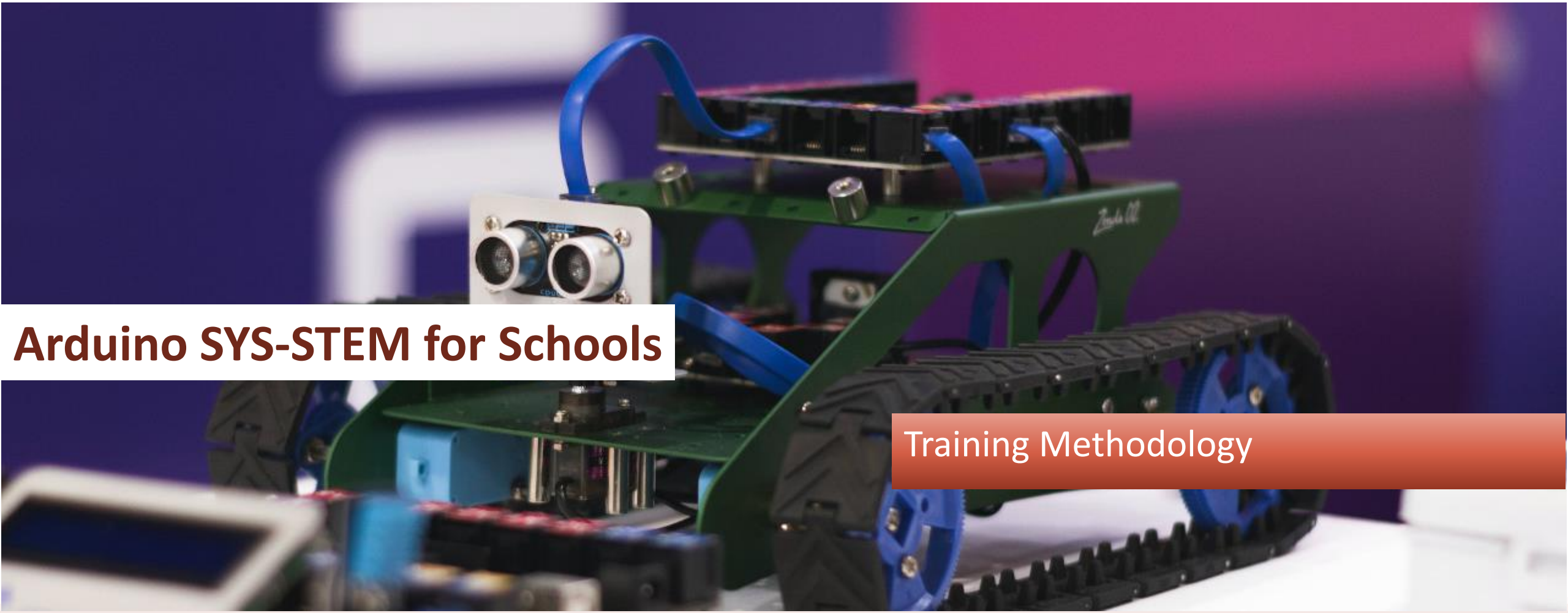


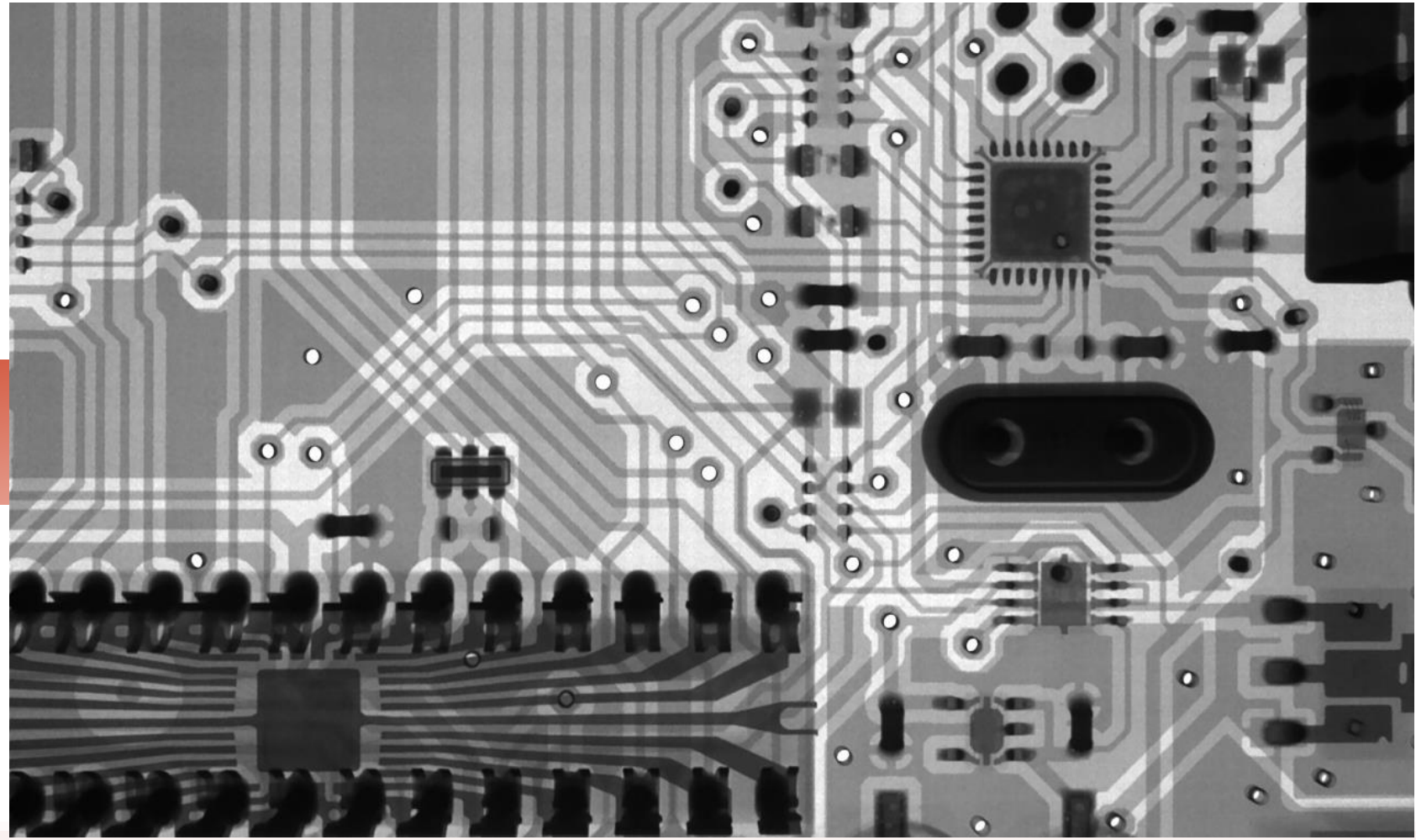
Arduino SYS-STEM for Schools

Training Methodology

SYS-STEM – Arduino SYS-STEM for Schools
Erasmus + Key Action 2 Strategic partnership - 2019-1-ES01-KA201-064454

