# SON 2026-2027 lecture 3

## 3TC Audio Project @ INSA Lyon

**Tanguy Risset**

https://inria-emeraude.github.io/son/

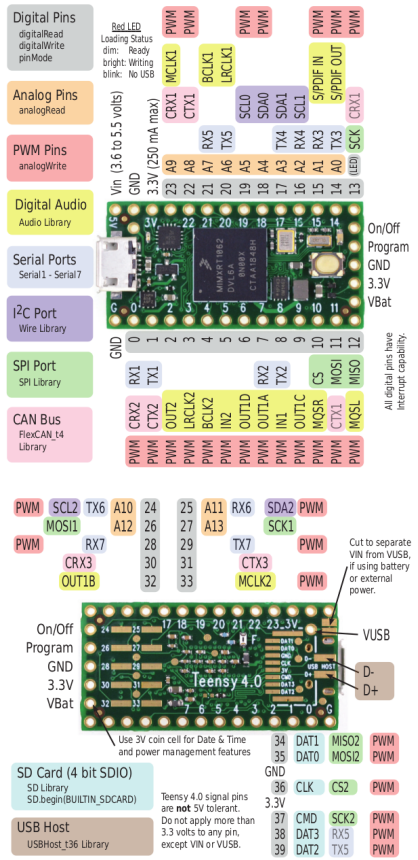
# Hardware Control and Audio Codec Configuration

# Electronic Basics

- Basic circuits do need to be implemented in order to *control* the various parameters of the DSP algorithms
- Hardware controllers such as buttons and potentiometers are used
- Hence, in case you feel like your electronics skills are a bit rusty, feel free to review the following page: https://ccrma.stanford.edu/wiki/Introduction_to_Electronics_(condensed).

# Teensy 4.0 Pinout (recall)

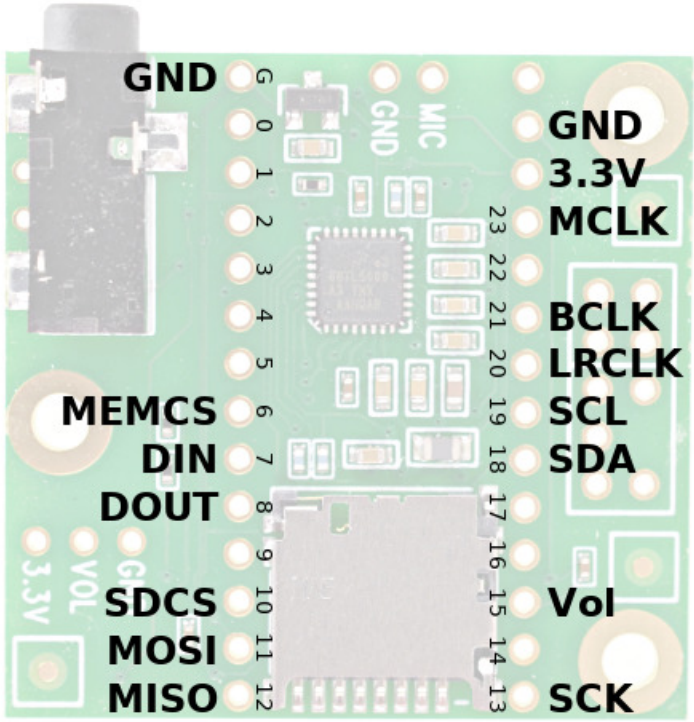
- Teensy pins map is on the PJRC website).
- Most pins can be used as digital I/Os. Some pins noted "A(N)" can be used as analog inputs.
- *3.3v power* can be retrieved from the top right corner pin and the *ground* from the top left pin.
- **Warning** Make sure to never connect the 5.5v Vin pin to any other pin of the Teensy




Teensy pinout.

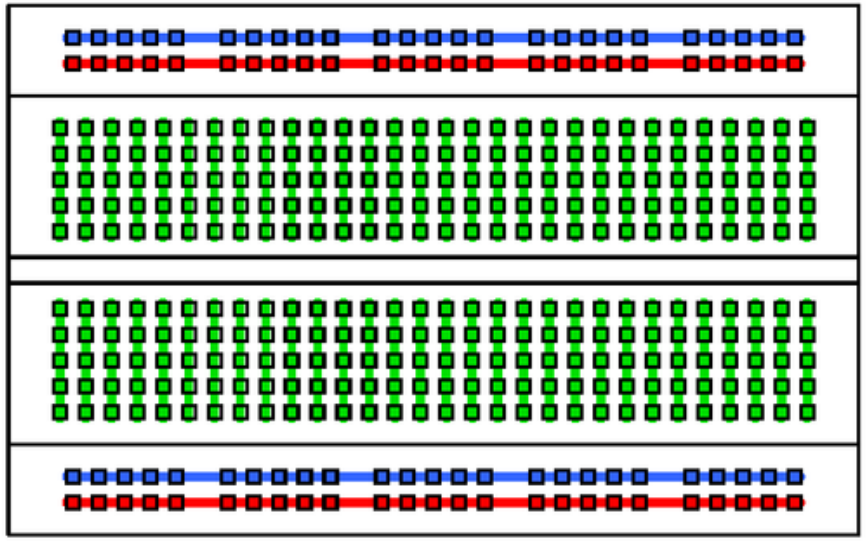
# Teensy I2S to Audio Shield (recall)

