INTRO | mki x es(.edu)

Hey there, thanks for buying this DIY kit! We – **Erica Synths** and **Moritz Klein** – have developed it with one specific goal in mind: teaching people with little to no prior experience how to design analog synthesizer circuits from scratch. So what you'll find in the box is not simply meant to be soldered together and then disappear in your rack.

Instead, we want to take you through the circuit design process step by step, explaining every choice we've made and how it impacts the finished module. For that, we strongly suggest you follow along using **LABOR[1],** which is an all-in-one circuit prototyping tool that allows you to experiment and play around with your components in a non-permanent way. To help you with this, we've included suggested breadboard layouts in select chapters.

In addition to this, you can also experiment with some of the chapter's circuits in a **circuit simulator** called CircuitJS. CircuitJS runs in your browser. You'll find weblinks in the footnotes which will direct you to an instance that already has example circuits set up for you. We strongly encourage you to fiddle with the component values and general structure of those circuits to get a better understanding of the concepts we're laying out.

Generally, this manual is intended to be read and worked through front to back, but there were a few things we felt should go into a dedicated appendix. These are general vignettes on electronic components & concepts, tools, and the process of putting the module together once you're done experimenting. Don't hesitate to check in there whenever you think you're missing an important piece of information. Most importantly though: have fun!

TABLE OF CONTENTS

---

[1] You can also use a standard breadboard, but this will require you to get a little creative when adapting the suggested layouts – plus you'll need to find some breadboard-friendly push buttons. You'll also need to do some additional engineering to get the different supply voltages.

THE mki x es(.edu) DRUM SEQUENCER

This is what most drum sequencers look like: packed with buttons and LEDs, all driven by a microcontroller.

I wanted to design my own. But wiring up this many buttons on a breadboard? That would get messy quick – and we haven't even started coding yet. So I decided to take a step back. What if I don't follow that blueprint at all? What if I start from scratch, and only add what I really need?

That was the challenge I set for myself: what's the most minimal drum sequencer I can come up with without compromising on playability? Here's what I ended up with: a two-button circuit that allows for full pattern programming, two types of accents, pitch modulation, and per-channel control over loop lengths and timing. And all of it runs on just two logic chips, with no code at all.
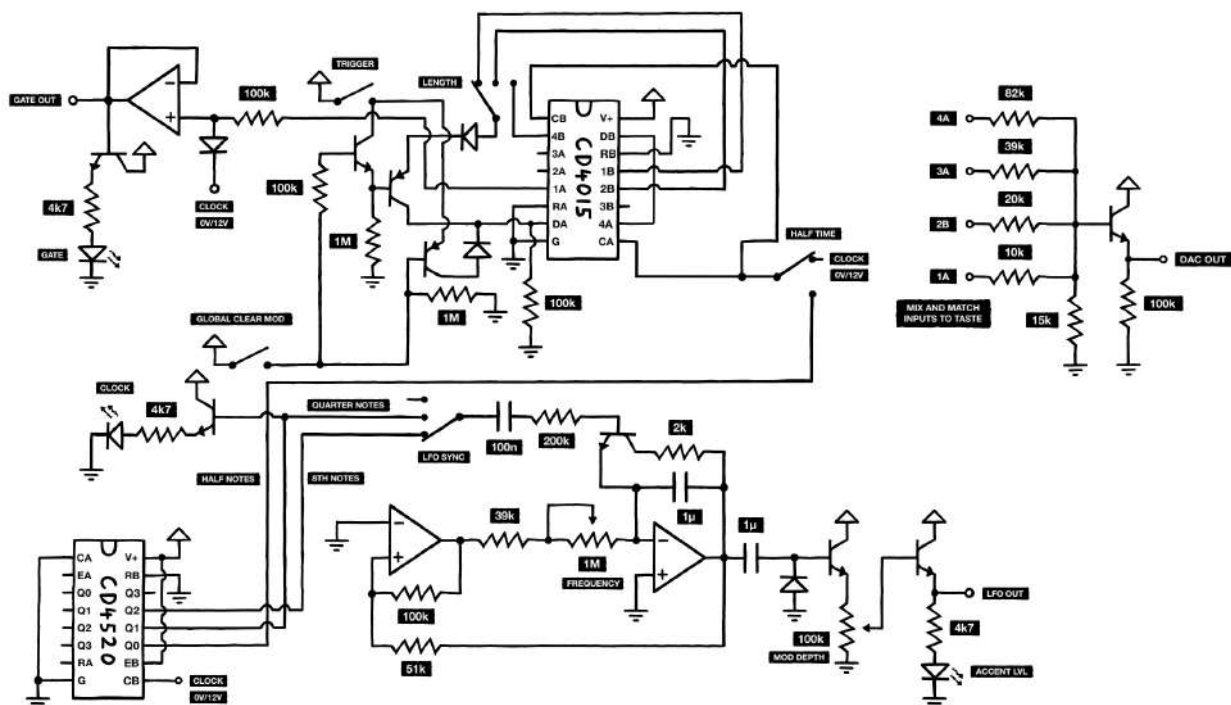
BILL OF MATERIALS

Before we start, please check if your kit contains all of the necessary components. In addition to a PCB, panel and power cable, your box should also contain:

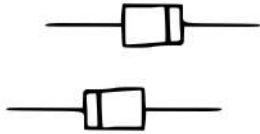**An array of resistors**. The specific values (in ohms, which you should check for with a multimeter) are

2M2	x2
1M	x8
200k	x2
100k	x21
82k	x4
51k	x2
39k	x6
27k	x1
20k	x4
15k	x5
10k	x14
4k7	x7
2k	x4
1k	x1
470	x13
10	x2
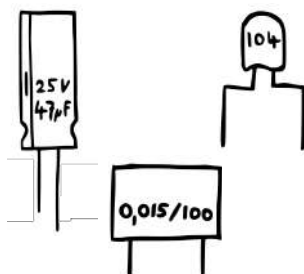
A bunch of capacitors. The specific values (which are printed onto their bodies) are

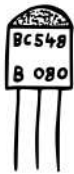|	
47 uF	x2
1 uF	x5
100 nF	x24

Some diodes. The specific model names (which are printed onto their bodies) are

1N4148 (signal)   x15
1N5819 (schottky) x2

A couple transistors. The specific model names (which are printed onto their bodies) are

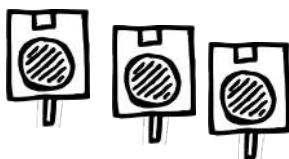**BC548 (NPN)**   x20
BC558 (PNP)   x9

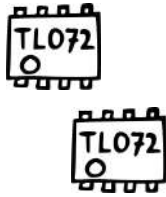**A handful of potentiometers**. Their specific values (which may be encoded & printed onto their bodies) are

1M (B105)   x2
250k (B254) x1
100k (B104) x2

A bunch of jack sockets. The specific models (which you can identify by their color) are

Switched mono (black)   x14

A couple chips. Their specific models (which are printed onto their bodies) are

TL074 (quad op amp)   x2
TL072 (dual op amp)   x1
CD4520 (binary counter) x1
CD4015 (shift register)   x8

A few switches and buttons. The specific models are

Single pole, double throw   x4
Double pole, double throw   x6
KS01-BV (push button)   x1
**Cherry MX (push button)**   x4

A couple LEDs (light emitting diodes). The specific model (which you can identify by measuring their body's width) is

3mm (red)   x3
1.8mm (red) x4

You will also find a few sockets that are only relevant when assembling the module in the end.

USAGE WITH MKI x ES LABOR

We highly recommend that you follow this guide using an **MKI x ES LABOR** prototyping board. **LABOR** comes equipped with everything you need for testing the circuits we lay out: a standard 830 tie point breadboard, an integrated dual power supply with over current protection, a manual gate/trigger/envelope generator, an LFO, a variable CV source, an output amplifier, and a modular interfacing section where you can insert all of your interfacing components like potentiometers, jack sockets, and switches.
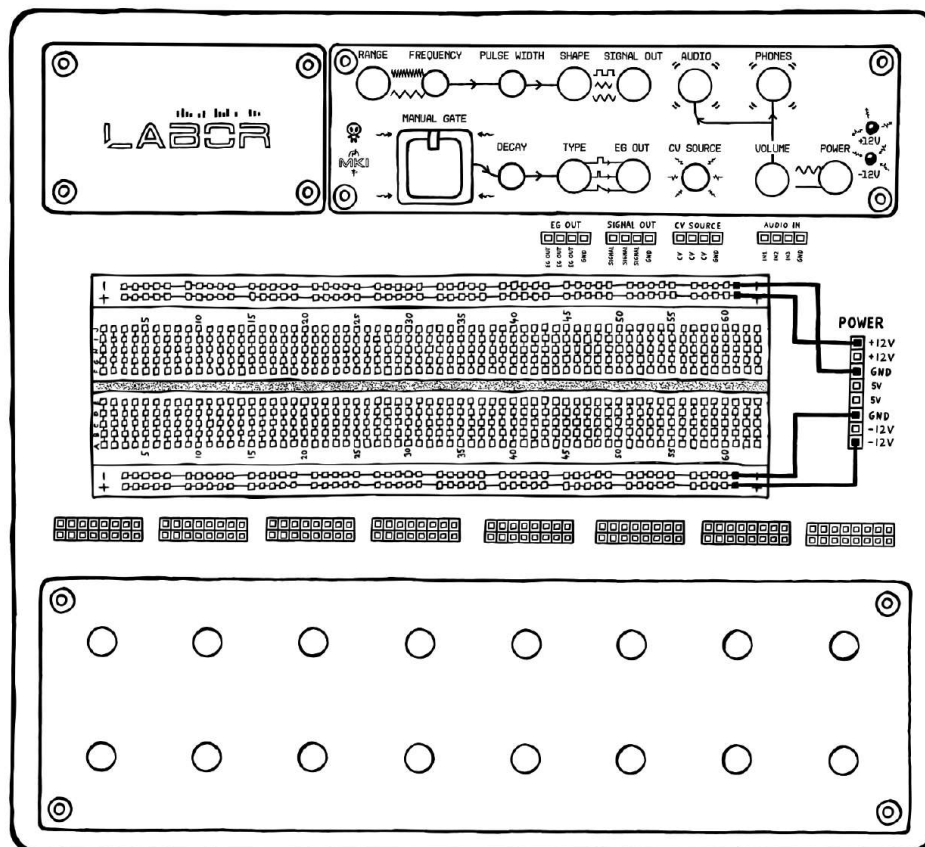
Before you get started, connect the slots labeled **GND** on the power header to both breadboard rails labeled **–** (minus). Next, connect one slot labeled **+12 V** to the top breadboard rail labeled **+** (plus), and one slot labeled **–12 V** to the bottom breadboard rail labeled **+** (plus).

To listen to your circuit, you don't even need to set up an output jack socket. Instead, use the built-in output amplifier at the top of the device. Just plug your circuit's signal output into the header labeled **AUDIO IN**, and then connect your headphones to the **PHONES** output jack (or a line-level device like a standalone external speaker to the **AUDIO** output jack).

Sometimes, this guide will ask you to use external gear like sequencers or LFOs to send CV, audio signals, triggers or gates into your circuit. With **LABOR**, there's no need for extra equipment – just use the built-in oscillator (audio/LFO), CV source or manual gate/trigger/envelope generator. You can grab all of those via the headers labeled **EG OUT**, **SIGNAL OUT**, and **CV SOURCE** and connect them to the designated points on the breadboard.
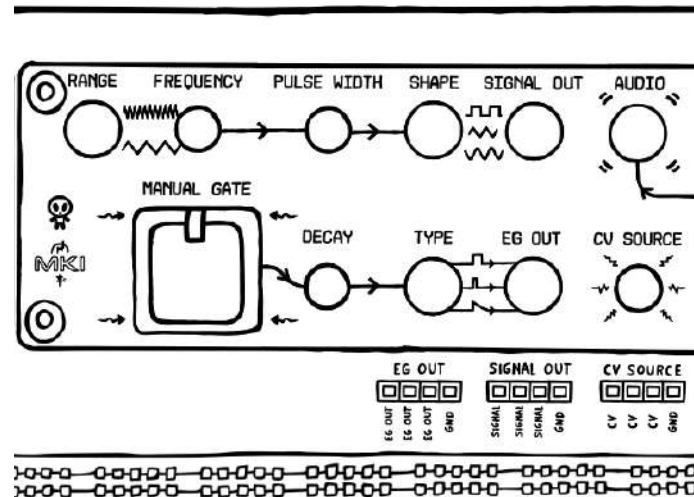
WHAT'S A DRUM BEAT?

Before we start patching things together, let's talk about what a drum beat actually is. Imagine a simple kick drum pattern playing. What makes it a beat, exactly? **Two things: first, it's repetitive**. **And second: it follows a timing grid**. There's an underlying pulse – a clock – and each tap either lands on a clock pulse, or it doesn't happen at all.
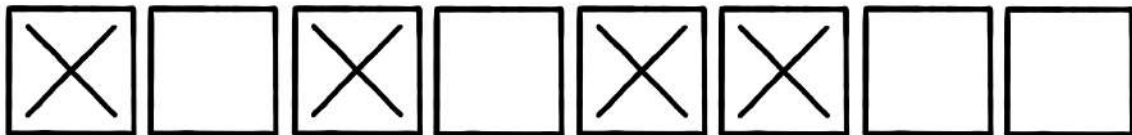
See these empty spots in our pattern? That's what we call *rests*. Okay, now how do we translate this to the world of electronics? **Well, if you squint, it's really just a stream of binary data**. Hit or rest. High or low voltage.
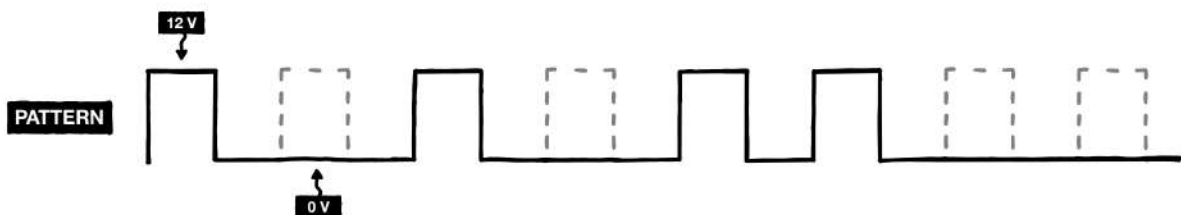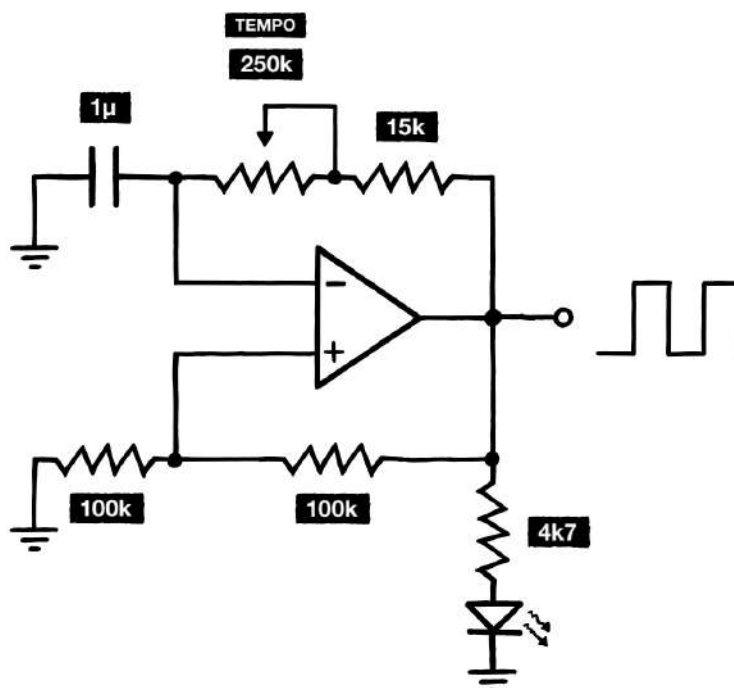

And the same goes for the clock – it's just a voltage flipping between high and low at regular intervals. And the shorter those intervals, the faster the tempo.
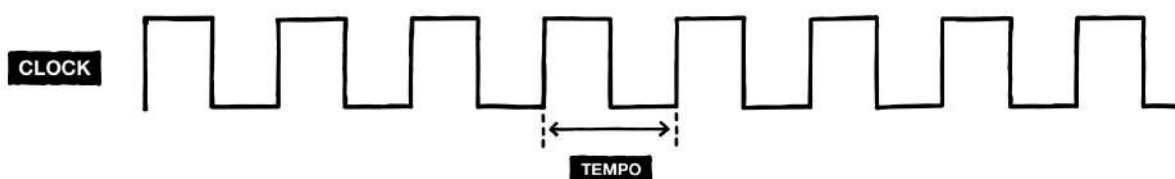
So how do we generate these signals in a circuit?

THE CLOCK GENERATOR

Let's start with the clock. All we need there is a simple free-running square wave oscillator.


I've explained this exact clock oscillator in the manual for our DIY sequencer kit[2] in detail, but here's the basic gist. **The op amp is wired in an intentionally unstable way, forcing it to charge and drain the capacitor on the left in an endless loop**.

This creates a steady square wave at the output, with a frequency that depends on the resistance between the op amp and the cap. To make it adjustable, I've set up a potentiometer in series with a fixed resistor – the latter enforces an upper frequency limit. Also, to get some visual feedback on what our oscillator is doing, I've added an LED at the op amp's output.[3]

---

[2] You can find that manual here: https://www.ericasynths.lv/media/SEQ_MANUAL_v3.pdf

[3] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/yu6yyx7b – you can change all values by double clicking on components.
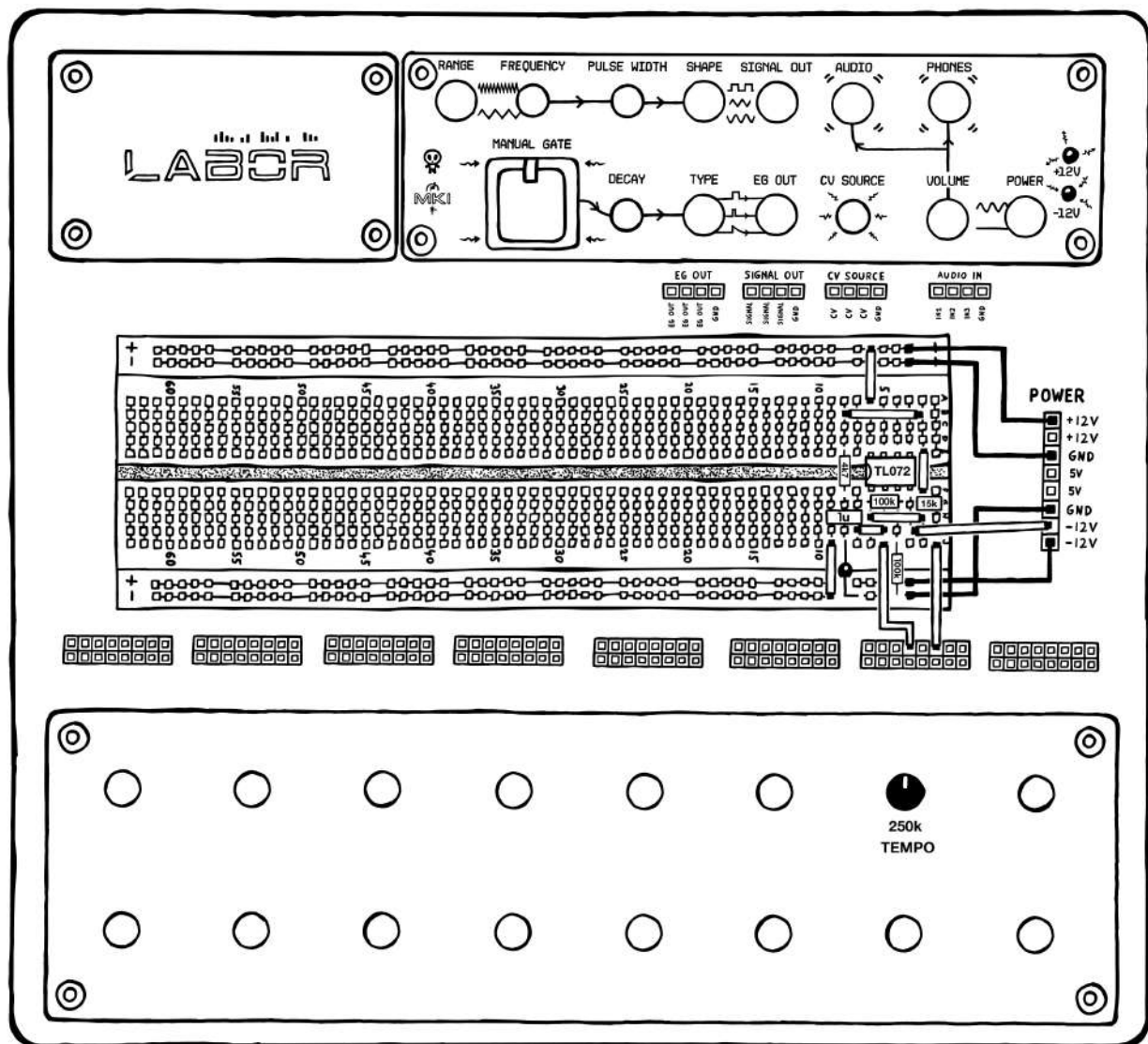
Once you've set this up as shown here, you should see the LED blinking when you turn on the power. Try turning the potentiometer – it should change the frequency of the blinking. Okay, now what about the drum pattern? Well, the simplest way to generate one is probably just… pressing a button. You can configure the **MANUAL GATE** button on LABOR so that every press gives you a high voltage (top setting). Next, connect the output socket (**EG OUT**) to the trigger input of a drum module and play a pattern in sync with the clock. And voila: you've got a drum beat.

Of course playing our beat manually is not quite what we're after. We want to design a circuit that plays this pattern for us. **But that means we need some way to store it – and play it back in a loop**.

THE TAP LOOPER | PART I

Now, that might sound like a job for a microcontroller, but there's actually a much simpler type of chip that's perfect for this kind of thing: a shift register. **Think of a shift register as a clocked chain of binary memory cells**.
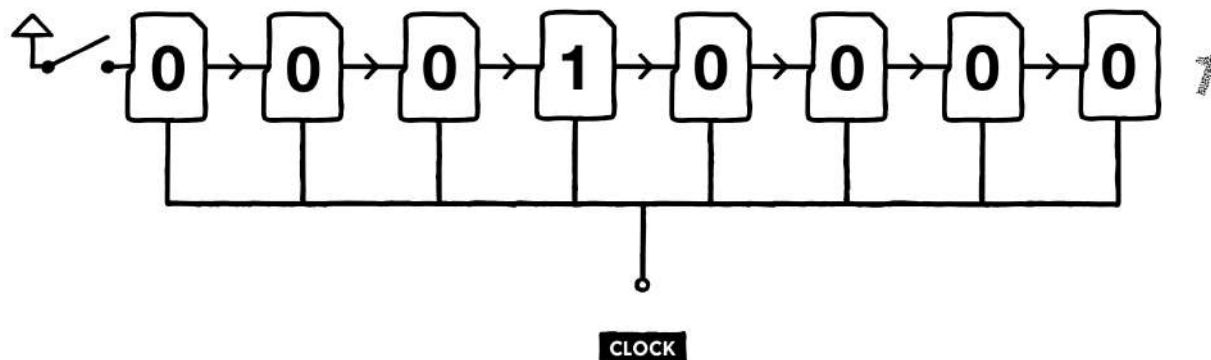

On each clock pulse, it shifts its contents one step to the right – while grabbing a new bit from the input. (A high voltage is interpreted as a logic 1, a low voltage as a logic 0.) Once a bit reaches the end of the chain, it drops off. But here's the trick: **if we connect the last cell back to the input, that bit circles around instead,** looping through the register indefinitely.
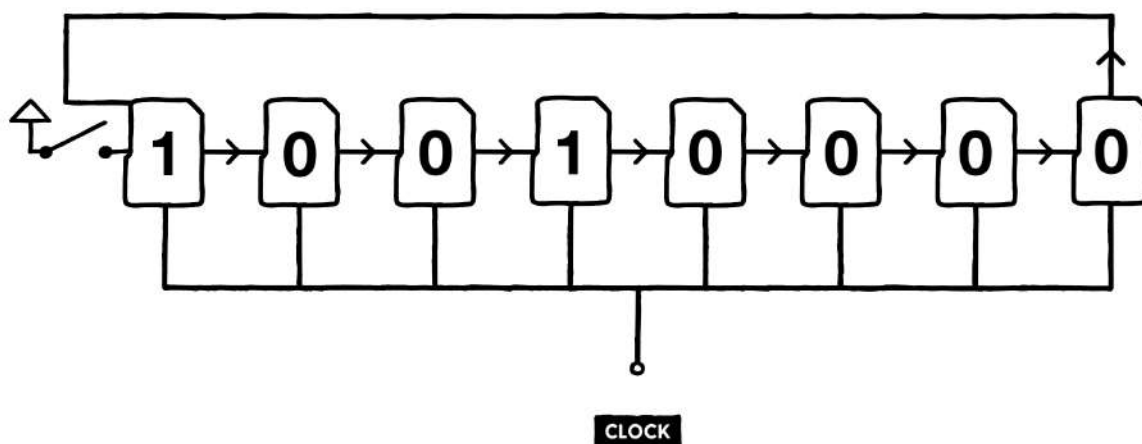

Add a few more logic 1s, and you've got a full drum pattern rotating through the chip. Okay, but how do we use this to trigger a drum sound? Easy: we just watch the very first cell. **It goes high whenever a 1 swings by, essentially playing back what we tapped in**. That's our trigger signal.