MODULE 2

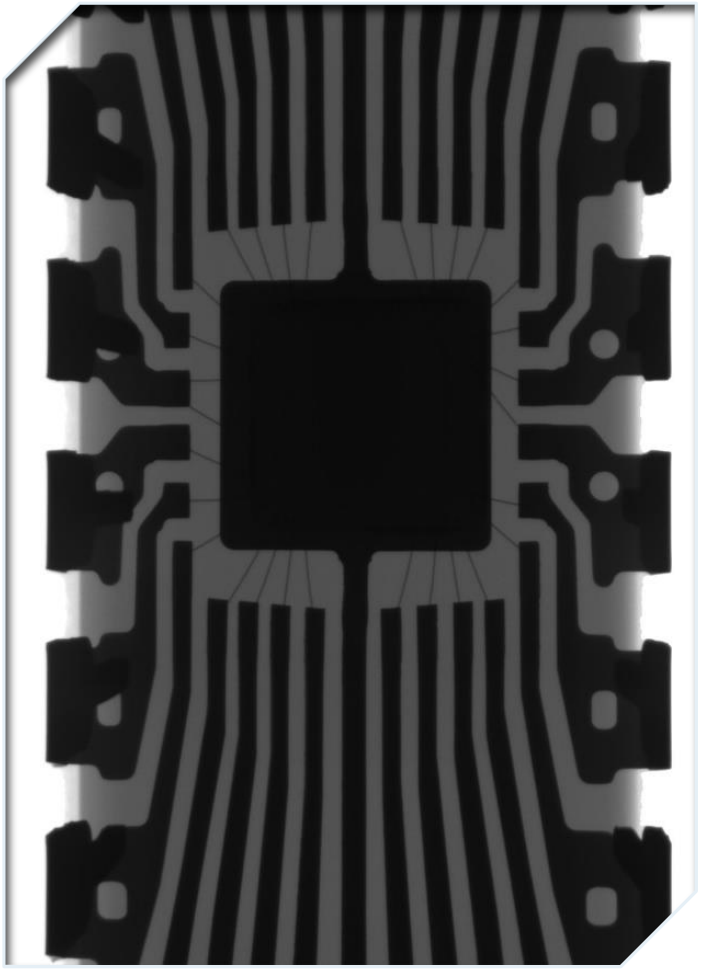
INPUTS, OUTPUTS AND INTERRUPTIONS



SYS-STEM – Arduino SYS-STEM for Schools
Erasmus + Key Action 2 Strategic partnership - 2019-1-ES01-KA201-064454

TRAINING MODULE CONTENTS

- Objective
- Learning Outcomes
- Unit 1 -
- Unit 2 -
- Additional reading materials
- Exercises/tests



OBJECTIVE

This module will teach the learner what the differences between digital and analog devices are, how to handle, connect and make use of them and what is the aim of interruptions and the way to implement them in a project.

Besides, the learner will also adquire knowledge about how and where to connect analog devices to Arduino.

As well, you will learn some rules and functions to be able to interact with those analog devices.

Knowledge

Competences and Skills

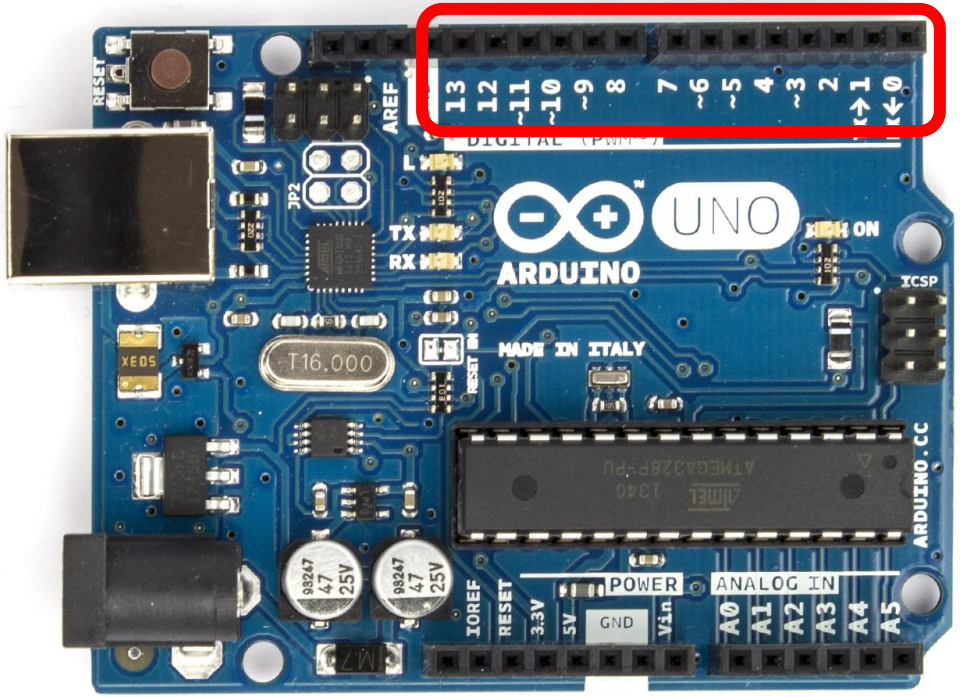
EXPECTED LEARNING OUTCOMES

UNIT 1

INTRODUCTION

Here is the introduction to some concepts that we will develop in this module

- Input
  - Any device that will collect information from the outside and provide it to the Arduino. Examples of inputs are a switch or a sensor.
- Output
  - Any device that will provide information of any kind to an external observator. This can be a LED that turns on, a buzzer that makes a sound or any other that makes some kind of indication.
- Interruption
  - This is a tool that Arduino features and is useful to carry out some special actions. This will help you to develop high priority for your projects. Some examples will be useful to help you understand the way interruptions work end their application.
- Digital technology
  - This concept means that for any given device or variable only two different values are posible. The easiest way to understand is a switch, that is crearly digital because it only accepts two values: pushed or released. Another device that can have digital behaviour is a LED: on and off, no more.



