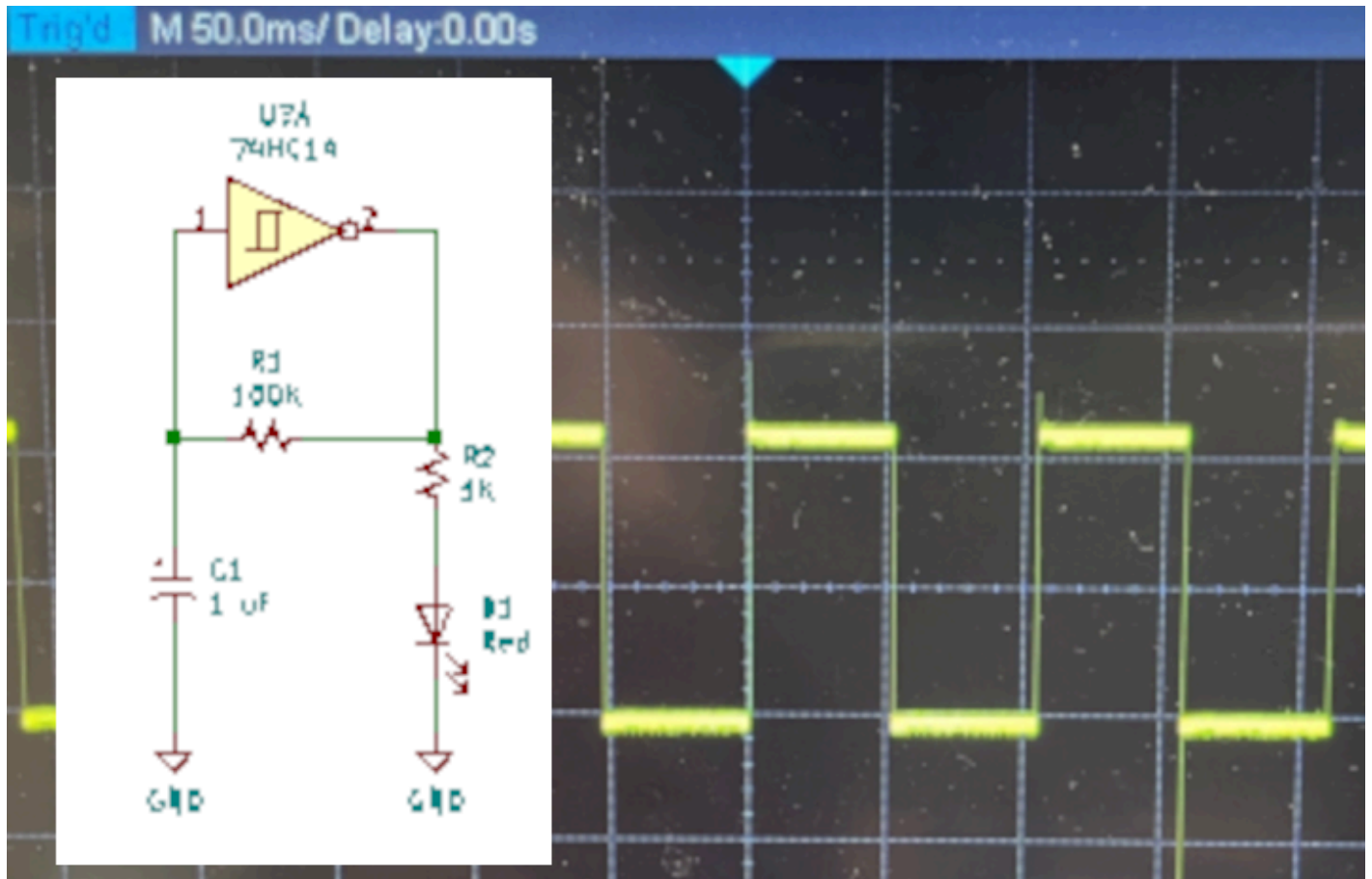


Super Simple Square Wave Generator

By [bdk6](#) in [CircuitsElectronics](#)