To try and implement this, we'll use the CD4015: a dual 4-bit shift register.
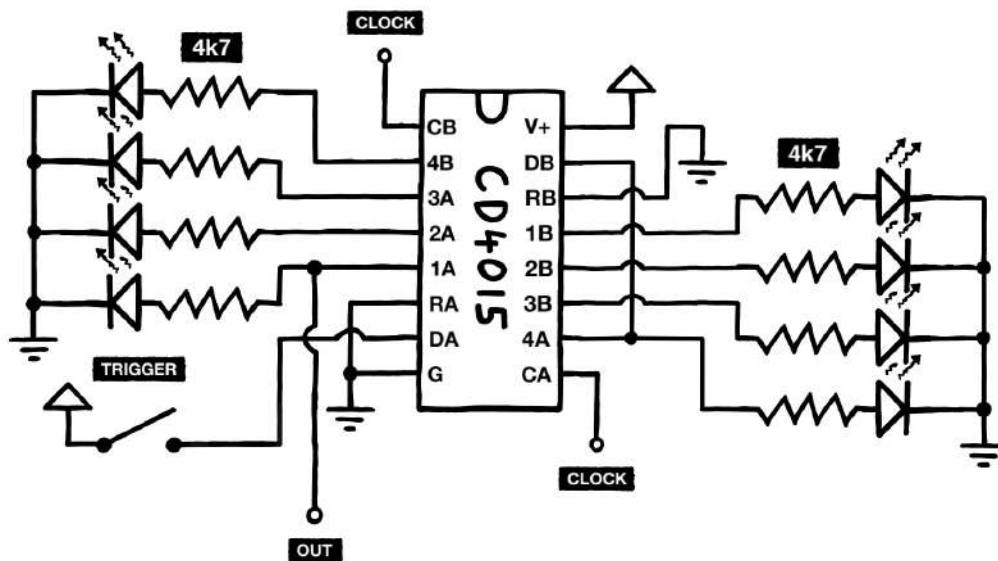

For the initial (non-looping) setup, we'll have to supply the chip with 12 V (**V+**) and ground (**G**). Next, we'll turn it into one single 8-bit register by chaining the two 4-bit registers. For this, we simply connect the last output of register A (**4A**) to the data input of register B (**DB**). And we'll tie both reset pins (**RA/RB**) to ground to enable both registers. To visualize the register's internal state, we'll set up 8 LEDs – one for each output (**1A** through **4B**). Then, we'll connect our pushbutton to the first register's data input (**DA**).

Finally, we need a clock signal for both registers. Unfortunately, if we just hook our oscillator up to the 4015 as-is, we risk frying it. **That's because the oscillator swings all the way from +12 to –12 volts, and the 4015 can't handle negative input voltages**. To fix this, we need to get rid of the negative part of the signal. For this, we'll run it through a diode, followed by a 100k pulldown resistor.
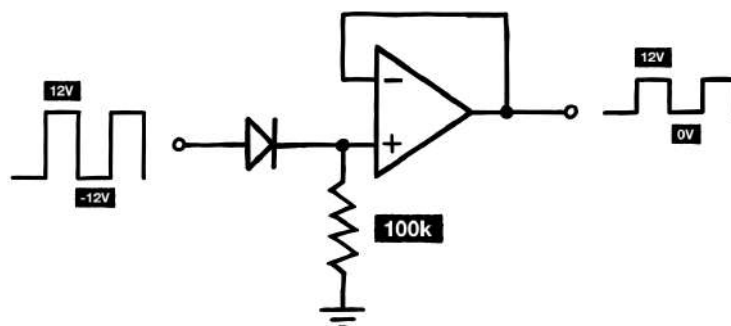
This lets positive voltages pass – while blocking anything negative. And to keep the output impedance low, we buffer the result with an op amp. Which we'll then hook up to the 4015's **CA** and **CB** clock inputs.[4]

---

[4] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/2x3b644r – you can change all values by double clicking on components.
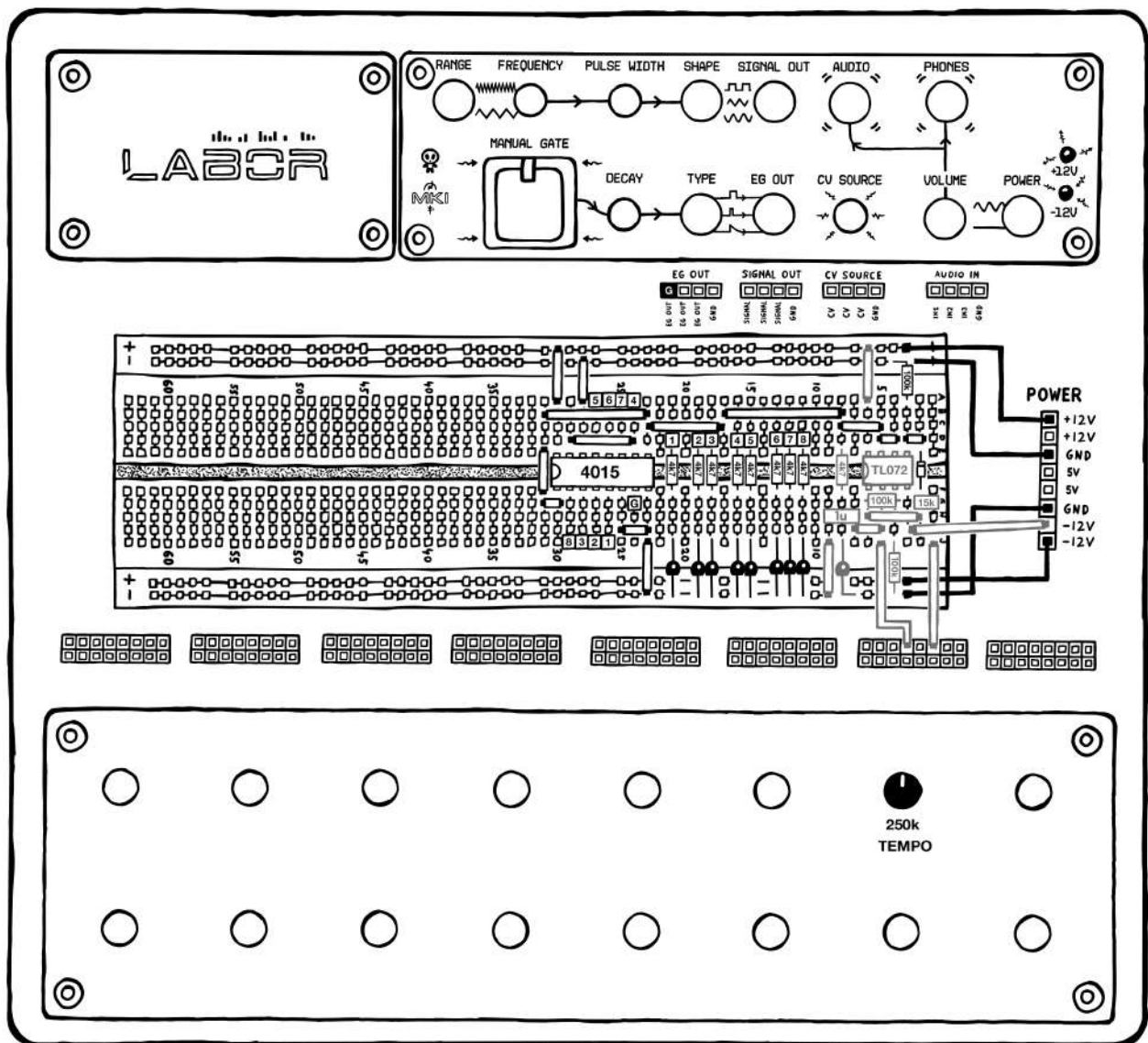
If you set it up like this and push the **MANUAL GATE** button, you should see a bit travel through the chip and then drop off at the end. Great, so how do we make this loop now?
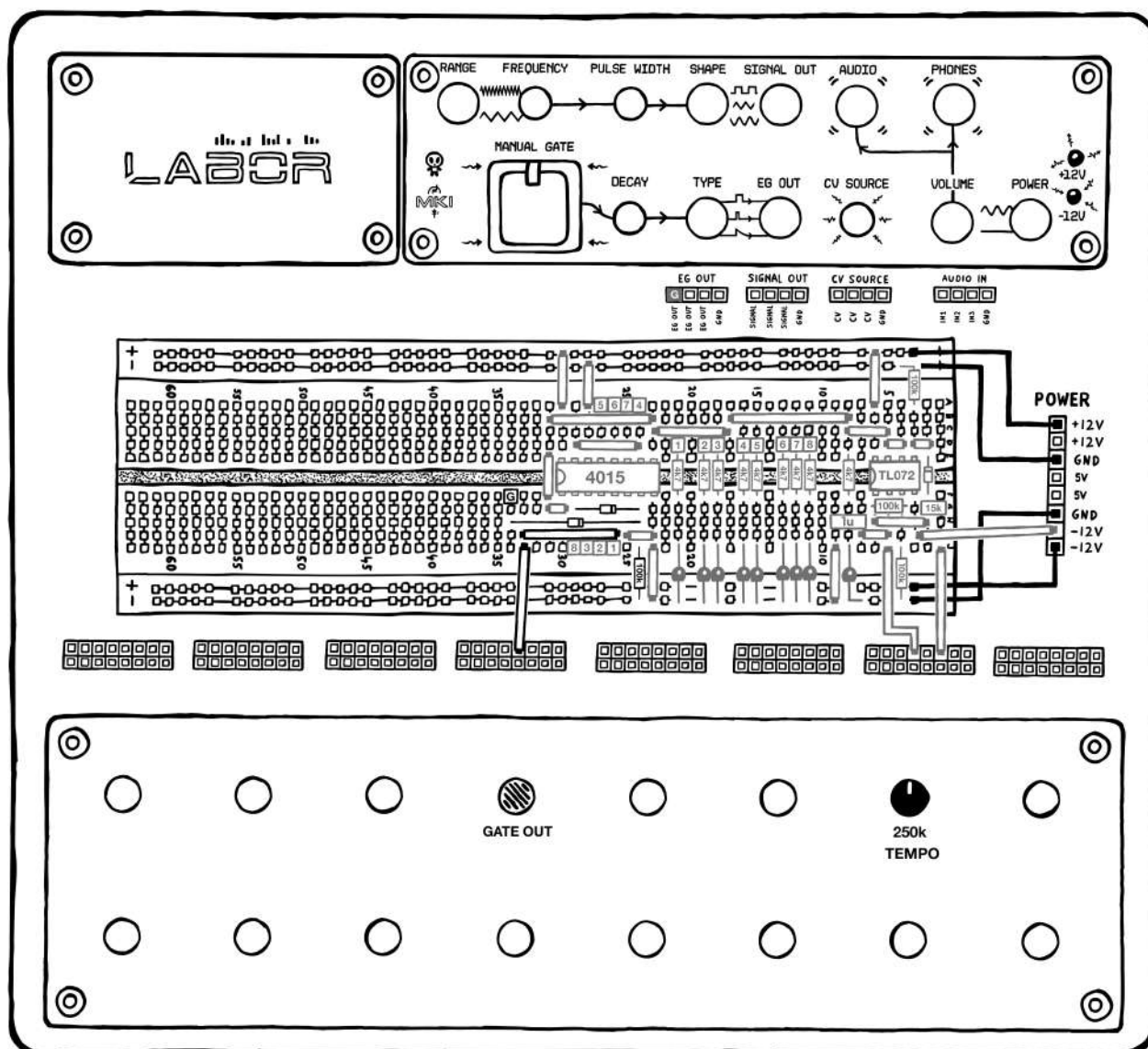
THE TAP LOOPER | PART II

Unfortunately, we can't just connect the last output to the data input like this.
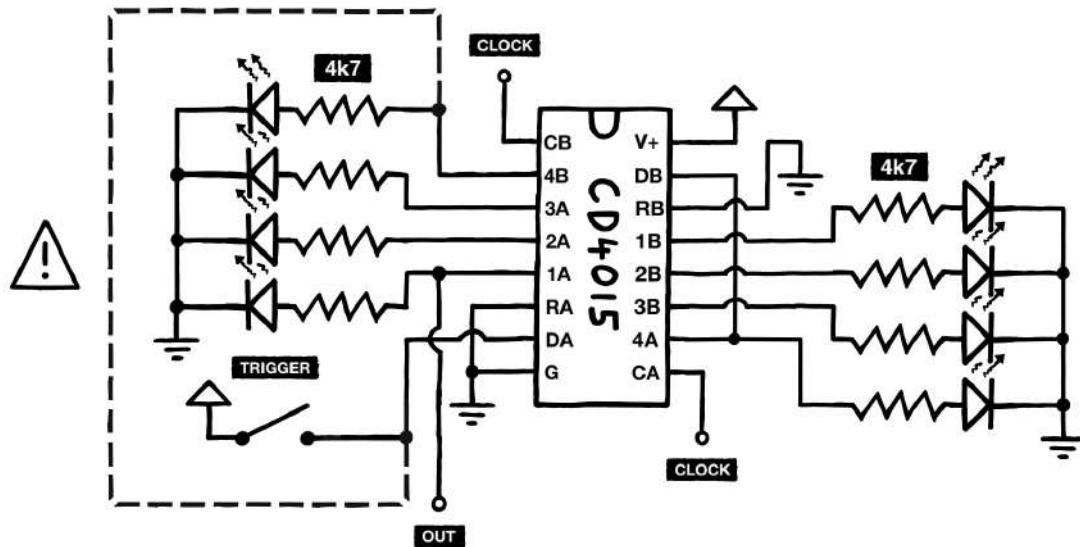
That's because both the output and the button are able to sink current – **meaning that if either is high while the other is low, we create a short circuit**. To avoid this, we'll set up two diodes – one after the button, and one at the **4B** output pin.
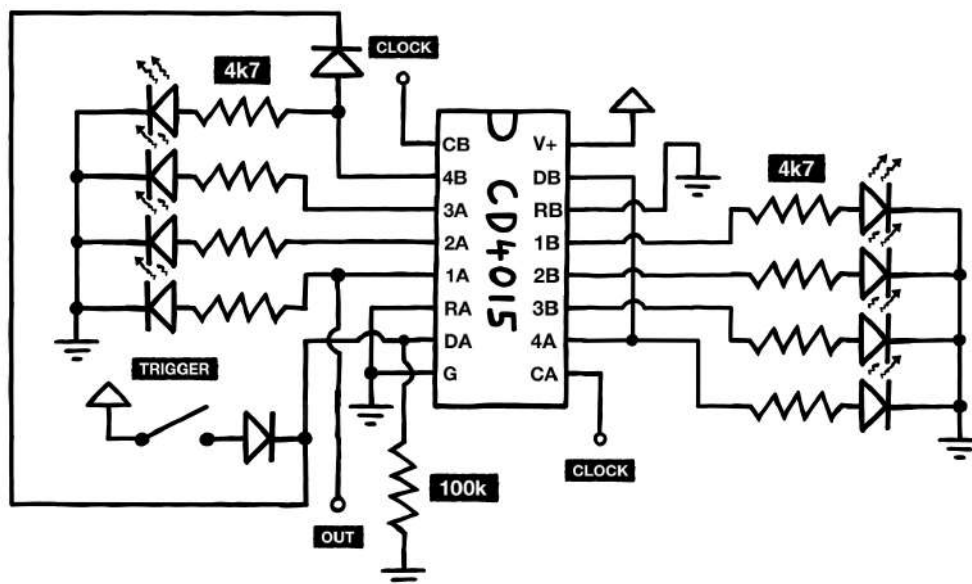
These will block any current from flowing in the wrong direction. And to make sure the voltage at the data input doesn't just float when both button and output are low, we also add a 100k pulldown resistor.[5]

So if you now push the button, the bit should loop around after reaching the end of the register. Next, connect the output socket at the bottom to a drum module. You should be able to tap in a pattern and have the sequencer loop it for you. But there's one issue: as soon as you have two bits right next to each other, you only get one drum hit – not two. What's up with that?

---

[5] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/yt62ow2p – you can change all values by double clicking on components.

THE CLOCK GATE HACK

The issue is that the drum module gets triggered when its input voltage goes from low to high (we call this a **rising edge**). Take a look at the sequencer's first output LED: every time it switches from off to on, we hear a drum sound. But if two high bits appear in a row, the LED never turns off in between – so there's no second rising edge.

How do we fix this? Well, you might've noticed that while the output LED stays on, the clock LED still blinks: on, off, on. **This means that if we found a way to pull our trigger output low whenever the clock is low, we should get two rising edges – and two drum hits**.
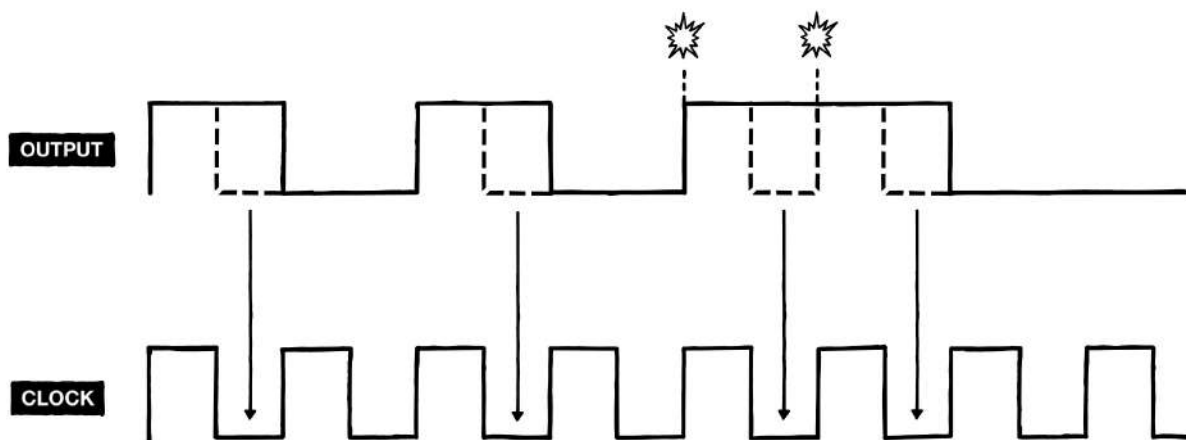

Great! But how do we do that? It's actually pretty simple. All we need are three components: an op amp, a 100k resistor, and a diode.
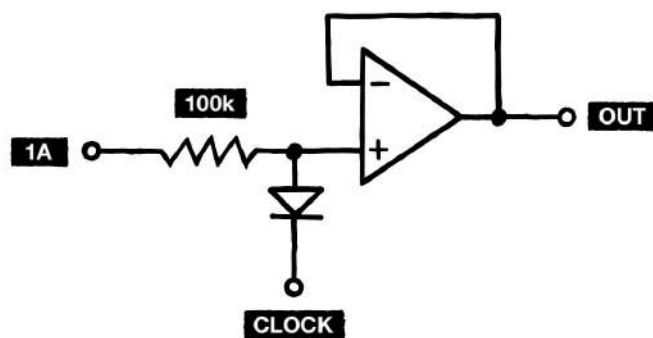
Here's how it works. Whenever the clock is high, the diode blocks, and whatever voltage is present at the trigger output gets buffered by the op amp unchanged. But when the clock is low and the trigger output is high, the diode conducts, pulling the voltage at the buffer's input low. As a result, the op amp's output drops as well – creating a clean rising edge on the next clock pulse.[6]

---

[6] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/yw6bx77p – you can change all values by double clicking on components.
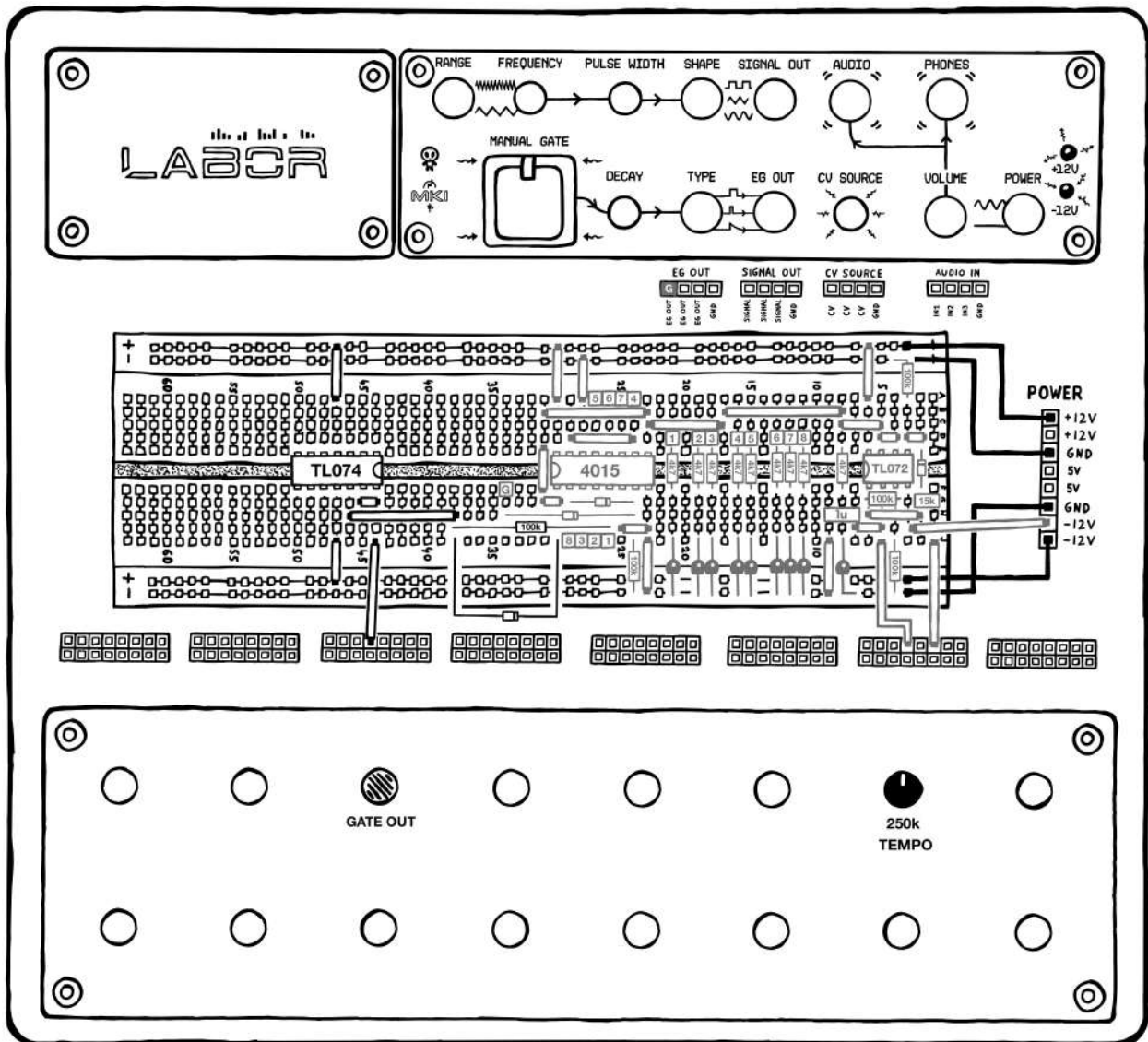
If you test this now, you should get two drum hits for two consecutive bits. Great! Now, try filling the entire register. This should work as well. But now we've got another issue: since we can only add bits, there's no way to edit the sequence. We could always reboot the circuit to wipe it – but let's be honest: that's a little crude.

