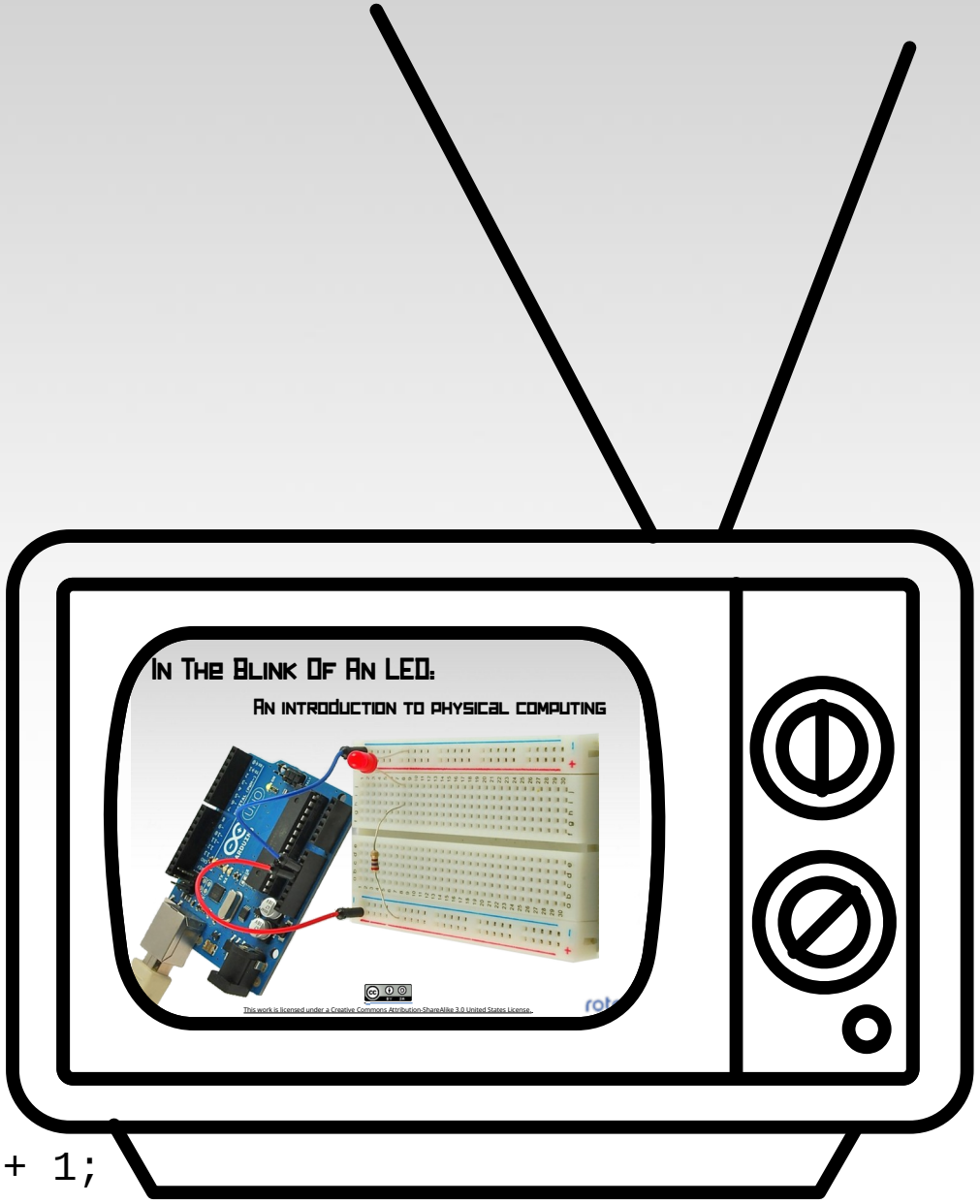
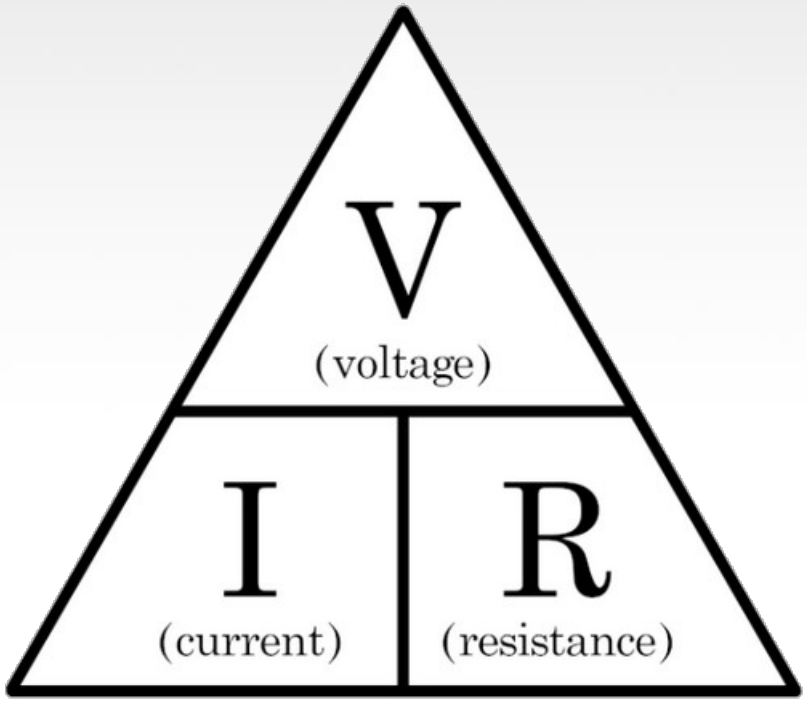
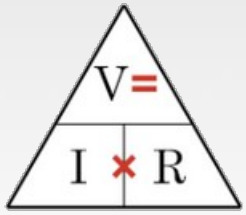
Previously On...

- Voltage, Current and Resistance
- Inputs and Outputs
- Digital Vs. Analog
- Commands
  - `pinMode(pin, INPUT/OUTPUT);`
  - `digitalWrite(pin, HIGH/LOW);`
  - `delay(ms);`
  - `analogWrite(pin, 0-255);`
- Functions
  - `void setup() {}`
  - `void loop() {}`
- Comments
  - `// Keep things clear`
- Variables
  - `int led = 9;`
  - `int brightness = brightness + 1;`

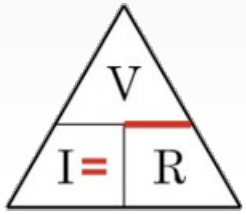


Ohm's Law

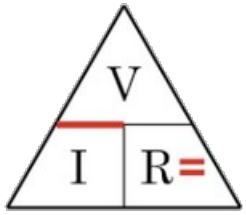




$$V = I \times R$$



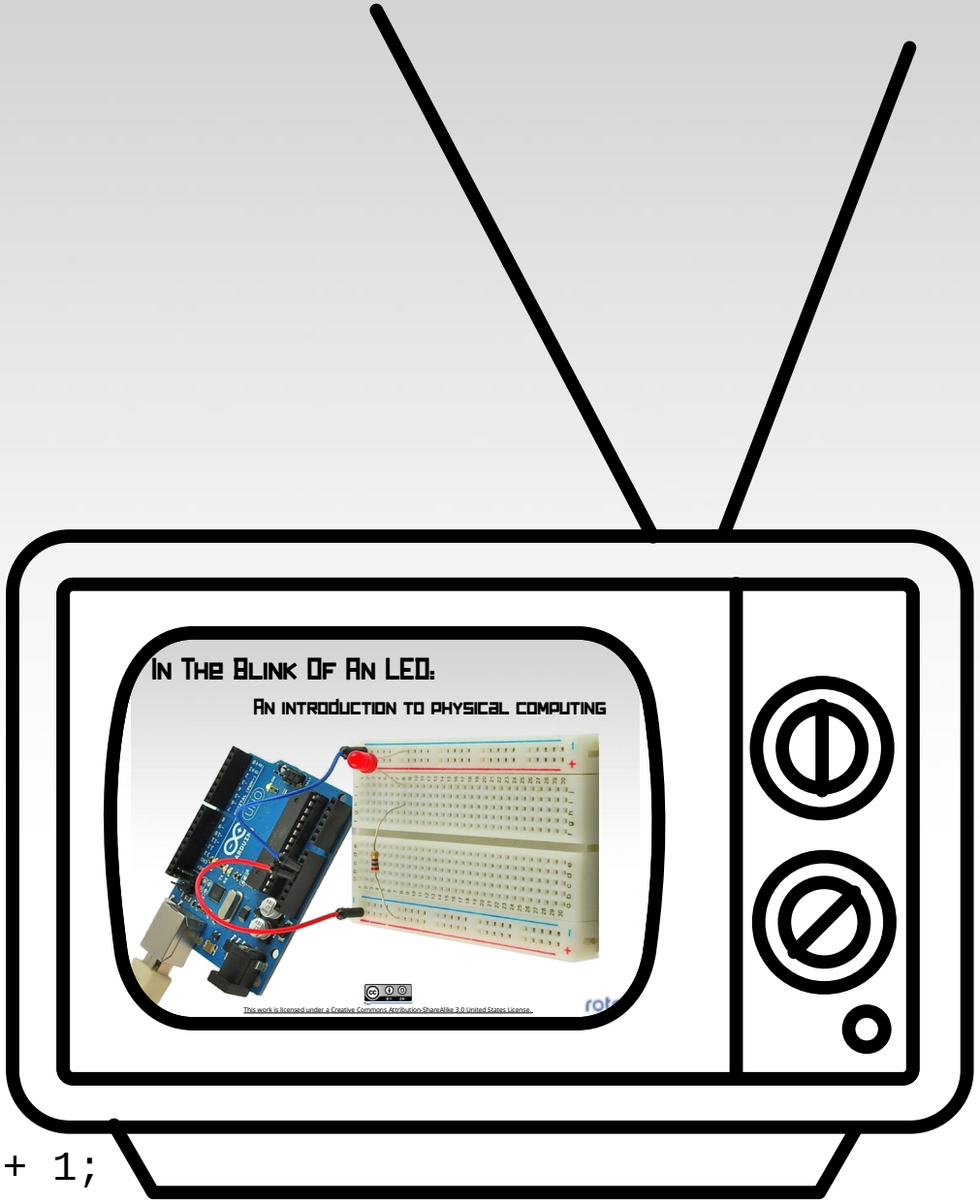
$$I = V \div R$$



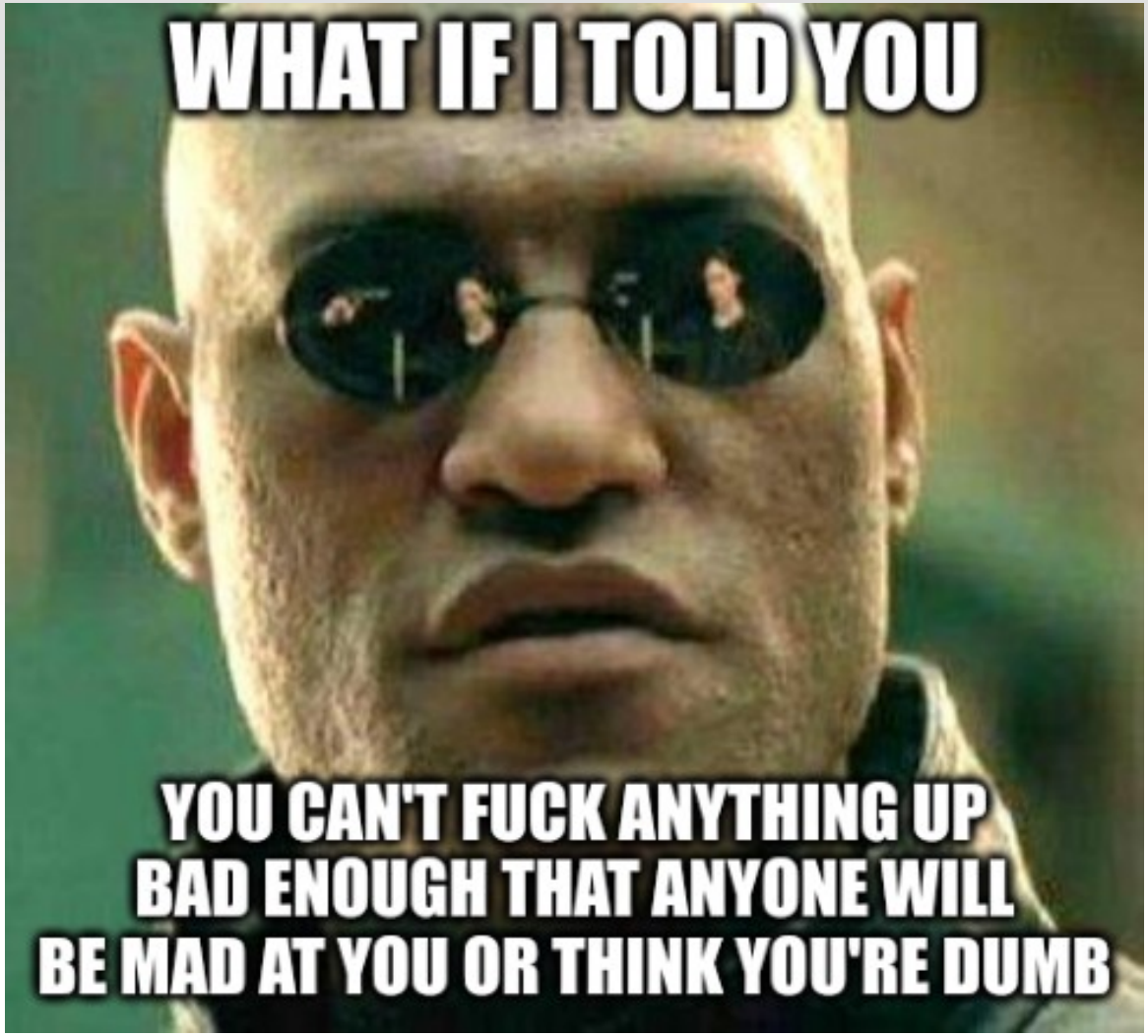
$$R = V \div I$$

Previously On...

- Voltage, Current and Resistance
- Inputs and Outputs
- Digital Vs. Analog
- Commands
  - `pinMode(pin, INPUT/OUTPUT);`
  - `digitalWrite(pin, HIGH/LOW);`
  - `delay(ms);`
  - `analogWrite(pin, 0-255);`
- Functions
  - `void setup() {}`
  - `void loop() {}`
- Comments
  - `// Keep things clear`
- Variables
  - `int led = 9;`
  - `int brightness = brightness + 1;`



The Most Important Lesson

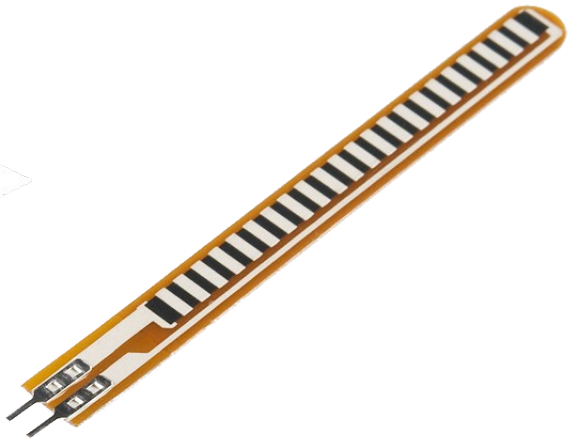


roto

Inputs!

Input is any signal entering an electrical system

- Both digital and analog sensors are forms of input
- Input can also take many other forms: a keyboard, a mouse, infrared sensors, biometric sensors, touch sensors or just plain voltage from a circuit
- This is what makes interactive exhibits *interactive*



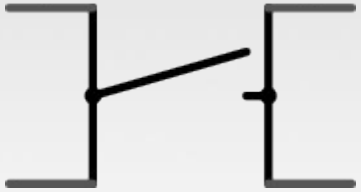
Digital Input

Like digital output, can only sense if a signal is high or low

- For Arduino
  - High is voltage above 3 V
  - Low is voltage below 1.5 V
- General category is “switches”
- Includes push buttons (momentary switches), toggle switches, knife switches (maintained switches), capacitive/touch switches, and many other sensors

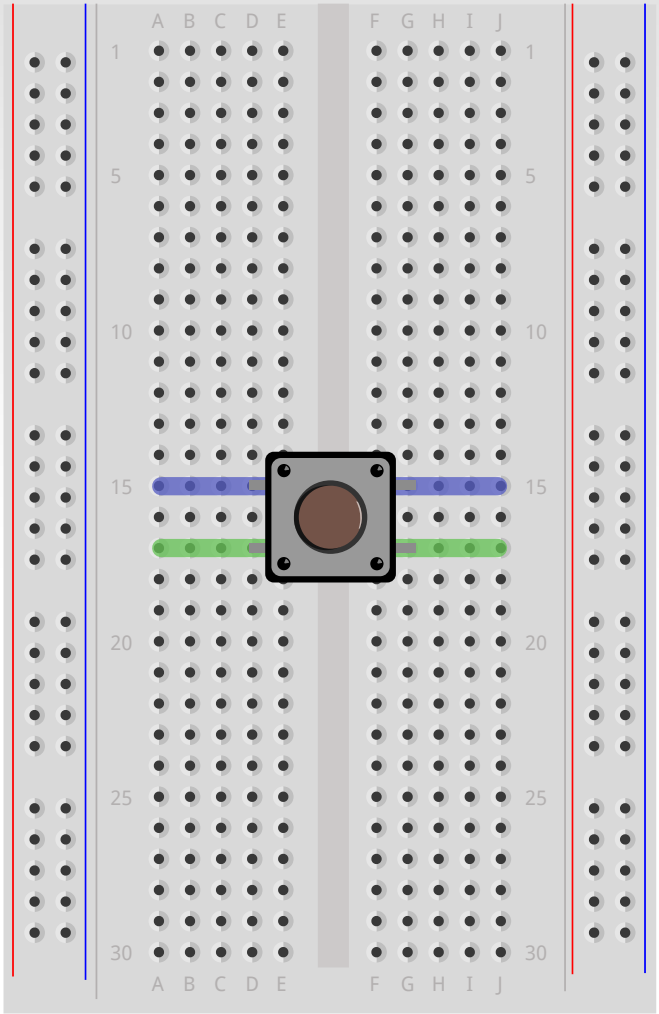
roto

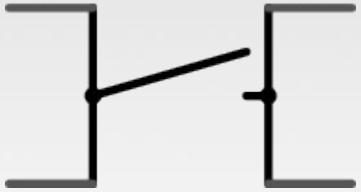
Component Spotlight: The Push Button / Momentary Switch



roto

Push Button and Breadboard Connections

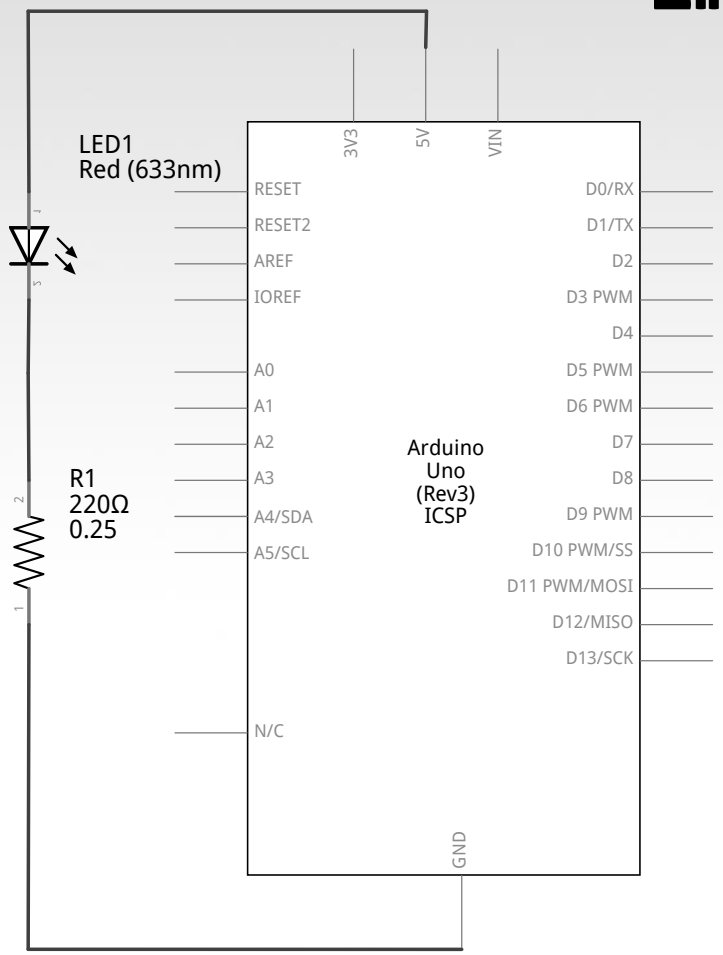




Our buttons have to bridge the gap on our breadboards

The top pins are connected to each other and so are the bottom ones

Using the Breadboard to Built a Simple Input Circuit

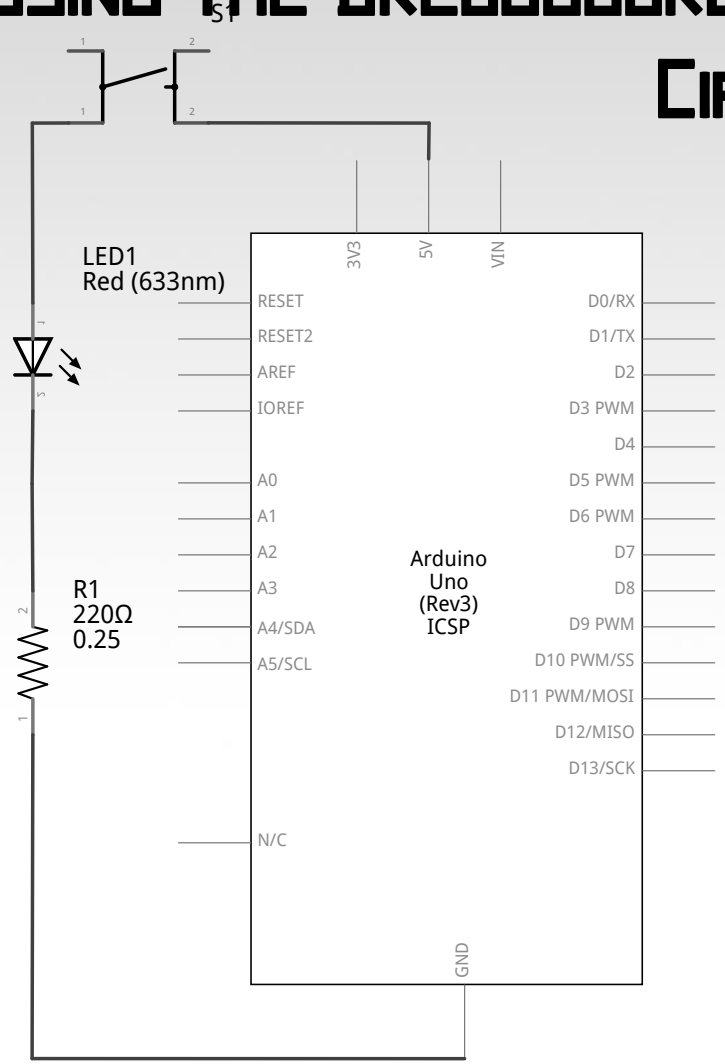


Remember our original circuit?

fritzing

roto

Using the Breadboard to Built a Simple Input Circuit



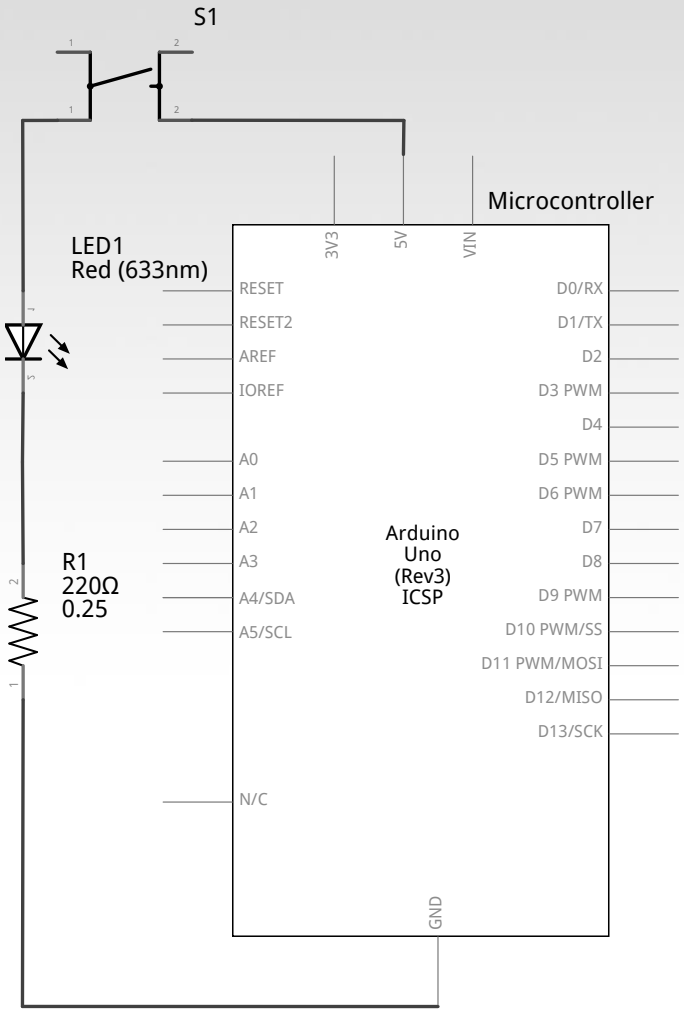
Remember our original circuit?