Introduction: Super Simple Square Wave Generator

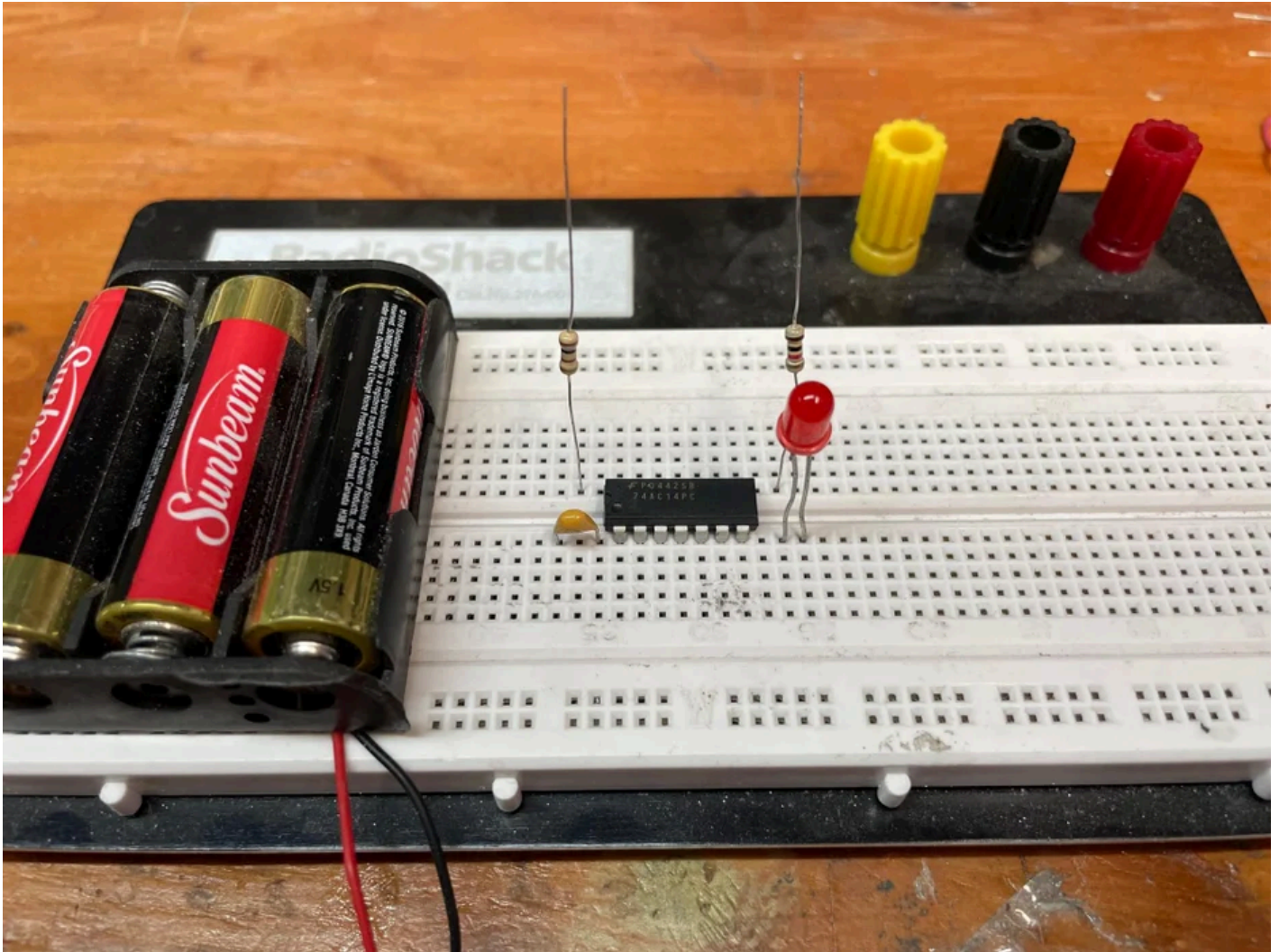


A lot of circuits need some type of clock signal. For hobbyists, perhaps the most chosen option is the 555 timer. The 555 is a great chip, but it isn't always the best choice. For example, we often need a signal that is exactly 50% duty cycle, a "square" wave. (A lot of people refer to non 50% digital waveforms as square waves, but rectangular is a more accurate name.) In many cases, and especially for square waves, the 555 is not ideal. It tends to use a lot of power in its standard form, it generally needs at least two resistors and one or two capacitors, it usually doesn't swing all the way positive, and takes even more components to "trick" it into generating a 50% waveform. In addition, it doesn't work well below around 4.5 volts.

