Oscilloscope Pong

Contents

Problem and Solution 1

Specification 2

Block Diagram 2

Full System Circuit Diagram 3

Full System Photo 5

Bill of Materials 6

Subsystems 7

- ✔ Triangle Wave Generators: 7
- ✔ Square Wave Generator (Bat Height): 9
- ✔ Low Pass Filter (Bat Height): 9
- ✔ Summing Amplifiers: 11
- ✔ Multiplexer and Control Clock: 12
- ✔ Ball Sides Comparators: 14
- ✔ Ball Height Comparators: 14
- ✔ Ball Reset: 16
- ✘ Speaker Output (Beeps and Boops): 17
- ✘ Counter Logic: 18

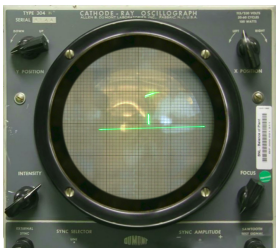
Whole System Test 20

Evaluation 21

User Guide 22

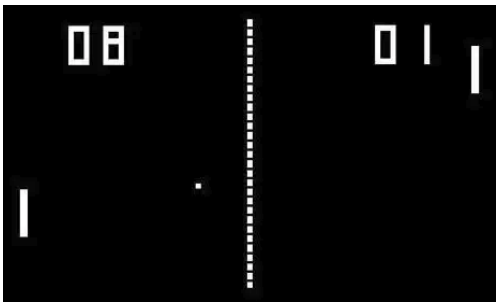
Problem and Solution

I cannot afford a games console, but need to entertain my friends with some local multiplayer games. I own no TV, so I need a console that can connect to my oscilloscope.



There is a long history of games on oscilloscopes, almost as old as video games themselves. Tennis for Two was invented in 1958, driven by an analog computer. As it was a one-off created for an exhibition, it was highly advanced compared to commercial games that followed, featuring drag and wind speed.

One of the simpler games that followed was Pong, a simple game made by Atari in 1972 that was ported to almost any and every device of the time. It has simple rules and simple gameplay that makes it very feasible to reconstruct it to work with my oscilloscope without requiring a microprocessor.



Specification

Quantitative variables:

- Ball bounce frequency between 0.5 and 1 Hz
- Screen draw frequency faster than 30Hz to avoid flicker
- Bat height of within 20% and 40% of the screen height
- The system must take 2 or less cycles to return to normal operation after a reset

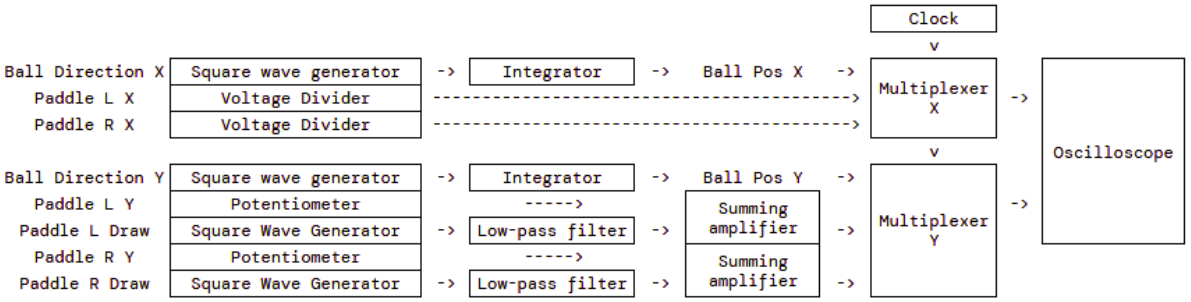
Qualitative variables:

- Responsive controls
- Somewhat accurate motion paths
- Controls to calibrate the device in case of inconsistent voltage, room temperature etc
- Multiple multiplexed elements drawn on screen at once

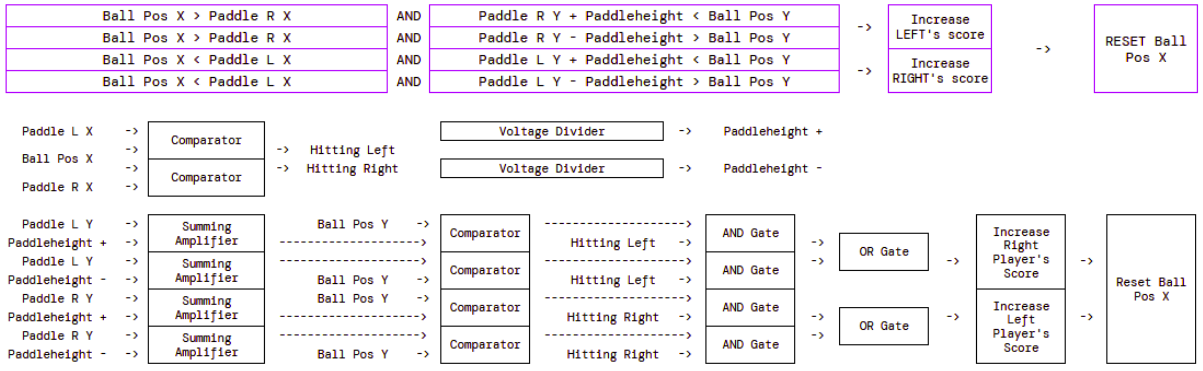
Block Diagram

I drew the block diagram in two parts as the logic system operates almost entirely independently of the majority of the drawing system.

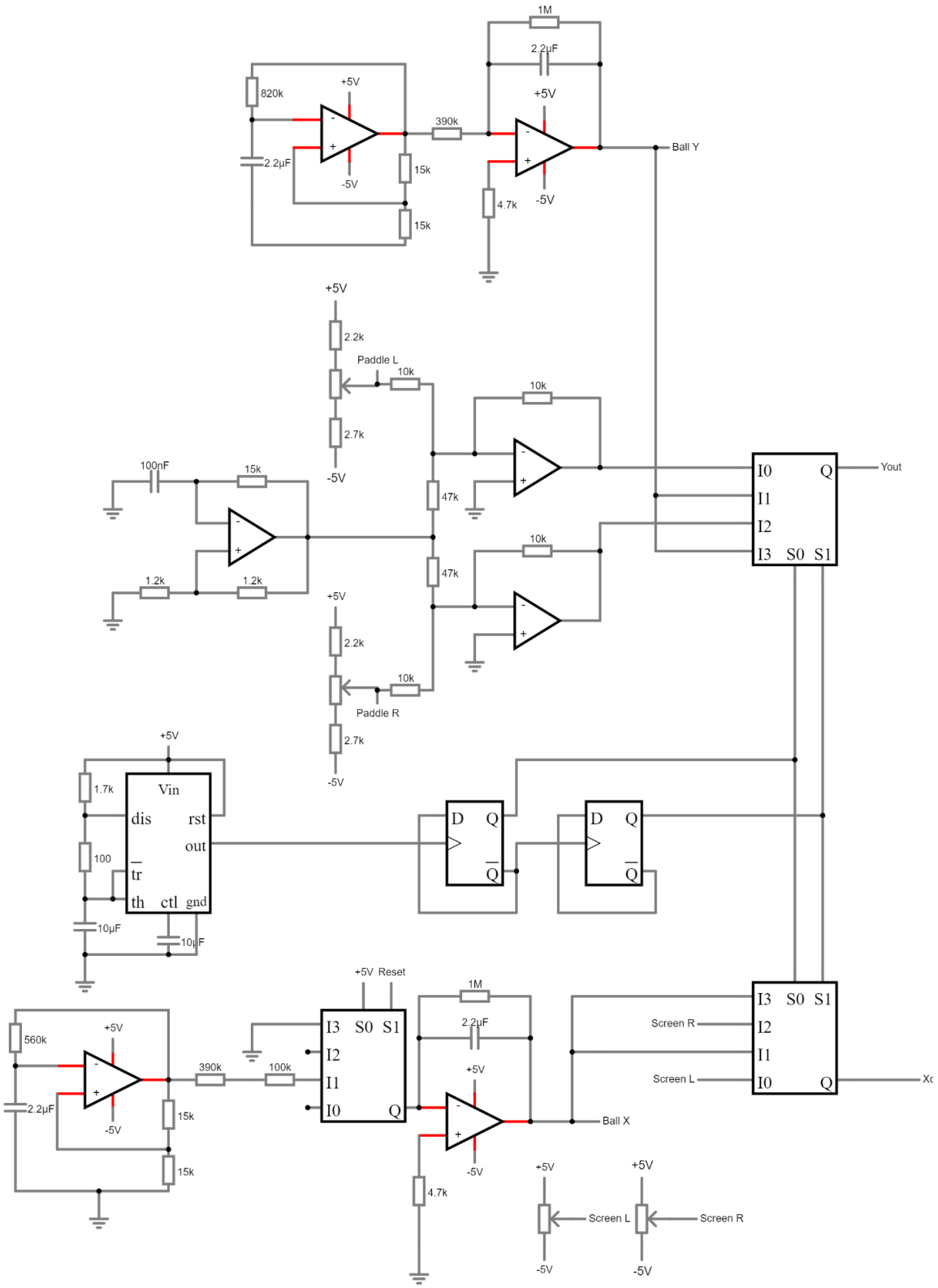
Drawing to the screen:

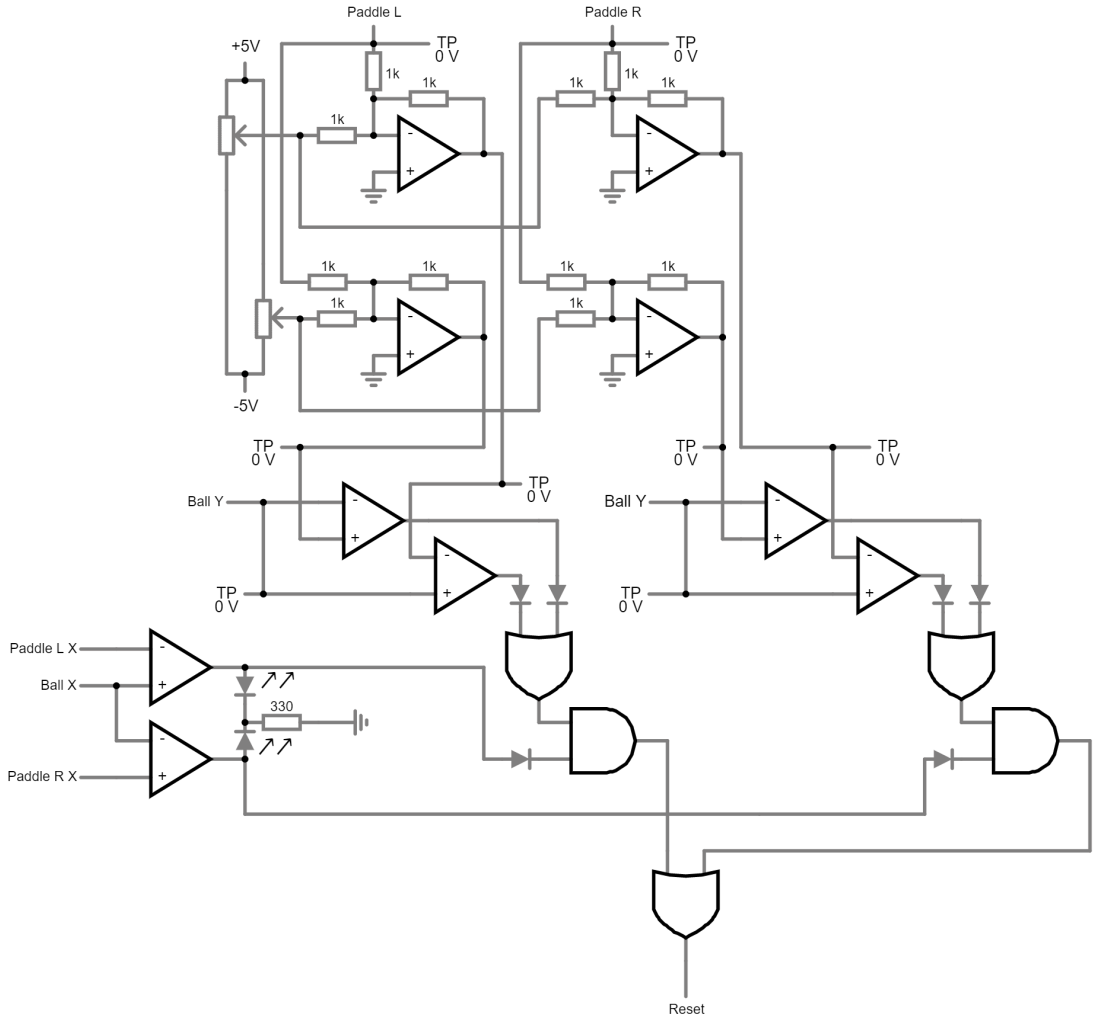


Logic: the purple blocks are a more programmatic way of writing the logic and provided to hopefully help explain what the logic is doing, while the black blocks are the more complex electronic implementation.

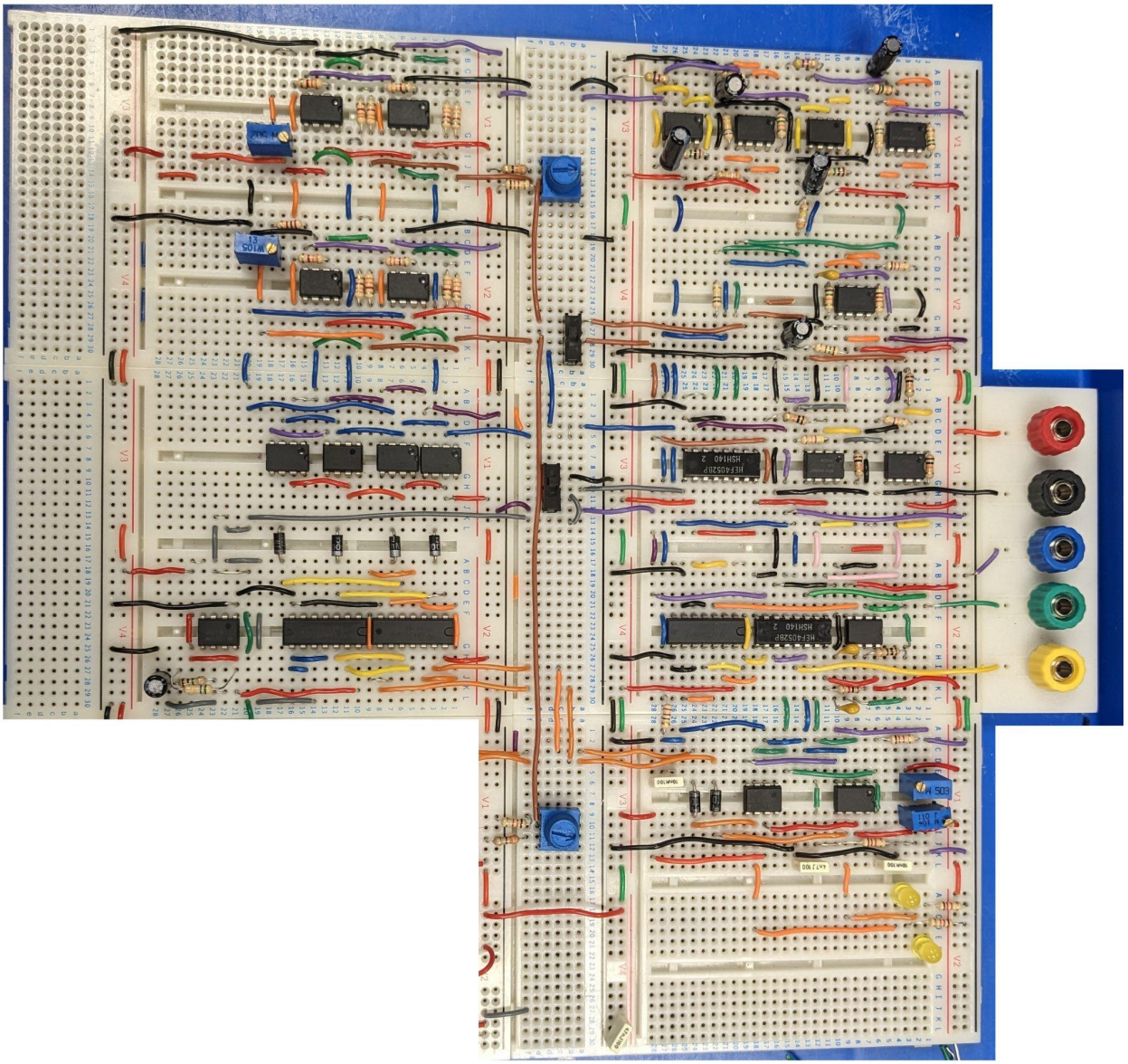


Full System Circuit Diagram




Paddle L
Paddle R
+5V
-5V
TP
0 V
1k
Ball Y
Paddle L X
Ball X
Paddle R X
330
Reset

Full System Photo



The gap in this circuit photo was to remove a subsystem I ran out of time to complete (the counter) as it was messy and had wiring not up to the standards of the rest of the circuit.

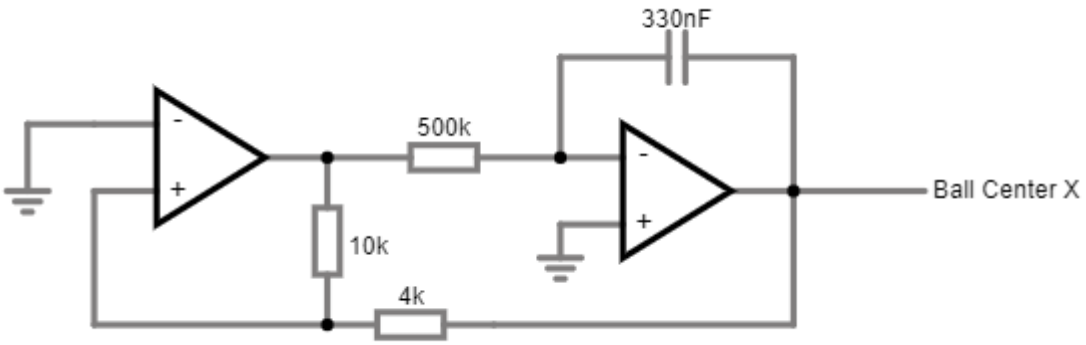
Bill of Materials

ICs	Potentiometers	Resistors	Capacitors	Diodes
17 x LM741s 2 x 555s 2 x CD4052s	2 x 5k pots 1 x 50k trimpot 1 x 100k trimpot 1 x 5k trimpot 1 x 1M trimpot	100: 1 330: 1 1k: 12 10k: 4 12k: 1 15k: 5 17k: 1 22k: 2 27k: 2 47k: 4 100k: 1 390k: 2 560k: 1 820k: 1 1M: 2	2 x 10u 5 x 2.2u 1 x 100n 4 x 47n (not shown in diagram)	6 x LN4001 diodes 2 x 2V LEDs

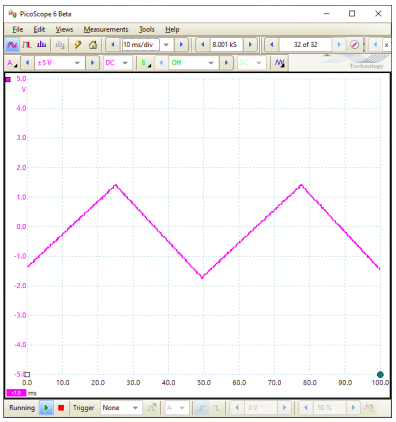
Subsystems

✔ Triangle Wave Generators:

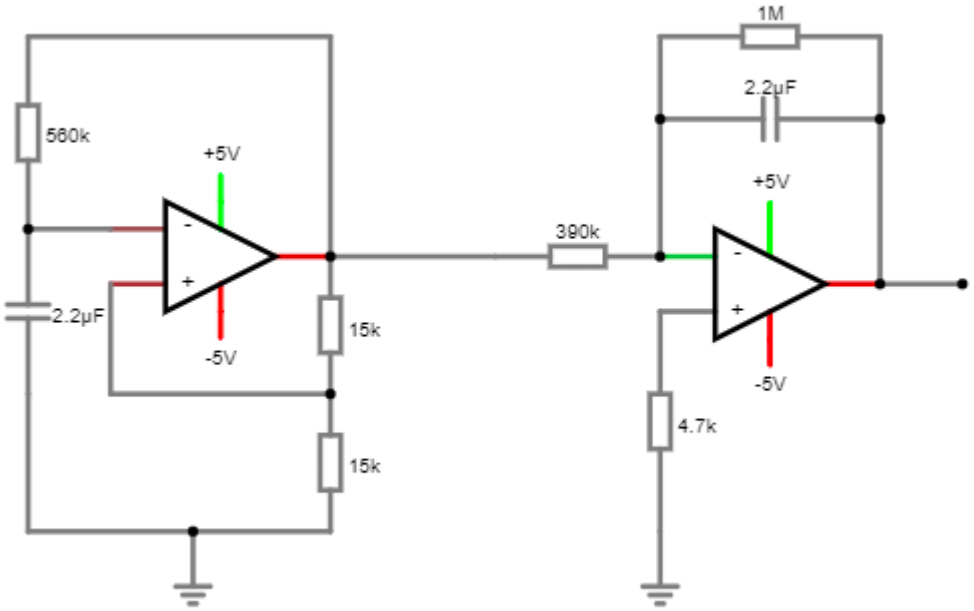
My first attempt at a triangle wave generator was this design:



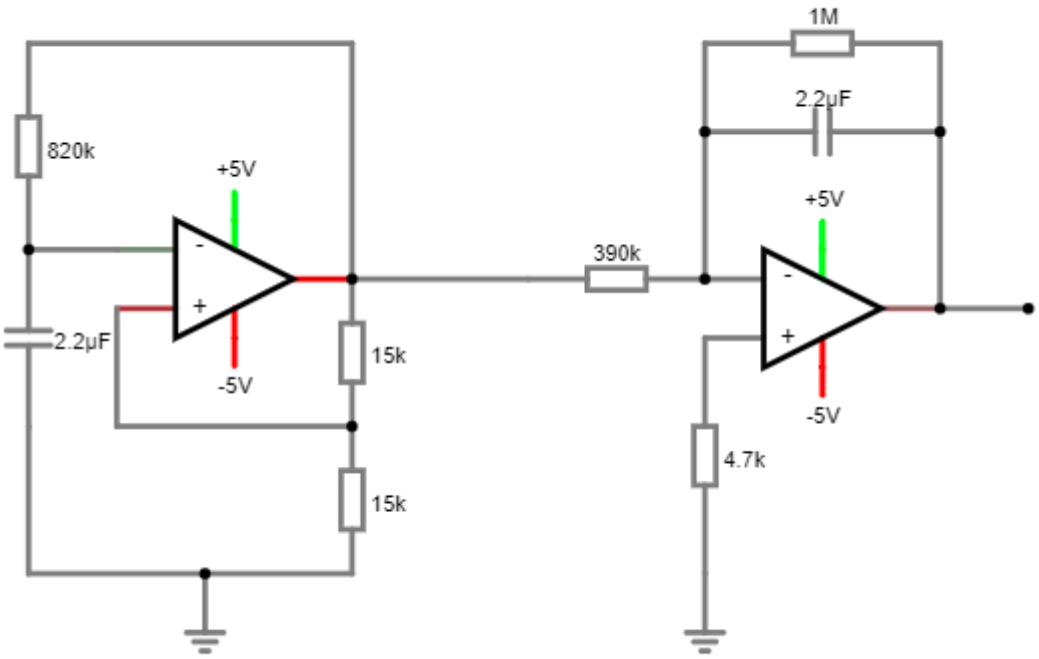
This combines a square wave generator and an integrator to save a few components, and produces a very good triangle wave at high frequencies (as seen in the graph to the right), however at low frequencies the capacitor discharge curve becomes clear and the triangle wave is lost. Small deviations in the gradient of the triangle wave are clearly visible as changes in velocity for the ball, so this was an unavoidable issue.



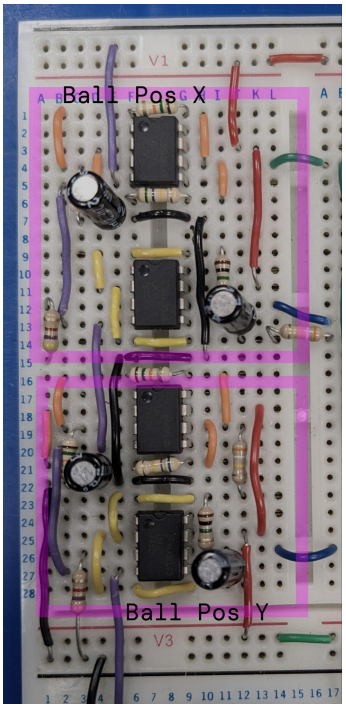
Splitting this system into two components, a square wave generator and an integrator, increases the number of components but makes the system much more flexible with lower frequencies.



The Ball X generator is identical to the Ball Y except for the 580k changed to be slightly higher to create a more interesting bouncing pattern (the less factors shared between the two oscillators frequencies the more locations the ball is able to reach)



Once I had a good frequency for the ball to move at, I recorded the bounds of the wave, as this would be needed to set the voltages for the screen boundaries later on.



The actual circuit was designed to fit compactly into one column of the breadboard, however later constraints led to requiring the only diagonal wire in the entire project.

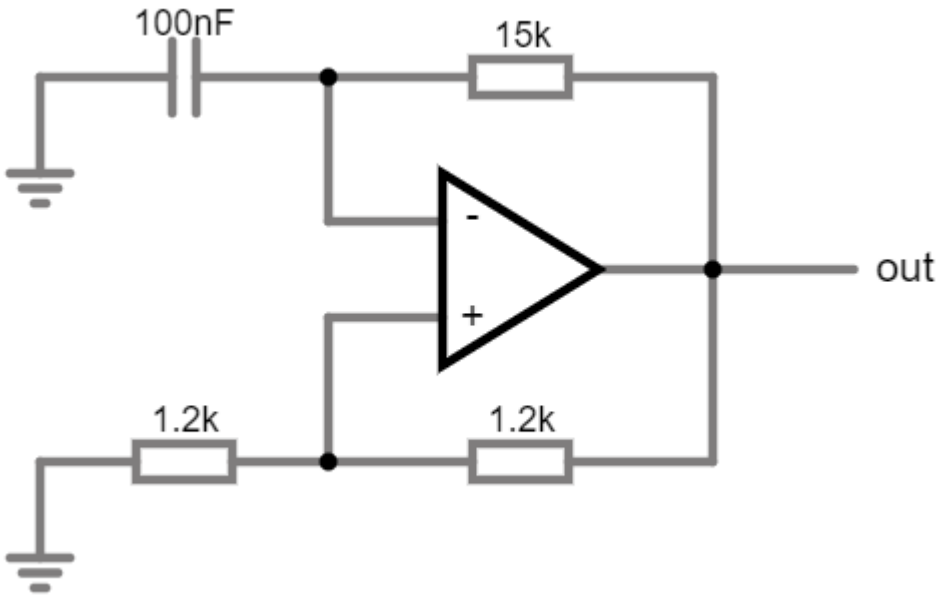
The Ball Pos X circuit has one additional feature that I will discuss later that requires the square wave and integrator to be separated to insert more circuitry in between them; this is why they are not connected and instead branch off out of this image and into the column to the right.

This circuit worked perfectly until one testing day I had a power supply issue that blew a few chips on the board. While the circuit itself was seemingly fine, the resistance of the board had changed significantly enough to create clipping on the triangle wave - its charge rate had become high enough to reach the supply voltage of the 741 amplifier. Visually, this had the effect of the ball sticking to walls and sliding along them. I solved this issue by adding a significant resistor in between the integrator and the square wave generator.

This of course changed the new minimum and maximum positions of the ball, so the entire circuit was recalibrated around the change.

✔ Square Wave Generator (Bat Height):

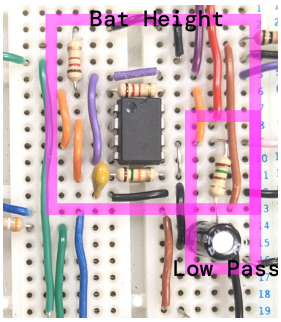
I did very little calculations for this block, as it did not need to be precise, just quick enough for the movement to be invisible to the human eye.



It is important that this component works at a frequency with as few common factors with the display multiplexer as possible, as we want the samples to cover the entire wave.