- The Teensy audio shield uses a bunch of pins on the Teensy for i2c and i2s communication:
- This means that these pins (besides GND and 3.3v, of course) **cannot** be used for something else (i.e., connecting external sensors).




Teensy audio shield pins.

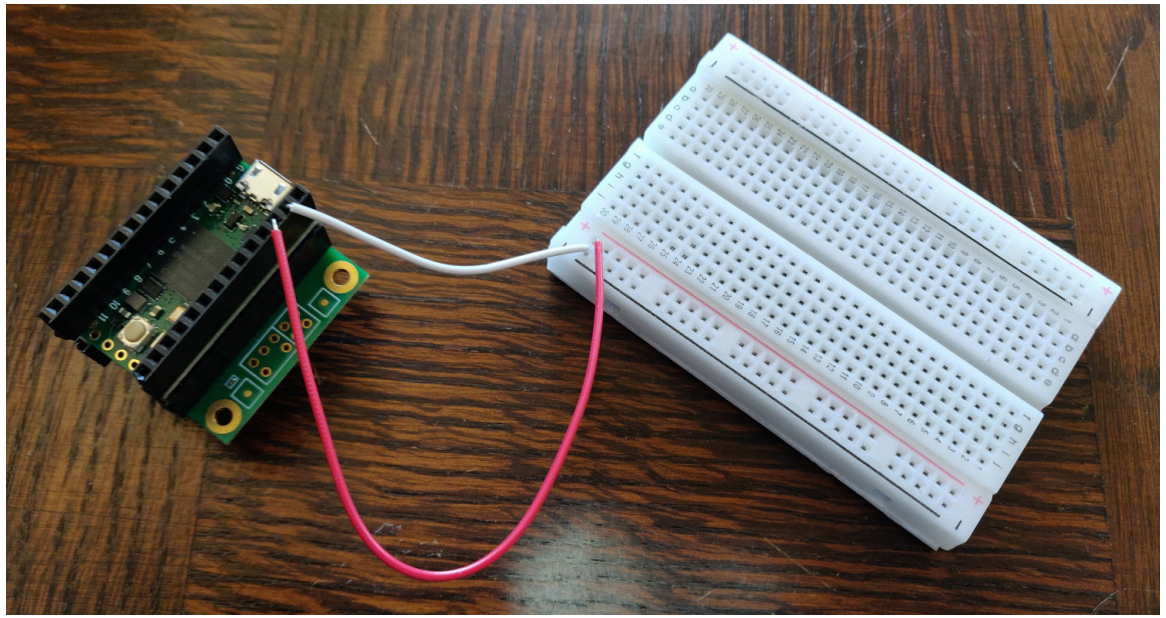
# Breadboard: how is it connected



Connected pins on the breadboard

# Bringing Power to Your Breadboard

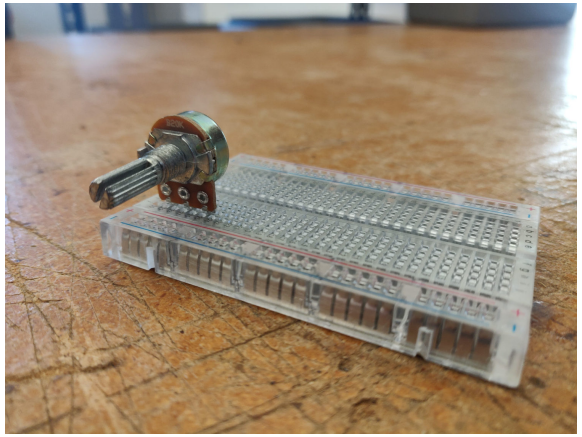
- Bring power to the breadboard included in your kit using jumper wires:
- Connect the 3.3v pin to the red strip and the GND pin to the black strip of the breadboard.
- **WARNING: Do not connect the 5.5v pin to the breadboard!**



Teensy connected to the breadboard.

# Adding a Rotary Potentiometer to the Circuit

- Your kit should come with a couple of rotary potentiometers:
  - Place it on the breadboard,
  - Connect its leftmost pin to power
  - and its rightmost pin to the ground.
  - Finally, connect its center pin to the A0 pin of the Teensy using a jumper wire.
  - Please, note that we're using A0 pin since it is not used by the audio shield (see previous section).



