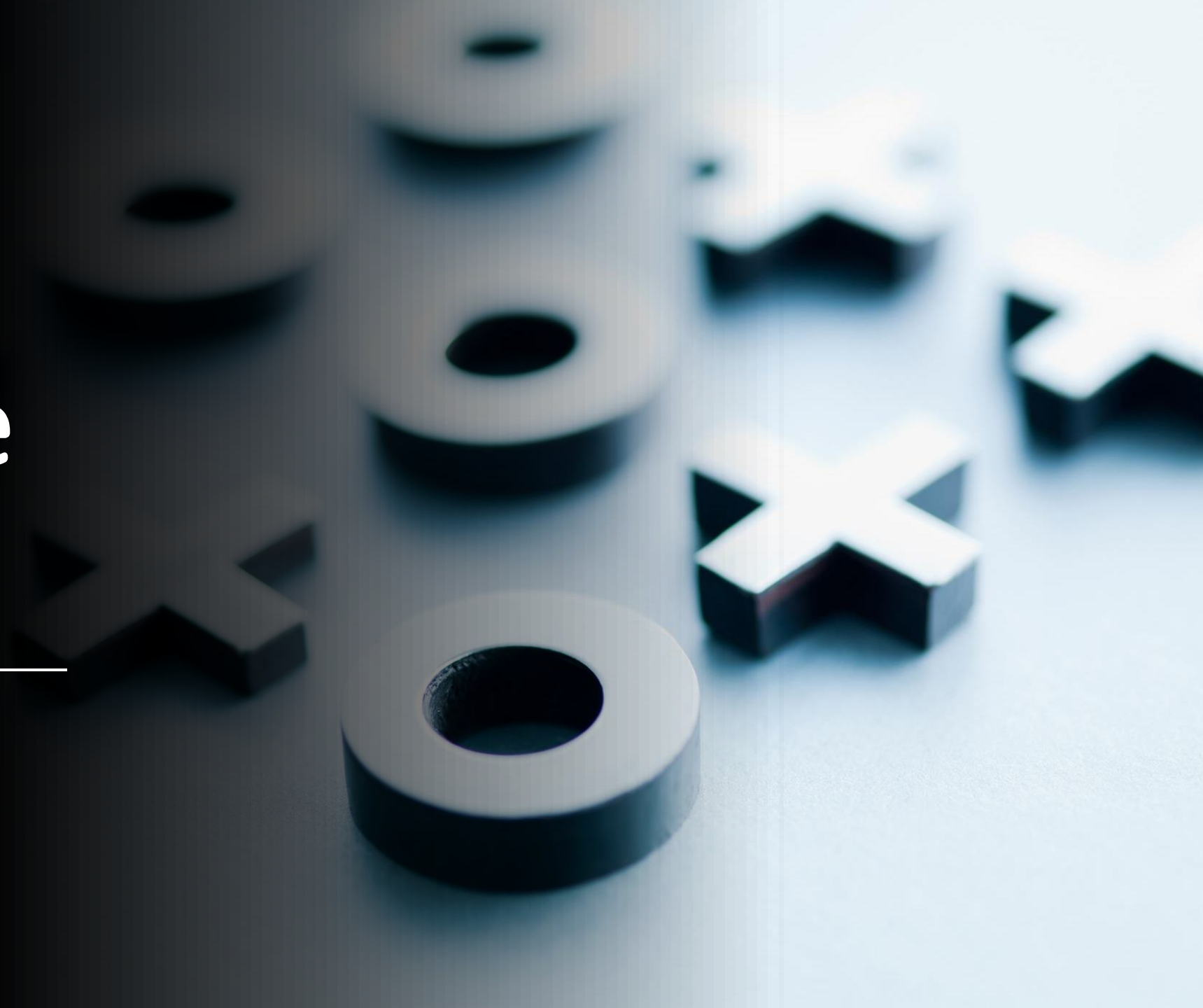


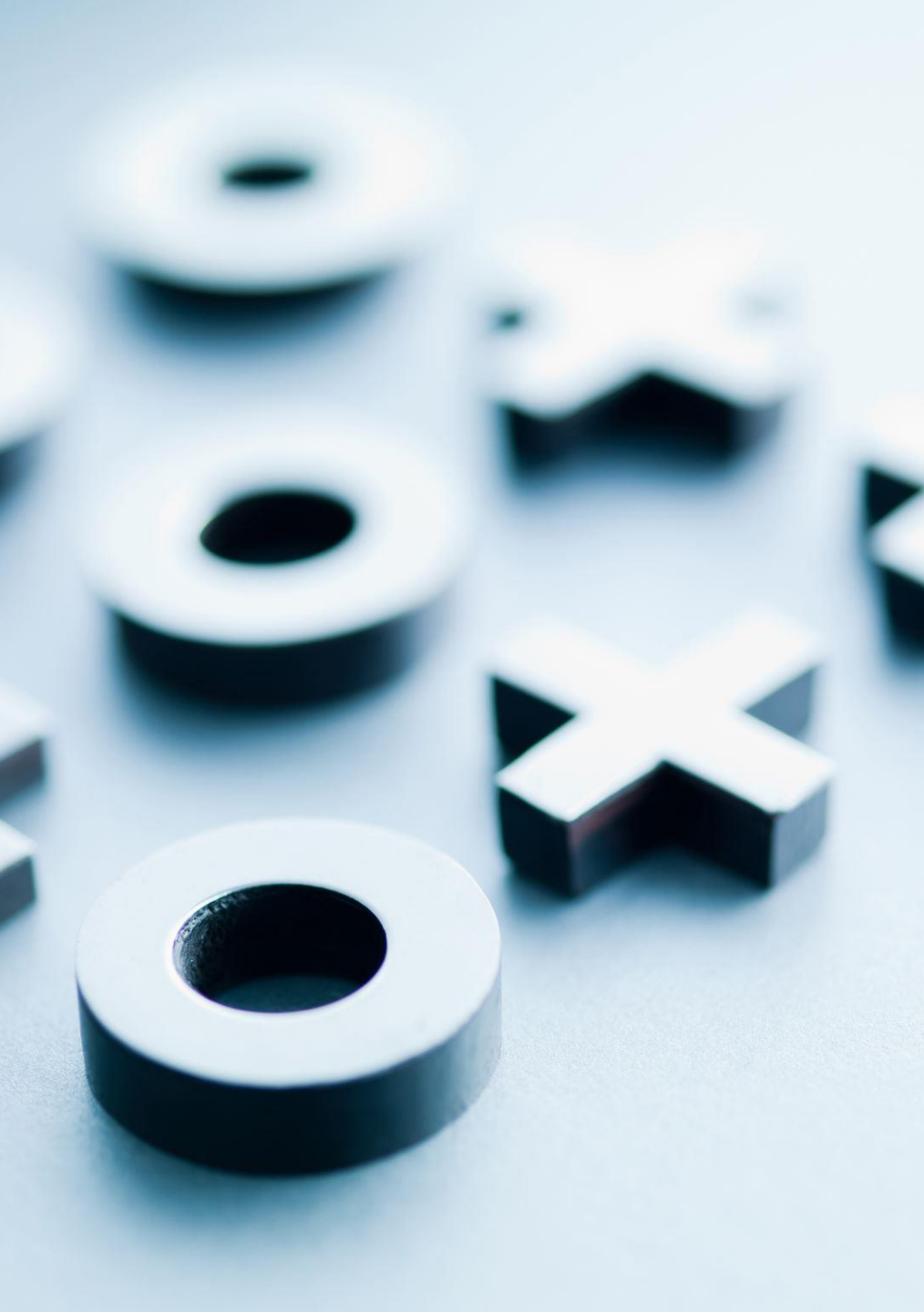


# Tic Tac Toe (Arduino)

---

By: [REDACTED]





