

# 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

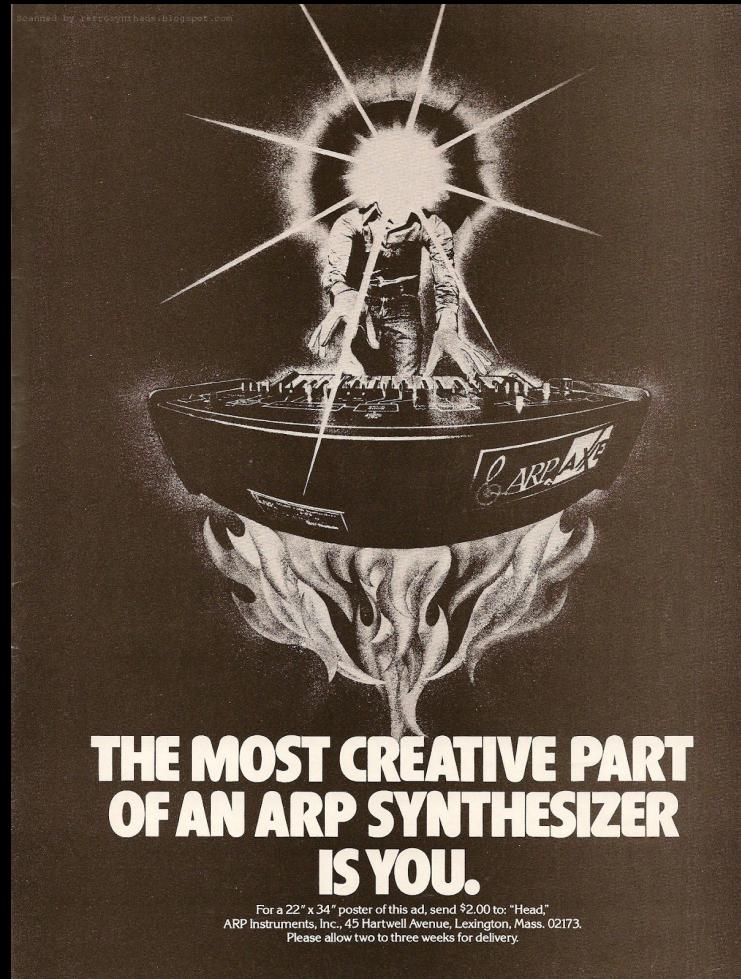
Setting Up the Breadboard

Introducing the Audio System Design tool

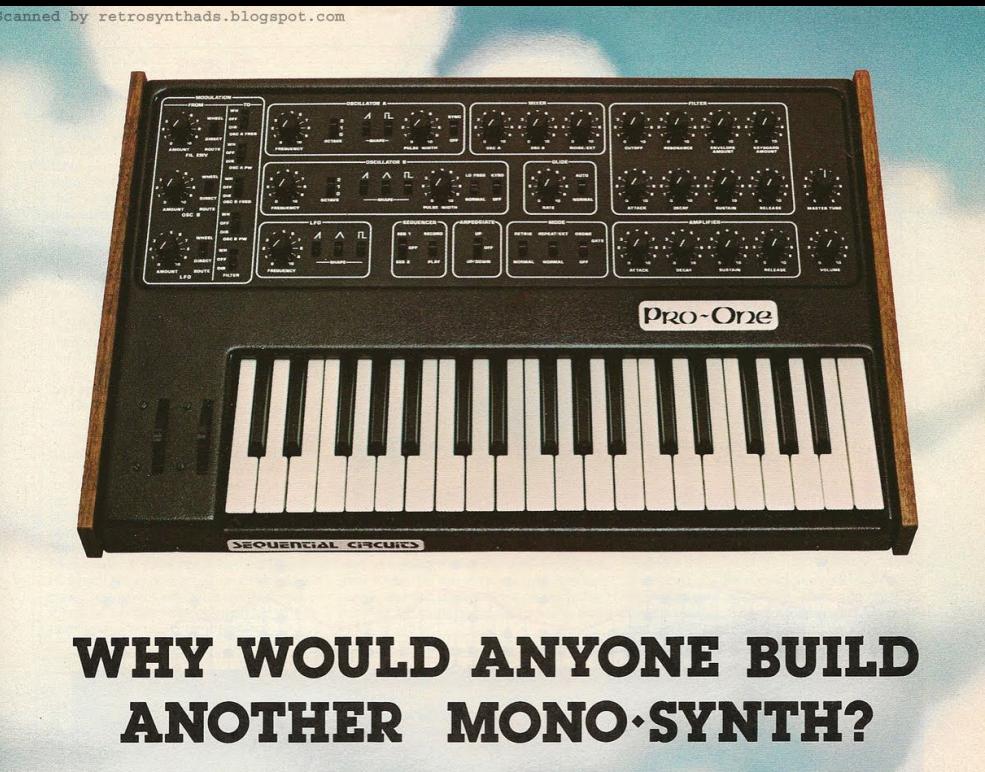
Our first sound!

Taming the Synth: Adding Knobs

# Intro



Scanned by retrosynthads.blogspot.com



Setting up the workbench

# Setting up the workbench: Arduino IDE

## 1. Connect to the Internet

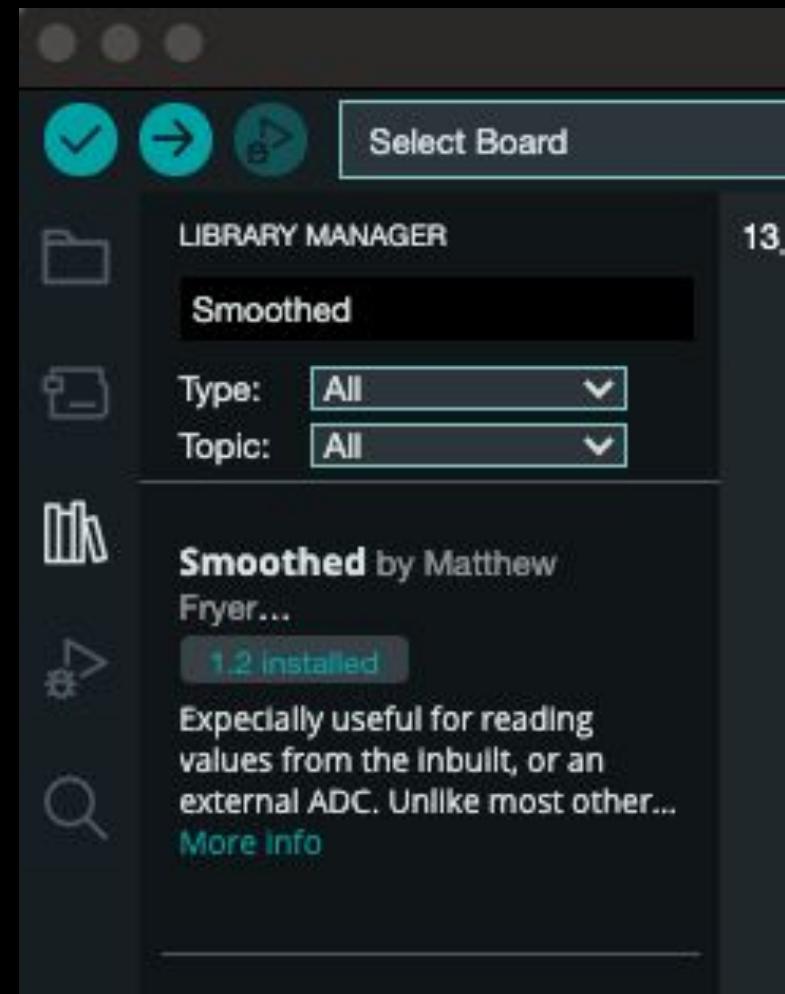
Wifi: WD4X Guest

Password: wifipasswordfortheguests

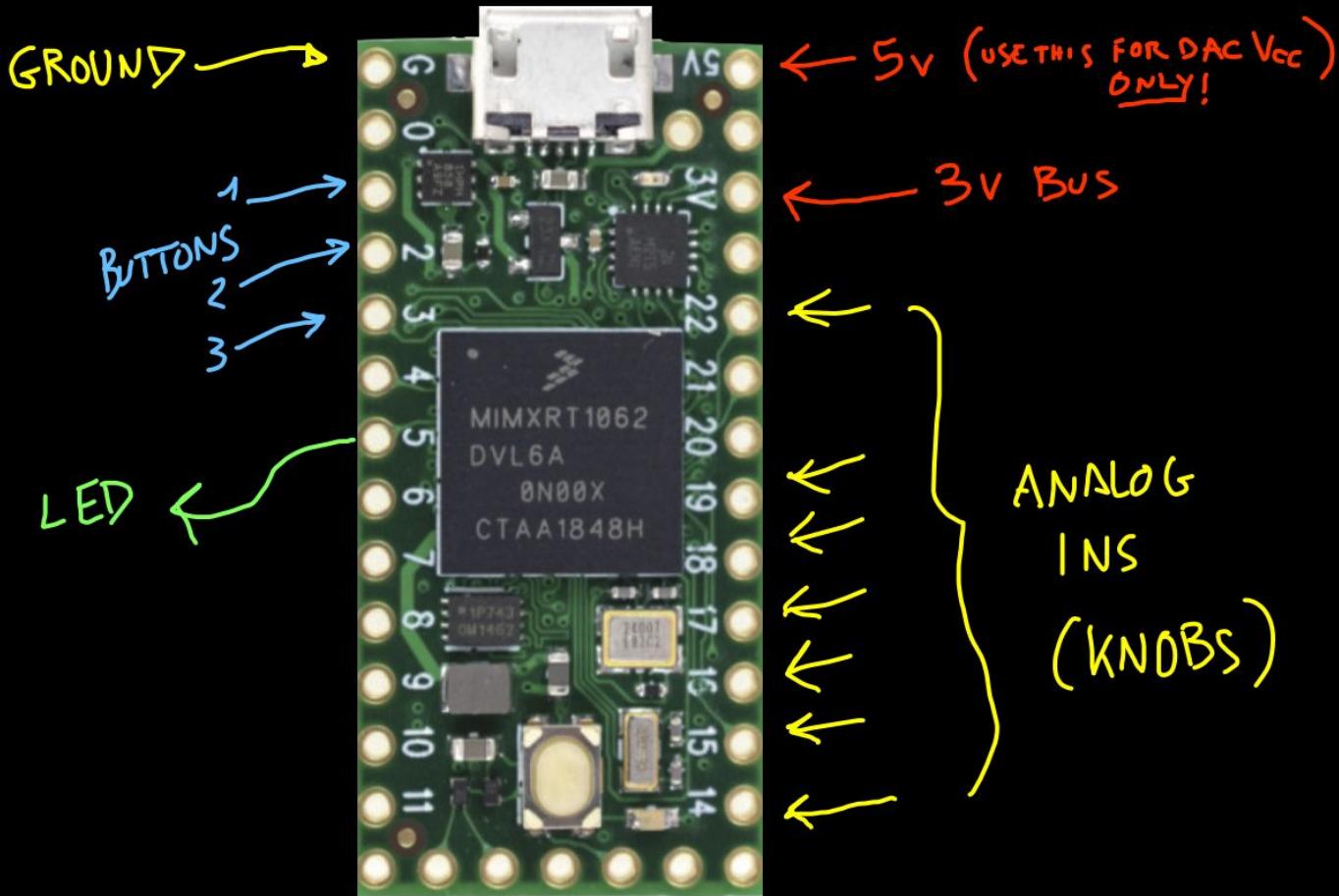
## 2. Install Arduino IDE + Teensy board libraries following these instructions: [https://www.pjrc.com/teensy/td\\_download.html](https://www.pjrc.com/teensy/td_download.html)