THE CLEAR BUTTON

Ideally, I'd want to remove bits the same way we add them – by pressing a button. Unfortunately, we can't selectively delete data once it's inside the register. But there is a workaround. **We might not be able to remove a bit directly, but we can stop it from looping**. Think of it like this: every time a bit reaches the end of the register, we have a choice – do we feed it back, or do we drop it? If we drop it, we're effectively deleting it from the sequence.
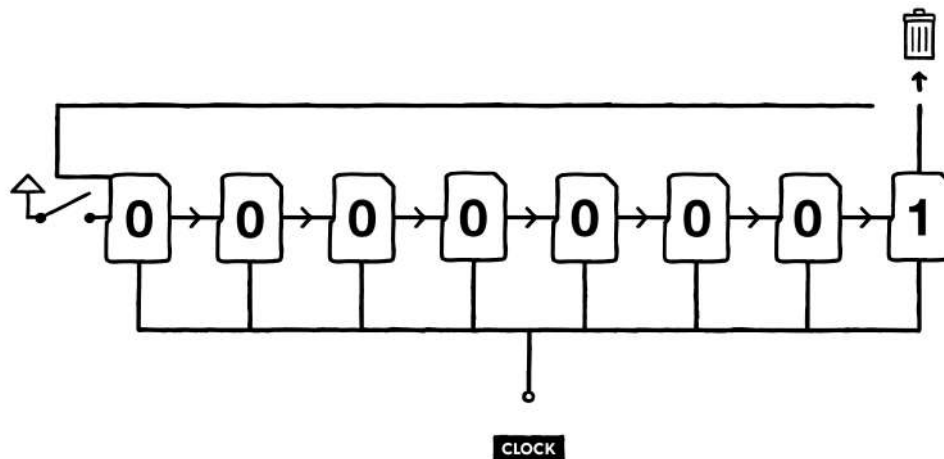
So if we could momentarily break the feedback loop by pressing a button – we'd be set. Implementing this isn't straightforward, though, because most push buttons are *normally open*: they only connect when pressed. **So if we'd put one into the feedback loop, the connection would be broken by default, and the sequence wouldn't loop unless we held the button down the entire time**. (Which is not very fun.) To work around this, we'll use a PNP transistor.
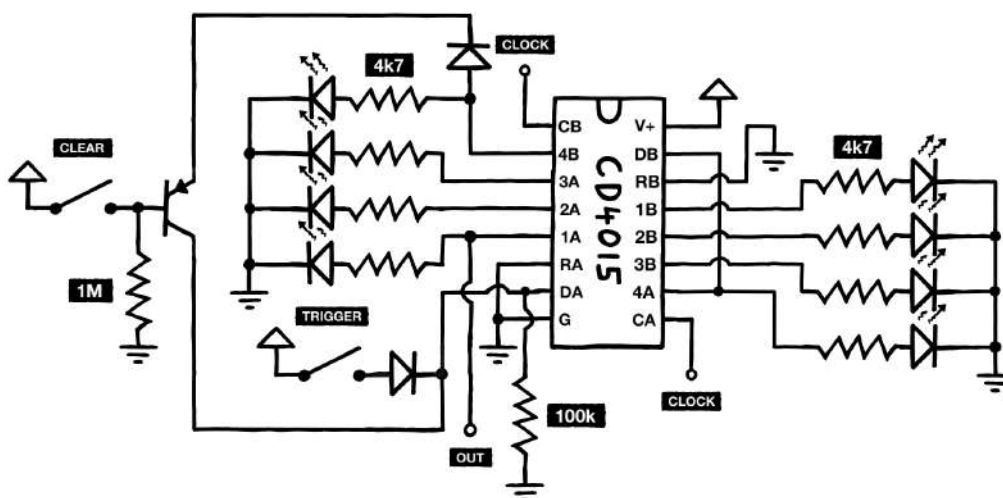
If we place it in the feedback loop like this, it conducts by default – unless its base gets pushed high by the button, in which case it blocks and breaks the loop.
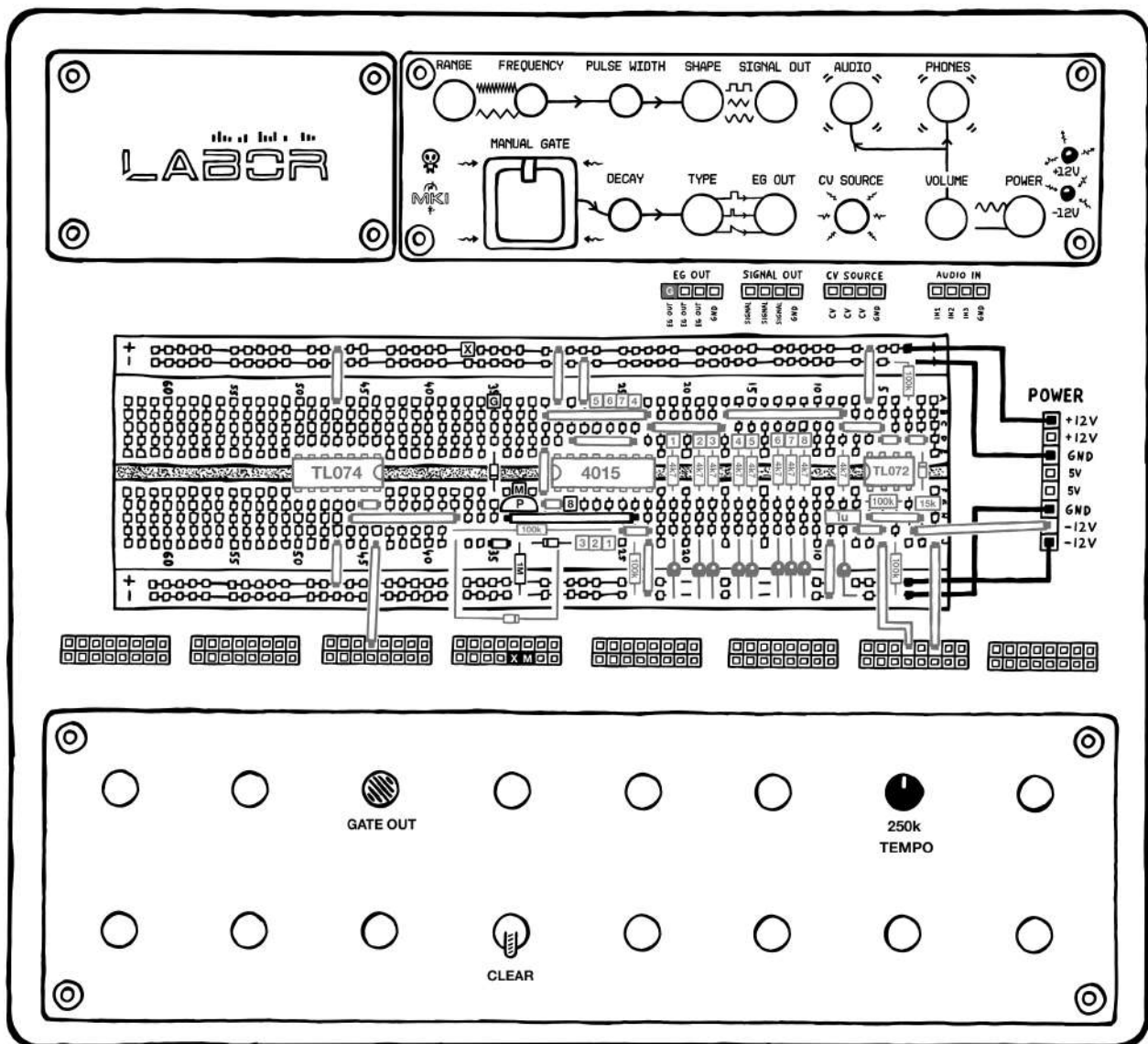
Since your kit does not contain a breadboard-friendly push button, we'll need to use a switch as a replacement. Let's see if it works: fill up the register with bits, then try removing some by briefly flipping the switch.[7]

While not ideal usability-wise, it should work in principle. In the final build, we'll use a proper push button for this. What's cool about this setup is that it lets you delete the step that's just about to play. It's like entering rests into the loop – the exact counterpart to adding triggers with the other button.
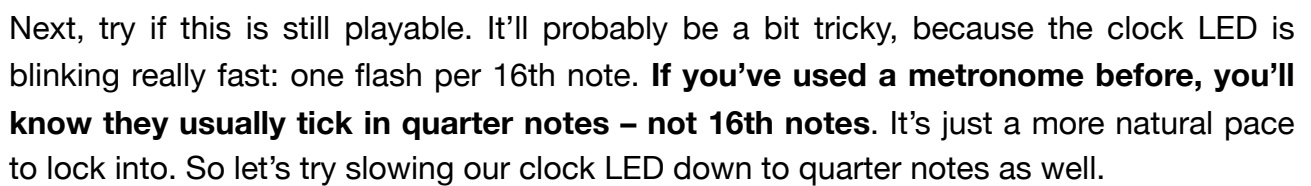
---

[7] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/ypbg842m – you can change all values by double clicking on components.

THE GATE LED

Okay, so at this point, we've got a fully functional drum sequencer – with an ultra-minimal interface. Just two buttons! Well… almost ultra-minimal. Currently, we're using eight LEDs: one for each shift register output. That's great for debugging, but a bit overkill for regular usage. **Since we're only using one output for triggers, we can simplify things**. If we get rid of all eight LEDs and replace them with one LED at the trigger output, we should still get enough visual feedback, but without all the extra wiring, clutter, and power draw.
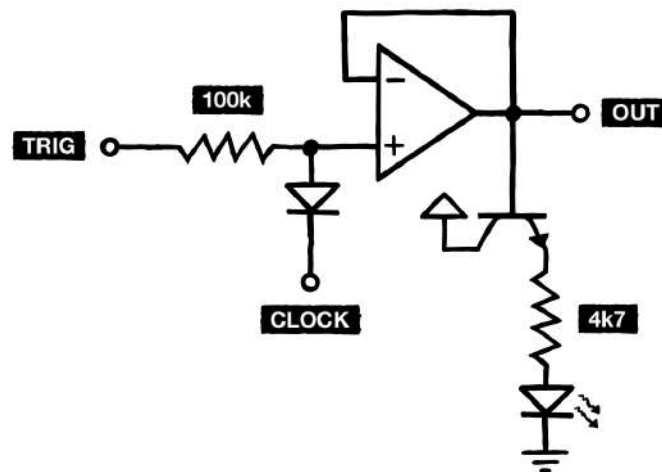
Speaking of power draw: to avoid overloading the op amp's output stage in the long run, we'll drive the LED through an NPN transistor.

Next, try if this is still playable. It'll probably be a bit tricky, because the clock LED is blinking really fast: one flash per 16th note. **If you've used a metronome before, you'll know they usually tick in quarter notes – not 16th notes**. It's just a more natural pace to lock into. So let's try slowing our clock LED down to quarter notes as well.

THE QUARTER NOTE CLOCK LED

**For this, we'll use a simple binary counter chip: the CD4520**. This chip actually combines two 4-bit counters into one package. To start, we'll need to supply it with 12V and ground. Next, we pick one of the counters we want to use, and activate it by pulling its RESET input low and its ENABLE input high.
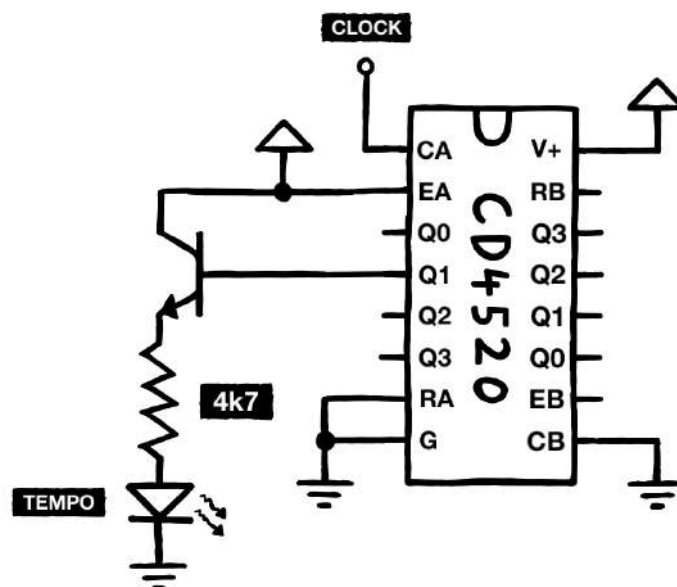
Now, when we feed it our 16th-note clock, the chip begins counting pulses in binary. Q0 toggles at half the input speed, giving us eighth notes. Q1 is half again, giving us quarter notes – and so on. **For now, we'll grab the quarter-note pulse from Q1, buffer it with an NPN transistor, and use that to drive our clock LED**. Finally, we'll disable the second counter by pulling its clock input low, so it stays inactive.[8]

---

[8] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/yrrdvw3j – you can change all values by double clicking on components.
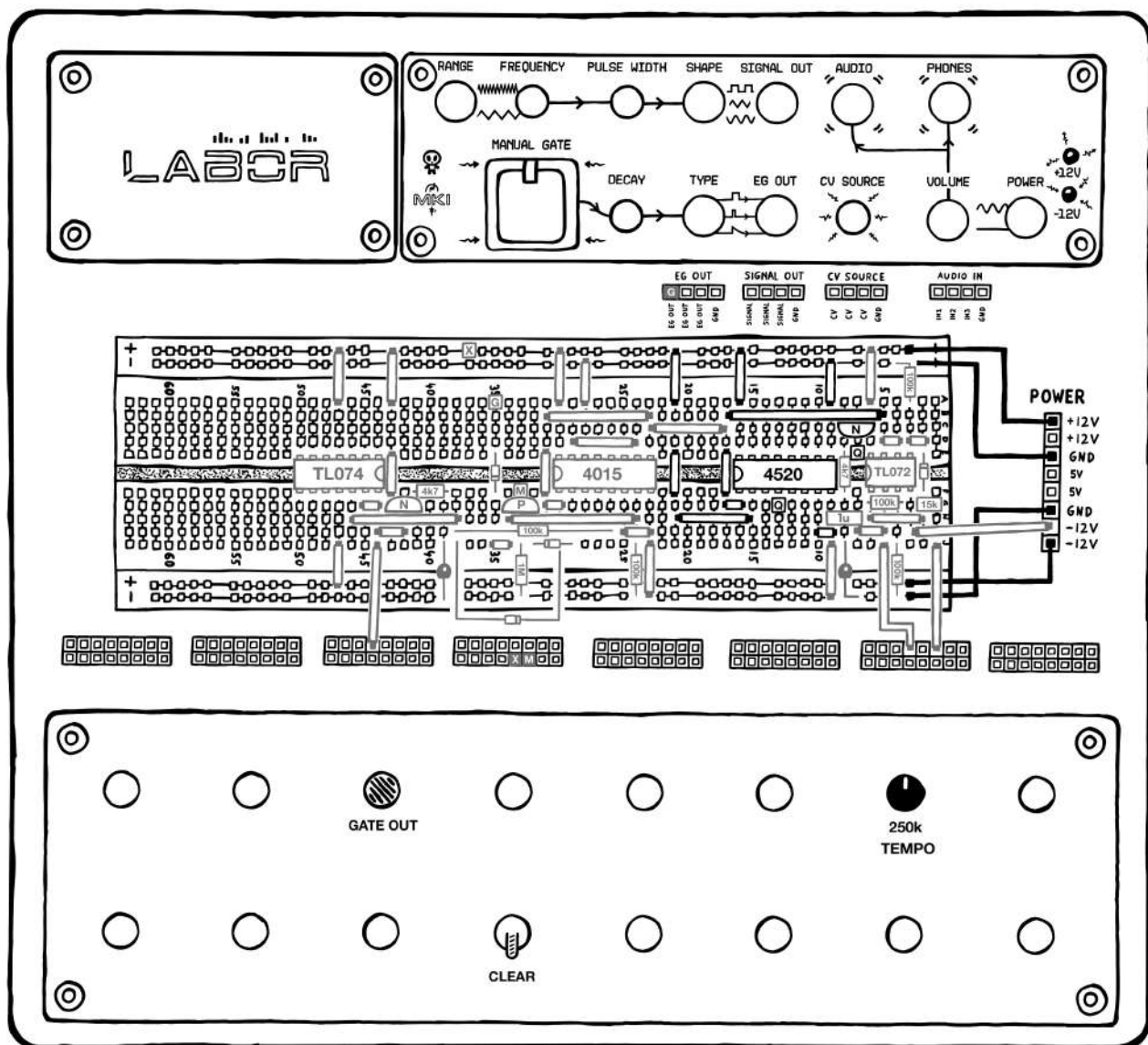
When you try this out, you'll notice that our LED blinks noticeably slower. Try if that makes tapping in a rhythm feel more natural! It should be significantly easier to lock into the groove like that. And if you want, you can even send that quarter note signal to another sound source – like a hi-hat or click module – for an audible metronome.

SMART ACCENTS

At this point, our sequencer is pretty fun to play. But it still sounds kind of flat. That's because every drum hit is identical: same volume, same intensity. **To fix this, most sequencers allow you to program accents – louder hits that give the pattern some dynamic variation**. Here's how it works: Drum circuits usually have both a trigger input and an accent input. The trigger tells the module *when* to play a sound – but the accent input controls the sound's *volume*: the higher the voltage, the louder the sound.

So in addition to our digital trigger sequence, we'll also need an analog accent sequence. The orthodox approach would be to generate one accent level per trigger.
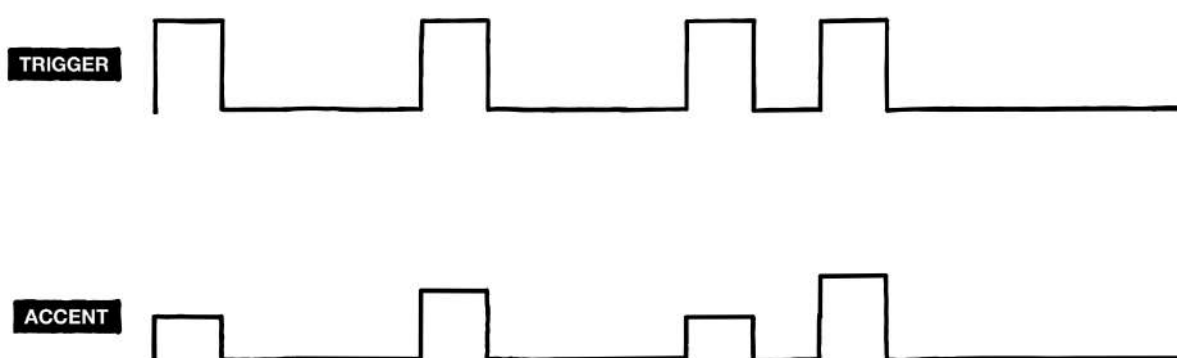

Doing that would add a lot of complexity to our circuit, though. So instead, let's see if we can cheat a little – and still get something that sounds great. Rather than generating accent levels per step, we'll just use a slow triangle wave. Initially, this probably sounds wrong. The voltage of a triangle wave is always changing, and doesn't stay fixed from one trigger to the next. Here's the trick though: most drum circuits don't track the accent voltage continuously — they just sample it the moment the trigger hits. **So the triangle wave's position at that instant becomes the accent level**.
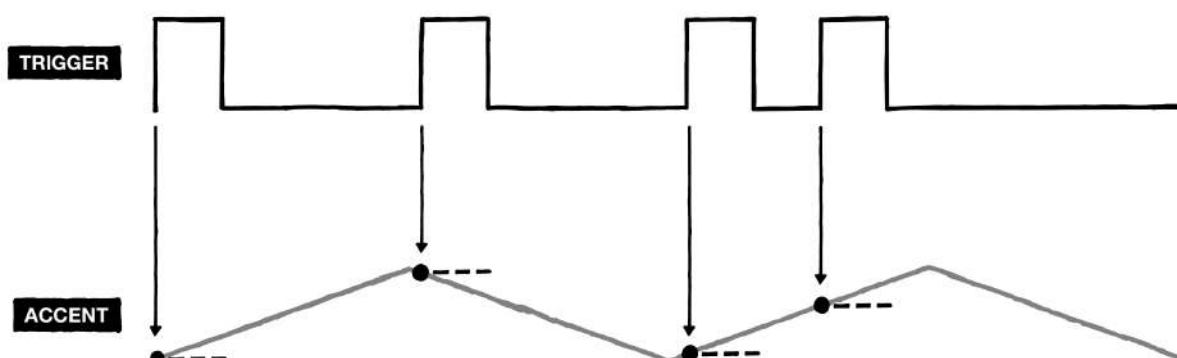
To generate a triangle wave, we'll set up a simple oscillator.
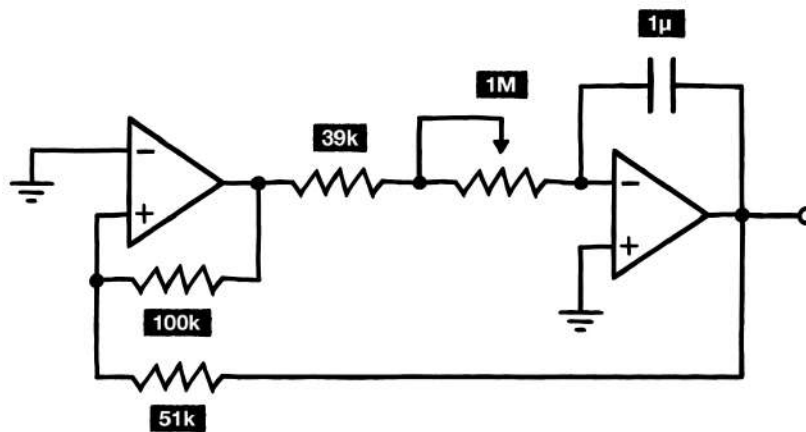
I've explained how this topology works in detail in our FM Drum manual, but here's a quick recap. **The circuit combines two building blocks: a Schmitt trigger (left op-amp) and an integrator (right op-amp), connected in a feedback loop**. On power-up, the Schmitt trigger latches either into its low or high state – let's assume it starts high. This pushes current through the integrator's input resistance and into the capacitor.
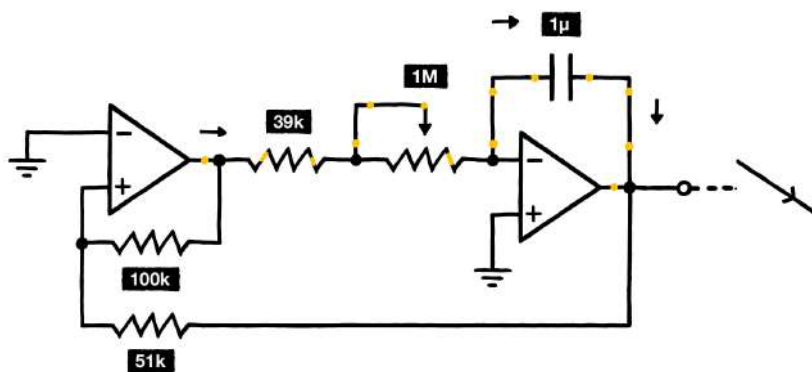
The integrator op amp is wired to neutralize any voltage change at its inverting input, so it counteracts by pulling current out of the other side of the capacitor. This results in a downward voltage ramp at its output. **Next, when the output has dropped far enough, the Schmitt trigger flips to its low state**. Now current flows the other way, causing the integrator's output to rise.
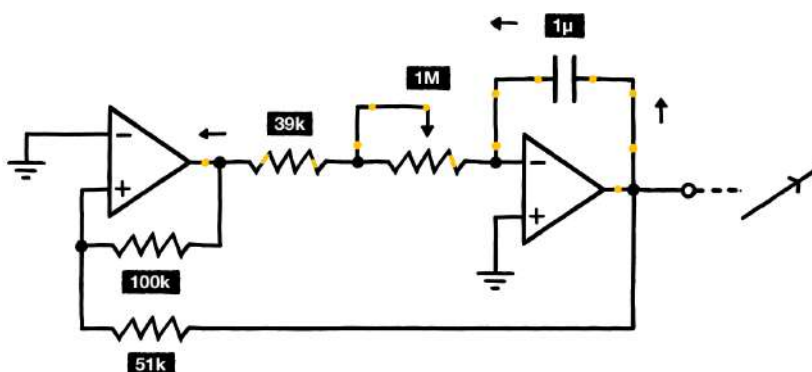
Eventually, that output climbs high enough to trigger another flip – and the cycle repeats. Unfortunately, there is one issue with this circuit. The output it produces swings between approximately 5 V and -5 V. **This is a problem, because some drum voices will not produce any sound when given a negative accent voltage**. To fix this, we'll need to lift the triangle above the 0 V line. How do we do that? Easy. All we need is a simple clamping circuit.
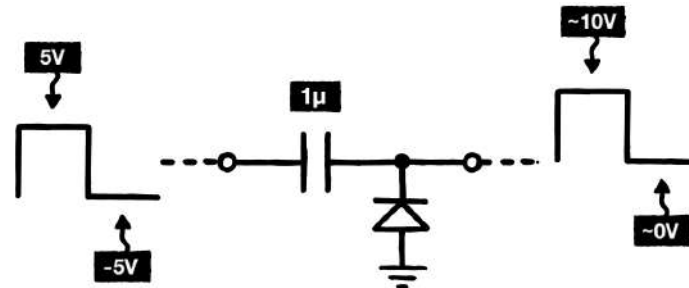

It works with any waveform – including our triangle wave – but it's easiest to explain using a square wave. So let's imagine we apply a ±5 V square to the input. During the square's first low phase, current gets pulled out of the cap and flows into it on the other side. This will leave us with -5 V on the left and about 0 V on the right. **Then, when the input swings high, the charge on the right side is trapped, since the diode blocks current from flowing back to ground**. So the voltage added on the left stacks on top of the 0 V on the right, giving us a 10 V output. And that's it: the signal is now oscillating above the 0 V line.

But there's a catch. We can't just use the signal directly – because pulling any current from the capacitor would mess with the clamping mechanism. So instead, we'll buffer the output using an emitter follower. That gives us a low-impedance version of the lifted signal, without interfering with the circuit.[9]
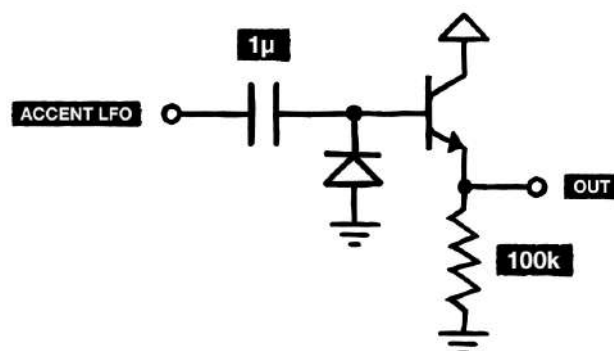
---

[9] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/ym4je73j – you can change all values by double clicking on components.
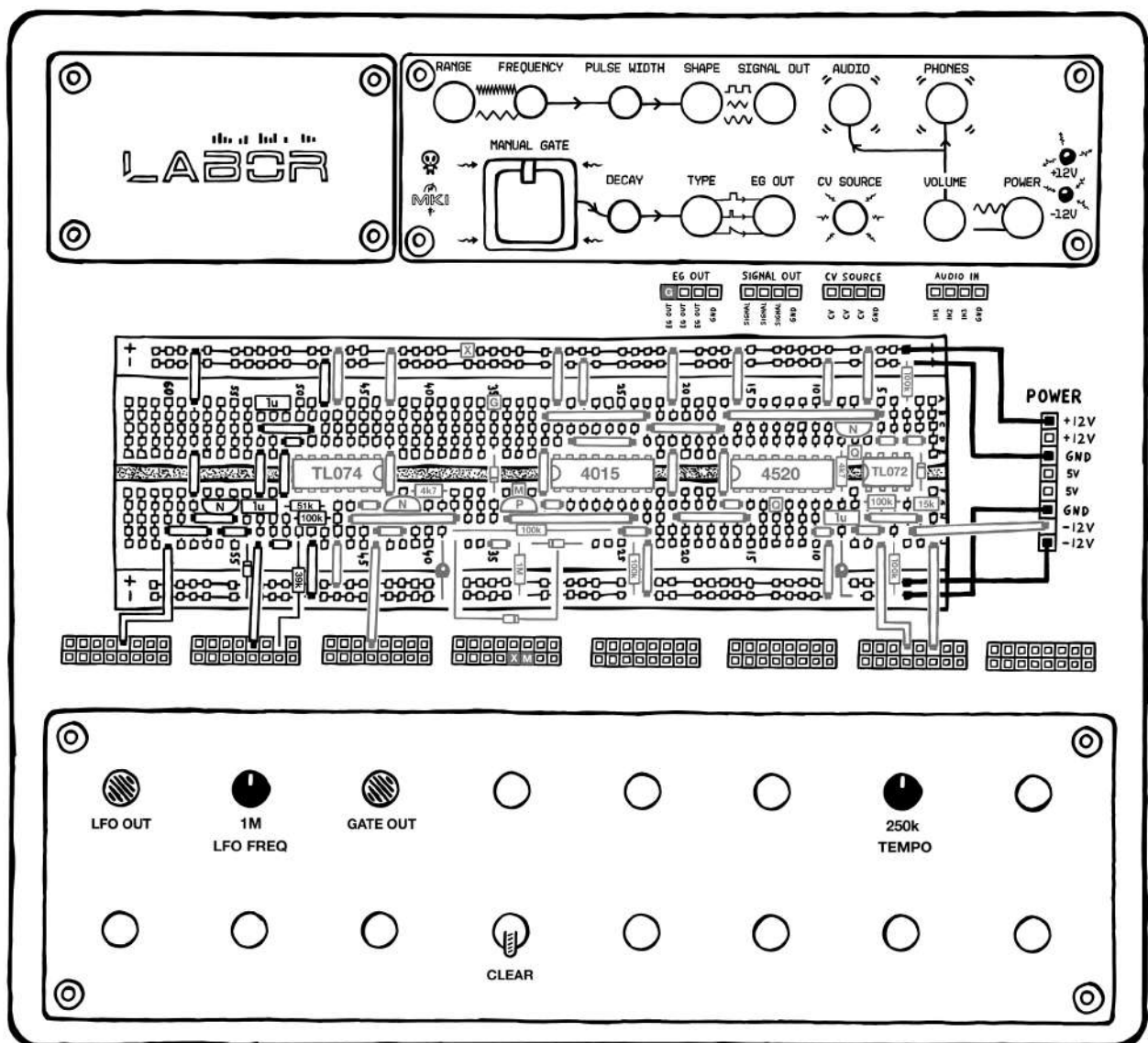
If you connect our new accent output to your drum module's accent input, you should get a wide range of dynamics, from barely audible to super loud. **Next, try adjusting the LFO's frequency**. Slow speeds should give you gentle swells, while faster ones create complex patterns that feel surprisingly deliberate.

