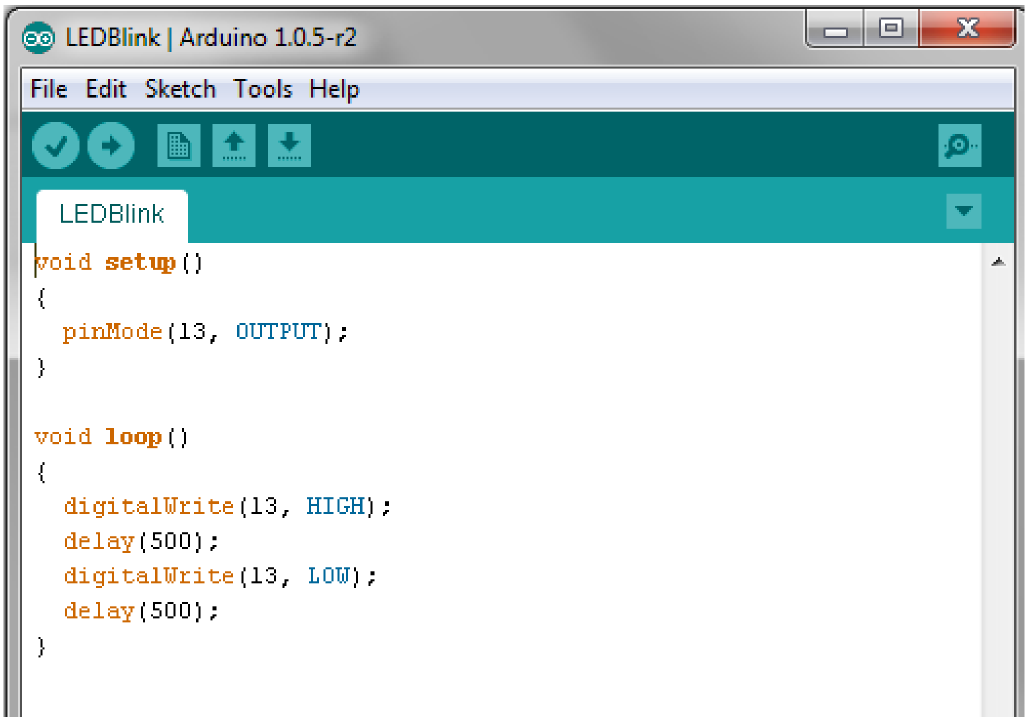
Blink! ...“Hello World” Arduino Style

If you have ever learned a new computer language you'll know that it is traditional for your first program to write “Hello World” on the screen. Now, Arduino boards do not have a screen, but they do have a blinky LED light on board. Our first Arduino program is “Blink” - the “Hello World” program of the Arduino!



```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(500);
  digitalWrite(13, LOW);
  delay(500);
}
```

The **LEDBlink** example sketch switches the Arduino's output pin 13 between its HIGH level (5 volts) and its LOW level (0 Volts). The LED on the Arduino board is connected to this pin which is why we see it switch on and off. Note how we say HIGH and LOW rather than “on” and “off” (More about this later)

Here we use the **pinMode** function to tell the Arduino that we want to use digital pin 13 as an OUTPUT (we can also set it as INPUT)

We put this in the **setup** function of our sketch. Everything in **setup** gets run just one time when the Arduino starts up.