## 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

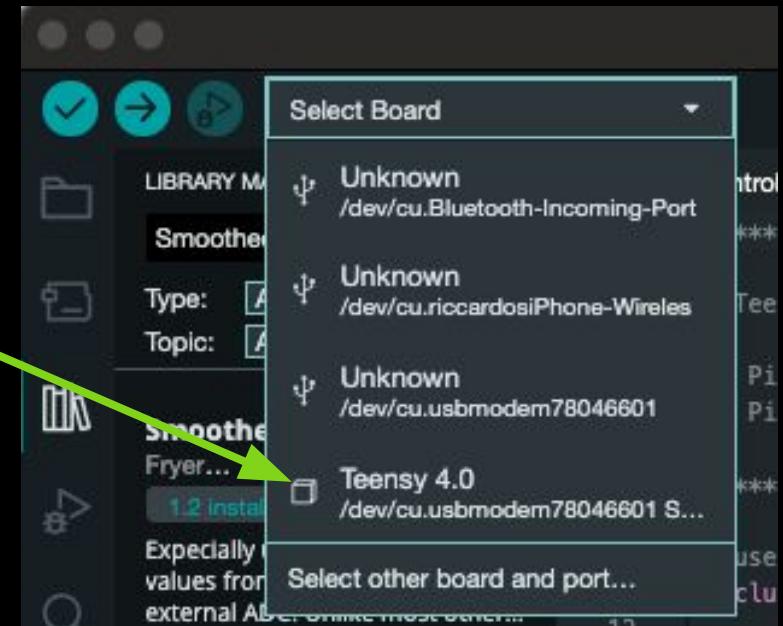
# Connect Teensy to Arduino IDE

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

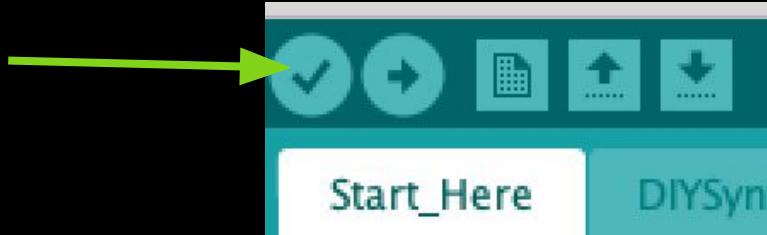
Unzip it in your favourite spot

[ 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

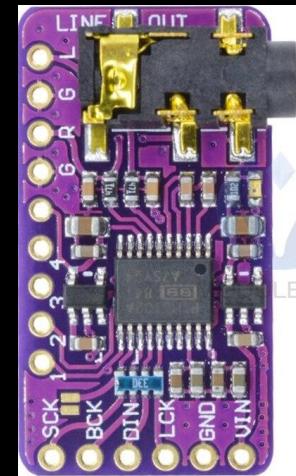
Scanned by retrosynthads.blogspot.com

A MUSICIAN'S WORK  
SHOULD BE ALL PLAY.

THE "T" STANDS FOR TOUCH.  
The new, fully programmable, eight voice Prophet-T8 is the most responsive touch-sensitive synthesizer available.  
The difference begins with the new, extended 76 note velocity range. As each key is depressed, the resulting modulation is individually

# The BreadBoard

TEENSY



DAC (AUDIO OUT)