In this Instructable, we look at how to make a very simple oscillator that naturally provides a 50% duty cycle with rail to rail outputs. It works from 2 to 6 volts. It only takes one resistor and one capacitor in addition to the 74hc14 hex Schmitt trigger chip. It can be persuaded to generate other rectangular waveforms, but that isn't its strong point. In addition, the 74hc14 has six inverters and only one is required; the others can be used for more oscillators or for other purposes. It can easily generate frequencies up to about 500 KHz, often more.

Since many digital circuits will need inverters or NAND gates (see below) for other purposes, you may end up with an oscillator almost for free. A Schmitt trigger gate can almost always be used in place of a standard gate of the same type, but not the other way around.

Supplies



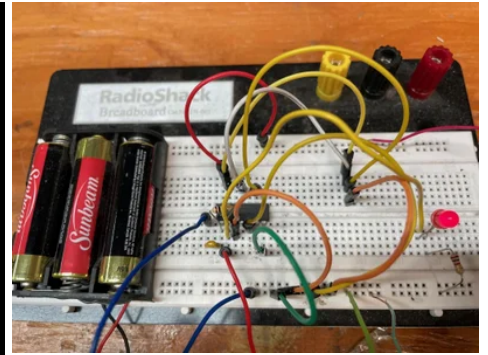
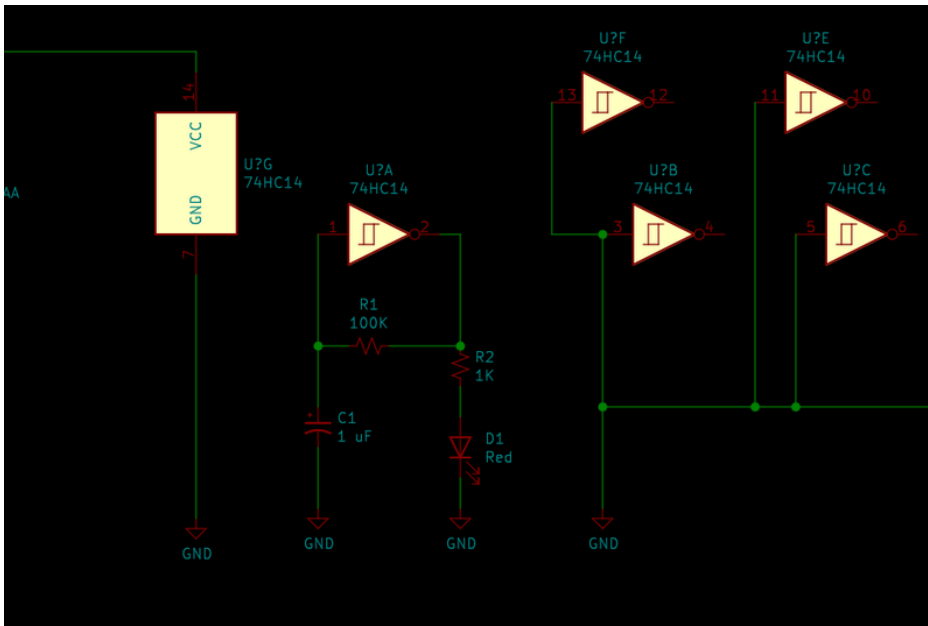
The parts needed for this circuit are very few:

- 74HC14 Hex schmitt trigger. **Don't substitute!** This MUST be a 74HC14. No type of TTL (7414, 74LS14, 74ALS14, etc) will work properly. Neither will any type of 74xx04 or even 74HCT14.
- Resistor at least 100 Ohms. For your first build I recommend a 100K. Any wattage from 1/8 watt up will be fine.
- Capacitor at least 0.001 uF in general, but 1 uF for the example circuit here.