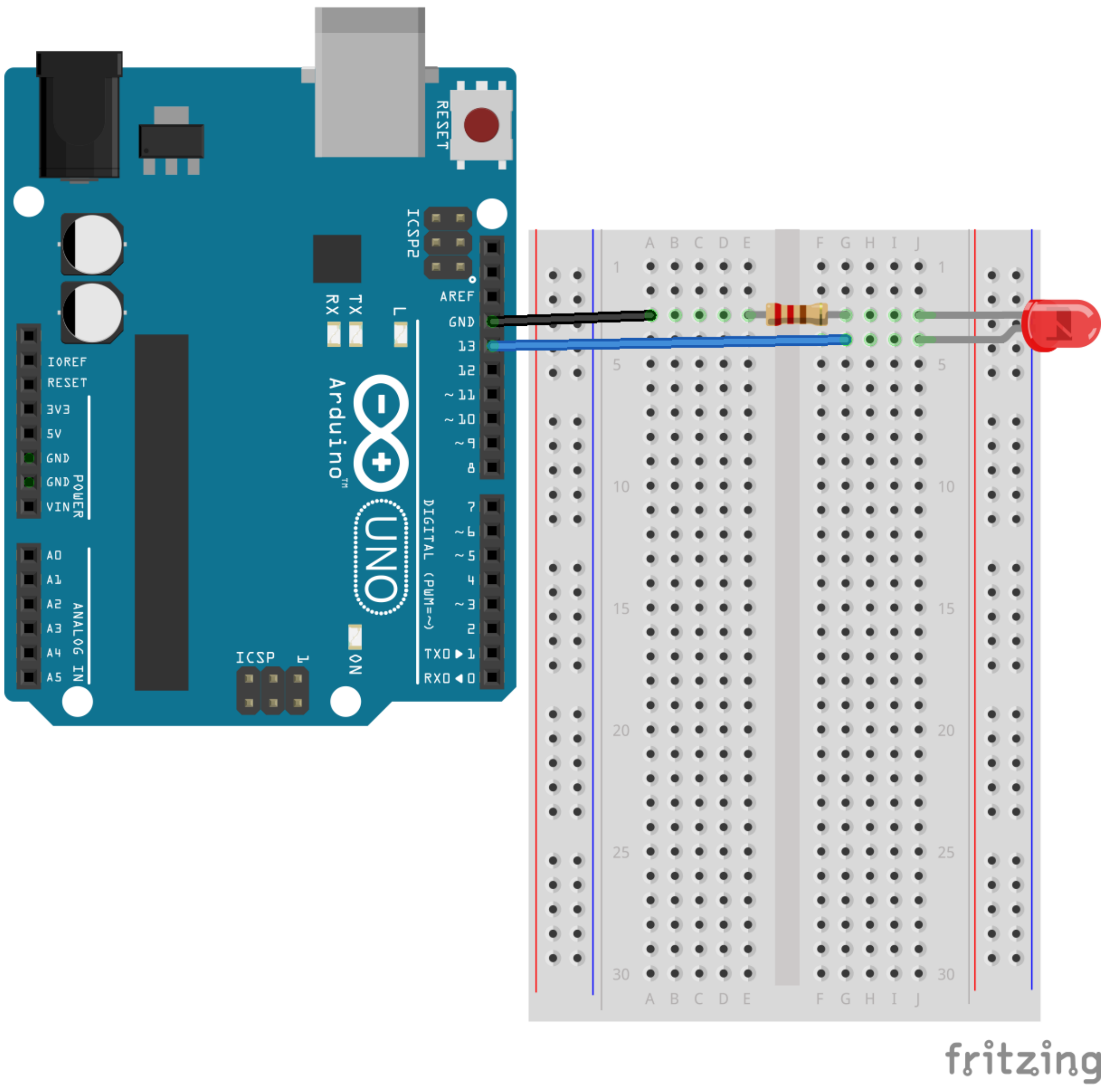
The **loop** function runs over and over after the sketch has started. Here we can the **digitalWrite** function to set a pin to a HIGH level or a LOW level. Without deliberately slowing it down, this would happen many thousands of times a second. This is what the **delay** function is used for. A delay of 500 milliseconds is half a second.

So we set pin 13 to a HIGH level (lighting up the LED), wait for half a second, set it to a LOW level (LED goes off), wait another half a second, then the whole thing repeats. Simples, huh!

Now lets make the Blink example a bit more interesting by making it light up our own LED in addition to the one on the Arduino board

We will costruct the circuit shown below. The stripey thing is a 220 ohm resistor. Be careful to put wires an pins in the same row of holes in the breadboard where shown (these rows of holes are all connected together).

Make sure you put the LED the right way around - the longer leg of the LED is its positive terminal (called the “anode”). This is the leg we connect to the wire that goes to Arduino pin 13. The shorter leg is the negative terminal (called the “cathode”) and this connects to the resistor



When the Blink sketch runs now, we should see our external LED lighting up at the same time as the LED on the Arduino board!

