DIY Synth Design Workshop

Third Edition - June 2023

Riccardo Marogna

Day #1 Plan

Welcome

Setup the workbench: install, connect, test

Meet Teensy

Connect Teensy

C Programming in a Nutshell

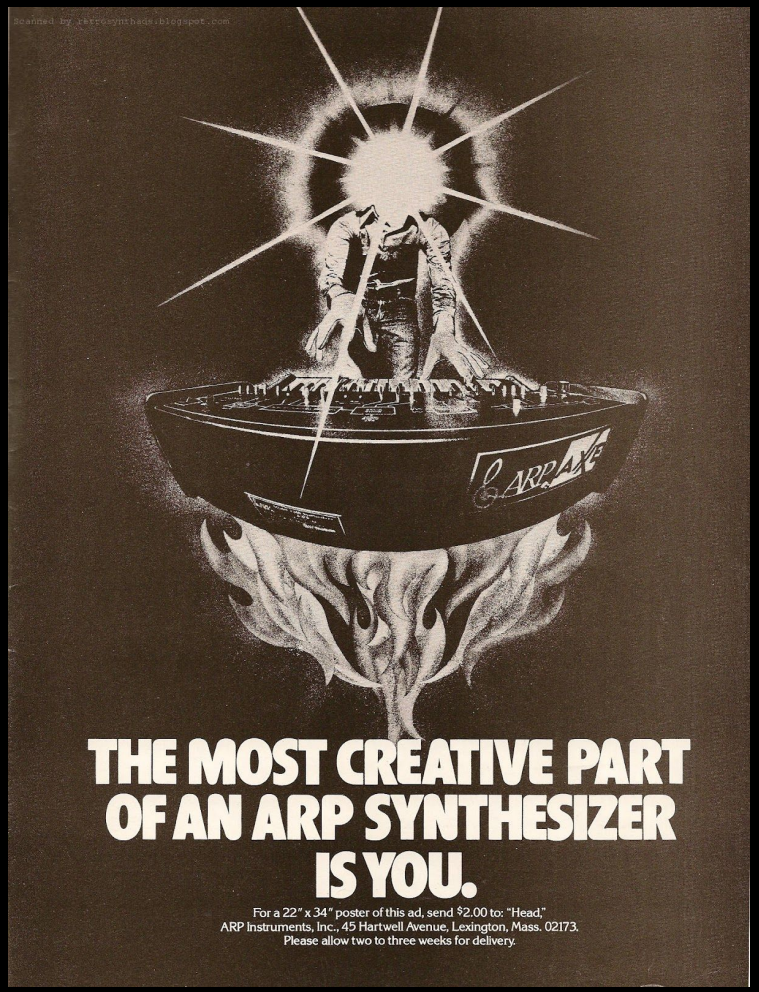
Setting Up the Breadboard

Introducing the Audio System Design tool

Our first sound!

Taming the Synth: Adding Knobs

Intro


Scanned by retrosynthads.blogspot.com
ARP AXE
THE MOST CREATIVE PART OF AN ARP SYNTHESIZER IS YOU.
For a 22" x 34" poster of this ad, send $2.00 to: "Head,"
ARP Instruments, Inc., 45 Hartwell Avenue, Lexington, Mass. 02173.
Please allow two to three weeks for delivery.

Setting up the workbench

Setting up the workbench: Arduino IDE

1. Connect to the Internet

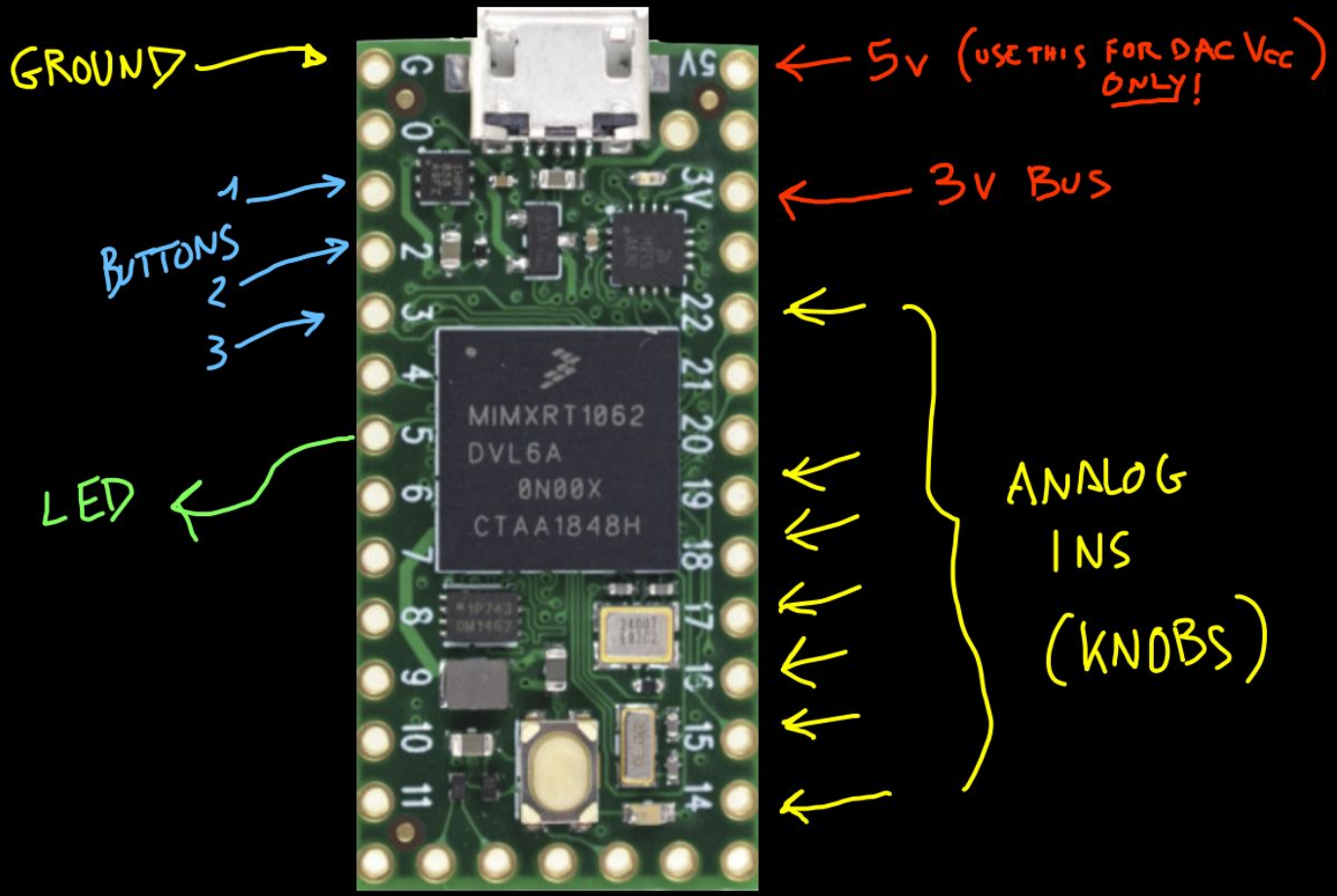
    Wifi: WD4X Guest
    Password: wifipasswordfortheguests

2. Install Arduino IDE + Teensy board libraries following these instructions: https://www.pjrc.com/teensy/td_download.html

Setting up the workbench: extra libraries

1) Open Tools → Library Manager
2) Search for *Smoothed*
3) Install it

Meet Teensy



Teensy is a very powerful MCU (Micro Controller Unit)

A MCU is a programmable device, and it can communicate with the external world through several inputs/outputs

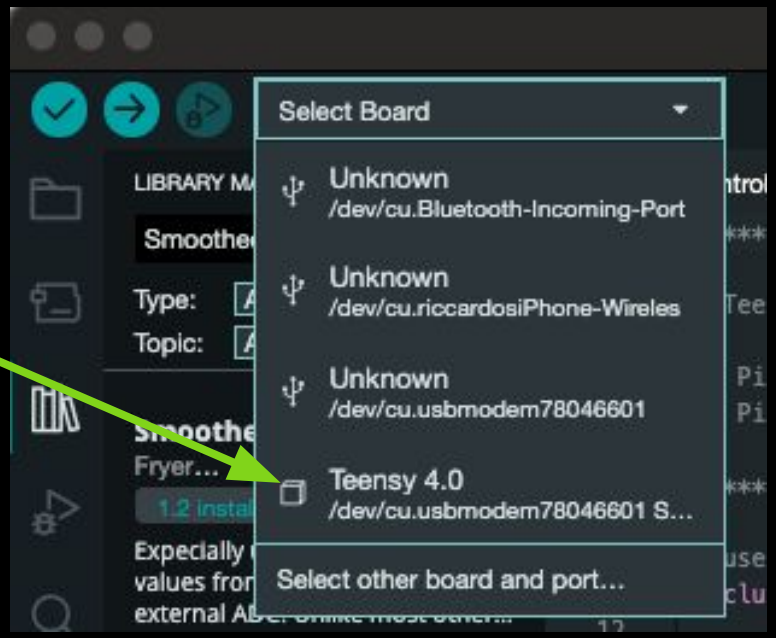
Connect Teensy to Arduino IDE

Teensy can be programmed using C/C++ Language in the Arduino IDE

Arduino IDE is a '*program for writing programs*' plus a collection of libraries which allows us to write less code and have more fun

1. Connect the Teensy to your Laptop using the micro usb cable

2. In the Arduino IDE, select the Teensy Board



Setting up the workbench:
The Synth DIY Template File

Open a Browser, go to the Synth DIY workshop repo:

https://github.com/spareknobs/diysynthworkshop

Click on CODE -> Download Zip

Unzip it in your favourite spot

[If you want, rename it (the folder, and the .ino file inside –

BUT use the same name for both!)]

Test that everything is working OK so far

File → Open → Navigate to the folder Start_Here → select Start_Here.ino

Click on the Compile Icon

Wait for it

Wait for it

Done Compiling. All good?

Click on the Upload Icon

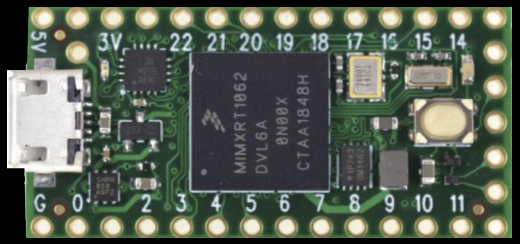
Wait for it

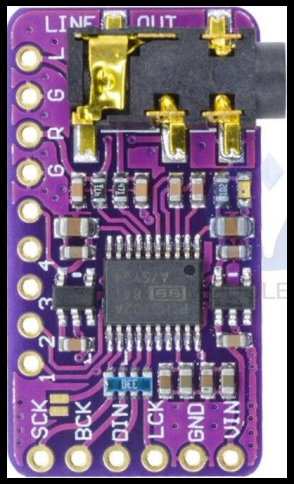
Done Uploading. All good?