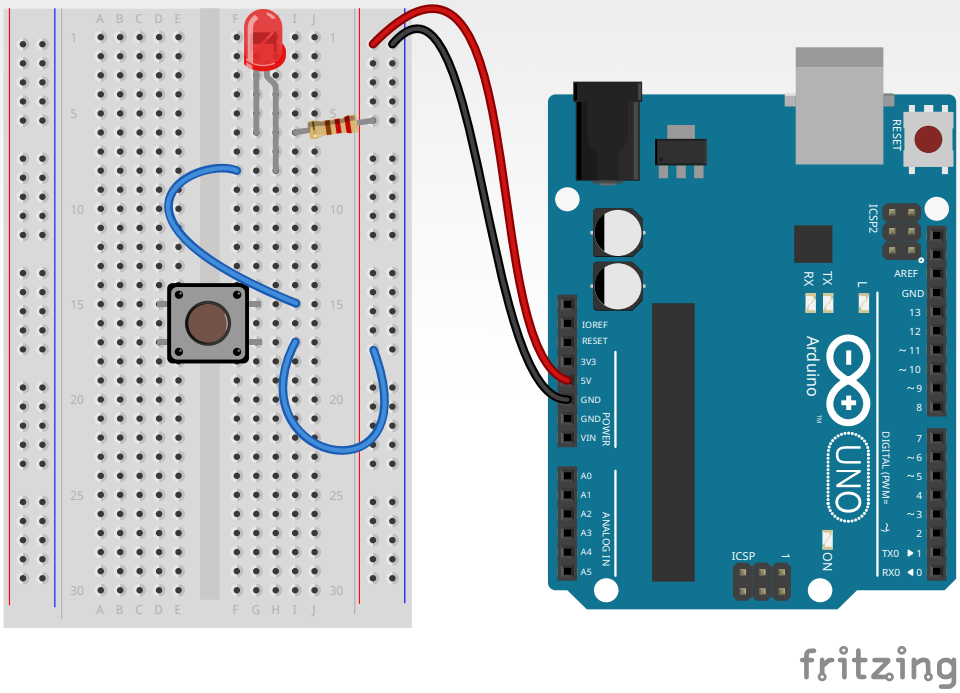
We're going to wire that up again, but this time with a switch

fritzing

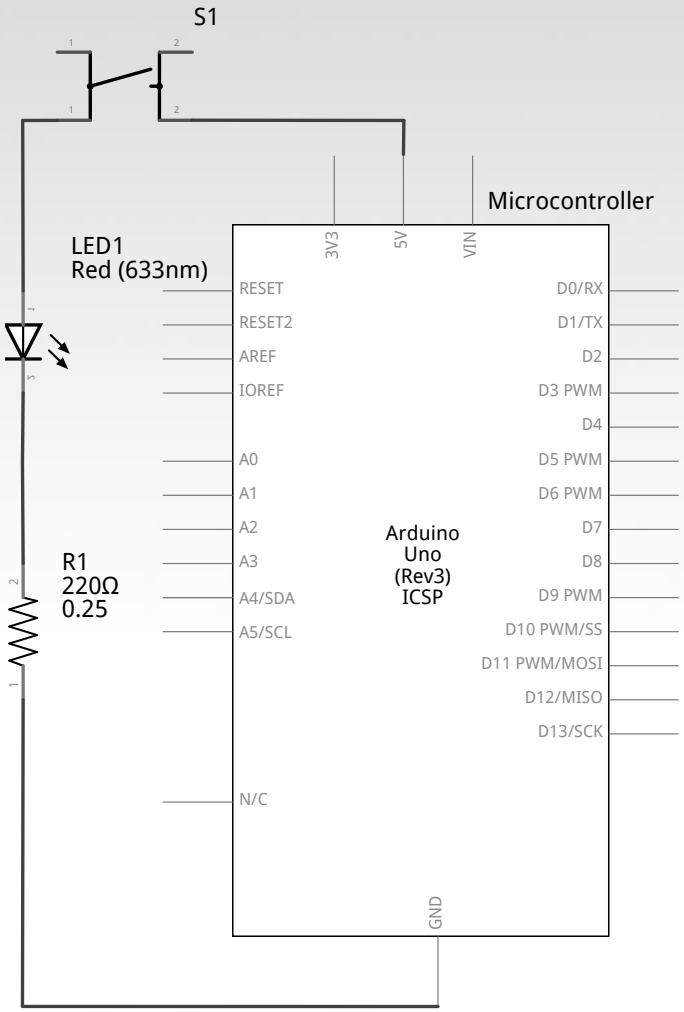
roto

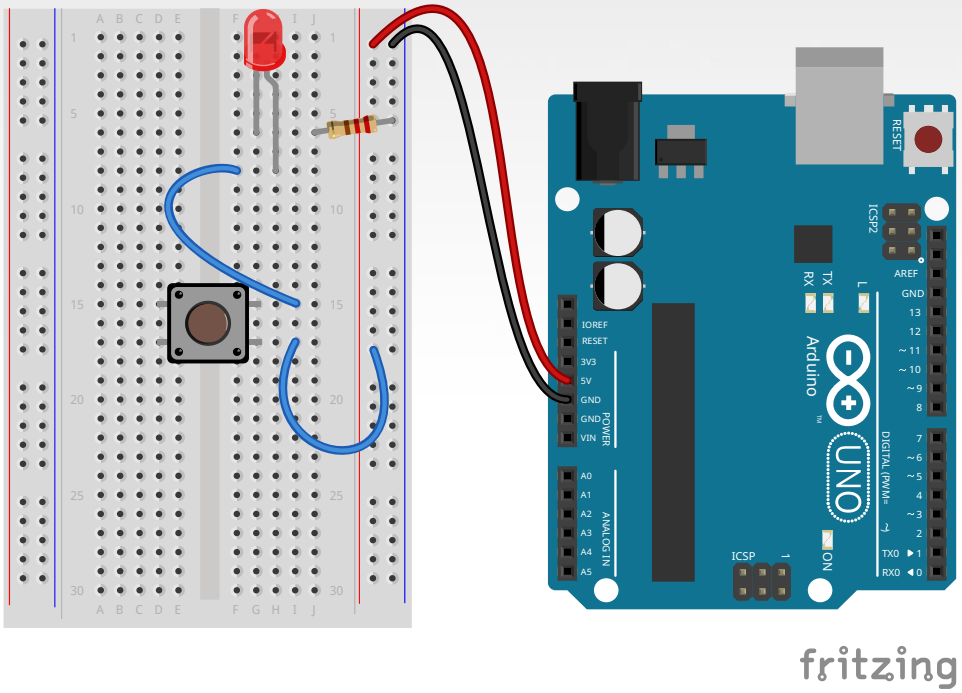
Using the Breadboard to Built a Simple Input Circuit





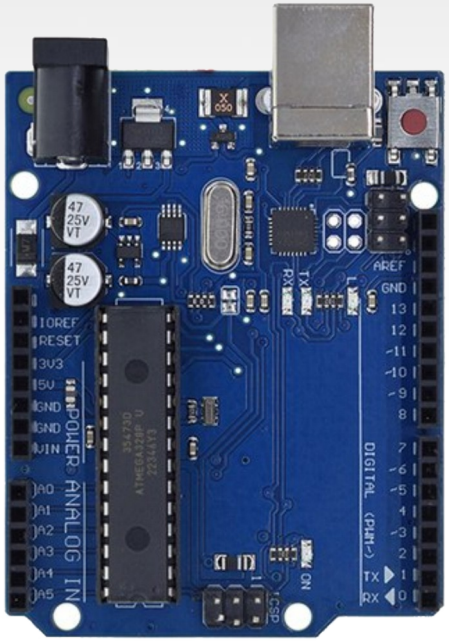
Using the Breadboard to Built a Simple Input Circuit



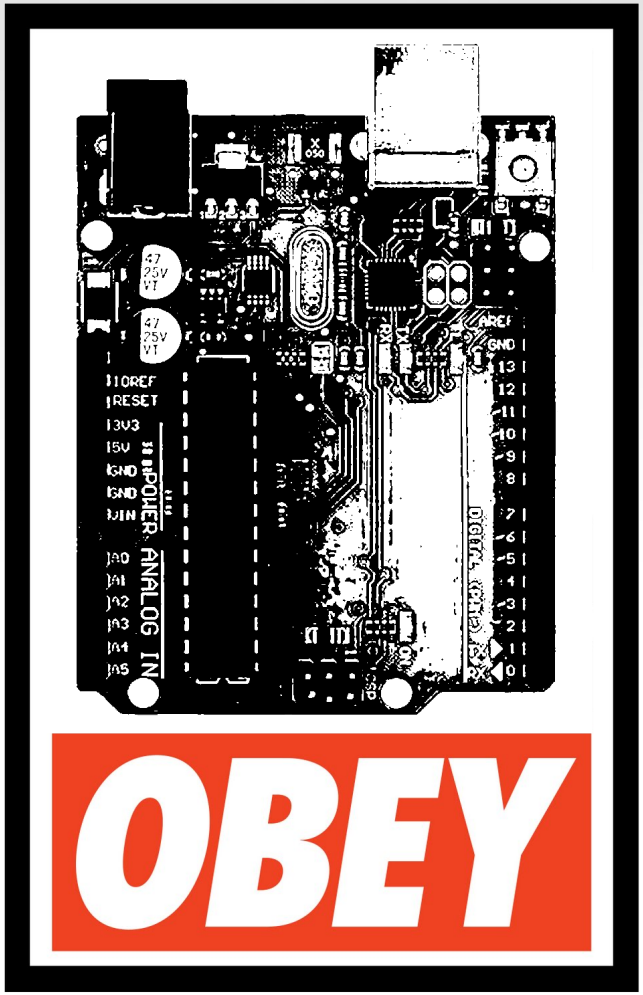


fritzing

Go ahead and plug your board in!

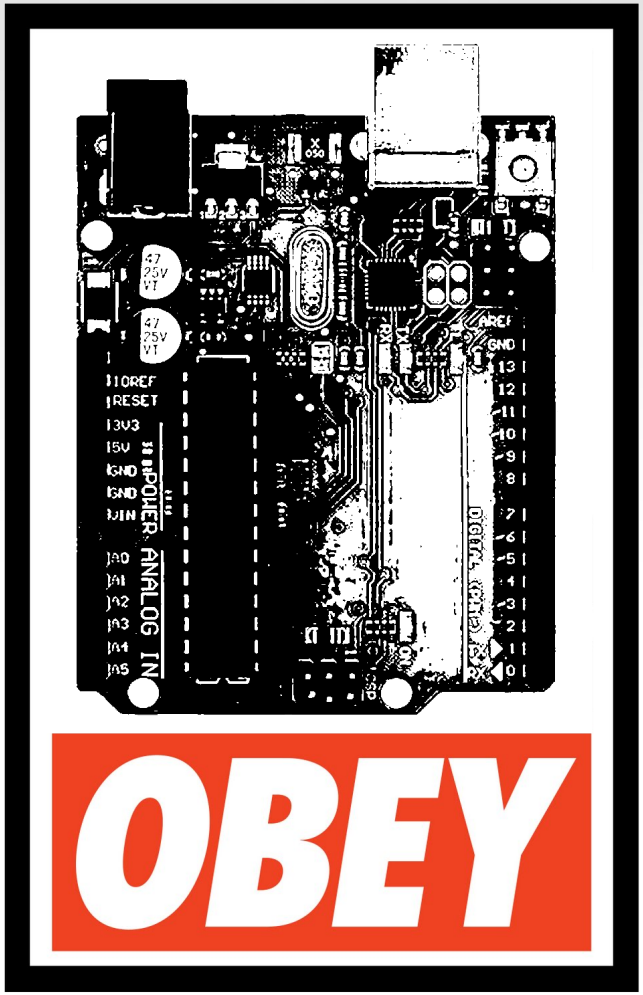


Adding Control



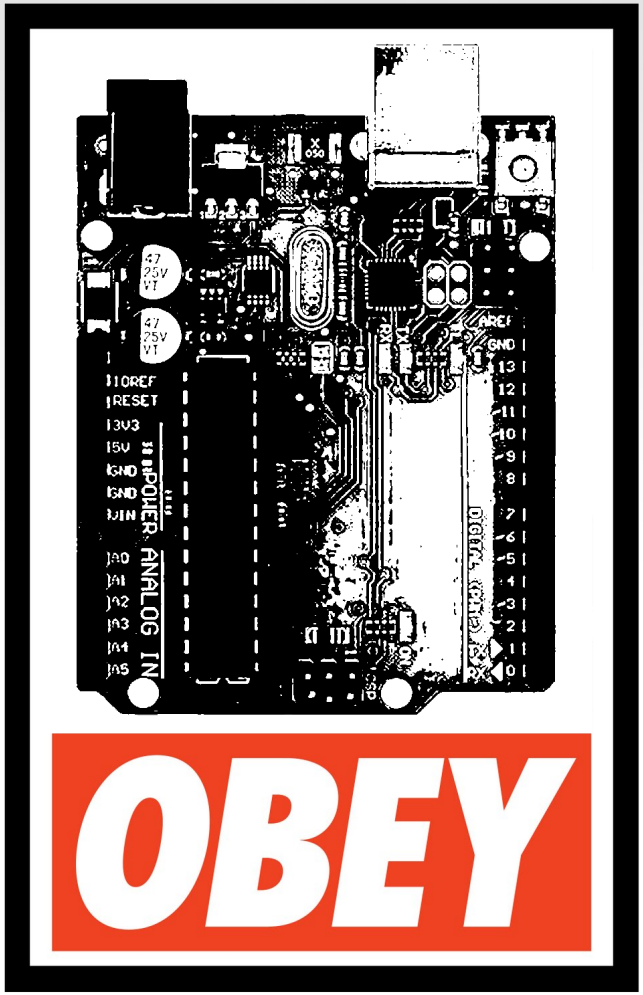
Adding Control

Any circuit that has meaningful input has some degree of control

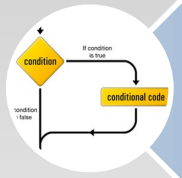


Adding Control

Any circuit that has meaningful input has some degree of control



Microcontrollers allow us to assign different relationships between inputs and outputs

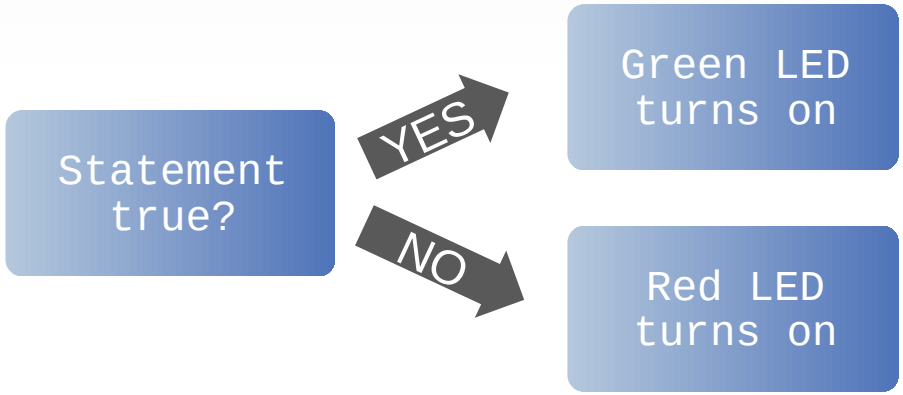


if() statements / Boolean logic

Project 3 – Lie Detector

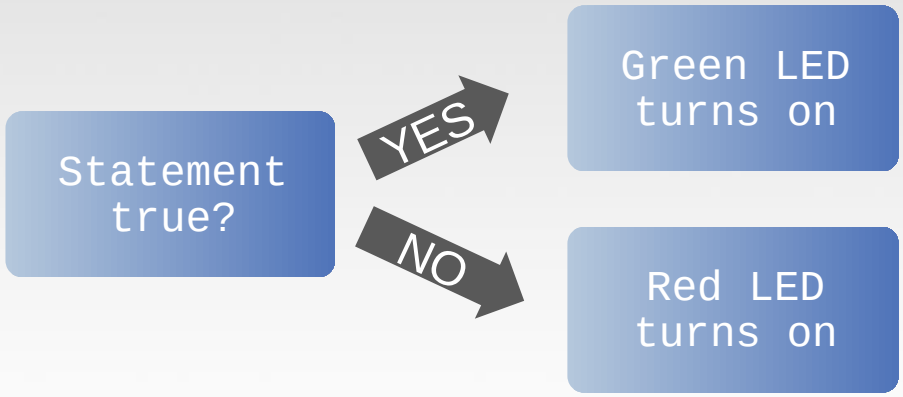
Just give me 3 clock cycles with the bastard

Psuedo-code – how should this work?



roto

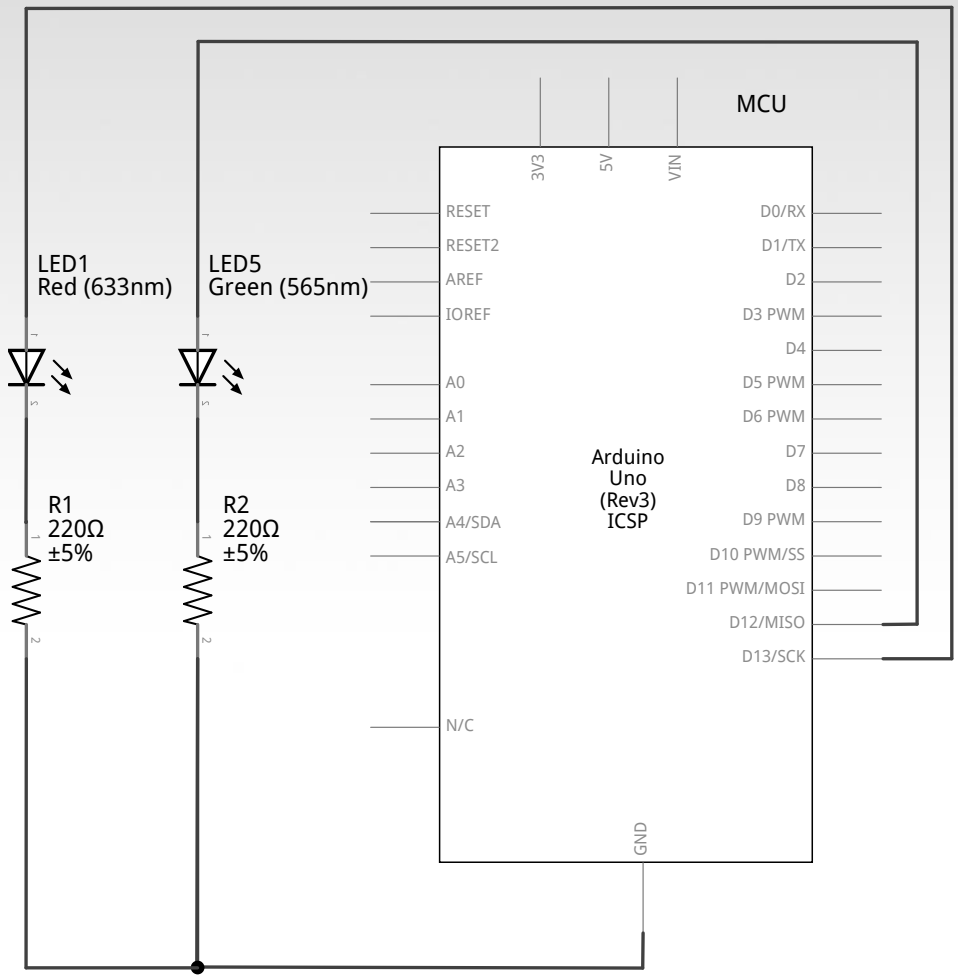
Project 3 – Lie Detector
Inputs and Outputs



Inputs	Outputs
None	Green LED
	Red LED

Project 3 – Lie Detector

Schematic



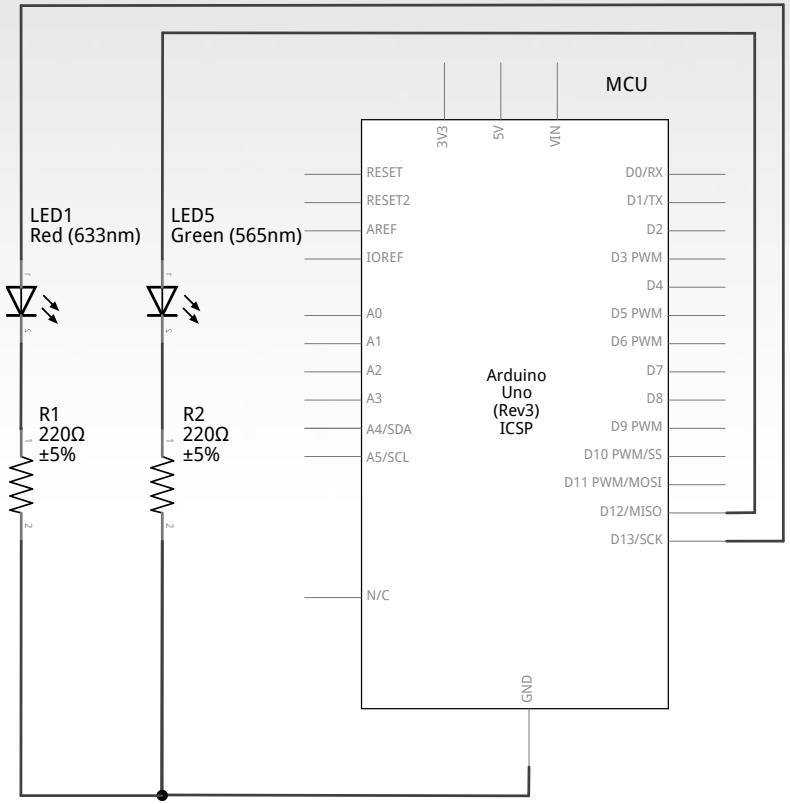
You can leave the button on the breadboard- we'll be using it again soon

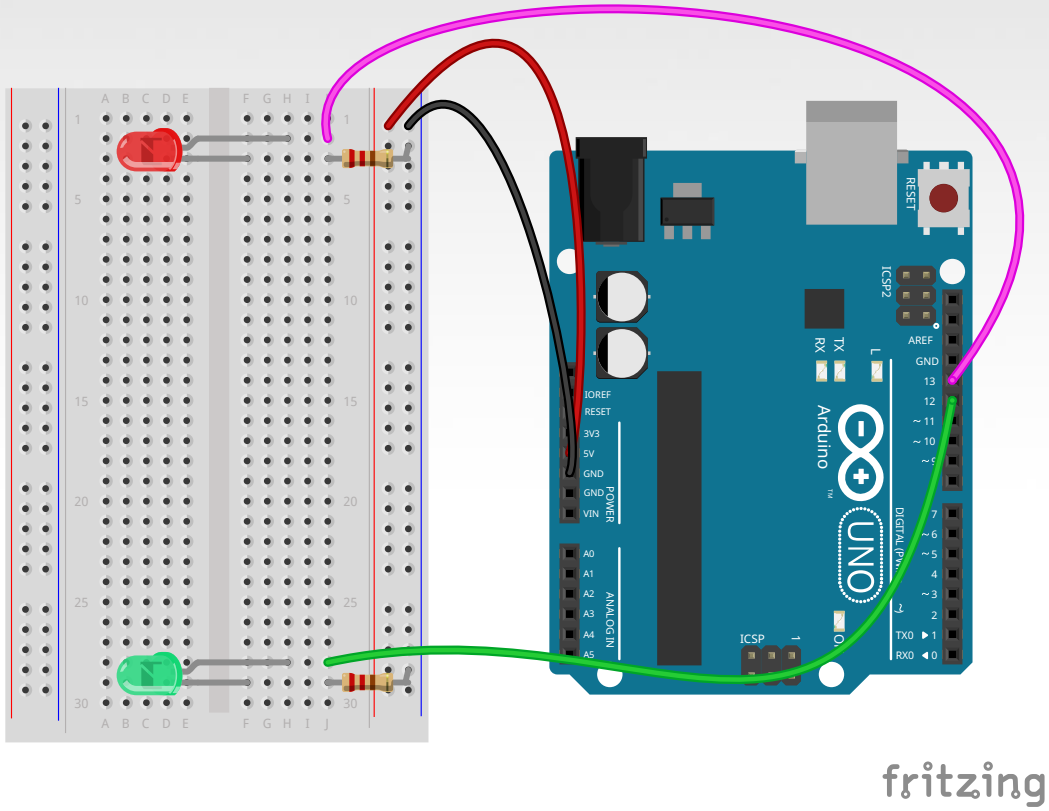
fritzing

roto

Project 3 – Lie Detector

Wiring Diagram





Project 3 – Lie Detector
Conditional Statements

the if() statement

```
if (varA > varB) {
  analogWrite(led, varA);
}
else {
  analogWrite(led, varB);
}
```