DIGITAL INPUTS AND OUTPUTS

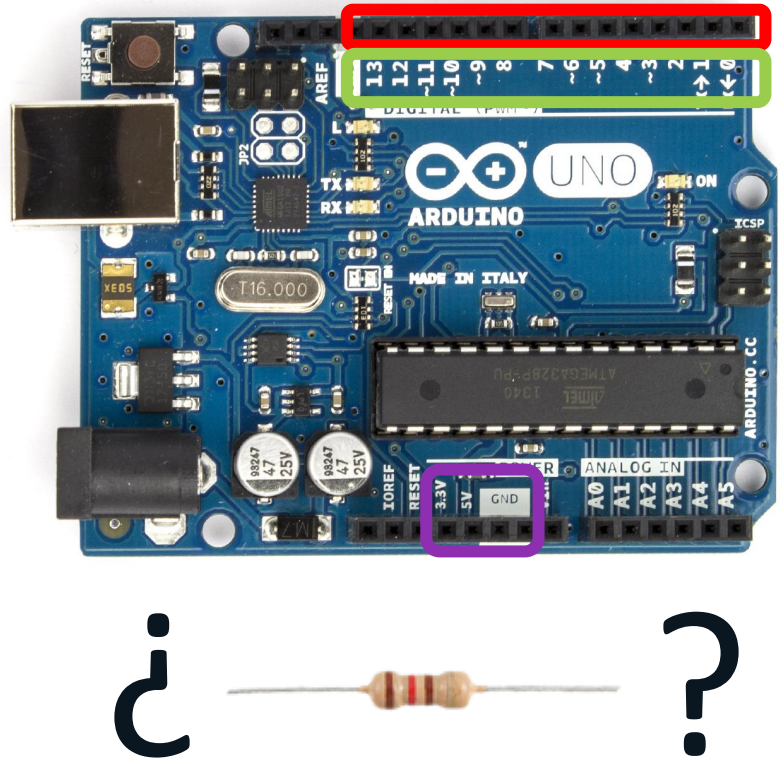
Digital inputs and outputs in Arduino must be connected to the highlighted pins, numbered from 0 to 13.

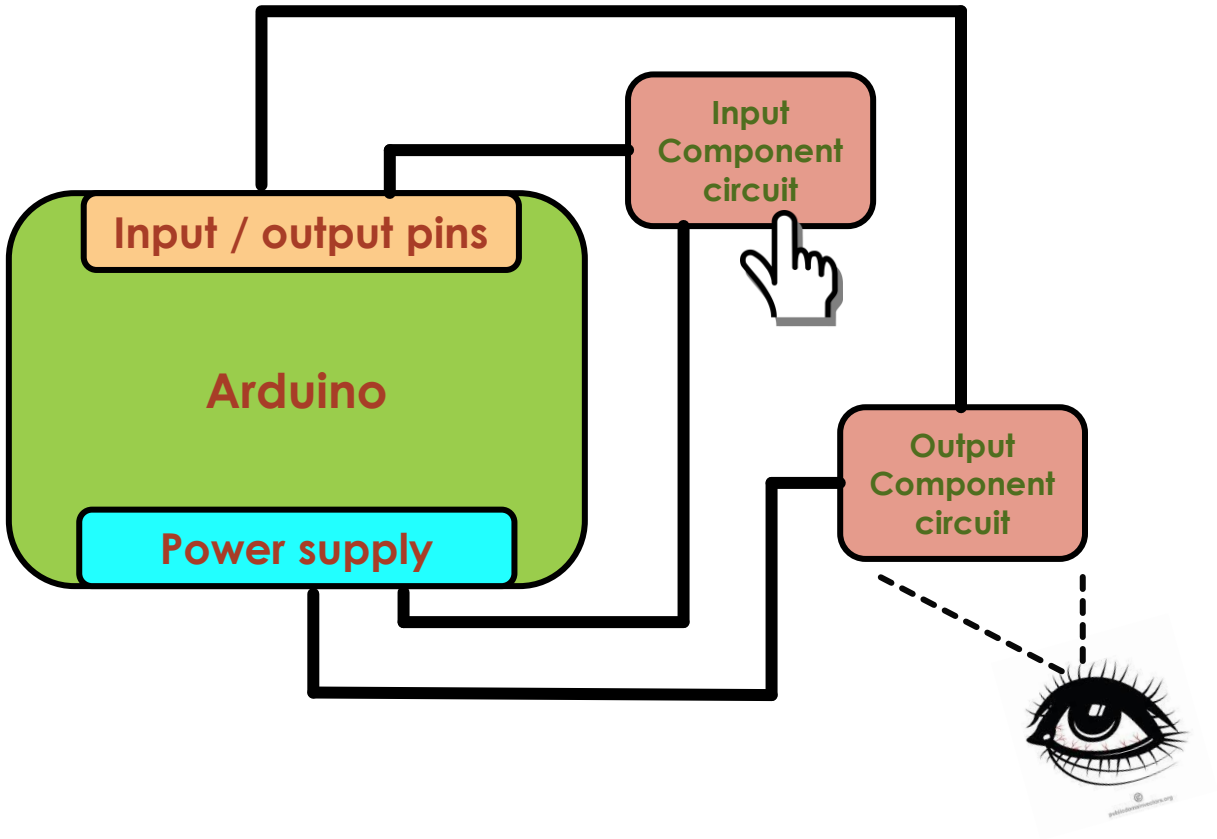
The number of the pin is extremely important, as you will have to use it in the sketch to reference the devices you connect to Arduino.