ACCENT DEPTH CONTROL

That said, the dynamic range might be a little too extreme in some scenarios. So let's add an accent depth control that allows us to tame it. To get there, we'll simply replace the 100k resistor with a potentiometer. **That way, the pot works double duty: acting as the emitter follower's load and attenuating the LFO at the same time**. To keep the output impedance low, we'll buffer the result with yet another emitter follower.
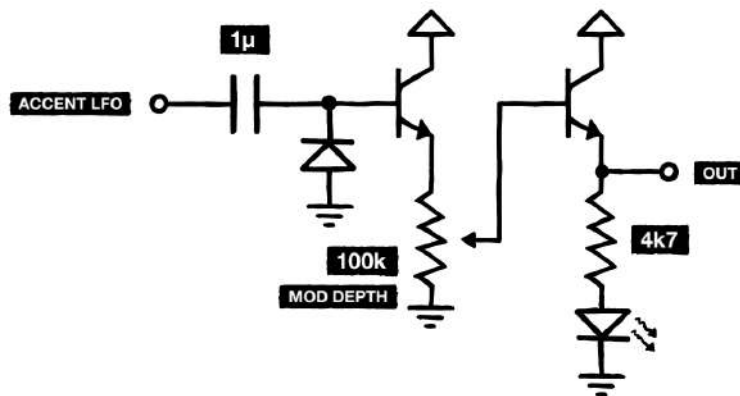
And why not throw in another LED for some visual feedback while we're at it?[10]
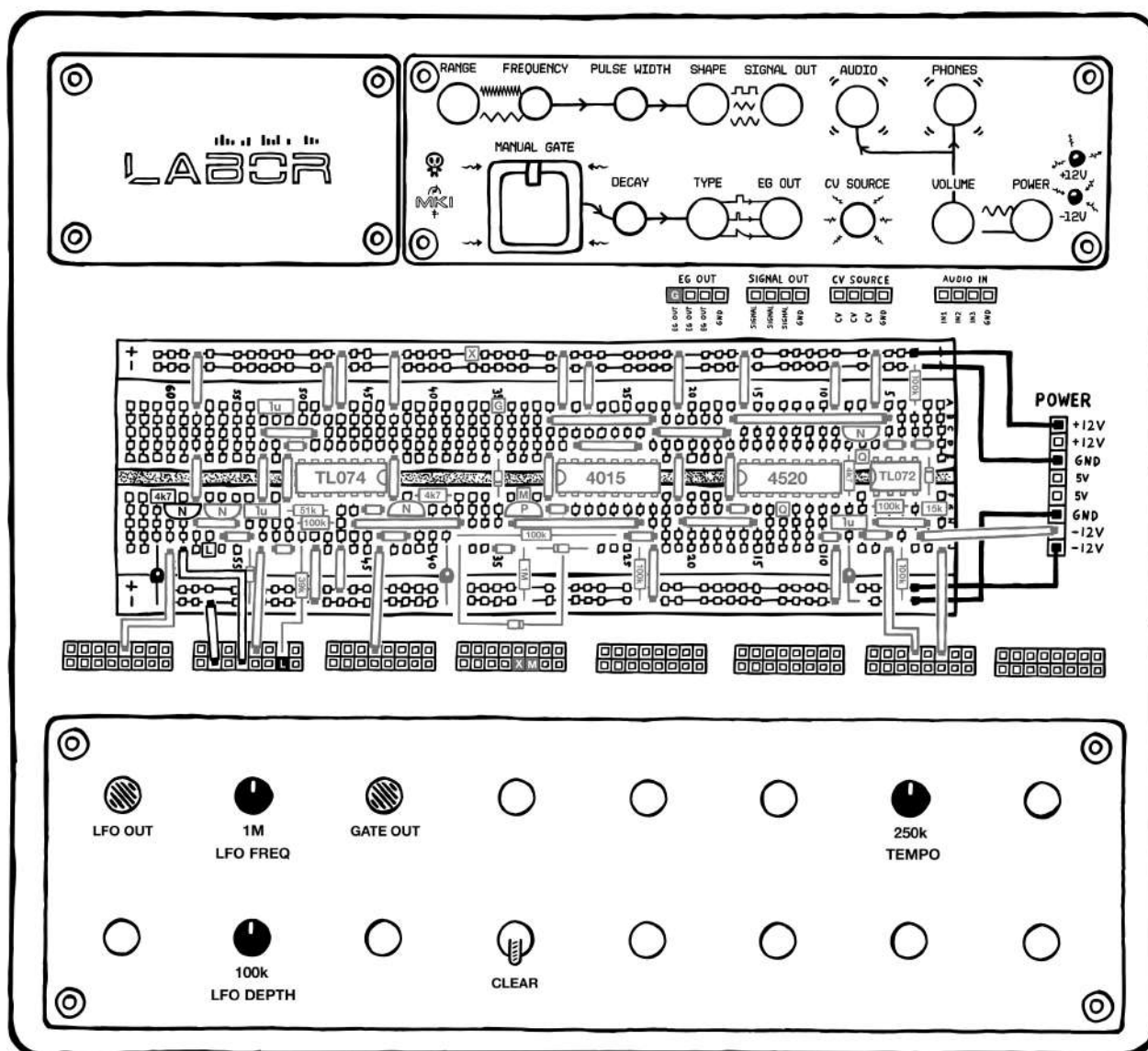
---

[10] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/ykzty2zx – you can change all values by double clicking on components.

Try turning the depth knob as you apply the LFO's output to your drum module's accent input. **You should be able to hear the maximum accent level drop**. At the lowest setting, the drum hits all come out at the same low level intensity. Crank it up, and we're back to the full dynamic range.

SYNCED ACCENTS

That said, you might've noticed something curious in how the accent pattern behaves. It's like the accents are sliding in and out of sync with the pattern itself. That's because our LFO is free-running. It's not synchronized to the clock. So the triangle wave cycles independently, and the accents fall in different spots each time the pattern loops.
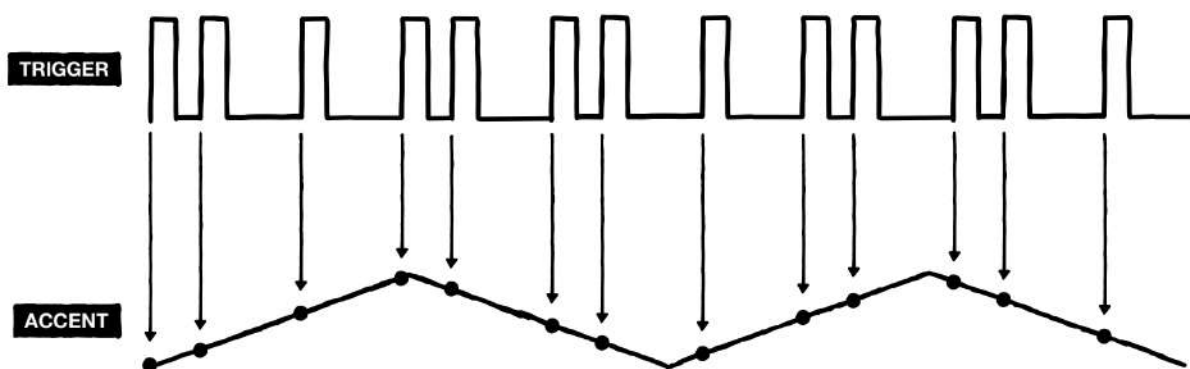

This isn't a bug, though. **It's what you'd call a polyrhythm: two rhythms with different rates running side by side**. This can sound surprisingly organic – especially in ambient or generative contexts.

But sometimes, you might want accents that line up with the beat predictably. How can we make that happen? One idea would be to carefully adjust the triangle's frequency so that the accent pattern repeats exactly once per loop. But in an analog circuit, that's easier said than done. **Even tiny mismatches add up, and the two signals start drifting apart**. So here's a better idea: what if we just forced the triangle to restart from zero at the beginning of each loop?
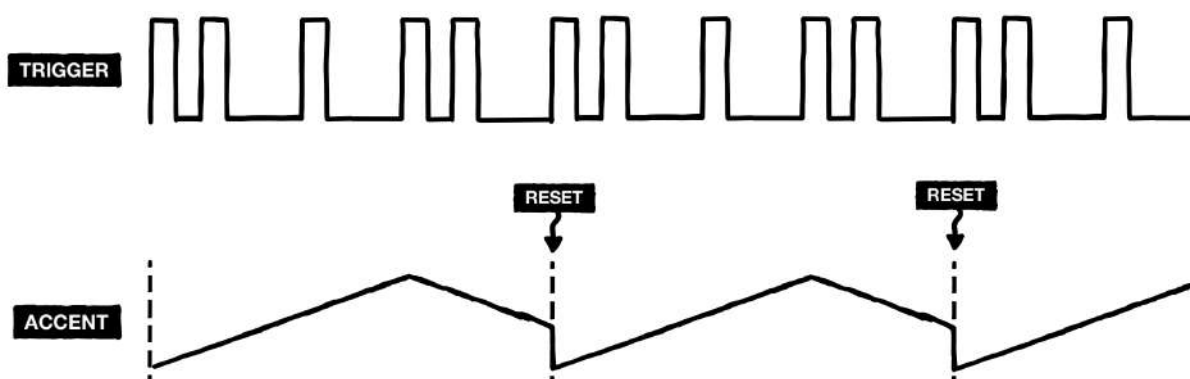

That would give us a perfectly repeating accent pattern – regardless of the triangle's base frequency. No drifting, no fine-tuning required. And while that might sound tricky to pull off, the actual solution is surprisingly simple.

Here's how it works. Our triangle wave is generated by charging and discharging this capacitor. **So the exact position in the wavecycle – whether we're on the way up or the way down – depends on how charge is distributed across the cap**.
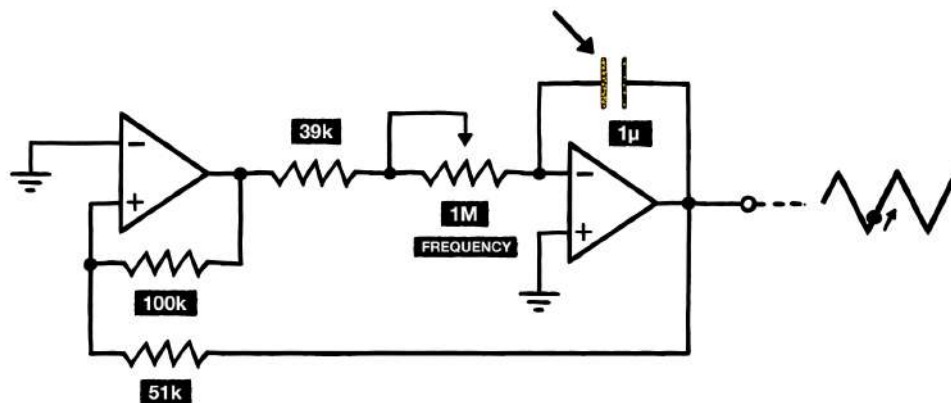

If we suddenly even it out, the oscillation should snap back to its starting point. Okay – but how do we do that? Easy: we just need to bridge the capacitor with a resistor and an NPN transistor.
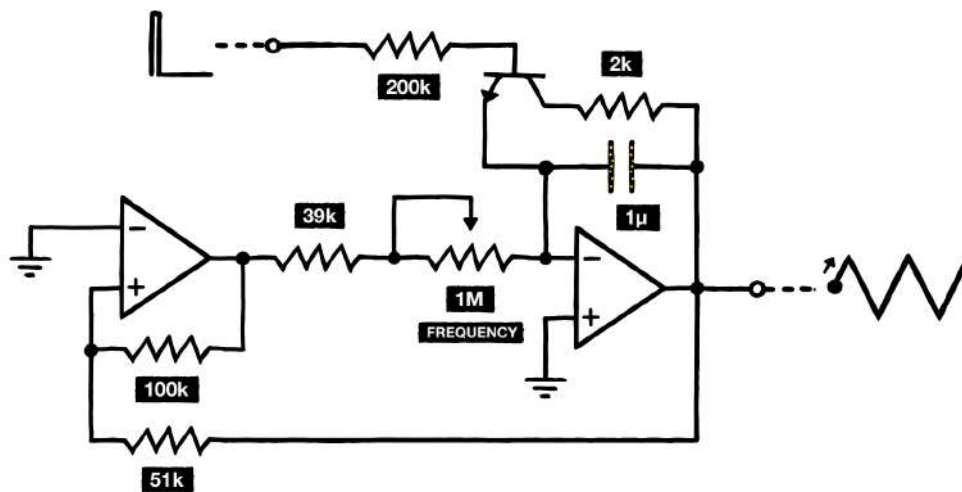
Then, we apply a short voltage pulse to the big base resistor, which activates the bridge and allows the cap to quickly discharge. **This resets the integrator's output to 0V**. Once the voltage pulse disappears, the oscillation simply resumes from that point.[11] Great – but where do we get this voltage pulse? Well, we said that we want to reset the triangle at the beginning of each pattern loop. Our pattern is currently 8 steps long. **Conveniently, we already have a signal that jumps from low to high every 8 steps: the CD4520's half note output**.

---

[11] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/ynapjssj – you can change all values by double clicking on components.

But there's a small catch. That output stays high for 4 steps, which is way too long for our purposes. We don't want the reset transistor to stay on – we just need a quick blip that discharges the capacitor and then gets out of the way. So before we can use it, we'll need to shape the chip's output into a short voltage spike by adding a small cap at the oscillator's reset input.[12]
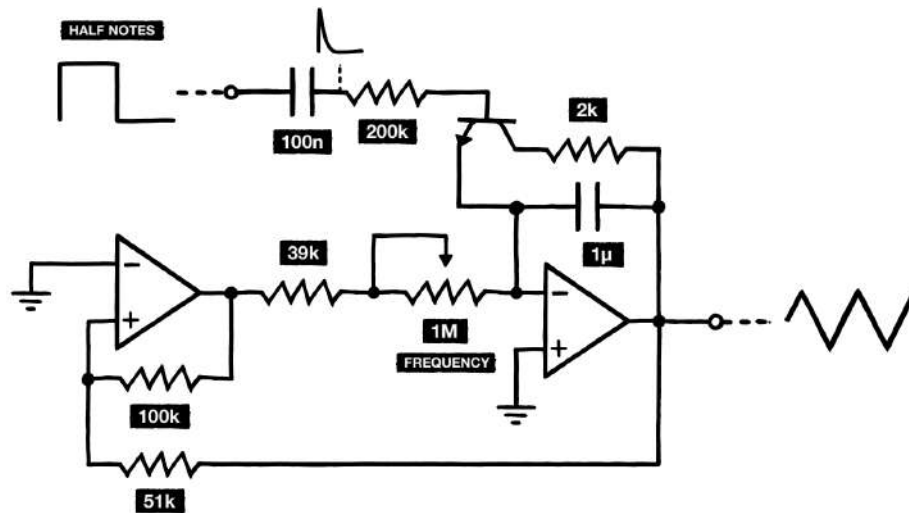
The cap only lets a quick burst of current through – right when the 4520's output first goes high – creating the short pulse we need. **And we can even take this a step further by setting up a three-way switch so we can either let the LFO run freely, sync it to the full pattern length, or sync it to half the pattern length for even straighter accents**.

---
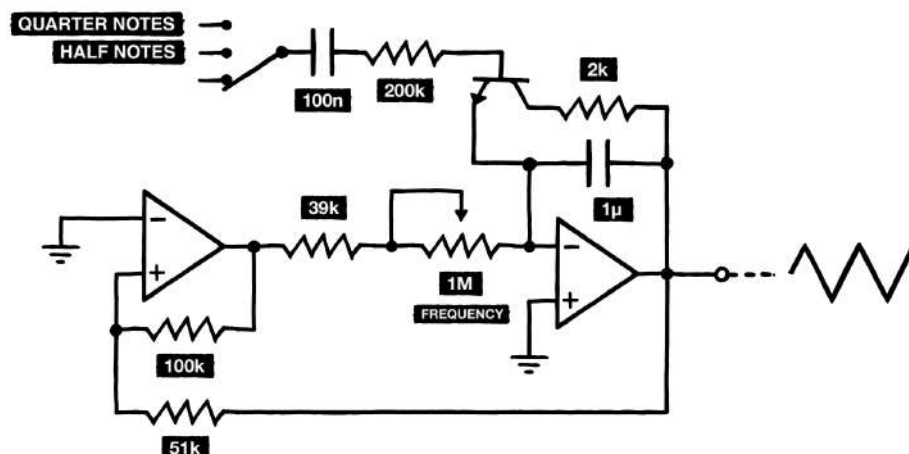
[12] Please note that this solution only works on a breadboard. On a PCB, we need to add a big 2M2 resistor to ground after the 100n capacitor to allow the cap to discharge between pulses. I believe this is due to small leakage currents that happen on a breadboard – but not on a PCB.
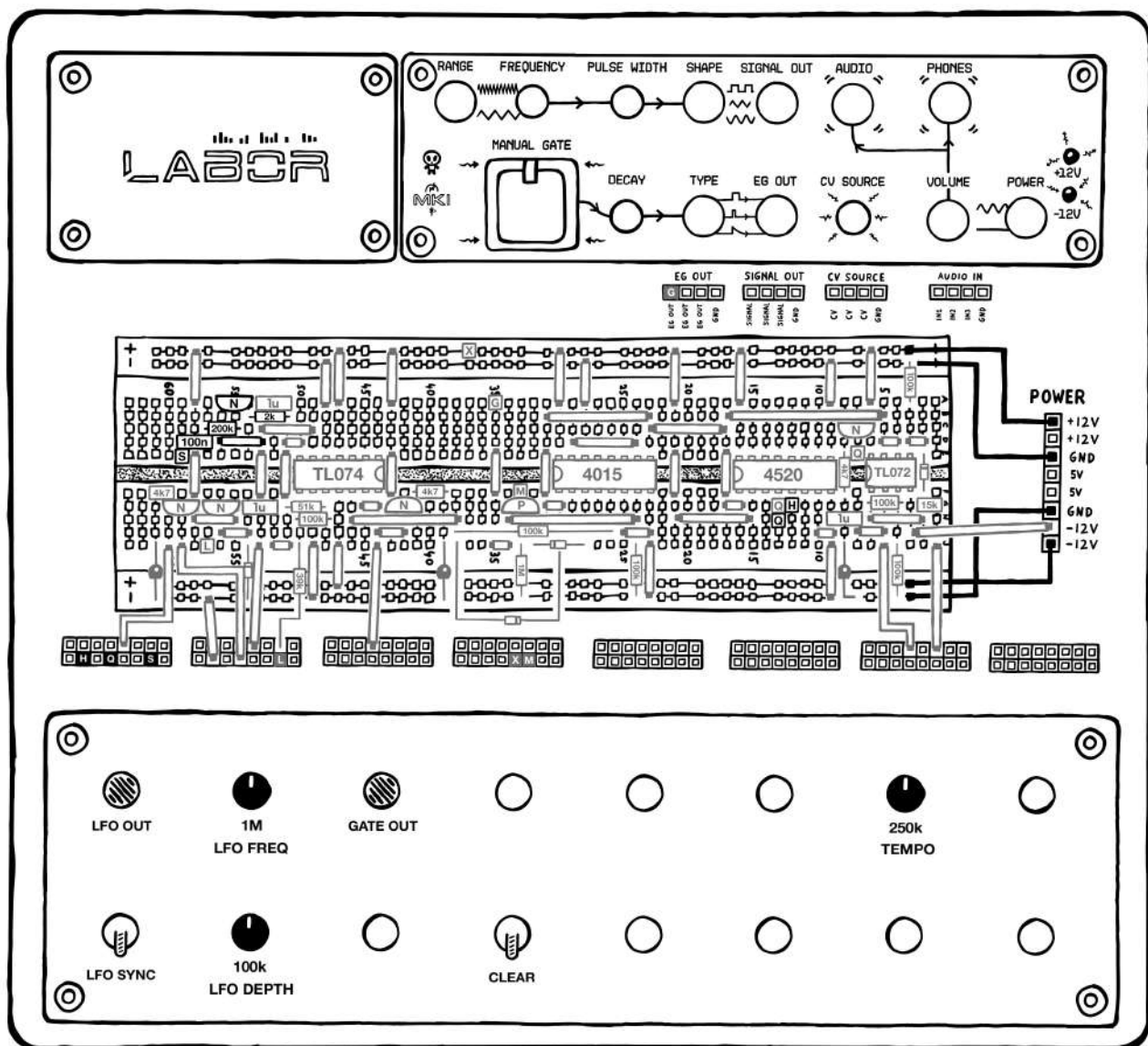
Okay, that was a long preamble – let's see if this actually works as intended. Try cycling through the three positions (no sync, quarter note sync, and half note sync), while adjusting the LFO's frequency. It should give you a huge range of possible accent patterns that sound deliberate even though they're basically generated by the circuit itself.

CLEAR MODE

At this point, we've built a feature-complete single-channel sequencer. But of course, a drum beat isn't just one sound. Even the most basic beats combine multiple voices – like a kick and hi-hat playing side by side. So realistically, one channel just isn't enough. **Thankfully, our design is so minimal — and so modular — that we can simply clone the sections we need**.

To get a second trigger pattern, we simply double the 4015-based tap looper. And for an additional accent sequence, we just need to set up another synced LFO. When I was setting up my second channel on another breadboard, I just had one small – and slightly silly – problem: I didn't have a second breadboard-friendly pushbutton for clearing steps. But that turned out to be a good thing: it gave me an excuse to simplify the interface even more.

Here's the idea: what if we could use the same button we tap to add steps… to remove them, as well? **All we'd need is a global modifier button that, when held down, turns every trigger button into a clear button**. That way, we could drop one button per channel — without losing any functionality. To recap: right now, pressing the per-channel clear button disables the PNP transistor in the feedback loop, which stops bits from looping back to the data input.
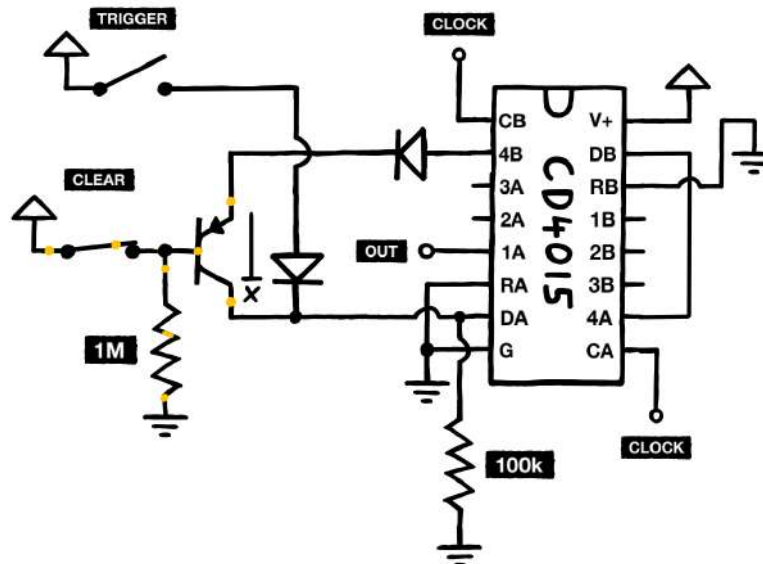

What we want now is to trigger that same behavior, but only when both a new, global clear mode button and the channel's trigger button are pressed together. To pull that off, we'll add an NPN transistor wired like this.
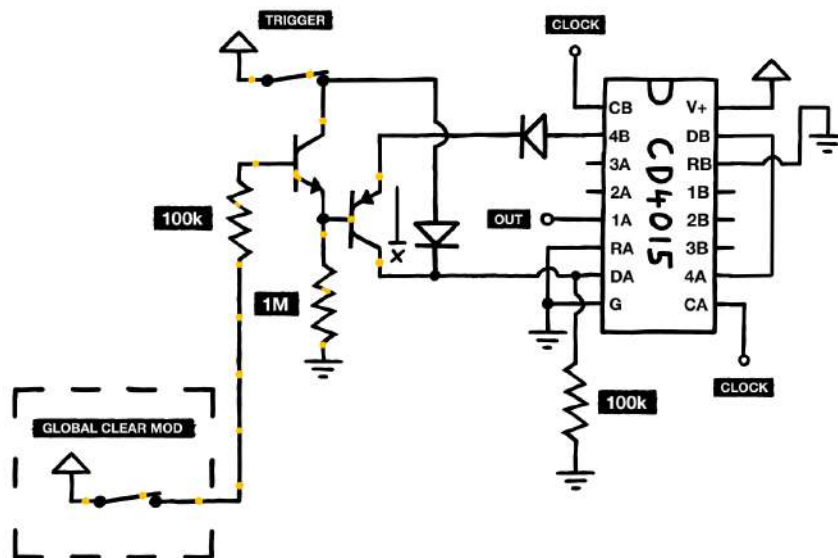
When both buttons are pressed, that NPN turns on, allowing current to flow. **This raises the PNP's base voltage, which turns it off and breaks the feedback loop**. But if either button is inactive, the NPN stays off and the loop continues.