Project 3 – Lie Detector
Conditional Statements

```
if (varA > varB) {
  analogWrite(led, varA);
}
else {
  analogWrite(led, varB);
}
```

If this is true

roto

Project 3 – Lie Detector
Conditional Statements

```
if (varA > varB) {
  analogWrite(led, varA);
}
else {
  analogWrite(led, varB);
}
```

Project 3 – Lie Detector
Conditional Statements

```
if (varA > varB) {
  analogWrite(led, varA);
}
else {
  analogWrite(led, varB);
}
```

If this is true

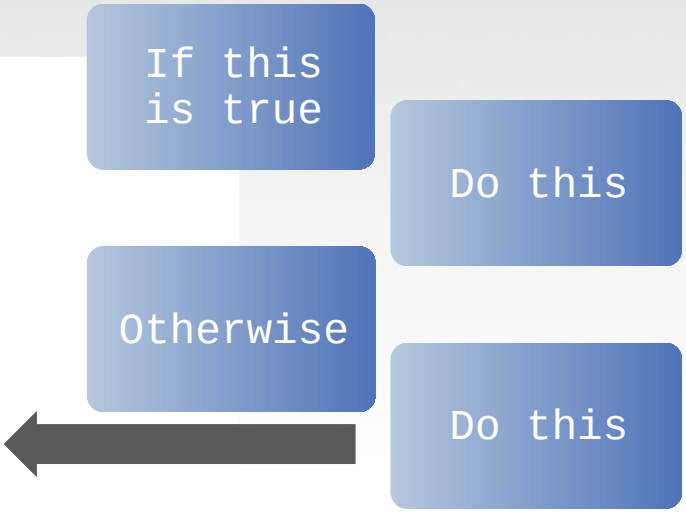
Do this

Otherwise

roto

Project 3 – Lie Detector
Conditional Statements

```
if (varA > varB) {
  analogWrite(led, varA);
}
else {
  analogWrite(led, varB);
}
```



Boolean Operators

<Boolean>	True if:
`(a)`	a is true, HIGH or does not equal zero
`(!a)`	a is false, LOW or equals zero
`(a) == (b)`	a is equal to b
`(a) != (b)`	a is not equal to b
`(a) > (b)`	a is greater than b
`(a) >= (b)`	a is greater than or equal to b
`(a) < (b)`	a is less than b
`(a) <= (b)`	a is less than or equal to b
`(a) && (b)`	both a is true AND b is true
`(a) || (b)`	either a is true OR b is true

Déjà Vu

Select Board

fade.ino

```
1  /*
2   * Gradually fade an LED on and off
3   */
4
5  // assign pin based off circuit
6  int led = 9;
7
8  int brightness = 0;     // holds the value for how bright the led is
9  int fade_amount = 5;    // how much the brightness changes each loop
10
11 // the setup function runs once when you press reset or power the board
12 void setup() {
13   // initialize digital pin 9 as an output.
14   pinMode(led, OUTPUT);
15 }
16
17 // the loop function runs over and over again forever
18 void loop() {
19   analogWrite(led, brightness);               // set the brightness of pin 9
20
21   brightness = brightness + fade_amount;      // update the brightness for the next loop
22
23   if (brightness <= 0 || brightness >= 255) { // if the LED is all the way off or on
24     fade_amount = fade_amount * -1;           // change the direction of the fade
25   }
26   delay(30);                                  // pause to see the dimming effect
27 }
28
```

Project 3 - Lie Detector

if_tester.ino

```
/*
 * Test our if() statements
 */

// assign pin based off circuit
int red_led = 13;
int green_led = 12;

// these are the variables we'll play around with
int a = true;
int b = false;

// the setup function runs once when you press reset or power the board
void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);

  // change the expression inside the if() and look at the results
  if (a) {
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HIGH);
  }
  else {
    digitalWrite(green_led, LOW);
    digitalWrite(red_led, HIGH);
  }
}

// the loop function runs over and over again forever
void loop() {

}
```

Project 3 - Lie Detector

if_tester.ino

```
/*
 * Test our if() statements
 */

// assign pin based off circuit
int red_led = 13;
int green_led = 12;

// these are the variables we'll play around with
int a = true;
int b = false;

// the setup function runs once when you press reset or power the board
void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);

  // change the expression inside the if() and look at the results
  if (b) {
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HIGH);
  }
  else {
    digitalWrite(green_led, LOW);
    digitalWrite(red_led, HIGH);
  }
}

// the loop function runs over and over again forever
void loop() {

}
```


roto

Project 3 – Lie Detector

if_tester.ino

```
/*
 * Test our if() statements
 */

// assign pin based off circuit
int red_led = 13;
int green_led = 12;

// these are the variables we'll play around with
int a = true;
int b = false;

// the setup function runs once when you press reset or power the board
void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);

  // change the expression inside the if() and look at the results
  if (!b) {
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HIGH);
  }
  else {
    digitalWrite(green_led, LOW);
    digitalWrite(red_led, HIGH);
  }
}

// the loop function runs over and over again forever
void loop() {

}
```

Project 3 – Lie Detector

if_tester.ino

```
/*
 * Test our if() statements
 */

// assign pin based off circuit
int red_led = 13;
int green_led = 12;

// these are the variables we'll play around with
int a = true;
int b = false;

// the setup function runs once when you press reset or power the board
void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);

  // change the expression inside the if() and look at the results
  if (!a) {
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HIGH);
  }
  else {
    digitalWrite(green_led, LOW);
    digitalWrite(red_led, HIGH);
  }
}

// the loop function runs over and over again forever
void loop() {

}
```

Project 3 - Lie Detector

if_tester.ino

```
/*
 * Test our if() statements
 */

// assign pin based off circuit
int red_led = 13;
int green_led = 12;

// these are the variables we'll play around with
int a = 9;
int b = 5;

// the setup function runs once when you press reset or power the board
void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);

  // change the expression inside the if() and look at the results
  if (a > b) {
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HIGH);
  }
  else {
    digitalWrite(green_led, LOW);
    digitalWrite(red_led, HIGH);
  }
}

// the loop function runs over and over again forever
void loop() {

}
```

Project 3 - Lie Detector

if_tester.ino

```
/*
 * Test our if() statements
 */

// assign pin based off circuit
int red_led = 13;
int green_led = 12;

// these are the variables we'll play around with
int a = 9;
int b = 5;

// the setup function runs once when you press reset or power the board
void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);

  // change the expression inside the if() and look at the results
  if (a < b) {
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HIGH);
  }
  else {
    digitalWrite(green_led, LOW);
    digitalWrite(red_led, HIGH);
  }
}

// the loop function runs over and over again forever
void loop() {

}
```

Project 3 – Lie Detector

if_tester.ino

```
/*
 * Test our if() statements
 */

// assign pin based off circuit
int red_led = 13;
int green_led = 12;

// these are the variables we'll play around with
int a = true;
int b = 5;

// the setup function runs once when you press reset or power the board
void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);

  // change the expression inside the if() and look at the results
  if (b > 5 || a) {
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HIGH);
  }
  else {
    digitalWrite(green_led, LOW);
    digitalWrite(red_led, HIGH);
  }
}

// the loop function runs over and over again forever
void loop() {

}
```

Project 3 – Lie Detector

if_tester.ino

```
/*
 * Test our if() statements
 */

// assign pin based off circuit
int red_led = 13;
int green_led = 12;

// these are the variables we'll play around with
int a = true;
int b = 5;

// the setup function runs once when you press reset or power the board
void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);

  // change the expression inside the if() and look at the results
  if (b > 5 && a) {
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HIGH);
  }
  else {
    digitalWrite(green_led, LOW);
    digitalWrite(red_led, HIGH);
  }
}

// the loop function runs over and over again forever
void loop() {

}
```

Project 3 – Lie Detector

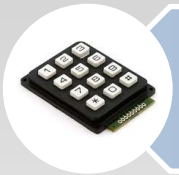
if_tester.ino

```
/*
 * Test our if() statements
 */

// assign pin based off circuit
int red_led = 13;
int green_led = 12;

// these are the variables we'll play around with
int a = true;
int b = 5;

// the setup function runs once when you press reset or power the board
void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);

  // change the expression inside the if() and look at the results
  if (b >= 5 && a) {
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HIGH);
  }
  else {
    digitalWrite(green_led, LOW);
    digitalWrite(red_led, HIGH);
  }
}

// the loop function runs over and over again forever
void loop() {

}
```


roto

Boolean Operators

<Boolean>	True if:
`(a)`	a is true, HIGH or does not equal zero
`(!a)`	a is false, LOW or equals zero
`(a) == (b)`	a is equal to b
`(a) != (b)`	a is not equal to b
`(a) > (b)`	a is greater than b
`(a) >= (b)`	a is greater than or equal to b
`(a) < (b)`	a is less than b
`(a) <= (b)`	a is less than or equal to b
`(a) && (b)`	both a is true AND b is true
`(a) || (b)`	either a is true OR b is true

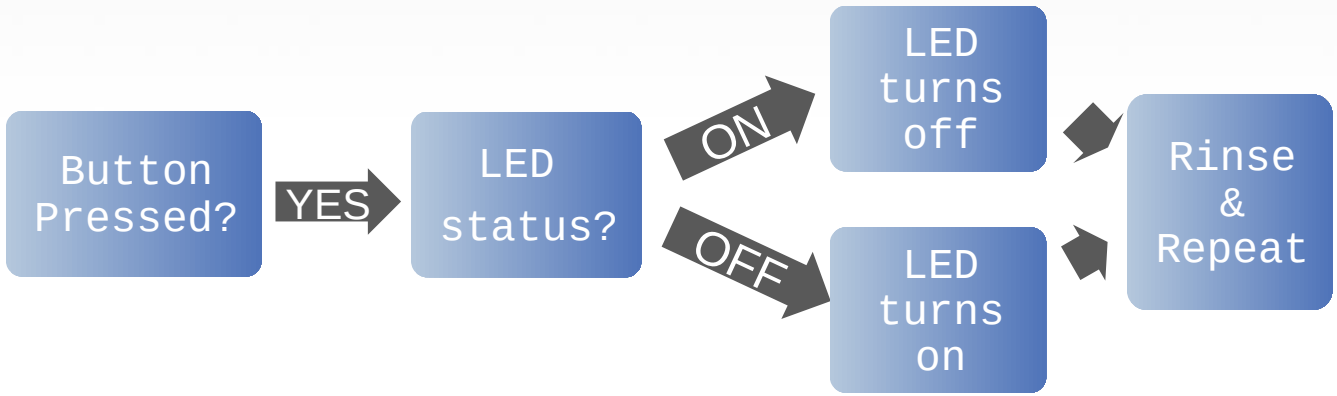


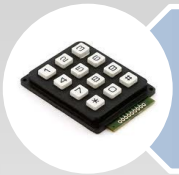
digitalRead()

Project 4 – Button/Maintained Switch Switch

Who you callin' momentary?

Psuedo-code – how should this work?



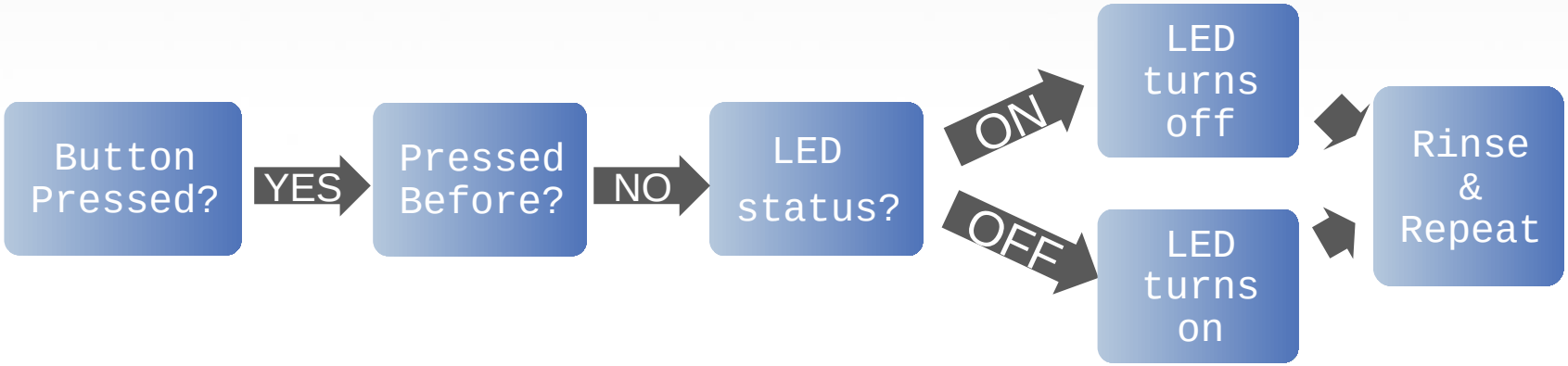


digitalRead()

Project 4 – Button/Maintained Switch Switch

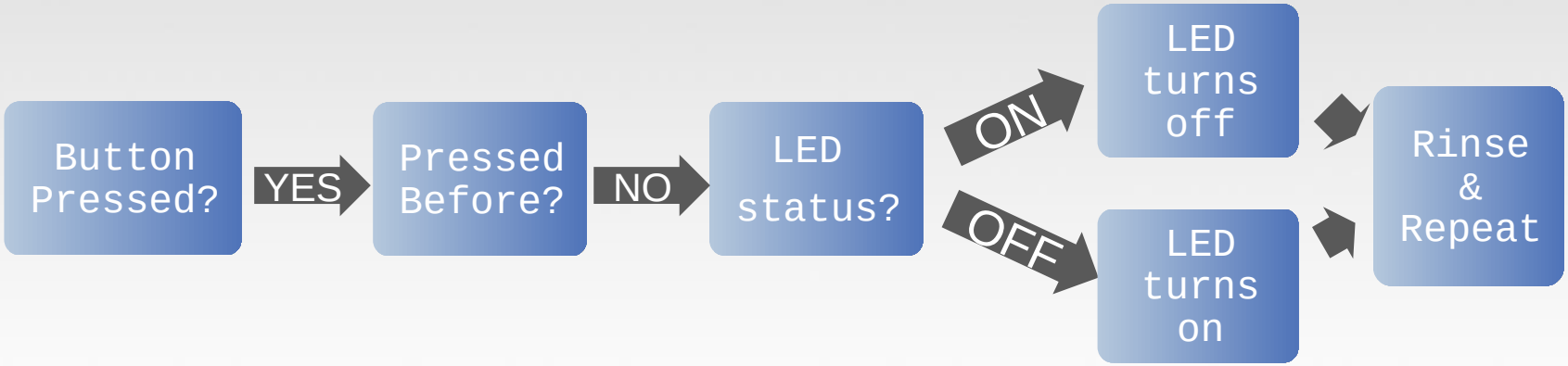
Who you callin' momentary?

Psuedo-code – how should this work?



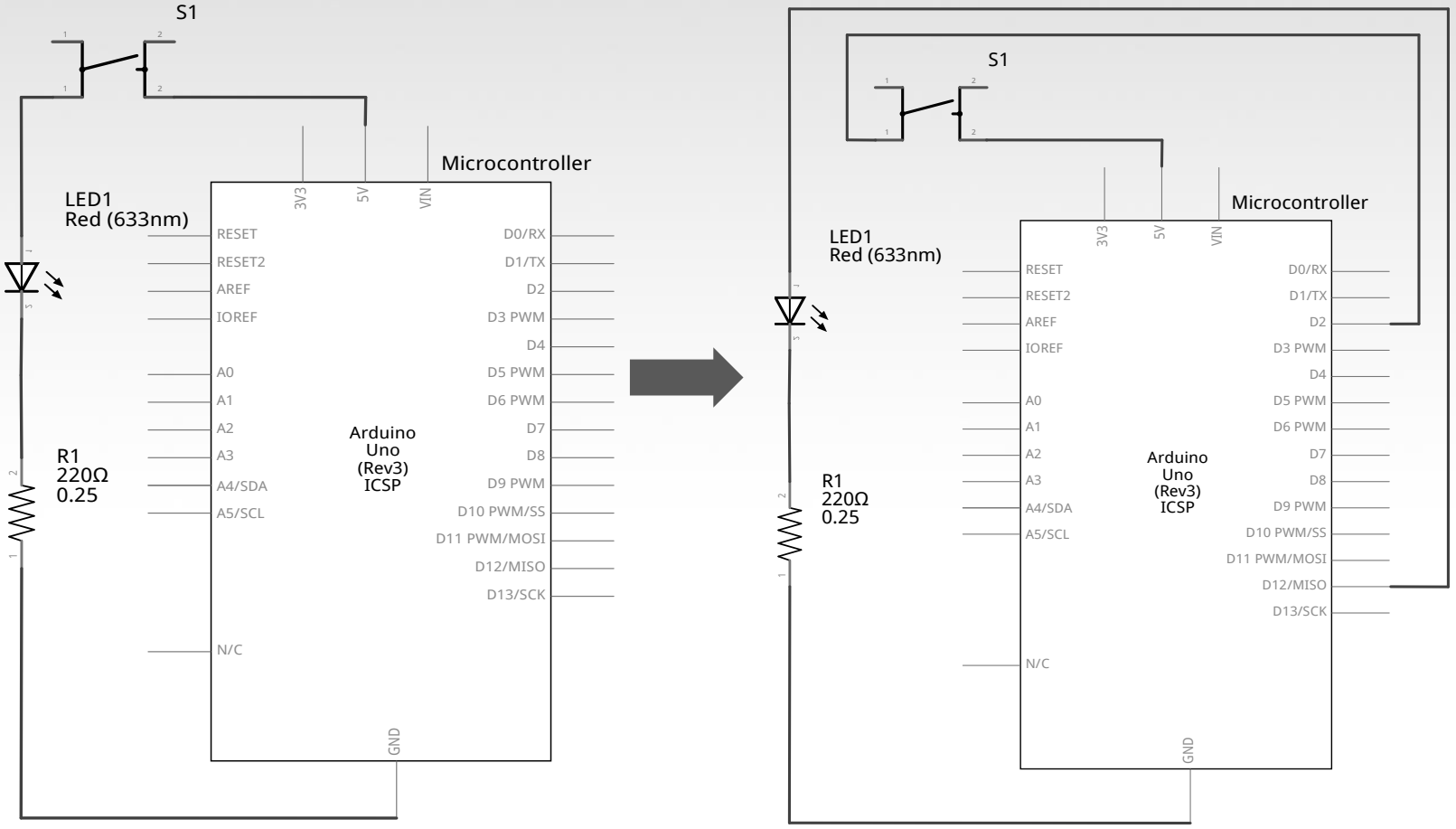
roto

Project 4 – Button/Maintained Switch Switch Inputs and Outputs

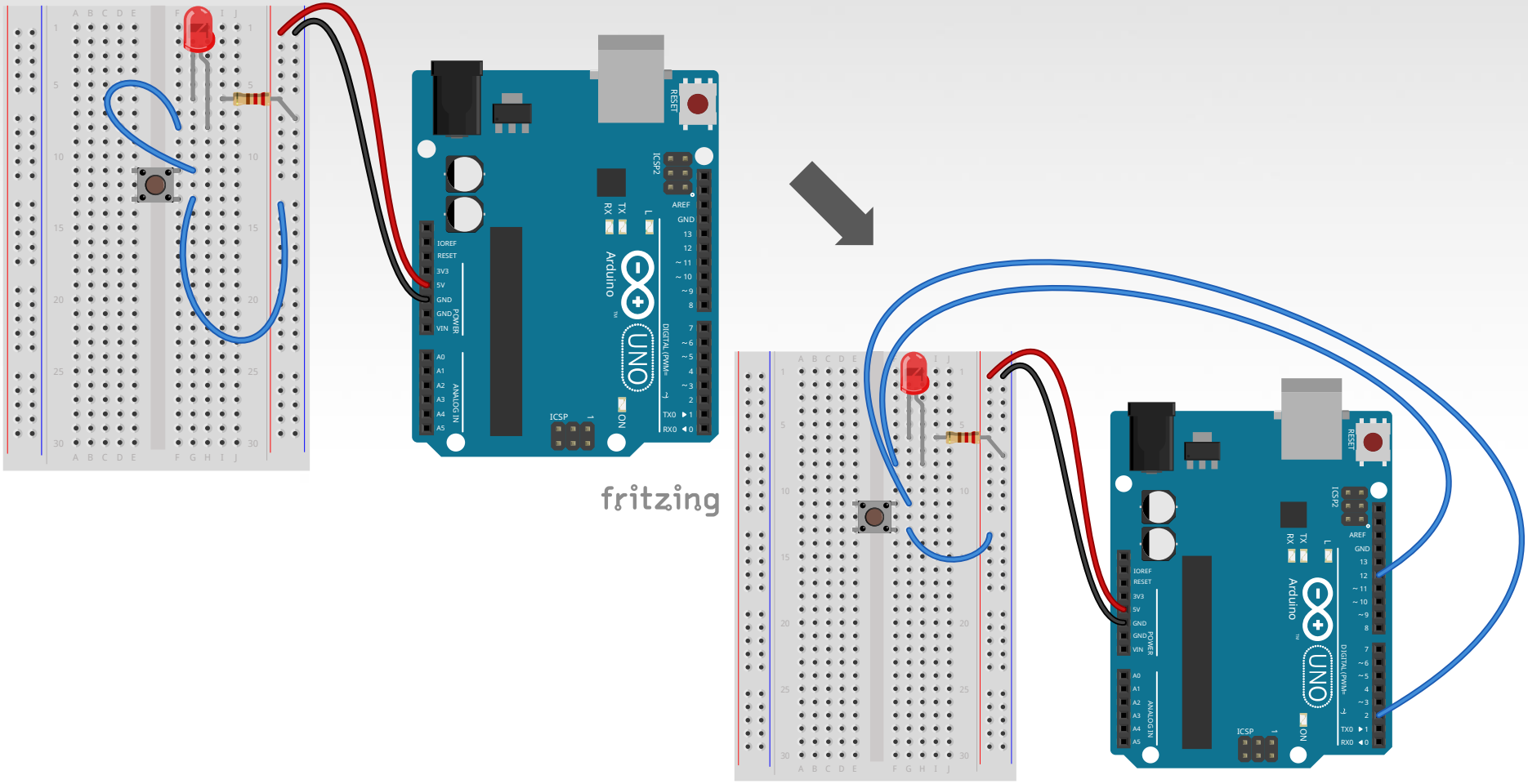


Inputs	Outputs
Push Button	LED

Transitioning to Micro-control Output & Input

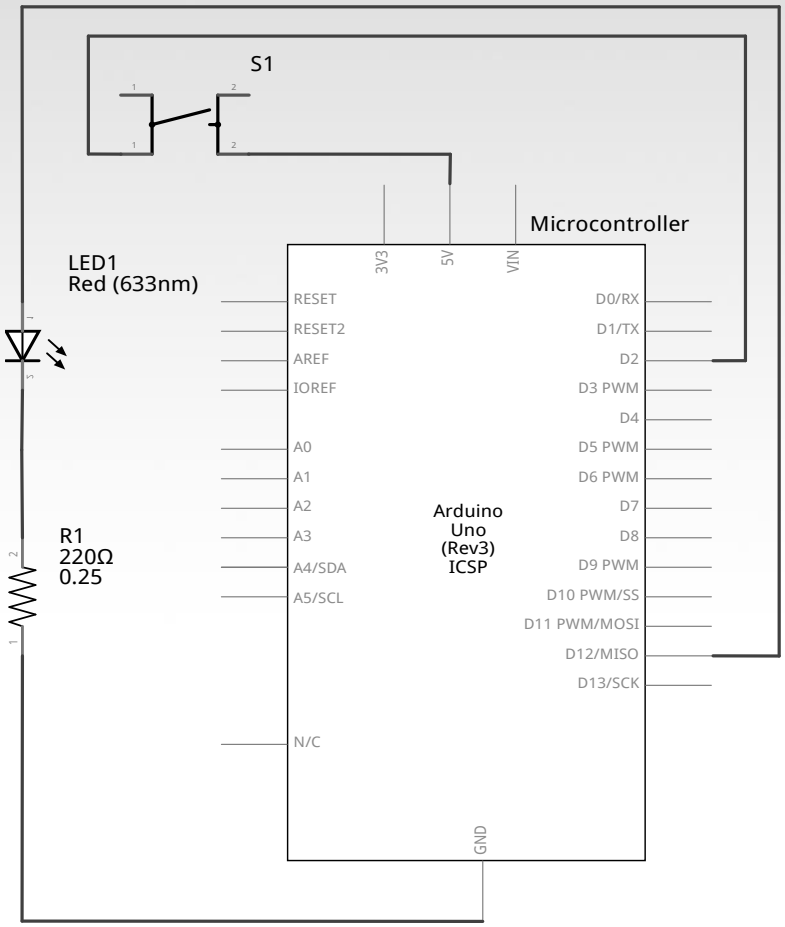


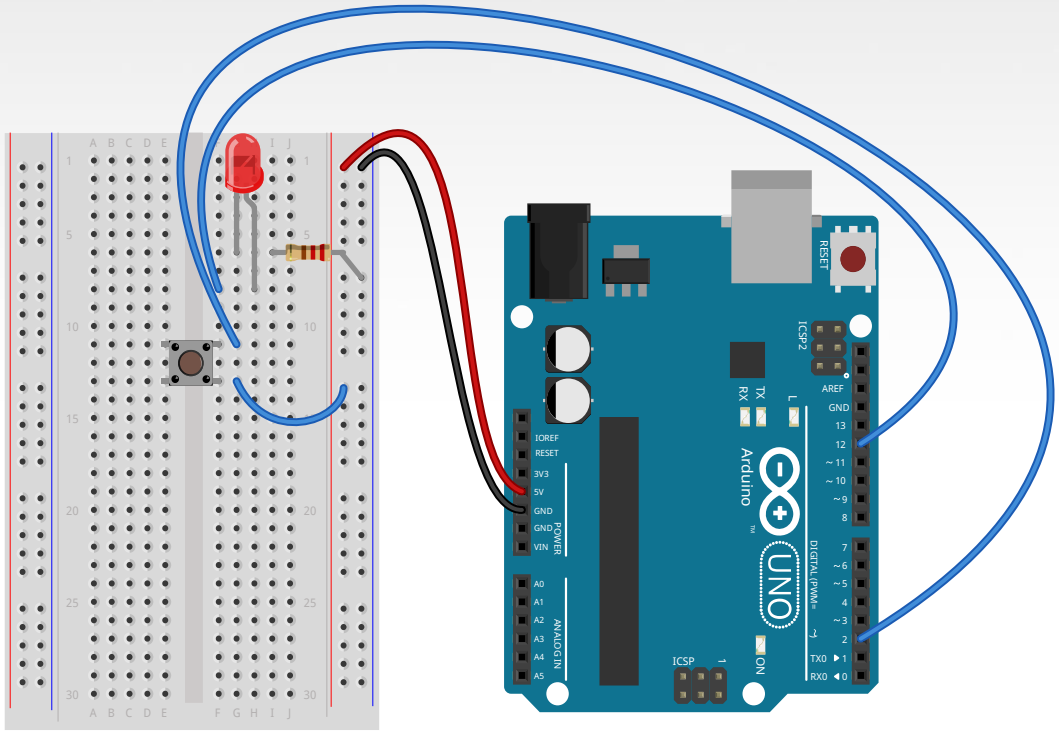
Transitioning to Micro-control Output & Input