CONNECTION OF COMPONENTS

There are, for Arduino, components of variated nature depending on the functionality. In this unit you will learn how to connect the very basic inputs and outputs, and to this end the fundamentals are the following:

- Usually a two terminal device will need connection to its input/output pin and to the power supply.
- Sometimes, you will have to add a resistor to achieve a good behaviour of the devices.
- You will have to use the number of the pin to which you connect the device in the sketch





PROPERTIES OF COMPONENTS

The aim of the digital inputs and outputs is to allow the connected devices to swap the voltaje supply they receive between 0 volts and 5 volts.

The swap will allow the circuit to detect the devices as "ON" or "OFF".

The most basic inputs are switches but there is a wide range of sensors that behave as inputs as well.

Among the outputs the most basic are LEDs and buzzers but there are other ones such as LCD screen, servos…

INTEGRATING INPUTS AND OUTPUS

Define the pin Inputs and outputs must be integrated into your sketches, this means that you have somehow to tell Arduino **what kind of device** (input or output) you are going to use and **where** (the I/O pin) you will connect it.

There are, therefore, two tasks

1. Define the pin number where you will connect the I/O device with a code line:

```
const int <name_of_device> = <pin_number>;
```

2. Let Arduino know the pin you are going to use and its function: input or output

```
pinMode(<name_of_device>, <kind_of_device>);
```

<kind_of_device> can be two different values: INPUT or OUTPUT, depending on the function of the component

INTERACTING WITH INPUTS AND OUTPUS

You will use inputs to provide information to Arduino be it through human action or through sensors and outputs to generate visible responses through components. To this aim, when the components you are handling are digital you will use two functions:

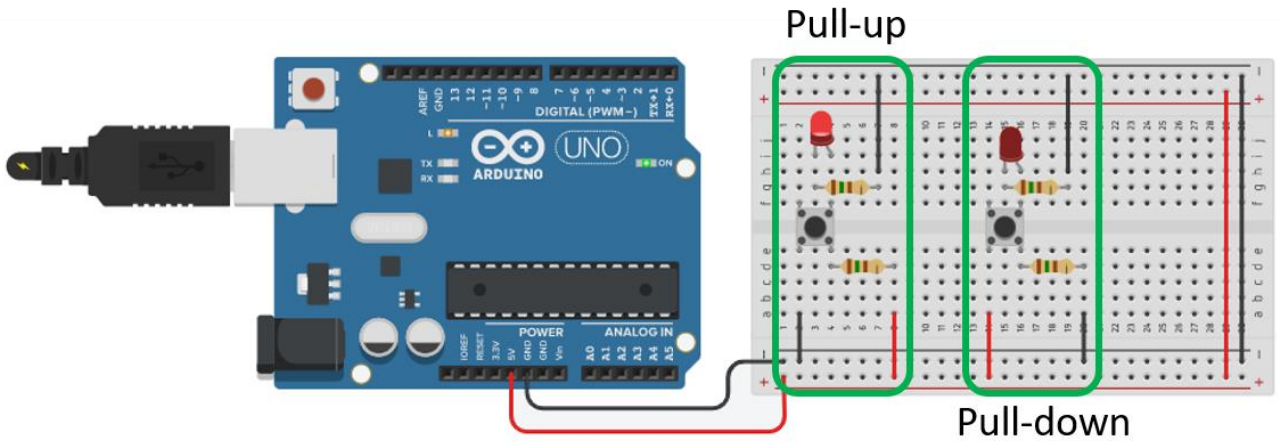
```
digitalRead(<pin>);
```

This function will read the status of a pin and return if it is set at a high level (1 digital value or *True*) or if on the other hand it is at a low level (0 digital value or *False*). The `<pin>` can be a raw number or the name of a variable that contains a number

```
digitalWrite(<pin>, <value>);
```