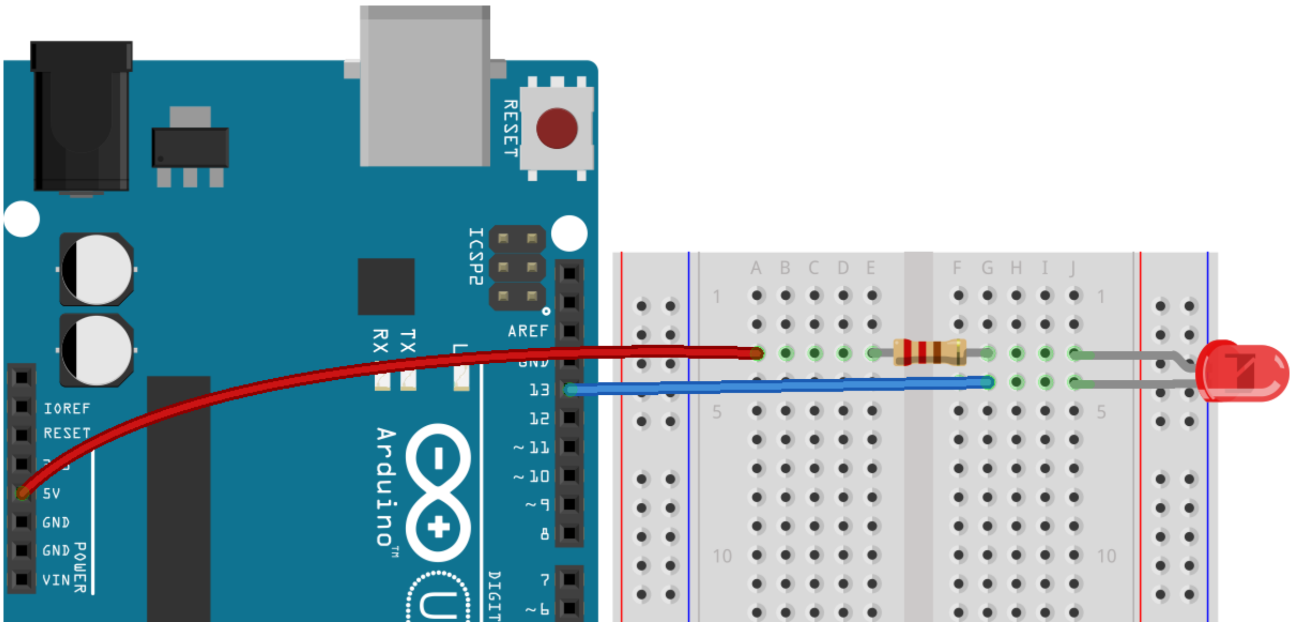
Congratulations on building your first Arduino circuit!

Q: So what is that resistor, and why did we need it?
Well if we put too much electrical current through an LED we can burn it out. The resistor reduces the amount of current that can flow through the circuit, so the LED is comfortable. Whenever you connect an LED to the output of an Arduino you need to remember to include a resistor. Usually we use a resistor value of 220 ohms or more.

The higher the resistance value, the less current can flow and the dimmer the LED will appear. At a certain point there is not enough current for the LED to light at all. Try replacing the 220 ohm resistor with a 1k Ohm resistor, 10k Ohm resistor, 100k Ohm resistor to see what happens.

Q: HIGH/LOW... so what is wrong with plain old "On" and "Off"?
Remember I said we use the terms HIGH and LOW rather than On and Off. Well lets see what this means.

Let's try connecting the LED in reverse - pull out the LED, turn it around and place it back into the breadboard. Remove the Ground connection and connect it to the 5V connection instead.