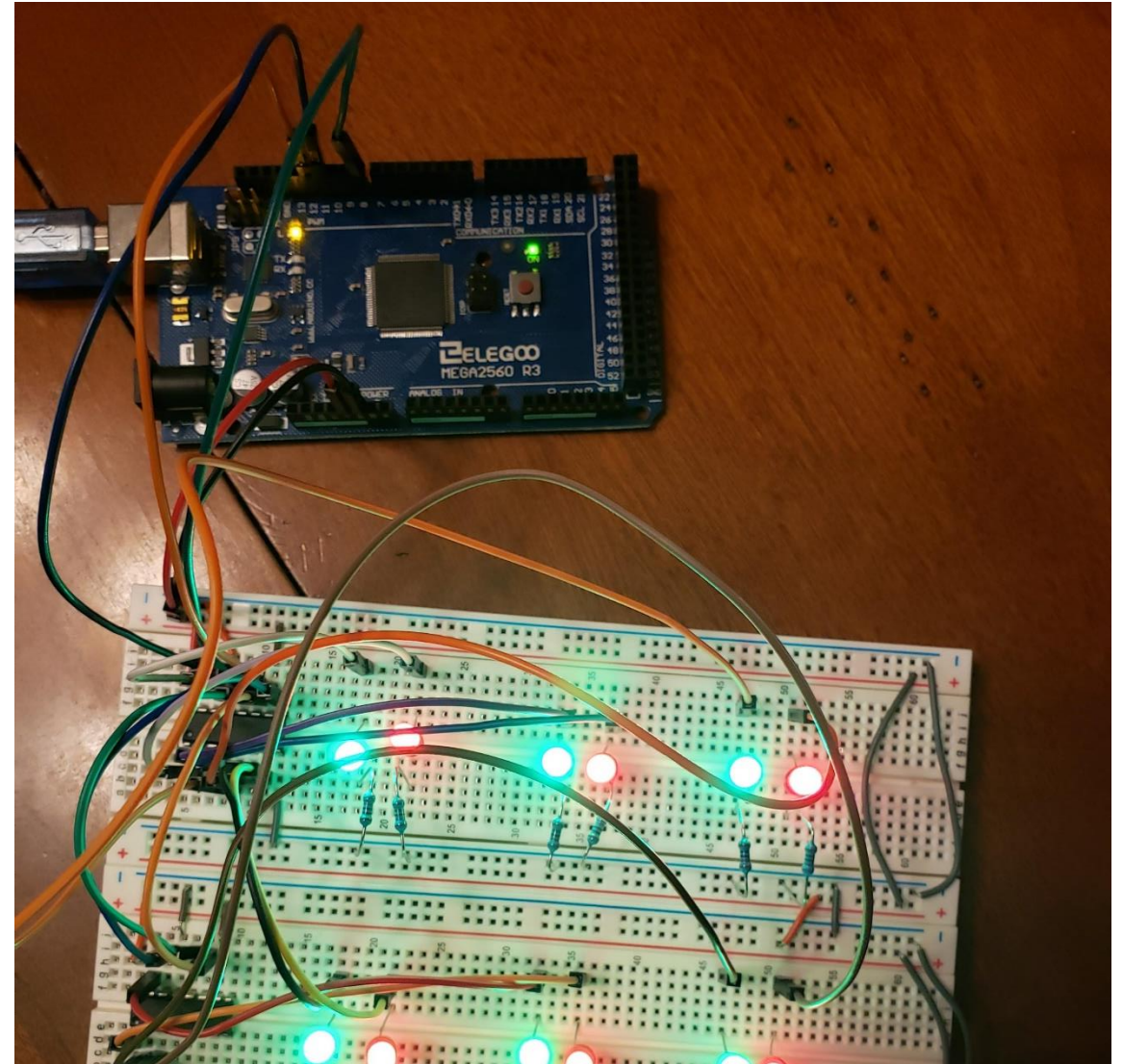
# Our Invention

After brainstorming all the possible machines that could be made, we determined that we were going to create something that was interactive between two users.

We determined that we would make a tic tac toe board, that is controlled by an infrared remote. Instead of X's and O's, we would do two different colored LED lights.