But there is one issue with this idea: even though we break the feedback loop, we're still pulling the data input high – since the trigger button is still connected to it via the diode. So we end up writing a new step while trying to delete one, which cancels it out. To fix this, we'll add another PNP transistor between the trigger and the data input, controlled by the clear mode button.
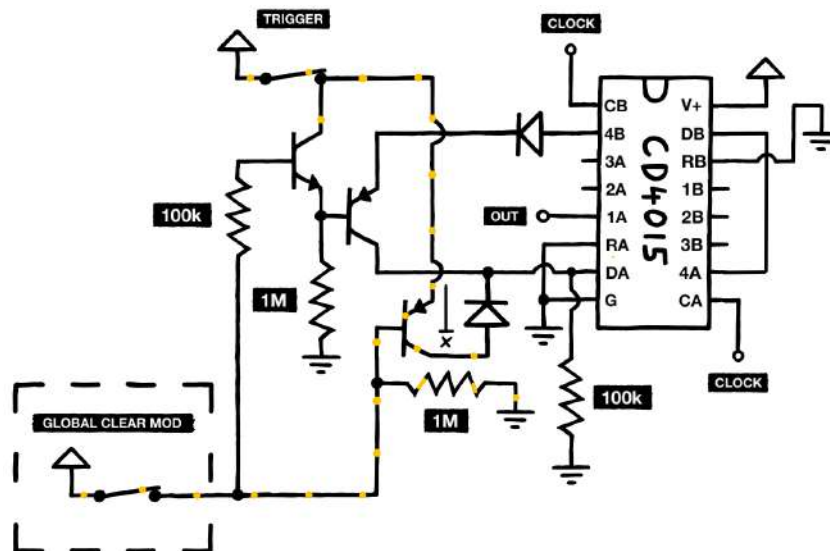
It works like this: when clear mode is off, the PNP conducts, and trigger presses behave normally. **But when clear mode is on, the PNP shuts off — cutting the connection**

and preventing new steps from being added. Now the trigger button should write steps or delete them — but never both at once.[13]

Now, to actually set up another sequencer channel, you'd need a second LABOR unit. I don't assume that you have one – but you can still test the global clear modifier. Again, since your kit does not contain a breadboard-friendly push button, we'll need to use a switch as a replacement.[14]
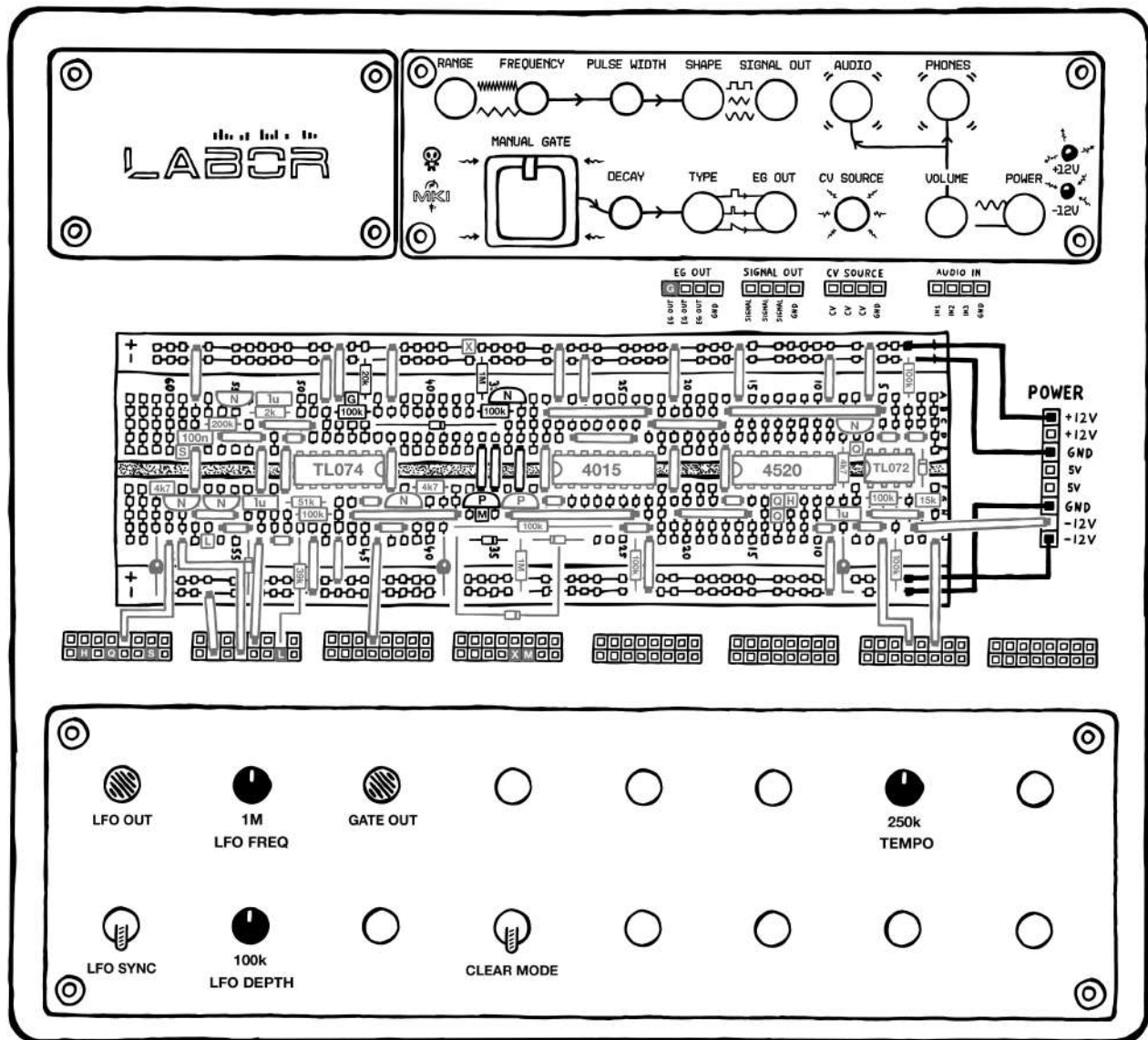
Try filling the entire register up with steps, then flip the clear mode switch and press the trigger button. You should be able to remove steps this way. Great! So now we have two voices (in theory), full control over each pattern, plus dynamic accents – all with just three buttons total. Not bad.

---

[13] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/2xfg6z7p – you can change all values by double clicking on components.

[14] Note that we also have to amplify the signal coming from the **MANUAL GATE** button, since it maxes out at 8 V – we need 12 V to fully close the PNP in the feedback loop though.

SEQUENCE LENGTH

But here's the exciting part: we're only just starting to squeeze real mileage out of this thing. With just a few small tweaks, we can push it even further. First up: sequence length. Earlier, we saw how a free-running LFO could create polyrhythmic accents. But that was modulation on top of a sequence. **What if we could get that same evolving movement between two trigger patterns?**

Right now, they both loop after 8 steps — so they always line up the same way. But if one looped, say, every 6 steps instead, we'd get a 3 over 4 polyrhythm, which should create some interesting rhythmic tension. Okay, so how do we shorten the pattern? Simple: by shortening the feedback loop. Right now, we're routing the shift register's 4B output back to the data input – which gives us an 8-step cycle. If we move that connection to the 2B output instead, we're down to a 6-step cycle. But why stop there? If we add a three-way switch to select the feedback tap point, we can toggle between multiple loop lengths on the fly.
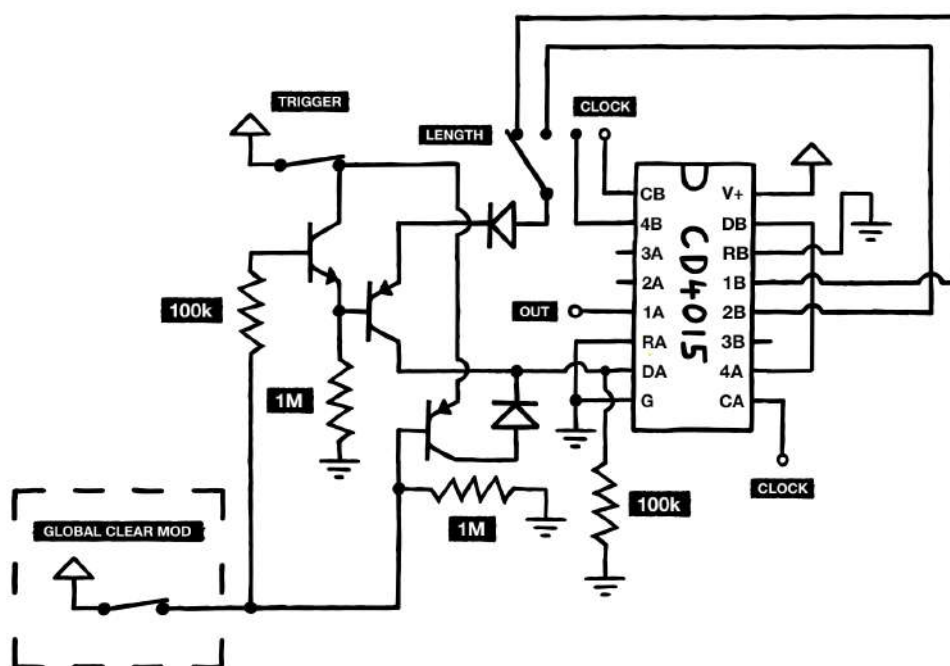
For example, 4B gives us 8 steps, 2B gives us 6, and 1B gives us just 5 — for an even more off-kilter feel.[15]

---

[15] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/yogt8qe3 – you can change all values by double clicking on components.

Since LABOR does not come with any more switches than we are already using, you'll have to source another double pole switch – or try this by connecting the different shift register outputs to the feedback path with a jumper cable. The finished module will use proper three way switches!
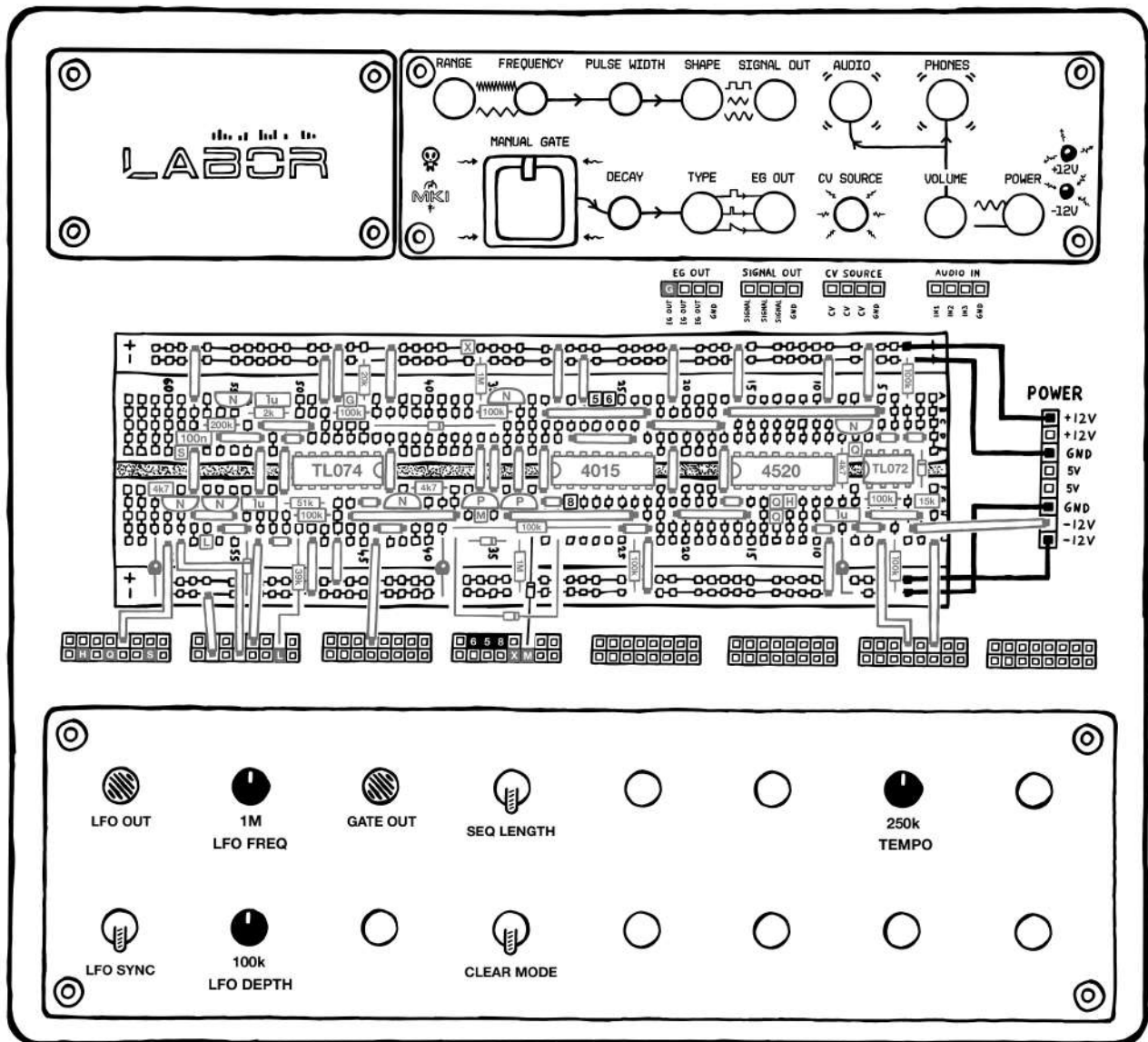
See what happens if you tap into the 2B or 1B outputs – you should get shorter patterns that feel less straight than the standard 8 step loop.

HALF TIME MODE

There's still more we can do – without adding much complexity. So far, our sequencer ran on a 16th-note clock. But what if we slowed it down to 8th notes? **That gives us the illusion of a longer pattern, at the cost of resolution – a feel that's usually referred to as half time**. To make it happen, we'll add a simple two-way switch at the tap looper's clock input.
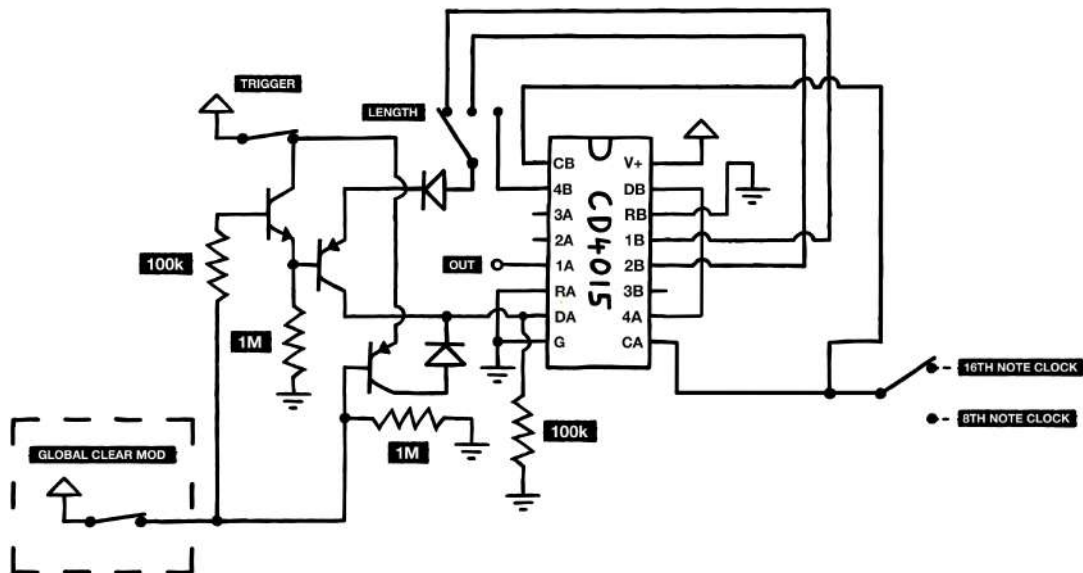
One side connects to the main 16th-note clock. The other taps into the 8th-note output from our clock divider.

Again, since both switches that come in the LABOR kit are already in use, you'll need to source another switch, or experiment with this by moving a jumper cable between the 16th note and 8th note clock.
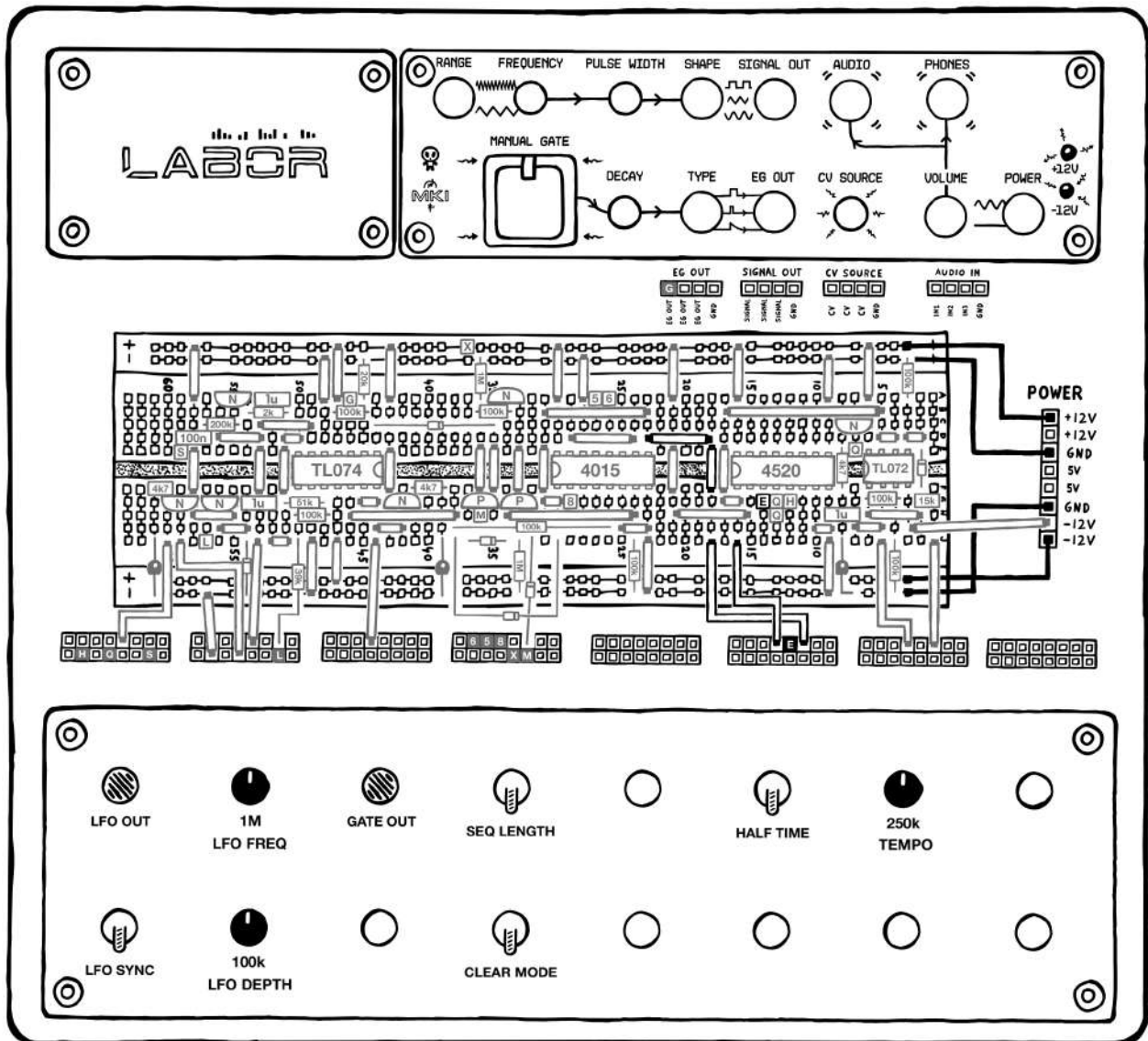
Switching to the 8th note clock should slow the playback speed down significantly. In the finished module, this will be very useful when you set one channel to half time mode, and another to the regular playback speed.

DAC CV

With just two switches and a handful of wires, we've greatly expanded what this circuit can do. But just before we wrap this up, I have one more idea. So far, we've used a triangle LFO for generating accents. But that's not our only option – we could also repurpose one channel's trigger output for this. This would give us full control over the accent pattern – but it's just binary: loud (12 V) or quiet (0 V). Let's try turning that into something a bit more nuanced.

By using the trigger output, we're only relying on a single shift register cell to drive accents. That cell just flips between high and low. But the register holds more data – 8 bits moving through it in parallel. **If we tap into multiple outputs at once, we can treat the combined signals as a binary number**.
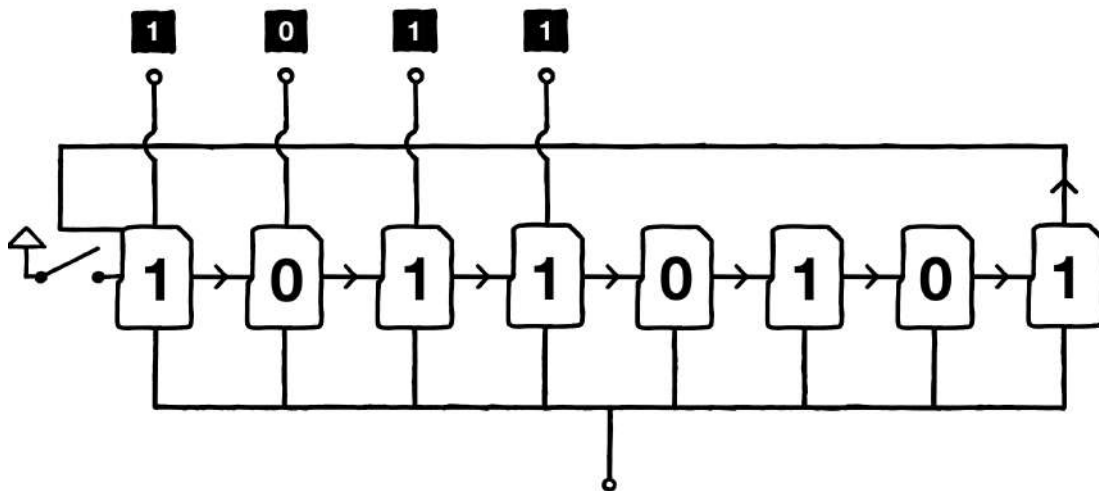

With four outputs, we get 16 distinct values. Now, here's the trick. We can convert each value into a corresponding analog voltage — and use that as our accent level. **Instead of just on/off, we'd then get stepped intensity: soft hits, loud hits, and 14 steps in between**. And the best part? We can do that conversion with just a couple of resistors and a buffer.
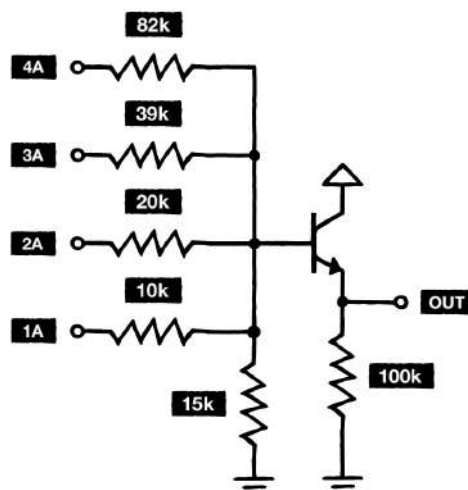
Here's how it works. We take four outputs from the shift register — let's say 1A through 4A — and run each through a resistor into the same point, like a simple passive mixer. But instead of using identical resistors, we'll pick values that roughly double from one to the next. **This way, each output has double the weight of the one preceding it – just like bits in a binary number**. This gives us 16 distinct voltage levels between 0 and 12 V, all evenly spaced.

