GamePad

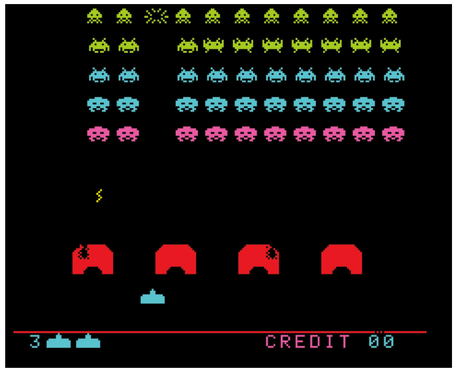
A tiny retro game console

Eduardo Rinaldi

Idea

Idea

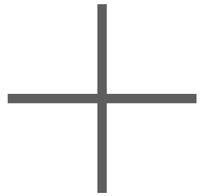
- When I was a kid **video-games** were simple and days spent playing with my Nokia 3310 in snake trying to beat my own record are countless
- It is thanks to video-games that today I am a computer scientist, as my dream has always been to make games to pay my salary
- So my idea is to **pay homage** to some of my favorite retro games by creating a **tiny retro console**
- In particular for this project the following games are available:
    - **Snake**
    - **Pong** (with its infamous artificial intelligence)



Idea

How can I realize it?

Arduino knowledge



C++ knowledge

Hardware components

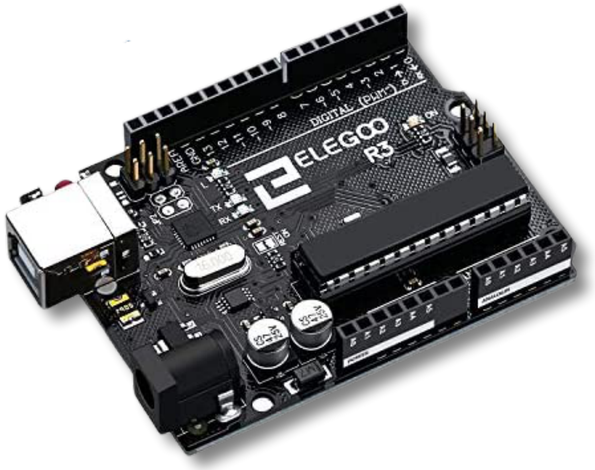
Components

Arduino and prototyping board

Arduino board

Model: Arduino UNO
Program memory (flash): 32 KB
SRAM: 2 KB
RAM: 1 KB

Programmable unit used for handling and coordinating all the other components

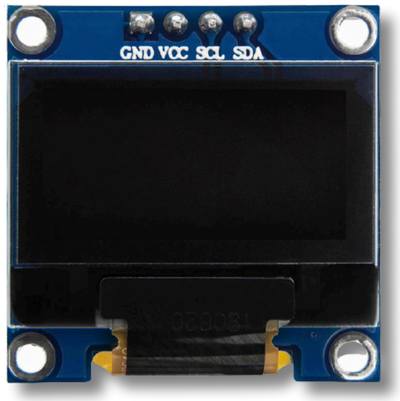


Prototyping board

Used for extending Arduino’s pins by adding additional 5V pins, additional GND pins and a mini breadboard for compact circuits

Components

Display and buzzer



OLED Display 0.96”

