The City College of New York

School of Engineering
EE22100 Electrical Engineering Laboratory 1
Spring Semester

Lab Report Experiment #7
Digital Clock

Instructor: Muhummad Ifran
Student Name: Laith Toom
Partner Name: Dennis Lee
4/20 2024

Contents

List of Figures

List of Tables

1 Objective

Using the knowledge gained from previous labs and prior classes, we will construct a 12-hour or 24-hour digital clock using ICs (integrated circuits) and seven-segment LED displays. While the ICs will assist in making a compact design and help simplify the circuit by handling the circuit design of digital logic, an understanding of boolean algebra and digital logic is required in order to design the clock.

2 Procedure

The clock, regardless of being a 12-hour or 24-hour version, must be able to count seconds, minutes, and hours. Each of these counters can be designed separately as they only need to be cascaded for the final design. Also, the seconds and minutes counters will actually use the same design, so the design process of the clock will be split into three phases:

Phase 1: Design the *seconds* counter.

Phase 2: Design the *hours* counter.

Phase 3: Design the clock signal generator.

For the third phase, the goal is to design a clock signal generator to replace the function generator provided by the lab. If the clock signal generator is successfully constructed, then the entire clock should only need a DC source to function, such as a 9-volt battery or a pack of 6 AA batteries.

Remark 1 (Seven-Segment Display). The seven-segment display consists of seven LEDs (light-emitting diodes) that are arranged to form the number eight.

Figure 1: Six individual seven-segment displays.

Each LED is assigned a letter from "a" to "f", and each LED will be activated (light up) when a voltage passes through from the anode to the cathode of the diode. We want specific diodes to light up in order to display a specific number from 0 to 9, thus there must way to control the voltage through each diode. Thus, we will use a driver for each display that will accept a 4-bit binary input and output a 7-bit code. When a bit on the output code is 1, some voltage will be passed through the diode it is wired to, and 0 will not send a voltage strong enough to light up the segment. The basic design for a single digit will consist of a counter and a driver; which is illustrated in Figure 2.

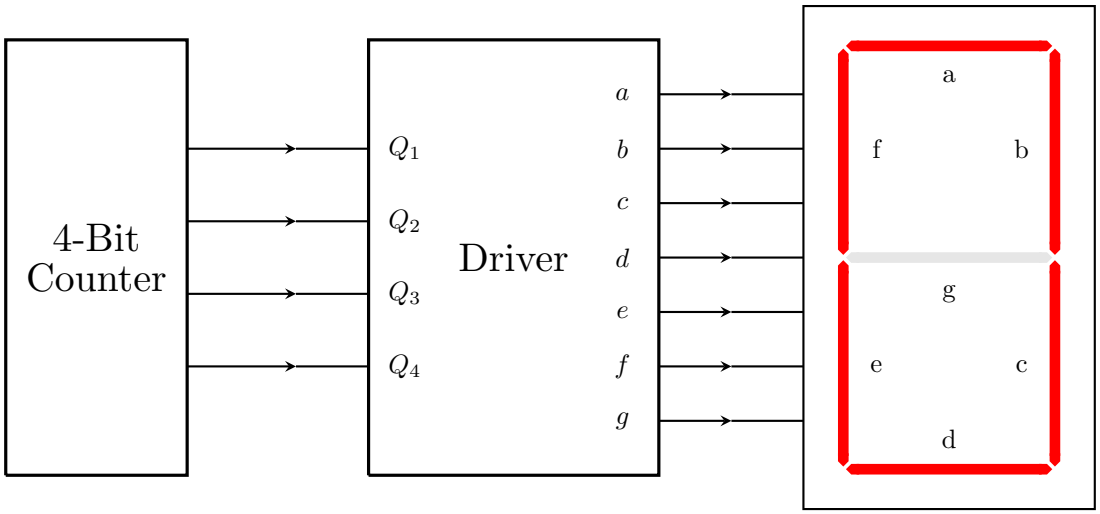


Figure 2: Driving a single seven-segment display.