Rotary potentiometer mounted on the breadboard.

# Testing the Potentiometer

- the potentiometer will deliver:
  - 3.3v current at its center pin if it is *fully turned to the right side*,
  - and 0v if it is *fully turned to the left side*.
- The `AnalogRead()` function samples this current between 0 and 1024.
- Check the serial monitor, make sure that the values you're getting are consistent.

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

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  delay(100);
}
```

## Controlling DSP Parameters With the Potentiometer

- Use the potentiometer values to control the frequency of the `Sine`
- Note that `sensorValue` needs to be turned into a frequency in hertz so we just add 100 to it to get a frequency between 100 and 1123 hertz.

```
#include <Audio.h>
#include "MyDsp.h"

MyDsp myDsp;
AudioOutputI2S out;
AudioControlSGTL5000 audioShield;
AudioConnection patchCord0(myDsp,0,out,0);
AudioConnection patchCord1(myDsp,0,out,1);

void setup() {
  AudioMemory(2);
  audioShield.enable();
  audioShield.volume(0.5);
}

void loop() {
  int sensorValue = analogRead(A0);
  float freq = sensorValue + 100;
  myDsp.setFreq(freq);
}
```

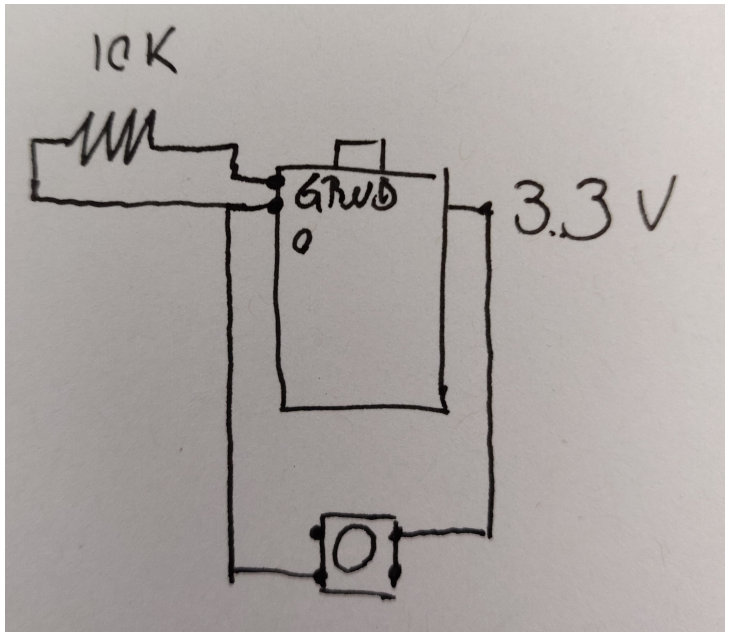


--- ## Schematic for a button

TODO https://arduinotogo.com/wp-content/uploads/2016/08/ch5-button-pins-schematic-

# Using a Button

- A button requires a *pulldown resistor* is required (alternatively, a pullup resistor could be used, of course) to ensure 0V when the button is not pressed and get a stable signal to be measured on the Teensy.
- Hence, the following circuit must be implemented:
- We are looking at discrete values here (0 or 1). Hence, the button shall be connected to a digital pin (number 0, for example).




Circuit to connect a button to the Teensy.

# Program Using a Button

```
#include <Audio.h>
#include "MyDsp.h"

[....]

void setup() {
  pinMode(0, INPUT); // configuring digital pin 0 as an input
  AudioMemory(2);
  audioShield.enable();
  audioShield.volume(0.5);
}