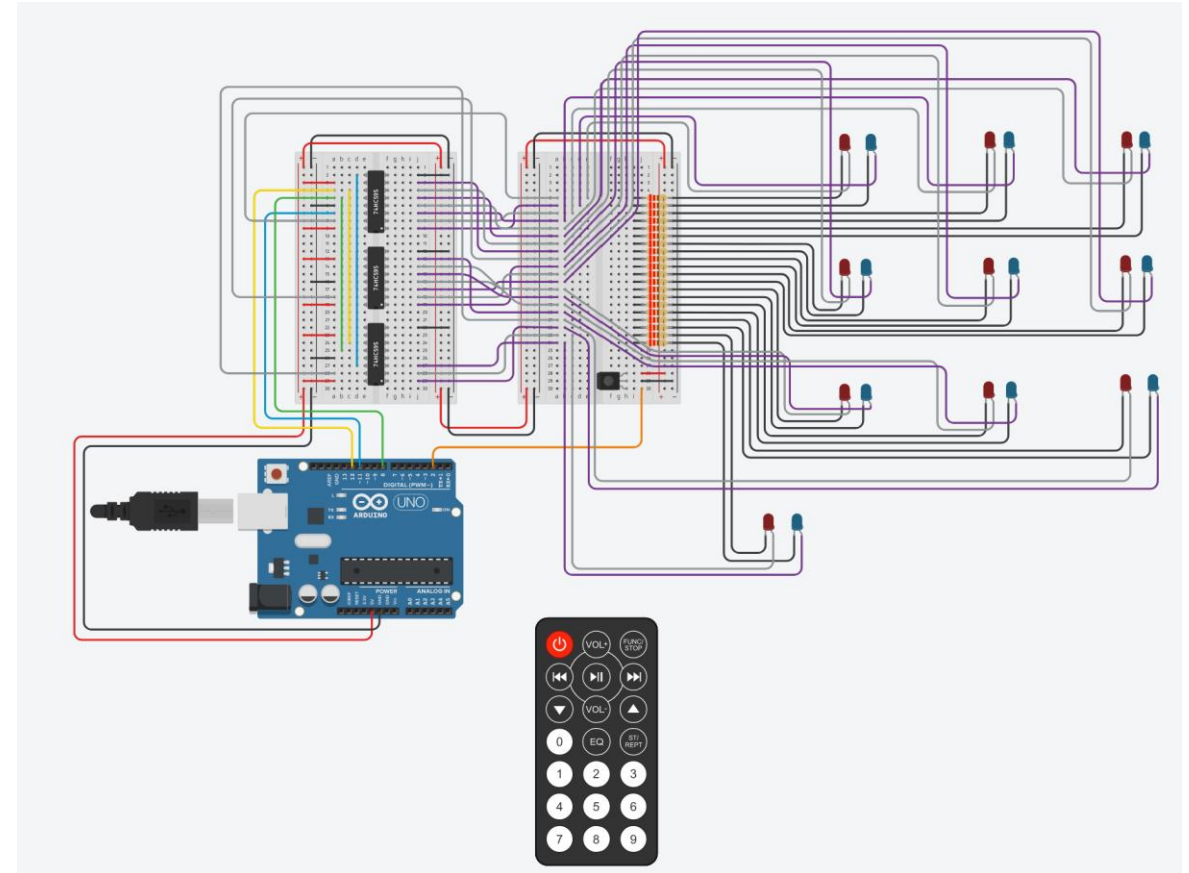
# Prototype

- Starting out working with LED lights, resistors, and shift registers.
- Eventually there will be 20 LED lights.
- Instead of using 20 output pins from the Arduino, we used only 3 outputs into three 74HC595 shift registers.
- These shift registers allow us to convert 3 output pins into theoretically infinite outputs depending on how many 595 chips we use ( $n * 8$  outputs where  $n$  = number of shift registers).



# Tinker CAD

- This Tinker CAD is a virtual version of the prototype.
- Tinker CAD allowed us to take our idea and make a virtual simulation to assist in the real-world build.
