**Lab-5: RLC Filter Design and beyond.**

In this lab, we implement an RLC bandpass filter to isolate an specific frequency component from a noisy signal. Objectives are:

1. Learn the basics of summing opamp amplifiers.
2. Design an RLC bandpass filter.
3. Enhance the filtered signal using amplifier and clamper stages.

**Design and Experiment:**

Since this circuit implementation is completely modular, we will integrate most of the design questions with the lab procedure. Note that the final circuit will have a lot of components, so be mindful of your circuit arrangement on the bread-board.

**Step.1: Creating an input signal with noise.**

Our first objective is to create an input signal with three different frequency components, a low-frequency sinusoidal component, a mid-frequency sinusoidal component, and a high-frequency square wave noise signal. Remember that our target during this lab is to isolate the mid-frequency component from the other two using an RLC pass filter. The high frequency and low-frequency components play the role of a noisy environment. In practical applications, many source noises will interrupt a signal, and it becomes necessary to distinguish the signal from a noisy environment.

First, let's create these signals. The two sinusoidal signals will be made using our knowledge of low-pass filters from lab-4. Two SPWM signals are created on Pin-6 and Pin-5 (use specifically these pins) and are passed through RC filters to create sinusoidal waves.

* Implement the following circuit and scope/plot the outputs of the three different signal components.

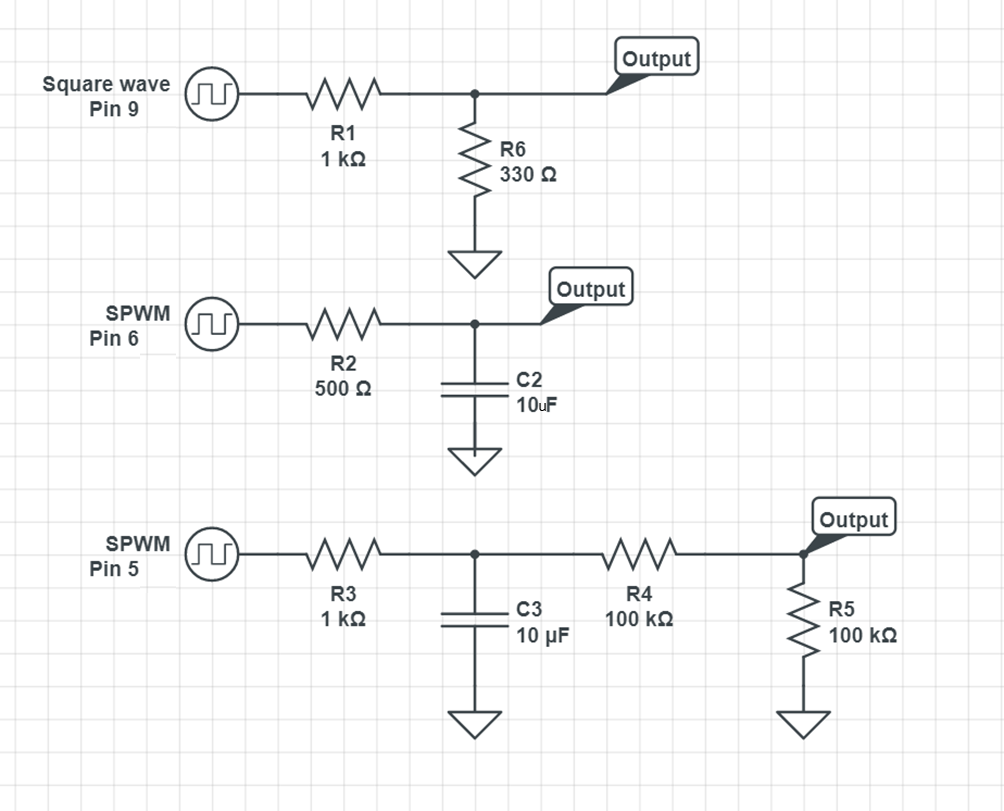


Figure.1. RC-low pass stage

For pin-5, we will use an SPWM signal with a frequency of about 16 Hz, and for pin-6, we will use an SPWM signal with a frequency of about 2 Hz. The square wave on Pin-9 is set at 1000 Hz. Following code might help you with implementation:

#include <PWM.h> //PWM library for controlling freq. of PWM signal

double angle = 0;

double anglee = 0;

double increment = 0.02;

double incrementt = 0.2;

void setup() {

// put your setup code here, to run once:

InitTimersSafe();

}

void loop() {

double sineValue = sin(angle);

sineValue \*= 255;

int plot = map(sineValue, -255, +255, 0, 255);

analogWrite(5,plot);

angle += increment;

double sineValuee = sin(anglee);

sineValuee \*= 255;

int plott = map(sineValuee, -255, +255, 0, 255);

analogWrite(6,plott);

anglee += incrementt;

pwmWriteHR(9, 32768); //duty cycle 50% = 32768

SetPinFrequencySafe(9, 1000);

…

**Step.2: Summing up the components using a non-inverting summing amplifier.**

For the next stage, we will combine the three different signal components using a non-inverting summing amplifier. First, let's get an intuitive feeling for how does a non-inverting summing amplifier works. Use the opamp rules that we discussed in class and solve for the output of the following non-inverting summing amplifier.

* Find output as a function of V1 and V2 and other circuit components.
* Find the output if Rb goes to infinity (open circuit).

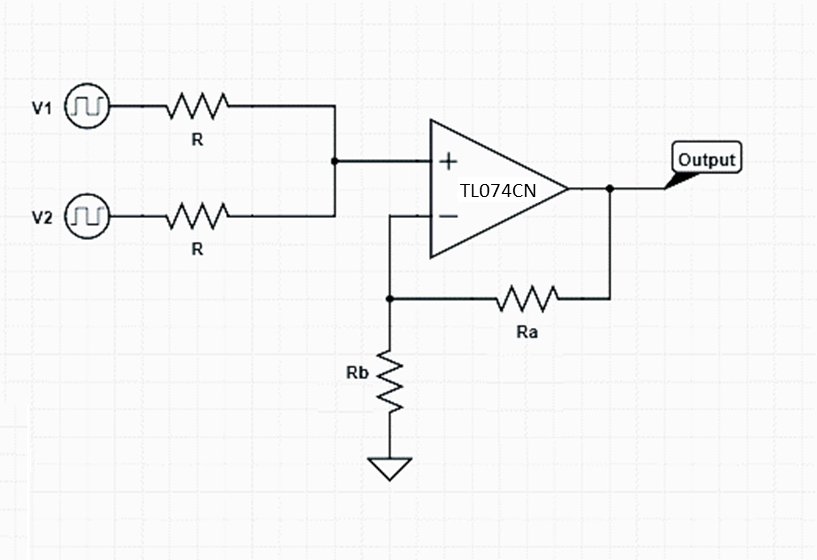


Figure.2. Basics of non-inverting summing amplifier

Note that in the previous example, we used only two inputs. However, nothing stops us from adding more and more inputs to the summing amplifier. Even the analysis would remain easy (Think about it using the superposition rule. You can input signals one by one while keeping all other sources off and see how the signals combine at the output). Also, note that in the previous example, we kept the input resistors equal to each other. This makes the weight of the V1 and V2 source at the output equal. If the two input resistors are not equal, input signals combine with different weights.

Now implement the following circuit on your bread-boards and answer the following:



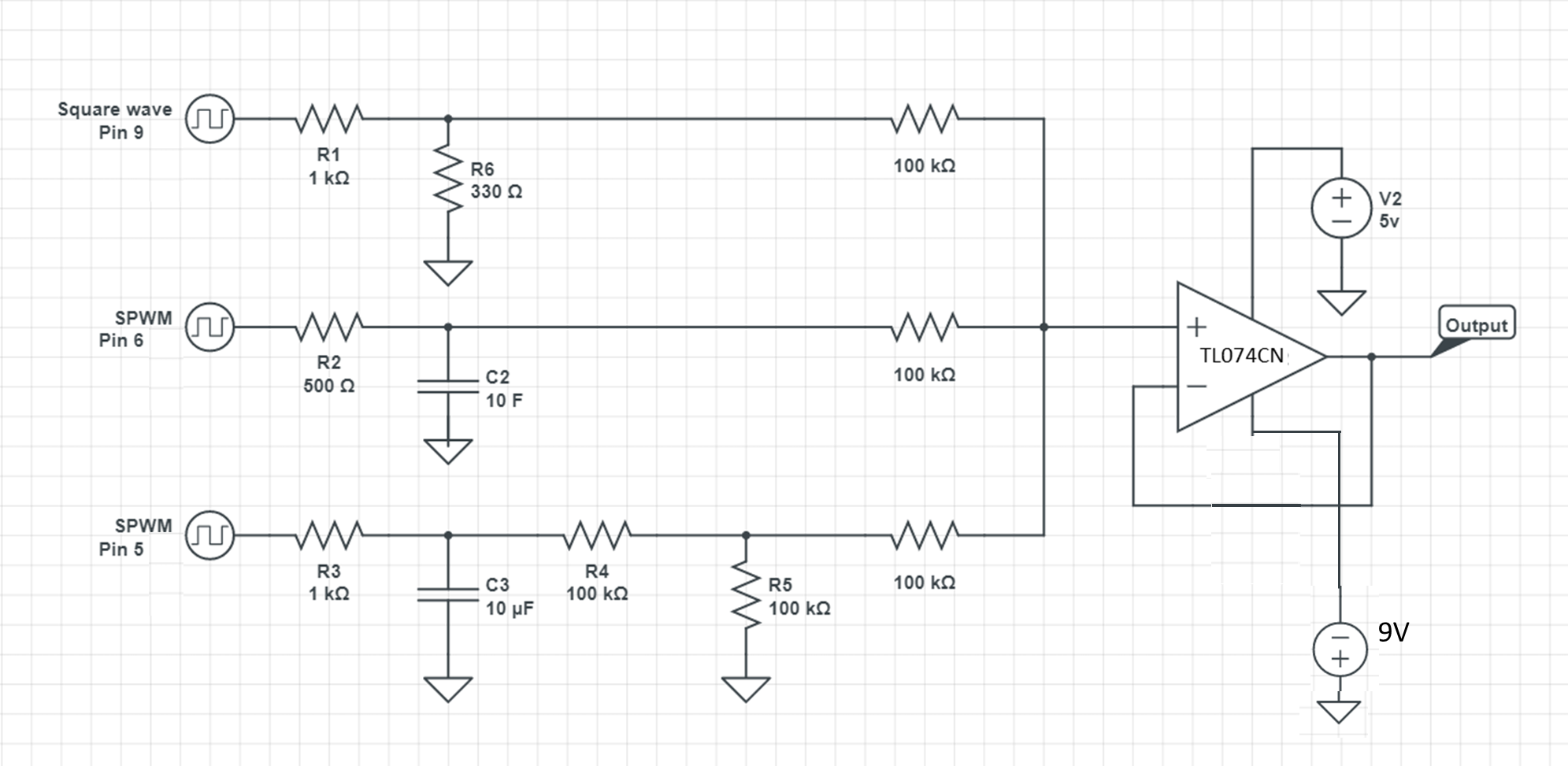


Figure.3. RC-low pass + Summing amplifier stage

**Important note:** Notice that we are using a [TL074CN](https://www.alldatasheet.com/datasheet-pdf/pdf/5777/MOTOROLA/TL074CN.html) opamp this time. TL074CN positive rail can be simply connected to Arduino's 5v. However, the negative rail should be connected to a -9V supply. The negative rail must be less than the ground due to an effect call phase reversal. [Phase reversal](https://www.analog.com/media/en/training-seminars/tutorials/MT-036.pdf) happens when the input signal (called common-mode voltage) becomes lower than the recommended operating voltage. For TL074CN, the minimum input voltage must be larger than Vcc + 4v, therefore our Vcc must be at least -4v for correct operation (For ALD1702 series opamp, since they use a rail to rail CMOS design, phase inversion does not happen).

To supply the 9v, use the 9v battery inside the Arduino package. Be careful at connecting this battery! **The positive terminal of the battery is connected to the ground rail of your breadboard (this rail is connected to your Arduino's ground), and the negative is connected to TL074CN negative rail supply (Vcc-).** I personally secured the battery and a jumper cable with scotch tape.