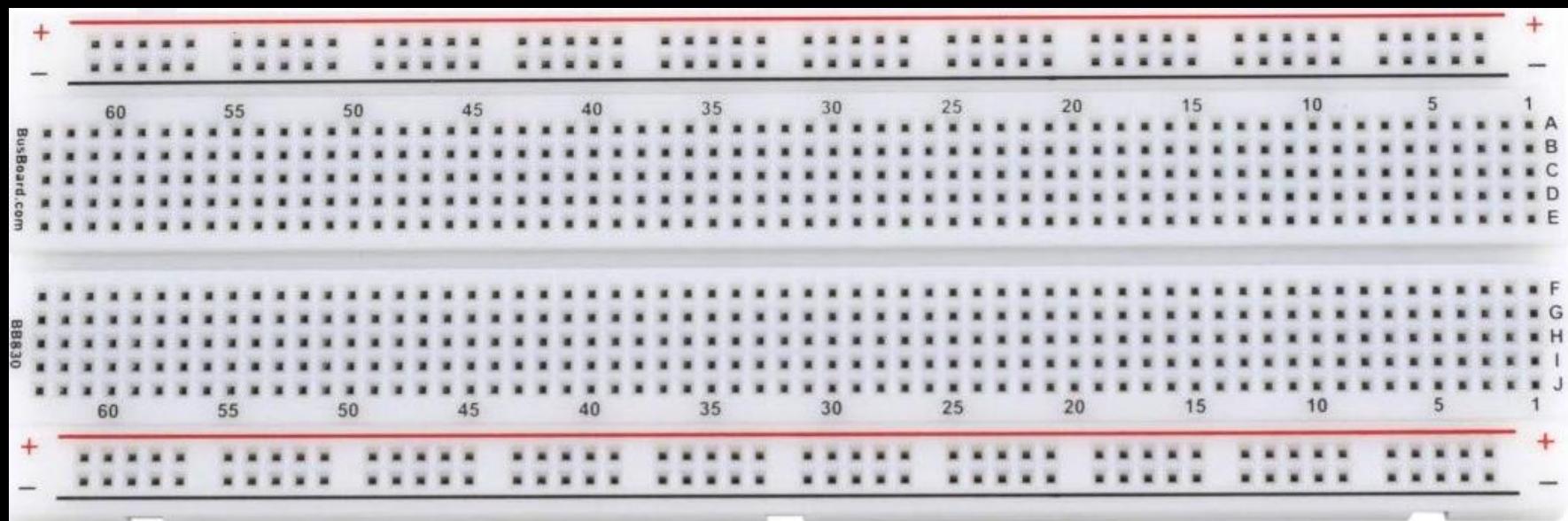
POTENTIOMETER



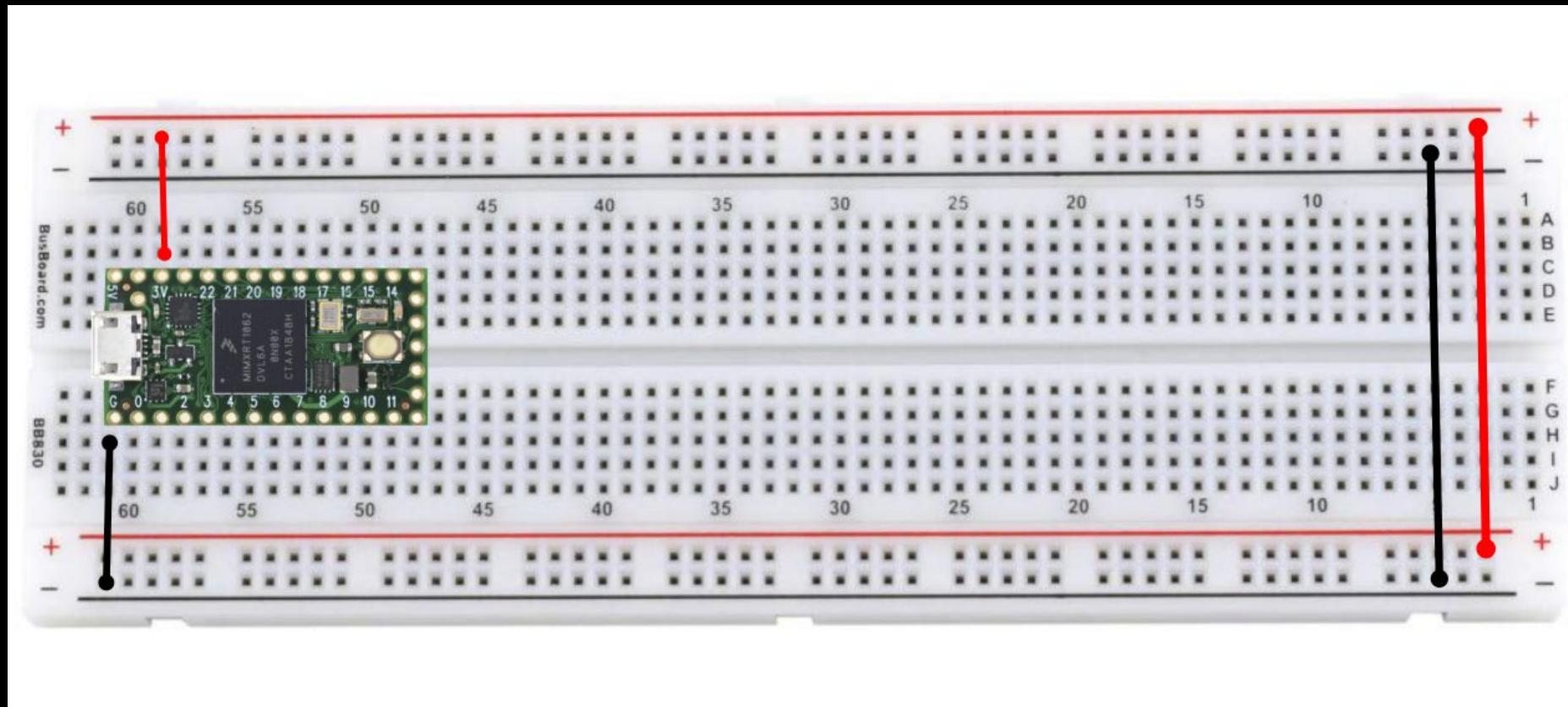
BUTTON



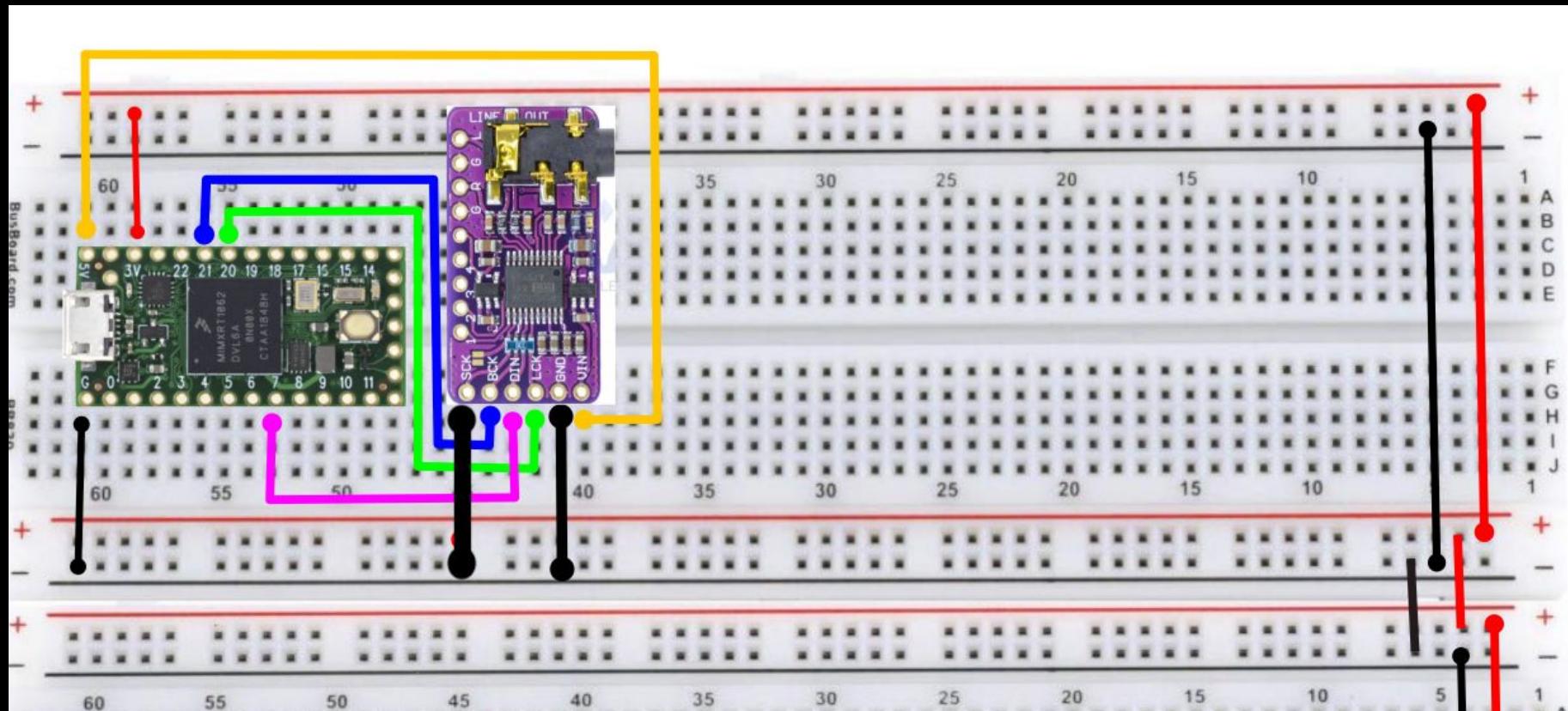
BREADBOARD



## 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

Scanned by retrosynthads.blogspot.com

**The ultimate in polyphonic control.**

Since E $\mu$  Systems introduced the 4060 microprocessor-based 16 channel keyboard/sequencer in 1977, it has represented the state of the art in sophisticated synthesizer controllers. It has been the overwhelming choice of recording studios, television music makers such as Herbie Hancock, Daryl Dragon, Pat Glaser, and Frank Zappa. Today it stands at the center of the most powerful system of electronic music hardware and software available to the serious synthesist.

The basic keyboard will control 16 independent synthesizer channels with programmable control of keyboard split, glide, transposition, and channel assignment mode. The built-in 16 channel digital memory controller allows the creation of multi-channel compositions with up to 6,000 notes in real time—without the need for a multi-track tape recorder. The integral tape interface allows the storage of sequences in digital form on standard audio tape. Programs from our special-function

software library add such capabilities as editing and merging polyphonic sequences, redefinition of keyboard tuning, non-realtime sequence generation, and interfacing to video synthesizers.

Our new 4070 floppy disc system allows fast, convenient storage and recall of sequences and special function software. Each 8" disc will hold six full sequences in memory, for a total capacity of 36,000 notes. Also included is a CRT terminal interface for use with forthcoming software.

For more information about the 4060 keyboard, the E $\mu$  modular synthesizer, and the incredible new AUMA 16 channel digitally controlled polyphonic synthesizer system, send \$10.00 for the Emu Systems Technical Catalog with photos, specifications, and functional descriptions of all E $\mu$  products. (Calif. residents be sure to add sales tax.)

**Emu Systems**  
INC.  
417 BROADWAY  
SANTA CRUZ, CA 95060  
(408) 429-9147

# 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);  
    ^  
    output (return value)    ^  
                           input (argument)
```

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);
```

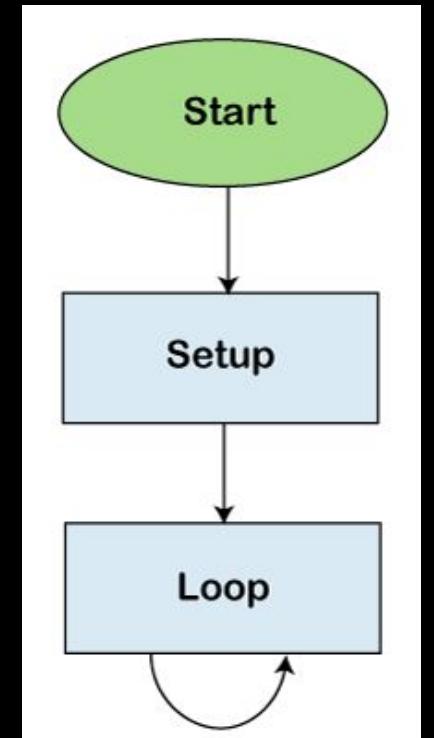
# The Boring Life of a Teensy

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.

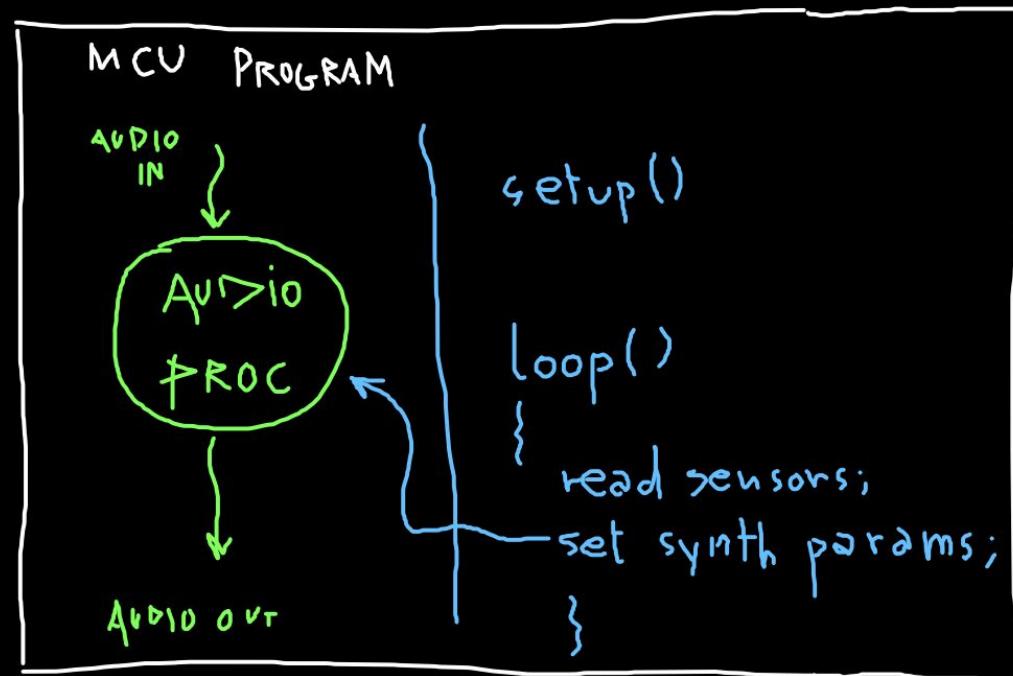


# How Teensy makes sound

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

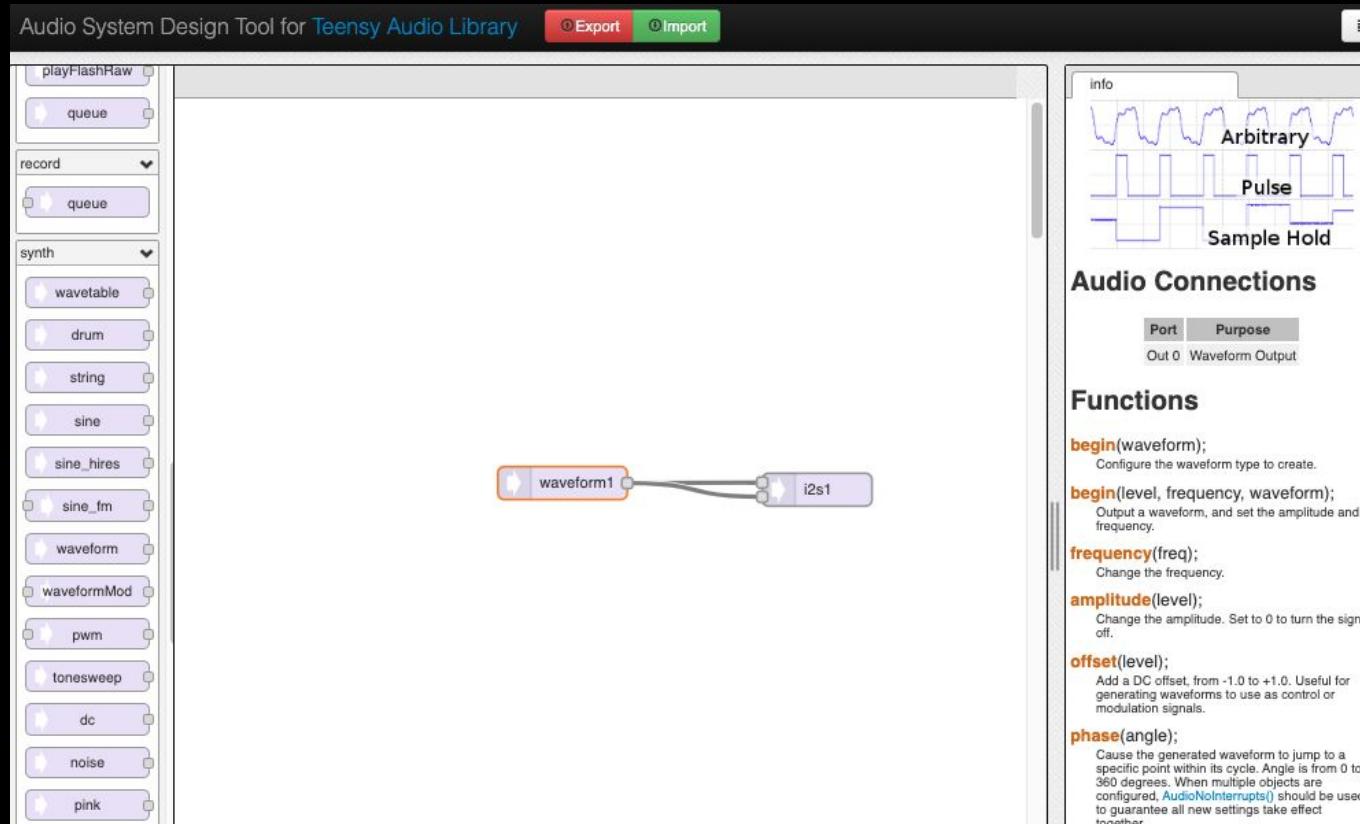
Luckily, the audio part is managed internally and we don't have to think about it.  
What is left to us is to patch together the audio modules and then control them in the `loop()`



# How to program Teensy for Audio Generation

*Teensy provides a visual programming environment to patch synthesizers and effects*