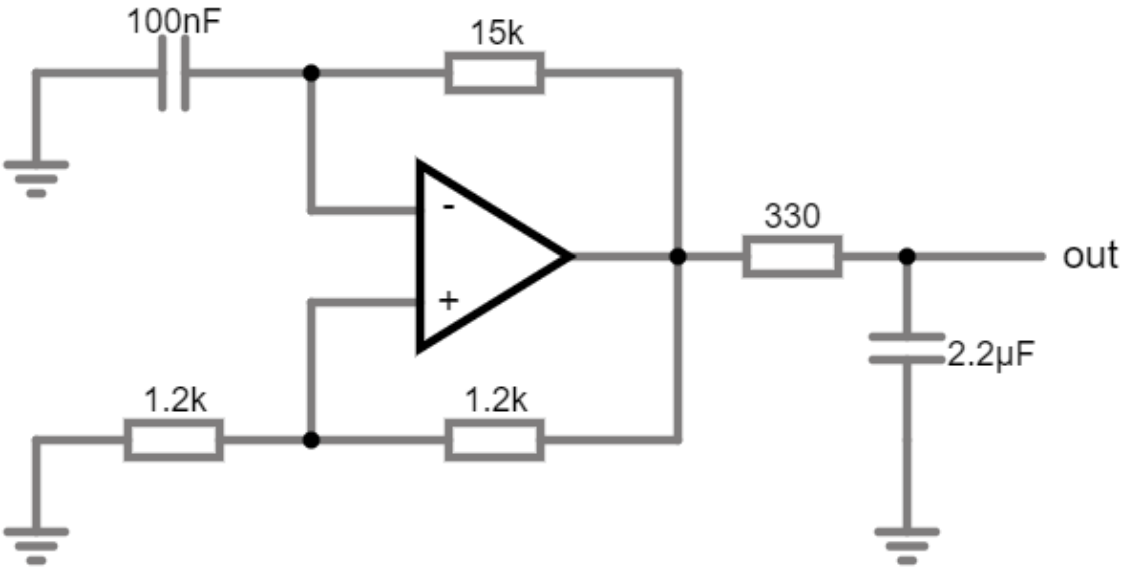
Interestingly, fluctuations in circuit operating temperature, power supply and even imperfections in the breadboard mean that the operating frequency of this component and the frequency of the display multiplexer interact in different ways every time I turn on the circuit, creating a variety of stationary and moving lissajous figures.

Ultimately though, the bat draws a solid enough line that it is acceptable for this system.

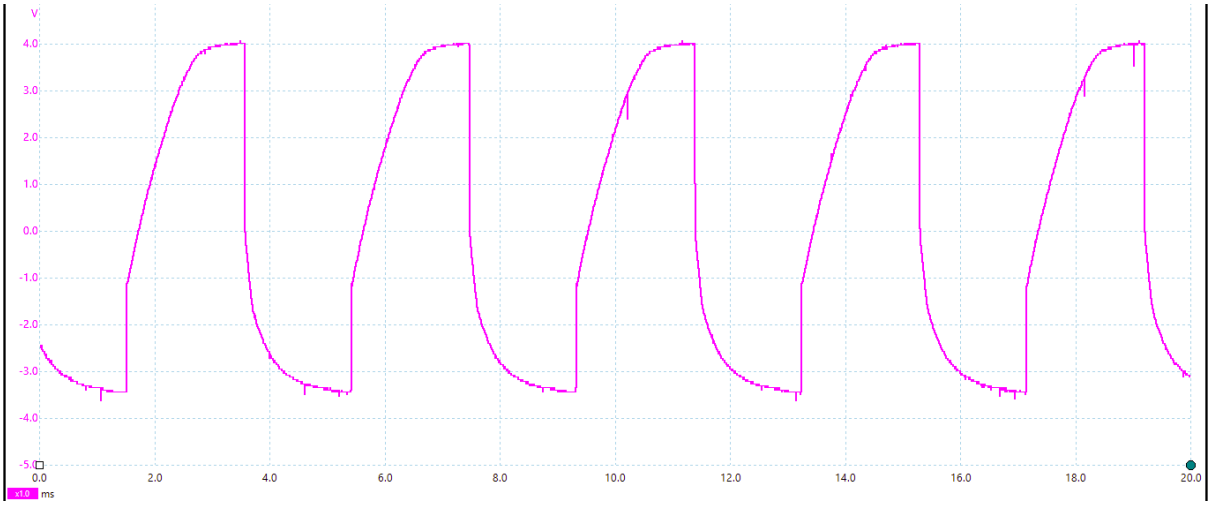


✔ Low Pass Filter (Bat Height):

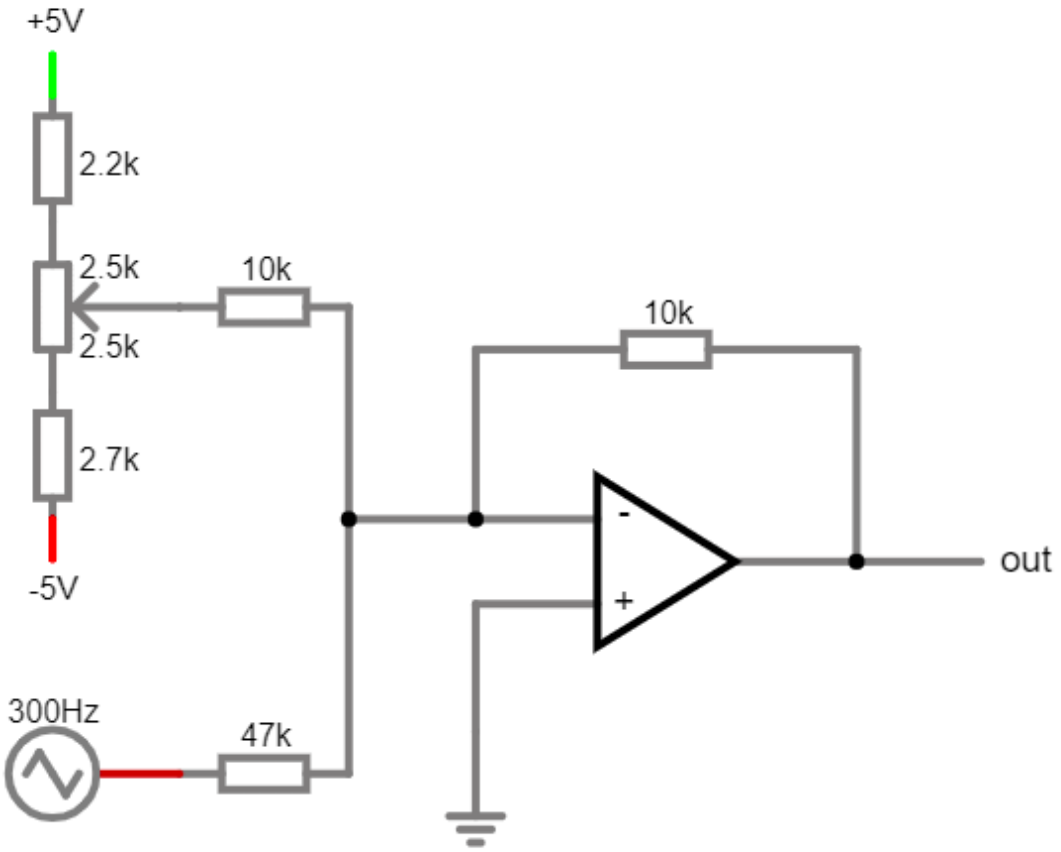
The low-pass filter added after the square wave generator was introduced to slow the movement of the scope between the two extremes of the bat, so the line of the bat was as bright as the endpoints. As it was another visual factor, it did not require much precise calculation to find the right frequency.

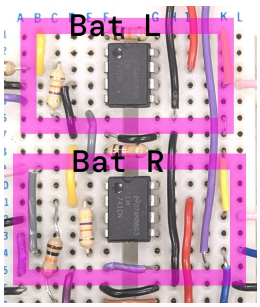


This overall circuit outputs a rough triangle wave at roughly 250Hz.



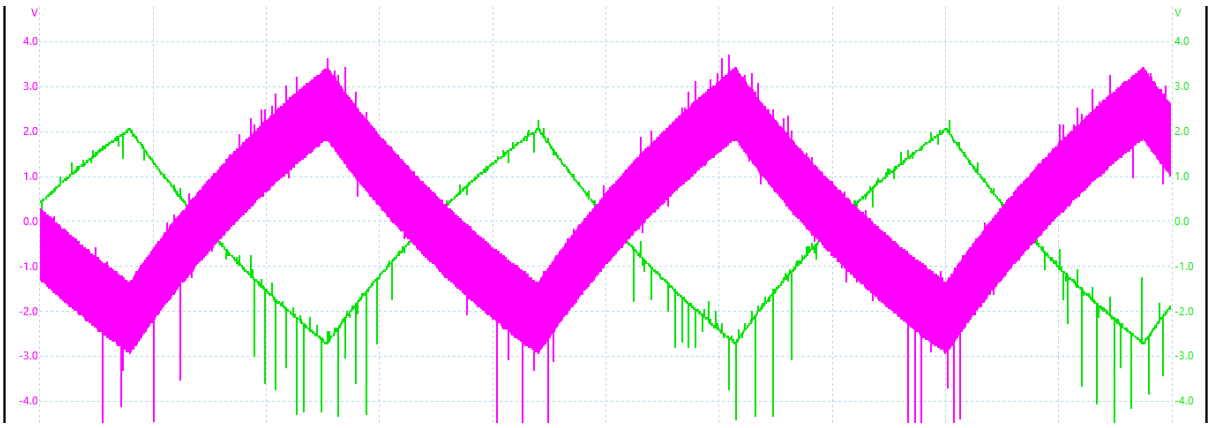
✔ Summing Amplifiers:





I then put my Bat Height signal into a summing amplifier to combine it with the paddle control potentiometer. This creates a line of fixed height that can be moved up and down the Y axis. The amplifier inverts the signal, but this isn't a problem as when we introduce logic later we re-invert the signal with another summing amplifier.