2.1 Phase One: Counting Seconds

Our system of time establishes that we count seconds from 0 to 59, meaning we reset counting seconds when we reach sixty. We can split the number of seconds into two digits: the high digit and the low digit. The low digit can range the entire range of numbers that a single seven-segment display can offer (0 to 9), but the high digit can only range from 0 to 5. Using Figure 2 as our base model, we want the low and the high digit to have their own 4-bit counter, driver, and display. However, the low digit will not need to be reset and we want it to count at the same frequency as the clock signal. For the high digit, we only want to increment by one when the low digit reaches 9, which is to say that the high digit should count only when the circuit counts 10 seconds. Thus, the high digit counter should only be enabled when the low digit counter reaches nine. We also want the high digit to reset when it reaches 6, thus the high digit counter will involve a reset and it will be cascaded with the low digit counter.

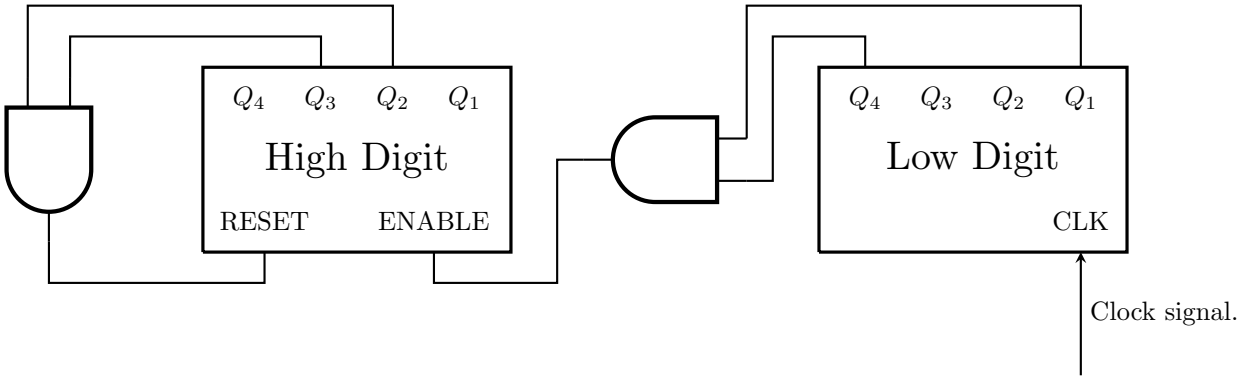


Figure 3: A two-digit counter design for seconds.

Naturally, the counters and logic gates will be implemented as ICs.

2.1.1 Minutes Counter

While we can copy this design for the minutes counter, we need to make a slight modification to the input of the minutes counter. The minutes counter should only count once every time the seconds counter reaches sixty seconds, and we can apply the same logic used for the AND gate connection shown in Figure 3. When the number of seconds is 60, the low digit is 0 and the high digit is 6, but the low digit being 0 occurs for

any value the high digit takes while the high digit being 6 is a unique event. Thus, we can use an AND gate to determine when the high digit for the seconds counter is six. We can use the AND gate output in this case to be the clock signal input of the minutes counter. If the bits on the top edge of the seconds counter are the bits of the high digit, then the clock input of the minutes counter should resemble Figure 4.