Project 4 – Button/Maintained Switch Switch

Wiring Diagram





fritzing

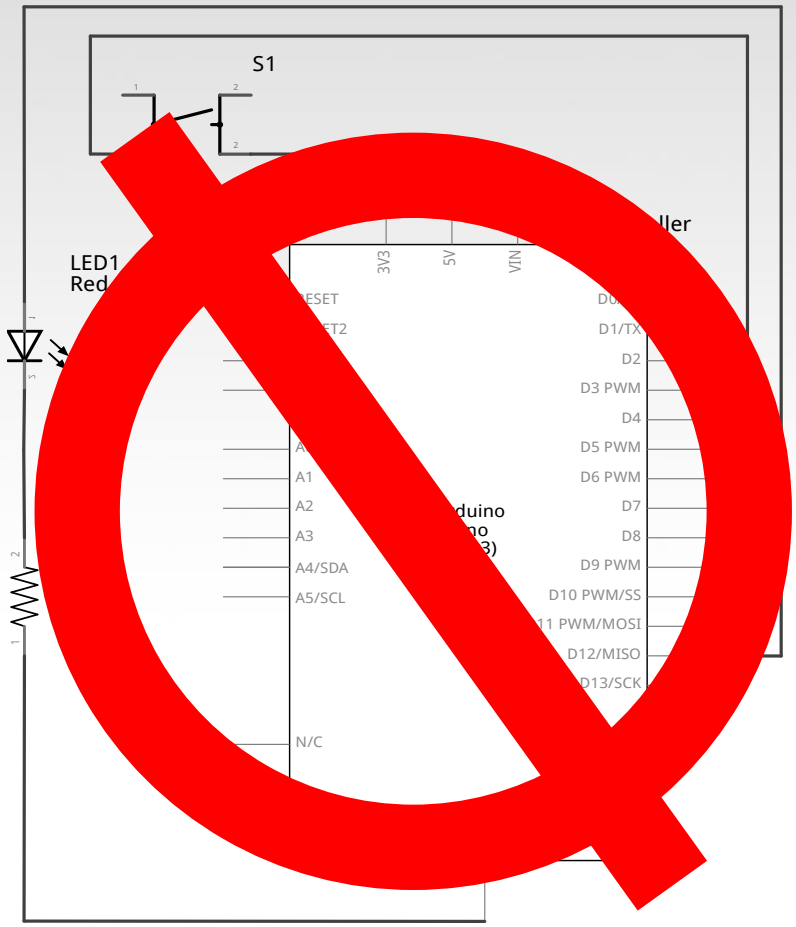
fritzing

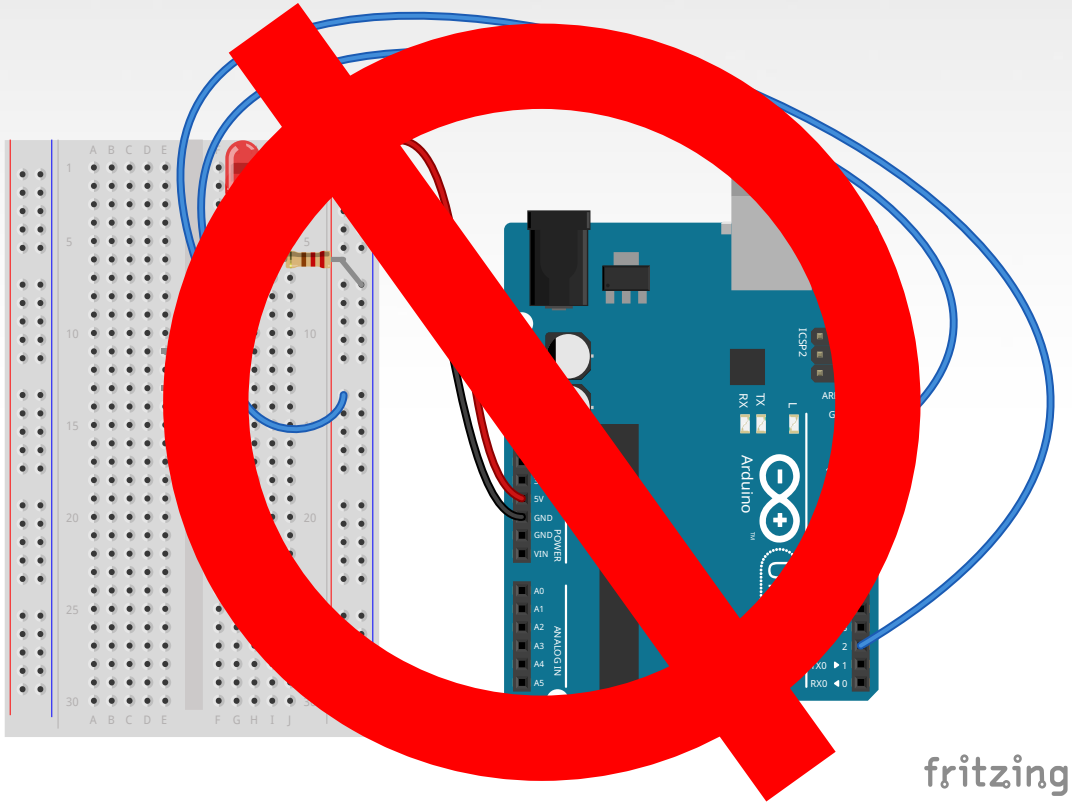
roto

Project 4 – Button/Maintained Switch Switch

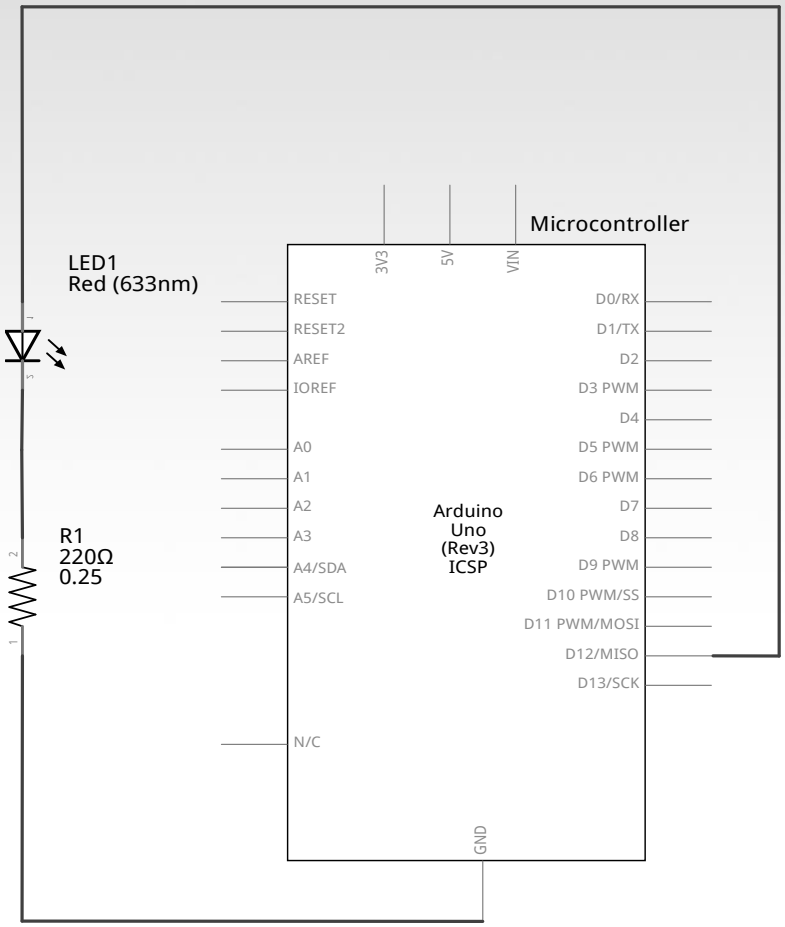
Wiring Diagram

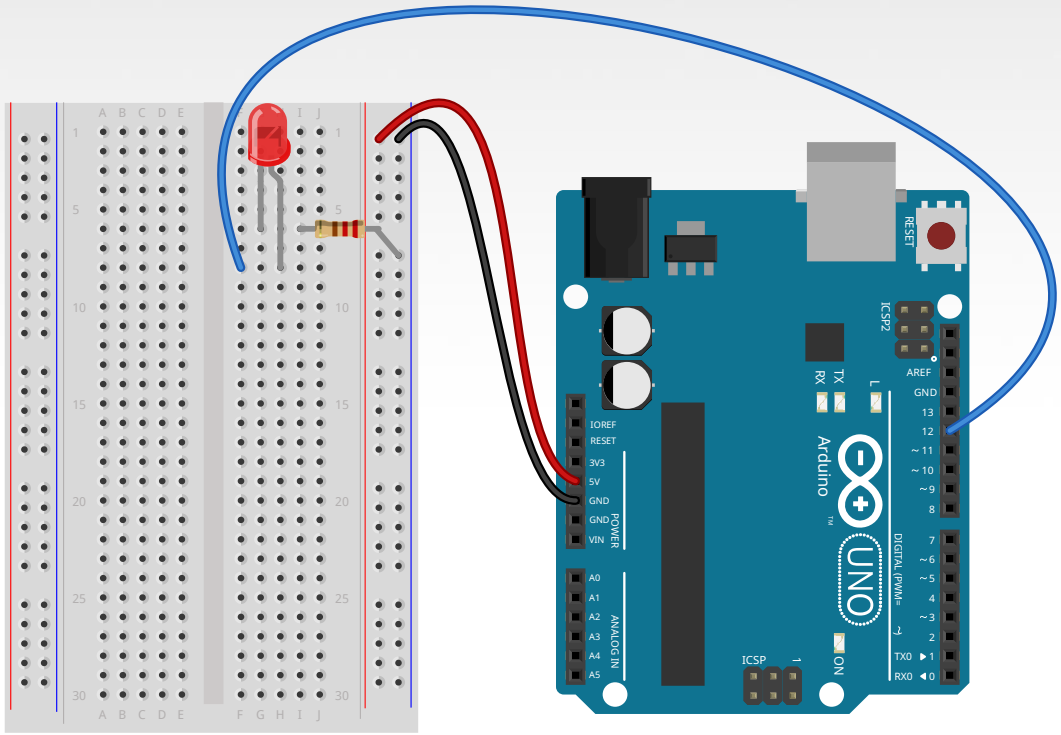
WRONG!!!



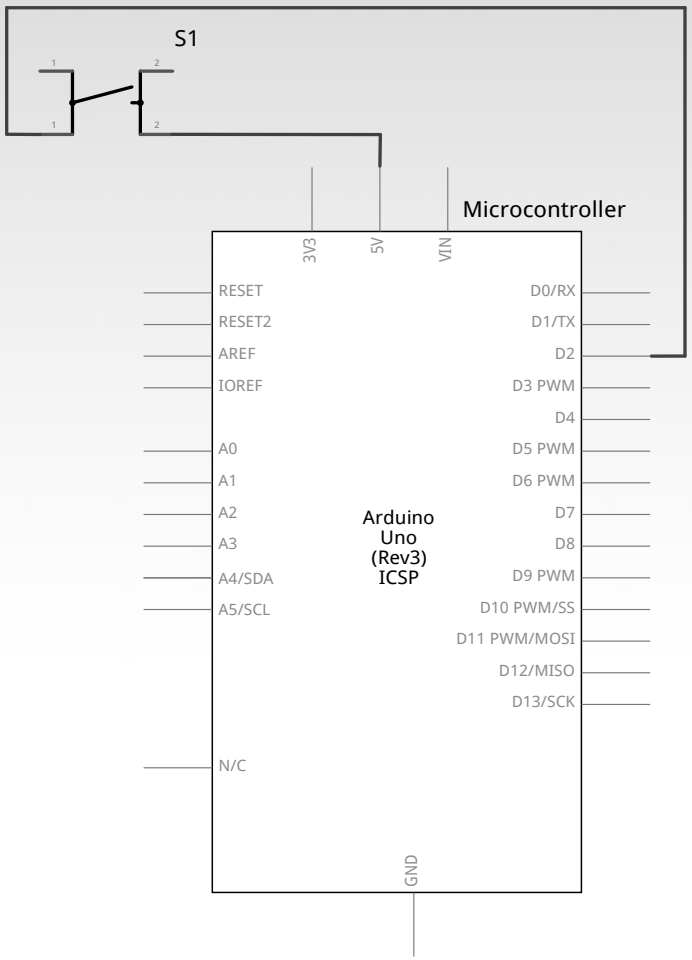


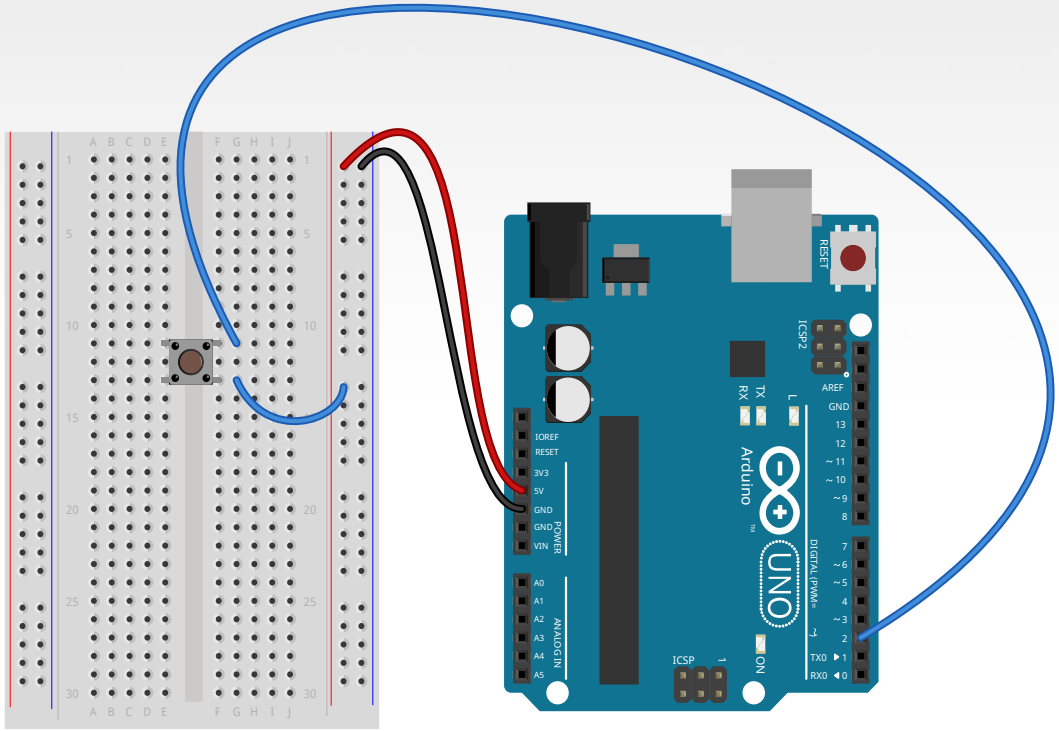
This Part Is Good





So This Must Be Bad...





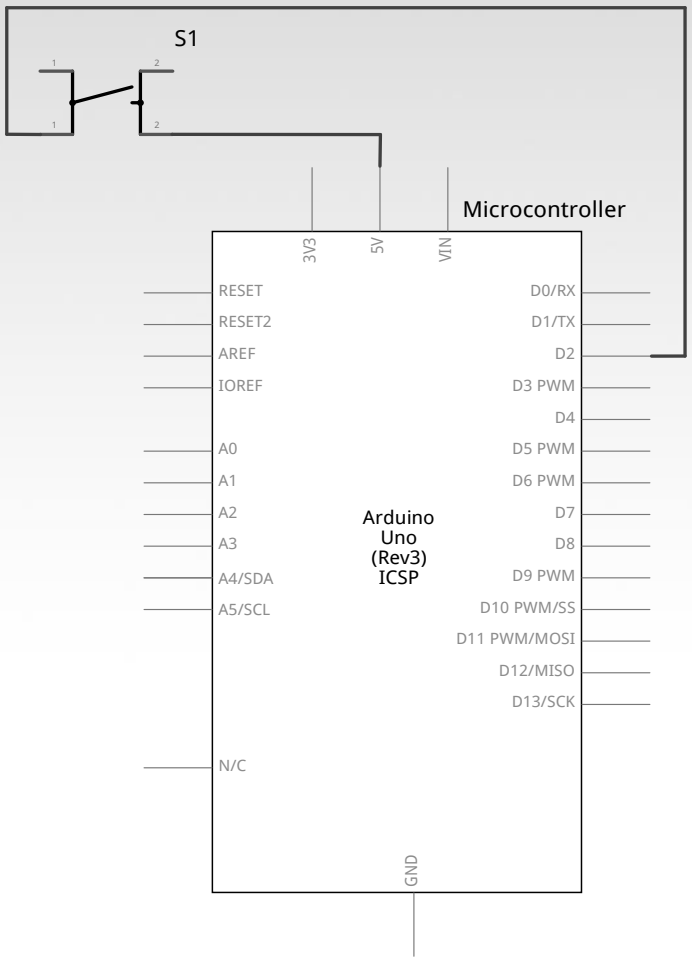
fritzing

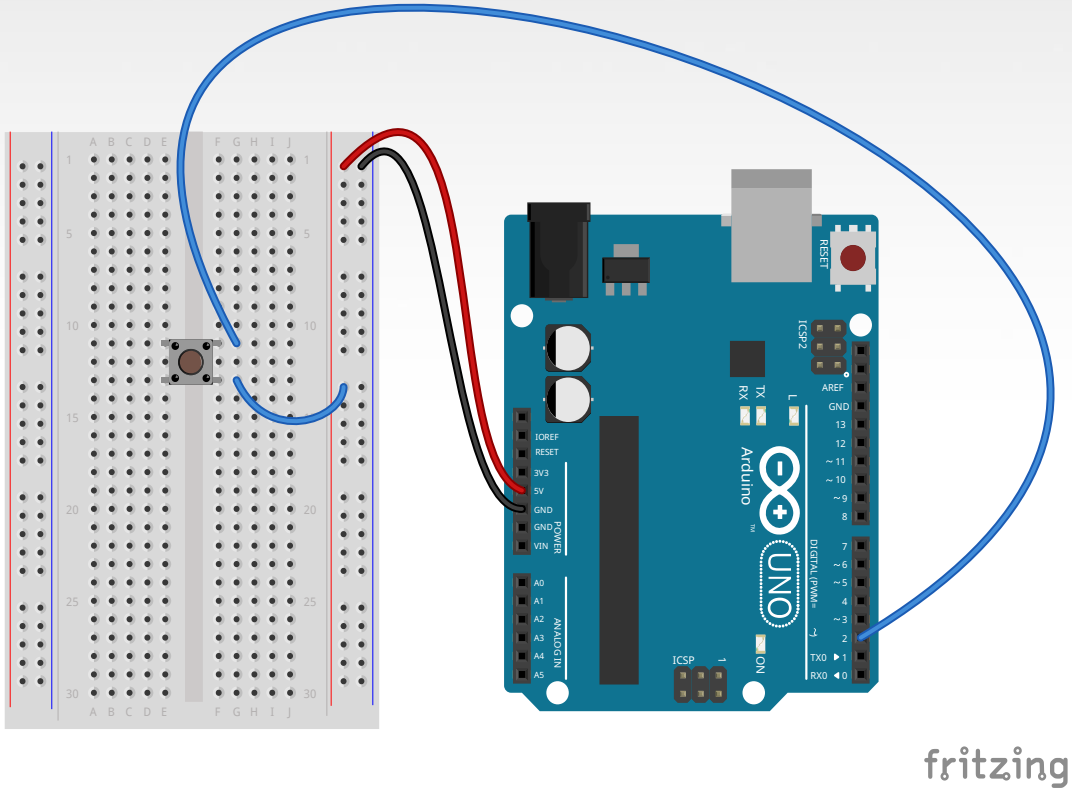
fritzing

roto

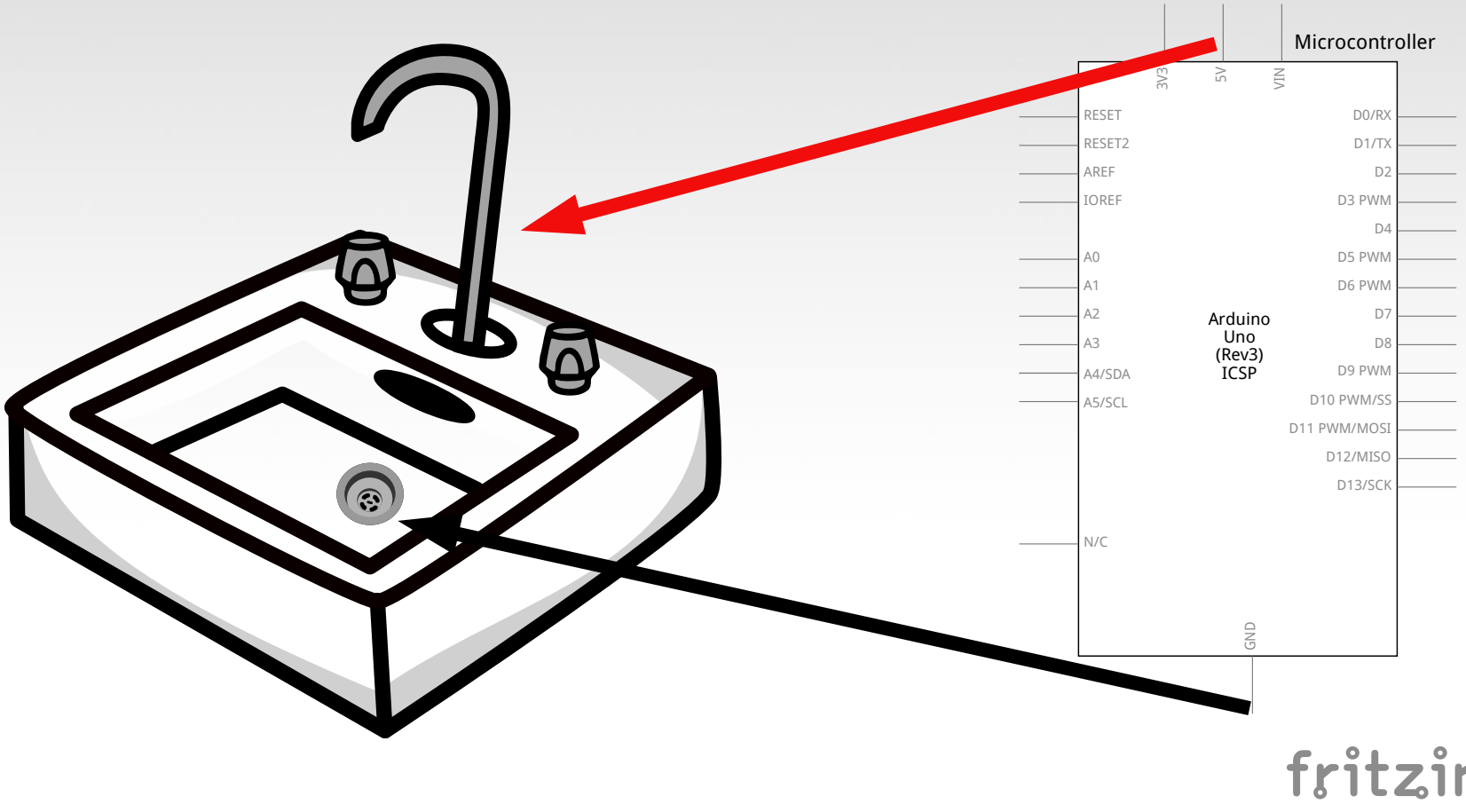
So This Must Be Bad...