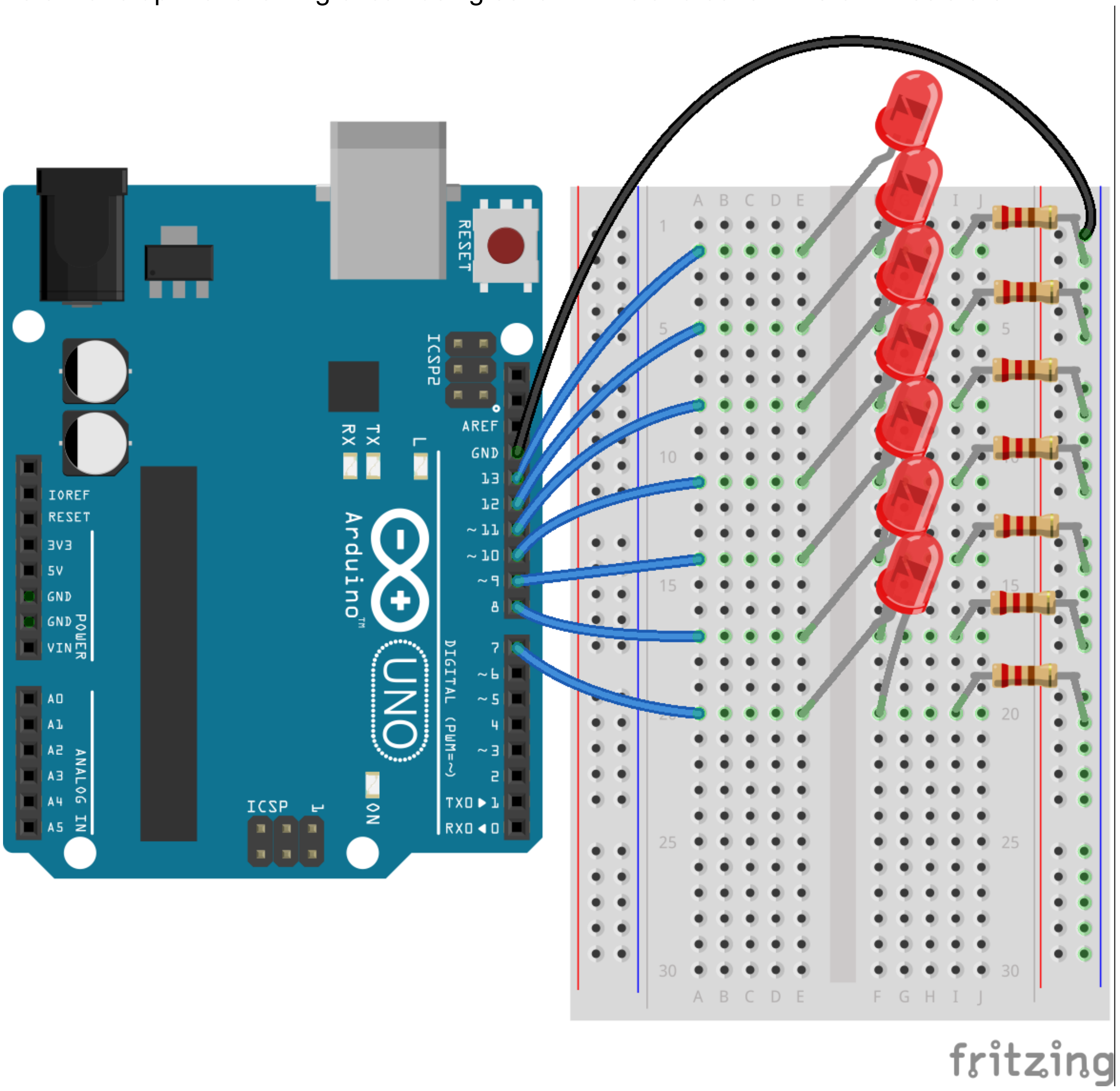


Now you should see the LED on the breadboard lights up when when the LED on the Arduino board is off, so the two LEDs alternate.

So, you see when the output is set LOW it is not "Off".. LOW means the output pin is connected by the Arduino to power supply "ground" where HIGH means the output pin is connected to the positive 5V power supply. A LOW output can still light an LED, so it is not "Off".

The “Larson Scanner”

Lets make up the following circuit using seven LEDs and seven 220 ohm resistors.



Note that all the LEDs have their anode (long leg) to the left side.