This function writes `<value>` into the component connected to Arduino through the pin whose number is `<pin>`. Take into account that `<pin>` can be here as well a raw number or the name of a variable that contains a number

PRACTICAL EXAMPLE: PULL-UP / PULL-DOWN



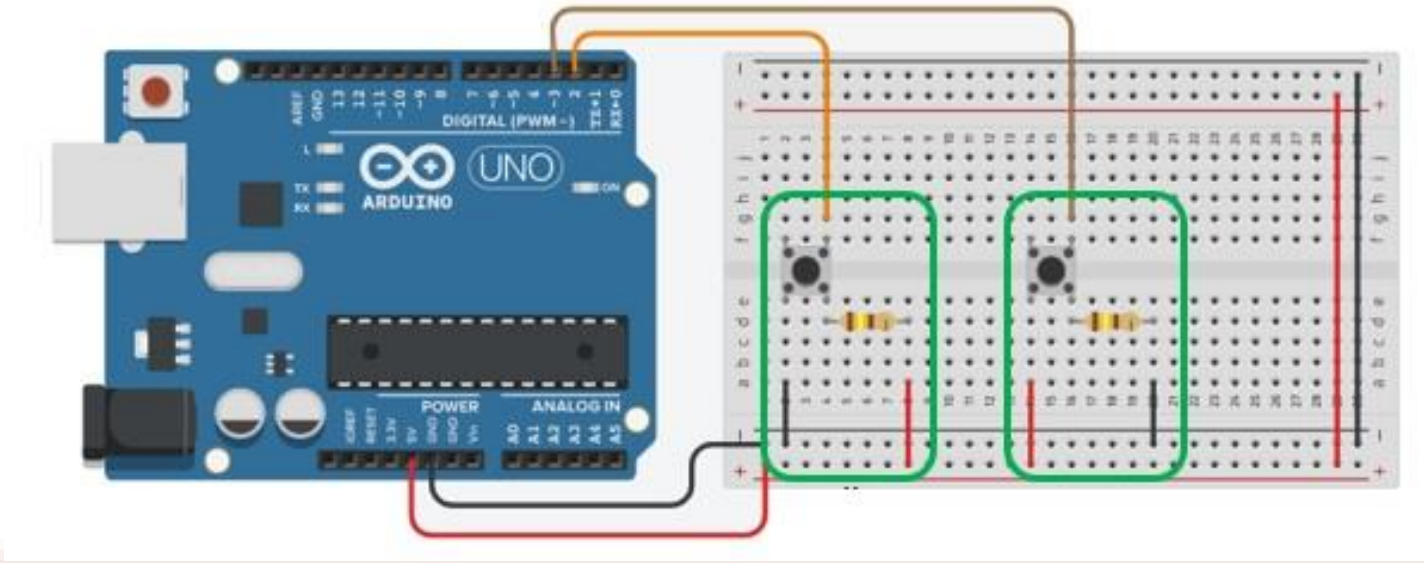
Go to example

You can use inputs such as buttons in two diferente ways

1. **With pull-up resistor**: the input pin will be at a high level if the button is not pushed and at a low level if the button is pushed
2. **With pull-down resistor**: the input pin will be at a low level if the button is not pushed and at a high level if the button is pushed

SWITCHES AS INPUTS

Taking into account the previous circuit, you are able now to create a circuit carrying the buttons to a digital input pin, this way you are ready to add them to a sketch



INTERRUPTIONS

Interruptions are useful when you want any given action to trigger a high priority function or another action before going on with the normal flow of the sketch.

You will have to program an interruption to take this to a good end.