But Why?!

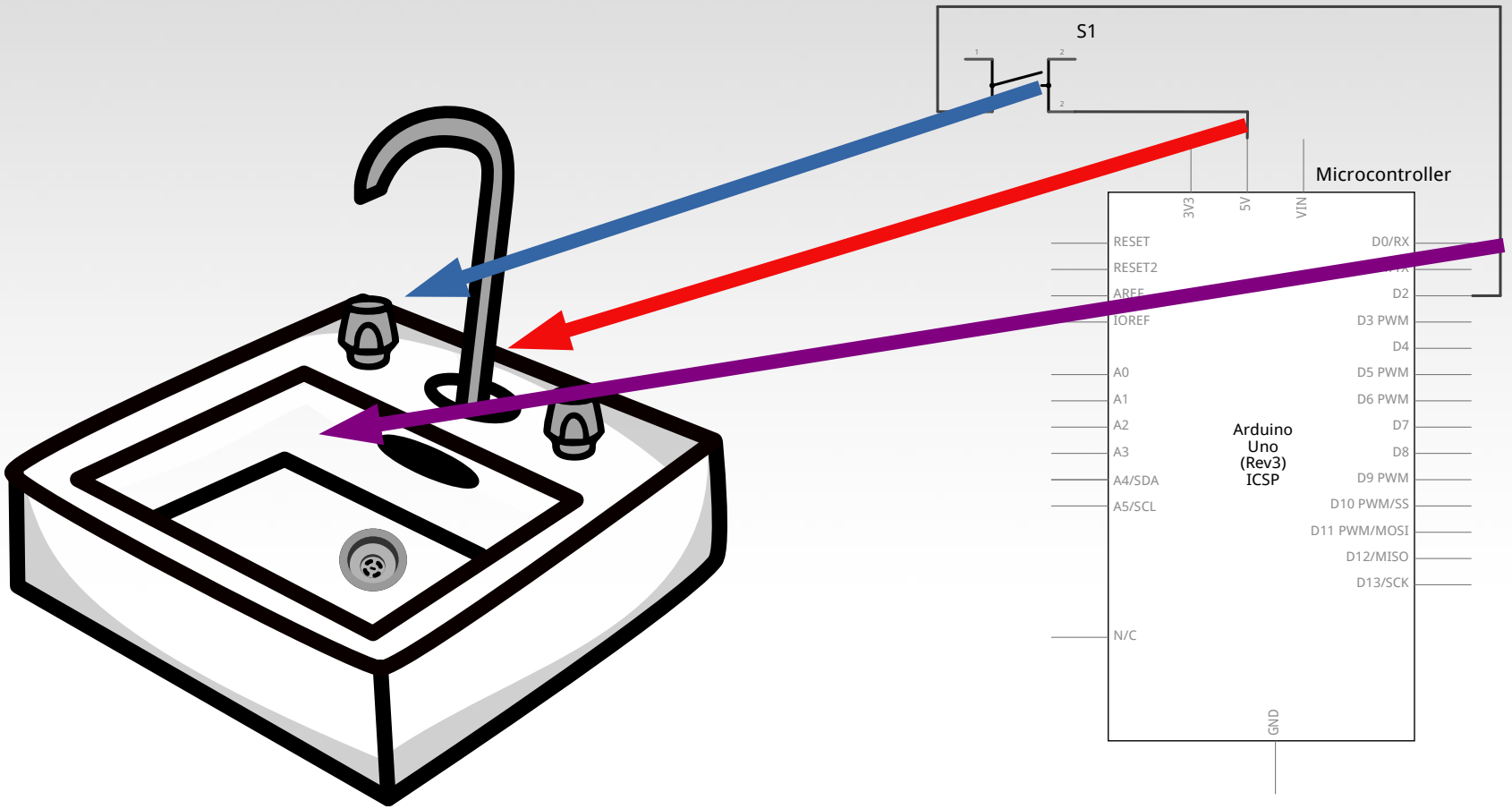




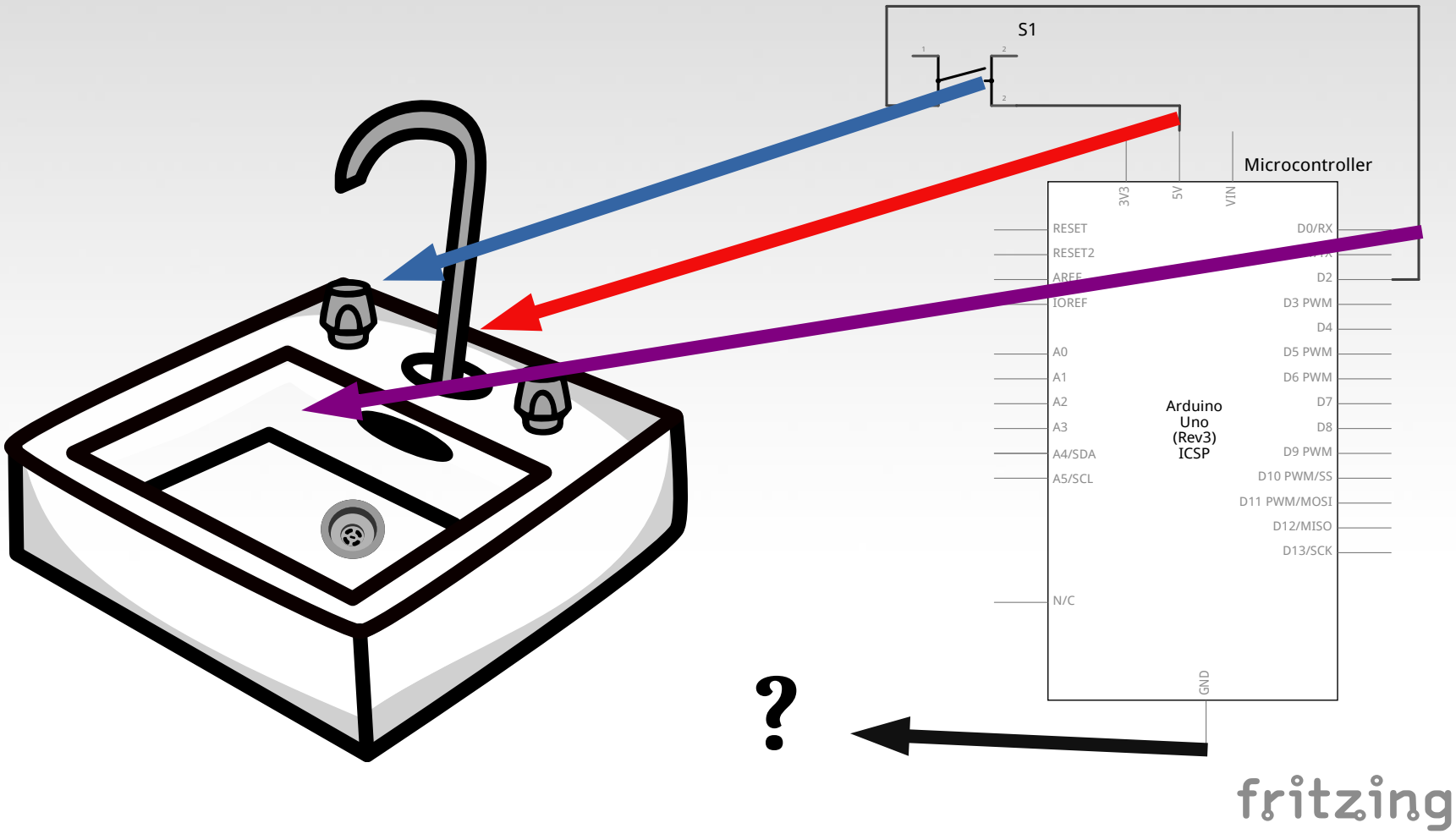
Another Water Analogy



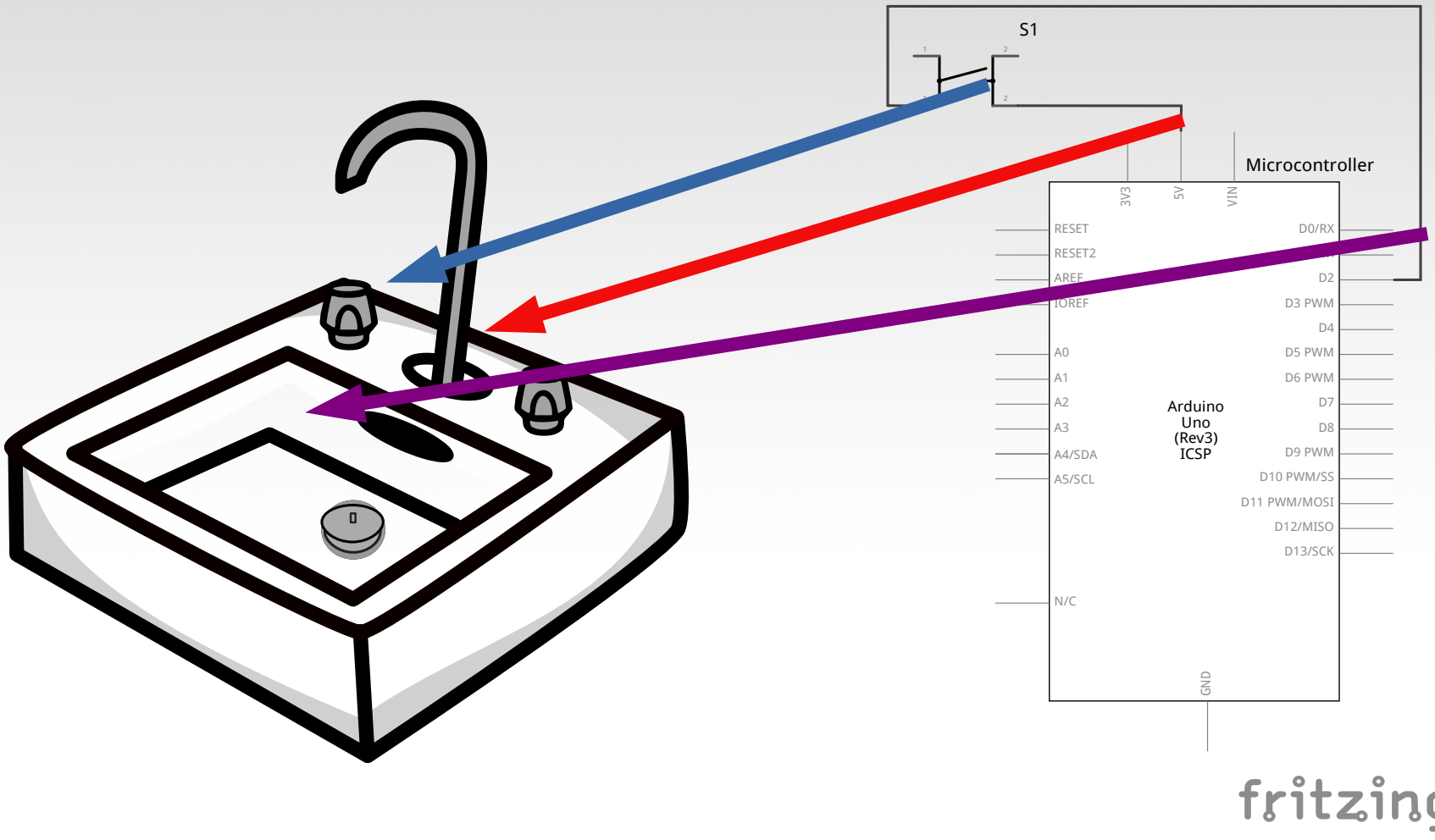
Analogy Continued...



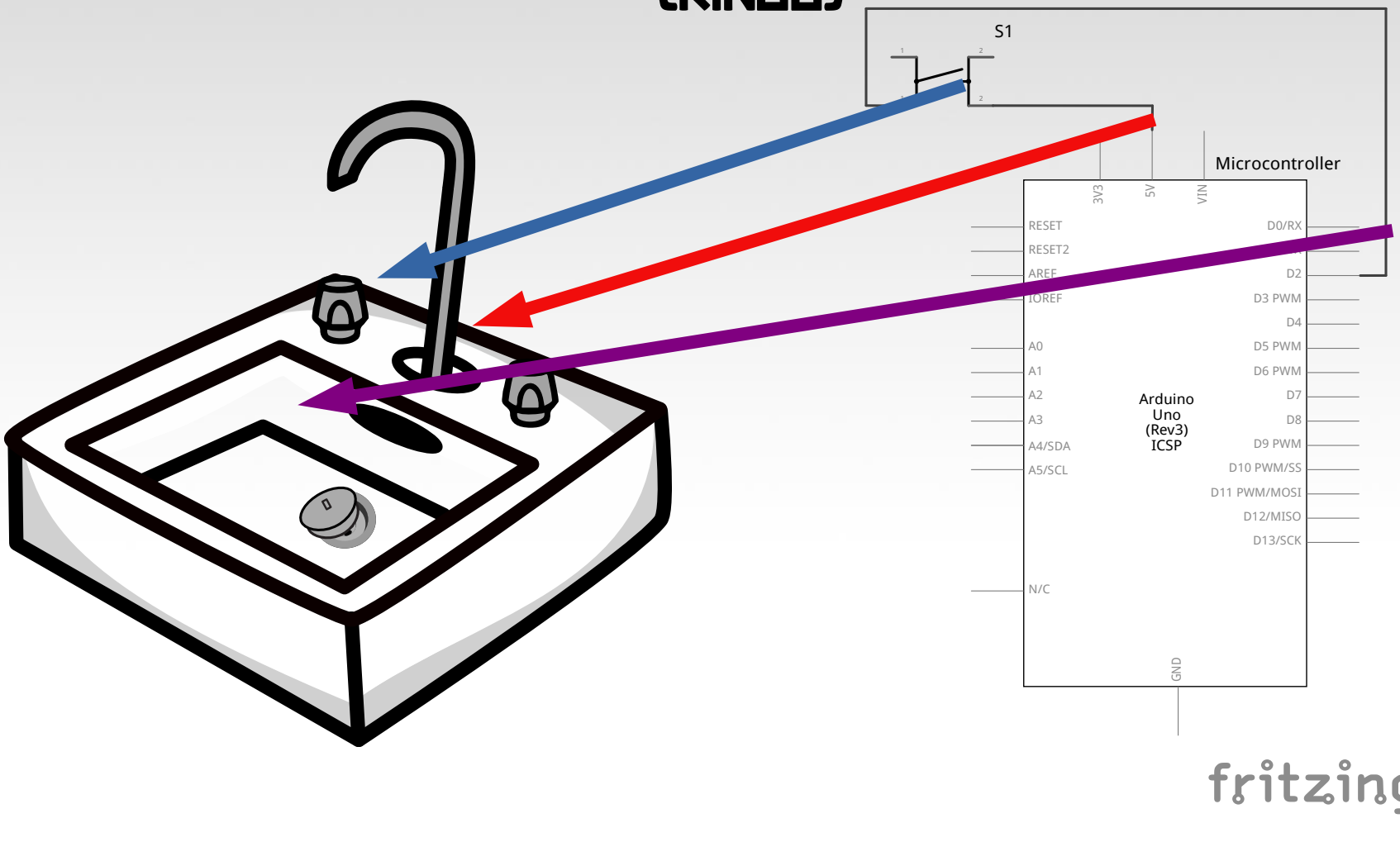
Analogy Continued...



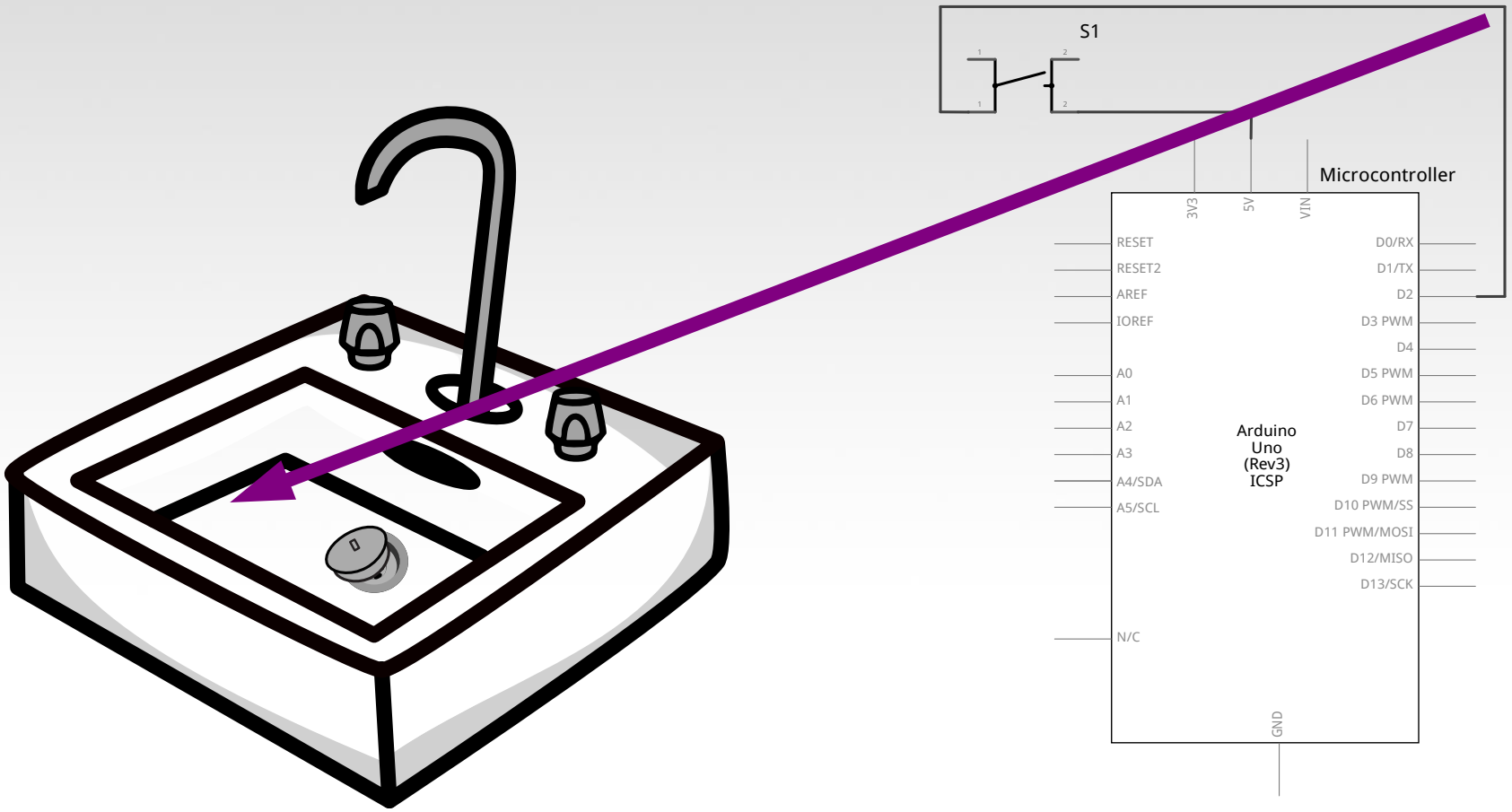
All Stopped Up



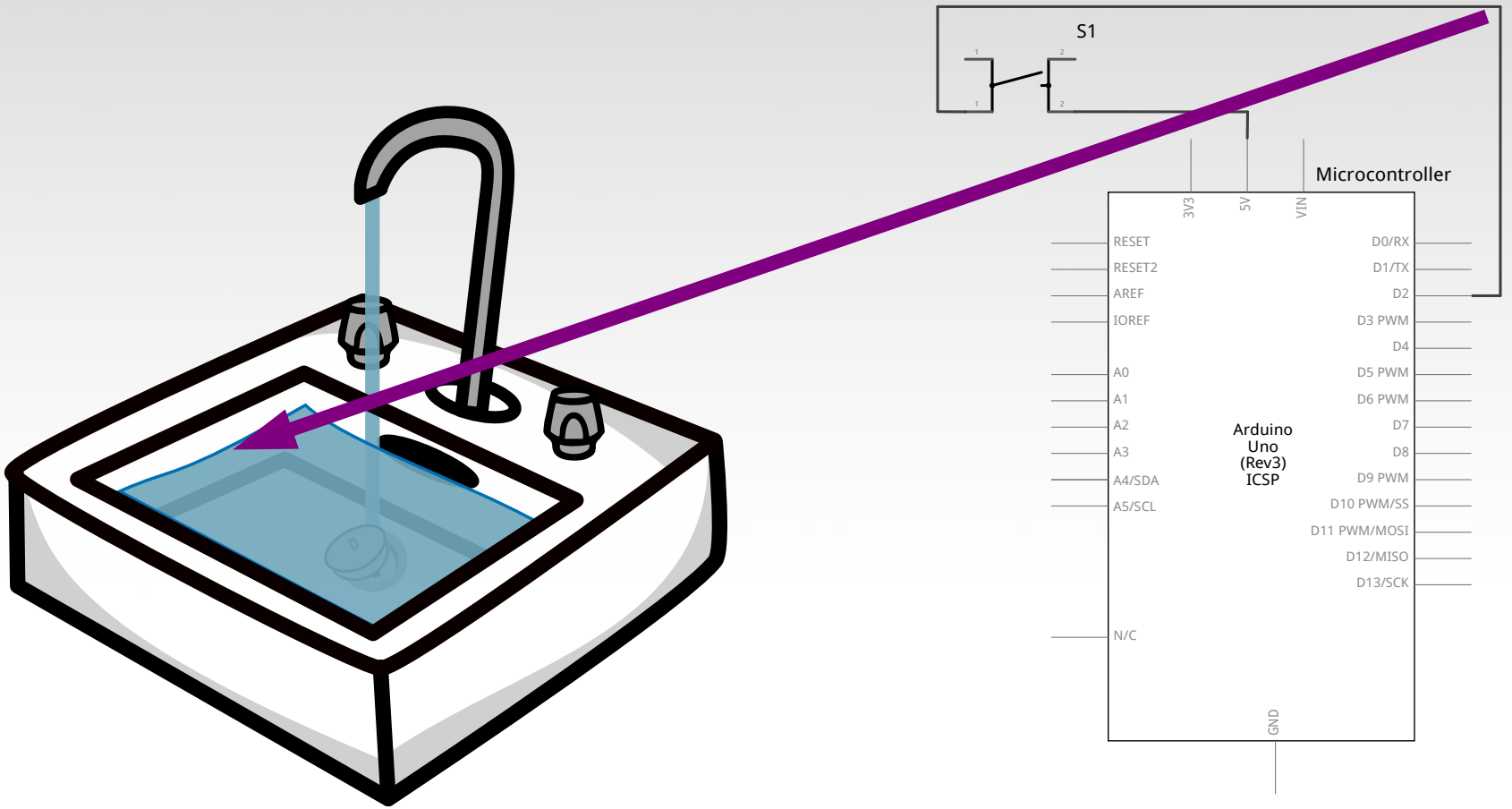
All Stopped Up (kinda)