- <https://www.tinkercad.com/things/gokXRAUv8Oz-tictactoe/editel?sharecode=MZv9XexcmKgdfNUHSCZAe5Kjxlg9fdEABSqau1aWibA>





# LED Light

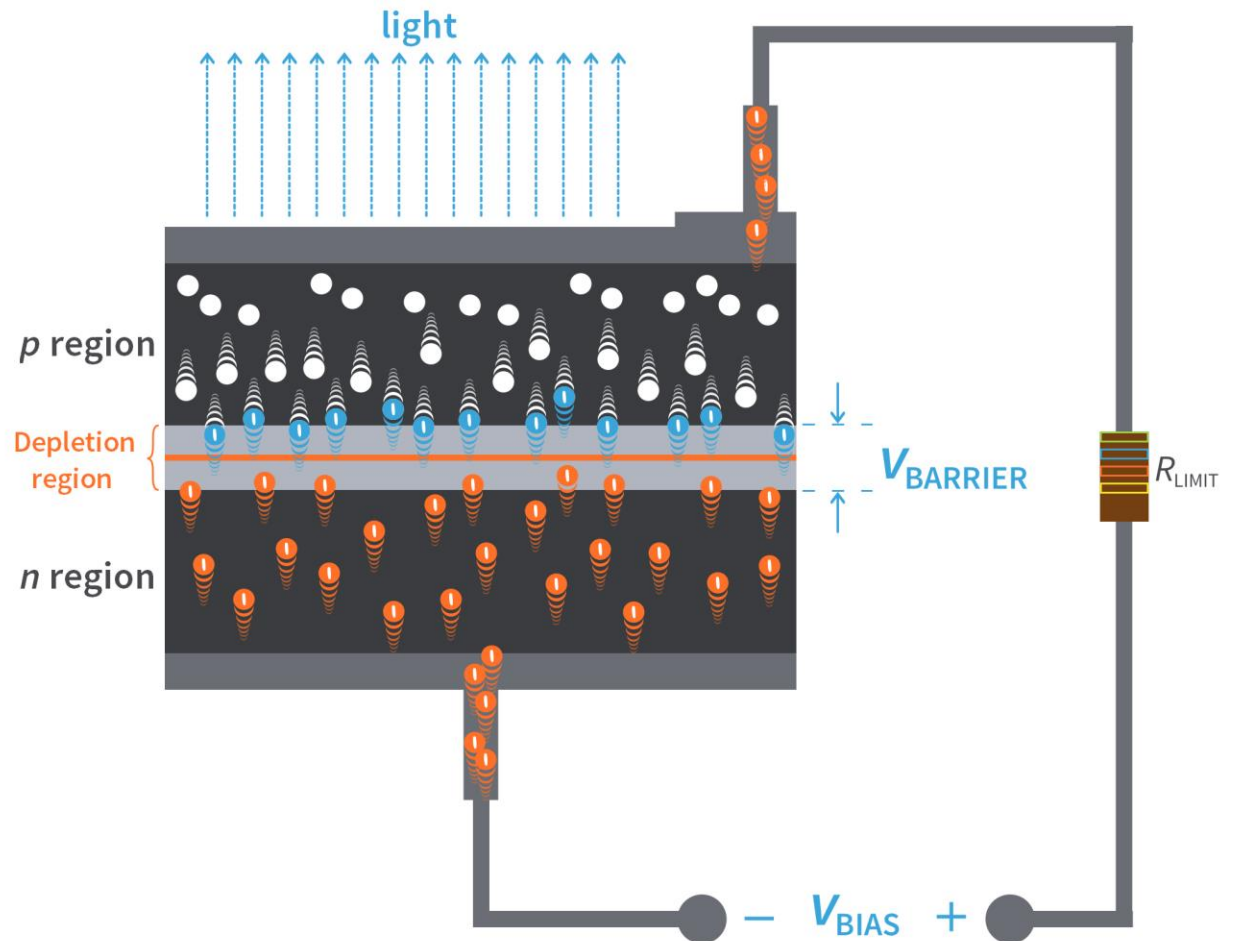
- LED or Light Emitting Diodes.
- A diode is an electrical device that allows the flow of electrons in only one direction.
- A LED has two types of semi-conducting materials in them, p-type and n-type.
- The n-type pulls electrons from the p-type semi-conductor, leaving a gap to fill.
- When a voltage source is applied, the electron “gap” are filled in which results in an emission of light at a certain wavelength.