Setting up the Hardware

The BreadBoard

TEENSY



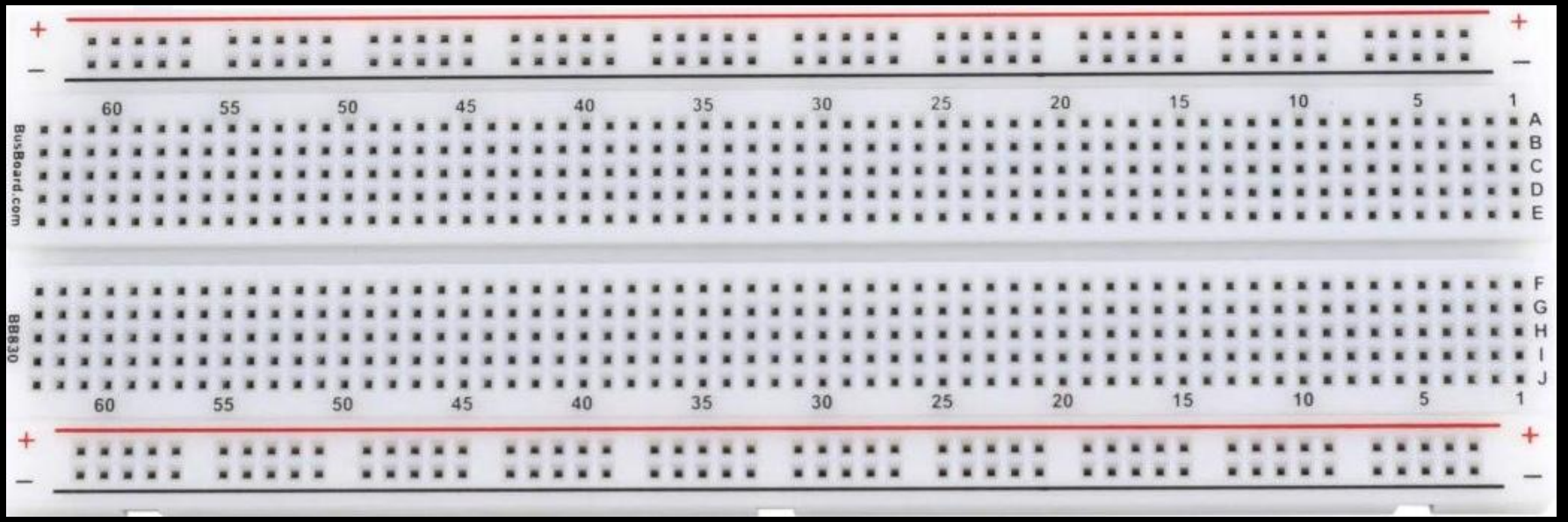


DAC (AUDIO OUT)

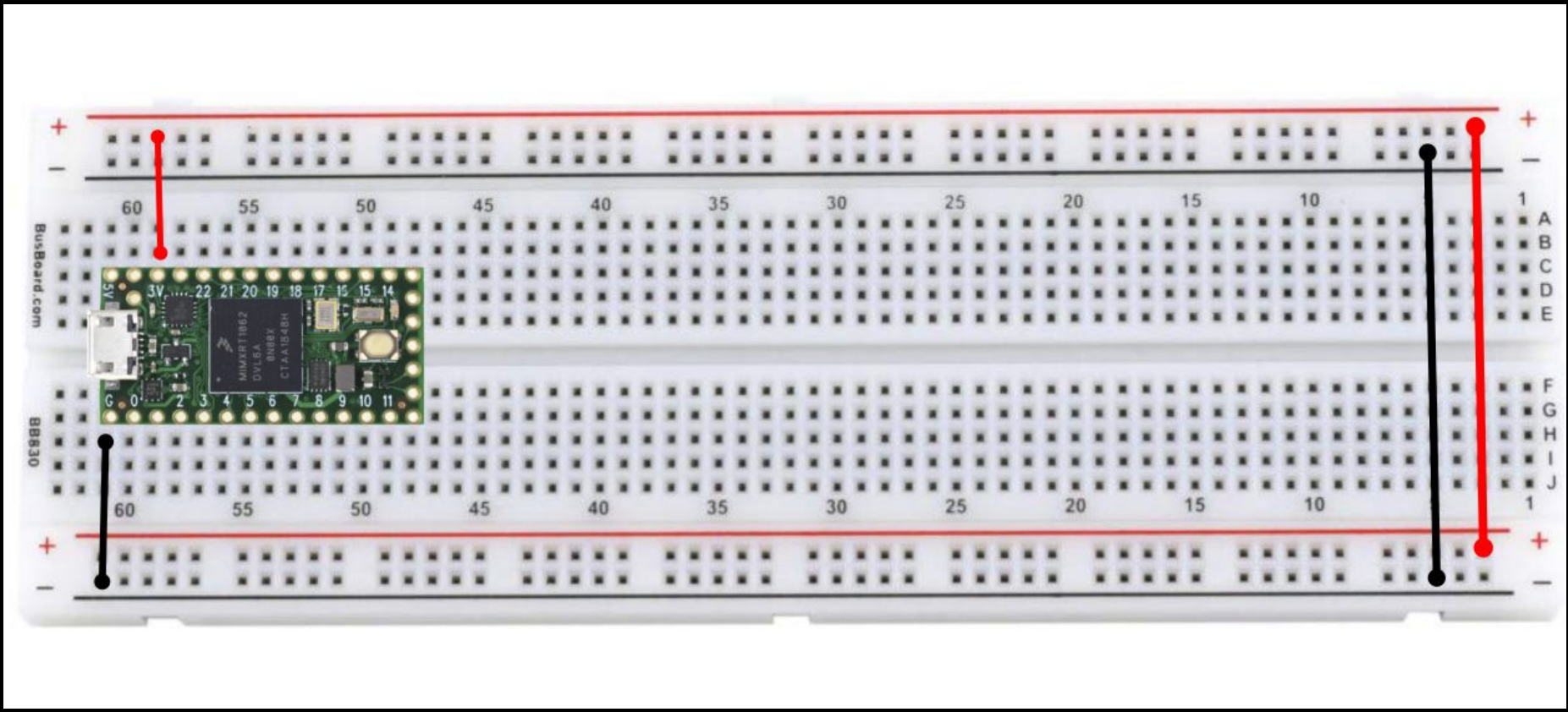
POTENTIOMETER

BUTTON

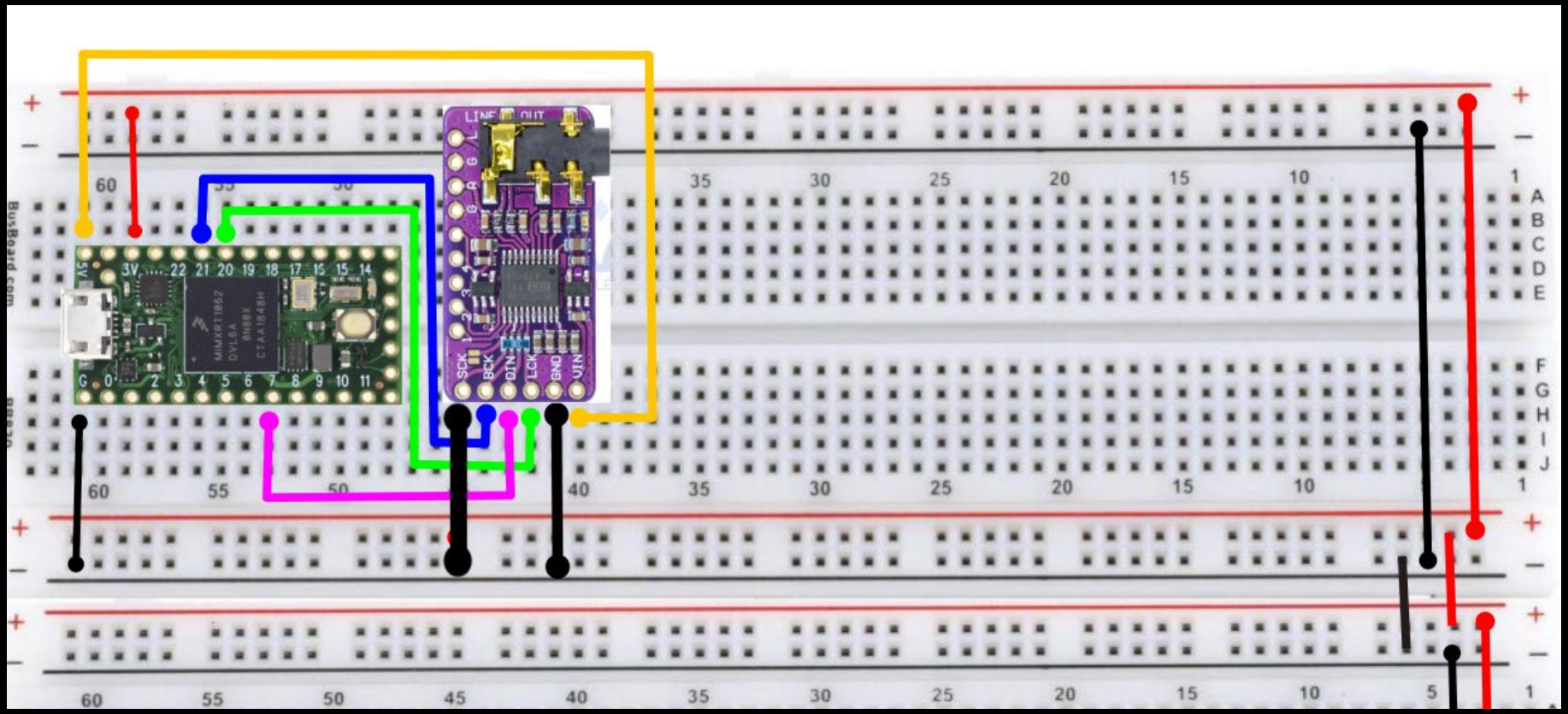
BREADBOARD



Prepare the Breadboard, connect Teensy



How to connect Teensy to the DAC



Pin 5v --> dac board Vcc
Pin 20 --> dac board LCK
Pin 21 --> dac board BCK
Pin 7 --> dac board DIN
Ground ← dac board SCK
Ground ← dac board GND

!! SAVE YOUR EARS !!

- Always take off your headphones before uploading a new code to Teensy

- Listen from a distance until you're sure everything is working fine

- In general, in every example the potentiometer at pin 14 will be the „Master Gain" to adjust the output level

Setting up the Software

C Programming in a nutshell

The language used in the Arduino Environment is C and C++

A Language defines a syntax, a set of native data types, a way of thinking...

C Programming 101: Syntax

```
// Any line starting like this one is a comment and will be ignored
// by the machine.

// Comments are reserved for us, poor human beings,
// to understand what's going on.


/* if you want to comment a lot and use more lines, you can also use
these kind of syntax */
```

C Programming 101: Variables

Variables are used to store stuff of different kind. Think of them as placeholders, boxes containing some kind of value.

A variable must be declared before using it. That is, write its type and a name.

Something like:

```
int var;
```

→ [Rule: each command in C must end with a ;]

For our purposes, we will use mostly use just 2 kinds of variables:

- `float  a floating point number`
- `int    an integer`

C Programming 101: Variables and Data Types

```
// declare a variable of type integer, named charles

int charles;



// declare a variable of type integer, named bob, and initialize it
to a value

int bob = 5;



// declare a variable of type floating point, named c, and
initialize it to a value

float  c = 3.14;
```

C Programming 101: Methods

A method is a piece of code designed to do something. Think of it as a function, or a task. It may have some inputs (arguments) and/or outputs.