Since modulation signals normally stay below 8 V, we'll scale this down using a 15k resistor to ground. And to keep the output impedance low, we'll finish it off with a simple emitter follower.[16]
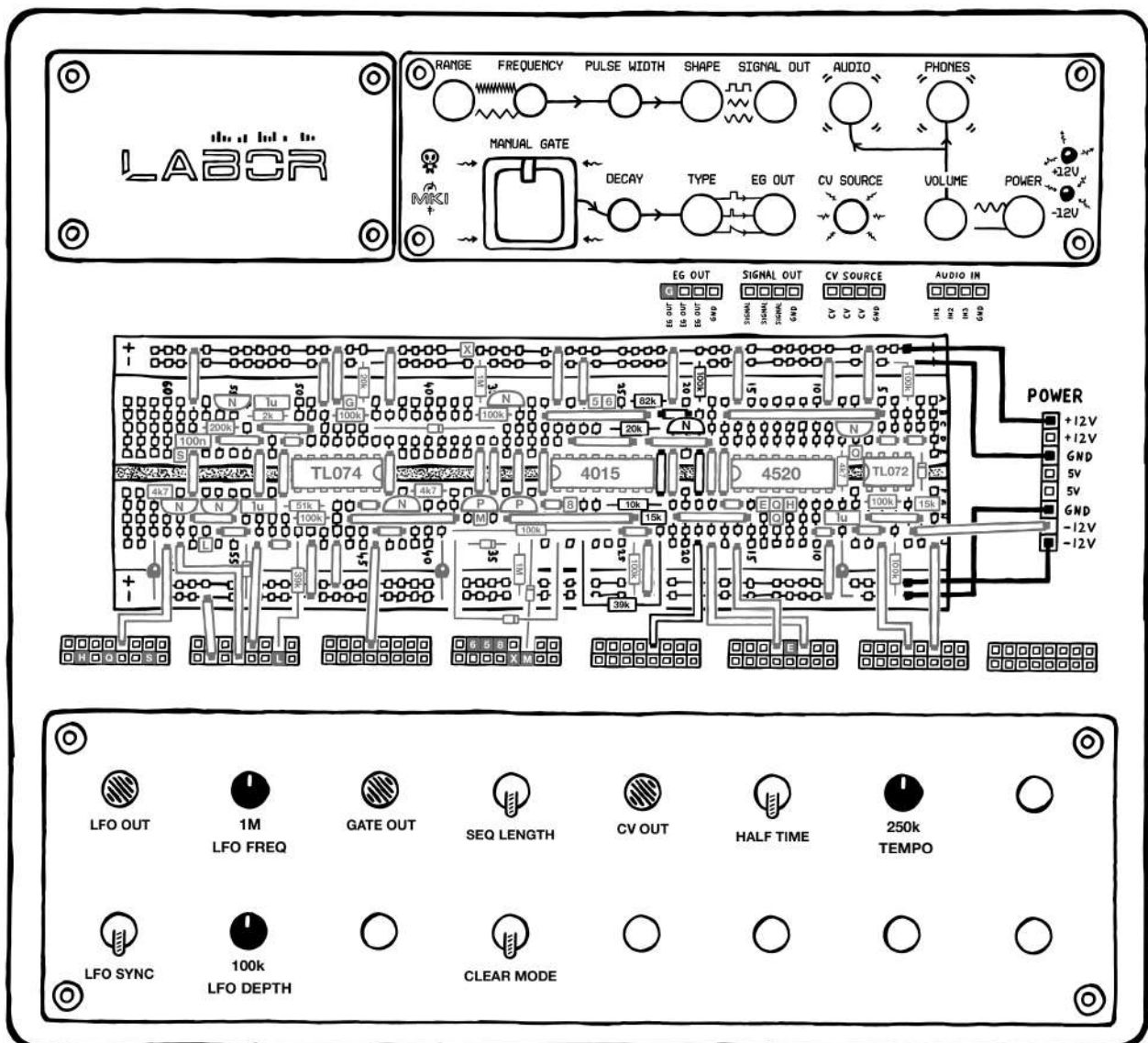
Once you've set this up, try patch the new CV output to the accent input of your drum module and hear how it sounds. (You can use the 16th note clock as a trigger signal.)

---

[16] You can try this chapter's circuit in a simulator. I've already set it up for you right here: https://tinyurl.com/yoejuo27 – you can change all values by double clicking on components.

Instead of just loud or quiet, you should now get a range of accent levels. The pattern you tap in is doing the heavy lifting — but it's padded with little ghost notes.

You can even try shuffling the shift register outputs to move those ghost notes around! And the best part? We're not limited to accents. Since this voltage jumps to a new fixed level on every clock pulse, we can also use it to control pitch – something that doesn't work with our LFO, since its output is always drifting. Try sending the CV output to your drum module's pitch/tune CV input. You won't have precise control over the notes — but that's kind of the point. It's fast and messy in a good way. Perfect for generating raw, rhythm-driven melodies with almost no effort.

And with this, our Drum Sequencer is complete. Once you're done experimenting, dig out the panel and PCB from the kit, heat up your soldering iron and get to building. You can find more information on how to populate the board & how to solder in the enclosed appendix.

COMPONENTS & CONCEPTS APPENDIX

In this section, we'll take a closer look at the components and elemental circuit design concepts we're using to build our module. Check these whenever the main manual moves a bit too fast for you!

THE BASICS:
RESISTANCE, VOLTAGE, CURRENT

There are three main properties we're interested in when talking about electronic circuits: resistance, voltage and current. To make these less abstract, we can use a common beginner's metaphor and compare the flow of electrons to the flow of water through a pipe.
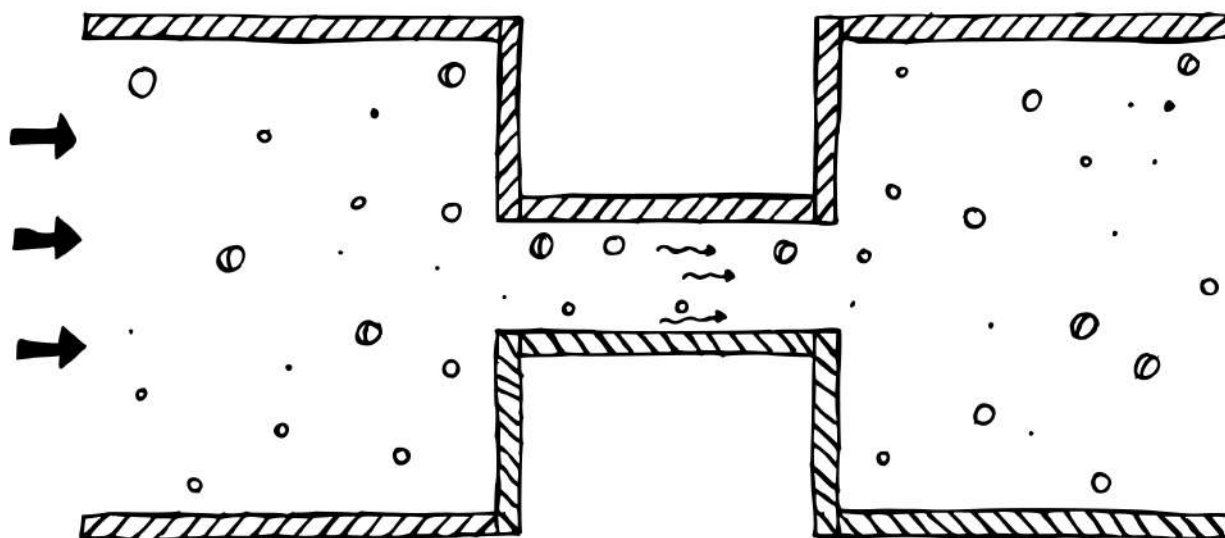
In that metaphor, resistance would be the width of a pipe. The wider it is, the more water can travel through it at once, and the easier it is to push a set amount from one end to the other. Current would then describe the flow, while voltage would describe the pressure pushing the water through the pipe. You can probably see how all three properties are interlinked: **more voltage increases the current, while more resistance to that voltage in turn decreases the current**.

USING TWO 9 V BATTERIES AS A DUAL POWER SUPPLY

Dual power supplies are great – and if you want to get serious about synth design, you should invest in one at some point. But what if you're just starting out, and you'd like to use batteries instead? Thankfully that's totally doable. **You just need to connect two 9 V batteries like shown here**. For this, you should use 9 V battery clips, which are cheap & widely available in every electronics shop.
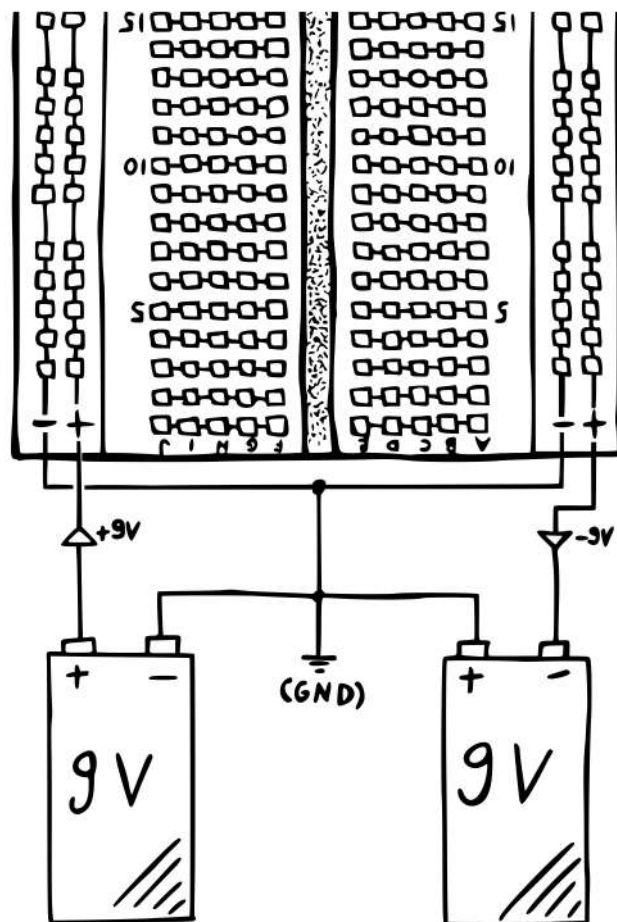
By connecting the batteries like this, the positive terminal of the left battery becomes your +9 V, while the negative terminal of the right is now your –9 V, and the other two combine to become your new ground.[17] **Please make sure you disconnect the batteries from your breadboard when you make changes to the circuit!** Otherwise you run the risk of damaging components.

---

[17] If you're struggling with setting this up, you can watch me do it here: https://youtu.be/XpMZoR3fgd0?t=742

RESISTORS

While a conductive wire is like a very big pipe where lots of water can pass through, **a resistor is like a narrow pipe that restricts the amount of water that can flow**. The narrowness of that pipe is equivalent to the resistance value, measured in ohms (Ω). The higher that value, the tighter the pipe.
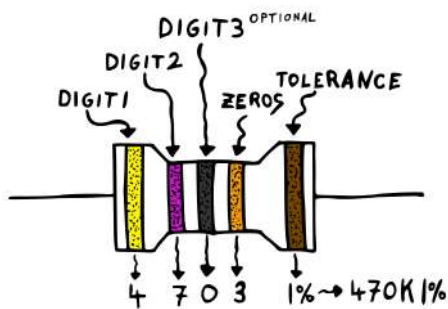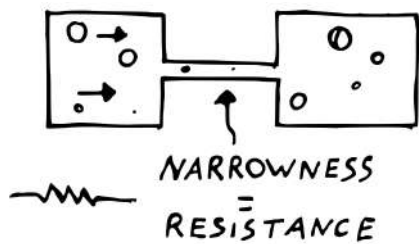
Resistors have two distinctive properties: linearity and symmetry. Linearity, in this context, means that for a doubling in voltage, the current flowing will double as well. Symmetry means that the direction of flow doesn't matter – resistors work the same either way.

On a real-life resistor, you'll notice that its value is not printed on the outside – like it is with other components. Instead, it is indicated by colored stripes[18] – along with the resistor's tolerance rating. In addition to that, the resistor itself is also colored. Sometimes, depending on who made the resistor, this will be an additional tolerance indicator.

For the resistors in this kit, a yellow body tells you that the actual resistance value might be ±5 % off. A dark blue body indicates ±1 % tolerance. Some kits will also contain light blue ± 0.1% resistors to avoid the need for manual resistor matching.

While in the long run, learning all these color codes will be quite helpful, you can also simply use a multimeter to determine a resistor's value.
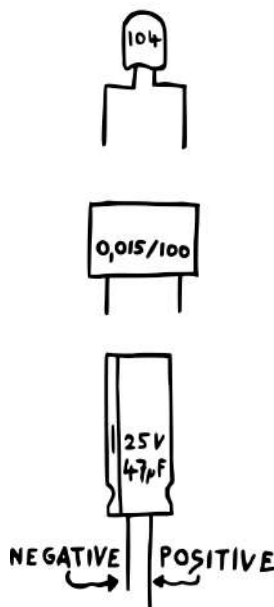

CAPACITORS

A capacitor is a bit like a balloon that you can attach to the open end of a pipe. If there's some pressure in the pipe, the balloon will fill up with water until the pressure equalizes. (Since the balloon needs some space to expand into, both of the capacitor's legs need to be connected to points in your circuit.)
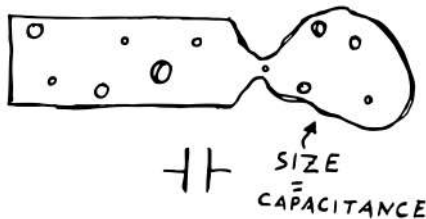
Then, should the pressure in the pipe drop, the balloon releases the water it stored into the pipe. The maximum size of the balloon is determined by the capacitor's capacitance, which we measure in farad (F). There are quite a few different types of capacitors: electrolytic, foil, ceramic, tantalum etc. They all have their unique properties and ideal usage scenarios – but the most important distinction is if they are polarized or not.

You shouldn't use polarized capacitors against their polarization (applying a negative voltage to their positive terminal and vice versa) – so they're out for most audio-related uses like AC coupling, high- & low-pass filters etc.

Unlike resistors, capacitors have their capacitance value printed onto their casing, sometimes together with a maximum operating voltage. **Be extra careful here!** That voltage rating is important. Your capacitors can actually explode if you exceed it! So they should be able to withstand the maximum voltage used in your circuit. If they're rated higher – even better, since it will increase their lifespan. No worries though: the capacitors in this kit are carefully chosen to work properly in this circuit.

Ceramic capacitors usually come in disk- or pillow-like cases, are non-polarized and typically encode their capacitance value.[19] Annoyingly, they rarely indicate their voltage rating – so you'll have to note it down when buying them.

Film capacitors come in rectangular, boxy cases, are non-polarized and sometimes, but not always, directly indicate their capacitance value and their voltage rating without any form of encoding.[20]

Electrolytic capacitors can be identified by their cylinder shape and silver top, and they usually directly indicate their capacitance value and their voltage rating. They are polarized – so make sure you put them into your circuit in the correct orientation.

---

[19] For a detailed breakdown, look up ceramic capacitor value code. There are also calculation tools available.

[20] If yours do encode their values, same idea applies here – look up film capacitor value code.

DIODES
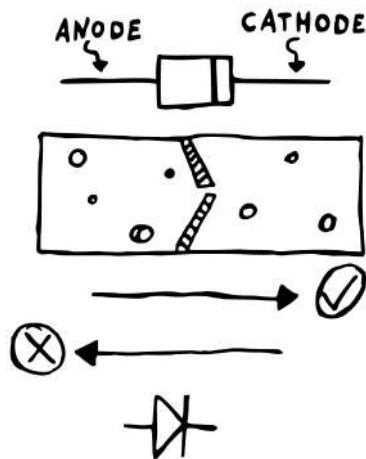
Diodes are basically like one-way valves. **Current can only pass through in one direction – from anode to cathode**. That direction is indicated by the arrow in the diode symbol and by a black stripe on the diode's casing. So any current trying to move in the opposite direction is blocked from flowing.

There are a few quirks here, though. For one, the diode will only open up if the pushing force is strong enough. Generally, people say that's 0.7 V**,** but in reality, it's usually a bit lower. Also, diodes don't open up abruptly – they start conducting even at much lower voltages, although just slightly.

There are a lot of different diode types: Zener, Schottky, rectifier, small signal etc. They all have their unique properties and ideal usage scenarios – but usually, a generic 1N4148 small signal diode will get the job done.

SCHMITT TRIGGER INVERTERS
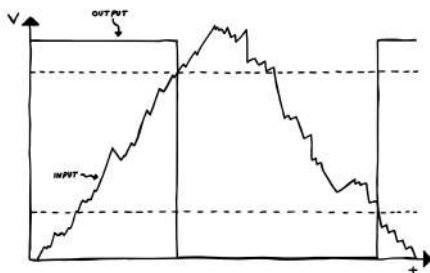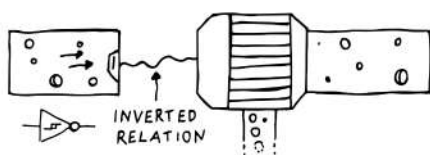
You can think of a Schmitt trigger inverter as two separate things. **On the left, there's a sensor that measures the pressure inside an attached pipe**. **On the right, there is a water pump**. This pump's operation is controlled by the sensor. Whenever the pressure probed by this sensor is below a certain threshold, the pump will be working. If the pressure is above a second threshold, the pump won't be working.

Here's a quick graph to visualize that. The squiggly line represents the voltage at the input, while the dotted line shows the voltage at the output. So every time we cross the upper threshold on our way up, and the lower one on our way down, the output changes its state. One thing that's very important to keep in mind: no current flows into the sensor! It's really just sensing the voltage without affecting it.

VOLTAGE DIVIDERS
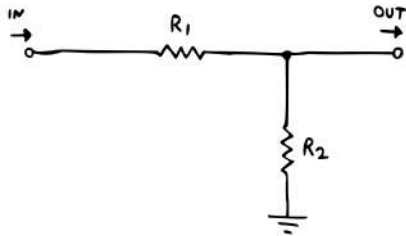
A voltage divider is really just two resistors set up like this: input on the left, output on the right. If R1 and R2 are of the same value, the output voltage will be half of what the input voltage is. How does it work?

Let's use our analogy again: so we have a pipe on the left, where water is being pushed to the right with a specific amount of force. Attached to it is a narrow pipe, representing R1, followed by another wide pipe. Then at the bottom, there's another narrow pipe, representing R2, where water can exit the pipe system. Finally, imagine we've set up a sensor measuring the voltage in the right hand pipe.

First, think about what would happen if R2 was completely sealed off. Our sensor would tell us that **the pressure on the right side is exactly the same as the pressure on the left**. Because the pushing force has nowhere else to go.

On the other hand, imagine R2 would just be a wide opening. Then **the pressure on the right would be 0**, because it'd all escape through that opening. But what happens if R2 is neither completely closed off nor wide open? Then the pressure would be retained to varying degrees, depending on the narrowness of the two resistor paths.

If pipe R1 is wide and pipe R2 is narrow, most of the pressure will be retained. But if it's the reverse, the pressure level will be only a tiny fraction. And if R1 and R2 are identical, **the pressure will be exactly half of what we send in**.
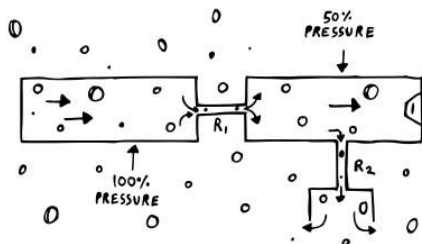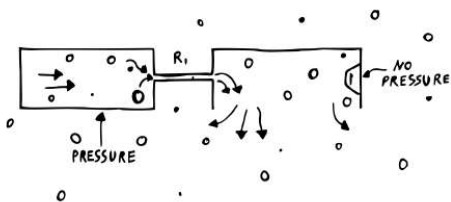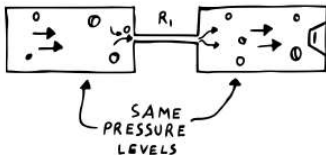
POTENTIOMETERS

Potentiometers can be used as variable resistors that you control by turning a knob.
But, and that's the handy part, they can also be set up as variable voltage dividers.
To see how that works, let's imagine we open one up.
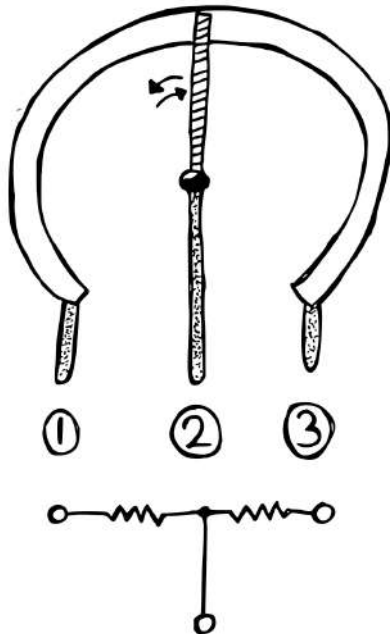
Inside, we would find two things: a round track of resistive material with connectors on both ends plus what's called a wiper. This wiper makes contact with the track and also has a connector. It can be moved to any position on the track. Now, the resistance value between the two track connectors is always going to stay exactly the same. That's why it's used to identify a potentiometer: as a 10k, 20k, 100k etc. But if you look at the resistance between either of those connectors and the wiper connector, you'll find that this is completely dependent on the wiper's position.

The logic here is really simple: **the closer the wiper is to a track connector, the lower the resistance is going to be between the two**. So if the wiper is dead in the middle, you'll have 50 % of the total resistance between each track connector and the wiper.

From here, you can move it in either direction and thereby shift the ratio between the two resistances to be whatever you want it to be. By now, you might be able to see how that relates to our voltage divider. If we send our input signal to connector 1 while grounding connector 3, we can pick up our output signal from the wiper. Then by turning the potentiometer's knob, we can adjust the voltage level from 0 to the input voltage – and anything in between.
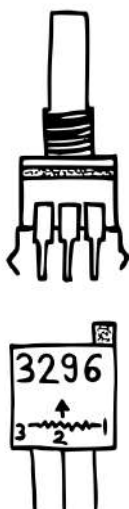
In these kits, you will encounter different types of potentiometers. First, there's the regular, full-size variant with a long shaft on top. These are used to implement user-facing controls on the module's panel and they usually – but not always – indicate their value directly on their casing. Sometimes, they'll use a similar encoding strategy as capacitors, though.[21]

Second, we've got the trimmer potentiometer, which is usually much smaller and doesn't sport a shaft on top. Instead, these have a small screw head which is supposed to be used for one-time set-and-forget calibrations. Trimmers usually encode their value.

---

[21] Look up potentiometer value code for a detailed breakdown.

AC COUPLING

What is AC coupling – and how does it work? Imagine two adjacent pipes with a balloon between them. Now, no water can get from one pipe into the other, since it's blocked by the balloon. But, and that's the kicker, **water from one side can still push into the other by bending and stretching the balloon, causing a flow by displacement**.
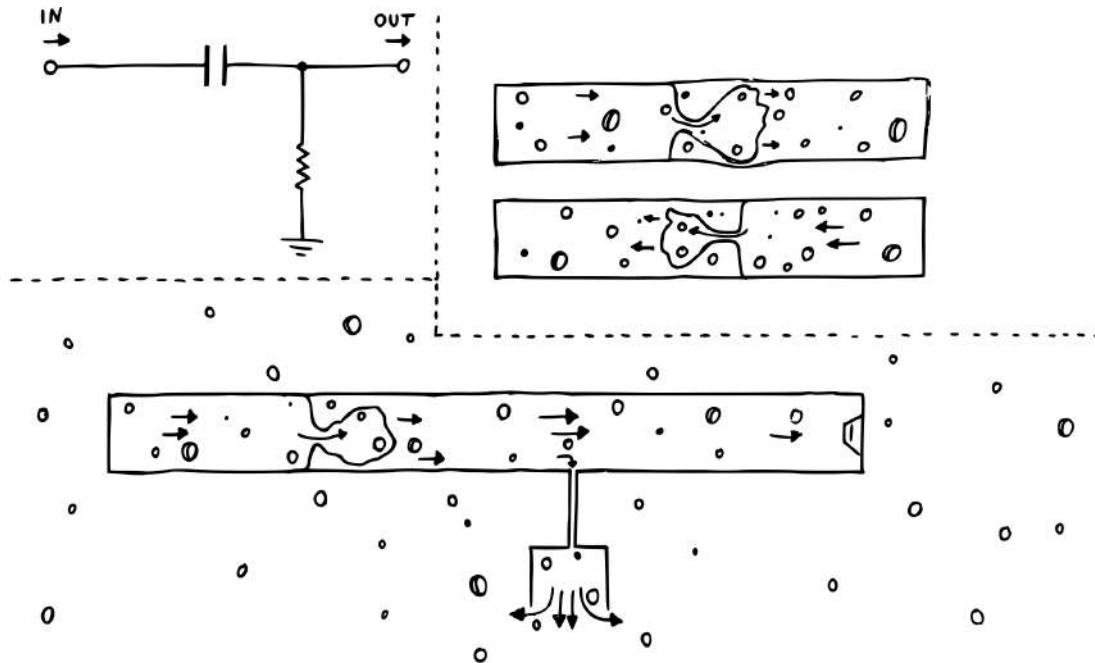

Next, we'll bring in a resistor after the coupling point, going straight to ground. **This acts like a kind of equalizing valve**. Now imagine we apply a steady 5 V from one side. Then on the other side, we'll read 0 V after a short amount of time. Why? Because we're pushing water into the balloon with a constant force, causing it to stretch into the other side, displacing some water. If we didn't have the equalizing valve there, we'd simply raise the pressure. But since we do have it, the excess water can drain out of the system. Until the pressure is neutralized, and no water is actively flowing anymore.

Okay, so now imagine that the voltage on the left hand side starts oscillating, let's say between 4 V and 6 V. When we start to go below 5 V, the balloon will begin contracting, basically pulling the water to the left. This will create a negative voltage level in the right hand pipe – like as if you're sucking on a straw, making the voltage there drop below 0 V. Then, once the pressure on the other side rises above 5 V, the balloon will inflate and stretch out again, pushing water to the right. And the pressure in the right hand pipe will go positive, making the voltage rise above 0 V. **We've re-centered our oscillation around the 0 V line.** Okay, but what about the resistor? If current can escape through it, doesn't that mess with our oscillation? Well, technically yes, but practically, we're choosing a narrow enough pipe to make the effect on quick pressure changes negligible!

OP AMPS

Op amps might seem intimidating at first, but they're actually quite easy to understand and use. The basic concept is this: every op amp has two inputs and one output. Think of those inputs like voltage sensors. You can attach them to any point in your circuit and they will detect the voltage there without interfering. **No current flows into the op amps inputs – that's why we say their input impedance is very high**. Near infinite, actually. Okay, but why are there two of them?
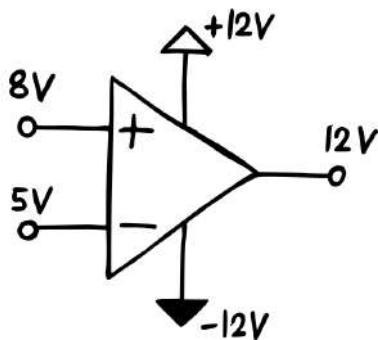
The key here is that op amps are essentially differential amplifiers. This means that they only amplify the difference between their two inputs – not each of them individually. If that sounds confusing, let's check out a quick example. So we'll imagine that one sensor – called the non-inverting input – is reading 8 V from somewhere. The other sensor – called the inverting input – reads 5 V. Then, as a first step, the op amp will subtract the inverting input's value from the non-inverting input's value. Leaving us with a result of 3. (Because 8 minus 5 is 3.) **This result then gets multiplied by a very large number – called the op amp's gain**. Finally, the op amp will try to push out a voltage that corresponds to that multiplication's result.