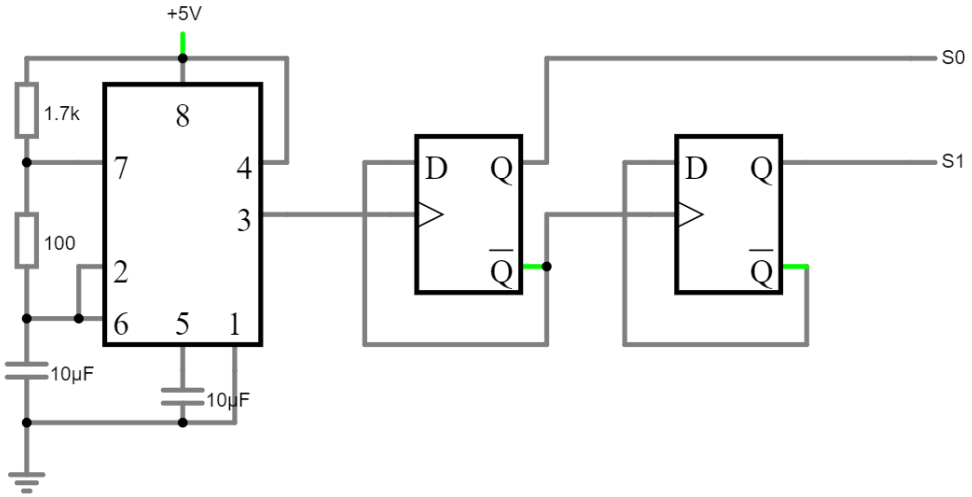
The 47k resistor was selected to take up 2 squares of the 8 that make up the Y-axis of my oscilloscope. This results in a near-perfect 25% height relative to the overall range of the ball's Y position. I selected this value because small bats became hard to maintain a rally, while bigger bats became too easy and you can almost win by accident.

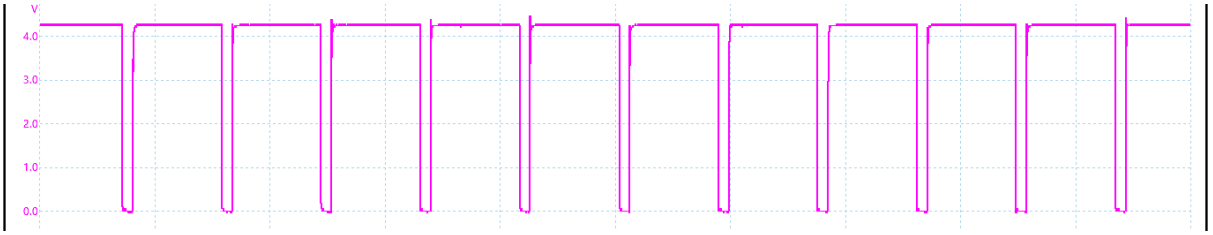


The green signal shows the output of the control potentiometer, while pink shows the output of the summing amplifier. The noise in both the input and output did not have a visible impact on either the drawing or logic circuit so I did not think it was worth applying a low-pass filter to.

✔ Multiplexer and Control Clock:

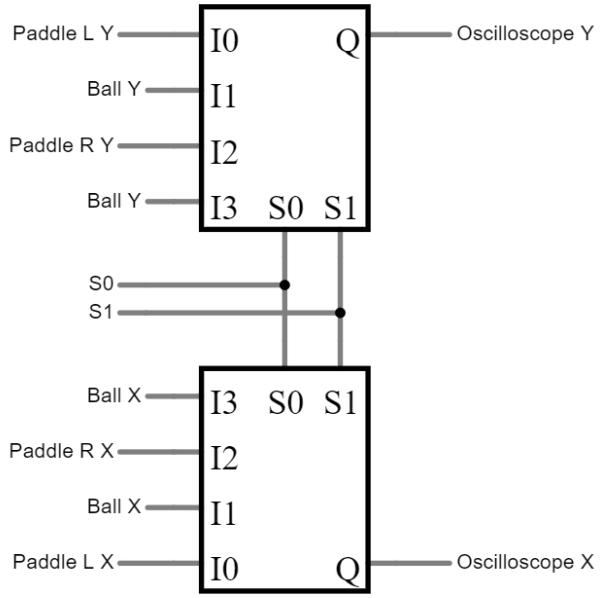
The two multiplexers are controlled by a very high-frequency 555 connected through two D-type flip flops to produce a 4 state cycle.

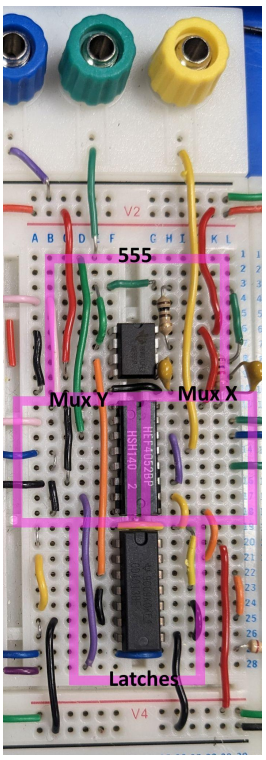


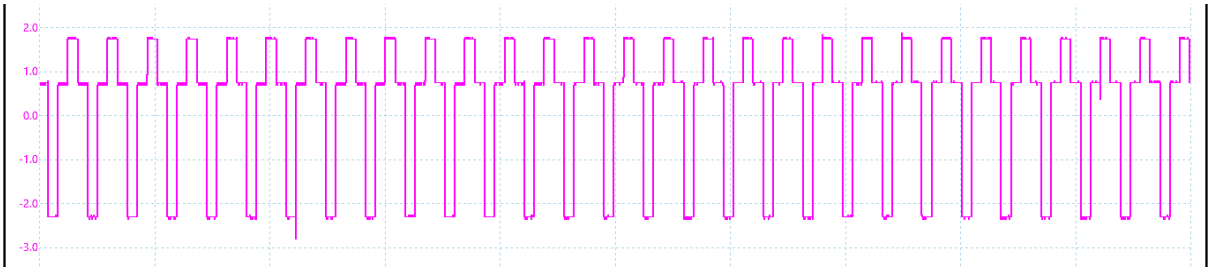


The 555 outputs a signal of 11.6kHz, meaning a full cycle of all inputs takes (1/11.6Khz) * 4 = 350 µs.

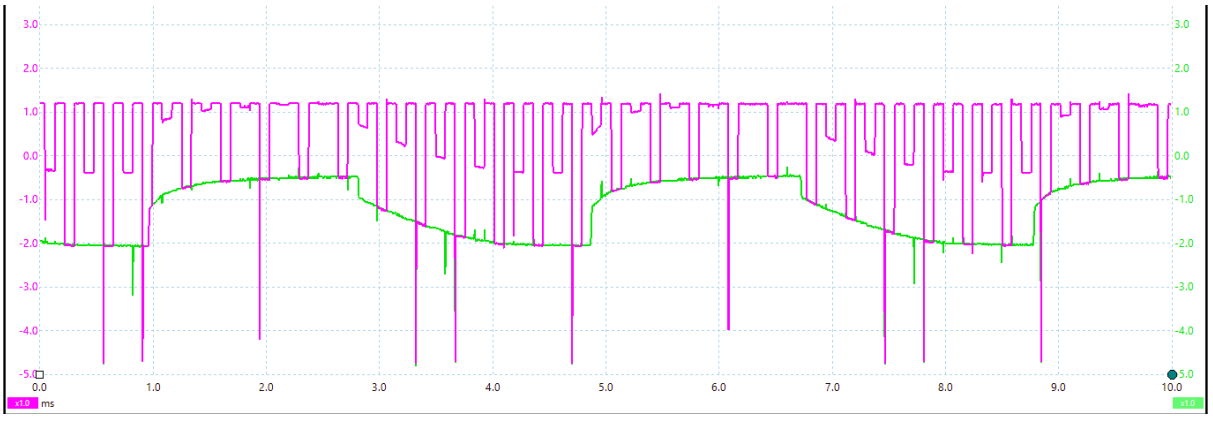
Because there are only 3 signals to draw on the screen, the multiplexers cycle in the order Left Paddle > Ball > Right Paddle > Ball to make sure the ball is bright and clear.





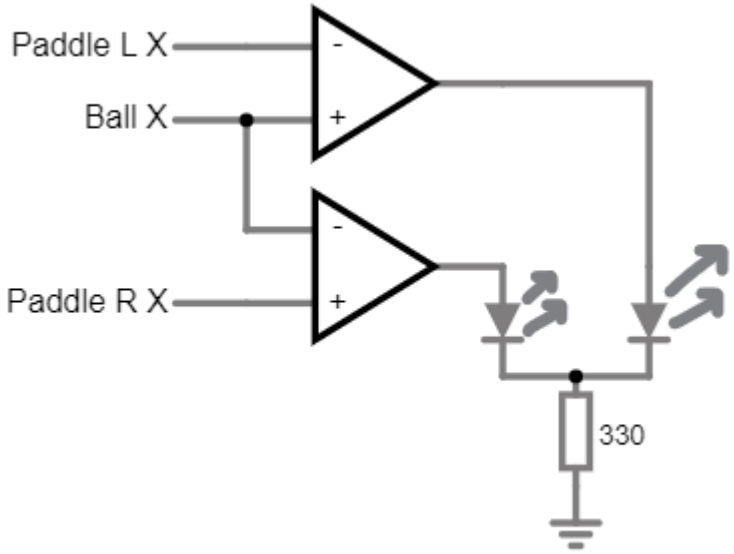


This is the output from the X Multiplexer - the high and low pulses are the bats while the signal in the middle is the ball moving left and right. Because the multiplexer changes so frequently the time scale of this scope output is so small the ball looks like it is not moving.

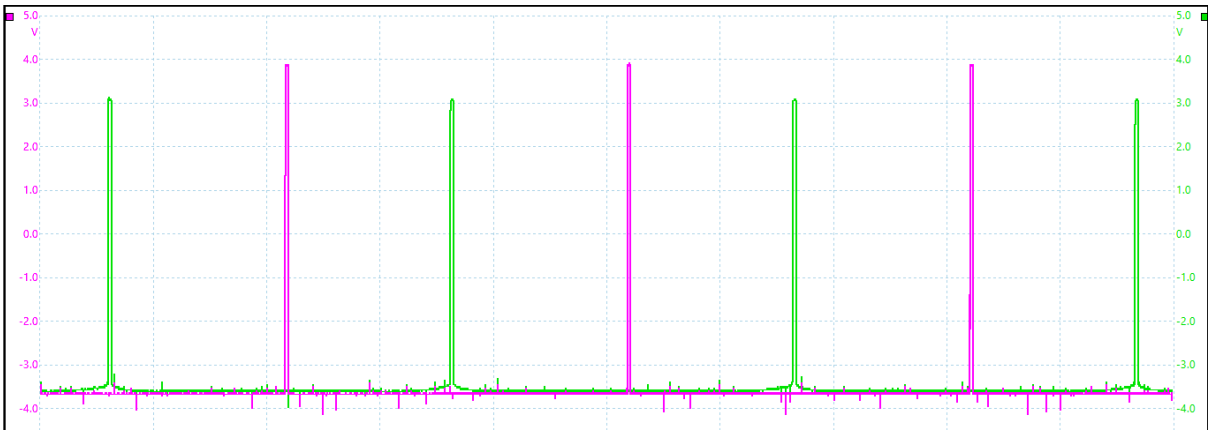


The Y Multiplexer output is more complex as all components in it are in motion. To demonstrate how it maps to its inputs, the green signal shows one of the bat drawing signals. You can see the shape of the signal of the other bat above it, with the ball position forming the hard edge at the top of the signal.