Model: SSD1306
Communication protocol: I2C
Integrated memory: none

Tiny display used with ***u8g2*** library for displaying the game

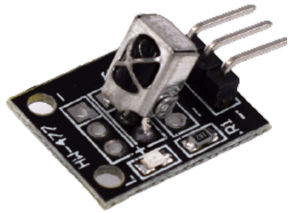
Passive buzzer

Used for playing some “beep” sounds

Components

IR Transmitter and Receiver

+



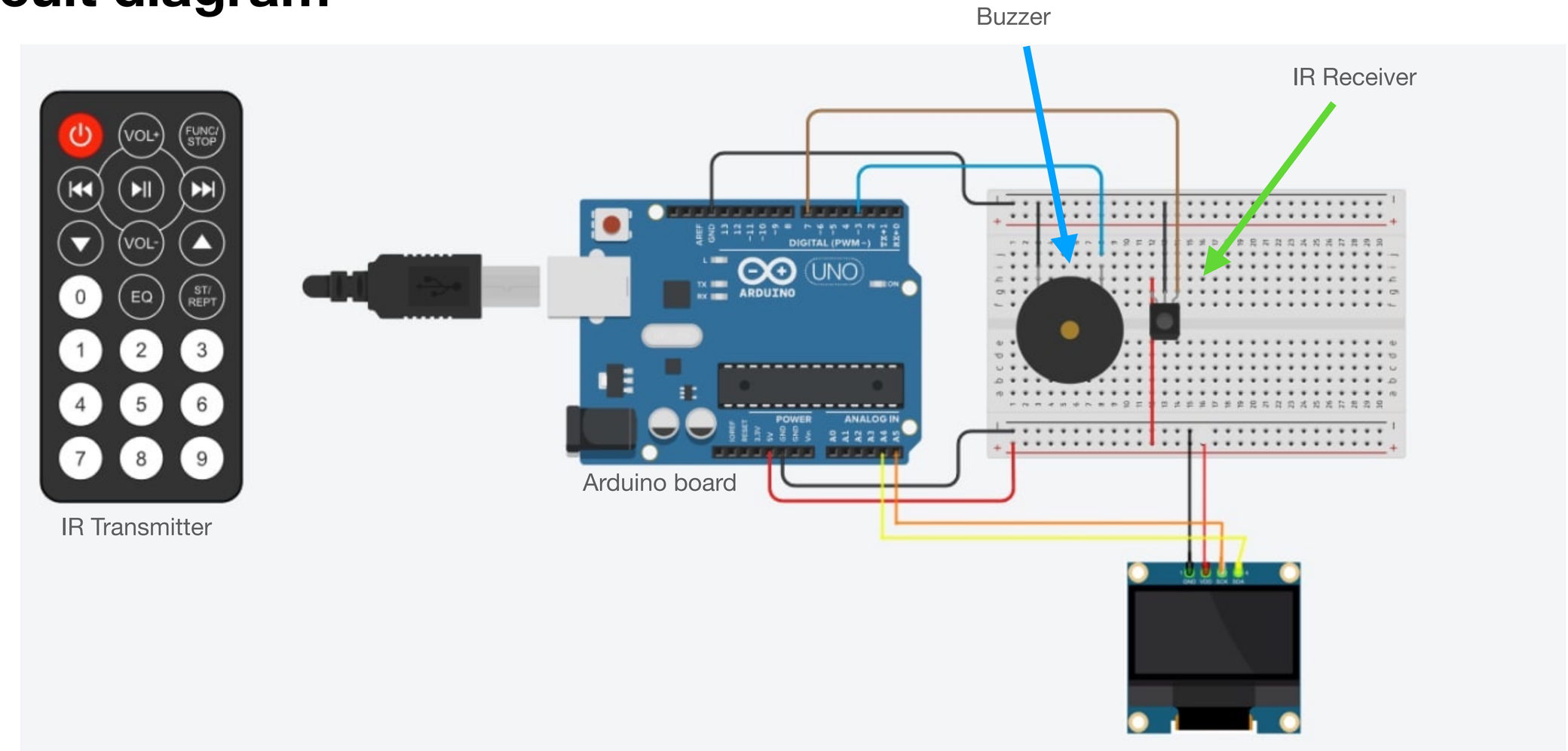
IR Transmitter and Receiver

Technology: Infrared

This will be **our wireless game controller**… yeah we are talking about a retro console, but who want cables?

