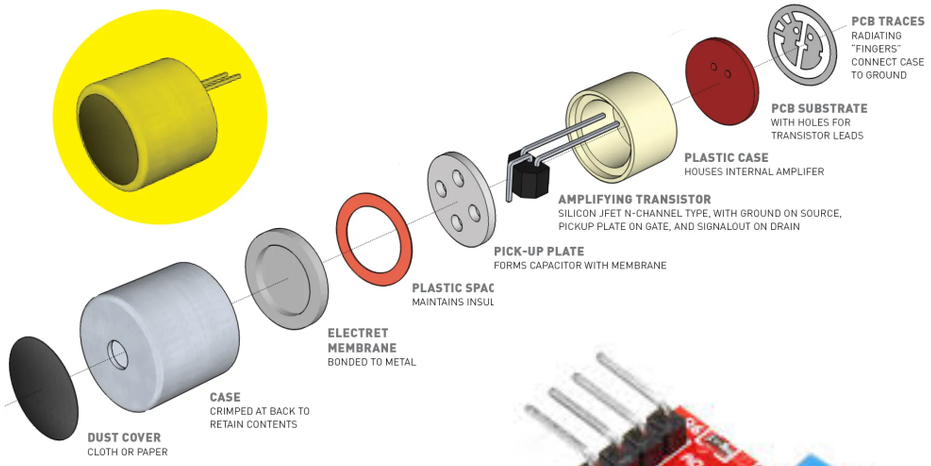
An electret microphone works by using a permanently charged material (the electret) to create an electric field. Sound waves cause a thin diaphragm to vibrate, altering the distance between the diaphragm and a conductive backplate, which changes the capacitance and generates an electrical signal.
This design eliminates the need for an external polarizing voltage, making electret microphones compact and energy-efficient.

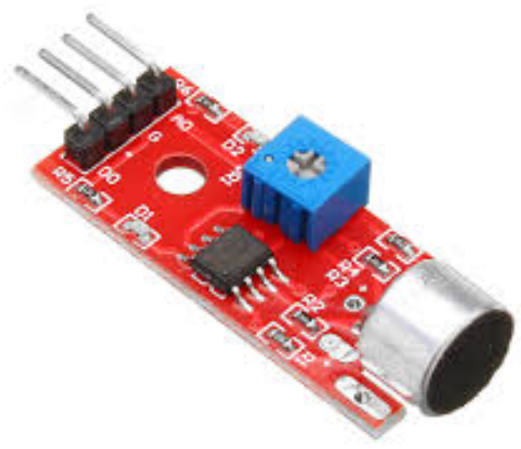
https://en.wikipedia.org/wiki/Electret_microphone

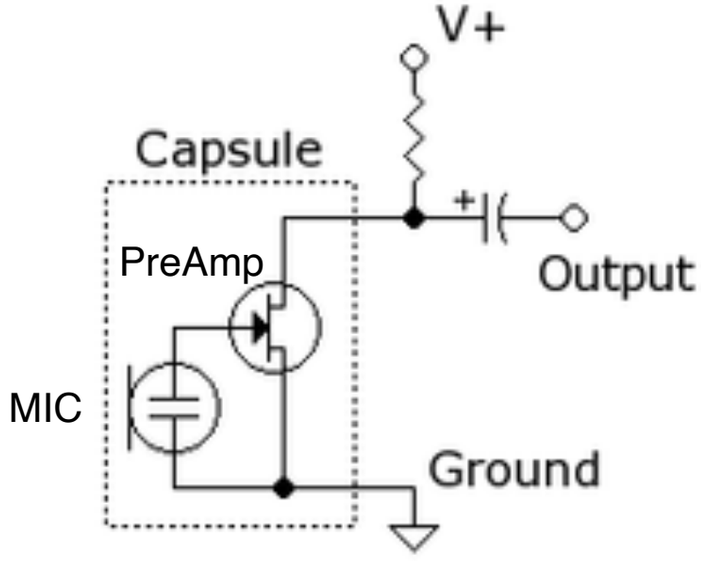
MICROPHONE SENSOR

- Source: acoustic sound (voice/background/audio)
- Sensor: microphone
- Read the signal output: Arduino digitiser to serial port