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

# LED Light

- This picture shows the movement of the electrons between the two semi-conducting “n” and “p” type regions.
- Using different chemical elements inside the regions create different wavelengths.
- The most common elements inside are Gallium and Aluminum mixed with either Phosphorus or Nitrogen.



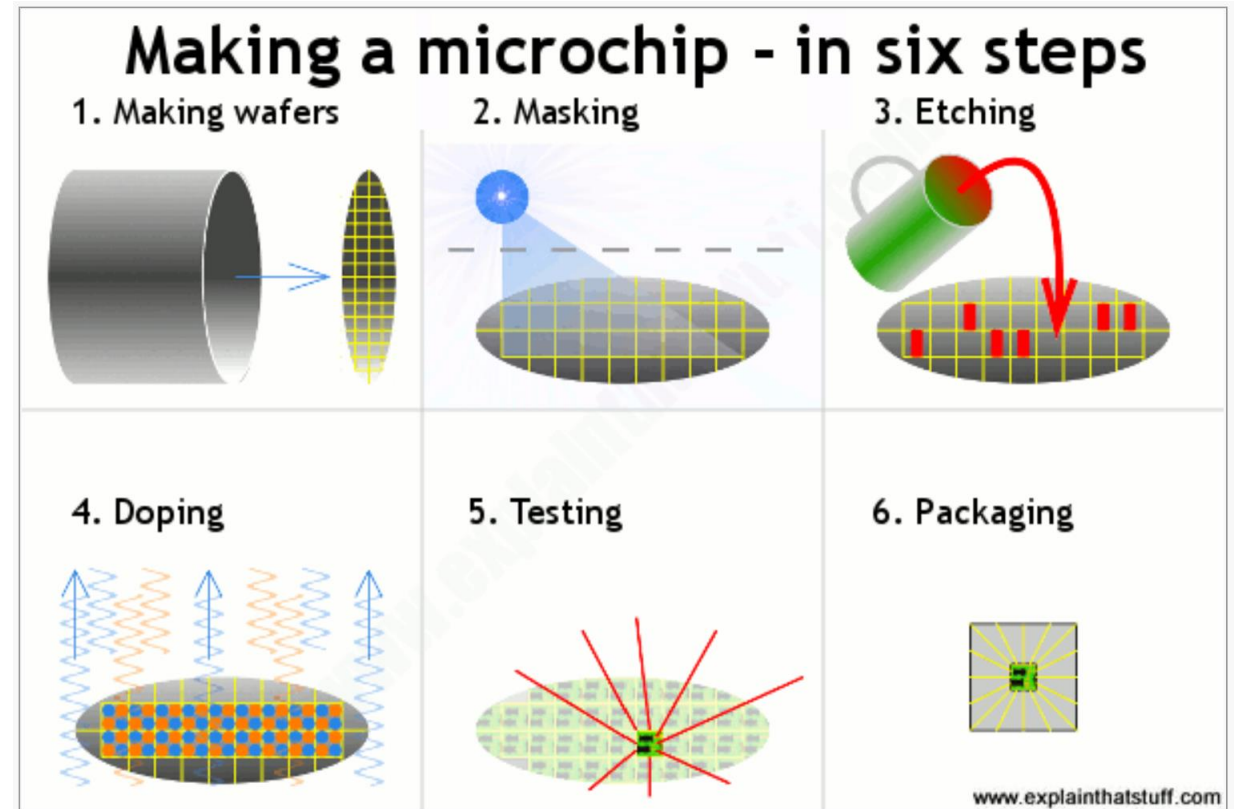
# Shift Register Chip

- Model number 74HC595.
- This integrated chip (IC) allows for a serial input of 8 bits to be converted to 8 parallel outputs.
- Multiple chips can be connected by using the data output from one chip to the input of the next.
- This daisy chain of IC allows many outputs to be controlled through just three pins on an Arduino controller.