Components

Circuit diagram



Software

Software

External libraries

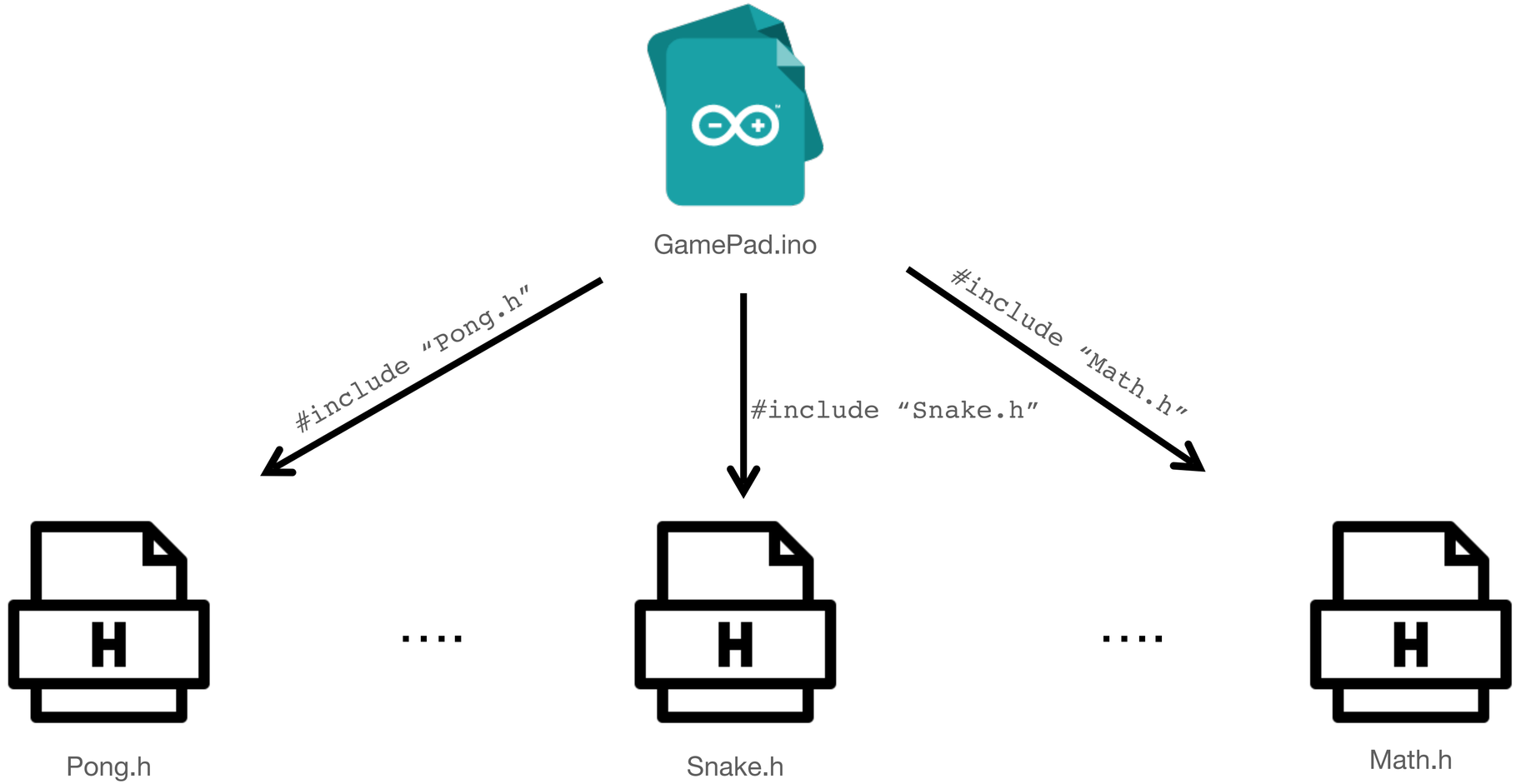
- **ListLib:** simple implementation of lists for Arduino [GitHub source] (*)
- **ezBuzzer:** non-blocking library for passive buzzer [GitHub source] (**)
- **IRemote:** library for decoding IR signals [GitHub source]
- **u8g2:** graphics library for drawing on an SSD1306 OLED display [GitHub source]; it provides some drawing primitives like:
  - `drawCircle(…)`
  - `drawBox(…)`
  - …

(*) Arduino does **not come with the STL** (Standard Library), so we don't have lists, queue or other data structure

(**) Arduino is a one core board and it does **not support multithreading**, so using the `Tone(…)` and `NoTone(…)` functions in combinations with `delay(…)`, means that we cannot *"beep"* and in the meanwhile do other stuff in code. This library provides an "escamotage" for achieving this particular result.