→ Open browser, go 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 SYNTI

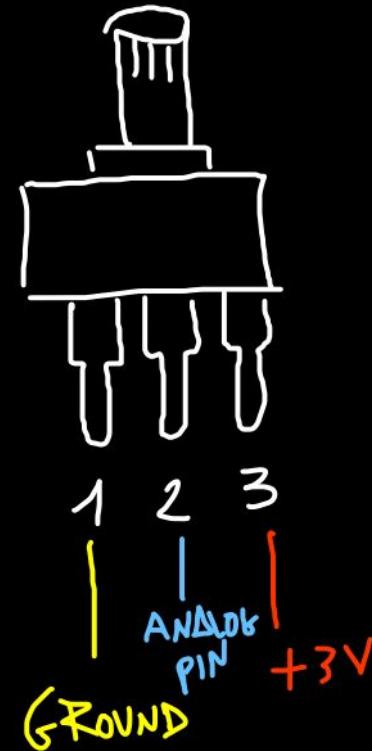
Electronic Music Studios (London) Limited  
49 Deodar Road London SW15 01-874 2363  
New York 408 East 78th Street N.Y. 10021 USA

The Synthi Range by **EMS**

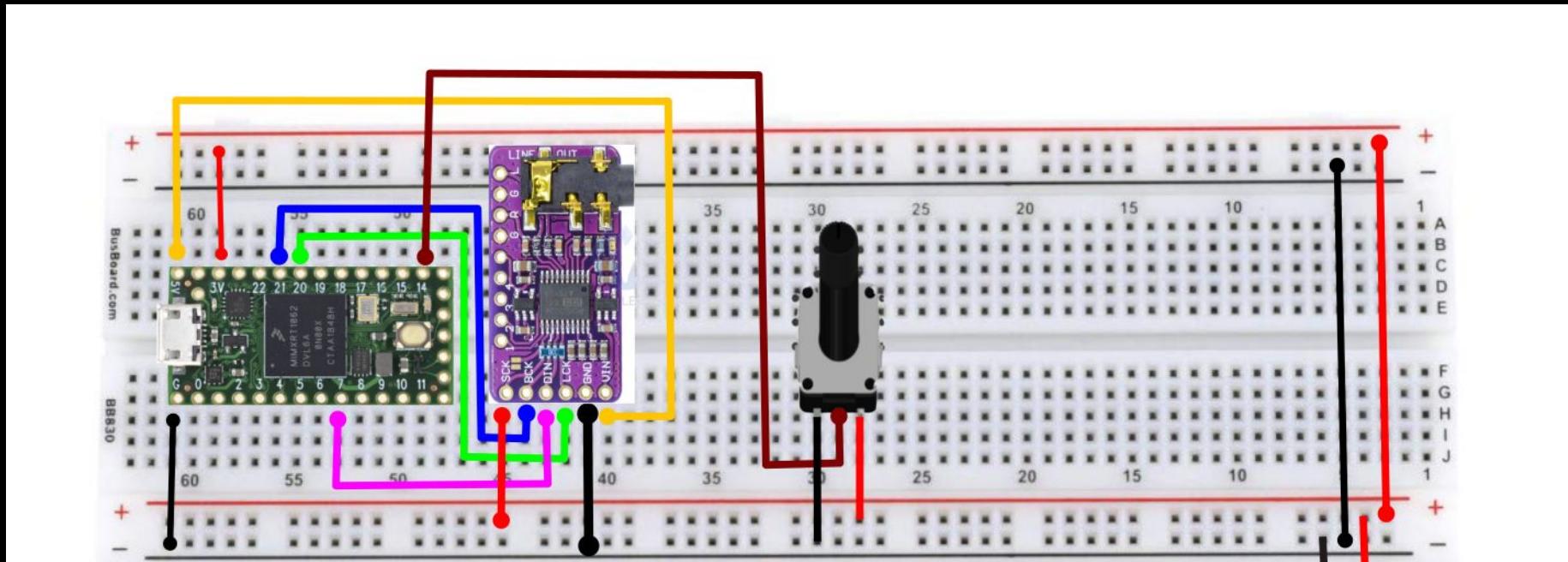


Control the frequency of the oscillator with a knob

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 console:

```
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 );
```

scanned by retrosynthads.blogspot.com

# WHY WOULD ANYONE\* WANT TO BUY THIS UGLY, MONOPHONIC SYNTHESIZER?

When you've got a synthesizer with endbells that look like deflated Uniroyals and a front panel design that could double for a rat maze in some scientific research program; yet professional musicians, songwriters and producers are using it to create hit songs . . . What's the story?

**The Oscar sounds great!** Digital Oscillators, Complex Sound Generation, Multi-Mode Filtering, Digital Sequencing, Programmable Arpeggiator, MIDI (In, Out & Thru), and the **Baddest Bass Sound You've Ever Heard** are some of the reasons this British synth is being used for both stage and studio applications. (Just think of it as the ugliest band member who gets that great sound.) If you haven't heard the Oscar, it's not too late; see your local music dealer for a demonstration.



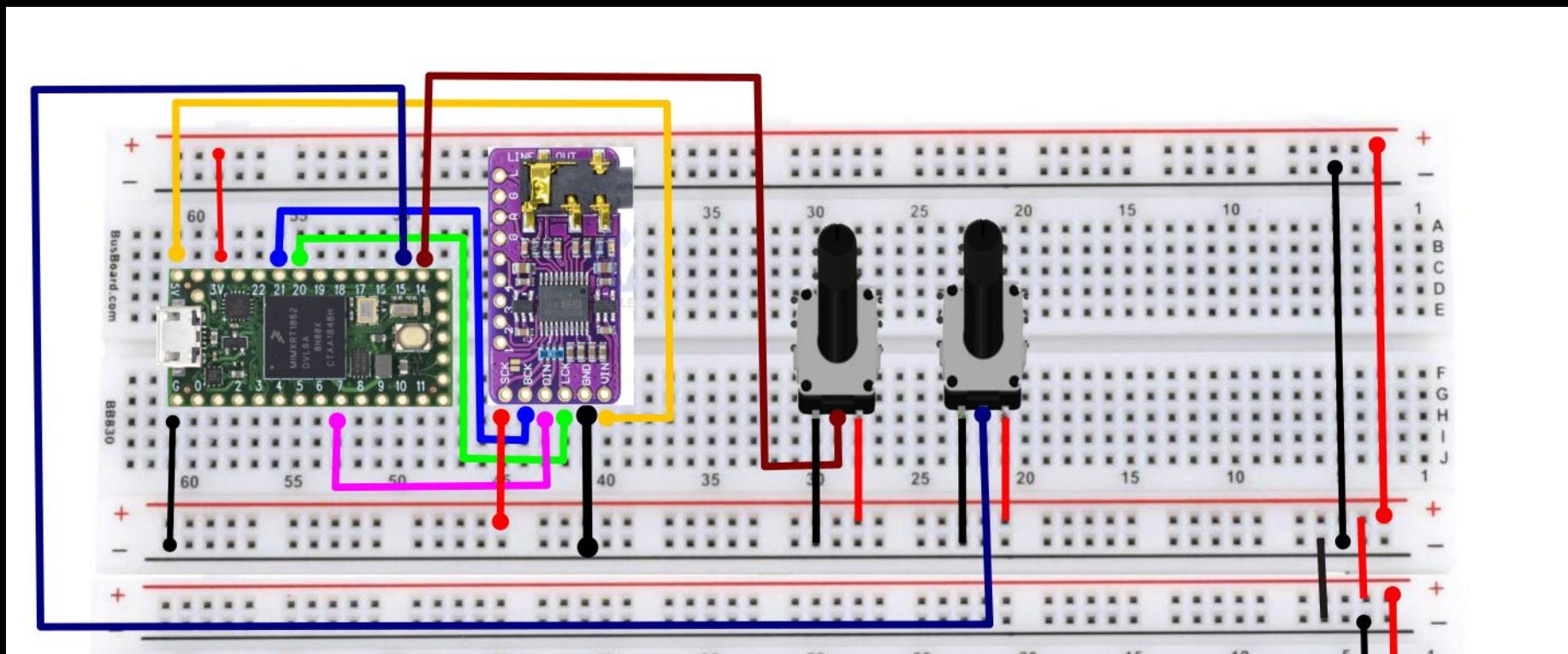
*Euryna*  
EUROPA TECHNOLOGY, INC.

1638 W. WASHINGTON BLVD., VENICE, CA 90291 • 213-392-4985  
TELEX: 506162  
CALL OR WRITE FOR MORE INFORMATION.

\* Go West  
Ultravox  
Asia  
Dead or Alive  
The System  
Michael Boddicker  
*SYNTHESIST*  
Duane Hitchings  
*SONGWRITER*  
Dave Holman  
*PRODUCER*  
John Farrer  
*PRODUCER*  
Unique Recording  
Studio