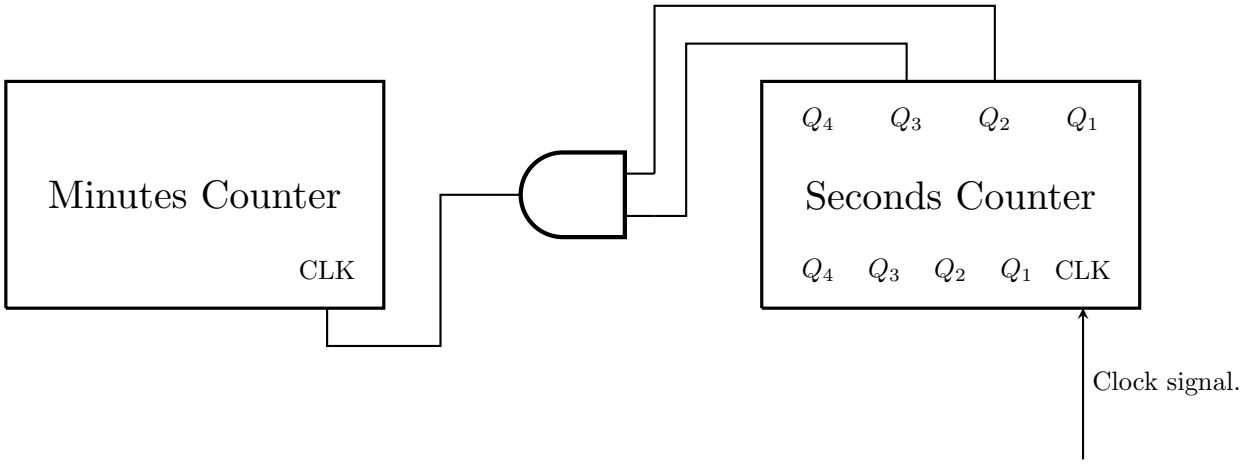


Figure 4: Cascading the seconds and minutes counter.

With this design, the frequency of the clock signal input of the minutes counter will be 60 times less than the frequency of the clock signal input of the seconds counter; in which the seconds counter recieves a clock signal from an AC source such as a function generator or, as planned for phase three of the design process, a time base generator.

Remark 2 (Logic gates and digit bits.). Each digit contained in the circuit will consist of four bits in descending order, which is to say Q_4, Q_3, Q_2, and Q_1. Each bit can only be 0 or 1 (LOW or HIGH voltage), and a table can be written to track the behavior of each bit for each decimal number.

Table 1: Behavior of bits for a 4-bit counter.

n	Q_4	Q_3	Q_2	Q_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Since we are counting up from 0, we only need to concern ourselves with the bits of the decimal numbers less than or equal our number of interest. In the case of the number 6, we see that if Q_2 and Q_3 are 1, then the number the counter reached is six. While this condition is also true for the number 7, we do not need to consider this because the counter will reach 6 before it reaches 7, thus the conditional statement $Q_2 \cdot Q_3$ will suffice. If we did want to design a counter that stops or resets counting at 7, then we would need a three-input AND gate, or two cascaded binary AND gates.

If instead of a single digit, you are interested in finding when a series of 4-bit counters reach a particular double digit such as 24 or 13, you can use two copies of Table 1 and find a condition that matches the desired number.

The type of ICs you use to implement these designs into the real world may depend, but the CD4543BD is a good choice for the seconds (and minutes) counter since it is a dual-sided counter; in which it contains two 4-bit counters. As for the use of AND gates, the CD4081 is a good choice since it contains four AND gates.

2.2 Phase Two: Counting Hours

Remark 3 (ICs used for the hour counter.). If you wish to make a 24-hour clock, then you could use the same design used for the seconds counter. However, in the event that you do not have enough CD4543BD ICs, which is most likely the case for the equipment you purchased for this lab, or you wish to build a 12-hour clock, you will be used the CD4510BD IC to count each digit.

2.2.1 A 12-Hour Clock

For a 12-hour clock, the hours should start at 1 and end at 12, in which the counter resets when the thirteenth hour is counted. Thus, we need to preset the low digit counter to 1 and the high digit counter to 0 so that the clock displays one hour first. We also want to reset both digits to zero when the number of hours reaches 13, in which the high digit is 1 and the low digit is three.