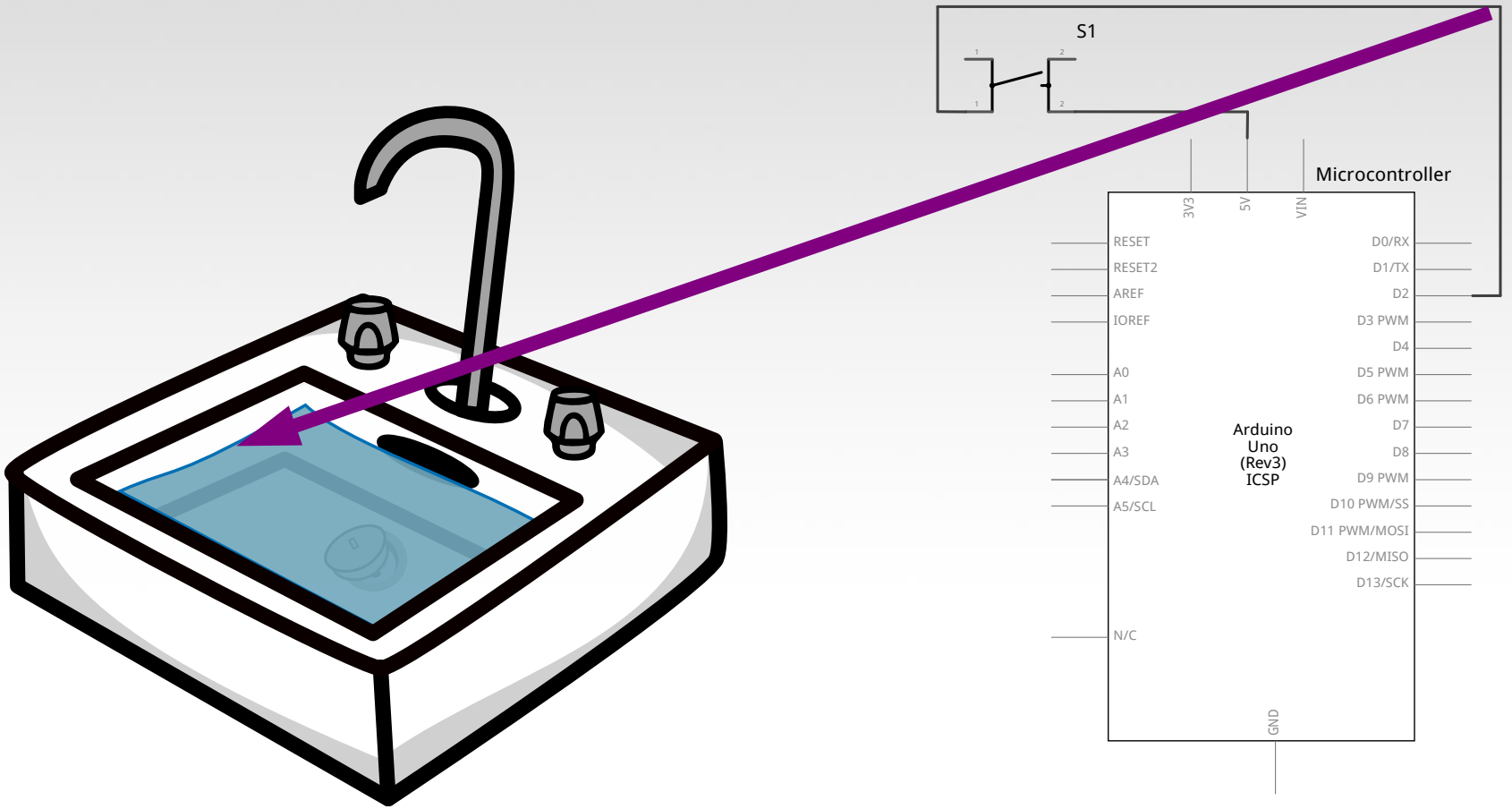
Switch Off



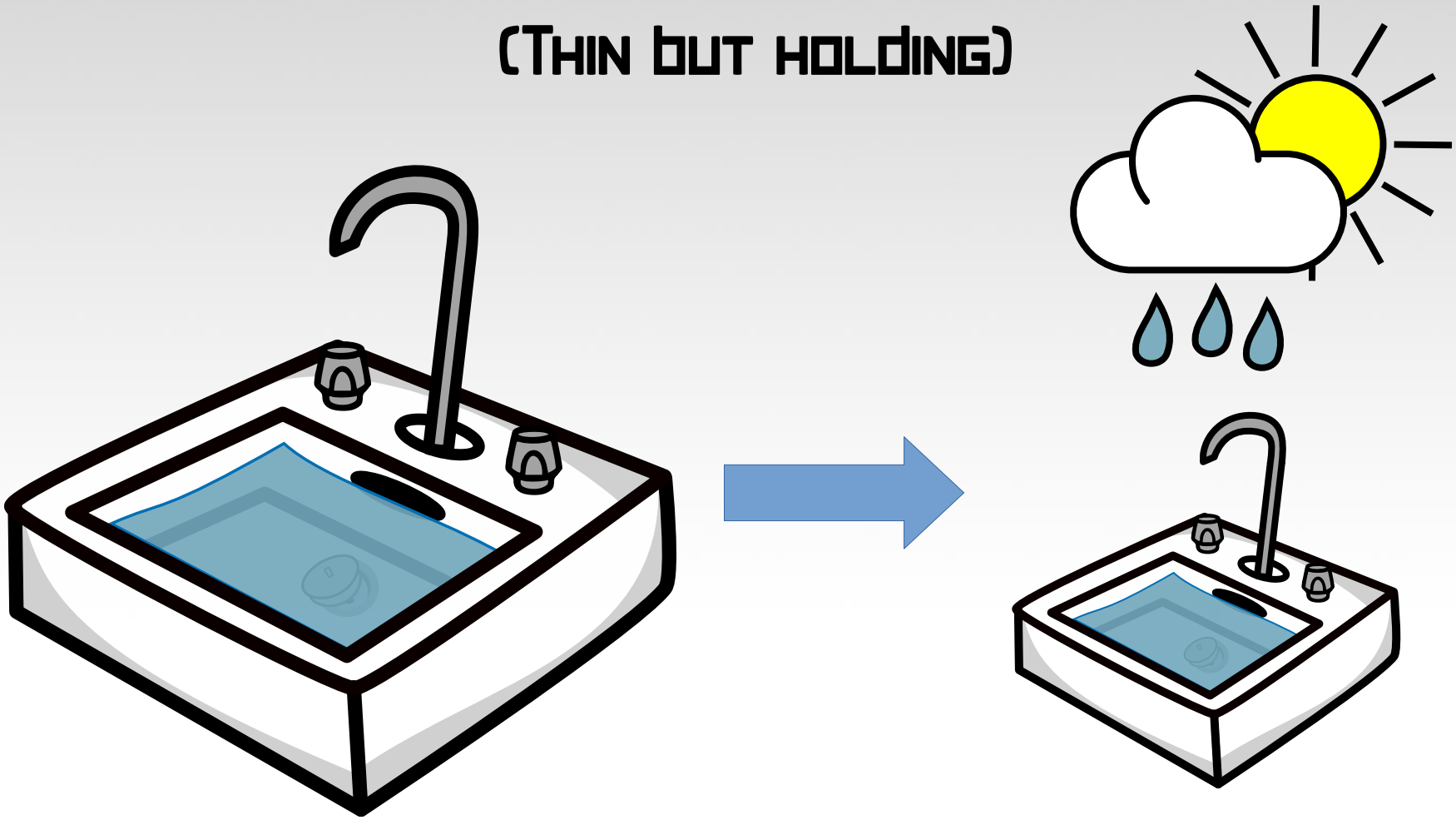
Switch On



Switch Off?!

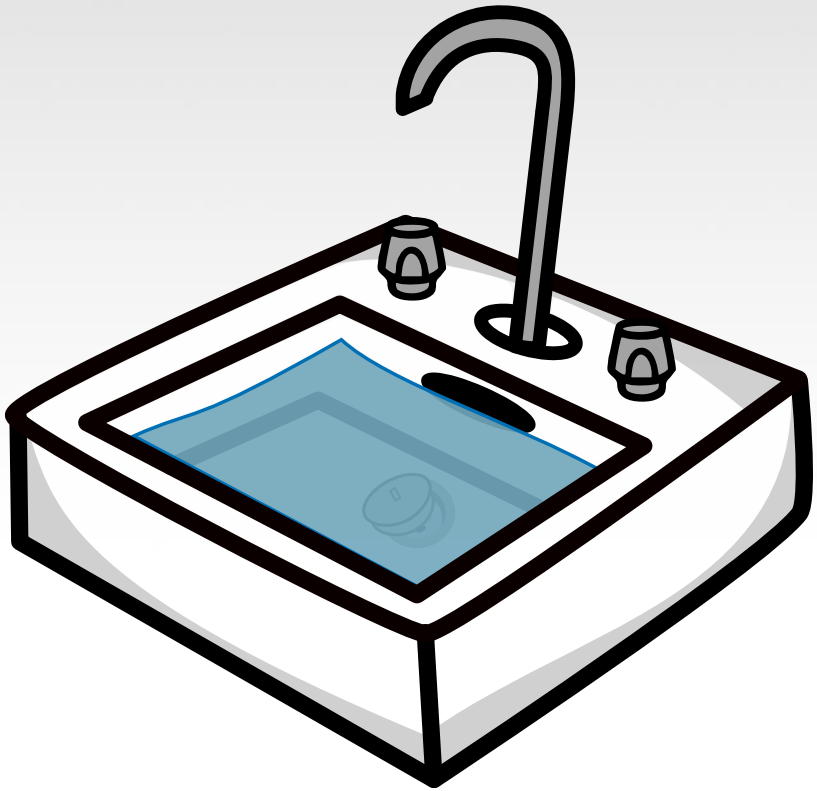


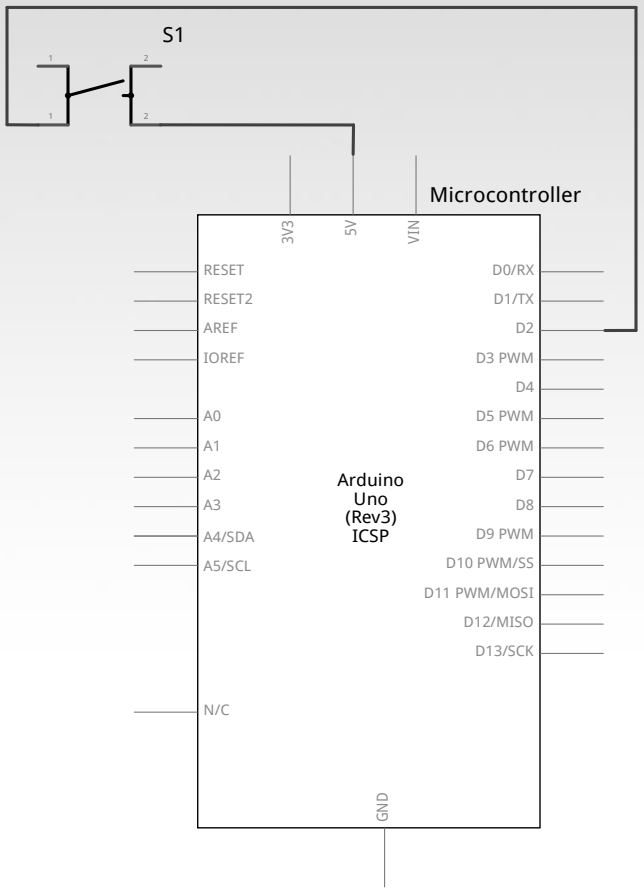
Analogy Extended
(Thin but holding)



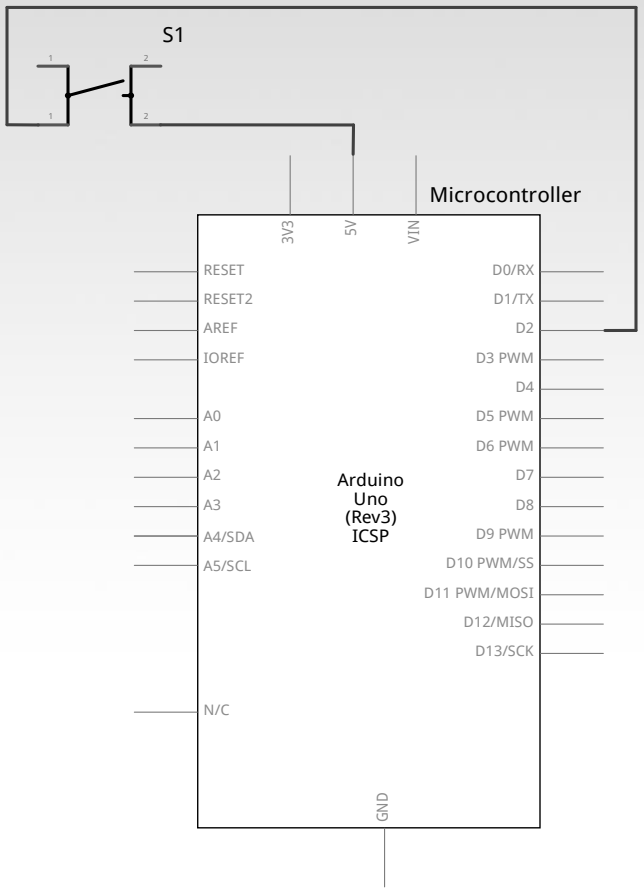
roto

So This Approach...

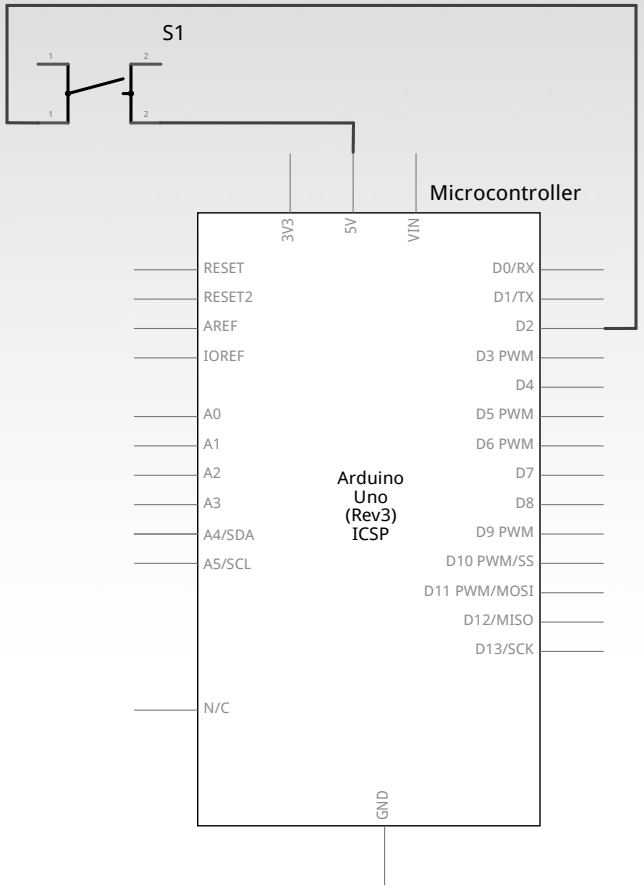




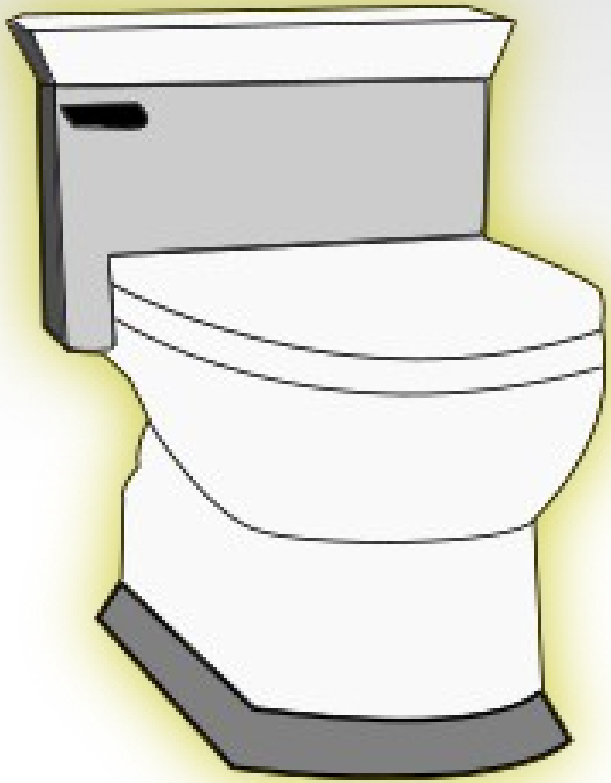
Doesn't Work!



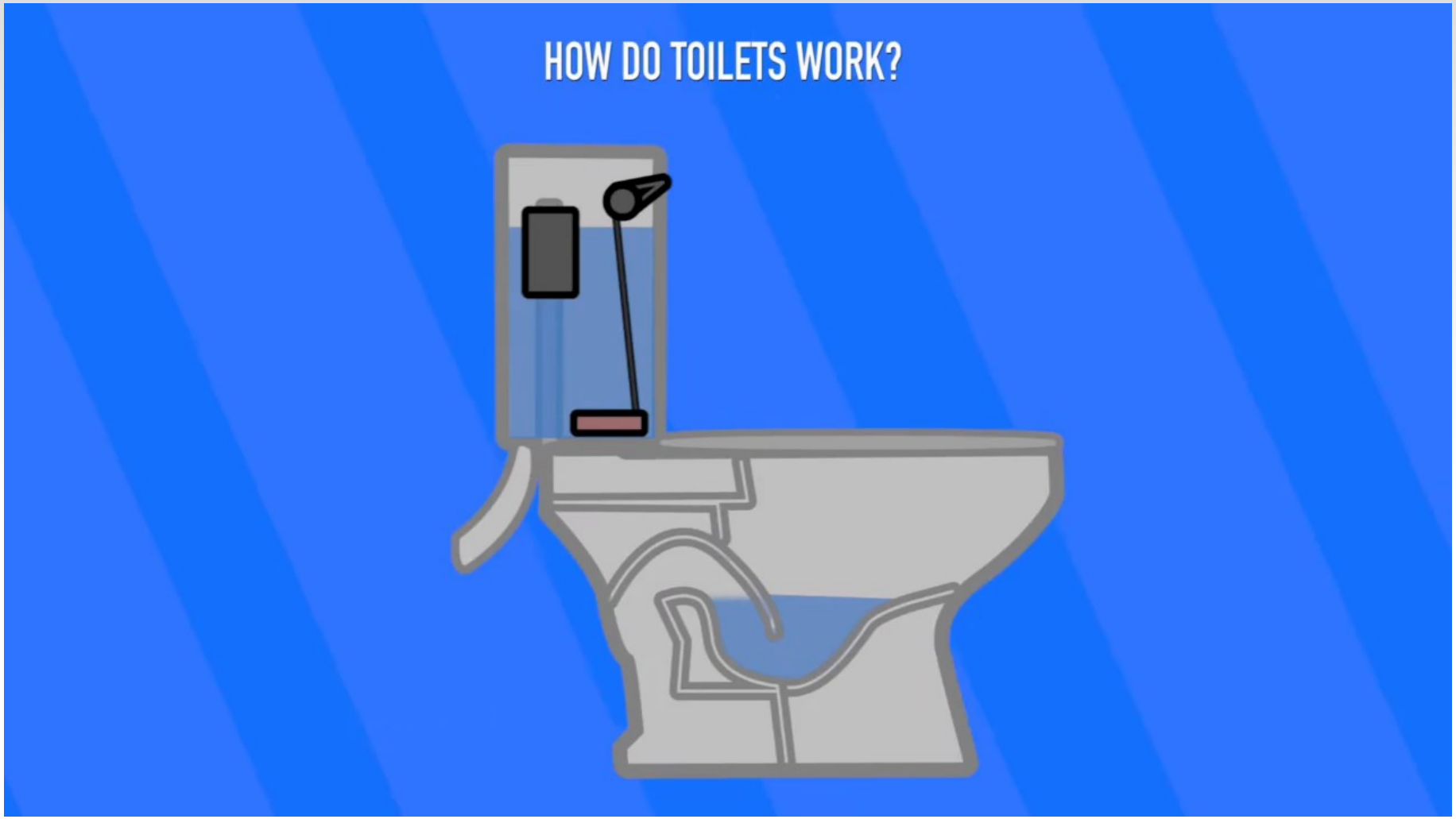
What About...



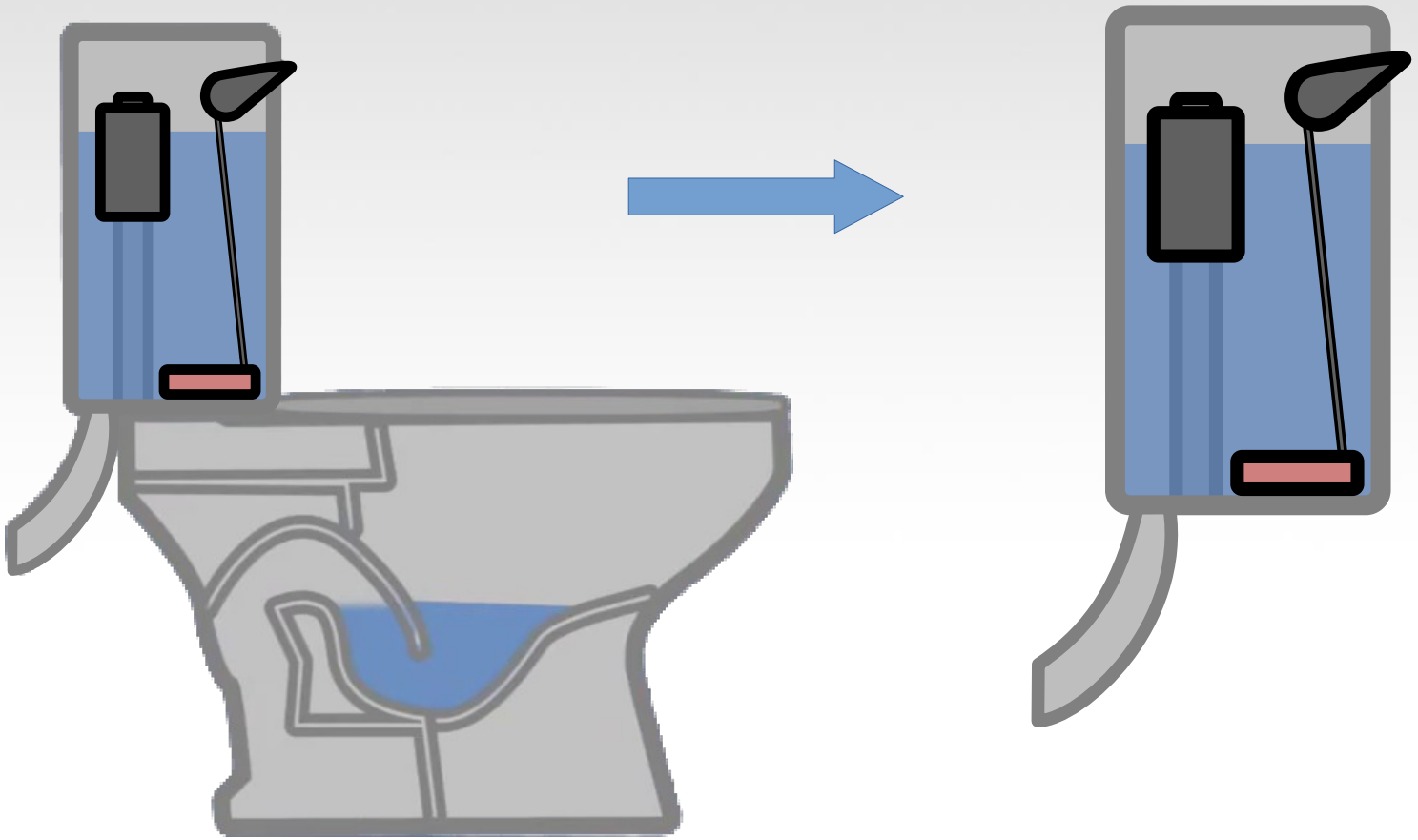
Changing Bathroom Fixtures?