It looks like this:

```
void setup(){
    // …do something…
}
```

In the Start_Here file you see two methods already prepared for you, setup() and loop()

C Programming 101: Methods

For building our synth, we will also *invoke (use)* some methods, like in:

```
int value = readAnalog(pin);
```

Sometimes we will use methods which are specific to a module/object . Think of them as *handles* to set properties of that object. For instance:

```
oscillator.frequency(440);
```

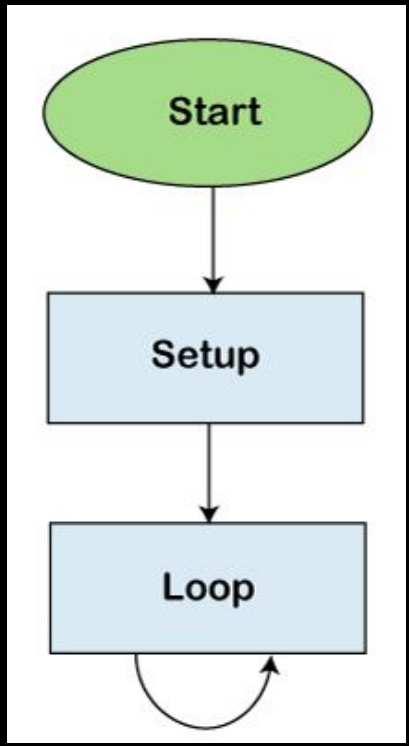
The Boring Life of a Teensy

Teensy ‘life cycle’ is organized into two distinct phases: setup() and loop().

The instructions written in setup() will be executed only once when Teensy turns on. Here we will *initialize* the synth status

The instructions contained in loop() will be executed continuously

The life of a MCU is made 99.9% of loops of code repeated over and over again.

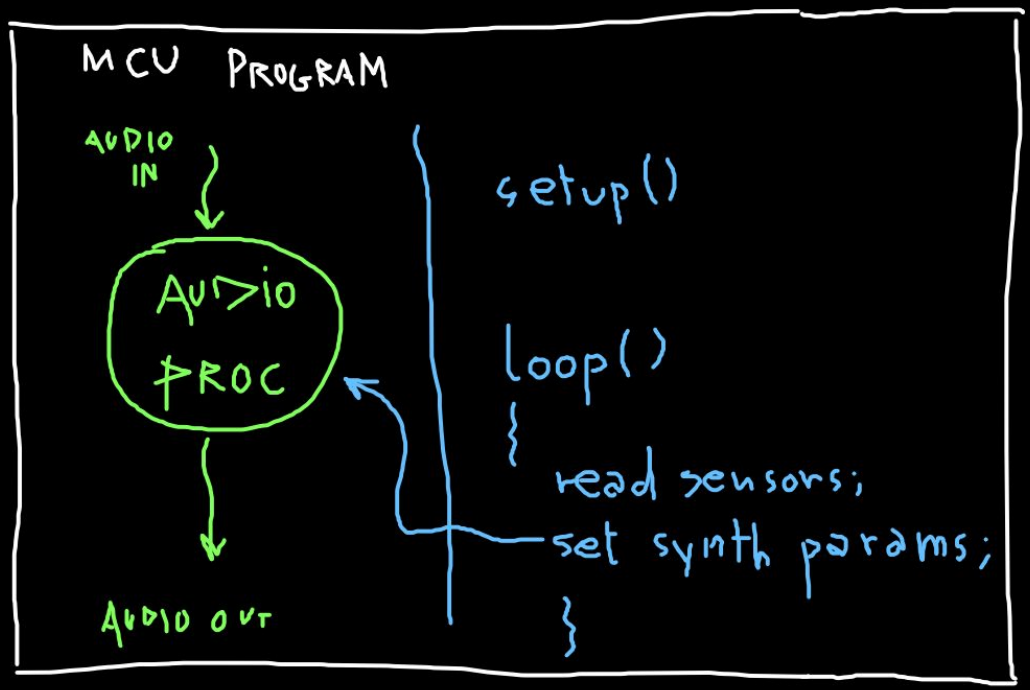


How Teensy makes sound

Then, it has to continuously synthesize sound and send it to the audio output

..while reading inputs from potentiometers, buttons, sensors... and also do whatever we ask it to do with these inputs – that is - control the synth parameters

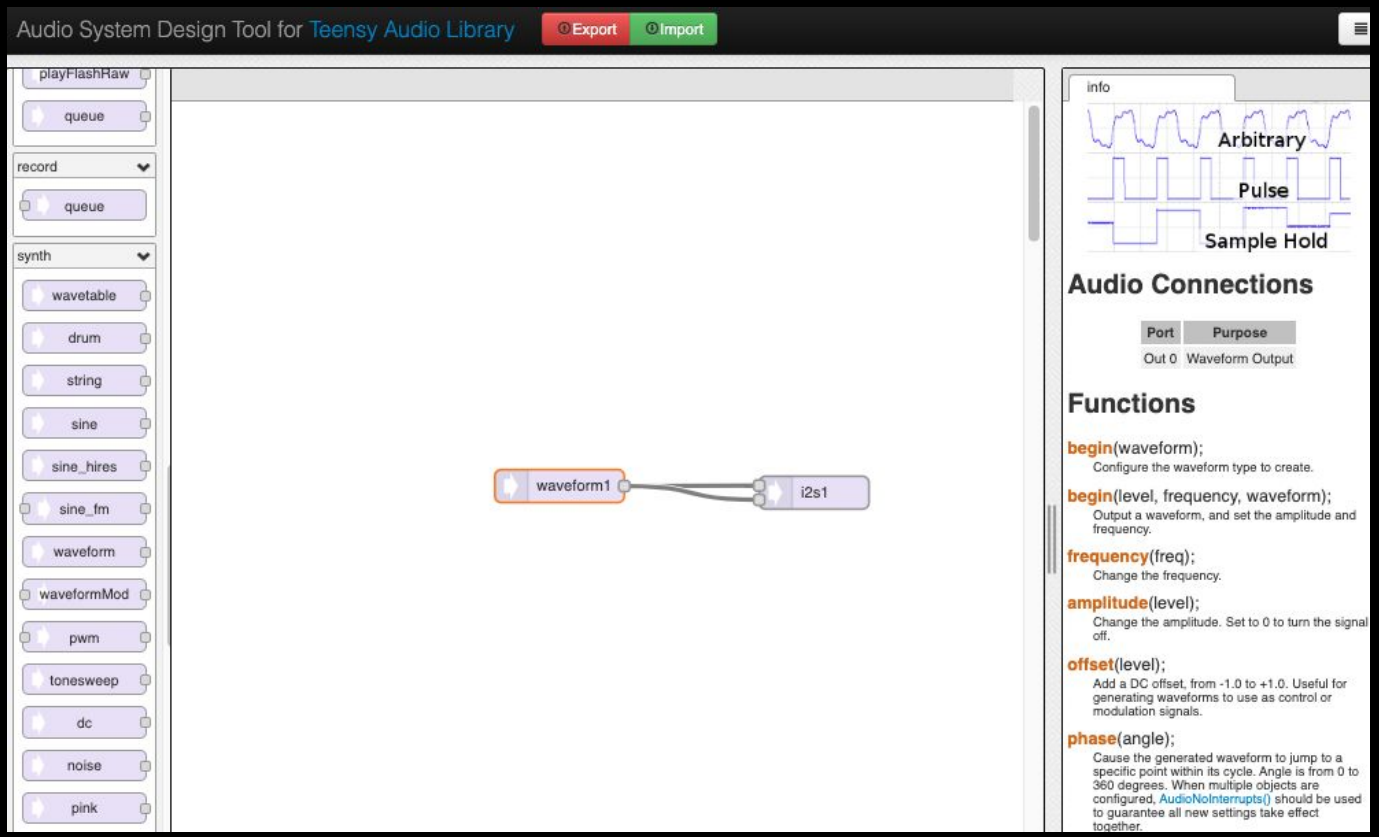
Luckily, the audio part is managed internally and we don't have to think about it.
What is left to us is to patch together the audio modules and then control them in the loop()



How to program Teensy for Audio Generation

Teensy provides a visual programming environment to patch synthesizers and effects

→ Open browser, got to: https://www.pjrc.com/teensy/gui/index.html



My First Oscillator

- Patch the synth modules in the Audio Design Tool as shown

- Rename the waveformMod1 module as osc

- Export → Copy the code snippet

- Open the Template File → Paste the code snippet where indicated

Locate the Setup Method → add this line of code:

```
osc.begin(0.2,300,WAVEFORM_SINE)
```

Compile

Upload

Listen!

Extra: let's make it random!

in loop()
add
these
lines:

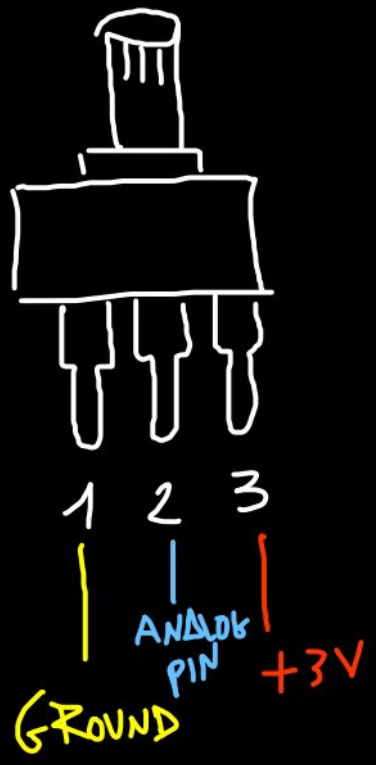
```
// generate a random frequency
int freq = random(1000);

// set the oscillator frequency
osc.frequency(freq);

// wait a little
delay(150);
```

BREAK!