# Shift Register Chip

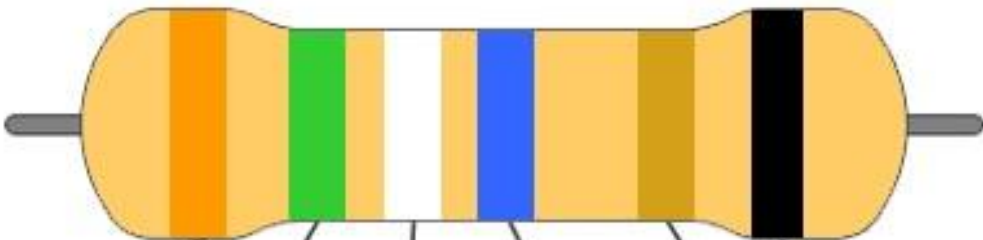
- Much like the LED, ICs use layers of p-type and n-type semi-conductors to store or release electrons depending on voltage applied.
- Silicone that has been baked at high temperatures into a thin plate is the main element of the chip.
- The chip is then etched with a road-like map to allow for the material to be applied.
- A process called “doping” is then added to the etched path to allow for the n-type and p-type areas.



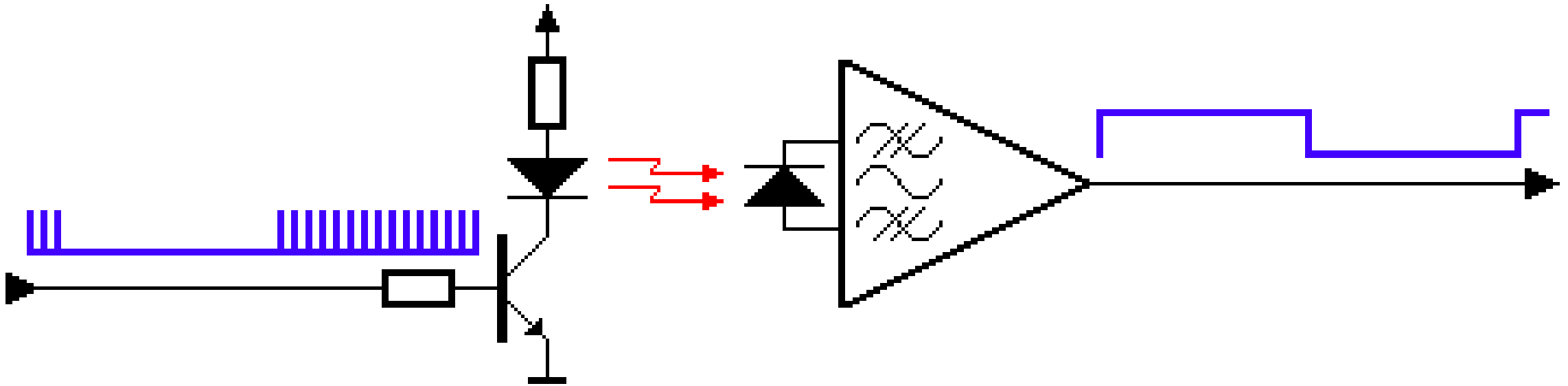


# Simple Resistor

- This project uses twenty 220  $\Omega$  resistors, one for each LED.
- When a voltage is applied across a resistor, the semi-insulating material inside restricts the flow of electrons through the resistor.
- When the current passes through the resistor, electric current is resisted, and the energy is transferred into heat.
- The inexpensive resistors used by lower voltage electronics use a carbon film to resist the flow of electrons.