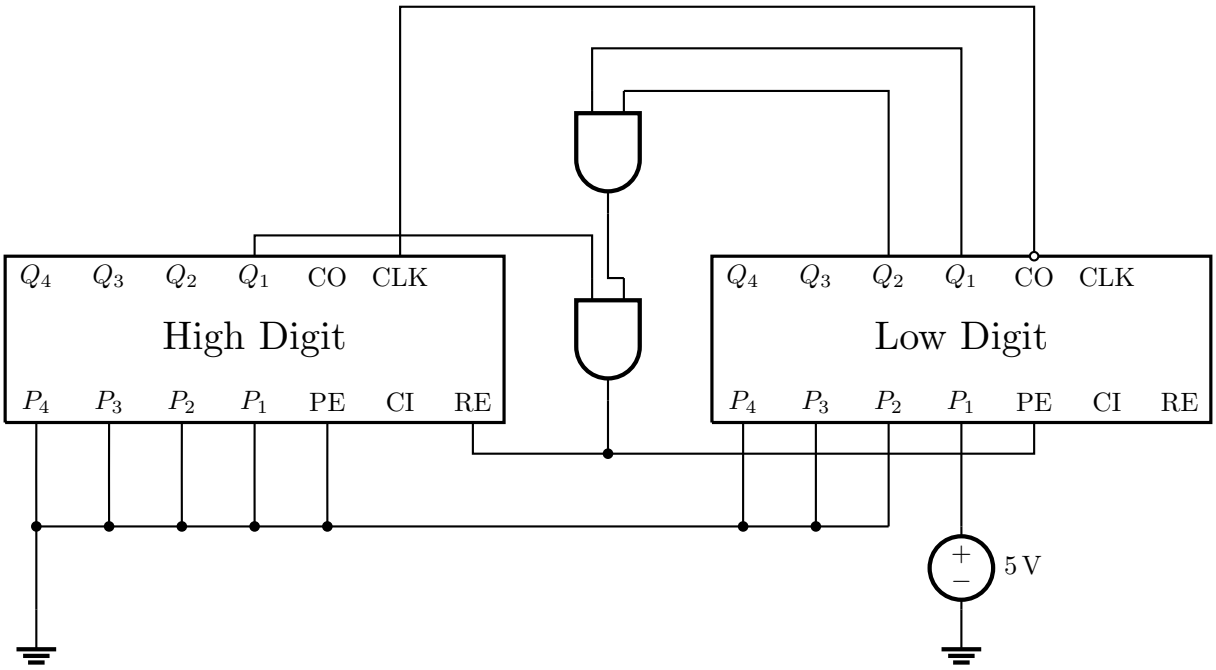


Figure 5: A two-digit counter design for a 12-hour clock.

The cascaded AND gates will output 1 once the low digit is three and the hight digit is one. When the AND gates output 1, the low digit will be set to the binary number 0001 and the high digit will be reset to zero. Each counter should only count from 0 to 9 despite having enough bits to count beyond 9, and so the carry out of the low digit counter will occur after the low digit counter returns to zero. This will then cause a pulse in the clock signal input of the high digit counter so that the high digit can go from 0 to 1. The cycle will then repeat, thus the hours will be displayed from 1 to 12 periodically depending on the clock input of the low digit counter.

2.2.2 A 24-Hour Clock

The design for a 24 hour clock will be much simpler since it does not require any of the digits to be preset. Instead, the hours will be displayed from 0 to 24 only.

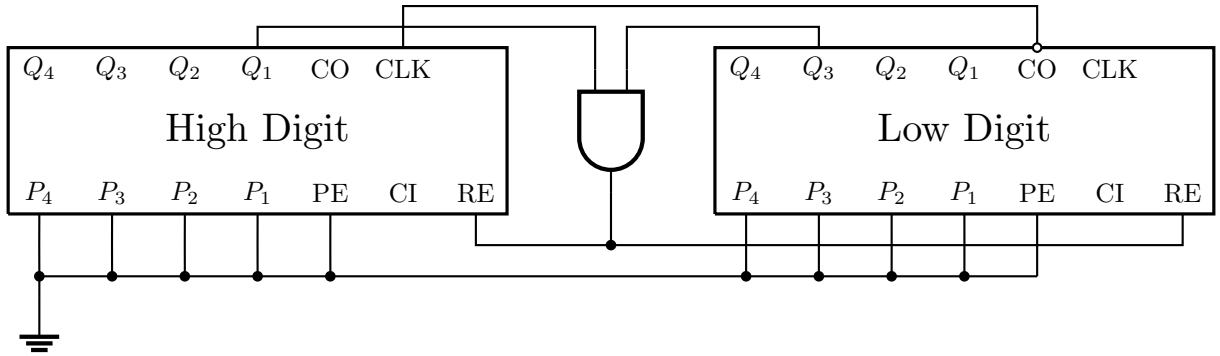


Figure 6: A two-digit counter design for a 24-hour clock.

Instead of requiring two AND gates and an enabled preset, the 24-hour clock only needs one AND gate to determine when the hour is 24, at which point both counters will be reset.

2.3 Phase Three: Signal Generator

A signal generator can be constructed by using a quartz crystal; where the crystal vibrates when a current is passed through it. This vibration allows the circuit to output a periodic signal, which then becomes our *clock* signal.