But of course, the op amp is limited here by the voltages that we supply it with. If we give it –12 V as a minimum and +12 V as a maximum, the highest it can go will be +12 V. So in our example, even though the result of that multiplication would be huge, the op amp will simply push out 12 V here and call it a day.

The handy thing though about op amp outputs is that they draw their power directly from the power source. This means that they can supply lots of current while keeping the voltage stable. **That's why we say an op amp has a very low output impedance**.

OP AMP BUFFERS/AMPLIFIERS

Buffering, in the world of electronics, means that we provide a perfect copy of a voltage without interfering with that voltage in the process. With an op amp-based buffer, the buffering process itself works like this. We use the non-inverting input to probe a voltage, while the inverting input connects straight to the op amp's output. **This creates what we call a negative feedback loop**. Think of it this way. We apply a specific voltage level to the non-inverting input – let's say 5 V.
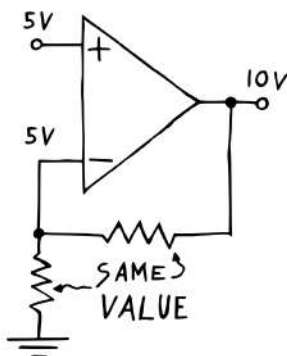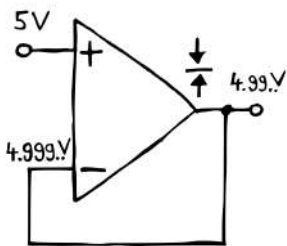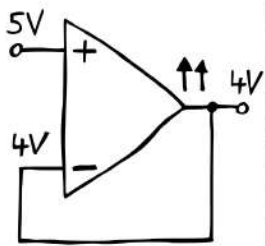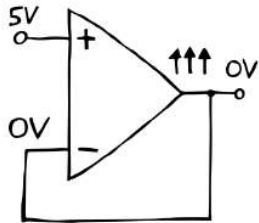
Before the op amp starts processing the voltages at its inputs, the output will be switched off. This means that **output and inverting input sit at 0 V at first**. So then, the op amp will subtract 0 from 5 and multiply the result by its gain. Finally, it will try and increase its output voltage to match the calculation's outcome.

But as it's pushing up that output voltage, the **voltage at the inverting input will be raised simultaneously**. So the difference between the two inputs is shrinking down. Initially, this doesn't matter much because the gain is so large. As the voltage at the inverting input gets closer to 5 V though, the difference will shrink so much that in relation, the gain suddenly isn't so large anymore.

Then, the output will **stabilize at a voltage level that is a tiny bit below 5 V**, so that the difference between the two inputs multiplied by the huge gain gives us exactly that voltage slightly below 5 V. And this process simply loops forever, keeping everything stable through negative feedback. Now if the voltage at the non-inverting input changes, that feedback loop would ensure that the output voltage is always following. So that's why this configuration works as a buffer: the **output is simply following the input**.

How about amplifying a signal though? To do that, we'll have to turn our buffer into a proper non-inverting amplifier. We can do that by replacing the straight connection between inverting input and output with a voltage divider, forcing the op amp to work harder. Here's how that works. Say we feed our non-inverting input a voltage of 5 V. Now, **the output needs to push out 10 V in order to get the voltage at the inverting input up to 5 V**. We call this setup a non-inverting
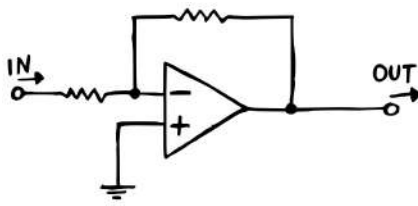
amplifier because the output signal is in phase with the input.

For an inverting buffer/amplifier, the input signal is no longer applied to the non-inverting input. Instead, that input is tied directly to ground. So it'll just sit at 0 V the entire time. The real action, then, is happening at the inverting input. Here, we first send in our waveform through a resistor. Then, the inverting input is connected to the op amp's output through another resistor of the same value.

How does this work? Well, let's assume that we're applying a steady voltage of 5 V on the left. Then, as we already know, the op amp will subtract the inverting input's voltage from the non-inverting input's voltage, leaving us with a result of –5 V. Multiply that by the huge internal gain, and the op amp will try to massively decrease the voltage at its output.

But as it's doing that, an increasingly larger current will flow through both resistors and into the output. Now, as long as the pushing voltage on the left is stronger than the pulling voltage on the right, some potential (e.g. a non-zero voltage) will remain at the inverting input. Once the output reaches about –5 V though, we'll enter a state of balance. Since both resistors are of the same value, the pushing force on the left is fighting the exact same resistance as the pulling force on the right. **So all of the current being pushed through one resistor is instantly being pulled through the other**.

And that means that the voltage at the inverting input will be lowered to about 0 V, allowing our op-amp to settle on the current output voltage level. So while we read 5 V on the left, we'll now read a stable –5 V at the op amp's output. Congrats – we've built an inverting buffer! **If we want to turn it into a proper amplifier, we'll simply have to change the relation between the two resistances**. By doing this, we can either increase (if you increase the right-hand resistor's value) or reduce (if you increase the left-hand resistor's value) the gain to our heart's content.

BIPOLAR JUNCTION TRANSISTORS

Bipolar junction transistors (or BJTs for short) come in two flavors: NPN and PNP. This refers to how the device is built internally and how it'll behave in a circuit. Apart from that, they look pretty much identical: a small black half-cylinder with three legs.
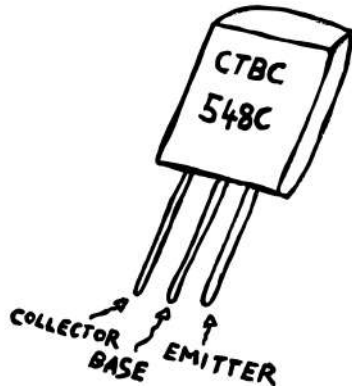
Let's take a look at the more commonly used NPN variant first. Here's how we distinguish between its three legs. **There's a collector, a base and an emitter**.[22] All three serve a specific purpose, and the basic idea is that you control the current flow between collector and emitter by applying a small voltage[23] to the base. The relation is simple: **more base voltage equals more collector current**. Drop it down to 0 V and the transistor will be completely closed off. Sounds simple – but there are four important quirks to this.
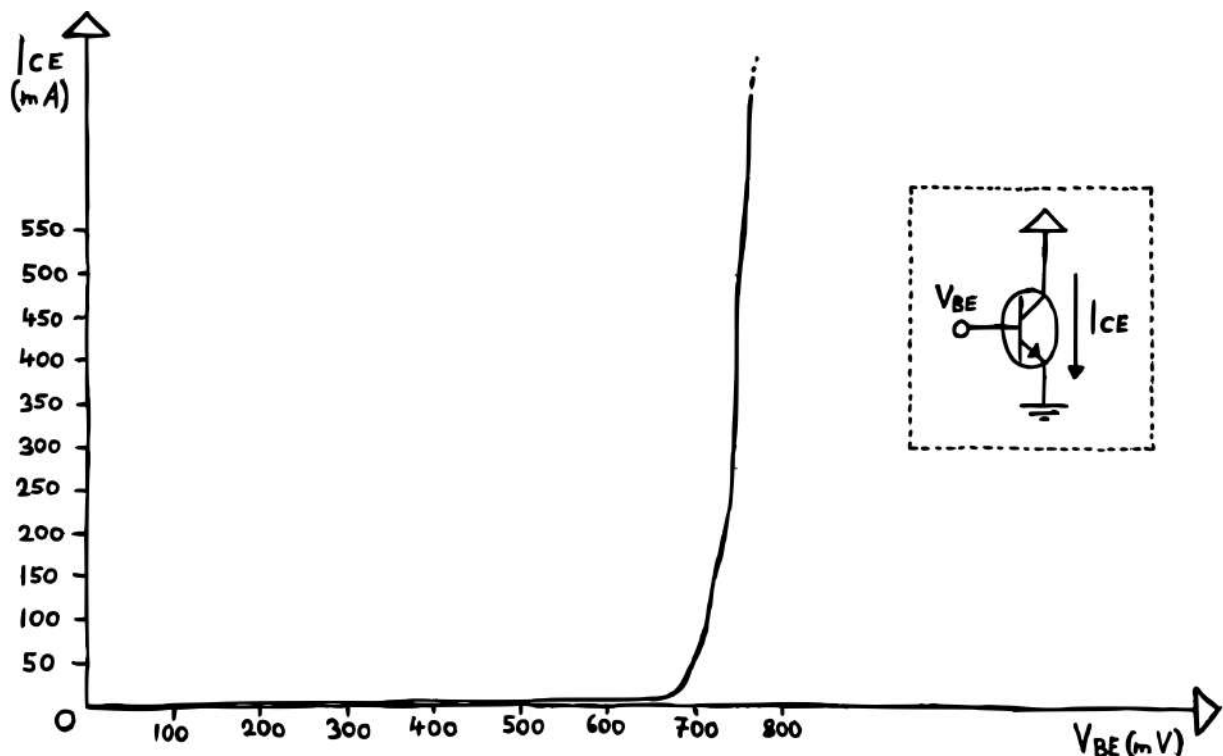
First, the relation between base voltage and collector current is exponential. Second, unlike a resistor, a BJT is not symmetrical – so we can't really reverse the direction of the

---

[23] The voltage is measured between base and emitter. So „a small voltage" effectively means a small voltage **difference** between base and emitter!

collector current. (At least not without some unwanted side effects.) Third, also unlike a resistor, a BJT is not a linear device. Meaning that a change in collector voltage will not affect the collector current. And fourth, the collector current is affected by the transistor's temperature! The more it heats up, the more current will flow.

Now, for the PNP transistor, all of the above applies, too – except for two little details. Unlike with the NPN, **the PNP transistor decreases its collector current when the voltage at its base increases**[24]. So you have to bring the base voltage below the emitter to open the transistor up. Also, that collector current flows out of, not into the collector!
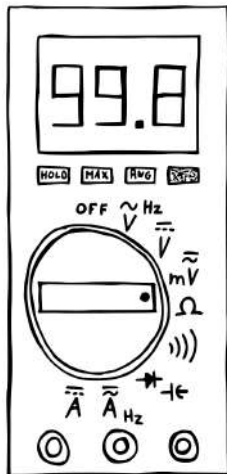

---
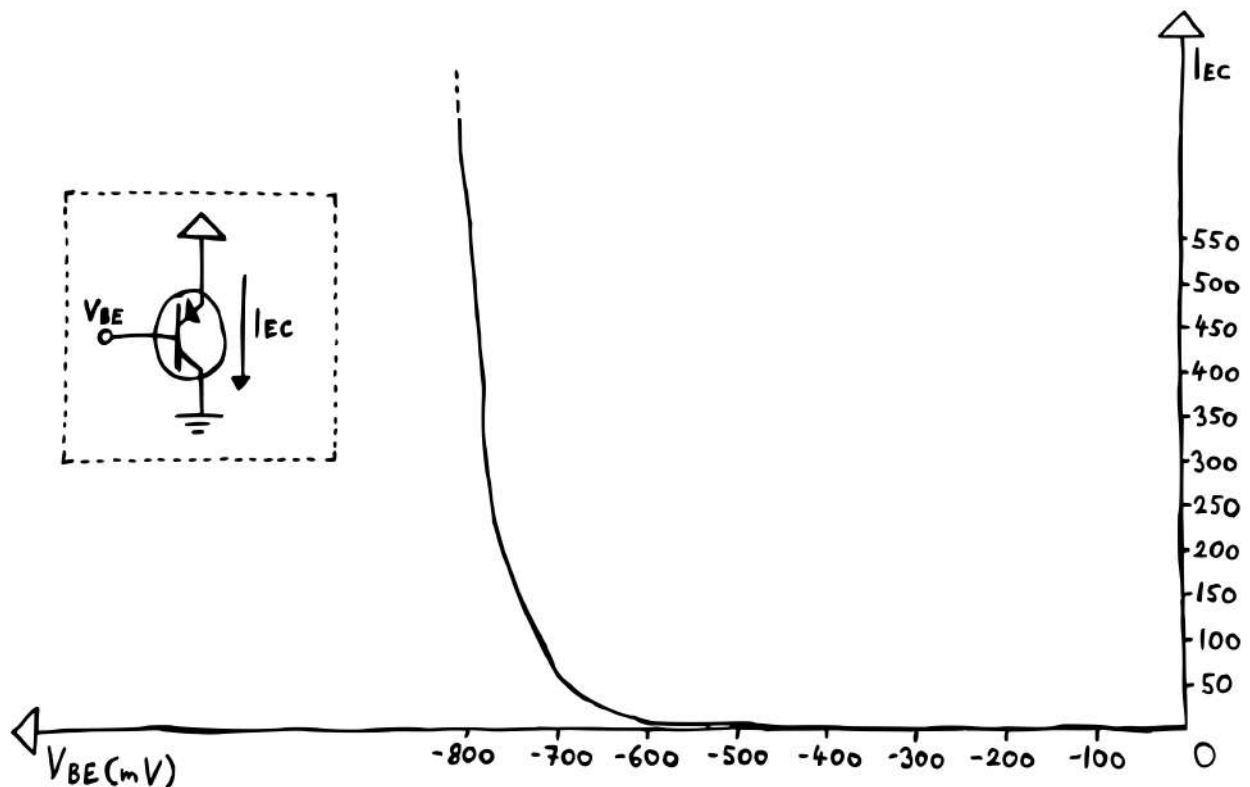
[24] Again, the voltage is measured between base and emitter.

TOOLS APPENDIX

There are two types of tools that will help you tremendously while designing a circuit: multimeters and oscilloscopes. In this appendix, we'll take a quick look at each of these and explore how to use them.

MULTIMETERS

Multimeters come in different shapes and sizes, but the most common type is probably the hand-held, battery powered variant. It can measure a bunch of different things: voltage, current, resistance, continuity. Some have additional capabilities, allowing you to check capacitance, oscillation frequency or the forward voltage drop of a diode.

When shopping for one, you'll probably notice that there are really expensive models boasting about being TRUE RMS multimeters. For our purposes, this is really kind of irrelevant, so don't feel bad about going for a cheap model!

Using a multimeter is actually really straightforward. Simply attach two probes to your device – the one with a black cable traditionally plugs into the middle, while the red one goes into the right connector. Next, find whatever you want to measure and select the corresponding mode setting.
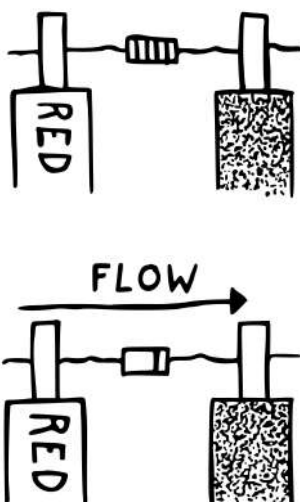
In some cases, it doesn't matter which probe you connect to which component leg or point in your circuit. This is true for testing resistors, non-polarized capacitors (foil/film, ceramic, teflon, glass etc.), continuity[25] or AC voltage.

In others, you'll have to be careful about which probe you connect where. For testing the forward voltage drop of a diode, for example, **the multimeter tries to push a current from the red to the black probe**. Here, you'll have to make sure the diode is oriented correctly, so that it doesn't block that current from flowing. For testing a DC voltage, you want to make sure the black probe is connected to ground, while you use the red one to actually take your measurement.

---

[25] Just a fancy word for saying that two points are electrically connected.

OSCILLOSCOPES
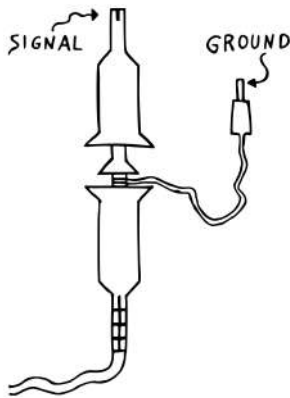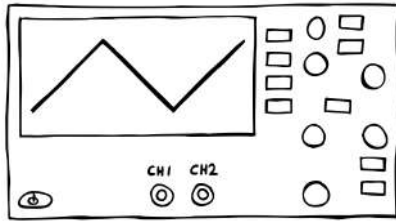
While multimeters are fairly cheap and compact, oscilloscopes are usually somewhat pricey and bulky. **If you're willing to make the investment, they are a huge help with the troubleshooting process, though**. Using one is, again, surprisingly straightforward – if you manage to work your way through the sometimes quite convoluted UI, especially on digital models.

To start using your scope, simply attach a probe to one of the channel inputs. These probes usually have two connectors on the other end: a big one that you operate by pulling the top part back – and a smaller one, which is usually a standard alligator clip. The latter needs to be connected to your circuit's ground rail, while you probe your oscillation with the former. Now what the oscilloscope will do is **monitor the voltage between the two connectors over time and draw it onto the screen as a graph**. Here, the x-axis is showing time, while the y-axis is showing voltage. You can use the device's scaling controls to zoom in on a specific part of your waveform.

Usually, digital oscilloscopes will also tell you a couple useful things about the signal you're currently viewing: minimum/maximum voltage level, oscillation frequency, signal offset. Some even offer a spectrum analyzer, which can be useful to check the frequencies contained in your signal.

BUILD GUIDE

MODULE ASSEMBLY APPENDIX

Because the Drum Sequencer is one of the most advanced projects in our DIY.EDU series and the PCB is densely populated, it requires maximum attention during the assembly. Before we start building, let's take a look at the complete **mki x es.edu Drum Sequencer** schematics (see next page) that were used for the final module's design and PCB fabrication. Most components on the production schematics have denominations (a name – like R1, C1, VT1, VD1, etc.) and values next to them. Denominations help identify each component on the PCB, which is particularly useful during **calibration, modification** or **troubleshooting.**

XS are inputs and outputs of various signals - 3.5mm jack sockets – these are the very same we've already been using on the breadboard for interfacing with other devices. In our designs, we use eurorack standard 3,5mm "Thonkicon" jack sockets (part number WQP-PJ301M-12).
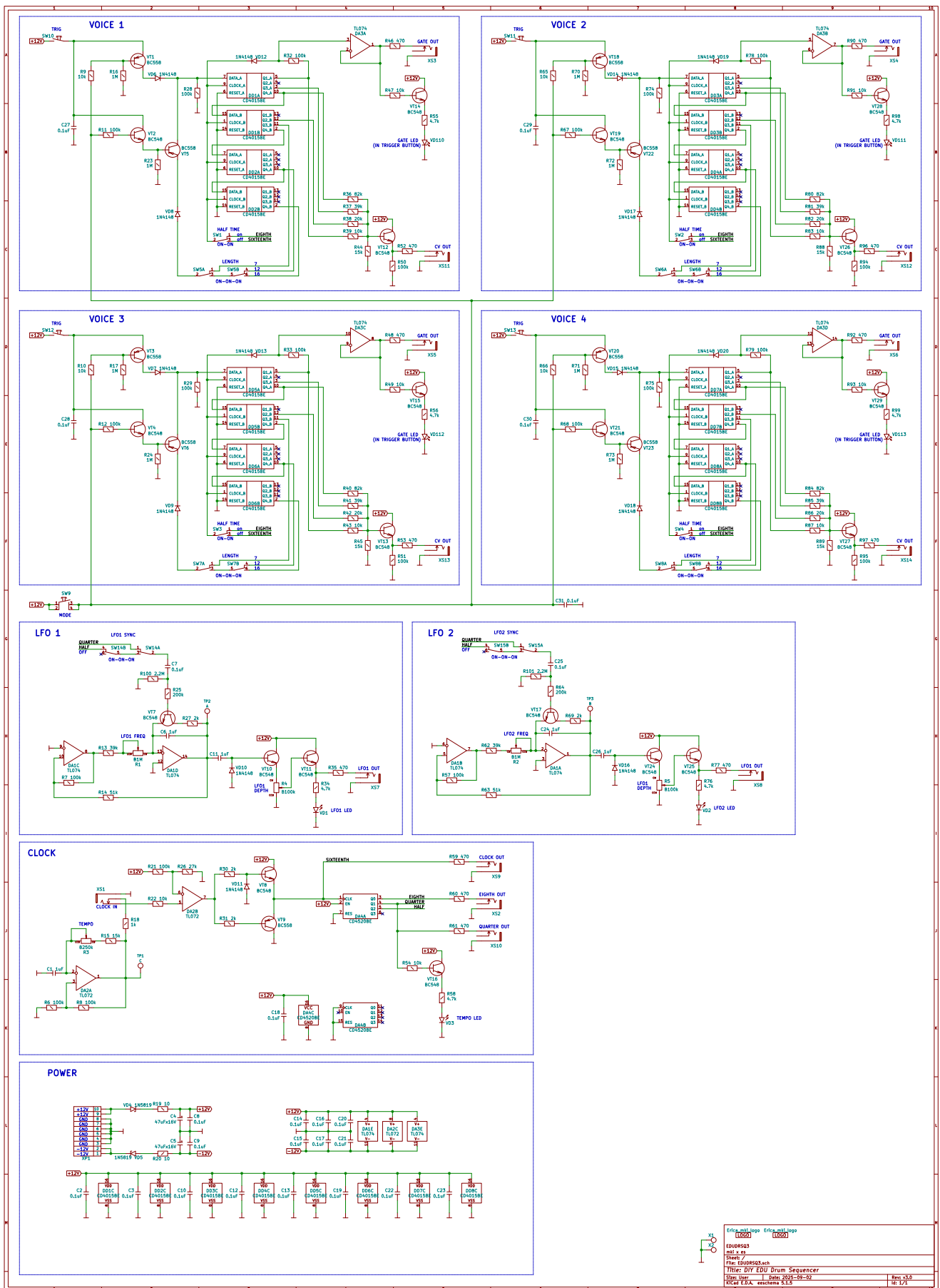
XP1 is a standard eurorack **power connector**. It's a 2x5 male pin header with a key (the black plastic shroud around the pins) to prevent accidental reverse polarity power supply connection. This is necessary because connecting the power incorrectly will permanently damage the module.

VD4 and **VD5** are **schottky diodes** that double-secure the reverse polarity power supply protection. Diodes pass current only in one direction. Because the anode of VD4 is connected to +12 V on our power header, it'll only conduct if the connector is plugged in correctly. If a negative voltage is accidentally applied to the anode of VD4, it closes, and no current passes through. The same goes for VD5, which is connected to -12 V. Because schottky diodes have a low forward voltage drop, they are the most efficient choice for applications like this.

Next, we have two **10 Ohm resistors (R19 and R20)** on the + and – 12 V rails, with **decoupling** (or **bypass-**) capacitors **C8, C9**. These capacitors serve as energy reservoirs that keep the module's internal supply voltages stable in case there are any fluctuations in the power supply of the entire modular system. In combination with R34 and R35, the large **47 microfarad pair (C4 and C5)** compensates for low frequency fluctuations, while C6 and C7 filter out radio frequencies, high frequency spikes from switching power supplies and quick spikes created by other modules. Often another component – a **ferrite bead** – is used instead of a 10 Ohm resistor and there's no clear consensus among electronic designers which works best, but generally for analogue modules that work mostly in the audio frequency range (as opposed to digital ones that use microcontrollers running at 8 MHz frequencies and above), resistors are considered to be superior.

Another advantage of 10 Ohm resistors is that they will act like slow "fuses" in case there's an accidental short circuit somewhere on the PCB, or an integrated circuit (IC) is inserted backwards into a DIP socket. The resistor will get hot, begin smoking and finally break the connection. Even though they aren't really fuses, just having them there as fuse substitutes is pretty useful - **you'd rather lose a cent on a destroyed resistor than a few euros on destroyed ICs.**

Capacitors **C2, C3, C10, C12, C14 - C17, C19 - C21, C23** are additional decoupling capacitors. If you inspect the PCB, you'll see that these are placed as close to the power supply pins of the ICs as possible. For well-designed, larger PCBs you will find decoupling capacitors next to each IC. Like the others, their job is to simply compensate for any unwanted noise in the supply rails. If the input voltage drops, then these capacitors will be able to bridge the gap to keep the voltage at the IC stable. And vice-versa - if the voltage increases, then they'll be able to absorb the excess energy trying to flow through to the IC, which again keeps the voltage stable. Typically, 0.1 uF capacitors are used for this purpose.

VOICE 1

VOICE 2

VOICE 3

VOICE 4

LFO 1

LFO 2

CLOCK

POWER

mki x es.edu

Title: DIY EDU Drum Sequencer
File: EDUDRSQ3.sch
Size: User   Date: 2025-09-02
KiCad E.D.A. eeschema 5.1.5
Rev: v3.0
Id: 1/1

Before you start soldering, we highly recommend printing out the part placement diagrams with designators and values and follow step-by step instructions below. As mentioned above, the Drum Sequencer is one of the most complex projects in our DIY.EDU line, so, this will help you to avoid mistakes in the build process. **NB!** During assembly, please, follow **the silkscreen on the PCB and part placement diagram!** Photos of the assembly process below are of V2 PCB, and there are minor differences compared to the final V3 version.