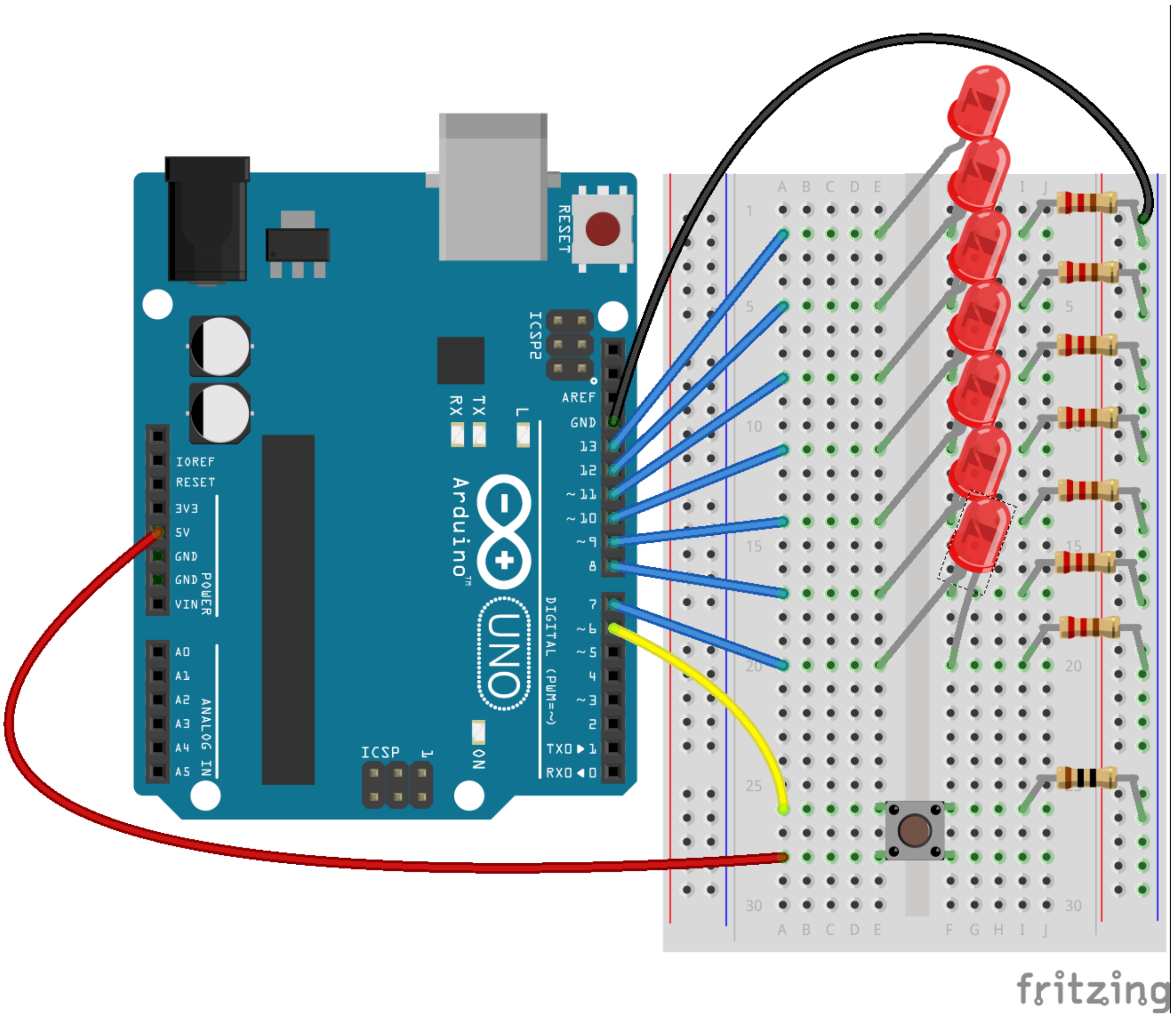
Here we introduce the important concept of the **power rail**. On our breadboard the very outer two columns of pins on each side are connected vertically with each other forming two pairs of “rails”. The intention is that we can use these for power connections (5 Volts and Ground). Many components will need to be connected to power, so having an easy way to provide power along the whole breadboard is very handy.

In this circuit each resistor is connected to the Ground rail and this rail is wired to one of the Arduino's three ground pins (labelled GND). When any of the digital pins 7 through 13 is set to a HIGH level, the associated LED will light up.