Electret microphone with breakout board







Pin	Wiring to Arduino
A0	Analog pins
D0	Digital pins
GND	GND
VCC	5V

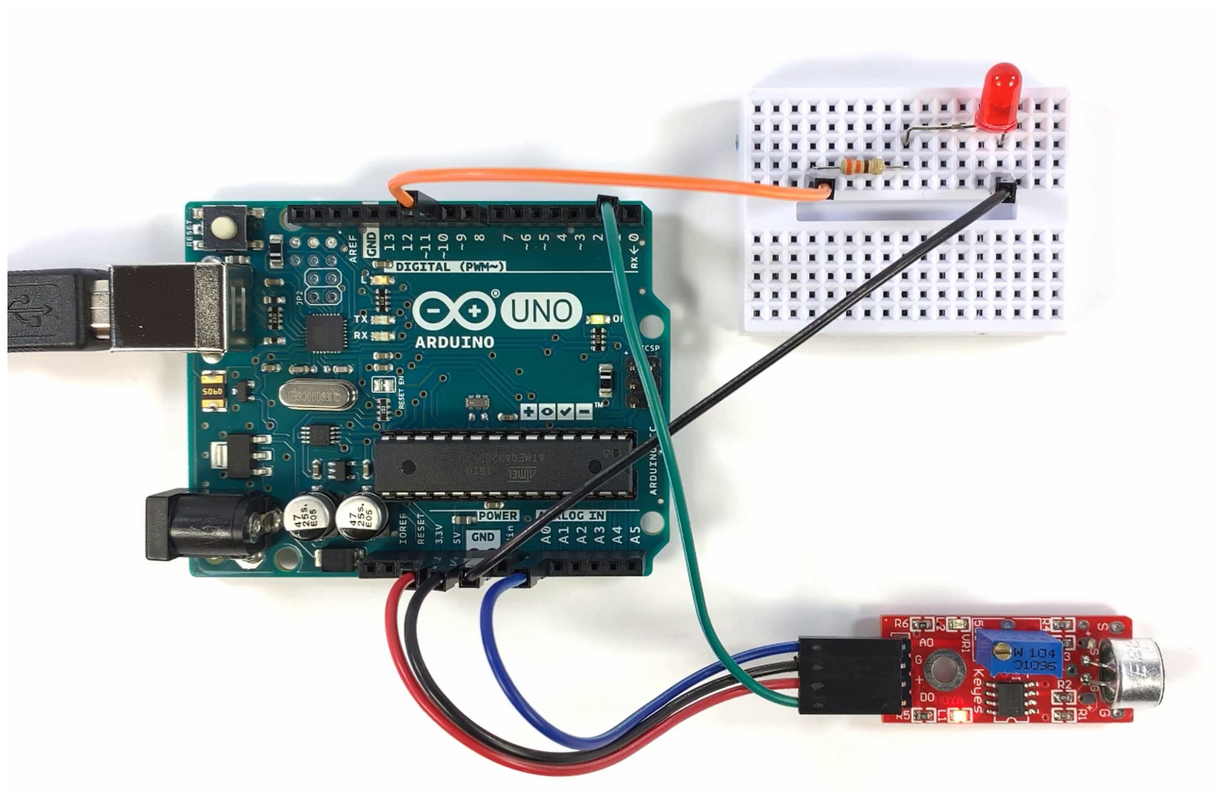
MICROPHONE SENSOR

- Source: acoustic sound (voice/background/audio)
- Sensor: microphone
- Read the signal output: Arduino digitiser to serial port

Reference example:

https://randomnerdtutorials.com/guide-for-microphone-sound-sensor-with-arduino/

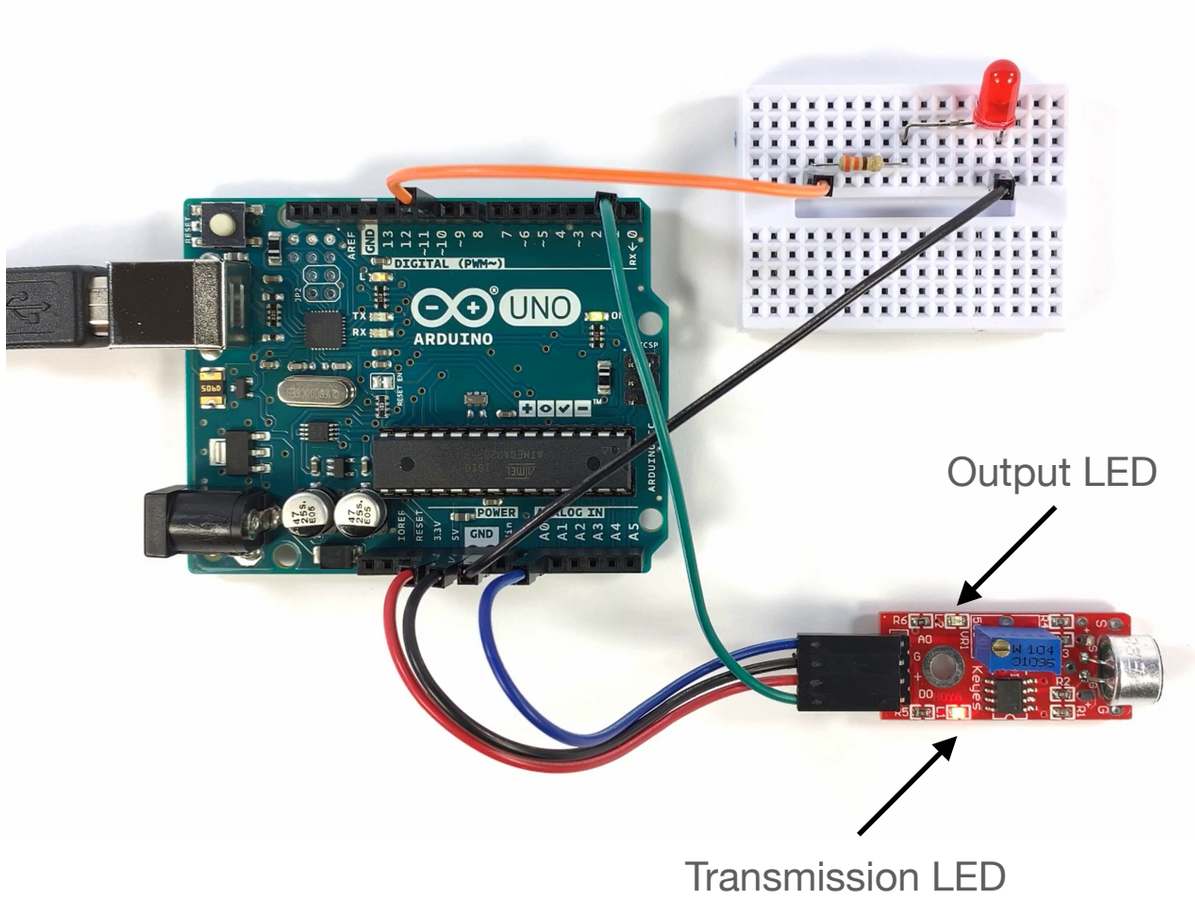
A microphone sensor will detect the sound intensity of your surroundings. A0 is the electrical/analog output from the sensor. The mics breakout board will produce a positive digital output (D0) if the recorded sound intensity is above a certain threshold.



MICROPHONE SENSOR

A microphone sensor will detect the sound intensity of your surroundings. A0 is the electrical/analog output from the sensor. The mics breakout board will produce a positive digital output (D0) if the recorded sound intensity is above a certain threshold.