roto

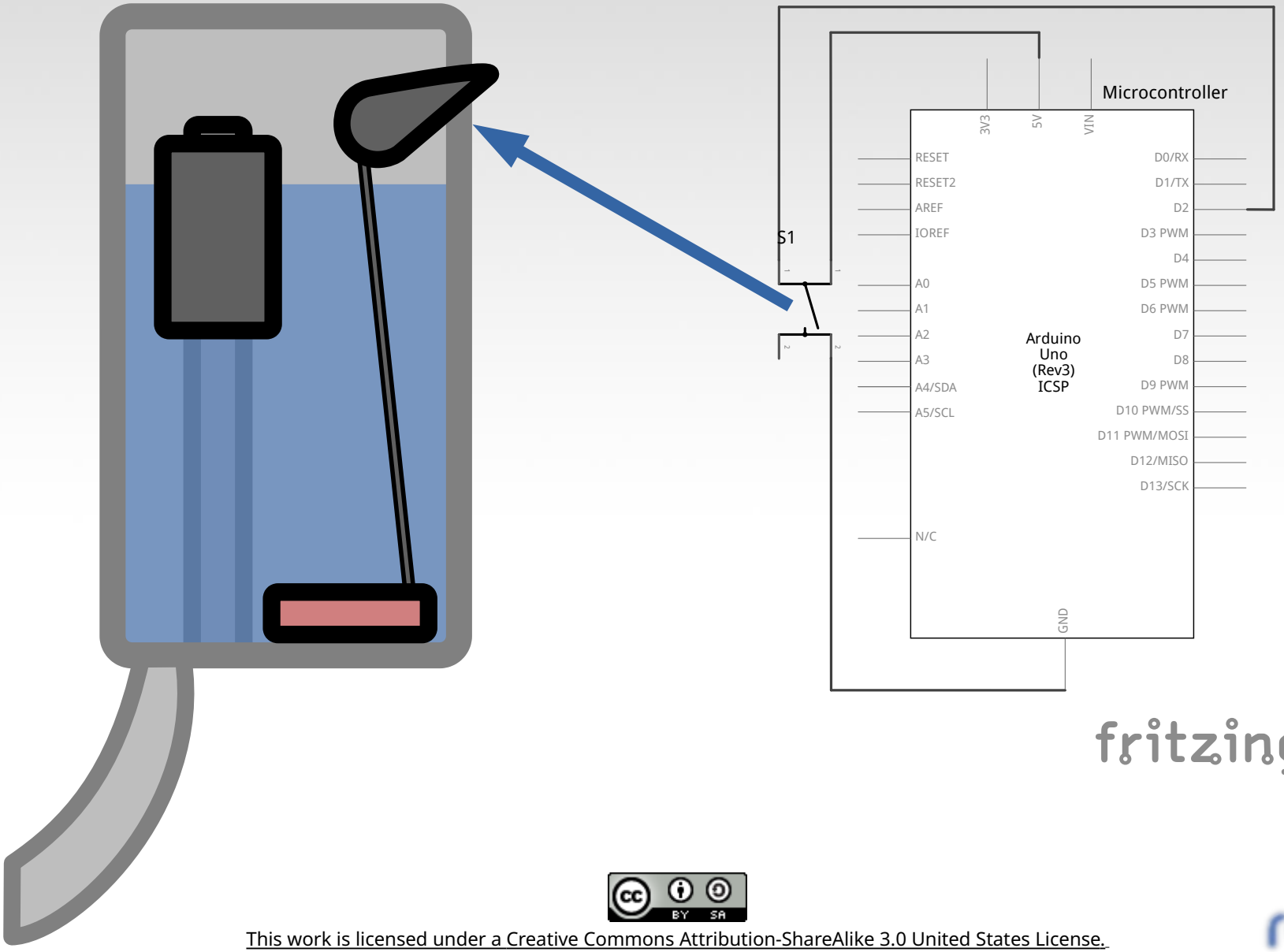


Getting Rid of the Unnecessary Crap

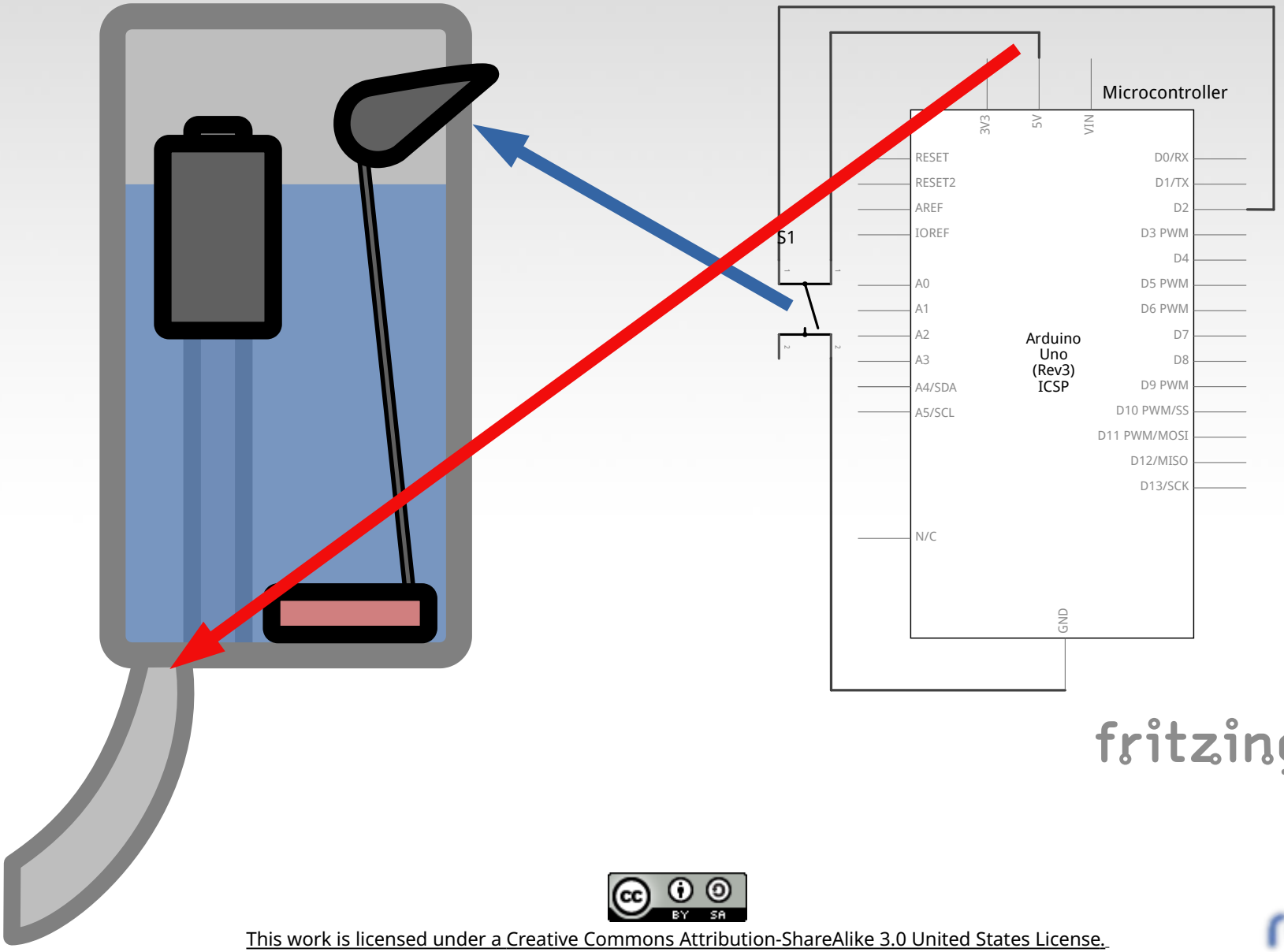


roto

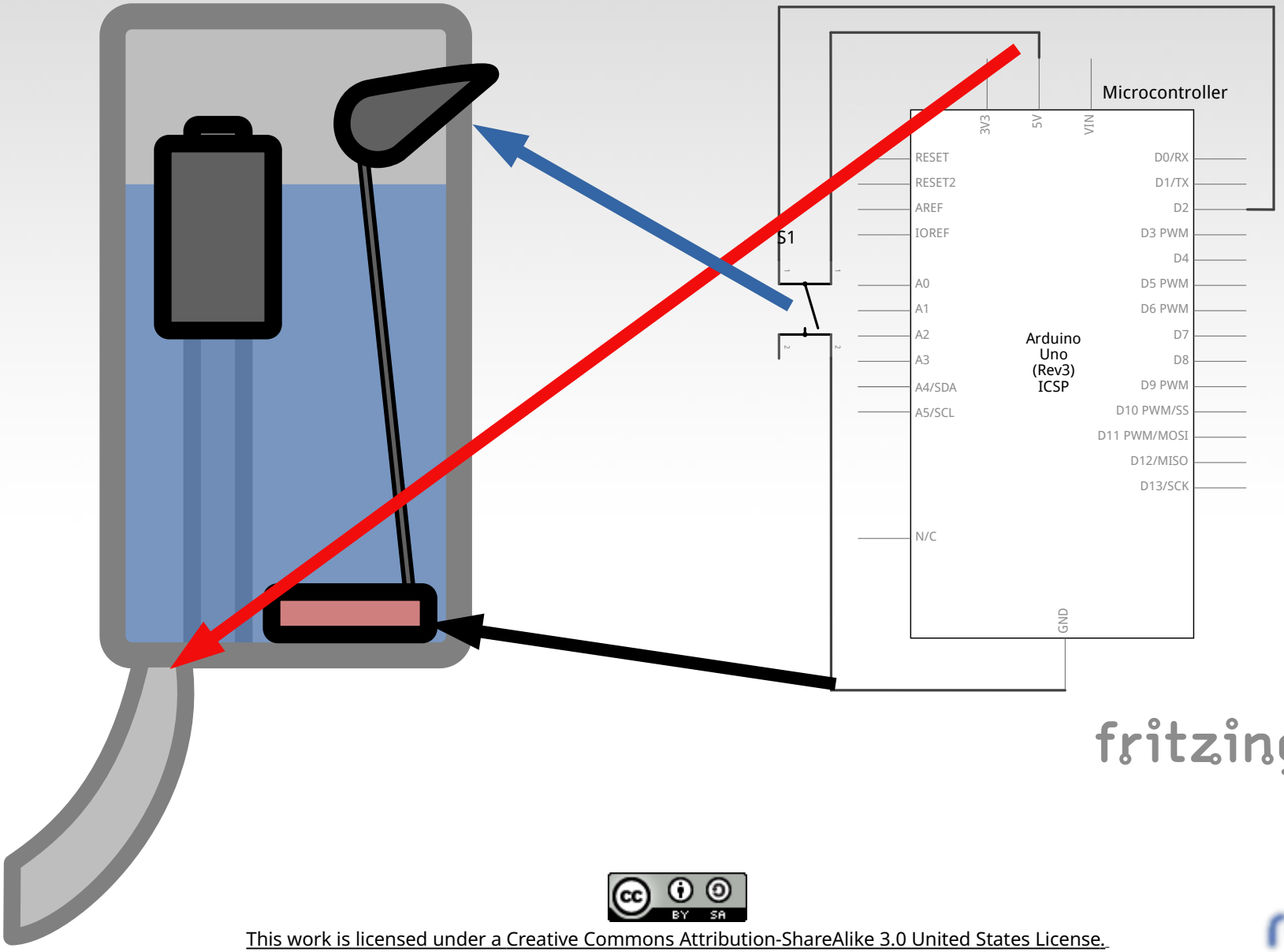
Changing the Viewpoint



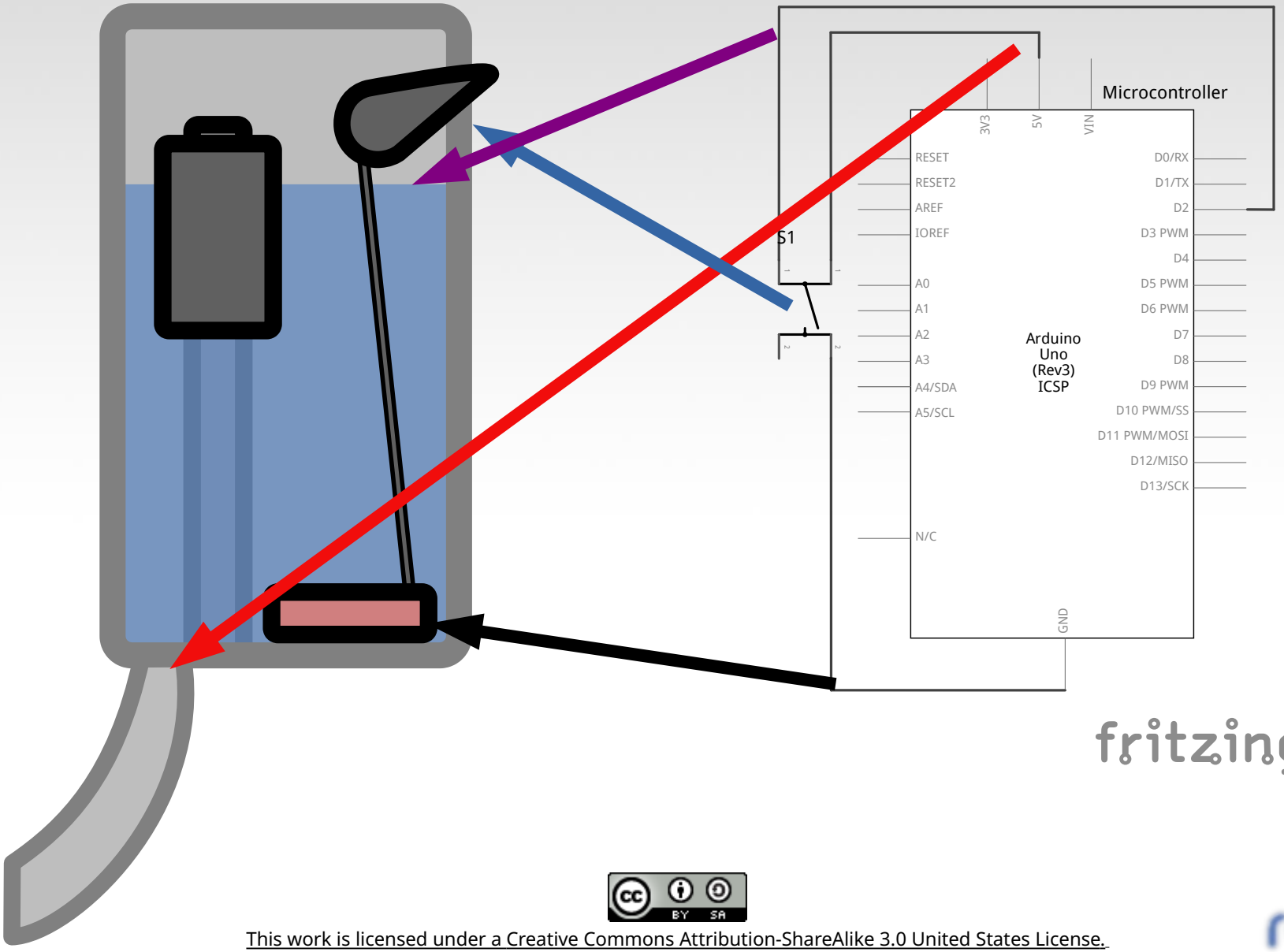
Changing the Viewpoint



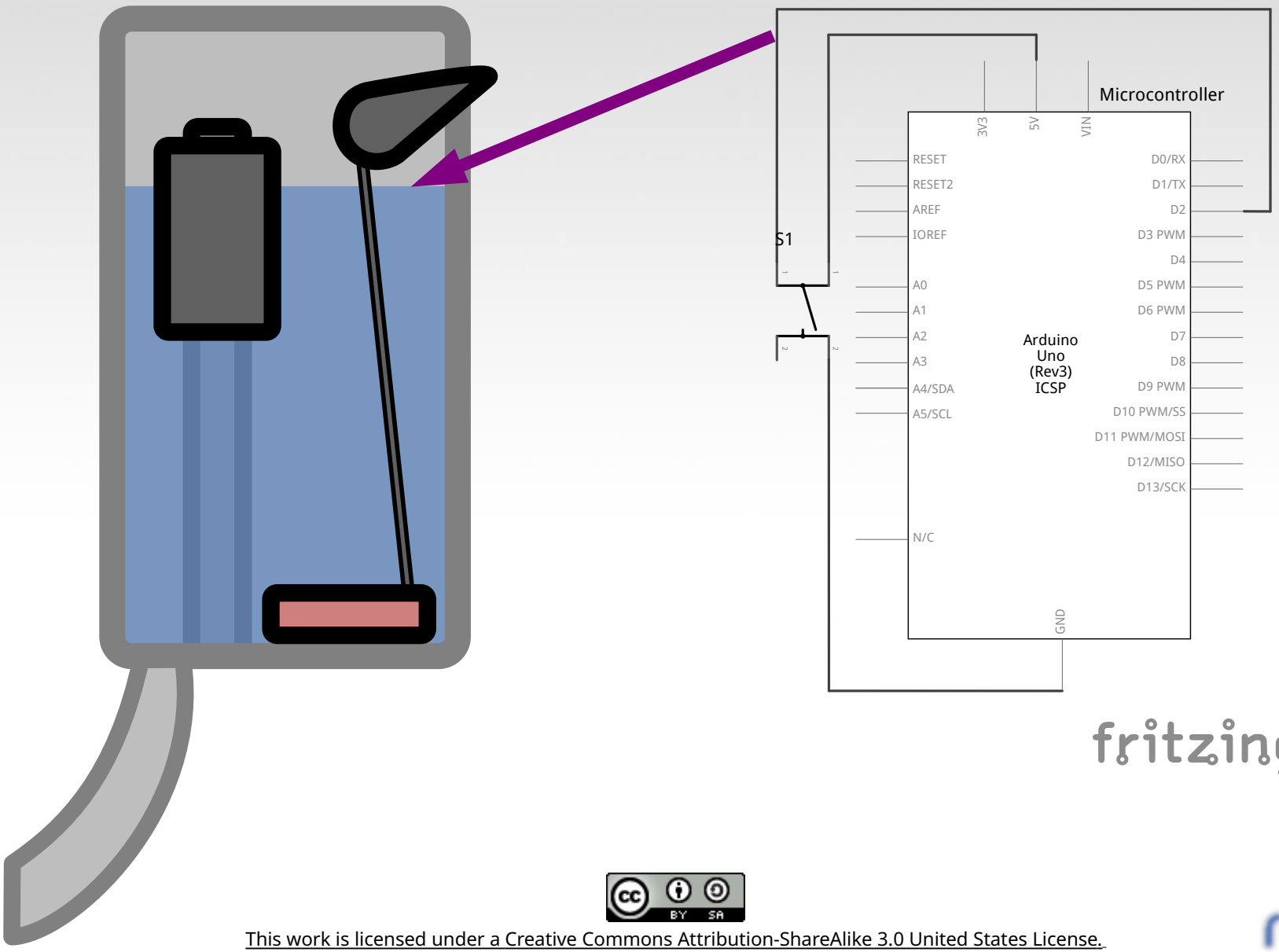
Changing the Viewpoint



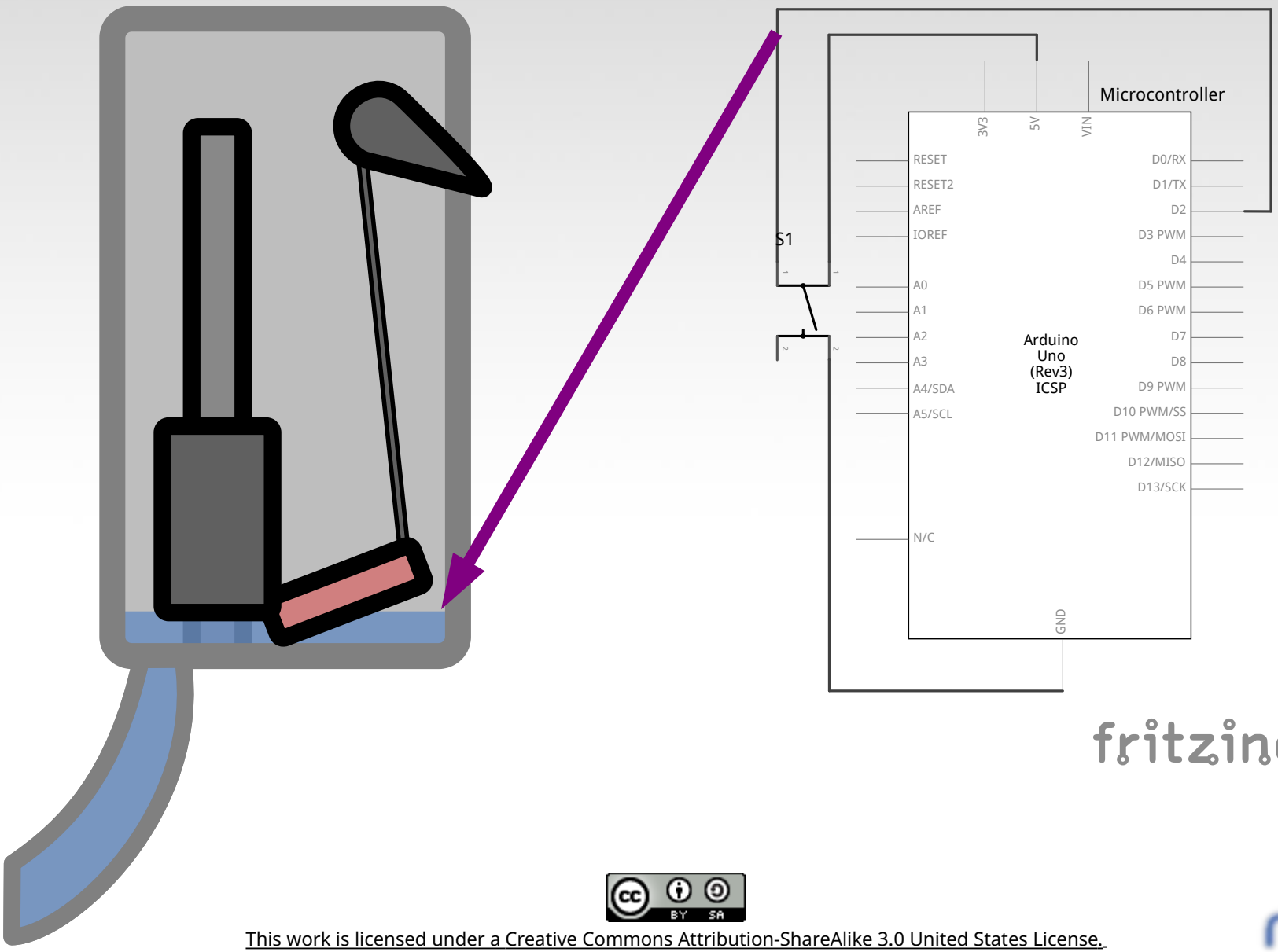
Changing the Viewpoint



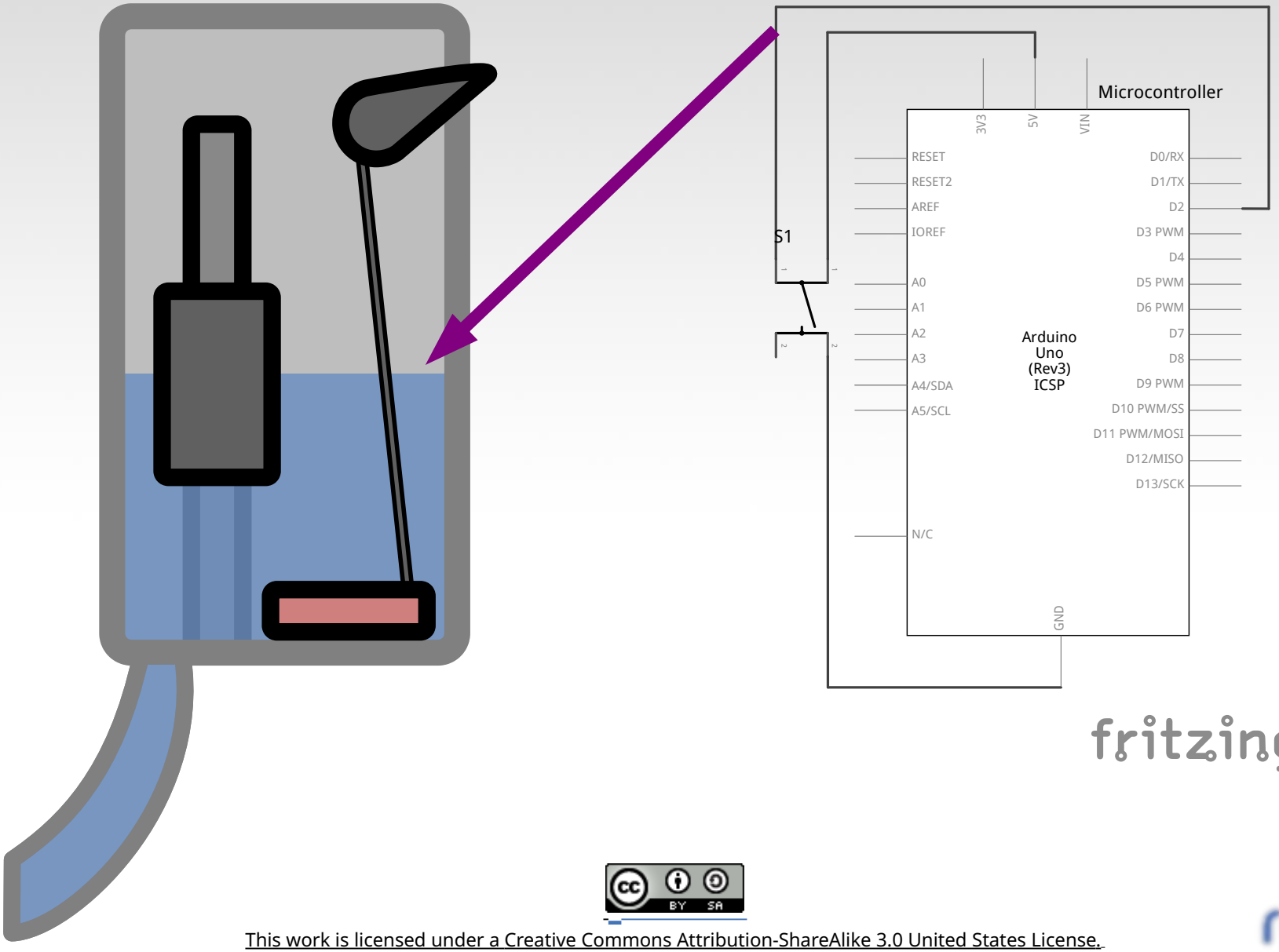
Switch Off



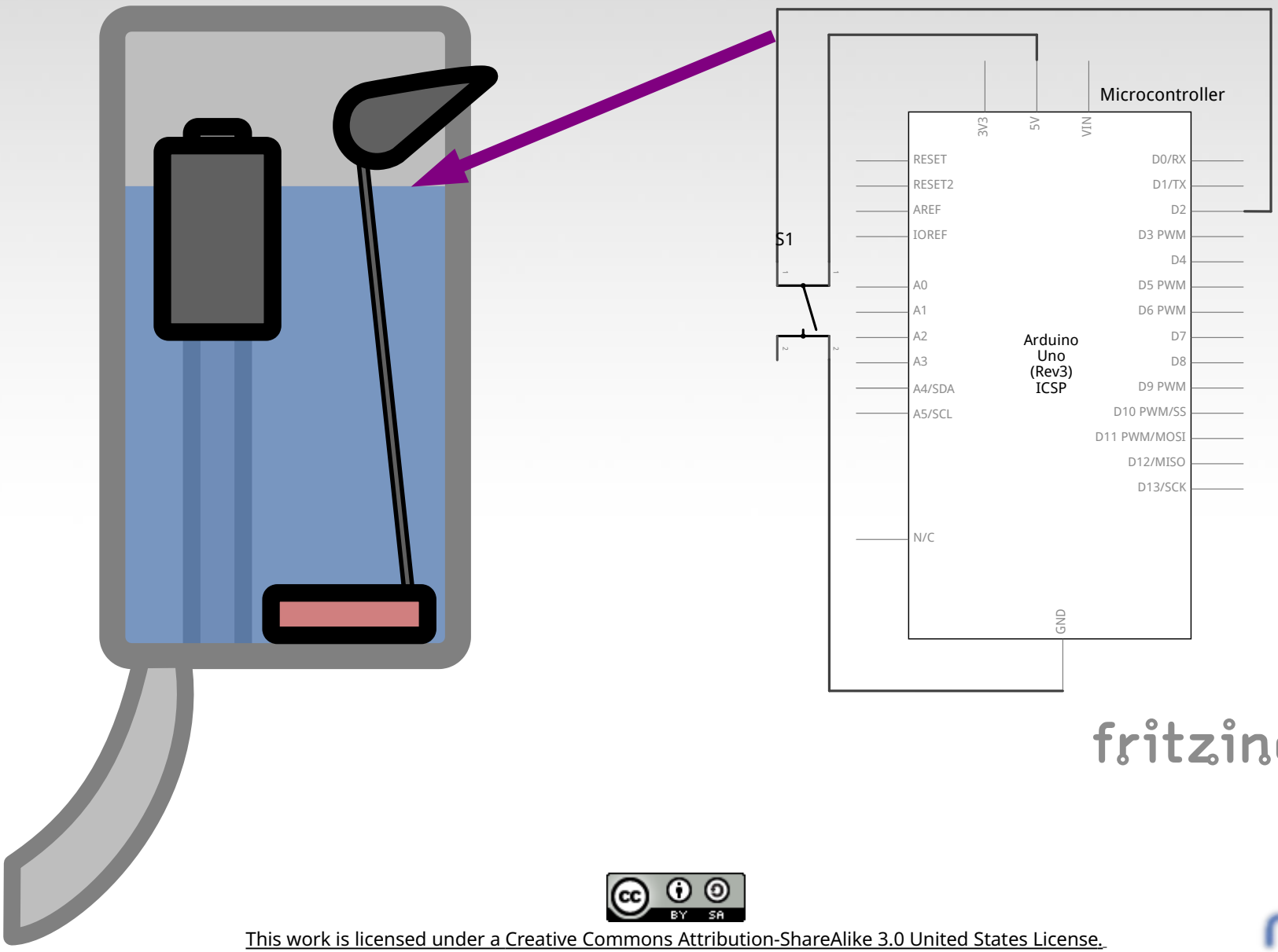
Switch On



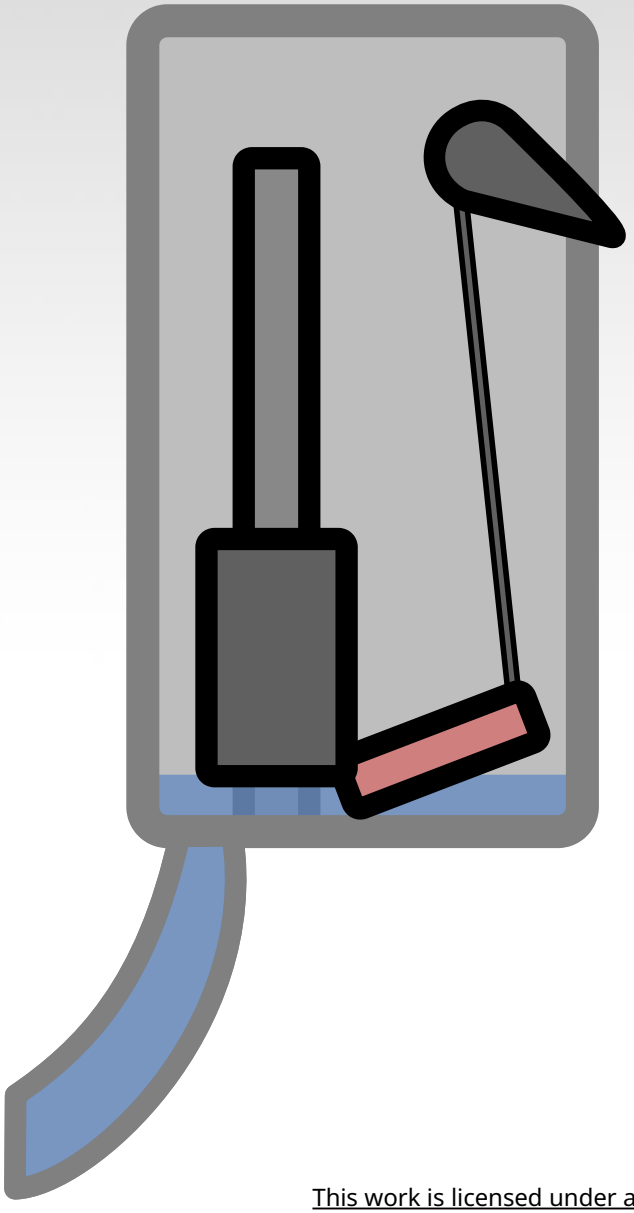
Switch Right After Off

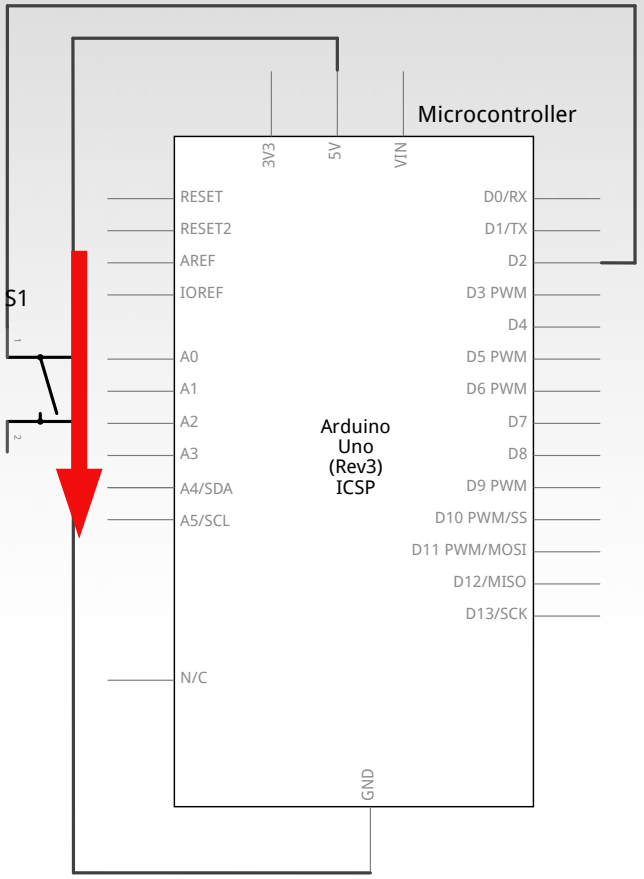