**Less important note:** You can also replace the third 100K resistor in the path of pin-5 (which is also at the input of the op-amp) with a 10K resistor. This will give you a more pronounced DC component to the signal which magnifies the effect of the filter. You're welcome to change other components of the noisy signal as well, however you might need to adjust some other components later in the lab yourself.

* Scope your output and verify that the summing amplifier stage is working. Your output signal should resemble the following:

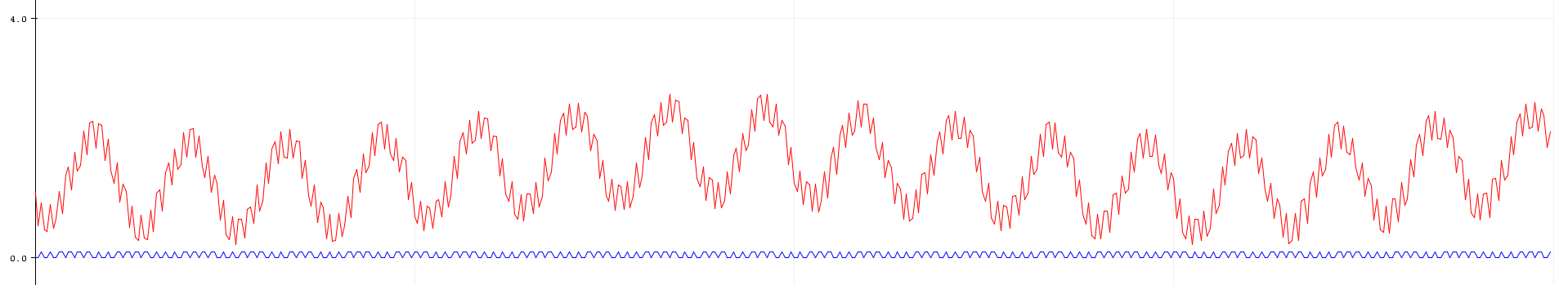


Figure.4. Final noisy signal

* Another option was to use an inverting summing amplifier. Why do you think we chose to go with the non-inverting option?
* Does the 100K ohm resistors at the input of the summing amplifier affect the RC pass filters' functionality? If no, why?

**Step.3: RLC- bandpass filter:**

Ok, so we have 1k Hz noise signal, a 16 Hz sinusoidal, and a 1.5 Hz carrier signal. Target is to design an RLC-bandpass filter that isolates the 16 Hz component. We are going to use a series RLC pass filter design. Frist consider the following circuit:

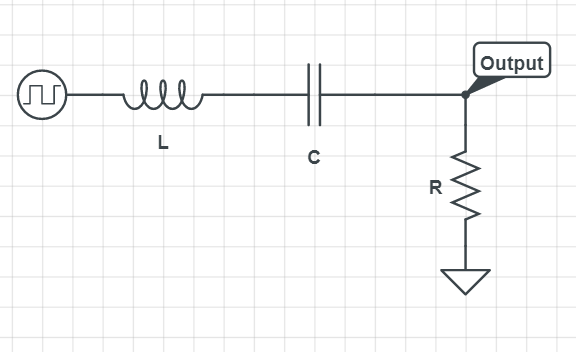


Figure.5. Basics of RLC Bandpass Filter

* Calculate the transfer function for Vout/Vin.
* Draw the bode plot of this circuit for L= 0.03H, R=40 and C=500uF. Determine the center frequency.
* Is this filter better at suppressing our high-frequency component or low-frequency component?

Note that while the above filter may not look like the best possible option for isolating the 16 Hz component. However, this circuit adheres to two critical design rules. One is that by assuming R=40 ohm, we have included the inductor's high series resistance in the calculation. Unfortunately, the filter's resistor will create an additional voltage divider with the inductor parasitic resistance (~20 ohms), which will lower our gain. Therefore, lowering the filter's resistance below 20 ohms is not recommended. Another important design rule is that the circuit is intentionally not designed to be resonant since the Arduino won't be able to supply enough current in resonance condition (if R is small, when the impedance of L and C cancel out, the resistor draws more current than what Arduino can handle). Overall, the filter is able to isolate the 16 Hz component. However, it will suppress the signal a little too, but this can be taken care of in the next recovery stage.