✔ Ball Sides Comparators:



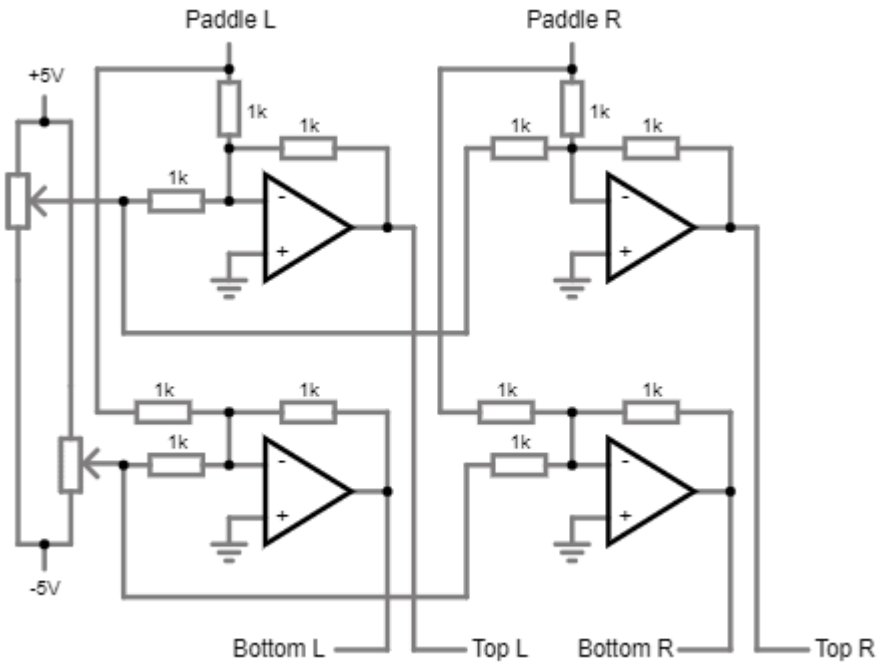
The LEDs are very useful for tuning the values of Paddle L X and Paddle R X, as the human brain is good at recognizing periods of time in the blinking lights as a representation of how close the logical bounds of the screen are to the bounds of the ball's X position. Of course, the real reason for this subsystem isn't for LEDs but for connecting up to logic later.



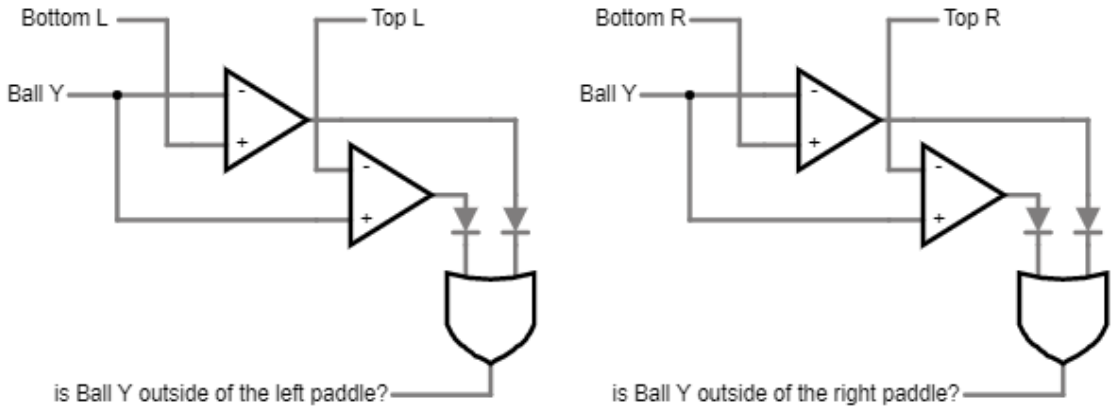
This is the output of the two comparators as the ball bounces from side to side.

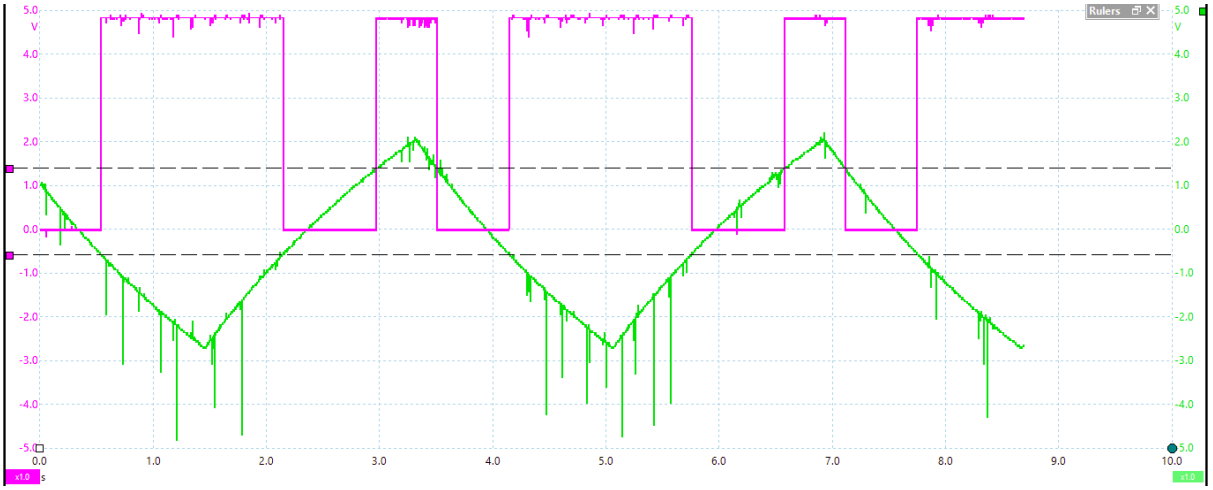
✔ Ball Height Comparators:

If ball hit sides, if ball above top of bat or below bottom of bat, send signal to reset
First we need to construct the 4 reference voltages for the top and bottom of each bat. To make sure this stays constant relative to the paddle position, I avoided a simple voltage divider to bias the signals as this would fail to buffer the original input signals and instead used a summing amp of gain -1. This also served to invert the voltages to stay in parity with the inverted signal used in the drawing circuit.



Then these reference points are compared with Ball Y to check if the ball would bounce off the paddle (ignoring the current Ball X position).