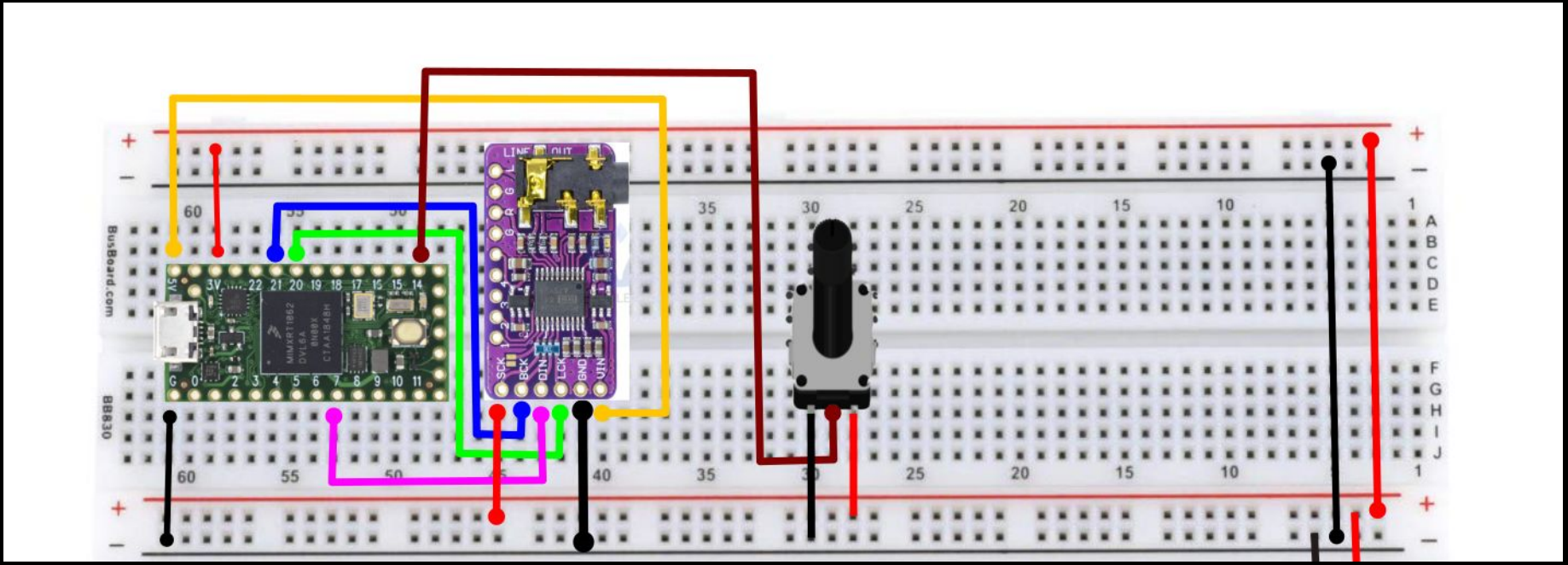

EVERY NUN NEEDS
A SYNTHI
Electronic Music Studios (London) Limited
49 Deodar Road London SW15 01-874 2363
New York 408 East 78th Street N.Y. 10021 USA
The Synthi Range by EMS

Control the frequency of the oscillator with a knob

This is a
Potentiometer



How to connect a potentiometer



Pot pin 1 --> Ground
Pot pin 2 --> Teensy Pin 14
Pot pin 3 --> +3v bus

Read a Potentiometer value and *Print it*

With the method readKnob(...) we can get a read of the position of the potentiometer.

we can store the reading in a int variable:

```
int value = readKnob(14);
```

and then print it on the Arduino consolle:

```
Serial.println(value);
```

- Compile, Upload

- Then open: Tools → Serial Monitor

- You can also view a plot of the output: → Serial Plotter

Read a Potentiometer value and *Map it*

With the method readKnob(...) we can get a read of the position of the potentiometer.

Problem is – this value is in the range 0 to 1023

Let's say we want to use it to control the volume of an oscillator

We need to rescale from the range 0-1023 to the range 0.0 - 1.0

→ Arduino provides a method for this: map()

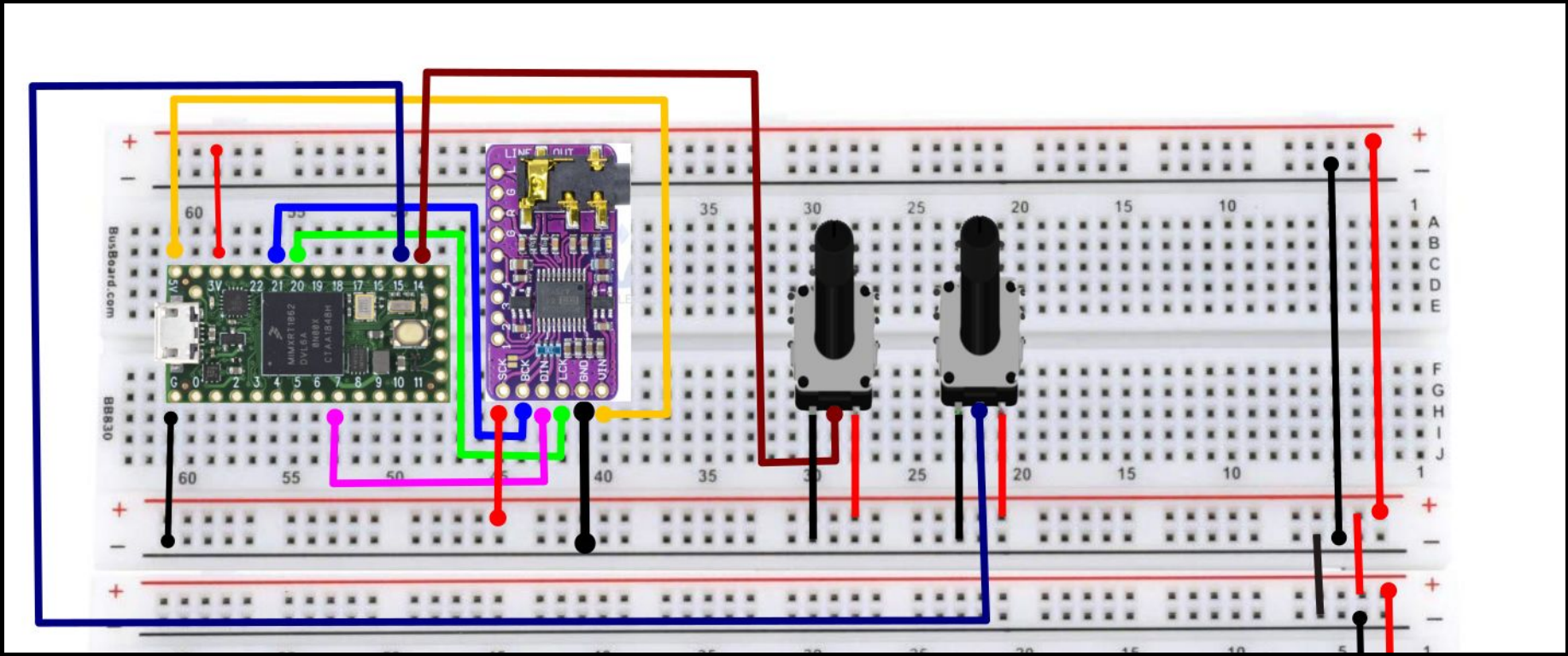
Syntax of map:

```
map( input, input_min, input_max, output_min,
output_max)
```

```
float gain = map( readKnob(14), 0.0, 1023.0, 0.0, 1.0 );
```

More Pots: Control the Frequency!

Control frequency and amplitude of the oscillator



Pot 1 pin 1 --> Ground
Pot 1 pin 2 --> Teensy Pin 14
Pot 1 pin 3 --> +3v bus

Pot 2 pin 1 --> Ground
Pot 2 pin 2 --> Teensy Pin 15
Pot 2 pin 3 --> +3v bus

Control frequency and gain of the oscillator with two knobs

In the Loop() add:

```
// read osc gain from knob at pin 14
float amp = map( readKnob(14),0.0, 1023.0, 0.0, 1.0 );
osc.amplitude(amp);

// read osc rate from knob at pin 15
float val = map( readKnob(15),0.0, 1023.0, 0.0, 1.0 );
float freq = (powf(val,4)*2000.0 + 30.0) * fratio;
osc.frequency(freq );
```

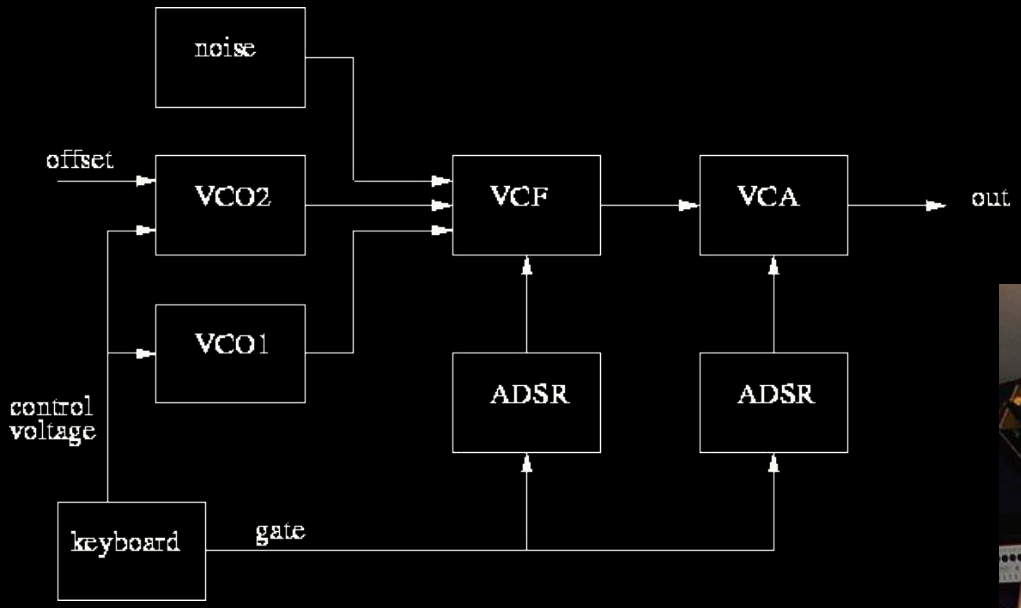
In the next episode:

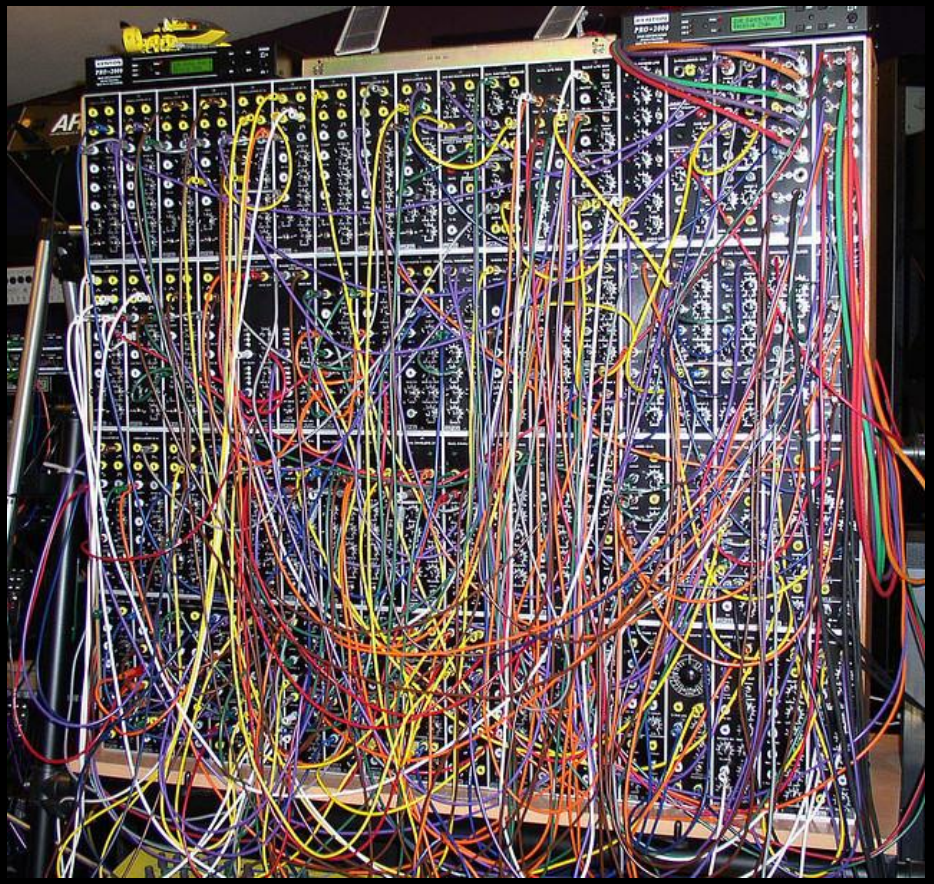
More Modules! Filters! LFOs!
Extras! FX! Touch Sensors!
Photodiodes!
Drum Machines!