## 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 );
```



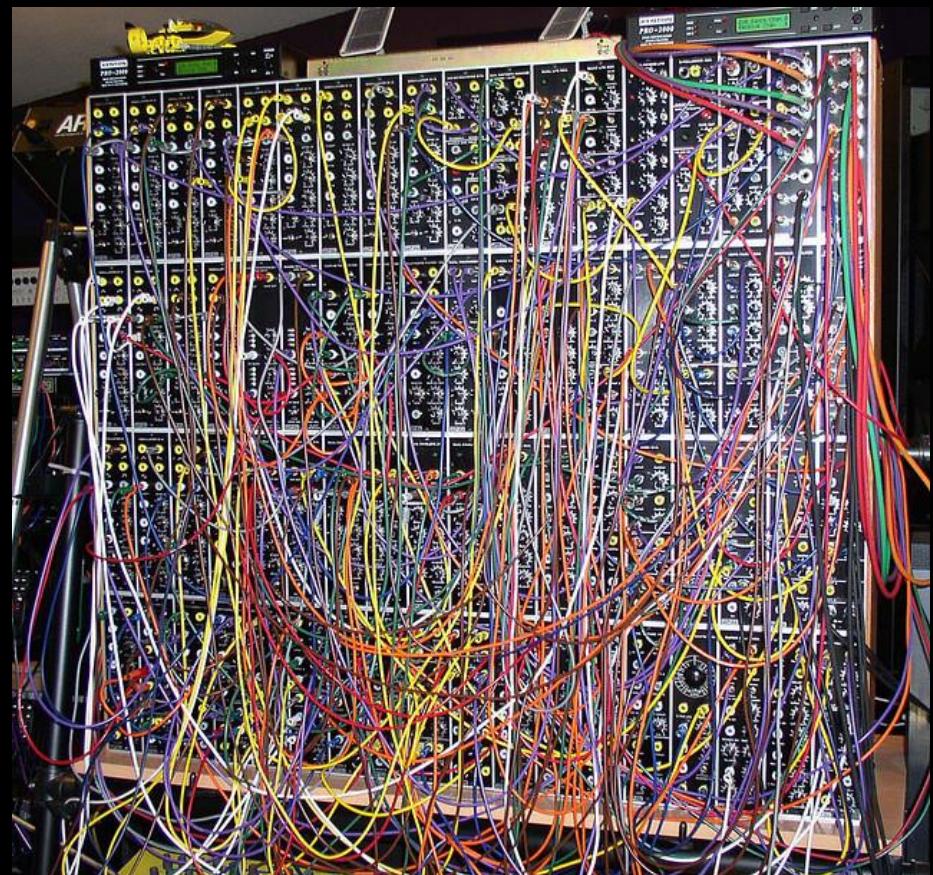
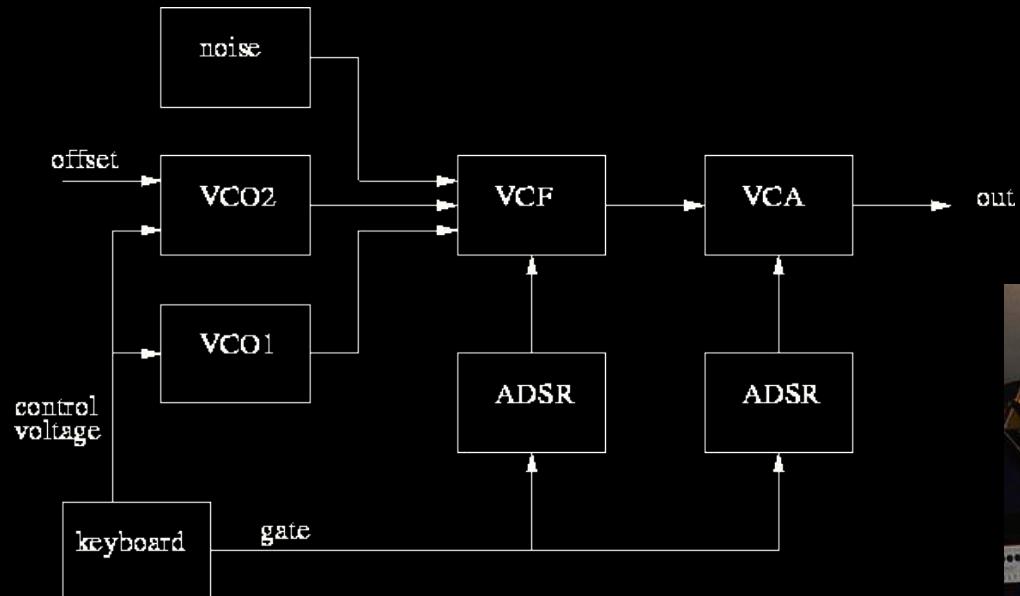
*In the next episode:*

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

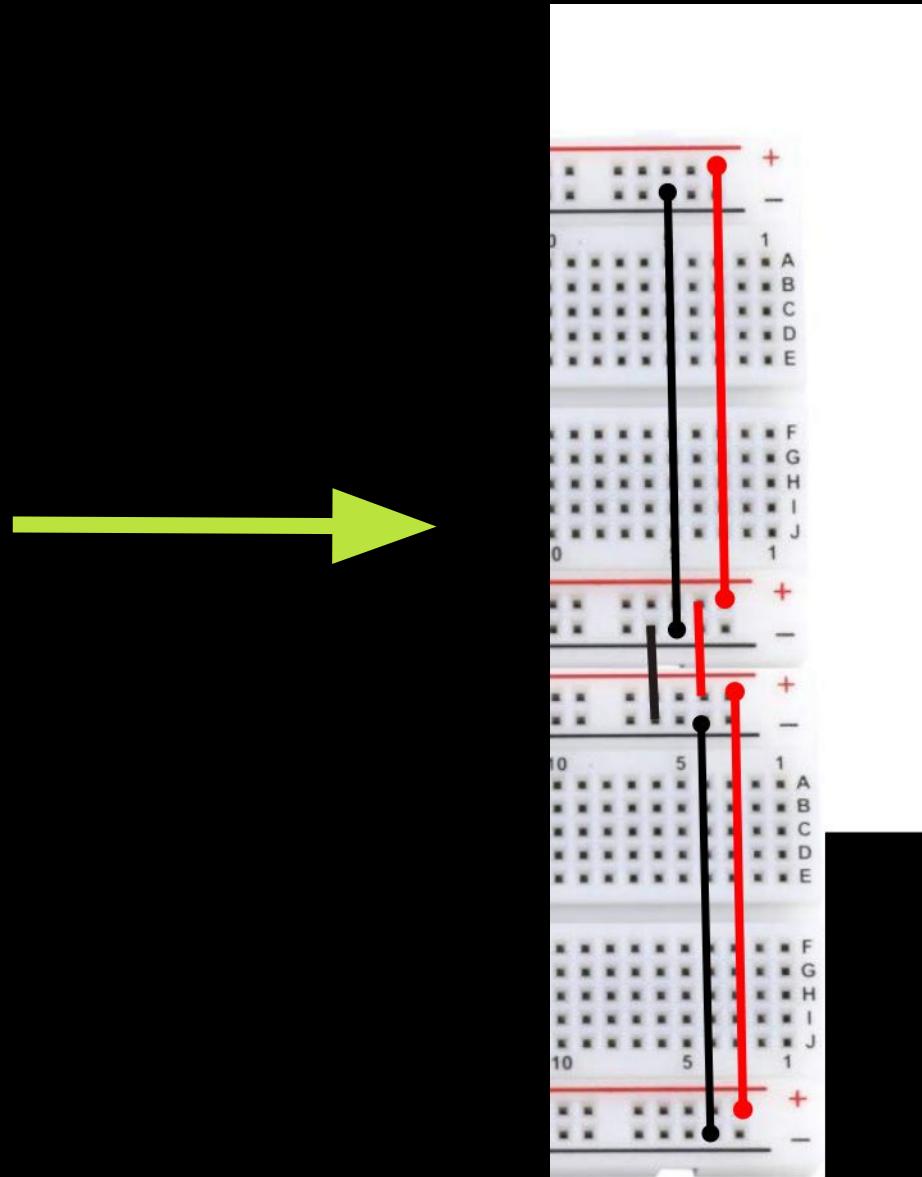
# Day #2 Plan

- 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

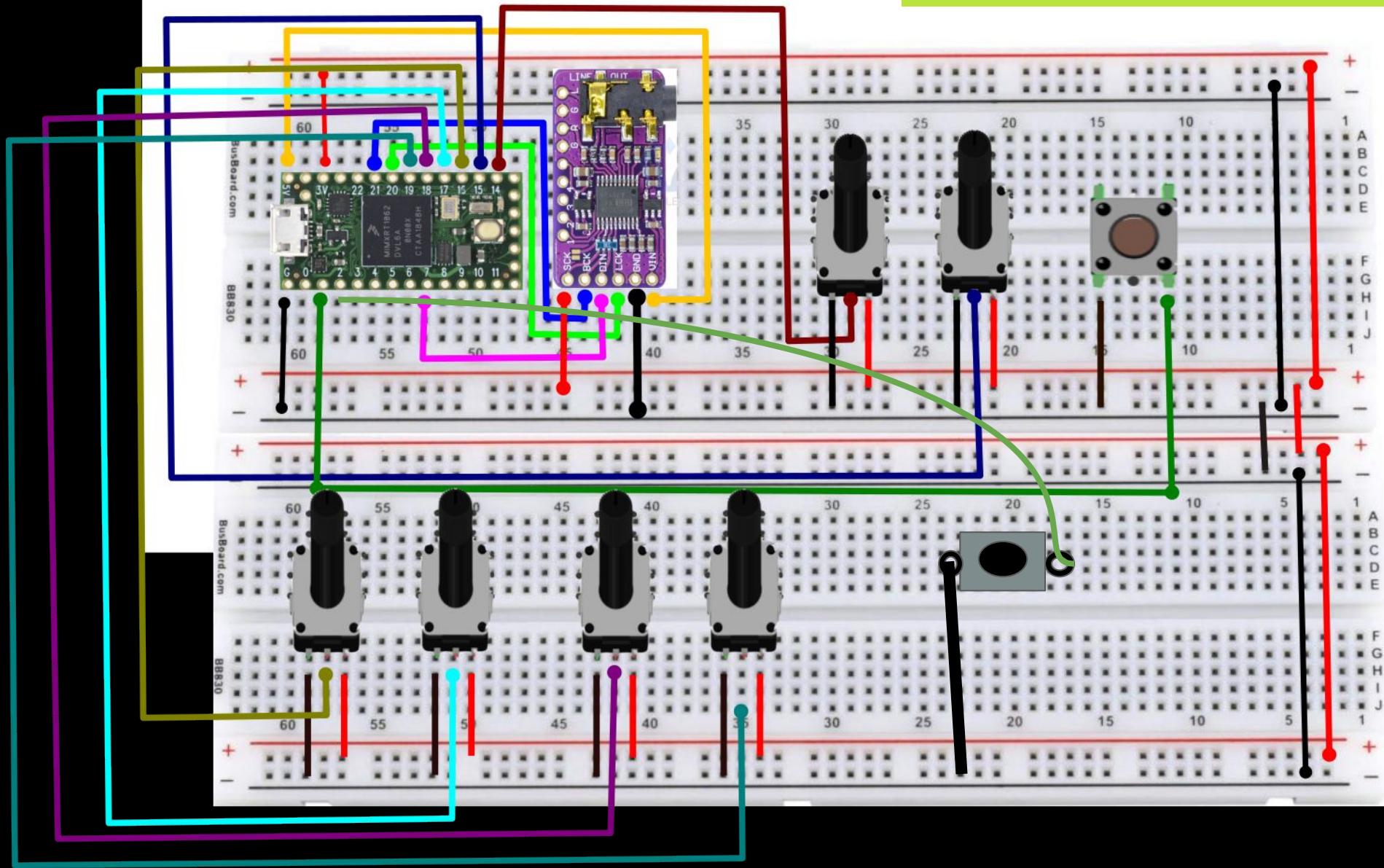


## Extending the Board – Connecting 3v and Ground Buses



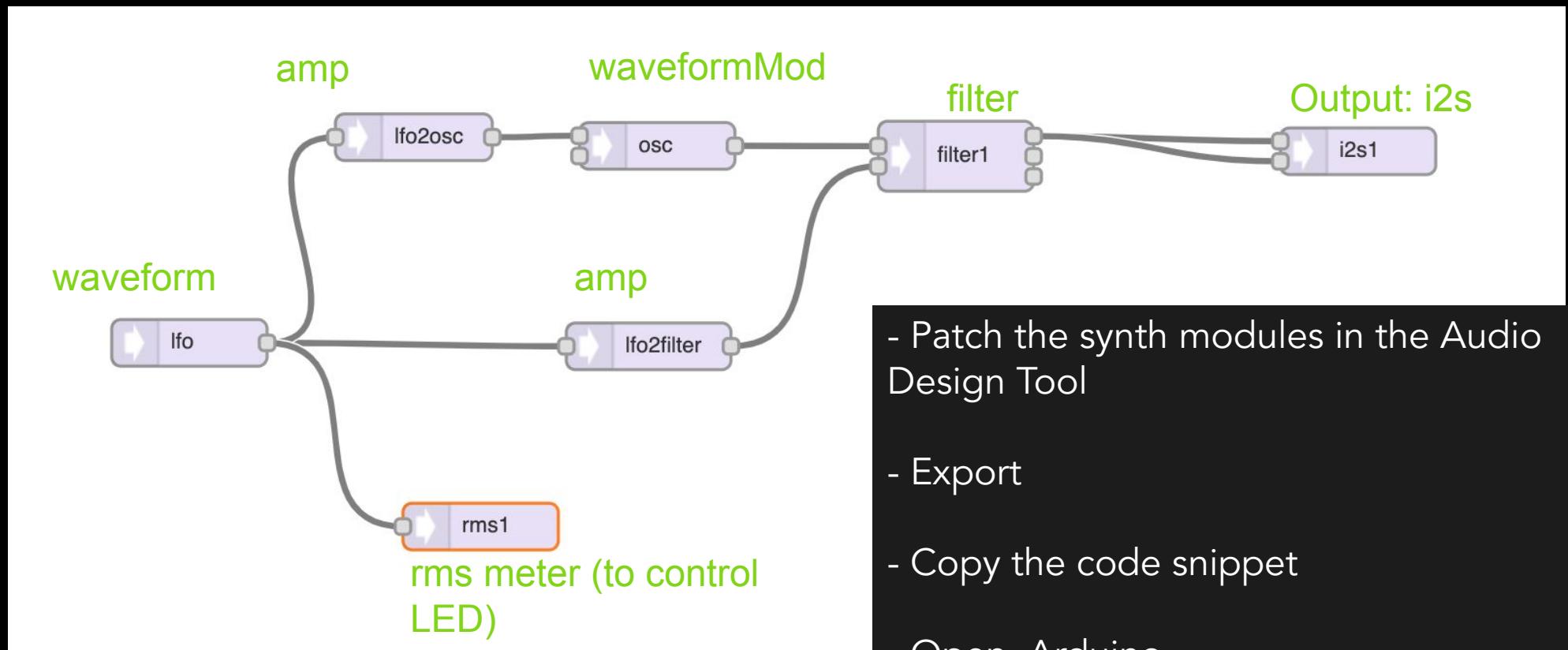