Those are all the parts required to build the oscillator. To build one and see it work, you will need some additional parts:

- Power supply. For this purpose it should be at least three volts but not more than six. Three or four AA batteries will be fine. I used a battery holder with 3 AA batteries.
- Red LED. Red LEDs usually don't need more than about 2 volts to light up, where some take as much as 5 volts.
- 1K resistor. This can vary from about 500 to 5000 Ohms, but try to get close to 1K for the initial test.
- Solderless breadboard or prototyping pc board. Something to build the circuit on.

Step 1: Build It



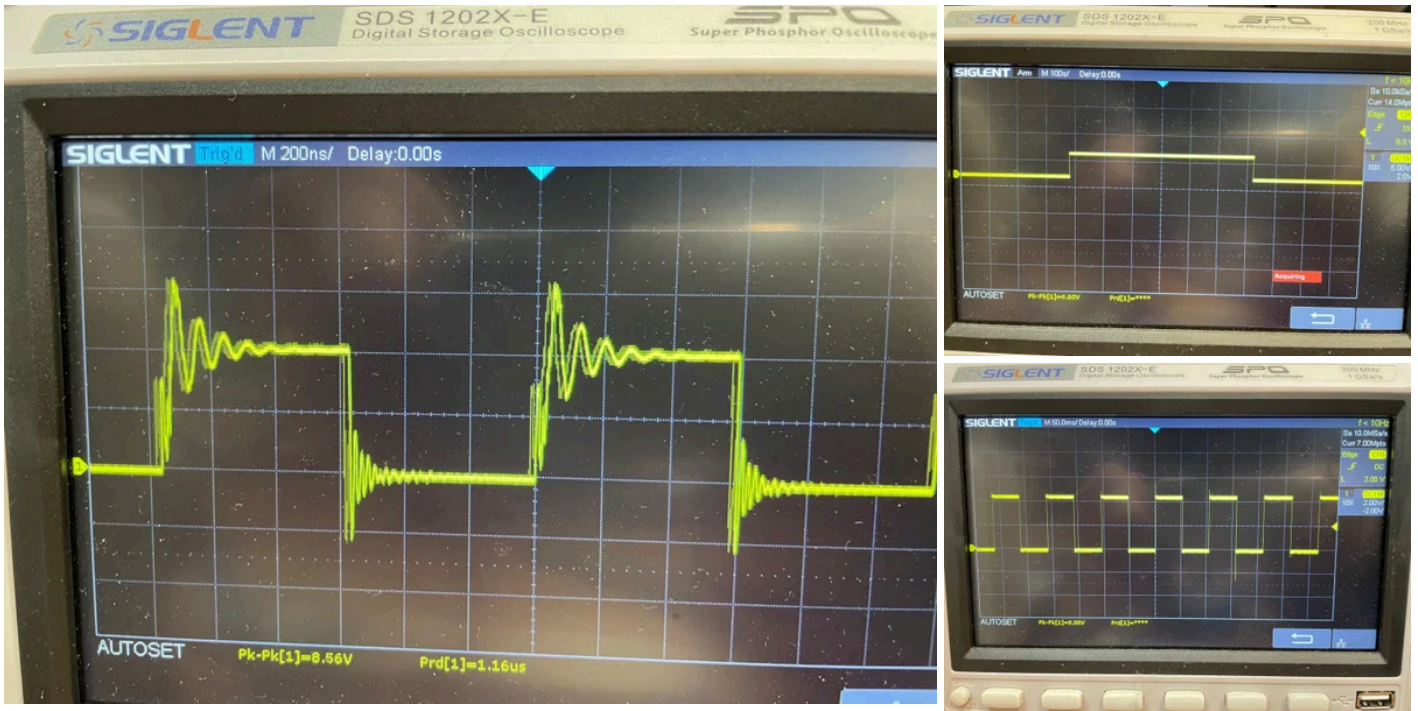
The schematic diagram and a breadboarded version I built are shown in the photos. In the schematic, the only parts that are actually parts of the oscillator are the 74HC14 chip, R1, and C1. R2 and the LED are just so we can see it working.