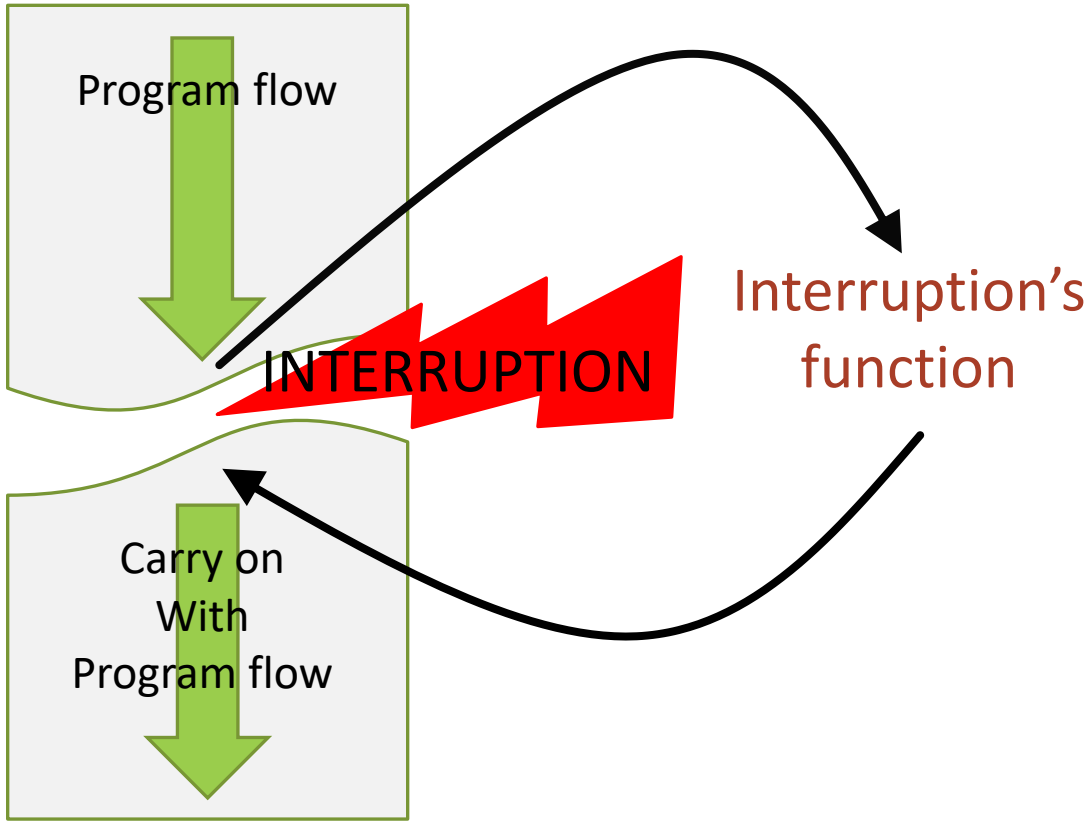
Interruptions are generated by some given device and the amount of interruption devices in Arduino is limited by the model of Arduino you use.

To implement an interruption correctly, it is necessary a specific assembly along with a particular programming structure to define it.

Interruption pins

Model	Int0	Int1	Int2	Int3	Int4	Int5
UNO	2	3				
Nano	2	3				
Mini Pro	2	3				
Mega	2	3	21	20	19	18
Leonardo	3	2	0	1	7	
Due	Any pin					

Interruption implementation



INTERRUPTION PINS AND PROGRAM FLOW

INTERRUPTIONS: IMPLEMENTATION

Implementing interruption requires some specific code in the sketch.

To define you interruption you will use two different template lines:

```
attachinterrupt(<interrupt>, <ISR>, <mode>);
```

```
attachinterrupt(digitalPinToInterrupt(<pin>), <ISR>, <mode>);
```

`<interrupt>`: the interruption number (not pin)

`digitalPinToInterrupt(<pin>)`: the pin in which the interruption happens

`<ISR>`: the function you call if the interruption is triggered

`<mode>`: the nature of the evento that triggers the interruption, there are 4 options

RISING: the interruption is triggered when the pin detects a rising edge, the status of the pin changes from 0 to 1.

FALLING: the interruption is triggered when the pin detects a falling edge, the status of the pin changes from 1 to 0.

CHANGING: the interruption is triggered whenever the pin detects a falling or rising edge.

LOW: the interruption is executed in loop while the pin receives low level signal .

UNIT 2

INTRODUCTION: ANALOGIC TECHNOLOGY

- This technology references those electric signals tan can take any given value inside an interval.
- The best way to understand is to compare analogic technology and digital technology.
  - Digital technology allows just two values, be them on and off, true and false, 0 and 1 or whatever expresions you want to assing
  - Analogic technology allows an endless amount of values.
- Let’s say you have a 5 volts electric signal, then a digital technology will make difference just between 0 volts and 5 volts while analogic technology can take any value between those two, like 2.4 volts or 3.456 volts.