EDUDRSQ3
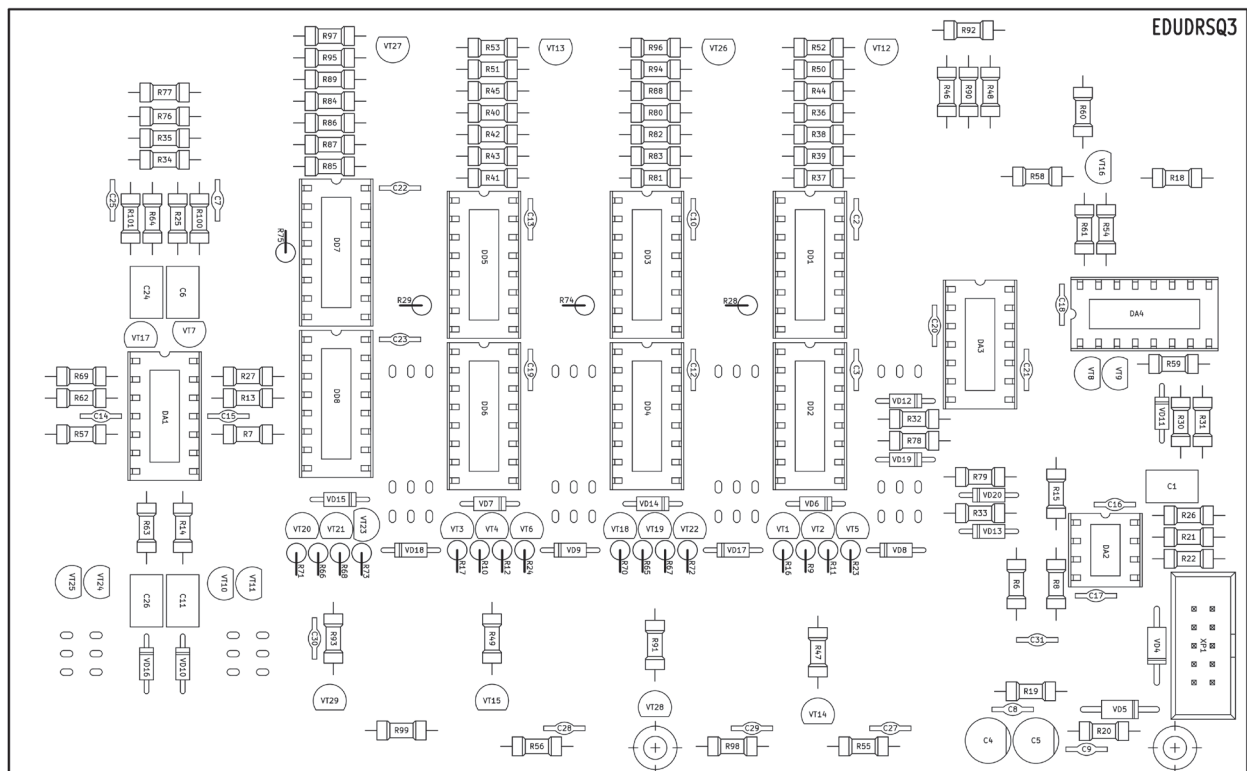
Place the Drum Sequencer PCB in a PCB holder for soldering or simply on top of some spacers (I use two empty solder wire coils here).


I usually start populating PCBs with lower, horizontally placed components, resistors specifically. As there are lot of **resistors of various values**, before soldering, we recommend you to **sort them by values** in order to avoid mistakes during build process. Bend the resistor leads and insert them in the relevant places according to the part placement diagram above. Flip the PCB over and solder all components. Then, use pliers to cut off the excess leads.
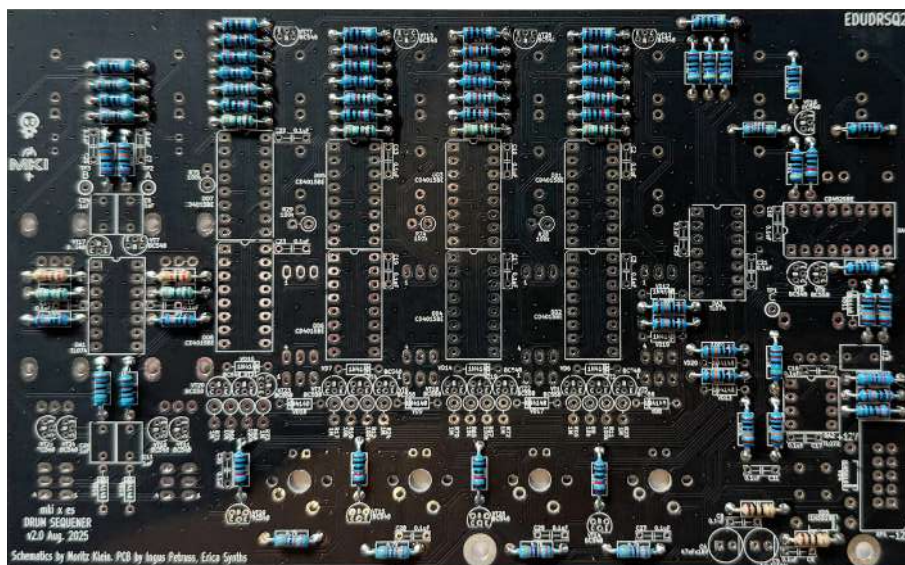
Now, proceed with **switching diodes** and the **power protection diodes.** Remember – **when inserting the diodes, orientation is critical!** A thick white stripe on the PCB indicates the cathode of a diode – match it with the stripe on the component. Solder all diodes.
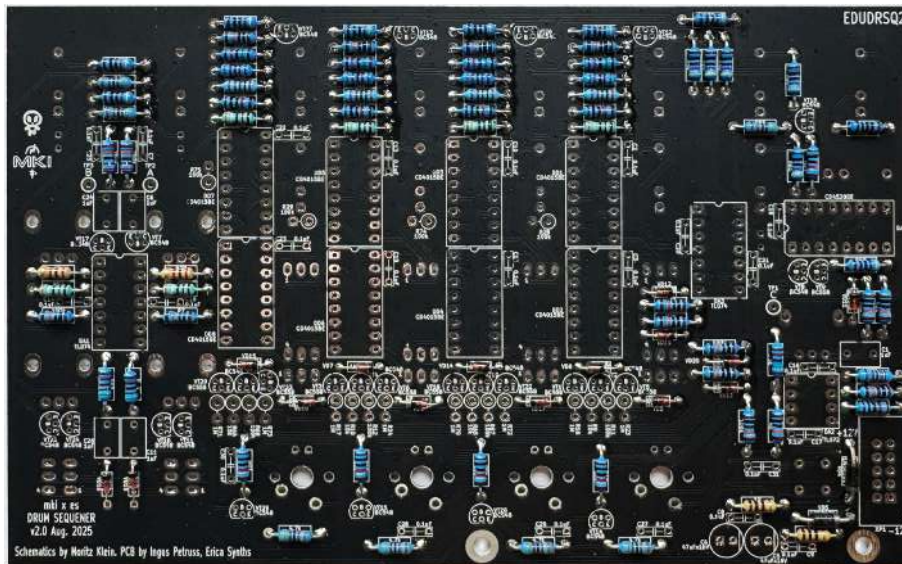
Next, insert the first **DIP socket**, hold it in place and solder one of the pins. Continue with other DIP sockets. Make sure **the DIP sockets are oriented correctly** – the notch on the socket should match the notch on the PCB's silkscreen. Now, turn the PCB around and solder all remaining pins of the DIP sockets. Compare your socket placement with the one on the picture below.
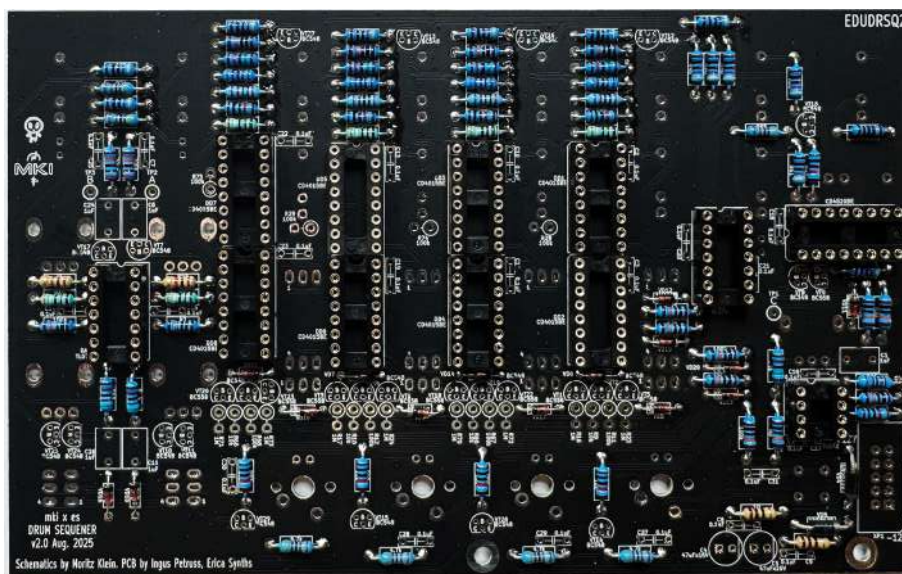
Then proceed with the ceramic capacitors. Start with soldering 0,1uF capacitors - place the PCB in your PCB holder or on spacers, insert the capacitors and solder them like you did with the resistors & diodes before. Then proceed with other ceramic capacitors. When completed, your PCB should look like this:
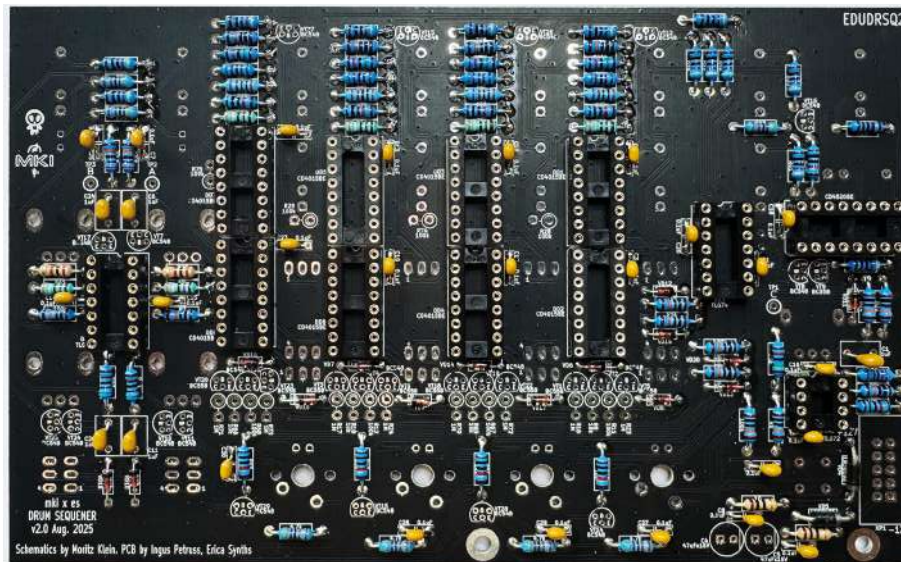
Next, insert and solder transistors. There are **PNP and NPN** transistors in the kit, therefore before soldering them, I highly recommend to **sort them.** Make sure you install them in correct places and pay attention on the orientation of the transistors – notch on the silk-screen has to match the flat part of the transistor.

In order to save space on the PCB, some of our projects, including the Drum Sequencer, have **vertically placed resistors**. The next step is to place & solder those. Bend a resistor's legs so that its body is aligned with both legs and insert it in its designated spot. Then solder the longer lead from the top side of the PCB to secure it in place, turn the PCB around and solder the other lead from the bottom. You can insert several resistors at once. Once done with soldering, use pliers to cut off excess leads.
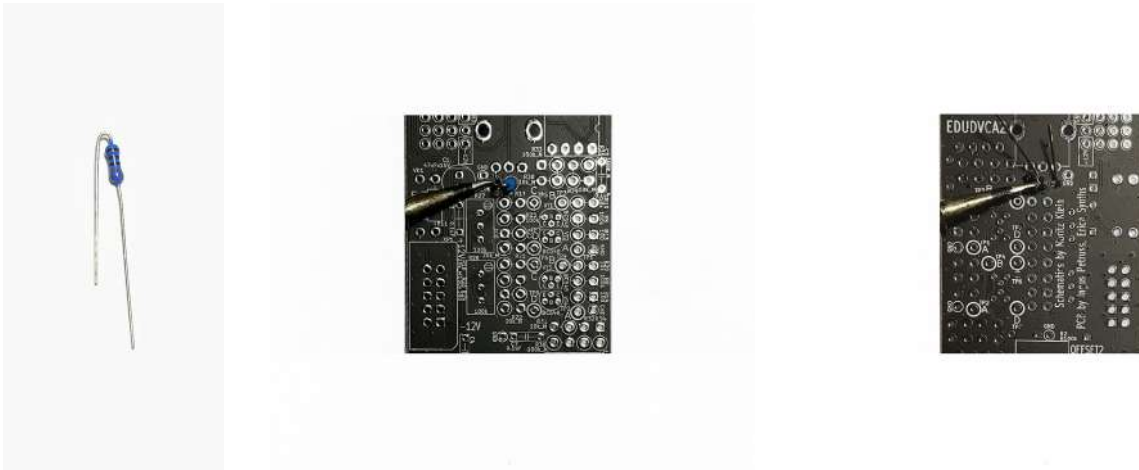
Once you are done with soldering all resistors, your PCB should look like this:

Also, **insert & solder the electrolytic capacitors.** Electrolytic capacitors are bipolar, and you need to mind their orientation. The positive lead of each electrolytic capacitor is longer, and there is a minus stripe on the side of the capacitor's body to indicate the negative lead. On our PCBs, the positive pad for the capacitor has a square shape, and the negative lead should go into the pad next to the notch on the silkscreen.


Then populate **film capacitors** and **2x5 PSU socket.** Make sure the orientation of the socket is as shown in the picture below – the arrow pointing to the first pin is aligned with a notch on the silkscreen. The key on the socket will be facing outwards the PCB. Now your PCB should look like this:

Now, turn the PCB around and inspect your solder joints. **Make sure all components are soldered properly and there are no cold solder joints or accidental shorts.** Clean the PCB to remove extra flux, if necessary. Next, use M3 screws supplied with a kit to fix **10mm spacers** in place and insert and solder **jack sockets.**
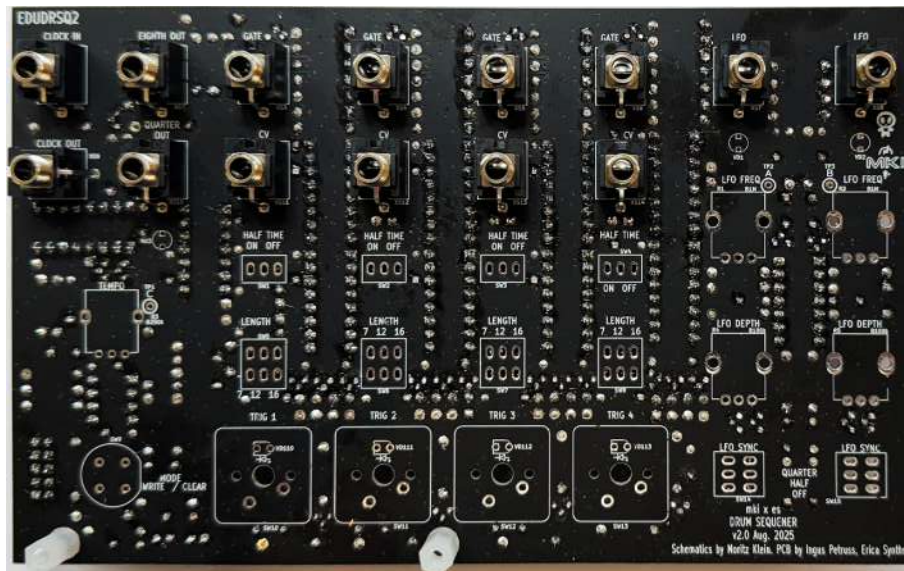
Toggle switches on the module require special attention. All components on the front panel are 10mm tall, except switches, therefore before soldering they must be tightened against front panel. Insert all switches, but **do not solder them, yet!**
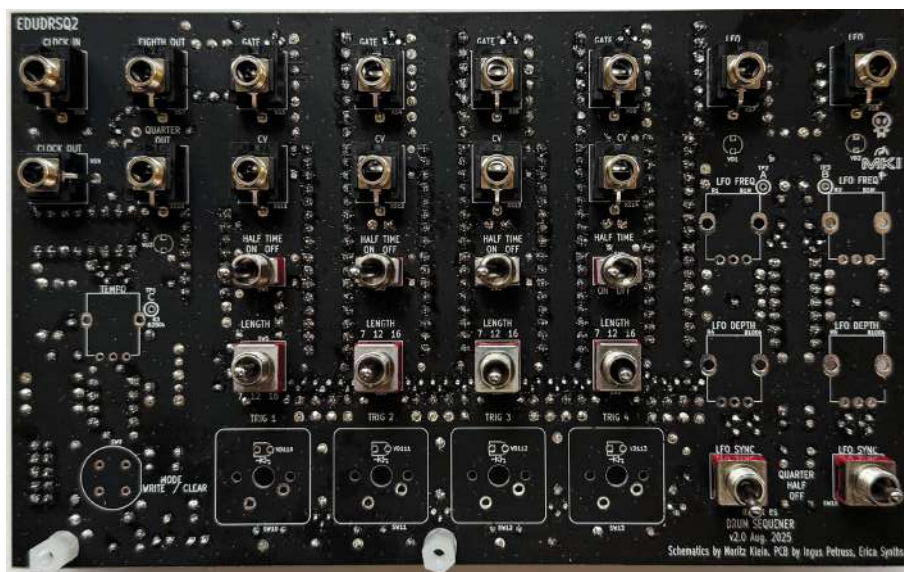
Install the front panel and fix it with few hexnuts on the jack sockets and two M3 screws for spacers. Then **tighten hexnuts** on switches so that switches are firmly **pushed against the front panel**; you may want to adjust alignment of the single pole switches as they tend to be bit loose in their places.

Once happy, solder the switches! Remove the panel, and your PCB has to look like this:

Now proceed with cherry keys. Insert a cherry key a relevant place and solder **one of its leads.** Then install a small **LED** (pay close attention to **orientation of the LED** – there's a silkscreen below the cherry key indicating orientation of the LED, and also pay attention to the direction of the rounded side of the LED) and solder it!
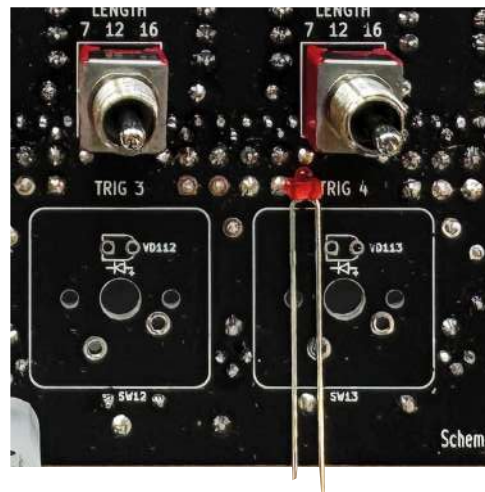
Now, adjust position of the cherry key, so it aligns with the silkscreen on the PCB and solder the second lead of the cherry key. Then proceed with other three cherry keys in the same manner. Also, populate and solder a pushbutton; make sure the notch on the pushbutton is aligned with the silkscreen. After completing this step, your PCB should look like this.

Insert **potentiometers**, but **don't solder** them yet! Also, install the **3mm LEDs** (make sure, the notch on the silkscreen is aligned with the one on the LED), but do not solder them!
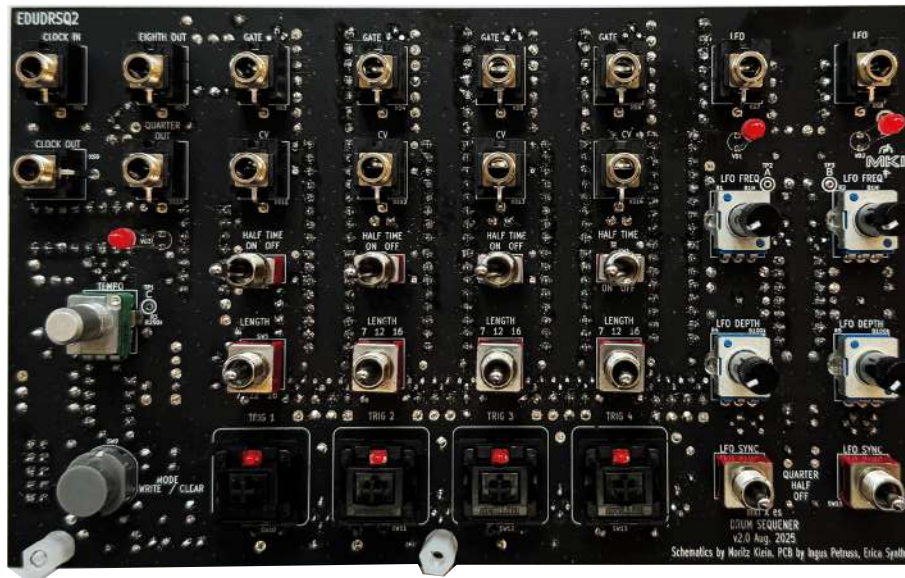
Fit the front panel, screw the nuts on the green potentiometer, jack sockets and switches and make sure that the potentiometer shafts are aligned with the holes in the panel – and that they're able to rotate freely. Now, go ahead and **solder the potentiometers and the LEDs** after fitting them in relevant holes on the front panel.

Now, push on the keycaps on the cherry keys and install the knob on the green potentiometer.

Now, **insert the ICs into their respective DIP sockets.** Mind the orientation of the ICs – match the notch on each IC with the one on its socket.

Congratulations! **You have completed the assembly of the mki x es.edu Drum Sequencer module!** Connect it to your eurorack power supply and switch it on. If there's no "magic smoke", it's a good sign that your build was successful. The module does not need calibration and, if your assembly is correct, it has to work straight away. Connect trigger outputs to some percussion modules and make a sequence!. **Enjoy!**
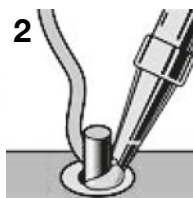
SOLDERING APPENDIX

If you've never soldered before – or if your skills have become rusty – it's probably wise to check out some **THT** (through-hole technology) **soldering tutorials on YouTube**. The main thing you have to remember while soldering is that melted solder will flow towards higher temperature areas. So you need to make sure you apply equal heat to the component you are soldering and the solder pad on the PCB. The pad will typically absorb more heat (especially ground-connected pads which have more thermal mass), so keep your soldering iron closer to the pad on the PCB. It's critically important to dial in the right temperature on your soldering station. I found that about 320 °C is the optimal temperature for most of parts, while for larger elements like potentiometers and sockets, you may want to increase that temperature to **370 °C.**
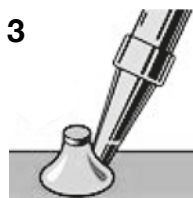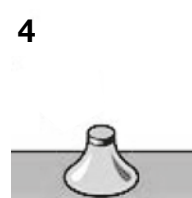
Here's the recommended soldering sequence:
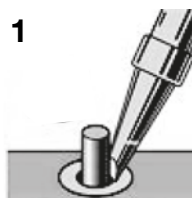
1	2	3	4
Heat part and pad 2 - 3 sec	Add solder	Continue heating 1 -2 sec.	Let cool

After you have completed soldering, inspect the solder joint:

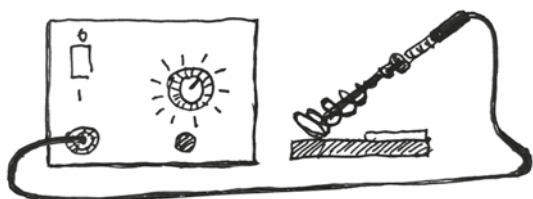| Perfect	Too much solder	Not enough solder	Cold joint	Too much heat	Short

DIY electronics is a great (and quite addictive) hobby, therefore we highly recommend you invest in good tools. In order to really enjoy soldering, you'll need:


A decent soldering station. Top-of-the-line soldering stations (brands like Weller) will cost 200€ and above, but cheaper alternatives around 50€ are often good enough. Make sure your soldering station of choice comes with multiple differently-sized soldering iron tips. The most useful ones for DIY electronics are flat, 2mm wide tips.
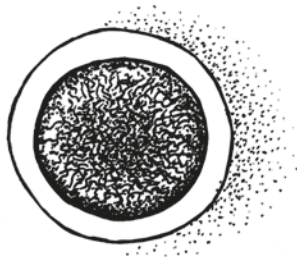
When heated up, the tips of soldering irons tend to oxidize. As a result, solder won't stick to them, so you'll need to clean your tip frequently. Most soldering stations come with a **damp sponge for cleaning the iron tips** – but there are also professional solder tip cleaners with **golden curls** (not really gold, so not as expensive as it sounds). These work much better because they do not cool down the iron.

**Solder wire with flux**. I find 0,7mm solder wire works best for DIY projects.
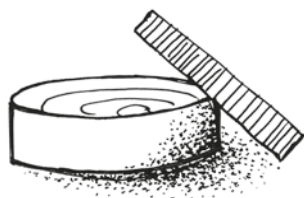
Some **soldering flux** paste or pen will be useful as well.

Cutting pliers. Use them to cut off excess component leads after soldering.

A solder suction pump. No matter how refined your soldering skills are, you will make mistakes. So when you'll inevitably need to de-solder components, you will also need to remove any remaining solder from the solder pads in order to insert new components.
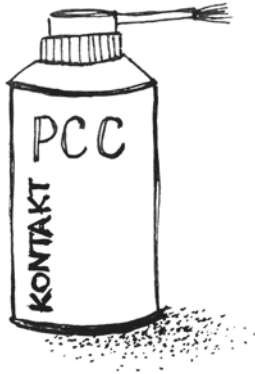
Once you have finished soldering your PCB, it's recommended to remove excess flux from the solder joints. **A PCB cleaner** is the best way to go.

All of these tools can be found on major electronic components retailer websites, like Mouser, Farnell and at your local electronics shops. As you work your way towards more and more advanced projects, you'll need to expand your skillset and your tool belt – but the gratification will be much greater.

"I just love the hypnosis of a single bass drum."

– Jon Hopkins