Build the circuit as in the schematic. I've shown it here on a solderless breadboard, which I recommend for experimenting. The capacitor connects from pin 1 to ground. R1 connects from pin 2 to pin 1. The positive power (Vcc) connects to pin 14 and ground is connected to pin 7. That's all there is to the oscillator! Of course we want to see it work, so connect R2 to pin 2 and to the anode of the LED. The cathode of the LED gets connected to ground. Remember to connect all the unused inputs, pins 3, 5, 9, 11, and 13, to ground. CMOS chips don't like having floating inputs. In a real circuit those other five inverters can be used for whatever purpose you want. CMOS chips are also sensitive to static discharge, so handle them carefully.

Now that it is built, the LED should be flashing. If you used the 100K resistor and 1 μ F capacitor for R1 and C1, it should be flashing about ten times per second: pretty quick but slow enough to see the flashes.

If it doesn't seem to be working, make sure you have everything wired correctly. The LED polarity would probably be the first I would check. Also make sure you used the values I recommended. Other values could make the frequency too high to see the flashes or a too-large value for R2 could make the LED too dim to see. Another color of LED could require more voltage than you have available. It's a simple circuit and shouldn't be too hard to get working.

Step 2: Analyze It



So how does it work? Let's analyze the circuit.

The first and most important aspect is the characteristics of the 74HC14. It is a Schmitt trigger. Unlike most logic gates, it only switches output states at certain, well-defined voltage levels. Those voltage levels are typically something like 30% and 70% of the supply voltage. Why two voltages? One is for a low to high going input (70%) and the other is for a high to low going input (30%.) A "normal" gate will switch both states at the same voltage, typically 50% of the supply voltage. That gap between the switching points is called "hysteresis" and is the key to this circuit working.

Let's look at an example. Assume we have a 5 volt supply. That makes the switching points 1.5V (30%) and 3.5V (70%.) If the input is at 0 volts but begins to rise, the output will be high (remember this is an inverter.) Once the voltage reaches 3.5V, the output will change to low. In our oscillator circuit, the voltage on the input will be the same voltage as the capacitor. When the capacitor voltage is low, the output will be high. But the high output charges the capacitor through R1. That is how the input voltage climbs from 0 to 3.5V. But once the input reaches that 3.5V, the output will suddenly switch to low. Now, the capacitor will discharge through R1 and the voltage on the cap, and the input, will begin to drop. When that voltage drops to the low-going threshold, 1.5V, the state will change again to high output. And the cycle will repeat. But, after the first cycle, the voltage on the cap will only vary between the two thresholds, 1.5V and 3.5V (or whatever the 30% and 70% values are for the applied supply voltage.)

That gap between thresholds, or hysteresis, is what makes the circuit work. It gives the R1/C1 combination time to charge and discharge between switches. Normal logic gates don't have that hysteresis so they won't work properly. There are other chips that have that hysteresis. They are always called "Schmitt triggers." The 74HC132 is one that is useful. To build an oscillator, the gate must be a Schmitt trigger and also inverting. The 74HC132 is a NAND gate, so it inverts and is a Schmitt trigger.

Another important characteristic is that the output should be symmetric when in the high and low states. What I mean by that is that the internal resistance, or current capability, should be the same or very close. In HCT type chips, the threshold levels are different. They will work, but not quite the same. In any type of TTL chips, the threshold levels are different and the outputs are not symmetric. They will work, too, but they will have very different characteristics.

You may know that a resistor-capacitor circuit charges and discharges with an inverse exponential curve. We won't get into that, the math hurts my head. What's important to us is that with symmetric outputs, the charge and discharge times will be nearly identical. That means the high time and the low time will be nearly

identical, as well. And, by coincidence, that time just happens to turn out to be very close to $\frac{1}{2} * R1 * C1$! The total period time is twice that, or $R1 * C1$. Which means that our frequency will be very near to $1/(R1 * C1)$! That's pretty convenient.