The sketches **LEDScanner1** and **LEDScanner2** show how we can light these LEDs in sequence; from one end to the other, then sweeping back and forth.

Adding A Digital Input

Let's add a small switch and a 10k Ohm resistor to our circuit, wired up as shown below (We leave everything in place from the last example)



Load and run the sketch **LEDScannerSwitch1.** You should find that the LEDs “scan” only when the button is held down.

In this sketch, we use **pinMode** to set pin 6 as an INPUT. Then we can use **digitalRead** to see if the input pin is reading a HIGH or a LOW level (Any of our digital pins can be set as an input like this)

When we press the switch down, pin 6 is connected to 5V and reads HIGH so the program runs the scanner.

Q: So what is the 10k resistor for?

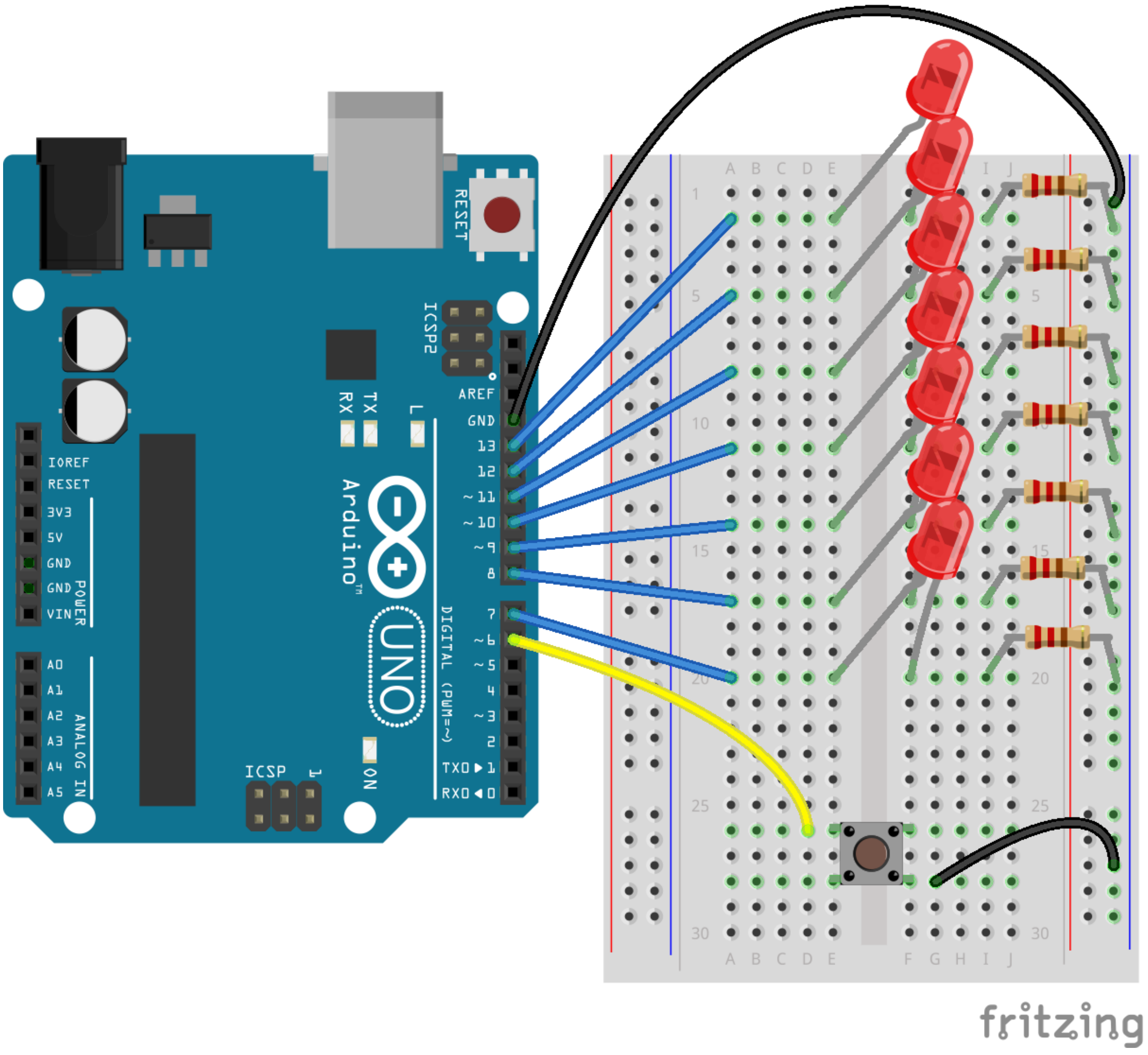
Well, why not remove it to see what happens? If you do this you'll probably find the scanner runs by itself, or maybe runs when your hand is close. Basically the switch no longer works properly.. so what is going on?