Now let's implement the filter. Connect the following RLC bandpass filter to the output of your last stage. Use L= 0.03H ( 3x10mH in series), C= 500uf (either 470 or 5x100uF), and R= 20 ohm. Note that R is 20 ohms since we are considering 20 ohm for the series resistance of the filter itself.



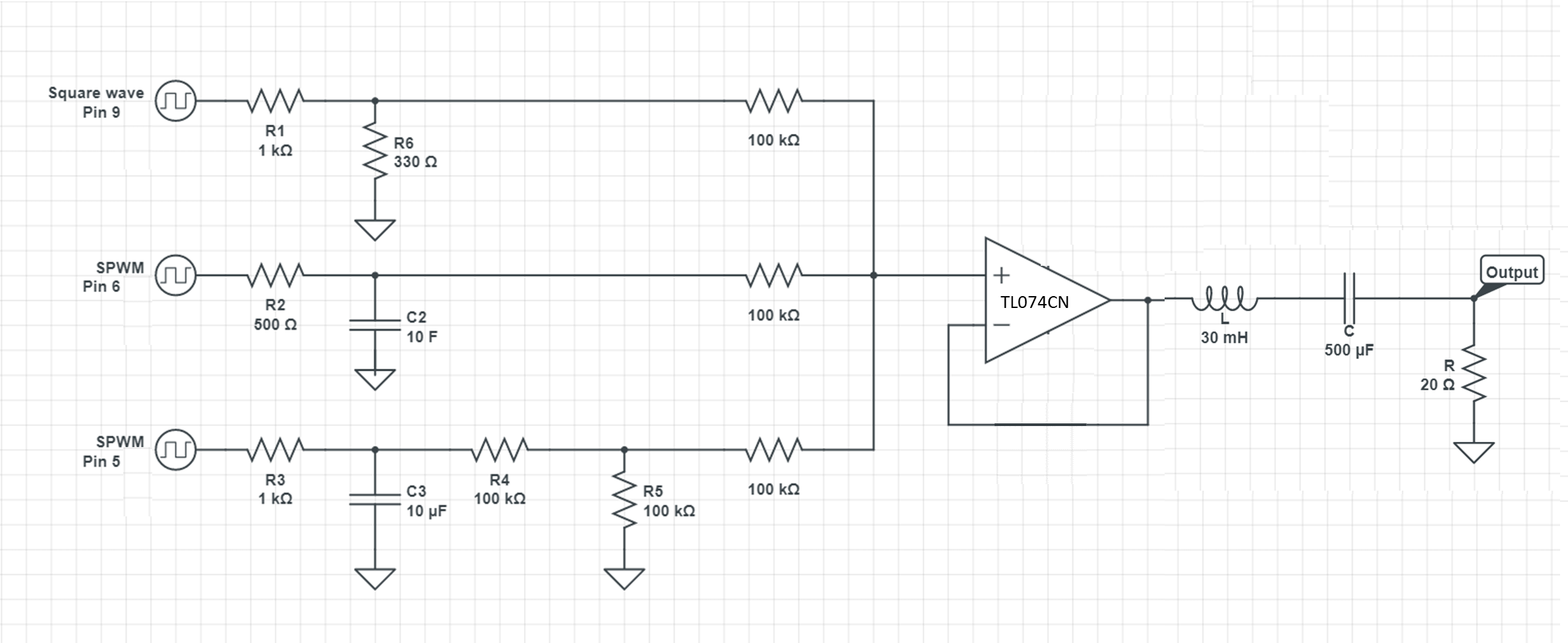


Figure.6. . RC-low pass + Summing amplifier stage + RLC band-pass stage

* Scope the output. Confirm the DC component of the signal is filtered. You should be able to see a weak 16hz oscillating signal. You can see only half of this signal since Arduino cannot read negative voltages.

**Step.4: Amplify!**

Ok, we have filtered the signal! But we faced some signal loss in the process. This is a very common downside of any filter design. The solution is to use an amplifier stage to boost the signal back to the desired level.