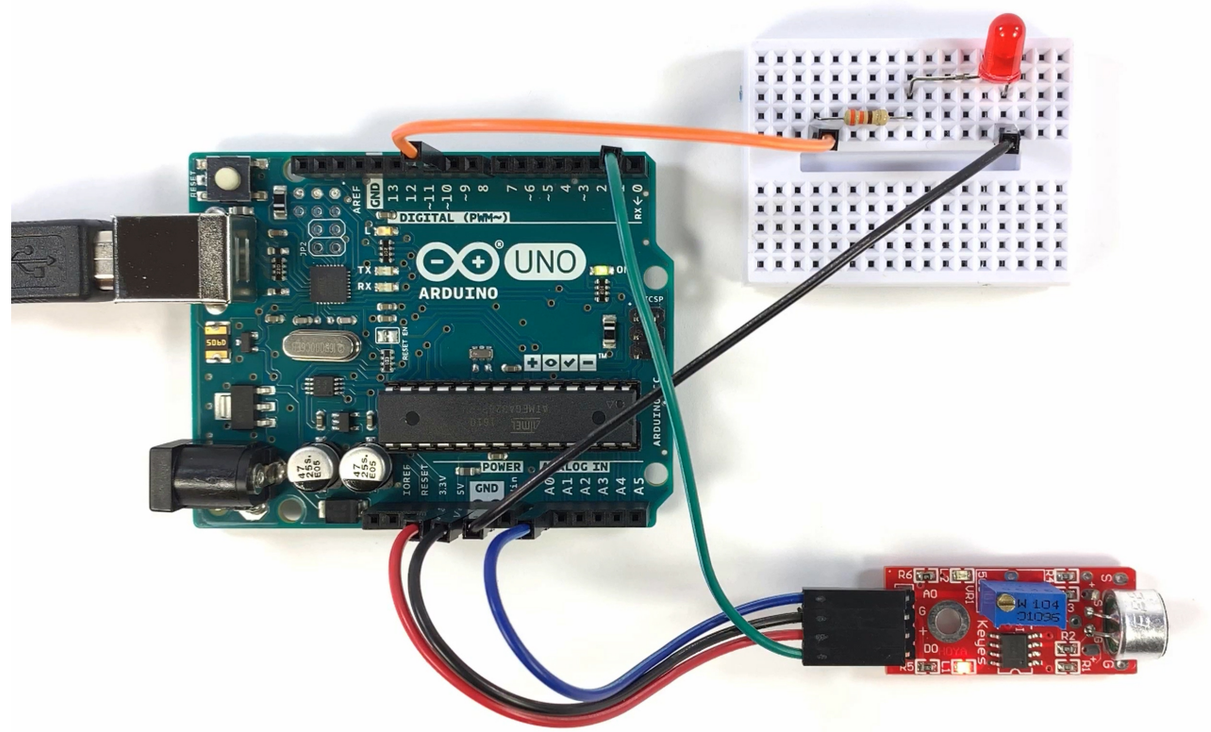
- Connect the Mic and LED to Arduino:
  - Mic-to-Arduino: A0 to analog in pin A0, G to pin GND, + to pin 5V, D0 to digital pin 1
  - LED connected to digital pin 10 (via R) and to pin GND
- Power on the Mic. Connect the USB cable
- **Optimise the Mic sensitivity**. Turn the screw on the voltage trimmer, until the red output LED (LED2) on the breakout boards turns on, with respect to just environmental noise. This means the Mics output is just above the threshold.



MICROPHONE SENSOR: SOUND SENSITIVE LIGHTS

A microphone sensor will detect the sound intensity of your surroundings and will **light up an LED if the sound intensity is above a certain threshold**.

```
////////////////////////////////////////
// Microphone sound sensor     Tutorial //
////////////////////////////////////////
int digitalPin = 1;    // mics digital interface
int analogPin = A0;    // mics analog interface
int ledPin = 10;       // LED pin
int digitalVal;        // digital readings
int analogVal = 0;         // analog readings

void setup()
{
  pinMode(digitalPin,INPUT);
  pinMode(analogPin, INPUT);
  pinMode(ledPin,OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  // Read the analog interface
  analogVal = analogRead(analogPin);
  // Read the digital inteface
  digitalVal = digitalRead(digitalPin);

  // Print analog value to serial
  Serial.print("Analog value: ");
  Serial.println(analogVal);

  if(digitalVal == HIGH)
  {
    digitalWrite(ledPin, HIGH); // Turn ON Arduino's LED
  }
  else
  {
    digitalWrite(ledPin, LOW);  // Turn OFF Arduino's LED
  }
  delay(100);

}
```



- Try with voice audio, hands clap vs music audio
- Try different delays
- Add more LEDs for a more spectacular effect!