Software

Code organization



Software

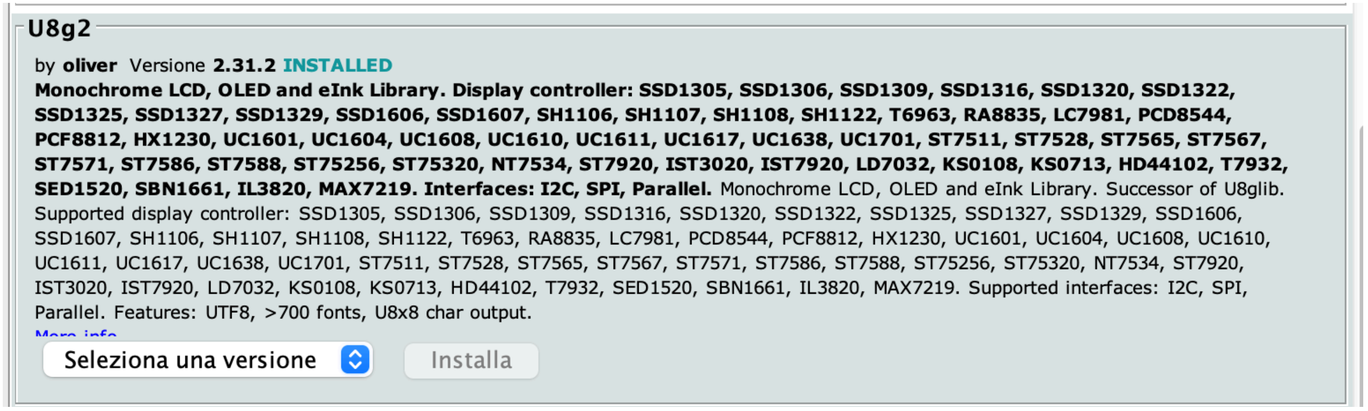
How to import the code

- Open the folder containing “`.ino`” file and “`.h`” files
- Open **ArduinoIDE** and `File` > `Open..` > select “`GamePad.ino`” file
- At this point you should have a similar view

Software

How to install external libraries

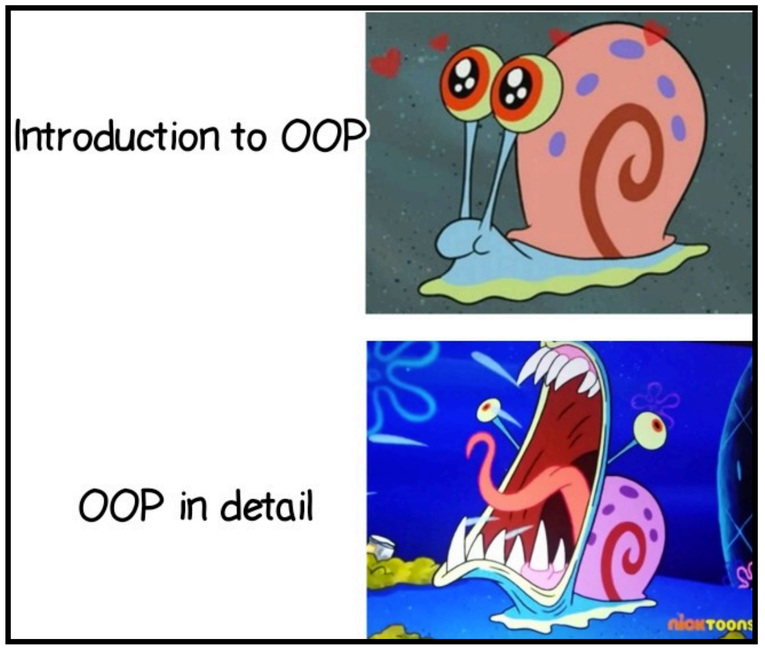
- Download the libraries from this link and unzip the content
- You should now have other 3 zip files, do not extract them! (If present ignore ".DS_Store" and "__MACOSX" files, they're automatically generated by Mac OS, my OS)
- Open **ArduinoIDE** and `"Sketch"` > `"#include library"` > `"Add library from .zip"` > select the 3 .zip files (one at time)
- For installing **u8g2** library just go on `"Tools"` > `"Manage libraries"` > search *"u8g2"* and select the one in the following image



Software

Code organization - OOP and classes

- I don’t want to scary you with some advanced OOP
- I just want to let you know that:
  - A `Game` is just an **abstract class** which *declares* a `Game::Update(…)` method to be *defined* by subclasses (`Snake`, `Pong`, ..)
  - This method takes the **infrared transmitter signal** as parameter and it’s called in `loop()` function for updating the game frame (the image)



Software

A *big* problem for a *small* memory

- As said before, Arduino is not the best when we talk about **memory** and in general, games don't get along very well with "*low memory*”
- In order to carry on this project, few (but very important) **optimization** have been made:
  - For representing numbers in the most cases we use **1 byte integers** (*)
  - Memory that actually concerns games is **dynamically allocated** and only when needed (for example when we press "play" on "play snake”), so C++ **pointers** are used
  - A lot of “`#define <constant> <value>`” are used; this because these constants will be **stored in program memory** (bigger) and not in the memory used at runtime for variable

(*) In order to obtain this optimization, I decided to write a very tiny math library for dealing with 2 byte 2D vectors/points

Final result

Final result

GamePad

Final result

GamePad - Menu

Commands

Start selected game

Go down in menu

Go up in menu

Final result

GamePad - Snake

Commands

End game

Pause game

Move up

Move left

Move right

Move down

Final result

GamePad - Pong Game

Player paddle

Commands

End game

Pause game

Move paddle down

Move paddle up

Demo

Youtube video link

or

SCAN ME!

Thanks for the attention!
Now it’s **your turn**, let’s **test together**
GamePad