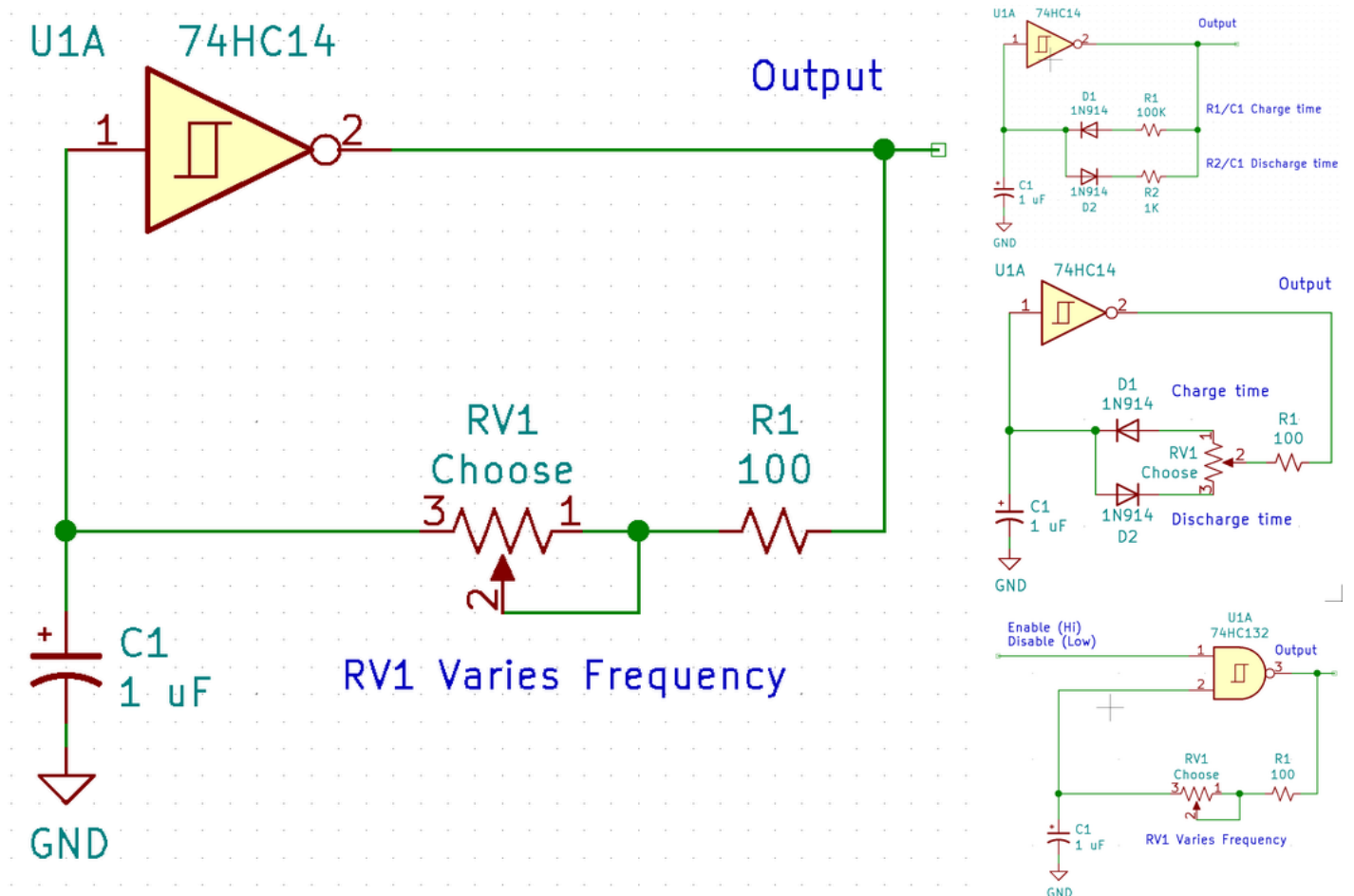
In our example circuit, we used a 100,000 Ohm resistor and a 1 uF (0.000001) capacitor. In that case, the $R1 * C1$ combination is 0.1. Since $1/0.1 = 10$ our LED flashes at approximately 10 Hz (10 times per second.) With extreme values of $R1$ and $C1$, that won't quite hold up. As my professors said, characterizing that is left as an exercise for the reader.