# More Pots!

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



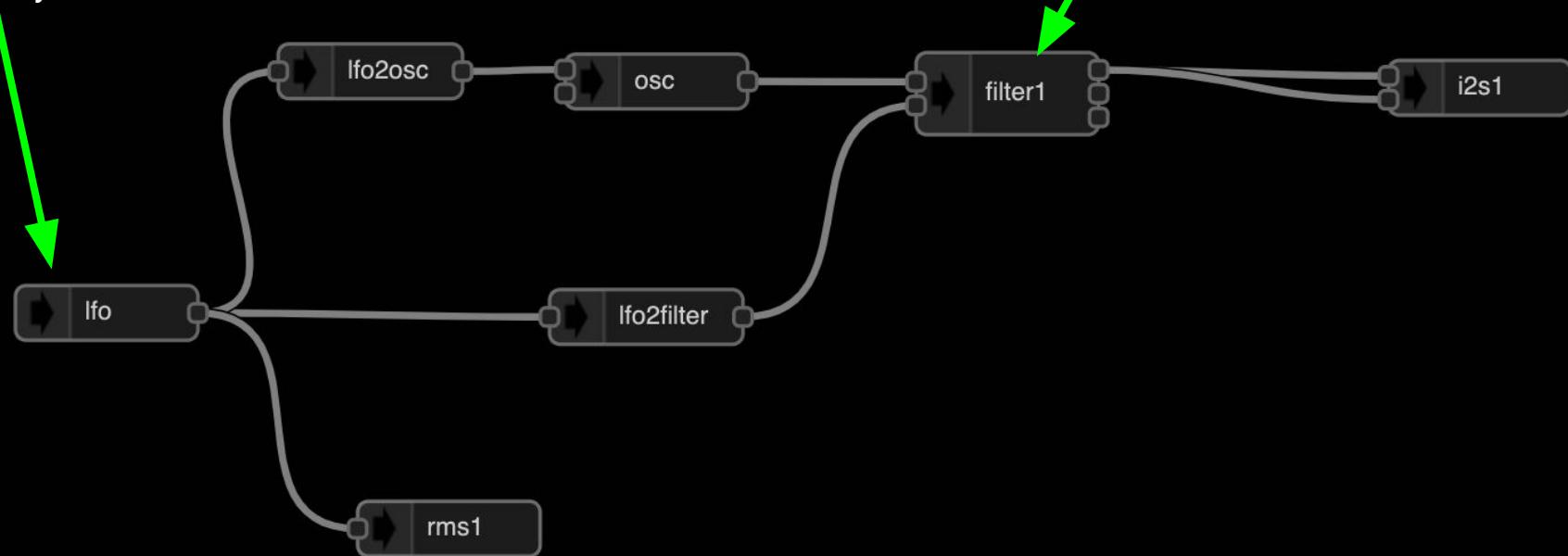
# Synth Design with the Audio Design Tool

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



## LFO: Low Frequency Oscillator

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



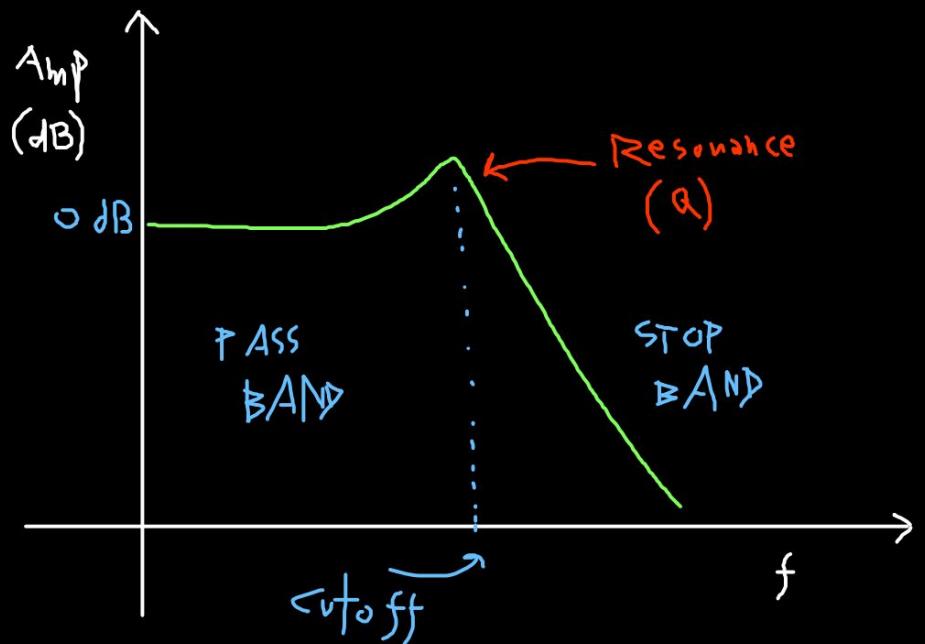
## Low Pass Filter

*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