Day #2 Plan

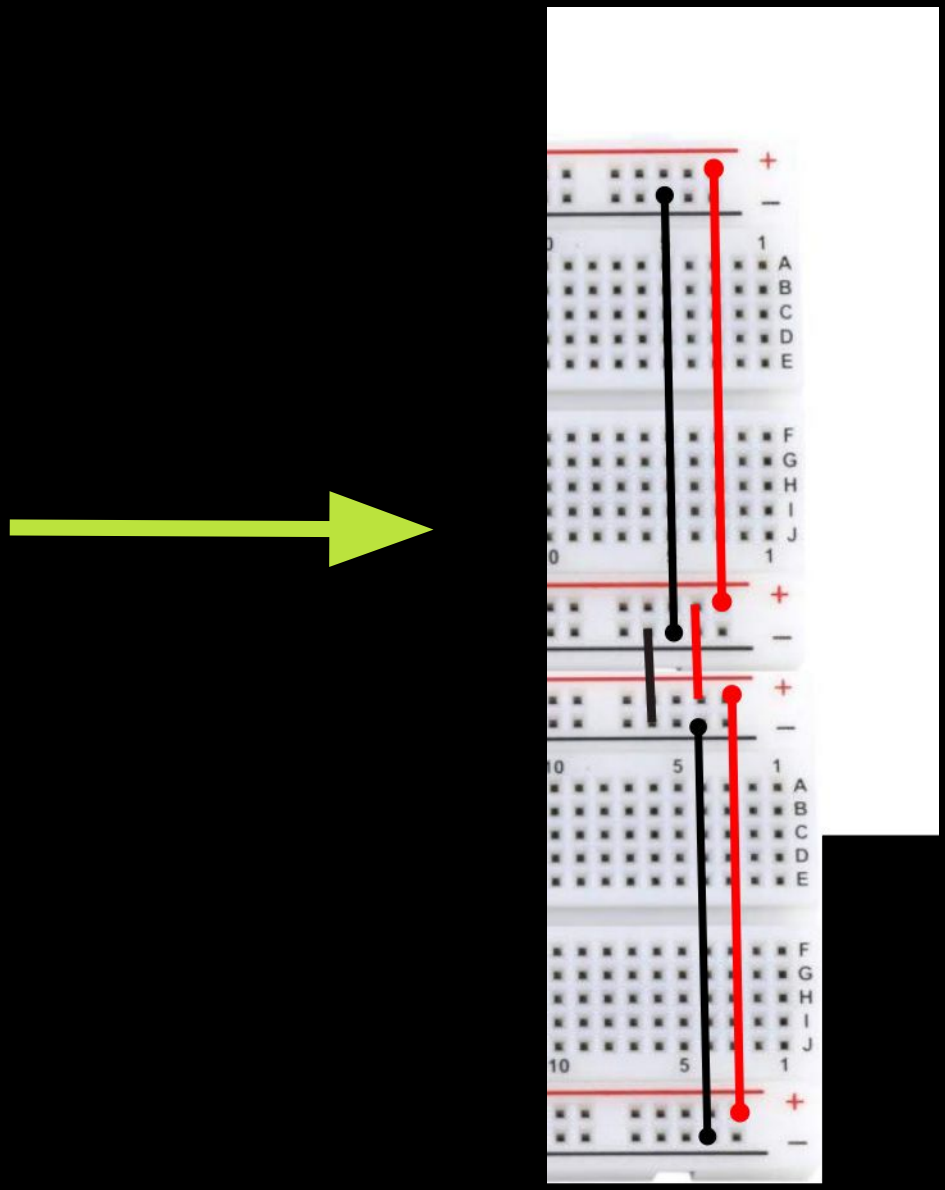
- Welcome back!
- Update
- The Modular Synth Logic
- More modules: Filter, LFO, Sequencer
- Finalising the „Simple Synth”
- BReak!
- Extras and Add-ons:
  - Delay FX
  - Adding LEDs
  - Touch sensing
  - Control with light (photodiodes)
  - Drum Machine
  - …

Synth Design: The Modular Logic



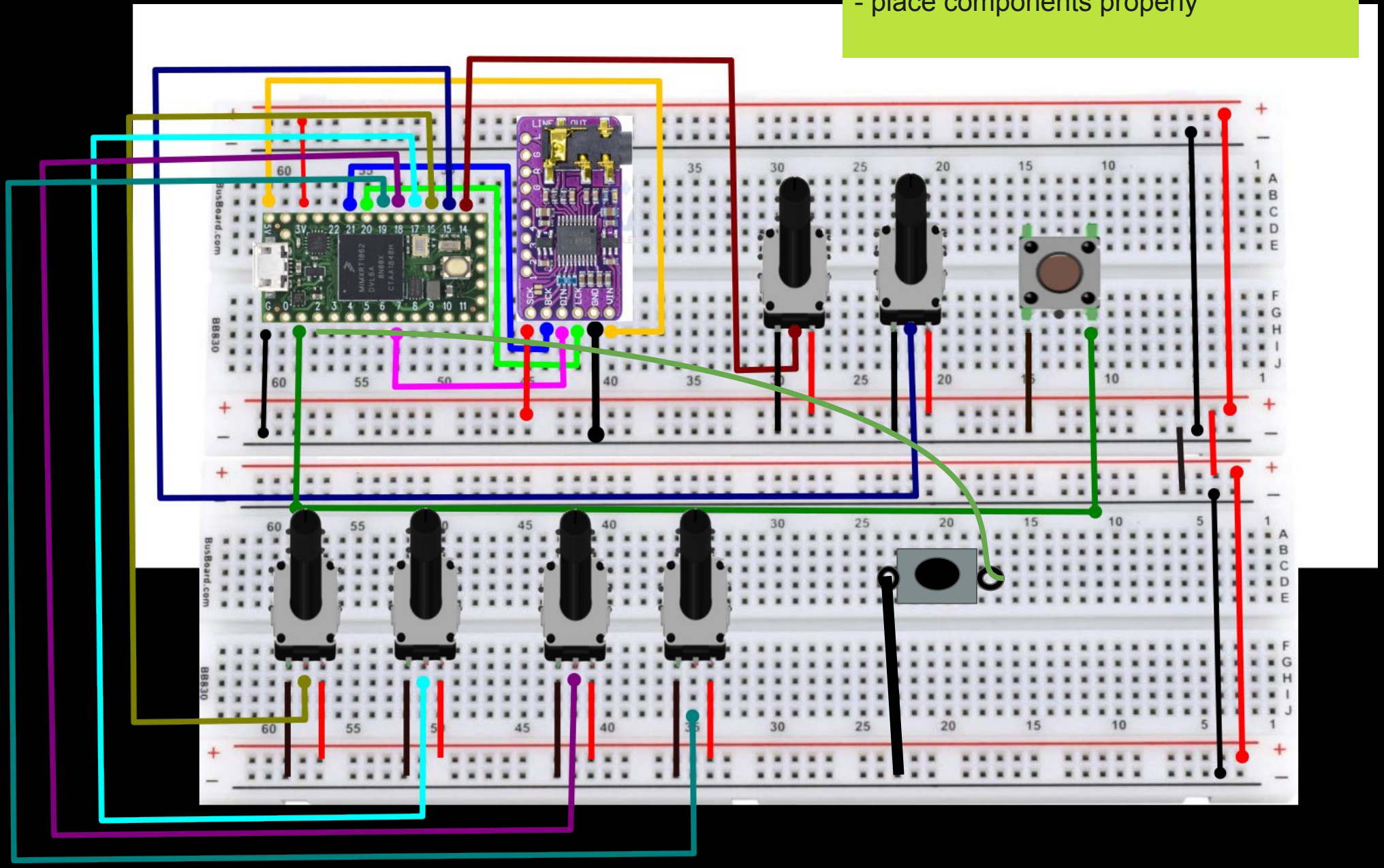


Extending the Board – Connecting 3v and Ground Buses



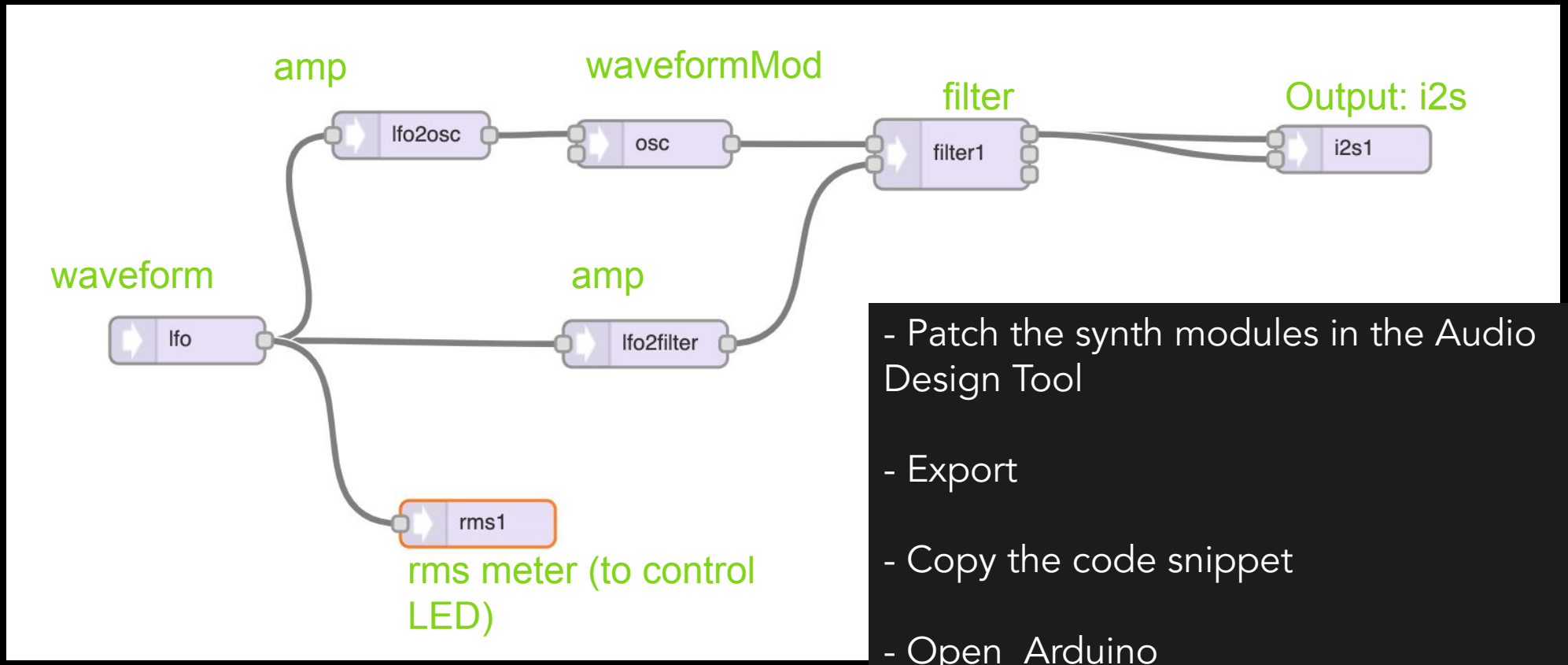
More Pots!