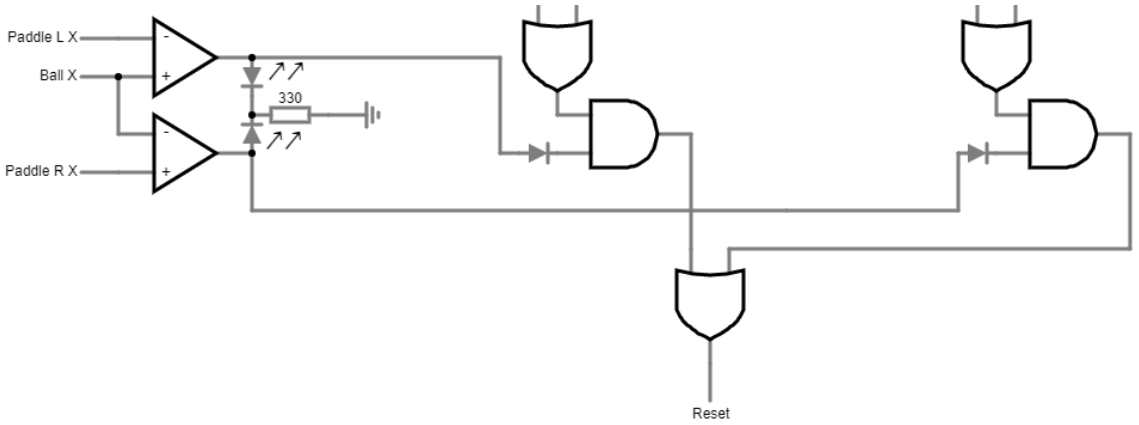




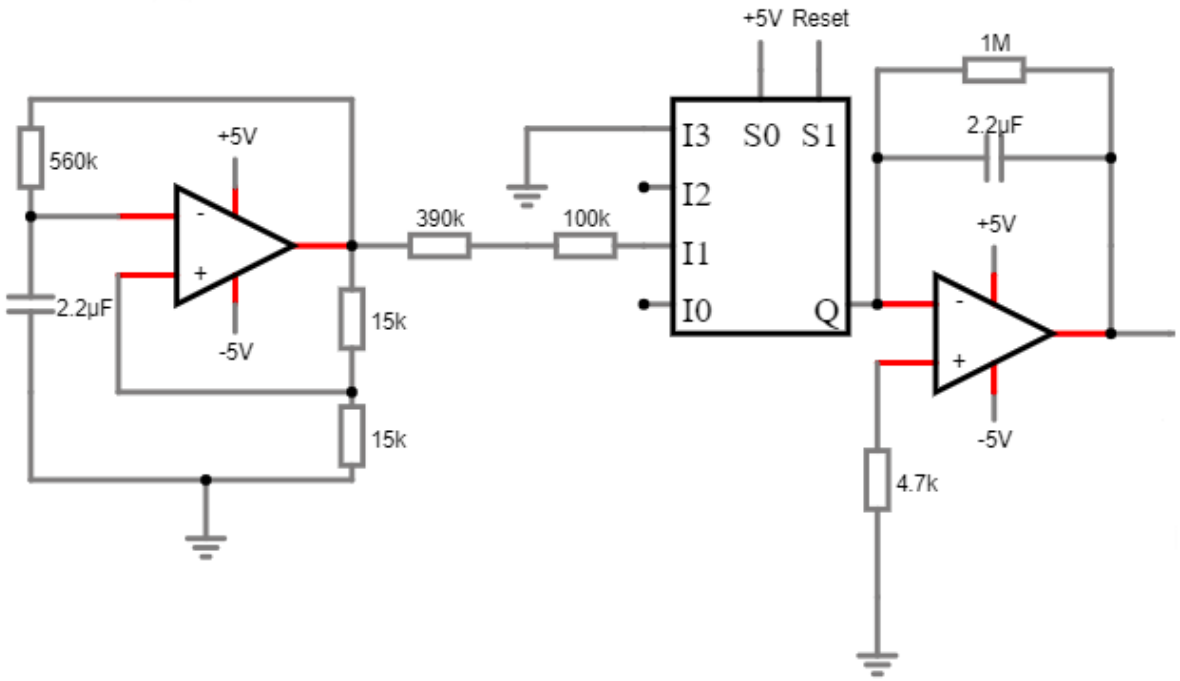
The green signal is the Ball Y position, with the two horizontal lines representing the top and bottom of the paddle. As you can see, the output of the subsystem goes high when the signal is outside the bounds of the paddle.

✔ Ball Reset:

Multiplexer connects the integrator to ground for a fixed amount of time for it to return to the middle of the screen.
Because of the voltage-sensitive nature of the integrator, a circuit involving transistors to link it to ground had too many internal voltage drops to accurately reset to ground while also letting through the square wave unchanged when not resetting the ball. Ultimately, I decided to use an analog multiplexer IC to connect the two, as it does not alter the signal passing through it.

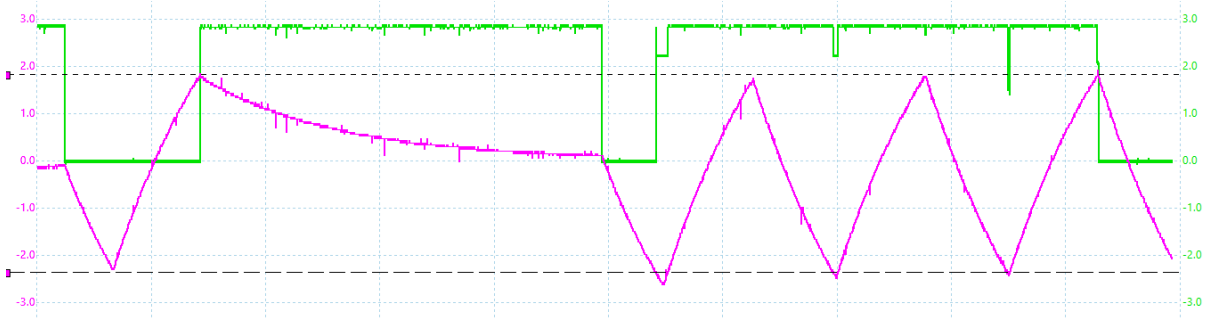


The two previous subsystems are connected through an OR gate and fed into the input for the multiplexer.



This diagram shows the modified Ball X circuit.

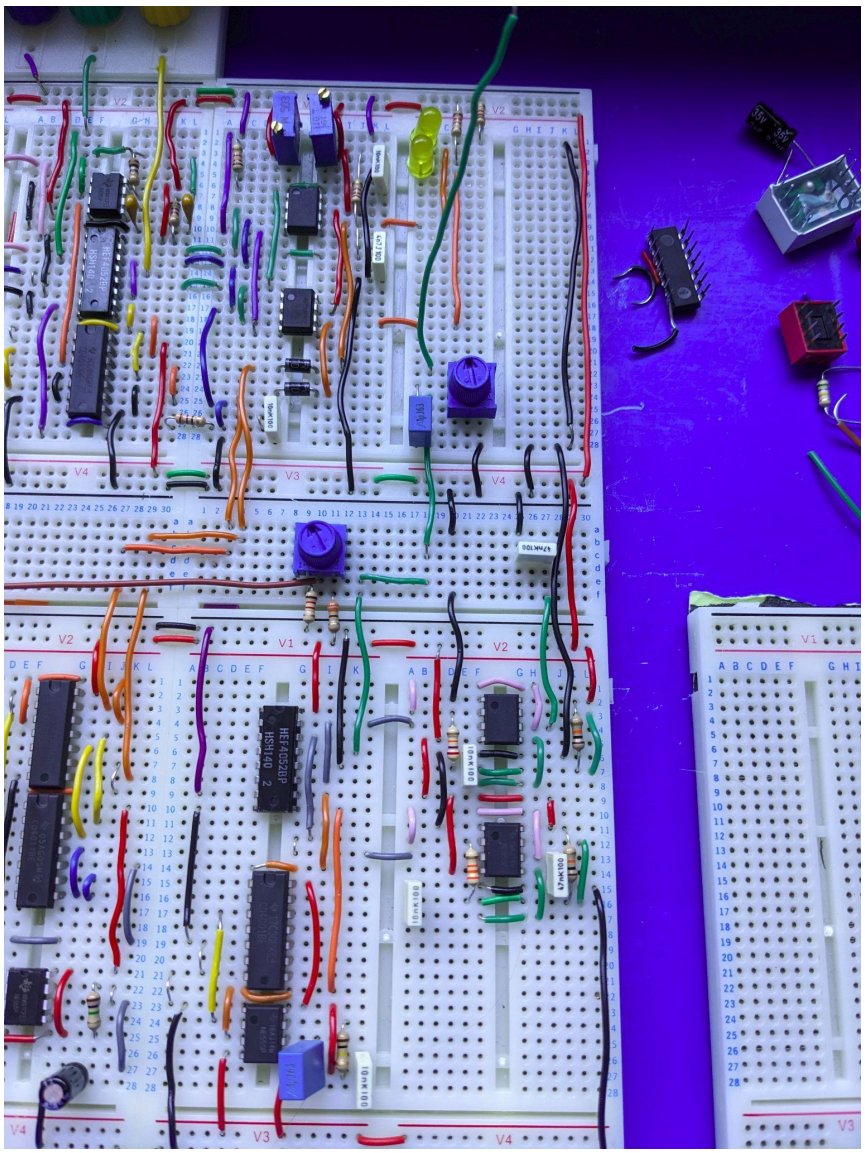
There is an issue with this system of resetting the circuit - only when given an infinite amount of time does the circuit reset to 0. This means when the circuit starts moving again after a reset, the ball's X position will be offset from its intended bouncing pattern. This means one player will have an increased difficulty of hitting the ball, while the other may not have to hit it at all. Luckily, this state only lasts one or two shots after resetting, so it doesn't have any major impact on the game as a whole.



The green signal is the reset (only the first pulse, as after this I disconnected it from the circuit to test the decay in the voltage offset over time. As a result the second green pulse has no effect on the pink wave)
As you can see the pink signal which is the Ball X Position overshoots its left-most position and does not reach its right-most position on the first cycle after a reset. This manifests in the left player having a harder shot while the right player may not have to hit the ball at all. These bounds return to normal for the next cycle, so this only affects the two shots following a reset.

✘ Speaker Output (Beeps and Boops):

This subsystem would have consisted of two different 555s generating tones, fed into a multiplexer and triggered by a monostable when certain conditions were met. This output was then fed through a low-pass filter to make the sound less harsh.

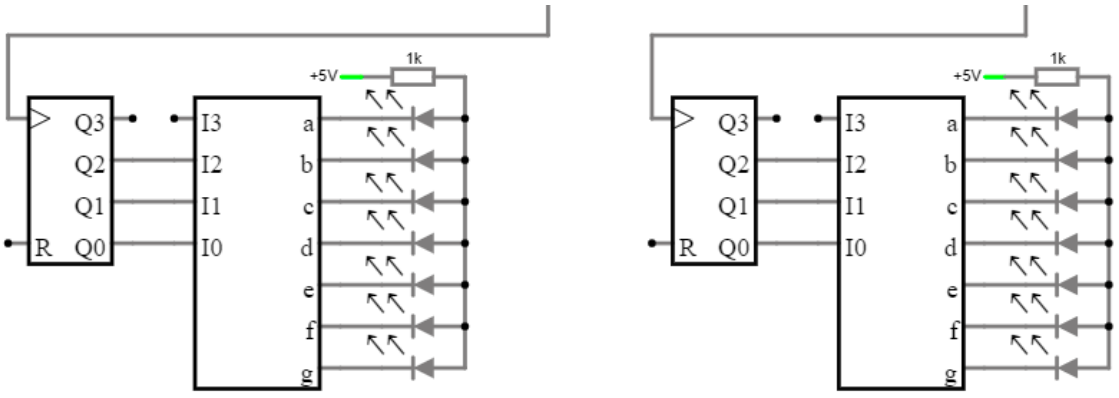


The sound output from this subsystem was very fitting and the low-pass filter successfully stopped it from being too harsh. However, when assembled, the large power draw caused by the speaker created interference in the voltage-sensitive logic circuit that totally broke the functionality of the game. I sadly had a lack of time to test potential solutions (e.g. isolated power supplies) so I scrapped the audio circuit and replaced it with a bar-graph score display.

✘ Counter Logic:

I considered a few potential solutions for the counter circuit. I was originally going to use a 4017 IC as it can drive the full 10 LEDs and operates as a built-in counter. However, its pin layout is chaotic and unordered, and as a result I would be unable to wire it neatly to my LED bar. I instead selected the 74138 IC, a binary decoder chip that has 8 outputs in a nice order. While this did require an additional binary counter IC, it enabled me to make my circuit much simpler and more compact.