Let's return to the design we had for non-inverting summing opamp in step.2. You can easily configure the same opamp to act as an amplifier. Let's only assume one input to this circuit. Find the relationship between input and output for the following circuit as a function of Ra and Rb. What will be the gain of this circuit for Ra=20K and Rb= 5k?

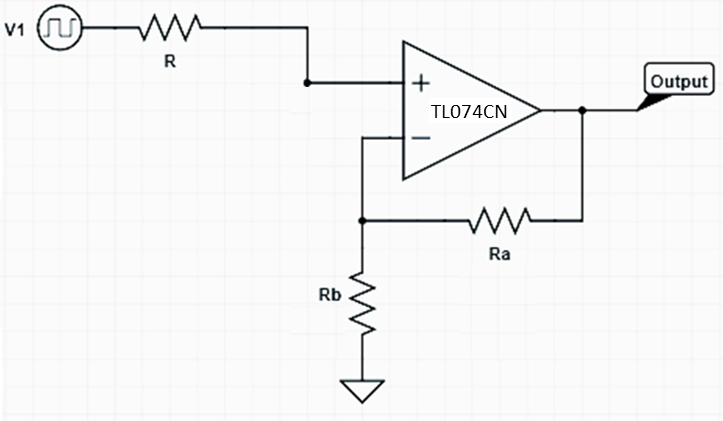


Figure.7. Non-inverting amplifier.

Let's now add this stage to our previous circuit to boost the filter's output signal. The circuit will now be:

**[Side note]:** You are probably using 2x10ohms resistors to create the 20ohms resistor in the RLC filter. If your amplification is too powerful (especially after step-5), you can take this stage's output at these two resistors' middle node.



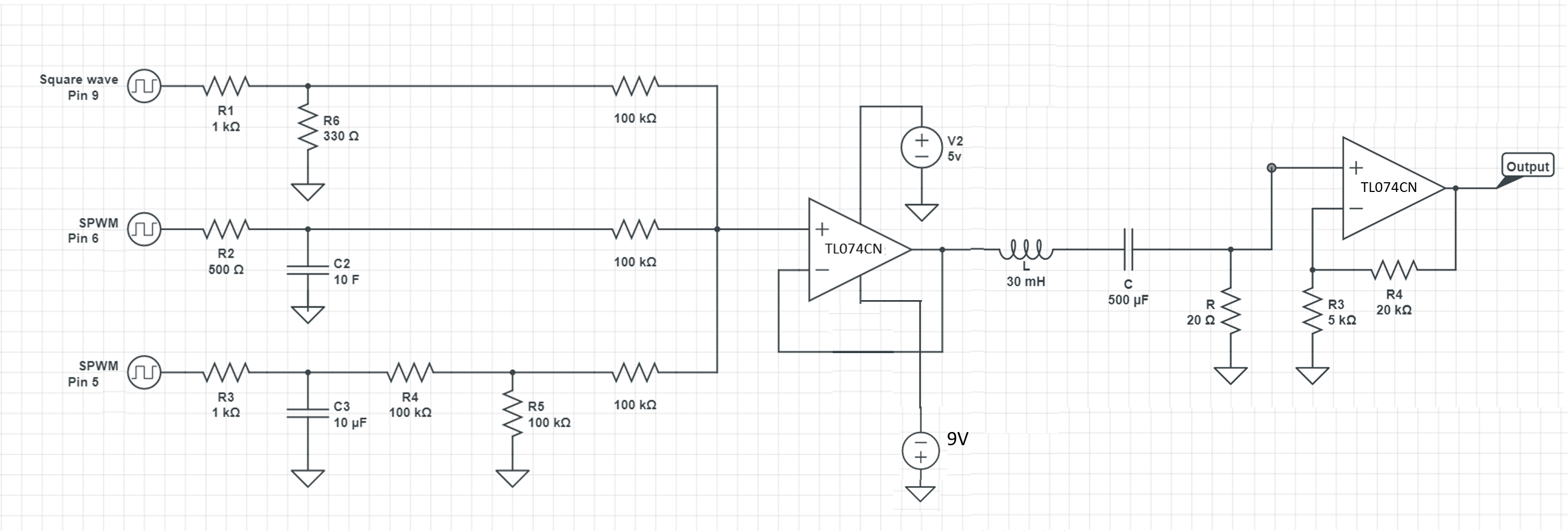


Figure.8. RC- lowpass + summing amp + RLC- pass filter + Amplifier stage.