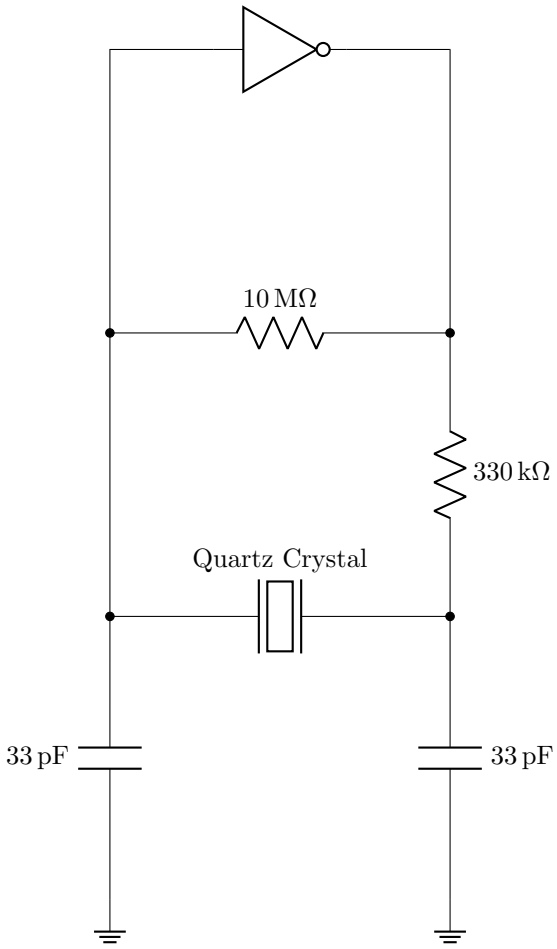


Figure 7: The signal generator circuit.

However, the frequency of the output signal will be, for the crystal included with your equipment, 32.768 kHz, but we need a frequency of 1 Hz for our clock. Thus, the frequency of this signal must be decreased, yet the frequency output of the circuit is constant. Instead, we can pass the output through a 16-bit counter, in which we will take the 15th bit to be our clock signal input for the seconds counter.

Table 2: Frequency as order of bits increases.

n	Q_{16}	Q_{15}	...	Q_4	Q_3	Q_2	Q_1
0	0	0	...	0	0	0	0
1	0	0	...	0	0	0	1
2	0	0	...	0	0	1	0
3	0	0	...	0	0	1	1
4	0	0	...	0	1	0	0
5	0	0	...	0	1	0	1
6	0	0	...	0	1	1	0
7	0	0	...	0	1	1	1
8	0	0	...	1	0	0	0
9	0	0	...	1	0	0	1
⋮	⋮	⋮		⋮	⋮	⋮	⋮
65530	1	1	...	1	0	1	0
65531	1	1	...	1	0	1	1
65532	1	1	...	1	1	0	0
65533	1	1	...	1	1	0	1
65534	1	1	...	1	1	1	0
65535	1	1	...	1	1	1	1

The first bit oscillates every other decimal number; the second bit oscillates every two decimals; the third bit oscillates every four decimals and so on. So, the 15th bit will oscillates every 2^{15} decimal numbers, or simply every 32768 seconds if the first bit oscillates every one second, which means the frequency of the output of the 15th bit is about 30.52 μHz. So, if we have a 16-bit counter and let the clock input be the output of the signal generator circuit, then the first bit will oscillate every 30.52 μs, and so the 15th bit will 2^{15} times less than the first bit, which leaves us with an output signal oscillating every 1 second or with a frequency of 1 Hz. Essentially, we were able to divide the frequency of the input frequency by using the behavior of bits as the order of bits increases (see Table 2).

If you do not have access to a 16-bit counter, which is the case for the equipment you have, then you can simply cascade two 8-bit counters or four 4-bit counters. For this design, we will cascade four 4-bit counters.