Hint: keep the breadboard nice & tidy!
- trim some wires at the proper length
- use colors properly
- place components properly



Synth Design with the Audio Design Tool

https://www.pjrc.com/teensy/gui/index.html



- Patch the synth modules in the Audio Design Tool

- Export

- Copy the code snippet

- Open Arduino

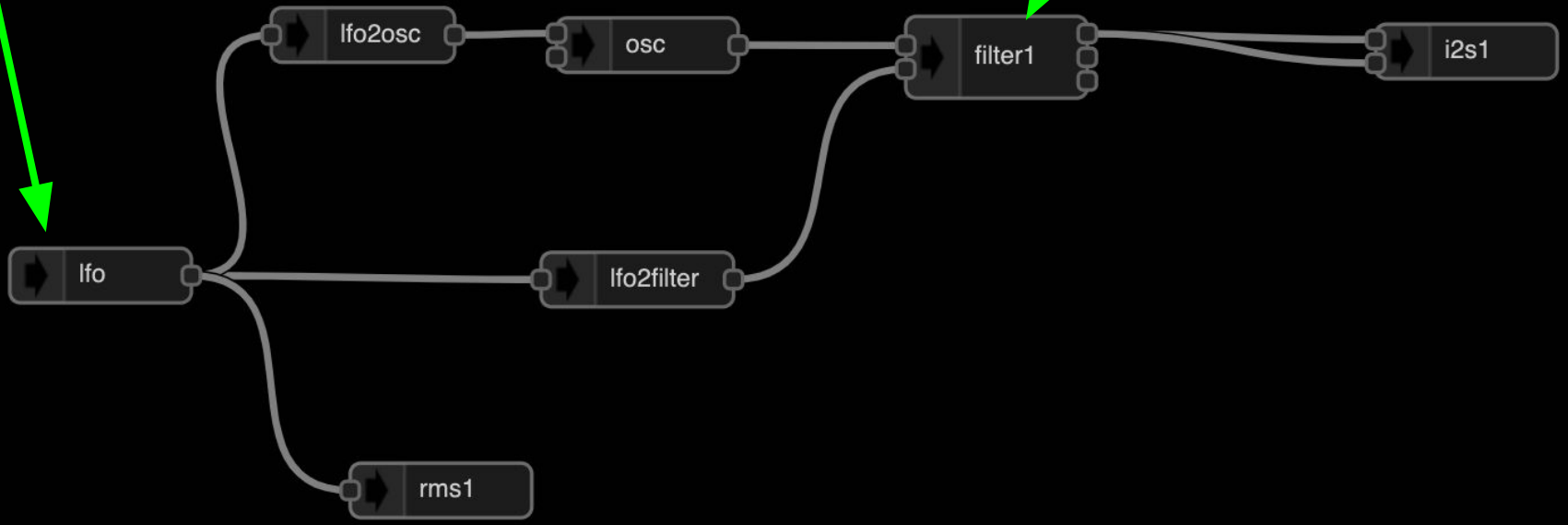
- Paste the code snippet

LFO: Low Frequency Oscillator

modulates the oscillator frequency and/or the cutoff frequency of the filter

Low Pass Filter

shapes the sound coming from the oscillator. Changes the timbre



Low Pass Filter

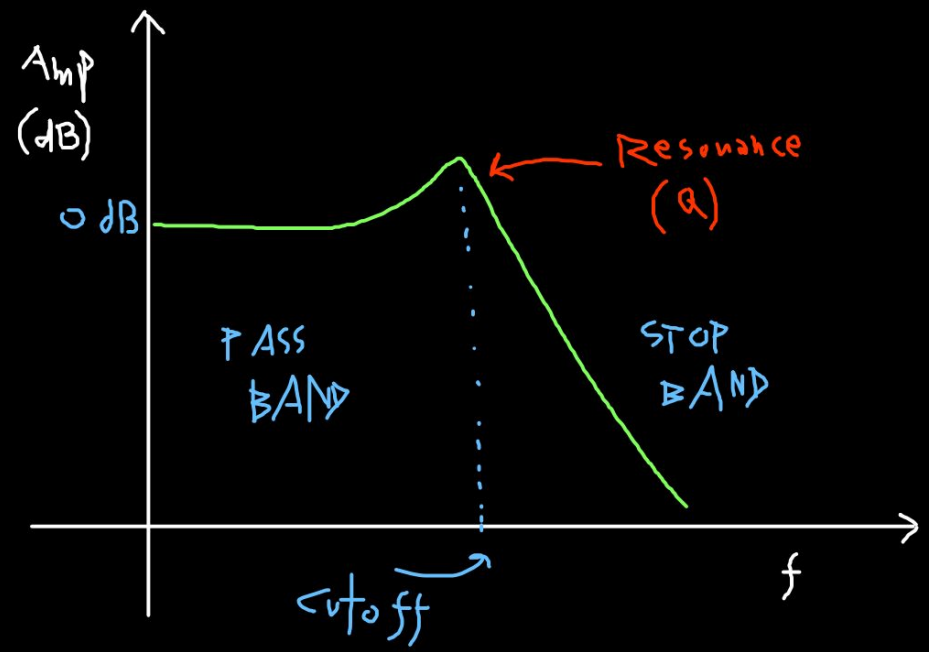
shapes the sound coming from the oscillator. Changes the timbre

> Initialize the Filter

> Connect a pot to pin 16

> Read the value

> map it and set the Cutoff Frequency



LFO: Low Frequency Oscillator

modulates the oscillator frequency and/or the cutoff frequency of the filter

> Connect a pot to pin 17

> Read the value

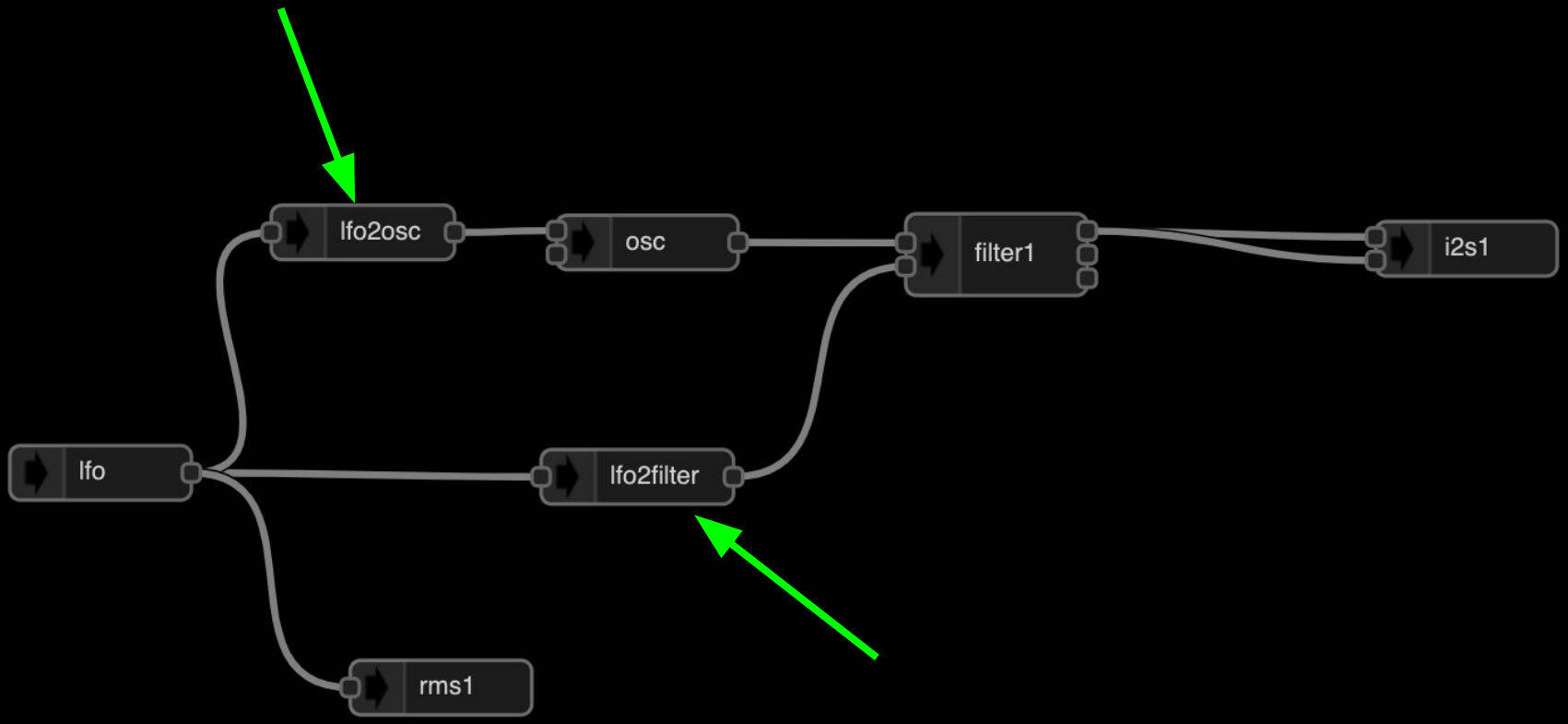
> Map it to the LFO rate

Switching the Modulations On/Off: Adding Buttons

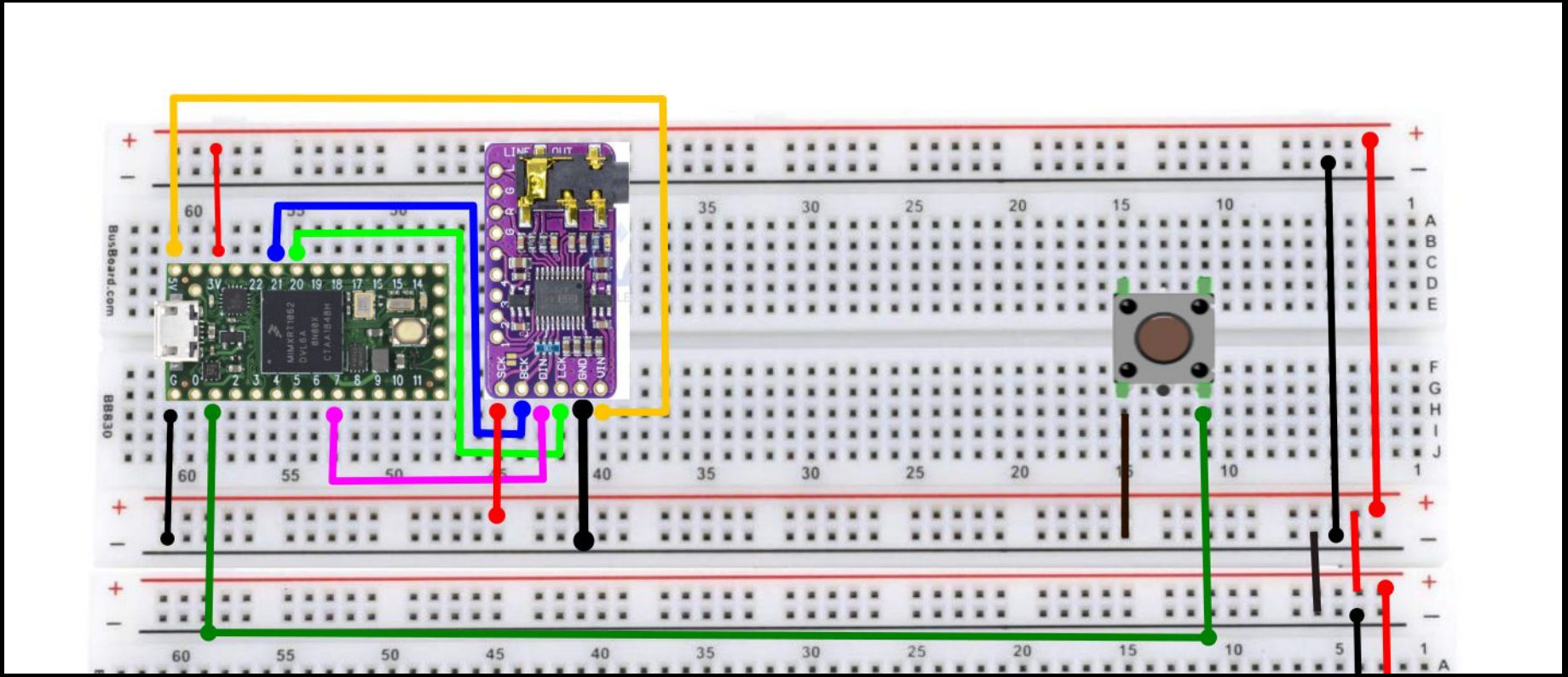
The LFO is connected to both the Oscillator frequency and the Filter cutoff.