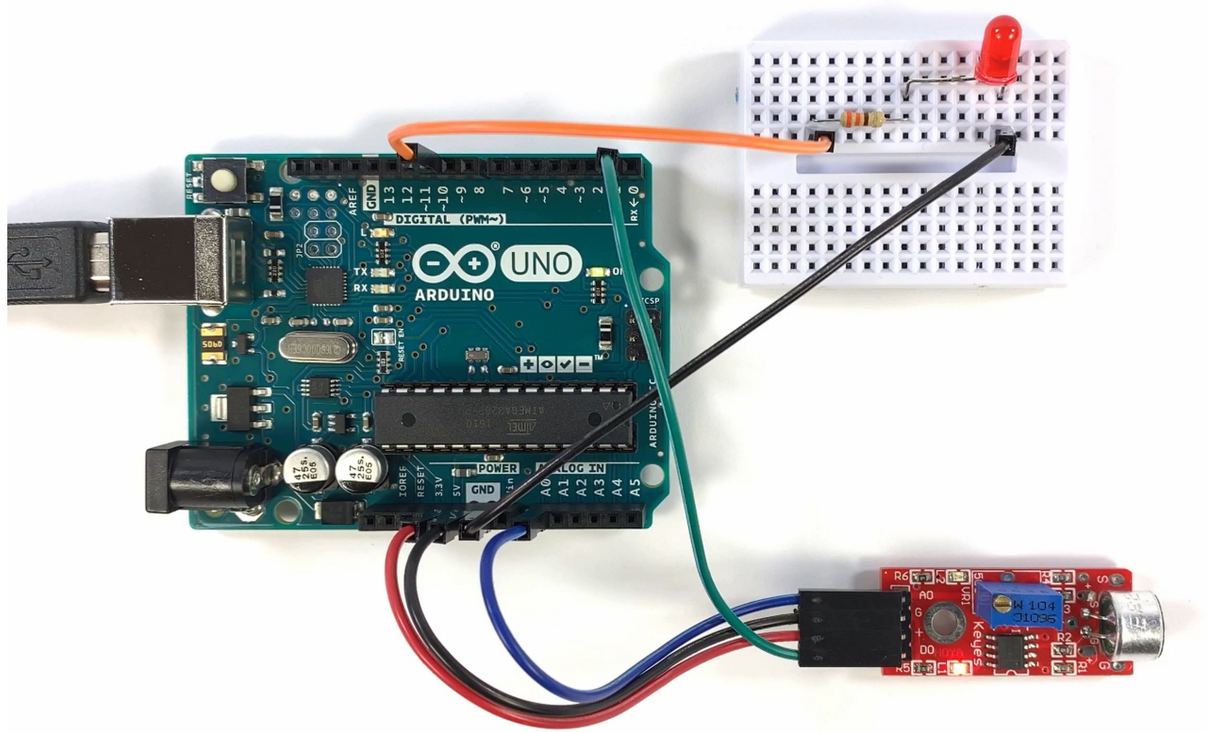
MICROPHONE SENSOR: TIME/FREQUENCY DOMAIN DATA STREAM

A microphone sensor will detect the sound intensity of your surroundings and record the time/frequency domain data stream

Recording the time domain data stream

- Print on ArduinoIDE serial output

```
int digitalPin = 1;   //digital interface
int analogPin = A0;   // analog interface
int ledPin = 10;      // Arduino LED pin
int digitalVal;       // digital readings
int analogValue = 0;  // analog readings

void setup()
{
  pinMode(digitalPin,INPUT);
  pinMode(analogPin, INPUT);
  pinMode(ledPin,OUTPUT);
  Serial.begin(9600);
}

void loop() {

  Serial.println("New audio sampling");

  static unsigned long previousTime = 0; //
  unsigned long currentTime = millis(); // Current time in ms

  int analogValue = analogRead(analogPin); // Analog reading

  unsigned long samplingTime = currentTime - previousTime; // Time between two samplings
  previousTime = currentTime; // Update time

  Serial.print("Analog Value: ");
  Serial.print(analogValue);
  Serial.print(" | Time (ms): ");
  Serial.println(currentTime);
  //Serial.print(" | Sampling Time (ms): ");
  //Serial.println(samplingtTime);

  delay(100); // small delay (ms) bewtween lectures

}
```



- How to save analog output on txt?
- Plot the audio graph for multiple samplings

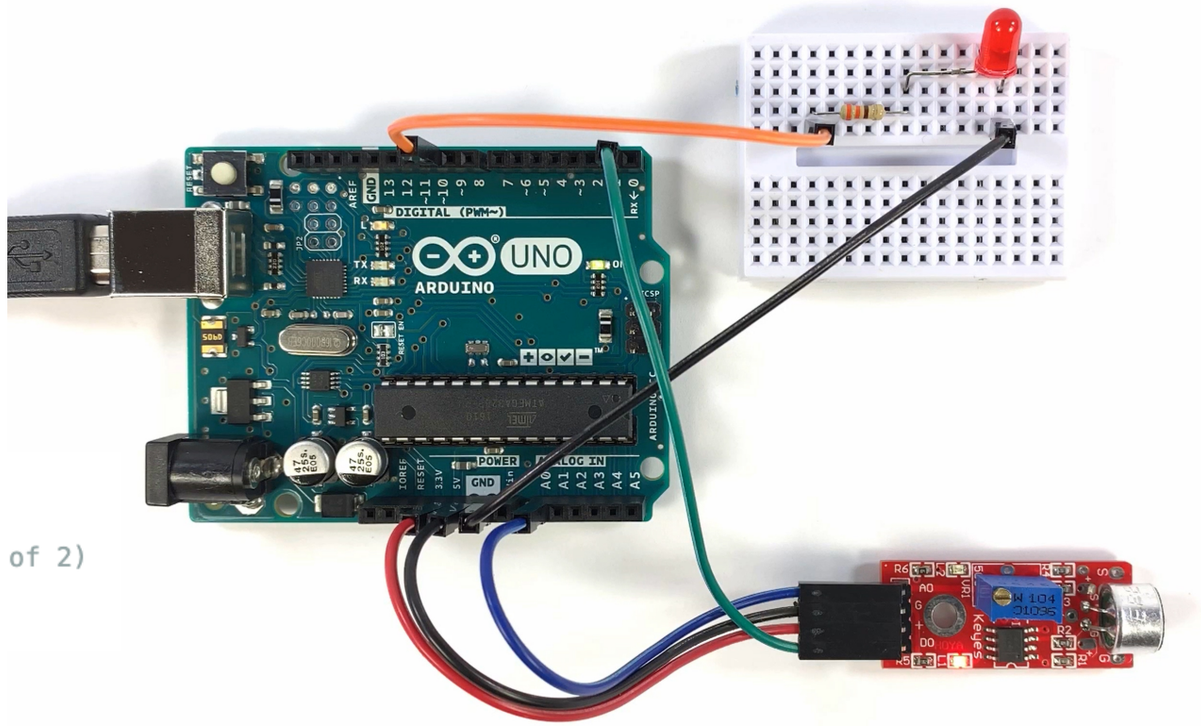
MICROPHONE SENSOR: TIME/FREQUENCY DOMAIN DATA STREAM