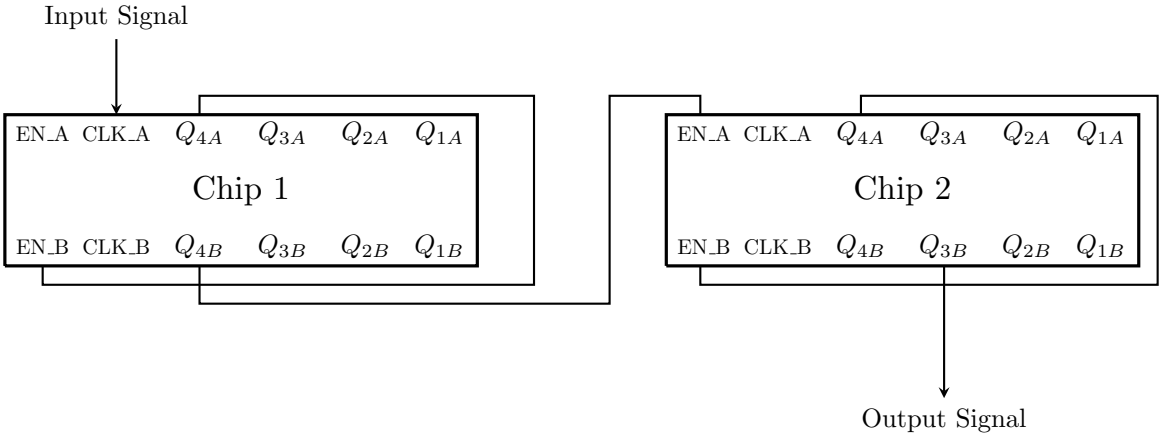
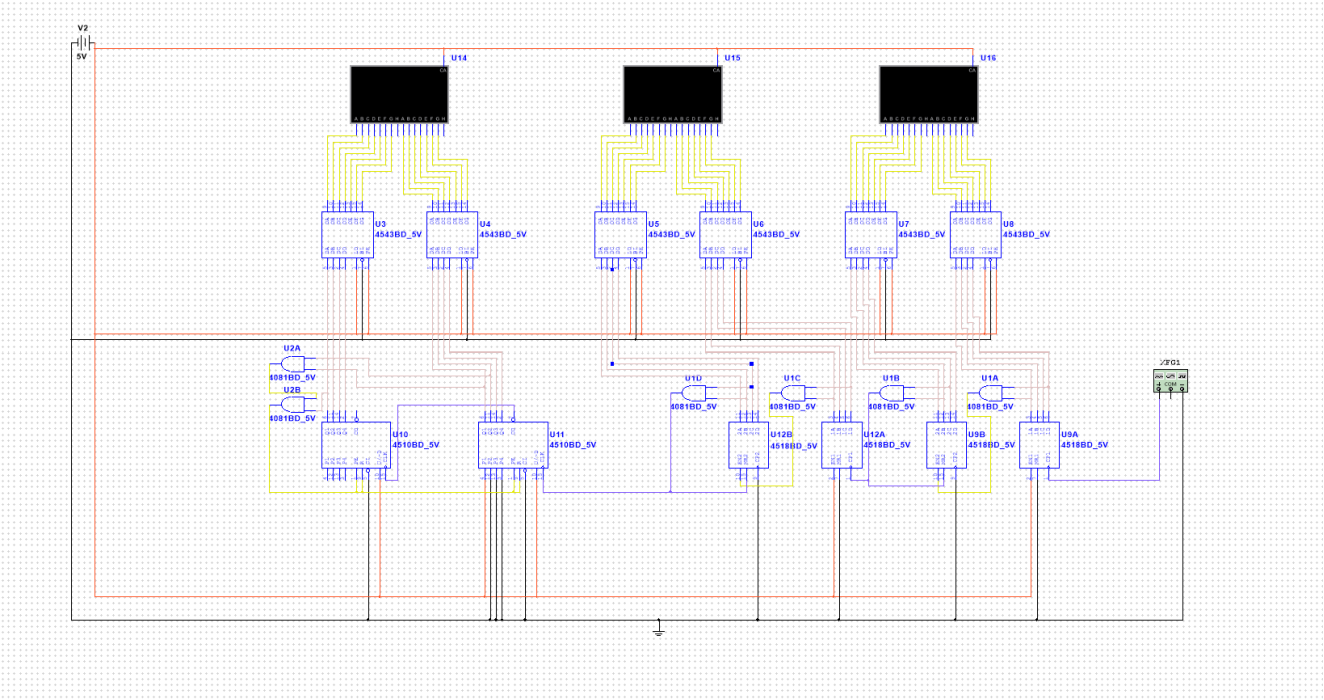