ANALOG INPUTS

Arduino has pins dedicated exclusively to analog input devices. They are named as A0, A1, A2…

Tt those pins you will have to make the connection of analog devices.

Analog devices are needed to measure several magnitudes and among others the learner will find:

- Temperature measuring devices
- Potenciometers
- Humidity
- Gas detector
- …

PULSE WIDE MODULATION (PWM) OUTPUTS

This Pulse Wide Modulation (PWM) technology allows to get intermediate values between Arduino voltaje Interval (0v and 5v)

This permits to get multiple output values from digital pins as they were analog outputs, a feature needed for some devices.

Outputs that can operate in PWM mode are highlighted on the picture, they are I/O pins with ~ symbol by them (pins 3, 5, 6, 9, 10 and 11)

DECLARING ANALOG INPUTS

Integrating an analog component in Arduino just requires to declare it, defining which analog pin you will use to connect the analog device and asign some meaningful name to it.

```
<device_name> = <pin_number>;
```

Let's say, as an example that the learner wants to use a potenciometer in his or her design. As `<device_name>` it is useful to write something meaningful, to make you know what you are handling with that variable.

As `<pin_number>` you will have to use some number between 0 and 5 (number of analog inputs)

This example could generate the following declaration line: ***potenciometer = 0;***

READING FROM ANALOG INPUTS

Analog inputs will provide information to Arduino and that information must be read, to do so you will use the following function

```
<read_value> = analogRead(<device_name>);
```

This will store the read information into `<read_value>` variable. Following with the previous potenciometer example, the use of this function could be: ***data = analogRead(potenciometer)***.

Note that ***potenciometer*** refers to a device connected to the analog input pin 0. The name ***potenciometer*** is just a name you give, but the meaning is that the variable ***potenciometer's*** value is 0, the previous line could be ***data = analogRead(0)*** but the understanding of the code would not be as clear as if you use a name such as ***potenciometer***.

WRITING ON ANALOG OUTPUTS

Writing an analog value requires to indicate a device to write on and the value to be written, with the following function:

```
analogWrite(<device_name>, <value>);
```

What this function will do is to write (store) a given `<value>` in a device that is connected to the pin defined by `<device_name>`.

Note that this time the device will be connected to an I/O pint, not to an analog input.

An example of this could be a LED, which would shine dimly or strongly depending on the value we write on it. Let's say that a ***LED =3;*** variable has been declared (remember that output components that demand PWM must be placed in pins 3, 5, 6, 9, 10 or 11)

If PWM allows values between 0 and 255 an example could be: ***analogWrite(LED, 180);***

MAP() FUNCTION

- Useful function when handling analog values.
- Analog inputs register values in the Interval 0 – 1023 (10 bits), on the other hand, PWM pins resolution is 8 bits deep.
- This means that often the value detected by an analog input must be converted to another scale to adapt it to the analog output device that will make use of the detected value
- Map function's prototype is:

```
map(<value>, <original_min>, <original_min>, <new_min>, <new_max>);
```

- `<value>` is the value to be scalated, `<original_min>` and `<original_max>` are the original edge values of `<value>`, `<new_min>` and `<new_max>` are the edge values of the rescaled value.

EXTRA READING MATERIALS

Infographics

Website, links, etc.

EXTRA READING MATERIALS

Studies, e-books

Videos

EXTRA READING MATERIALS

EXERCISES / TESTS / QUIZZES

TESTS

Question test to self asses the adquired knowledge

TEST

EXERCISE 1

Design a circuit, assemble and sketch, that fulfils the following requirements: a button with a **pull-down** resistor and a LED. The programming will allow you to turn the LED on and off by pushing the button. For the time being, the aim of this exercise is to get familiar with inputs and outputs and the way they are assembled

SOLUTION

EXERCISE 2

Design a circuit, assembly and sketch, that fulfils the following requirements: a button with a **pull-up** resistor and a LED. The programming will allow you to turn the LED on and off by pushing the button. For the time being, the aim of this exercise is to get familiar with inputs and outputs and the way they are assembled

EXERCISE 3

Design a circuit, assembly and sketch, that fulfils the following requirements: a potenciometer connected to an analog input that will define the level of brightness of a LED. Swirling the potenciometer will make the LED fade or glow.

CONGRATULATIONS

You have completed SYS-STEM Module XX