* Scope the output. Confirm the amplification.

**Step.5: Adding the original DC level to AC signal using a Clamper circuit.**

Now we are in good shape. One last touch is to resupply the signal with the original DC level it had. Setting the DC level of AC signals has many electrical engineering applications, and one easy way to achieve that is to use clamper circuits.

* Read the following [link](https://www.tutorialspoint.com/electronic_circuits/electronic_clamper_circuits.htm#:~:text=A%20Clamper%20Circuit%20is%20a,levels%20using%20the%20clamping%20circuits.&text=A%20simple%20clamper%20circuit%20comprises,a%20dc%20battery%20if%20required.) ( or any other resource you like) and describe how does a clamper circuit functions in a paragraph.

Finally, let's add the icing on the cake. We will use the Arduino's 3.3v, 47 uF capacitor and a 10K resistor to add a 3.3V to bring up the signal level. Finish your implementation as follows. - Scope/plot the output of your circuit and confirm the DC raise of the signal. How do you think we can improve this design?



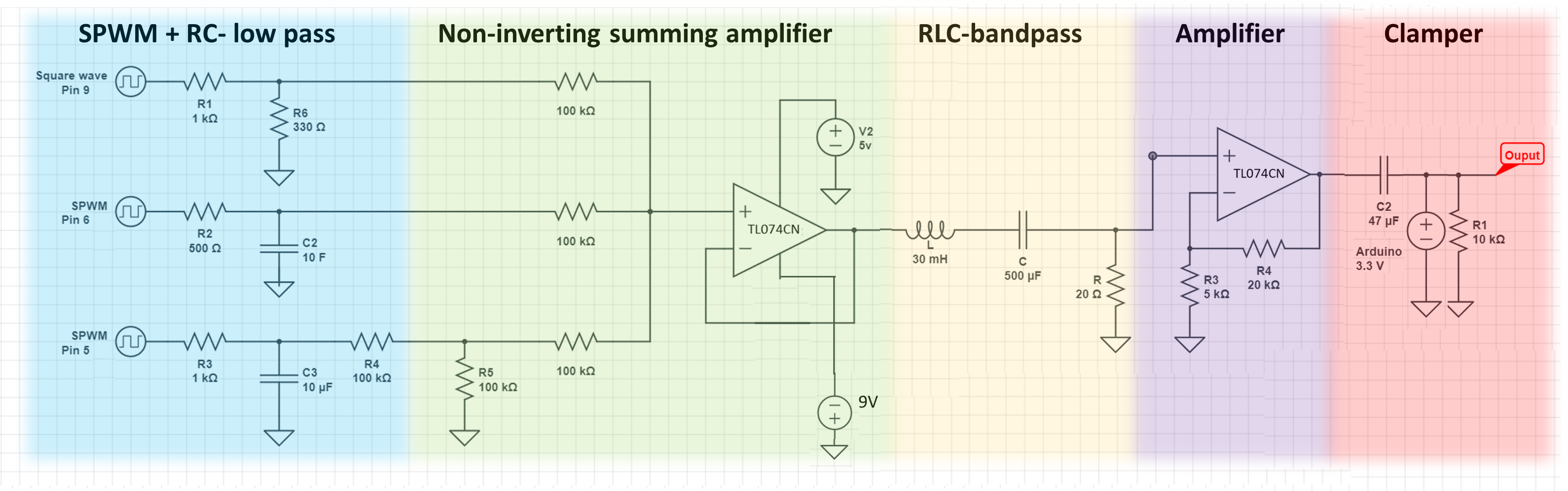


Figure.9. RC- lowpass + summing amp + RLC- pass filter + Amplifier stage + Clamper stage.

As a final note, I hope you now have an intuition for electrical circuits' modular design process. Note that how opamps, in addition to the functions they offer, also provide impedance matching and isolation between separate stages of the circuit. Finally, don't break-up the bread-board. We will use the first two-stages of the same design in the next lab.