	1 <sup>st</sup> digit	2 <sup>nd</sup> digit	3 <sup>rd</sup> digit	multiply	tolerance	TCR (ppm/K)
Black	0	0	0	1	1% (F)	100
Brown	1	1	1	10	2% (G)	50
Red	2	2	2	100		15
Orange	3	3	3	1K		25
Yellow	4	4	4	10K		
Green	5	5	5	100K	0.5% (D)	
Blue	6	6	6	1M	0.25% (C)	10
Violet	7	7	7	10M	0.1% (B)	5
Gray	8	8	8	100M	0.05% (A)	
White	9	9	9	1G		
Gold				0.1	5% (J)	
Silver				0.01	10% (K)	
None					20% (M)	

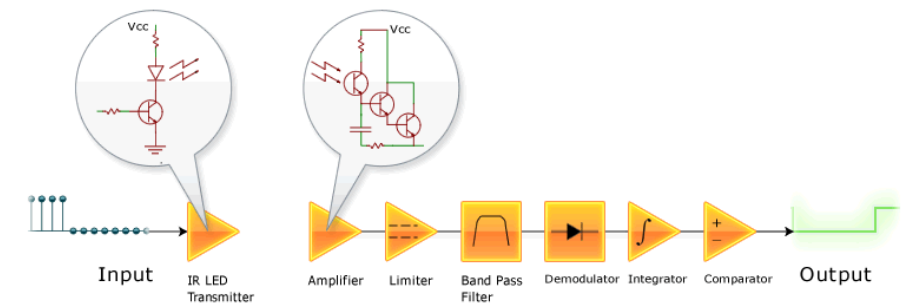


# Infrared Transmitter (TX)

- Uses IR LED, in the 740-760nm wavelength.
- Button is pressed on the remote -> microchip will send a signal corresponding to that button in a pattern of flashes from the IR LED
- Electricity flows from the battery to the IR LED in a high/low pattern -> IR light waves travel in the air.

# Infrared Receiver (RX)

- Photodiode absorbs photons and generate an electric current corresponding to the pattern send by the TX (transmitter)
- They use a series of modulation methods with analog filters to convert the sinusoid carrier signal into the original input signal
- Carrier signal exists on specific frequency to prevent interference from other noise
- Signal is then outputted from the RX, usually as an 8-byte hexadecimal code
- Code in Arduino activates if appropriate hex code is detected



# Piezo Buzzer

---

- Uses a piezoelectric crystal or ceramic
- Voltage applied to buzzer in oscillations
- Results in vibration/stressing of piezoelectric crystal, which produces sound
- The reverse can also be done, and the crystal can be used to detect vibrations and is so sensitive, it can be used as a mic





# Code Flow Chart

**Receive**

Receive input from IR Remote control.

**Process**

Process input, make sure that box is not previously used.

**Turn on LED**

Turn on correct LED light for the player and box selected.

**Check**

Check for a winning Tic-Tac-Toe Pattern, if found show winner.

**Move On**

If no winner, move on to next player. Loop until win or tie.

# Code: Receive IR Input

```
96 #include <IRsmallDecoder.h>           //Needed Libraries for the IR remote
```

For this project, we used a premade IR library that allowed for decoding the signal from the IR Remote.

```
102 const int IR_PIN = 2;                 //Constant for the IR reciever digital pin.
```

This code sets a constant integer named IR\_PIN and set the value to two. Using constants in code allows for better clarity and easier to update one line if something changes.

```
137 IRsmallDecoder irDecoder(IR_PIN);     //IR reciever
138 irSmallD_t irData;
```

```
149 pinMode(IR_PIN, INPUT);               //Set the IR reciever for input.
```

This code creates an IR decoder named “irDecoder” and sets the input pin to our pin constant = 2. The second line creates a variable “irData” that is used to store the received code from the IR receiver. The third line is in the set-up method that tells the Arduino that pin #2 will be used as input.

```
163 void loop() {
164     // put your main code here, to run repeatedly:
165     if (irDecoder.dataAvailable(irData)) { //If a button has been pressed
166         int btnPress = getButtonPressed(irData.cmd); //Variable for the button pressed
```

This code is in the loop method that will constantly run. The “if” statement will run when an IR code has been received. This code is then passed to a method called “getButtonPressed” that will return an integer of the number pressed on the IR Remote.

# Code: Process Input

```
194 //Gets the integer value of the button pressed
195 int getButtonPressed(int data) {
196     int btn = 0;
197     switch (data) { //These are the command inputs from this particular IR remote.
198         //Would need to change possibly if using a different remote.
199         case 12 : //Button 1
200             btn = 1; //Assign the integer to return.
201             break;
202
203         case 24 : //Button 2
204             btn = 2;
205             break;
```

This method named “getButtonPressed” takes the data from the IR decoder and uses a switch/case call to take the decoded data and return an integer value of the number of the button pressed on the IR remote.

# Code: Process Input

```
322 //Checks the space to see if it is filled
323 boolean checkSpace(int num) {
324     num--;
325     if (arrayLED[num * 2] == 0 && arrayLED[(num * 2) + 1] == 0) { //Check the LED for the correct LED for that box.
326         if (currentPlayer == 1) //If it is free and current player is 1, light the LED.
327             arrayLED[num * 2] = 1;
328         else if (currentPlayer == 2) //If it is free and current player is 2, light the LED.
329             arrayLED[(num * 2) + 1] = 1;
330         return false; //Return false to show that that spot was free.
331     }
332     else {
333         return true; //Return true to show that that spot was already filled
334     }
335 } //and another choice must be made.
```

Once the remote data has been processed to find the button pressed by the user, the code calls the method “checkSpace” to make sure that the space selected is available or not.