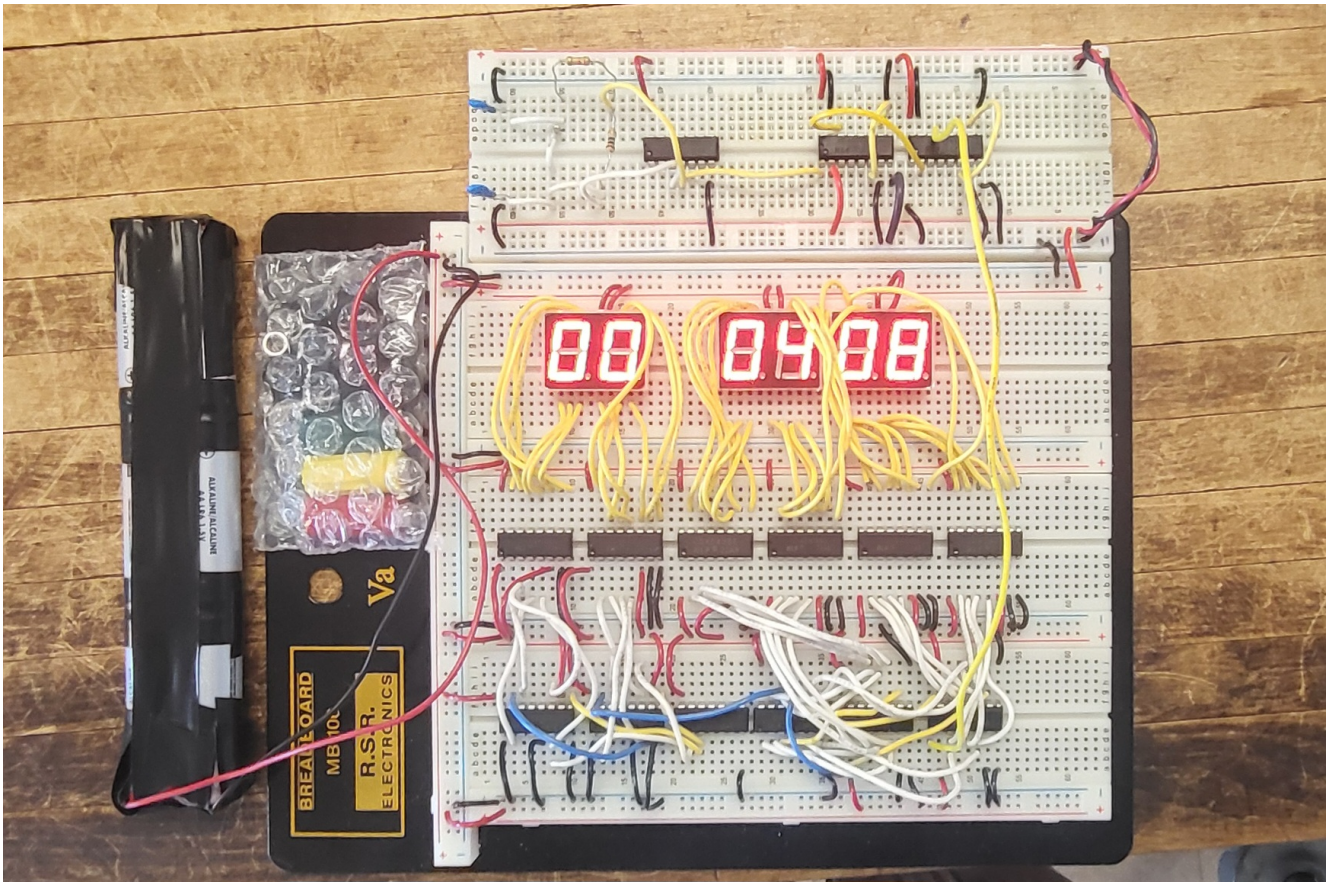
Figure 8: Dividing the frequency of a signal by cascading two dual 4-bit counters.

3 Clock Design

The clock was designed on a protoboard and is shown below.



The real design is shown below.


R.S.R.
ELECTRONICS