We want to add 2 buttons to switch these connections on/off

The switches will control the two blocks in figure:



How to connect a push button



How to read a push button

Before Setup() add:

```
#include <Bounce2.h>

Bounce button1 = Bounce();
bool button1State = 0;
```

In Setup() add:

```
// setup button 1 on pin 1
button1.attach(1,INPUT_PULLUP);
button1.interval(25);
```

In the Loop() add:

```
// Read button status
 button1.update();
  if (button1.fell()){
   button1State = !button1State;
   Serial.print("Switch Button1: ");
   Serial.println(button1State);
   lfo2osc.gain(button1State);
 }
```

Adding the Random Sequencer

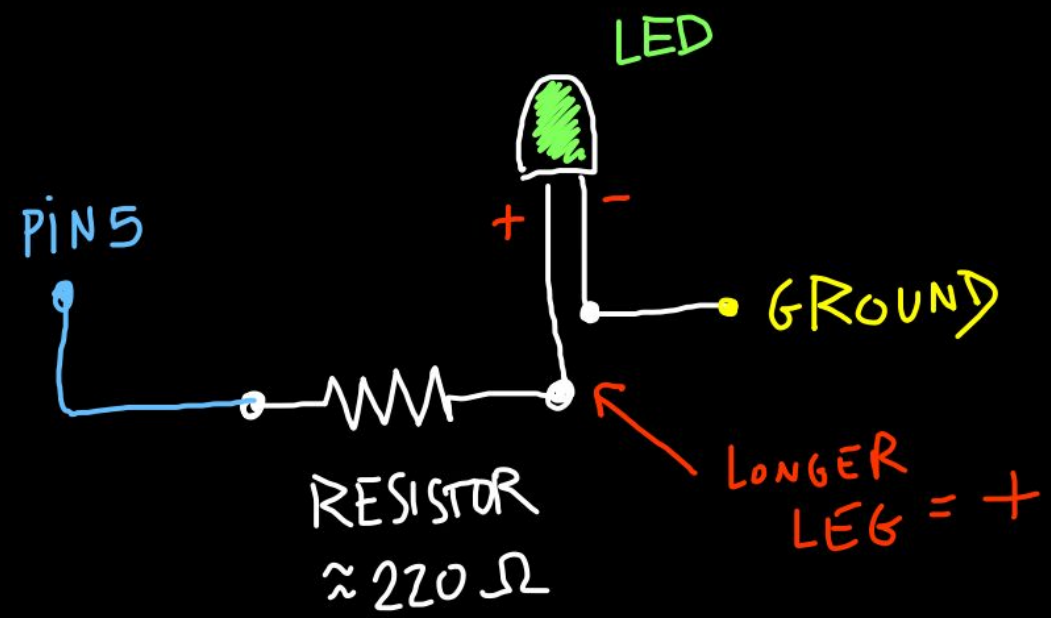
Before Setup() add:

```
// Sequencer
RandomPitchSequencer sequencer;
```

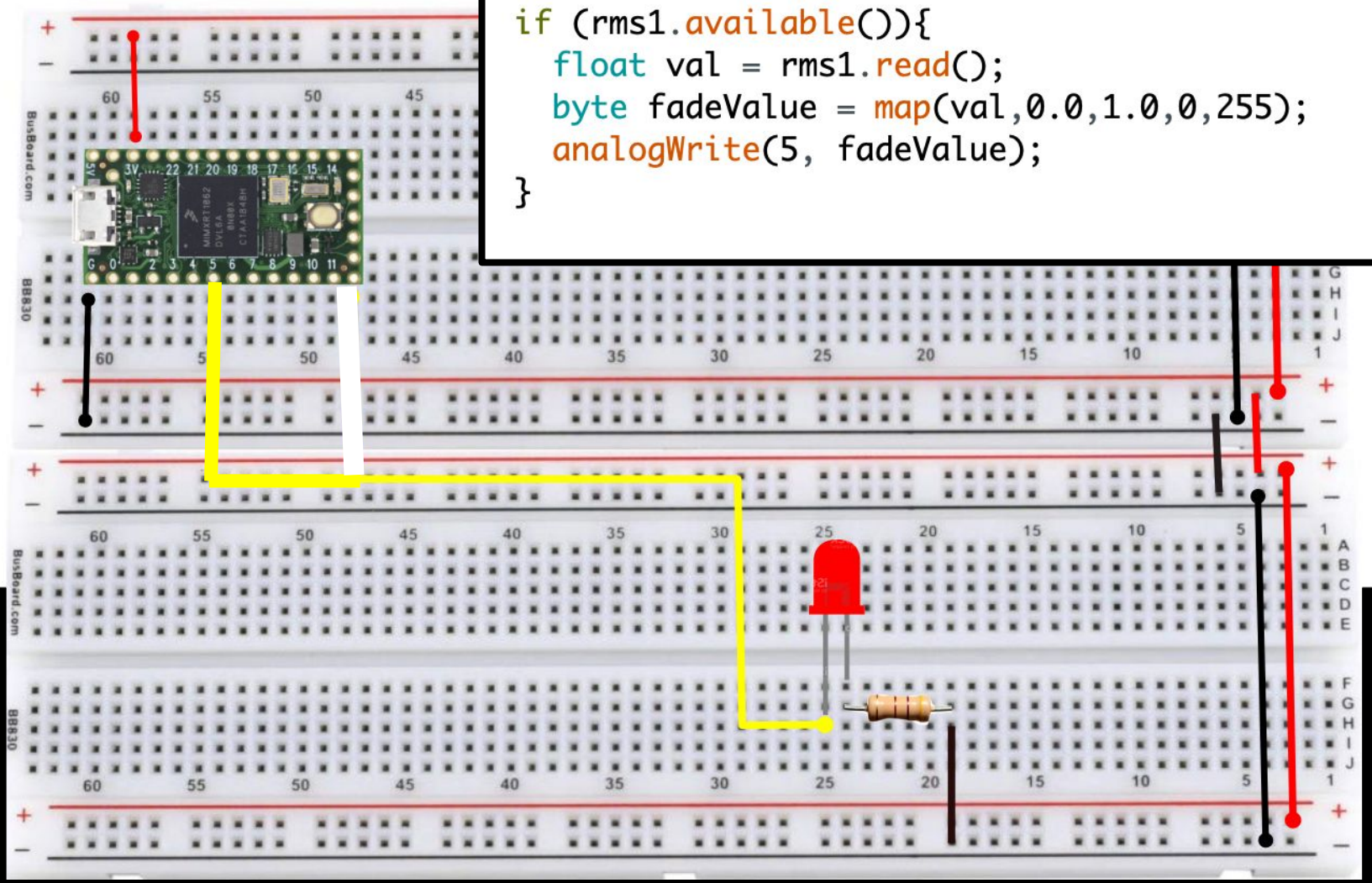
In the Loop() add:

```
// declare the frequency ratio variable and initialize it
 float fratio = 1.0;

// read sequencer rate from knob at pin 18
 float seqrate = map( readKnob(18),0.0, 1023.0, 0.1f, 16.f );
 int noctaves = map( readKnob(19),0.0, 1023.0, 0, 6 );

// disables the sequencer if the octave range is at zero
 if (noctaves>0){
   fratio = sequencer.GetPitch(seqrate,noctaves);
 }
```

Bonus: How to connect a LED and control it



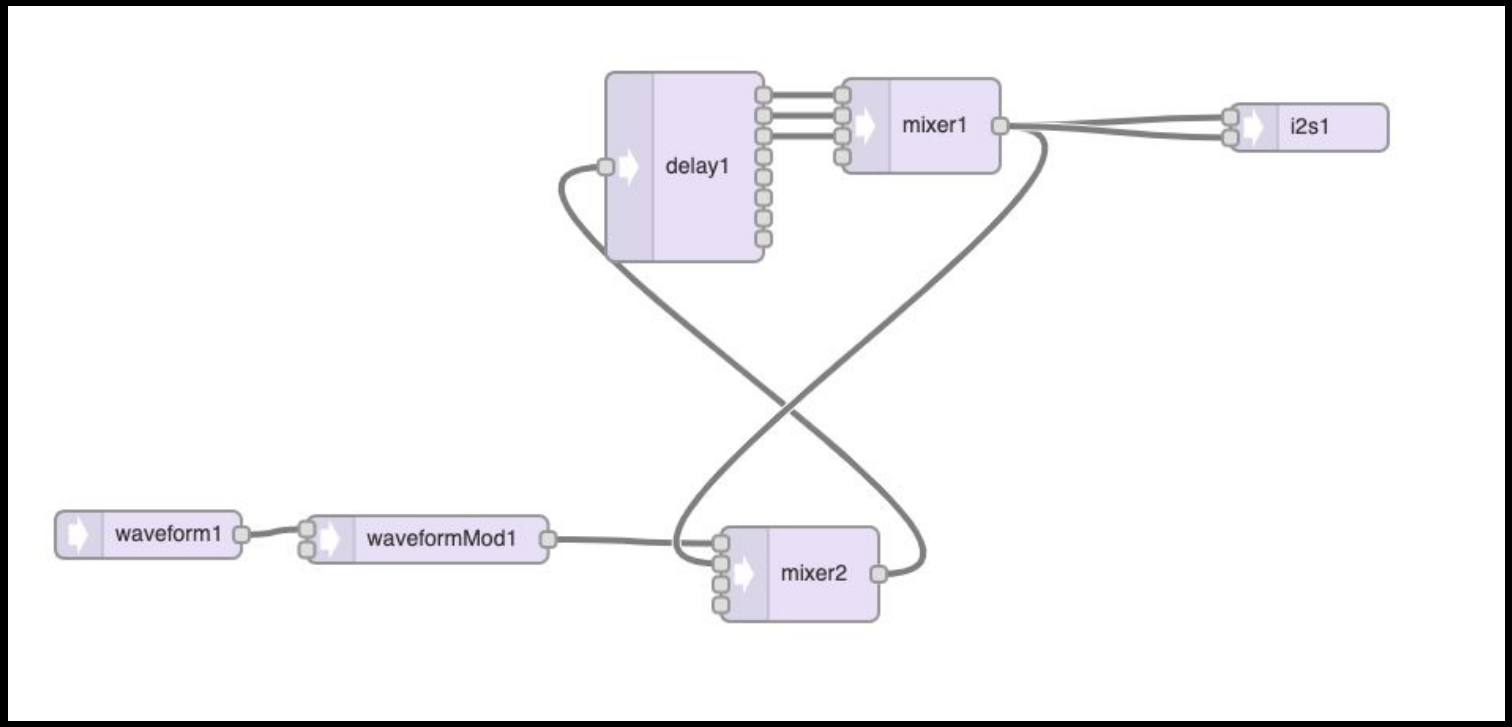
Bonus: How to connect a LED and control it