void loop() {
  if (digitalRead(0)) { // button is pressed
    myDsp.setGain(1);
  }
  else {
    myDsp.setGain(0);
  }
  int sensorValue = analogRead(A0);
  float freq = sensorValue + 100;
  myDsp.setFreq(freq);
}
```

# Exercise: Looping Between Notes by Pressing a Button

- Expand the "note looper" that you implemented as part of the audio system course so that new notes are triggered when a button is pressed (as opposed to be triggered automatically).
- Every time the button is pressed, a new note is produced.
- This means that you'll have to turn your *push button* into a *switch* using software techniques...
- Finally, make sure that gain is controllable using a rotary potentiometer.

# Audio Codec Configuration

## Communication with Audio Codec

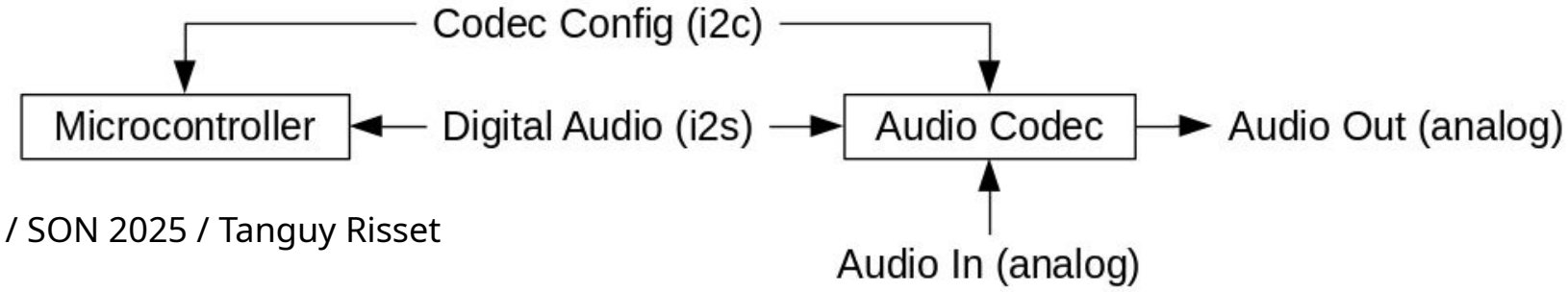
Audio codecs usually use two different communication channels to connect to the processor unit (whether it's a CPU, a microcontroller, etc.):

- an I2C bus is used to configure the codec,
- an I2S bus is used to transmit digital audio data.

I2C (pronounced I-squared-C) is a serial communication protocol heavily used in the field of microelectronics. Most digital elements in an electronic circuit communicate using i2c.

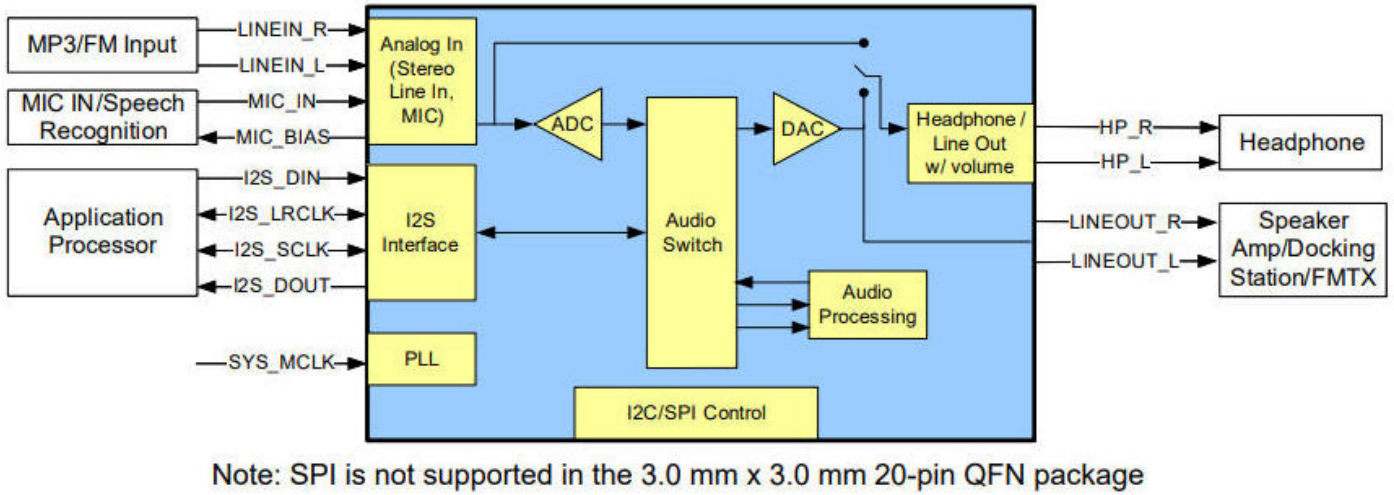
I2S (pronounced I-squared-S) is also a serial communication protocol but targeting specifically audio applications. We'll see that they're very close to each other in practice later in this class.

The Teensy Audio Shield hosts an audio codec (SGTL5000) which is connected to the Teensy using the following model:



# Quick Tour of the SGTL5000

- SGTL5000 is a low-power 24-bit, 8 kHz to 96 kHz audio codec. I




Note: SPI is not supported in the 3.0 mm x 3.0 mm 20-pin QFN package

- by default, headphone and microphone are activated.
- If you want to use Line Input/output (adapting pre-amplification), you will have to *configure the codec* using methods of `AudioControlSGTL5000` class (methods for changing `volume`, `micGain`, `lineInLevel`, etc.)
- In most cases, writing that will be sufficient: `audioShield.enable();`

# End of Lecture 4