Remember our input pin wants to read a HIGH or a LOW value. When we press the switch we see a HIGH value from the 5V connection, but where is our LOW value? LOW means “connected to Ground” but when the switch is open (not pressed) the input pin is not connected to anything... we say it is “floating” and will just read garbage HIGH or LOW values.

The 10k resistor fixes this. It is what we call a “pull down” resistor because when the switch is open, the input pin is “pulled down” to a LOW level. However because the resistance is very high, when the switch is pressed the HIGH level easily overcomes the LOW level from the resistor. At no time is the input “floating” and reading garbage, so our switch behaves as it should. Nice!

Using Internal Pull-Ups

Lets make a slight change to our circuit. Remove the 10k resistor and the connection to 5V and connect one side of the switch to the Ground rail as shown



Run the sketch **LEDScannerSwitch2.** This should work exactly as the previous sketch did - the scanner runs when we press the button - however our circuit is much simpler now, avoiding the extra resistor.

In this sketch there are two important changes; firstly we set our input pin to have a **pinMode** of INPUT_PULLUP and instead of looking for a HIGH input when the button is pressed, we look for a LOW input.

Now, instead of a “pull-down” resistor to hold our floating input LOW, we are using a “pull-up” resistor to it HIGH. The magic here is that the resistor is actually inside the Arduino, saving us an extra component on the board!

There are some technical reasons why pull-up resistors are usually better than pull-down ones (they waste slightly less power) and, since we get them included for free inside the Arduino they are usually the way to go. We just need to enable the pull ups when appropriate (i.e. to avoid reading garbage data from a floating input pin)

Switch Bounce

Now lets say that instead of the scanner running as long as the switch is pressed, we want to “count” presses. We only want the scanner to move on one LED position each time the button is pressed.

With the same breadboard circuit as before, run sketch **LEDScannerSwitch3**. Here we only advance the LED when the state of the switch changes from open to closed.

Press the switch a few times... something is wrong! Sometimes we get multiple steps from one press of the switch. It may be hard to get a single step at all.

The problem is not in our sketch...the input pin really is seeing the switch open and close multiple times for one press! but how?

We call this “switch bounce”. As the metal switch contacts come together they might slip over each other slightly or vibrate slightly for a tiny fraction of a second. The Arduino is checking the switch so rapidly (many thousands of times a second) that it sees this movement as multiple LOW/HIGH changes and counts multiple presses.

It is very common for switches to behave like this when read very rapidly. We fix this by slowing down how often we read the switch, to let it settle down. This type of measure is called “debouncing”

Lets look at the most simple type of debounce measure... simply making our program wait for a bit. Now we should find the switch behaves sensibly and produces single counts.