What values can we use for $R1$ and $C1$? Well, now, that depends. First, don't use a resistor of less than 100 Ohms for $R1$. A lower limit of about 220 Ohms is better, but 100 should be fine. The reason for that is below about 100 Ohms the charging current out of (or into) the chip into the capacitor is likely to go beyond the rated current. With good quality ceramic capacitors I have used resistors up to 10 MOhm. You can perhaps go higher, but I haven't tried. It depends a lot on the capacitor. Most capacitors are "leaky," meaning that some current going into them "leaks through" as though there is an unseen resistor across them. Cheap electrolytic capacitors are usually the worst. The leakage current takes away from the charging current. In an extreme case, there may not be any charging current left. So large resistors with large electrolytic caps probably won't work well, but it depends largely on quality of the cap. They will definitely not adhere closely to the $f = 1/(rc)$ rule above. Better quality (more expensive) capacitors will do better. Other types of capacitors are usually less leaky. Ceramic are good. Polystyrene and tantalum should do well. Try different ones and see what you get. On the low end, I have used 0.001 uF ceramic caps and small resistors to get around 500,000 Hertz. A range of 1 Hz to 500 KHz is easily achievable. Two of the pictures above are scope traces of two extremes. One is with a 1000 Ohm resistor and .001 uF cap. The scope is at 200 nano seconds per division. The frequency is about 833 KHz. The other is with a 100 uF capacitor and 10 MOhm resistor. The period is about 1200 seconds (20 Minutes!) for a frequency of about 833 micro Hertz! The final scope trace is the 10 Hz (100 K and 1 uF) example circuit we built above.

One note: the frequency is likely to be voltage dependent as well as temperature dependent. This isn't a high precision oscillator. But in my experiments it has been pretty stable. Different chips will have different characteristics which will affect the frequency. Especially if the chips come from different manufacturers. More exercises for the reader.

There are various other factors that will affect circuit operation, but, just like when book authors don't want to bother, I will leave those as an exercise for the reader, as well.

Step 3: Going Further



The basic circuit is quite simple. But what else can it do?

First, it is easy to make a variable frequency square wave generator. Use a 100 Ohm resistor for R1 and add a potentiometer in series with it. A 1 MOhm pot will give a frequency range of about 10,000 to 1. See the first schematic.

Or you can change the duty cycle by replacing R1 with two resistors and two diodes, similar to how one gets 50% on a 555. One resistor for the charge time (high output) and the other for the discharge (low output.) The diodes select which resistor gets used. Note that the diodes will affect the charge and discharge times to the $1/(R \cdot C)$ frequency calculation won't be accurate any more. Although it should still be fairly close in most cases. You can see this in the second schematic.

Or how about a fixed-frequency, variable duty cycle? Leave a 100 Ohm resistor in for R1, but connect it to the wiper of a potentiometer. The two ends of the pot connect to diodes as in the fixed duty cycle above. So now you have different high and low times with each variable, but the sum is the same. This is shown in the third schematic. Very useful for PWM control of motors and light dimmers. It should be easy to get very close to 0% and 100% pulse widths.

If you use the 74HC132 NAND gate instead of the 74HC14, you can have a gated oscillator. Use one NAND gate as before, but one of the inputs is connected the R1/C1 junction and the other is used to enable (high) or disable (low) the oscillations. That is shown with variable frequency in the fourth schematic.

There are more things you can do with this circuit. But again, I will leave those things as exercises for the reader. I shouldn't get to have ALL the fun!

Enjoy your experimenting.