However, this choice to select unknown and not very documented chips did not come without flaws. The binary counter IC, 74LS90, was a BCD counter, while the decoder only accepted inputs in a 3-bit form. As a result I had to wire the 4th bit of the BCD to the reset pins of the chip to make it loop without issues. The 74LS90 also has active high outputs, while the 74138 has active low outputs. This would have been an issue had I used 7-segment LEDs, but as a bar the countdown to the end of the game still works very well as a score indicator.

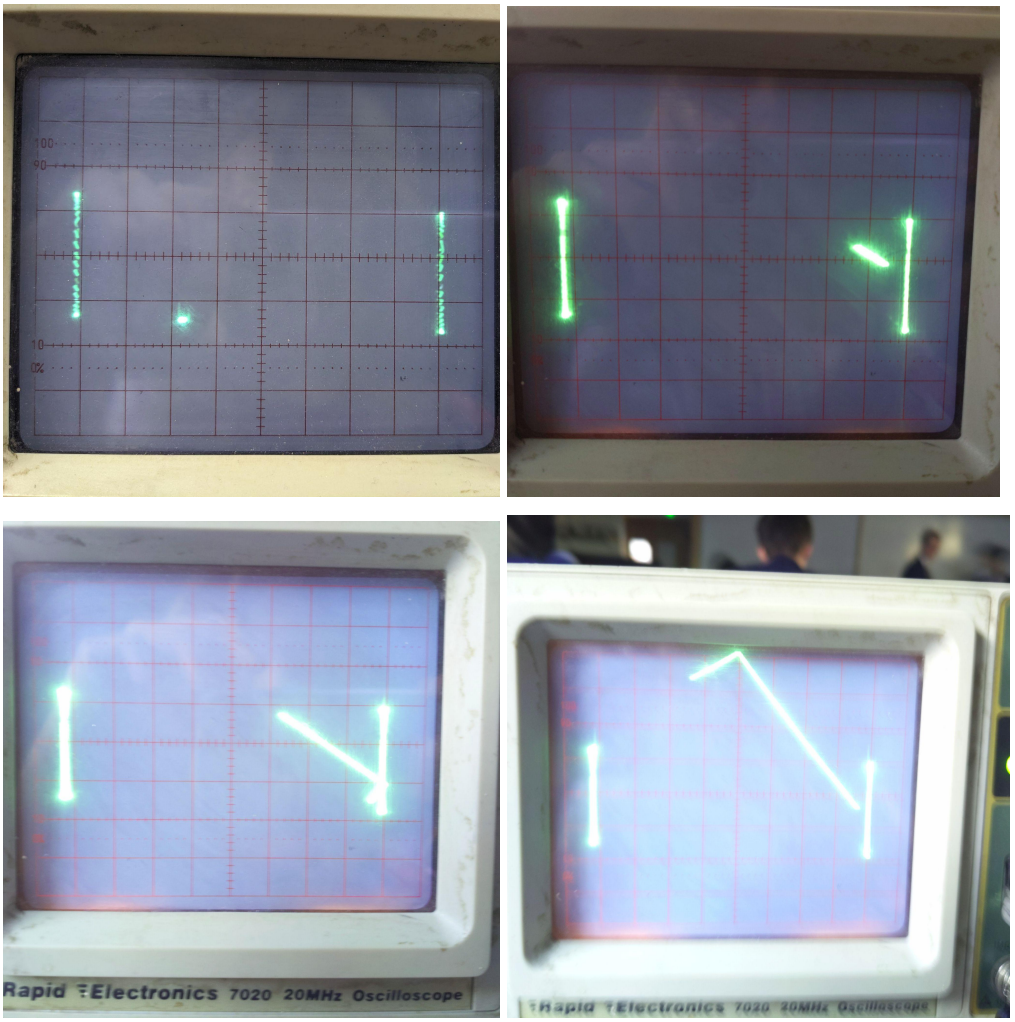


The counters are triggered by the comparator that checks if the ball has hit the left/right of the screen and the ball reset signal put through an AND gate. This method only returns a high signal when the ball hits a side and the opponent fails to hit it, which is exactly what we need while saving on requiring a more complex logic system. However when trying this method out, the counter decreased by many steps at a time. Examining the output reveals the signals from the comparator are very noisy.

This AND gate is actually a NAND gate in the final circuit so to debounce the signal, we need an active low input to connect to a monostable. This creates a short pulse that smooths out all “bouncing” in the signal and ensures the counters only go up one at a time.

Alas I also had to scrap this subsystem as I believe a faulty breadboard connection was causing counters to count up at once. After rebuilding the system a few times using 4017 ICs without success, I ran out of time to get this system working.

Whole System Test



4 photos taken of the system in operation, with increasing levels of exposure to show the ball's path of motion.

The system plays as expected when all subsystems are connected. As it is a game to be displayed on an oscilloscope, there are not many variables to test, but qualitatively it functions as a game well and is fun to play. I intentionally calibrated the bat hitboxes to span further than the bats do visually, as players feel less cheated by a successful yet unexpected bounce, than an unexpected miss caused by a bad collision.

Evaluation

Quantitative variables:

- Ball bounce frequency between 0.5 and 1 Hz: the time between bounces is 1.6s, making the frequency 0.625Hz, falling within specifications.
- Screen draw frequency faster than 30Hz to avoid flicker: The screen is drawn at a frequency of 2.9kHz, far beyond the goals set at the start of the project
- Bat height of within 20% and 40% of the screen height: the bat takes up three of the 8 squares that make up the oscilloscopes height, placing it at just under 37% of the screen height.
- The system must take 2 or less cycles to return to normal operation after a reset: as shown in the Ball Reset section it manages to return to normal after just one cycle, exceeding the specification.

Qualitative variables:

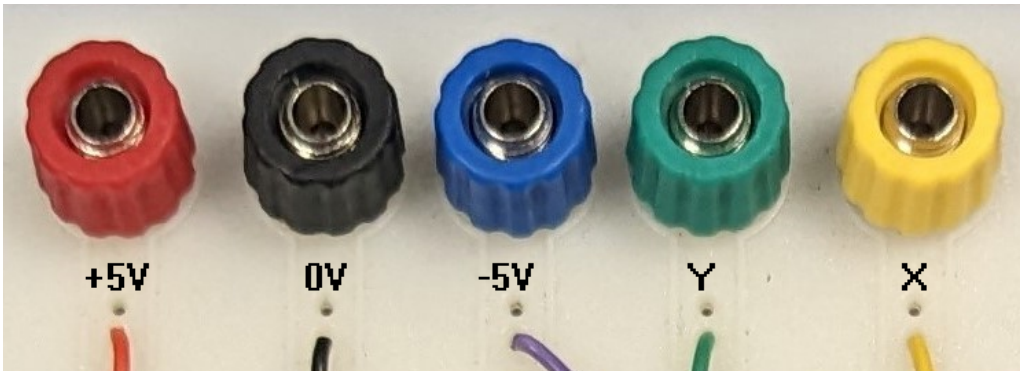
- Responsive controls: the game feels easy and natural to play. The potentiometers available to me do limit how satisfying the gameplay actually is, but they work well enough.
- Somewhat accurate motion paths: the ball moves with no visible speed-up or slow-down, it fits the movement you would expect from Pong.
- Controls to calibrate the device in case of inconsistent voltage, room temperature etc: I included multiple potentiometers to control thresholds for triggers; this was the best place for them as these triggers need to be incredibly accurate for the game to function as expected, something that could not be achieved consistently with fixed resistors.
- Multiple multiplexed elements drawn on screen at once: Both bats and the ball are drawn on screen at once with no visible signs of multiplexing, successfully giving the illusion of them all appearing at once.

I am very pleased with the results from this circuit. It works almost exactly as intended, with a perfectly playable form of the game Pong to be used on an oscilloscope. I am disappointed that I did not have time to add more superfluous features to the game- such as a counter or audio - but these were not key to the functionality of the game, and where any other subsystem missing the circuit would not function. Therefore, I believe my time was well-allocated into making the core components of the game work.

Were I to rebuild my circuit, I would have allocated more space around the triangle wave generators - it became quickly apparent any of the more interesting features I wanted to add would have to interface with them, such as the reset system. As such, the cramped nature of that area of the board made routing cables awkward. I would also have adopted a latch-based system instead of a square wave to power the integrator - this would let you set an upper and lower voltage for the ball's position that it would always reach

before changing direction - this would enable for a variable ball speed without needing recalibration of screen bounds.

User Guide



Connect +5V, 0V and -5V to the red, black and blue terminals.
Set your oscilloscope to XY mode and connect the scope leads from yellow to ground for the Y-axis, and green to ground for the X-axis.

The two black switches in the middle of the board represent different config options: the one on the left can be toggled to make Player 2's paddle move automatically, while the one on the right can be toggled between normal play and disabling the ball reset (in this mode the ball will bounce forever)

The two blue trimpots at the top of the board are reference voltages for the sides of the screen. If the game feels unfair at any point, it is worth checking these are correctly calibrated. To do so, toggle the right black switch in the middle of the board, and adjust trimpots until both yellow LEDs are blinking on for the same amount of time.

The lower left two trimpots are reference voltages for the top and bottom offset for the hitbox of the bat. By default this is calibrated to be slightly bigger than the visual bat itself but the user may wish to make the game harder by making it smaller. There is no easy way to calibrate this system without a picoscope, as you need to compare the output of one of the summing amps the trimpot feeds into with the output of the bat's drawing signal.

Lastly and most importantly, once everything is set up, the two players use the large blue potentiometers to control their paddles. Turning the pot moves the paddle up and down, with the aim of deflecting the ball bouncing around the screen away from your side. If a player fails to do this, the ball will slow down to roughly half speed and gradually approach the centre of the screen, before play starts again.