The code uses a twenty-space integer array named “arrayLED” to keep track of which LED light is on or off using a one for on, and zero for off.

This routine checks the button pressed against the array to find whether that space is currently used. This method returns a Boolean false if the space is available and true if the space is taken.



# Code: Turn on LED

```
380 //Turns on an LED no matter what previous state was.
381 void onLED(int i) {
382     arrayLED[i] = 1;
383     registerWrite();
384 }
385
386 //Turns off an LED no matter what previous state was.
387 void offLED(int i) {
388     arrayLED[i] = 0;
389     registerWrite();
390 }
```

These two methods either turn on or off a specific LED light when an input of integer “i” is given. The “registerWrite” calls a method that sends the data to the shift registers to then light up the correct LEDs.

# Code: Turn on LED

```
420 //This shifts out the LED array into the registers
421 //Then Locks in the output pins (Latch)
422 void registerWrite() {
423
424     //Ready the latch pin to accept new outputs
425     digitalWrite(LATCH_PIN, LOW);
426
427     //Read the array from biggest number first down to 0
428     //This is because the registers shift from largest bit
429     //To the smallest bit (Pushes bit from Right to Left).
430     for (int i = (TOTAL_OUTPUTS - 1); i >= 0; i--) {
431
432         //Read from the LED array to get LED ON = 1 or LED OFF = 0
433         if (arrayLED[i] == 1)
434             digitalWrite(DATA_PIN, HIGH);
435         else
436             digitalWrite(DATA_PIN, LOW);
437
438         //Pulse the shift clock to shift the new bit into the register.
439         digitalWrite(SHIFT_CLK, HIGH);
440         digitalWrite(SHIFT_CLK, LOW);
441     }
442
443     //Latch the output pin to allow the outputs to function.
444     digitalWrite(LATCH_PIN, HIGH);
445 }
```

This method writes the bits to the shift registers and opens the outputs to allow the LEDs to turn on or off.

Shift registers take a serial input of one or zero each time the SHIFT\_CLK on the chip goes from logic LOW to HIGH. (This is called the leading clock edge).

Each time the SHIFT\_CLK is activated within the “for” loop, the bits inside the shift register move one space to the left and the new bit is inserted in the “0” position.

Finally, when all the bits have been received, the LATCH\_PIN is set to logic HIGH and the outputs activate the LEDs.

# Code: Check for a Winner

```
337 //Checks to see if current player wins
338 boolean checkWinning(int player) {
339     player--; //Makes it player 0 or player 1 (just easier programming wise).
340     if (arrayLED[8 + player] == 1){ //Checks middle box first (most possibilities of winning score).
341         if (arrayLED[0 + player] == 1 && arrayLED[16 + player] == 1) { //Diagonal Right
342             winGame(0 + player, 8 + player, 16 + player); //Win game and the LED combo that won.
343             return true;
344         }
345         if (arrayLED[2 + player] == 1 && arrayLED[14 + player] == 1) { //Vertical Middle
346             winGame(2 + player, 8 + player, 14 + player);
347             return true;
348         }
349         if (arrayLED[4 + player] == 1 && arrayLED[12 + player] == 1) { //Diagonal Left
350             winGame(4 + player, 8 + player, 12 + player);
351             return true;
352         }
353     }
```

This code calls a method named “checkWinning” with an integer input of the player number. The code here is only a portion of the full method. Each winning combination must be checked (vertical, horizontal, diagonal) and if a winning combination is found, the method returns a Boolean TRUE, else it returns FALSE. If a winner is determined, the game plays a sound through a buzzer and flashes the winning combination. The “power” button on the remote will reset the game.

# Code: Move On...Repeat

```
262 //Change Player
263 void changePlayer() {
264     if (currentPlayer == 1) {
265         currentPlayer++;           //Change to player 2
266         arrayLED[18] = 0;         //Turn off player 1 LED
267         arrayLED[19] = 1;         //Turn on player 2 LED
268     }
269     else {
270         currentPlayer--;           //Change to player 1
271         arrayLED[18] = 1;         //Turn on player 1 LED
272         arrayLED[19] = 0;         //Turn off player 2 LED
273     }
274 }
```

If no winning combination is found for a player, the “changePlayer” method is called that changes the integer “currentPlayer” to the next player. The method then tells the LED array to switch which LED needs to be lit (1 for on, 0 for off).

The shift register will then be updated to light the correct LED, and the “void Loop” process will start over and wait for a new input from the IR remote receiver. This loop will continue until a winner is found or all spaces have been used (tie).