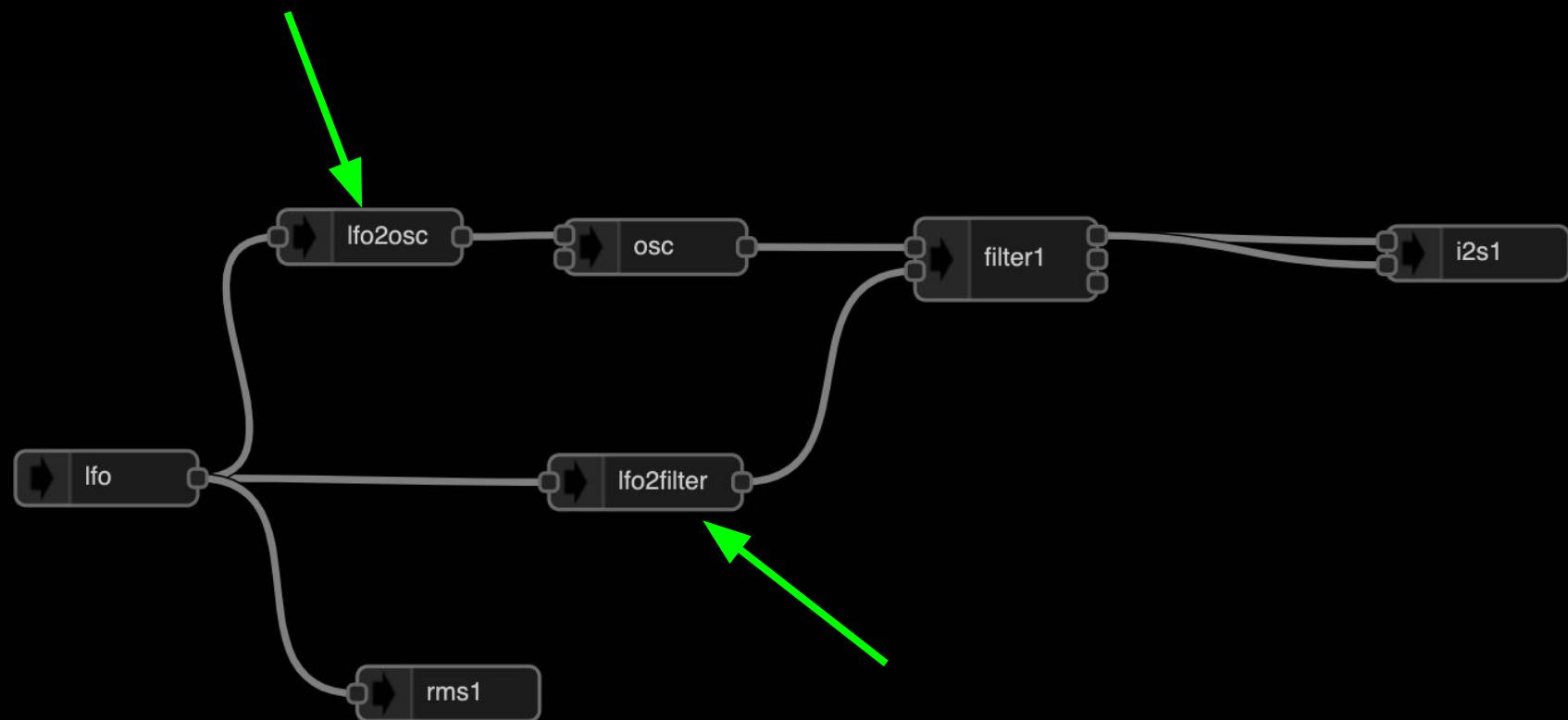


## Switching the Modulations On/Off: Adding Buttons

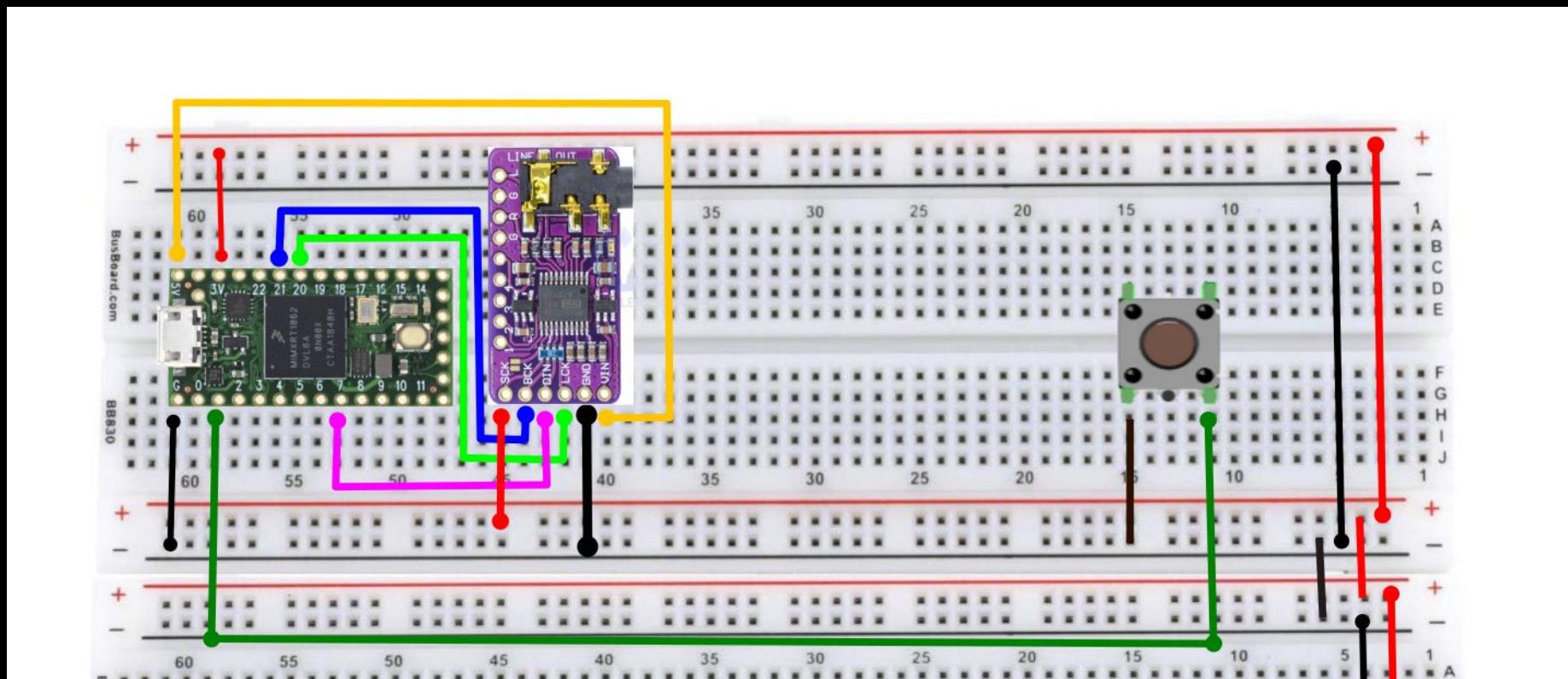
The LFO is connected to both the Oscillator frequency and the Filter cutoff.

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



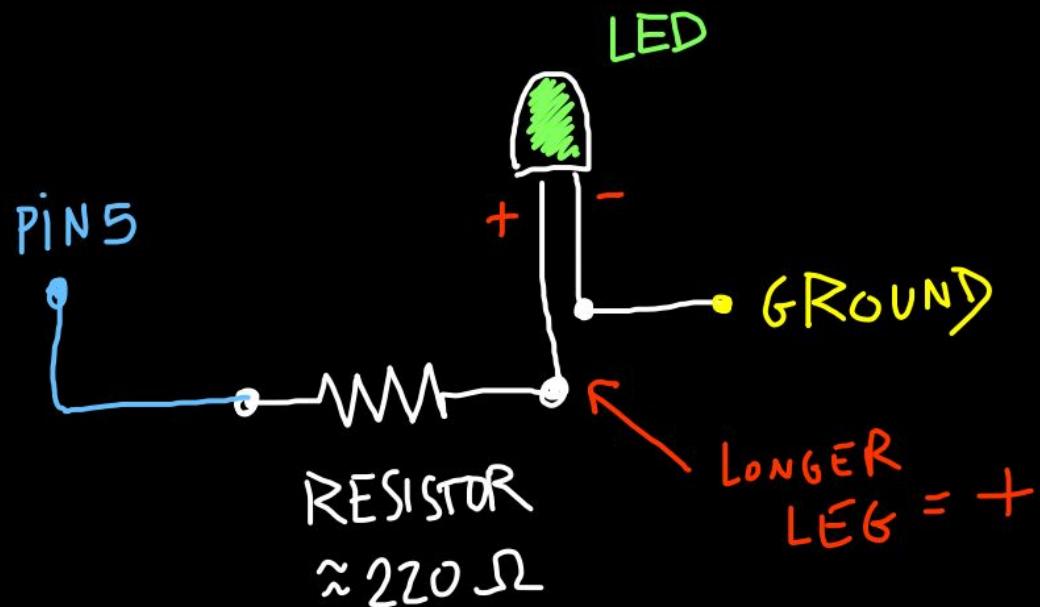
Before Setup() add:

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

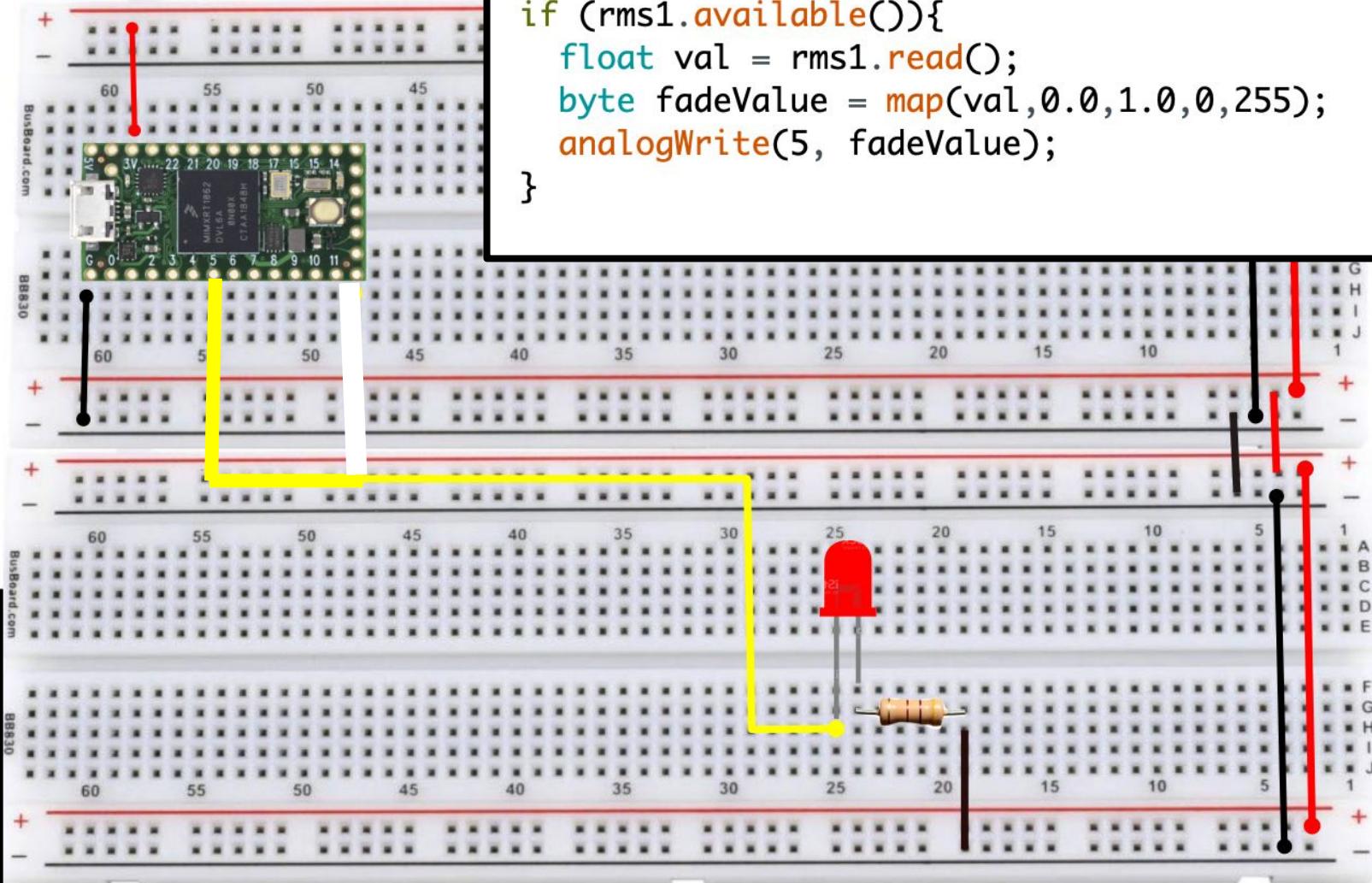
In the Loop() add:

```
// 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

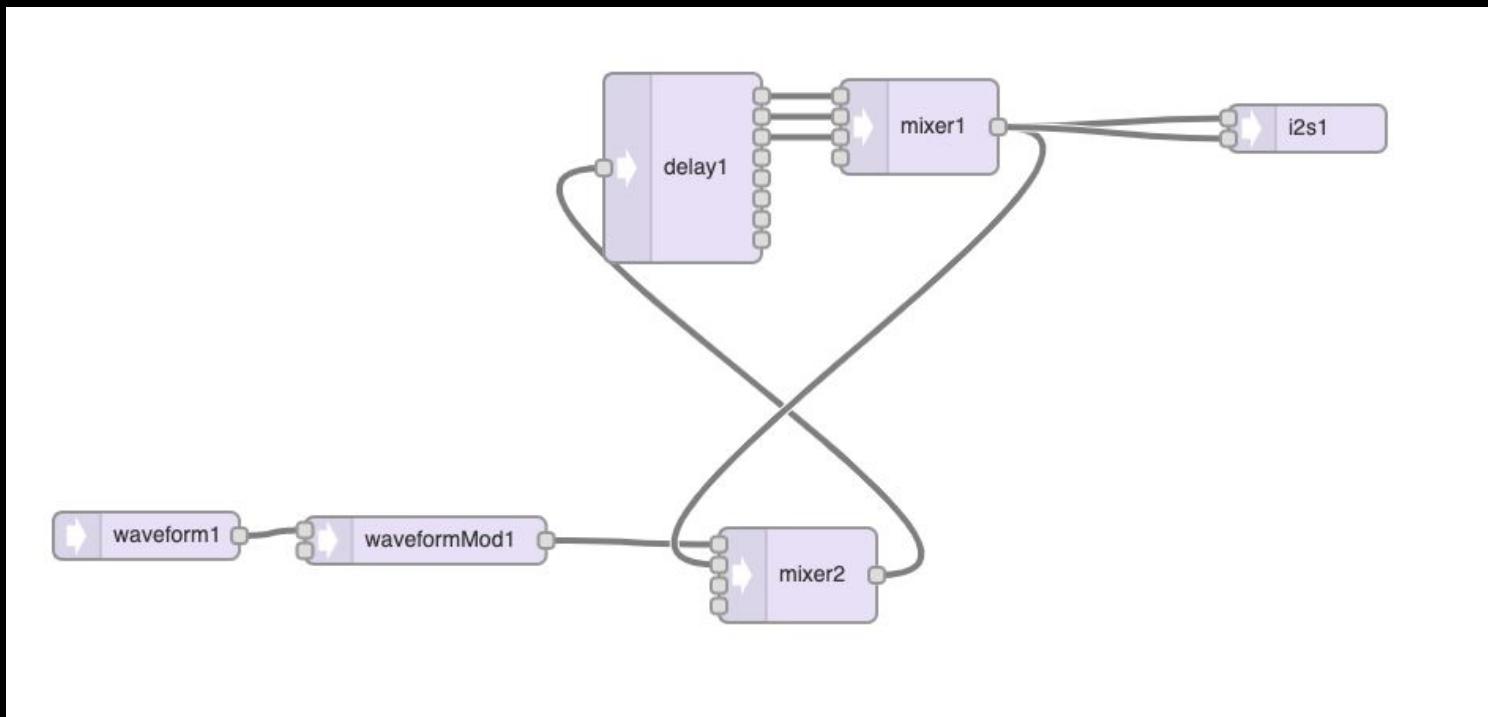


## Bonus: How to connect a LED and control it

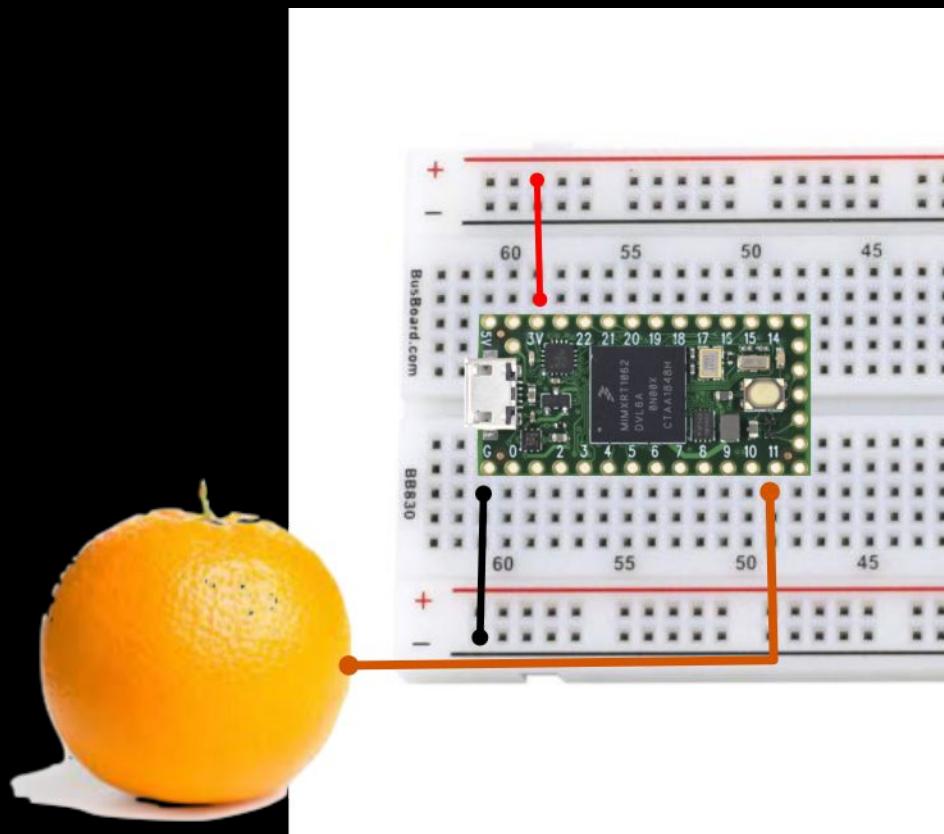


## More Bonuses:

### Adding a Delay FX



## Bonus: How to connect a Touch sensing pad



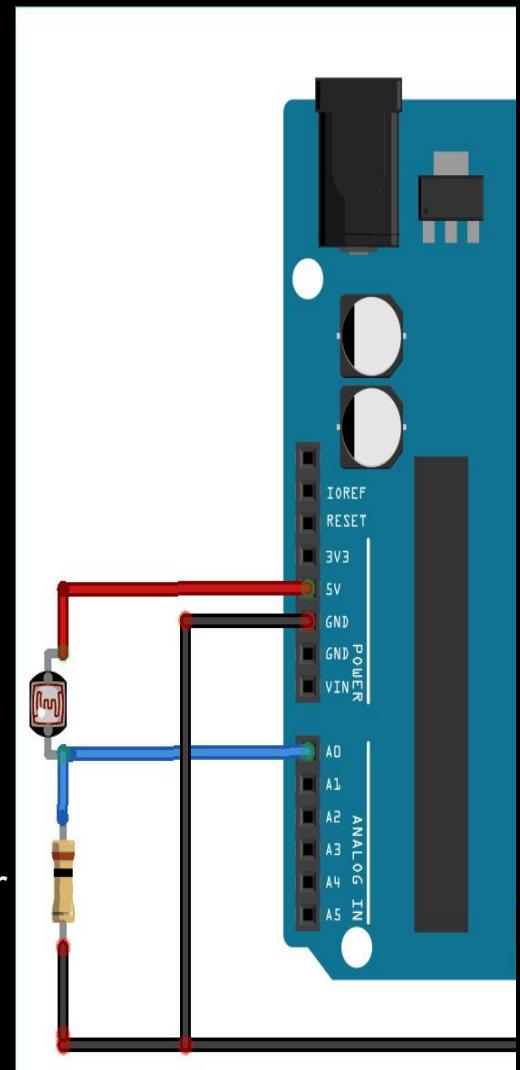
[see code in extra folder]

## Connecting a photosensor

A photoresistor is a light-dependent resistor

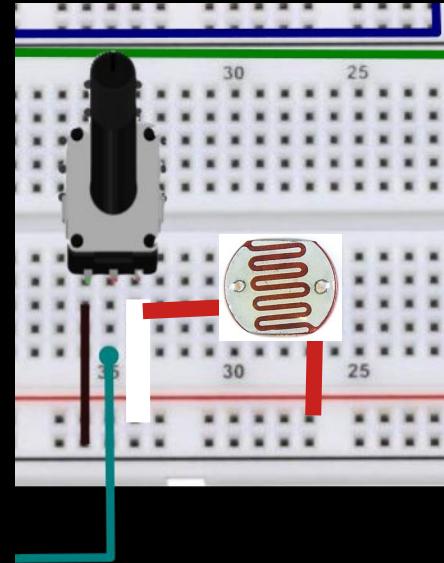
With the proper circuit, we can convert the light signal into a voltage signal and read it through an analog input.

1 K resistor



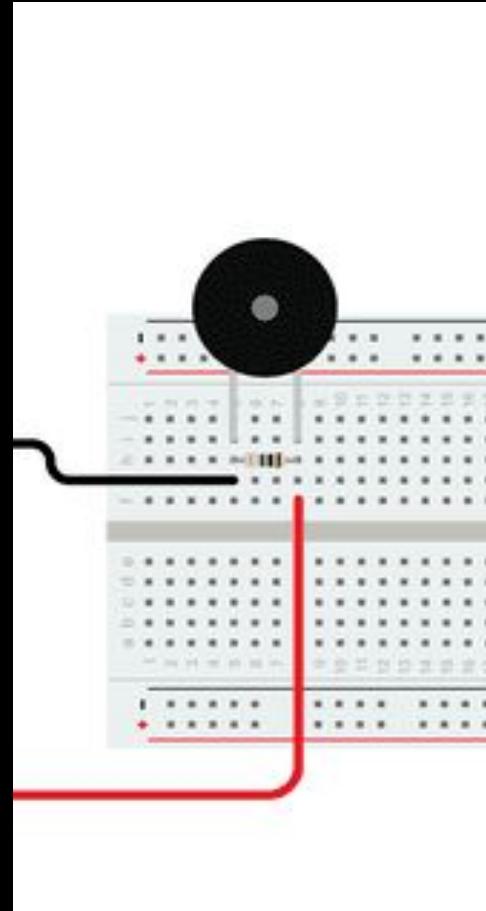
## Connecting a photoresistor

→ 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](https://docs.arduino.cc/built-in-examples/sensors/Knock)

More Bonuses:  
drum machines, envelopes, MIDI...  
(see extra folder)

**"The JVC K-B500.  
I'd get one  
even if I couldn't play."**

Frankly I thought portable single keyboards belonged on the toy counter at Woolies. But when JVC asked me to play their new K-B500, I was prepared to be diplomatic. They wanted me to play for 10 minutes. They asked me to stop after an hour. It was stereo. Real stereo. Through its own speakers or through my hi-fi. Then I saw this button. Ultra-chord it said. So I pressed it. And when I fingered a note I got the kind of chord accompaniment that only a 3 handed Martian could play. How do I get one? I asked. You can try saying nice things about it, they said. That's easy I said. So listen. The K-B500 is the best portable keyboard I've heard. You can play it anywhere - the back of a bus, on the street, even in a car. As long as somebody else is driving. It can sound like 8 different instruments.

Everything from a harpsichord to a jazzy clarinet. It's the only single keyboard with two ensemble settings - brass or strings. I could play it for hours. So can you. Even if you're the sort of sap that never got beyond G, D7 and C on your guitar you can play the K-B500. Press a button, play a note and you'll hear the rhythm of your choice, a chord and a base line. Or use the Compucoorder to programme up to 126 bars of accompaniment chords. Then you've got two hands free to pick out the tune. Whizzing! And so's the price. Less than 400 quid. At your JVC specialist electronic organ dealer. See him soon!

*Dick Hollard*

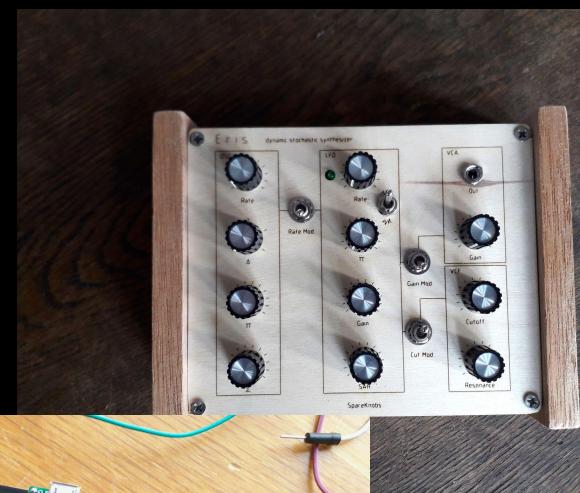
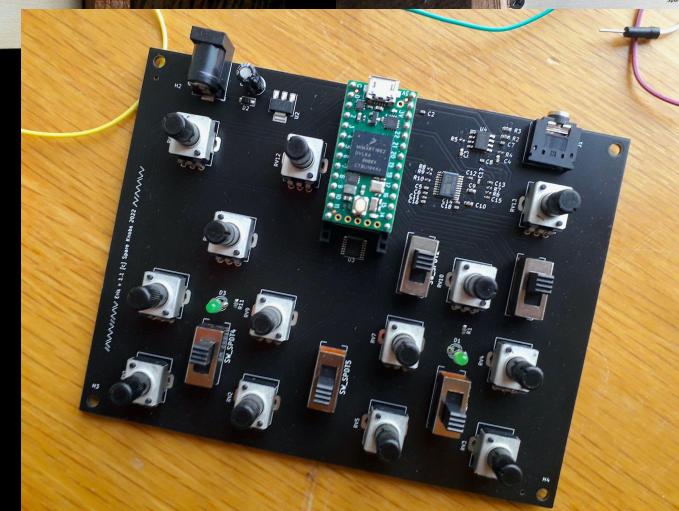
To JVC (UK) Ltd, Electronic Organs Division,  
Eldonross Trading Estate, Staples Corner,  
6-8 Priory Way, London NW2 7AF.  
I can't wait to get my hands on a K-B500.  
Tell me where I can.

NAME \_\_\_\_\_  
ADDRESS \_\_\_\_\_  
PHONE \_\_\_\_\_

**JVC**  
JVC is the trademark of the Victor Company of Japan

## 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/>