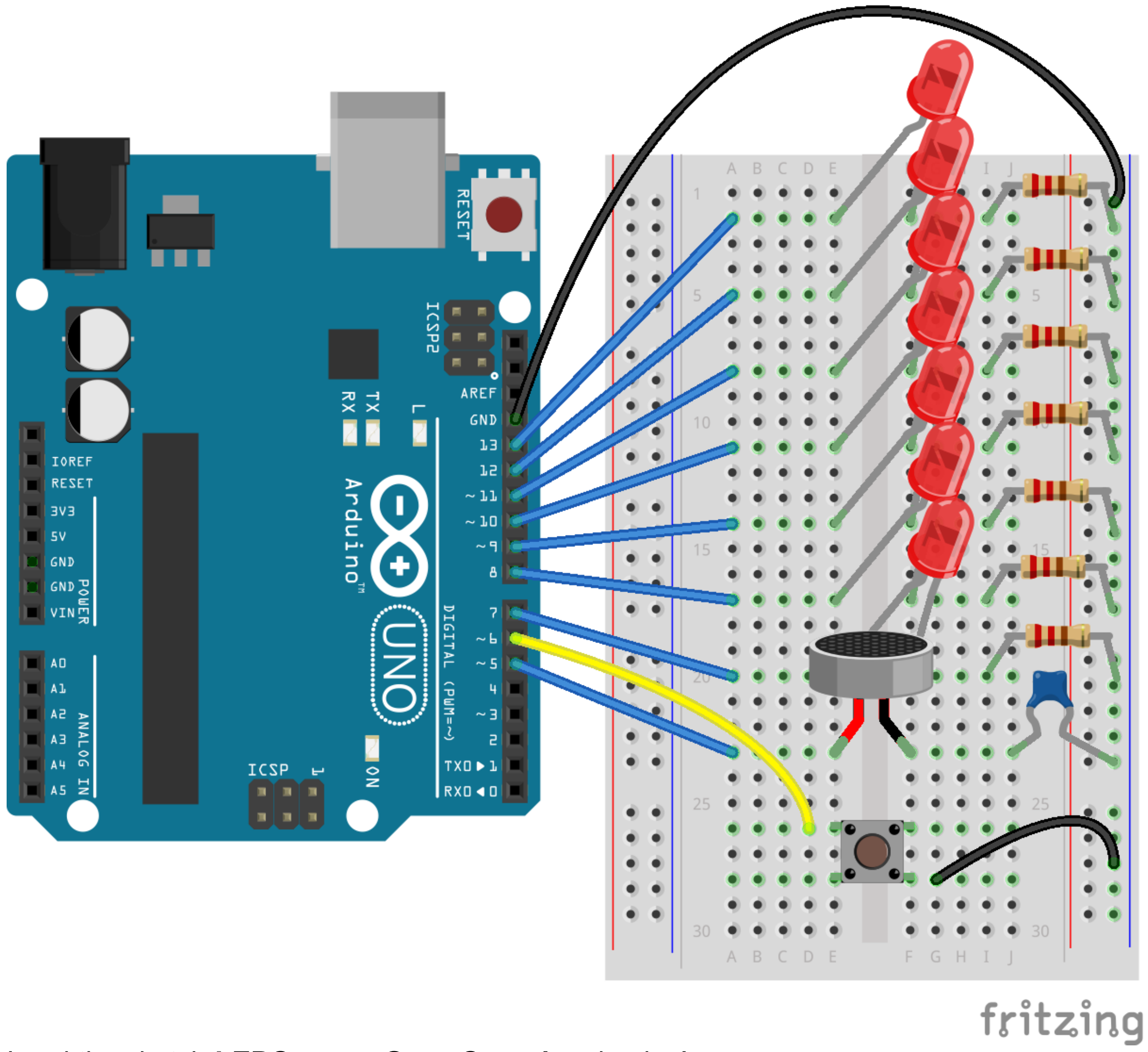
A Game!

Let's have some fun! Replace the fourth red LED with a green one and load the sketch **LEDScannerGame**

Your mission is to press the button when the green LED is lit. Each time you do this, the game gets faster. You have 3 lives and need to press the Reset button of the Arduino to start a new game.

Adding Some Sound

What is a game without sound effects? Lets add the piezo buzzer (check polarity) and one of the 100nF ceramic capacitors to the circuit, as shown below



Load the sketch **LEDScannerGameSound** and enjoy!

We can use the **tone** and **noTone** functions to make sounds, but all they are really doing is switching an ordinary digital output pin on and off very fast (hundreds of times each second) to make the piezo speaker vibrate at a frequency we hear as a beep.

Q: Why do we need the capacitor?

When we pulse the piezo speaker, we just want to vibrate it. There is no point passing a current through it longer than is needed to make it move one way or the other (a prolonged current could damage it)

A capacitor can be used to pass a “signal” (i.e. a changing current) while blocking a steady current - which is just what we want here.

Capacitors come in many sizes have many uses (they can function like rechargeable batteries). Here we are using a low value capacitor as a type of “filter”