A microphone sensor will detect the sound intensity of your surroundings and record the time/frequency domain data stream

Recording the FFT of the data stream

- Install Arduino FFT library
- Set fixed fSampl and time interval:

fSampl 1kHz; nSamples 128




```
#include "arduinoFFT.h"

#define SAMPLES 128             // Number of samples (must be a power of 2)
#define SAMPLING_FREQUENCY 1000 // Sampling frequency in Hz
```

- Collect and print on serial the FFT of given time intervals

MICROPHONE SENSOR: TIME/FREQUENCY DOMAIN DATA STREAM

A microphone sensor will detect the sound intensity of your surroundings and record the time/frequency domain data stream

Recording the FFT of the data stream

- Install Arduino FFT library
- Set fixed fSampl and time interval: fSampl 1kHz; nSamples 128
- Collect and print on serial the FFT of given time intervals

```
void loop()
{
  Serial.println("New audio sampling");

  unsigned long previousMicros = micros();
  for (int i = 0; i < SAMPLES; i++) {
   while (micros() - previousMicros < (1000000 / SAMPLING_FREQUENCY)) {
     // Wait for the next sampling interval
   }
   previousMicros = micros(); // Update the previous time
   vReal[i] = analogRead(analogPin);
   vImag[i] = 0;              // Imaginary part is zero
   //delayMicroseconds(1000000 / SAMPLING_FREQUENCY); // Delay to maintain sampling rate
  }

  // Perform FFT
  FFT.windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  FFT.compute(vReal, vImag, SAMPLES, FFT_FORWARD);
  //FFT.complexToMagnitude(vReal, vImag, SAMPLES);
  Serial.print("Frequency ");
  Serial.print(" | Amplitude ");
  Serial.println(" | Phase ");

  // Print results
  for (int i = 0; i < (SAMPLES / 2)+1; i++) {
   double frequency = (i * (double)SAMPLING_FREQUENCY) / (double)SAMPLES; // Frequency bin
   double amplitude = sqrt(vReal[i] * vReal[i] + vImag[i] * vImag[i]); // Amplitude
   double phase = atan2(vImag[i], vReal[i]) * 180 / PI; // Phase in degrees
   Serial.print(frequency);
   Serial.print(" ");
   Serial.print(amplitude);
   Serial.print(" ");
   Serial.println(phase);
  }

  delay(10000); // Wait before the next analysis
}
```

MICROPHONE SENSOR: TIME/FREQUENCY DOMAIN DATA STREAM

A microphone sensor will detect the sound intensity of your surroundings and record the time/frequency domain data stream

Recording the FFT of the data stream

- Install Arduino FFT library
- Set fixed fSampl and time interval: fSampl 1kHz; nSamples 128
- Collect and print on serial the FFT of given time intervals

- Analyze the FFT output on serial at the different frequencies exposing the mic to different sources (different samplings):
  - Environmental noise
  - Voice (pronounce a vowel)
  - Audio - ‘LA’ 440Hz sound
  - Frequency sweep
- How to save FFT output on txt?
- Plot the audio PSD (IFFTI^2) graph for multiple samplings of the same source (and/or different sources)