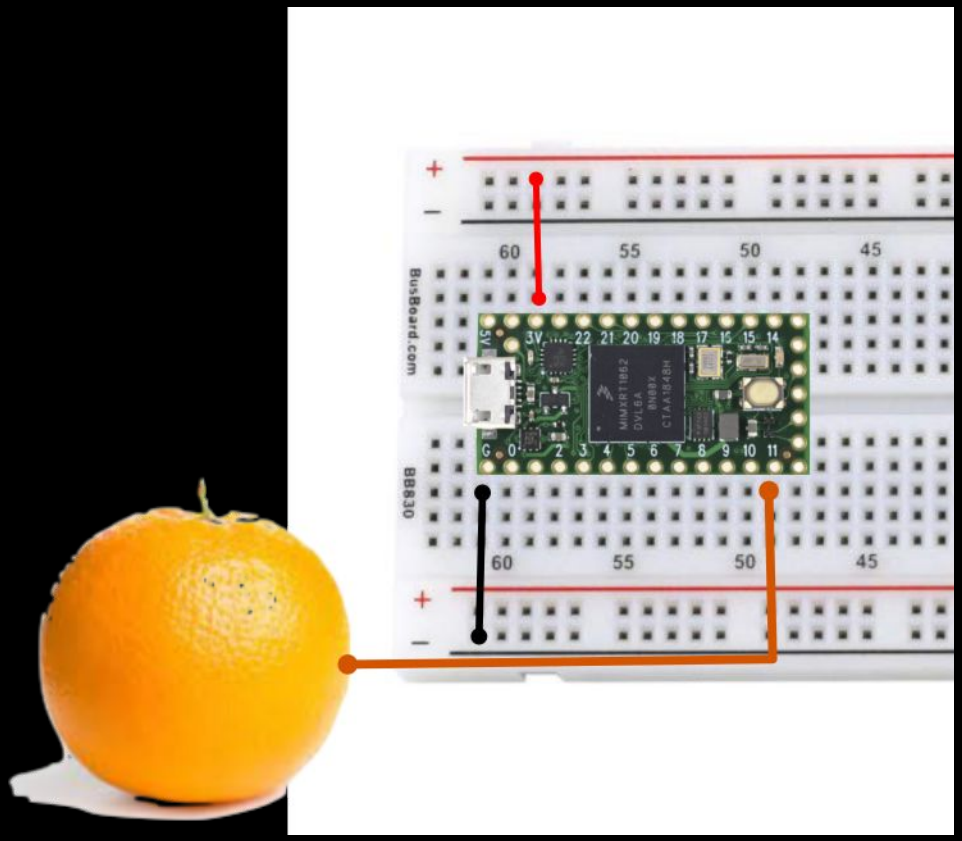
```
// control LED with LFO
if (rms1.available()){
  float val = rms1.read();
  byte fadeValue = map(val,0.0,1.0,0,255);
  analogWrite(5, fadeValue);
}
```

More Bonuses:

Adding a Delay FX



Bonus: How to connect a Touch sensing pad



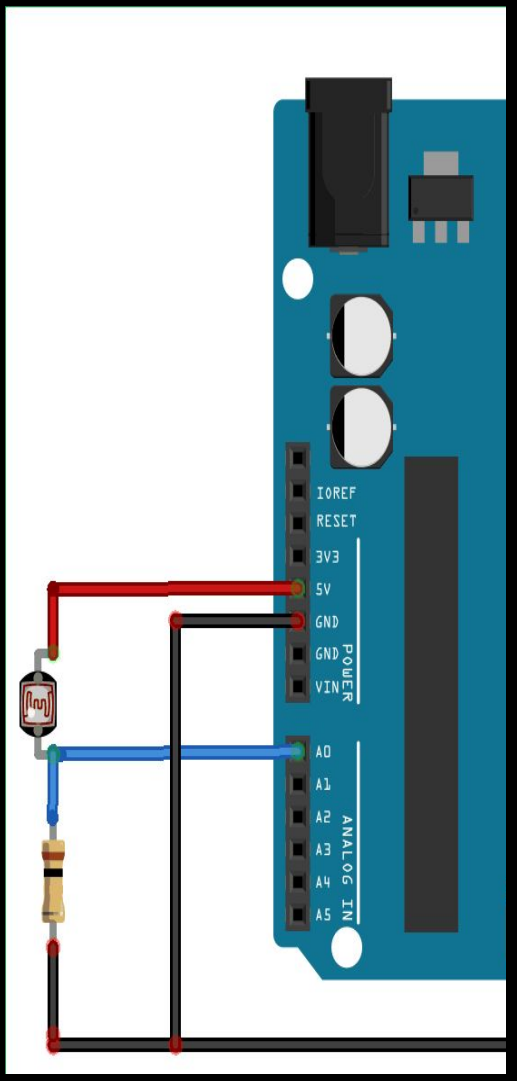
[see code in extra folder]

Connecting a photosensor

A photoresistor is a light-dependent resistor

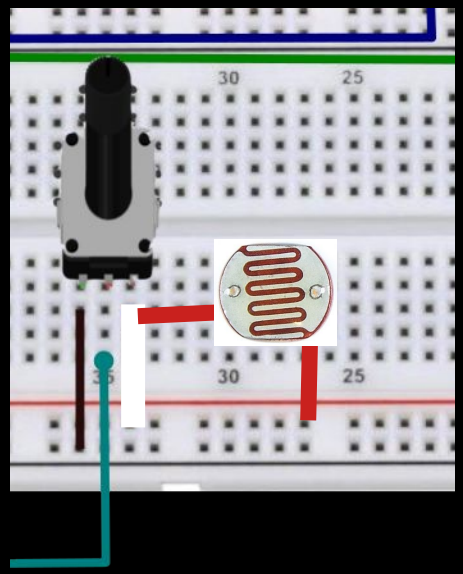
With the proper circuit, we can convert the light signal into a voltage signal and read it through an analog input.

1 K resistor



Connecting a photoresistor

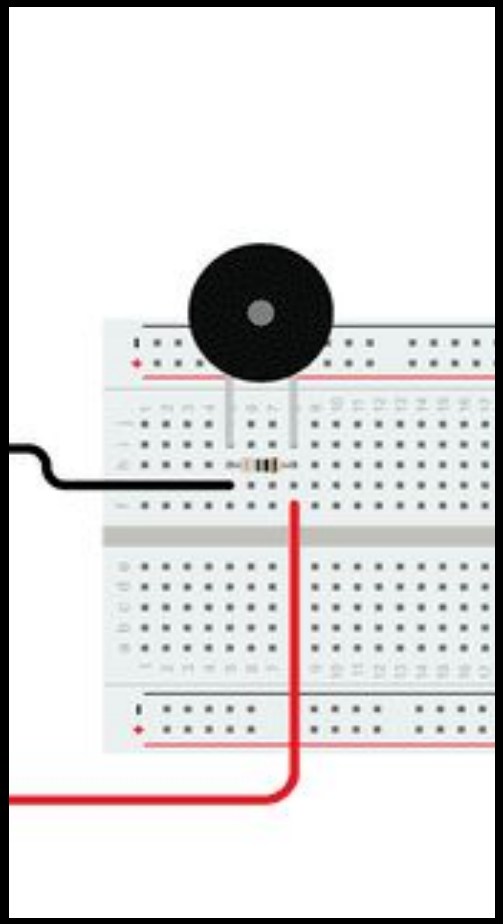
→ You can modify the synth circuit by inserting a photoresistor between pin 3 of a potentiometer and +3v



Connecting a Piezo

- Piezos are **polarized**, meaning that voltage passes through them (or out of them) in a specific direction. Connect the black wire (the lower voltage) to ground and the red wire (the higher voltage) to an analog pin

- Additionally, connect a 1-megohm resistor in parallel to the Piezo element to limit the voltage and current produced by the piezo and to protect the analog input.



Source:

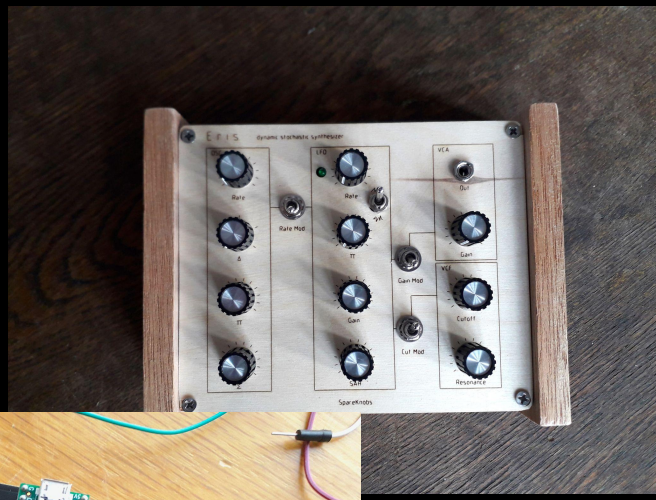
https://docs.arduino.cc/built-in-examples/sensors/Knock

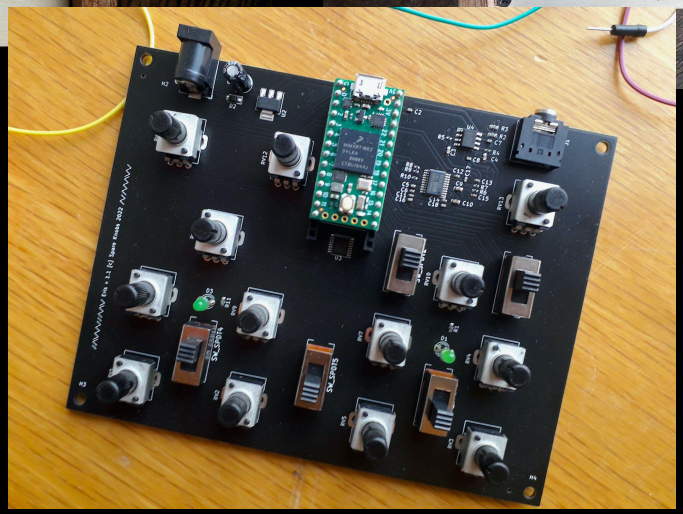
More Bonuses:

drum machines, envelopes, MIDI…
(see extra folder)

Where to go from here…

> From breadboard to protoboard
> Soldering stuff
> Enclosure
>...





References, Links, Help…

DIY Synth FB group
https://www.facebook.com/groups/805860653977158

Teensy Forum
https://forum.pjrc.com/

Music From Outer Space
http://www.musicfromouterspace.com/