Switch Off



Stop Playin' With the Commode!!

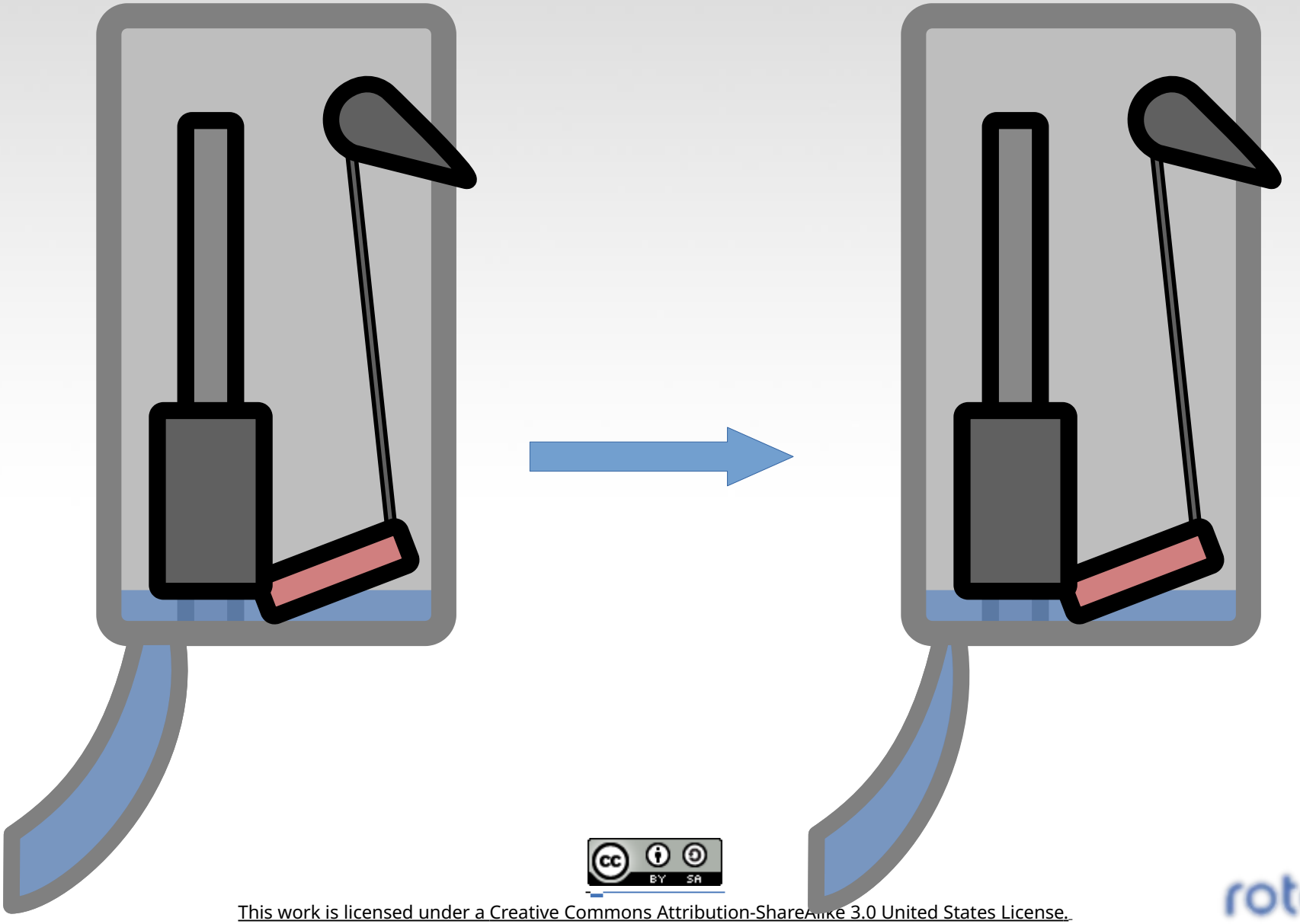




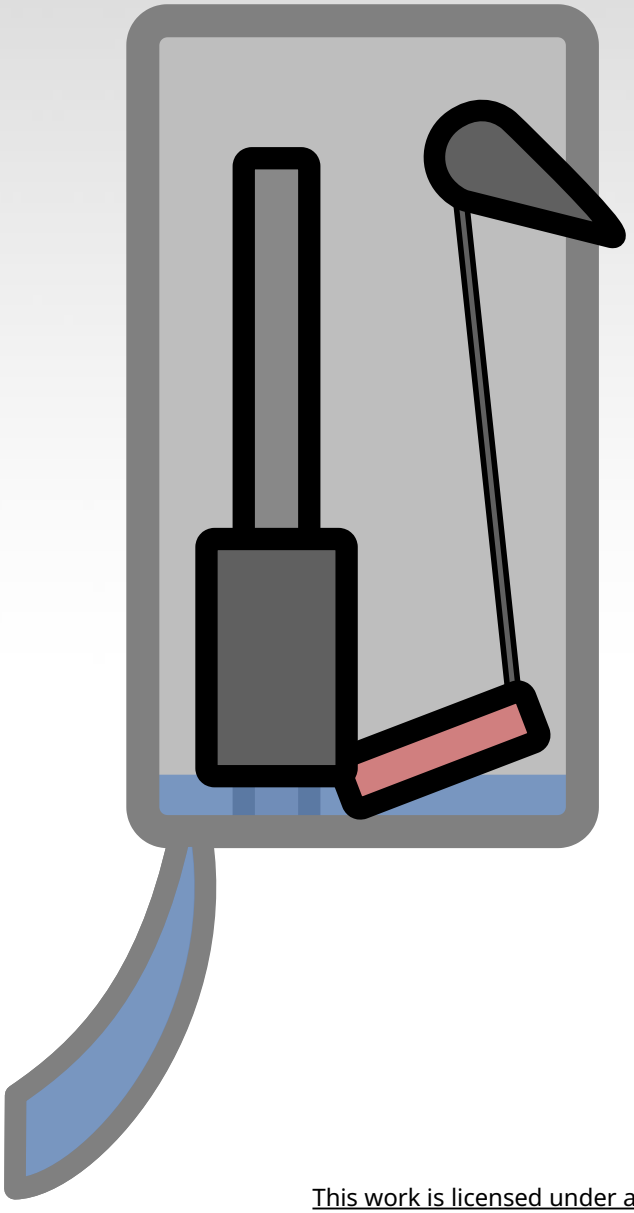
fritzing

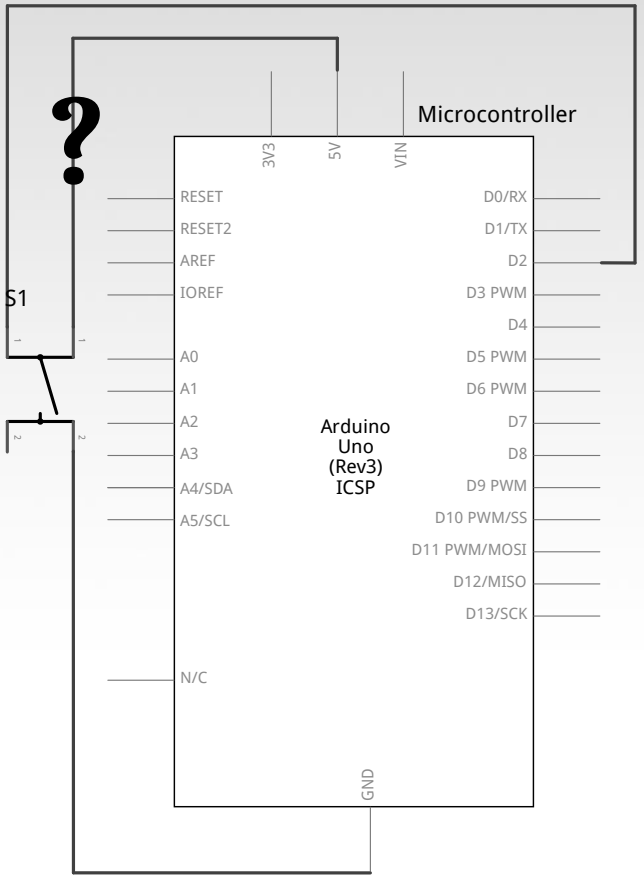
roto

Possible Solution



How Do We Get Here?



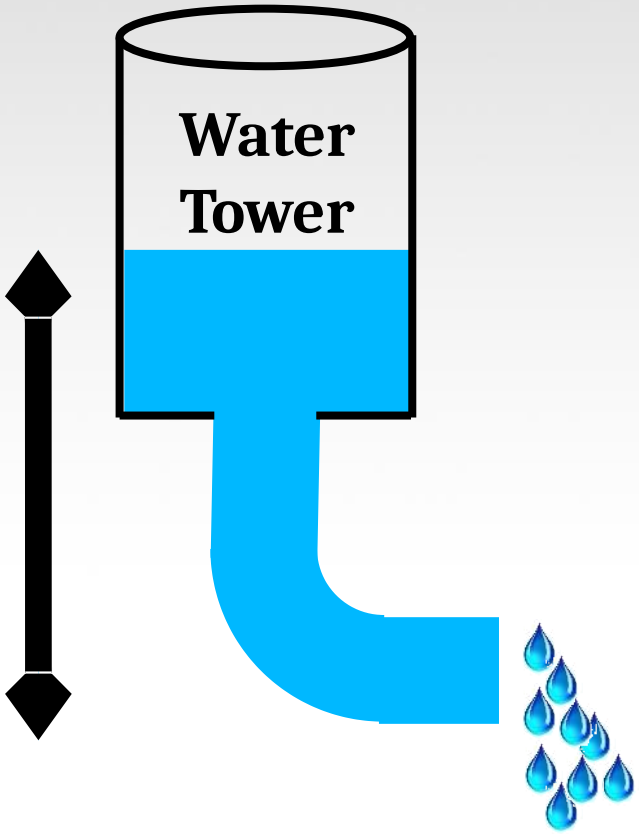


fritzing

roto

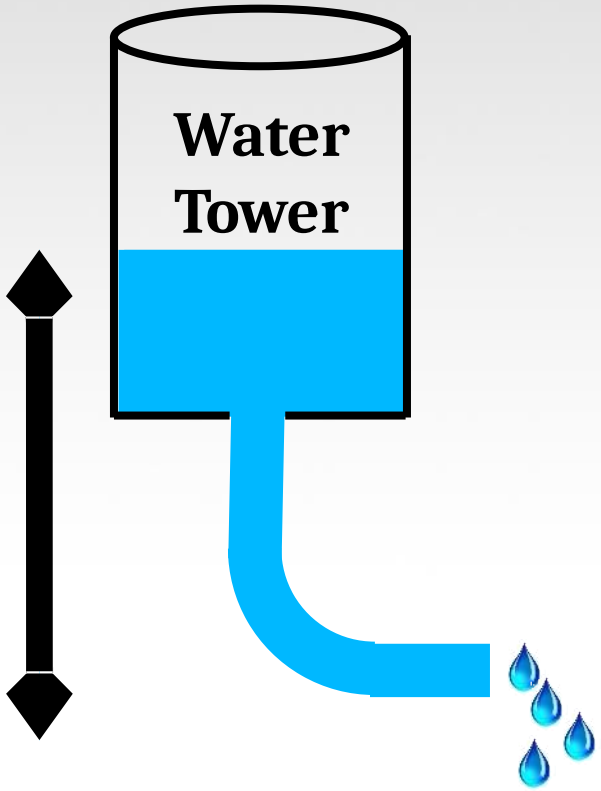
$$V = I R$$

Resistance Analogy



Big Pipe == Lower Resistance

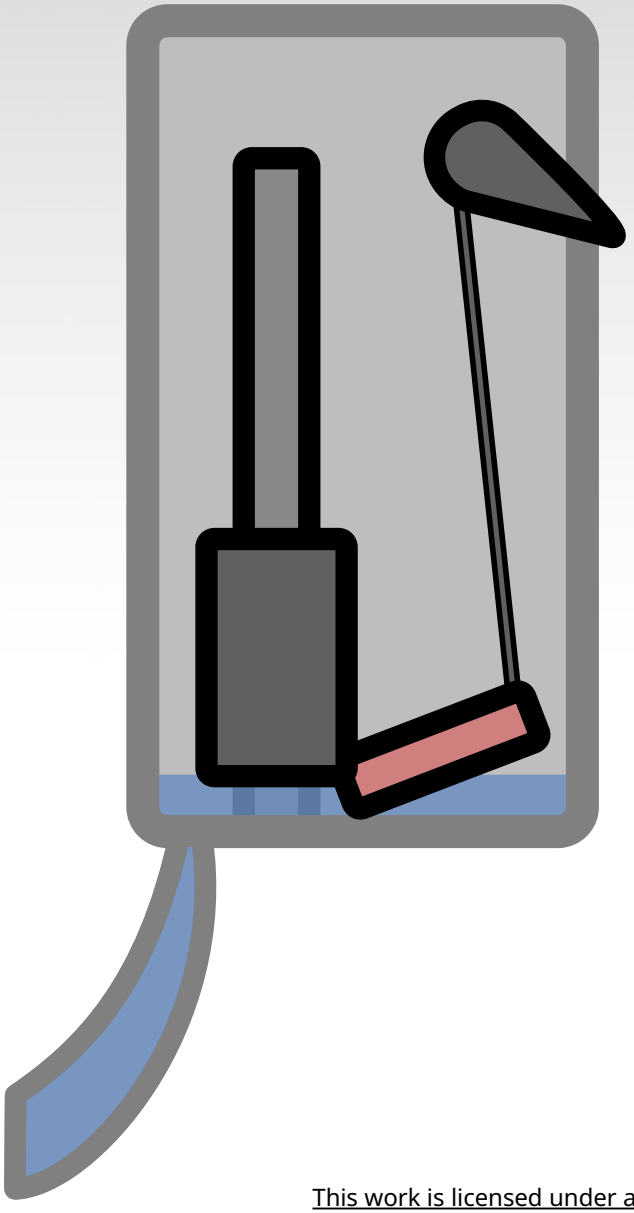
$$V = I R$$

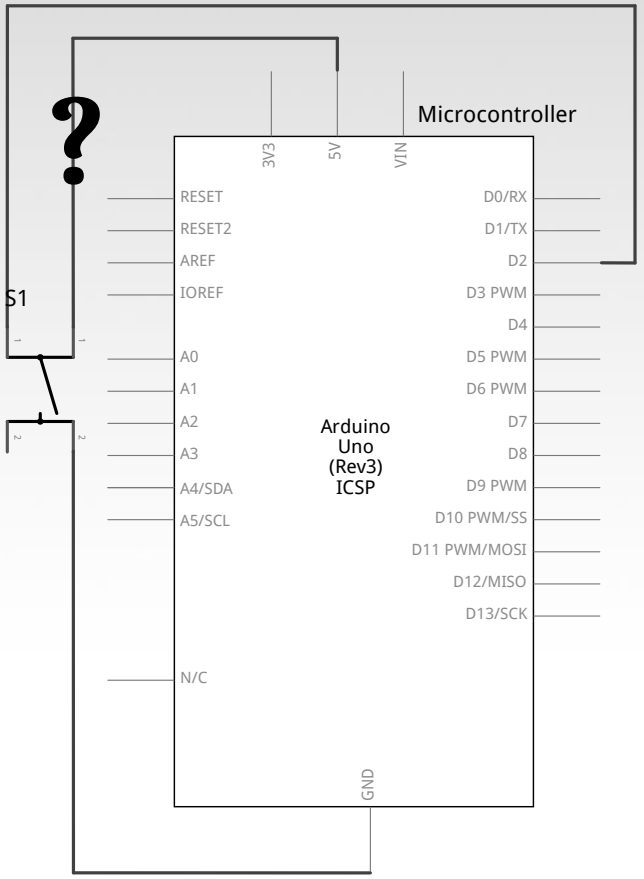


Small Pipe == Higher Resistance

$$V = I R$$

How Do We Get Here?

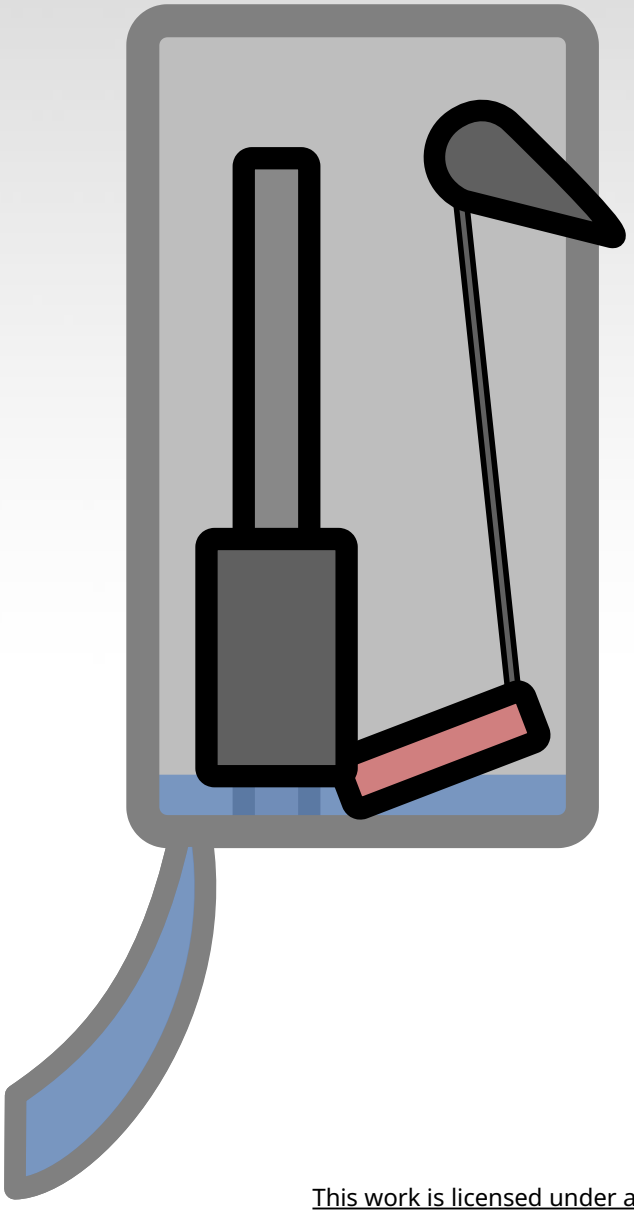


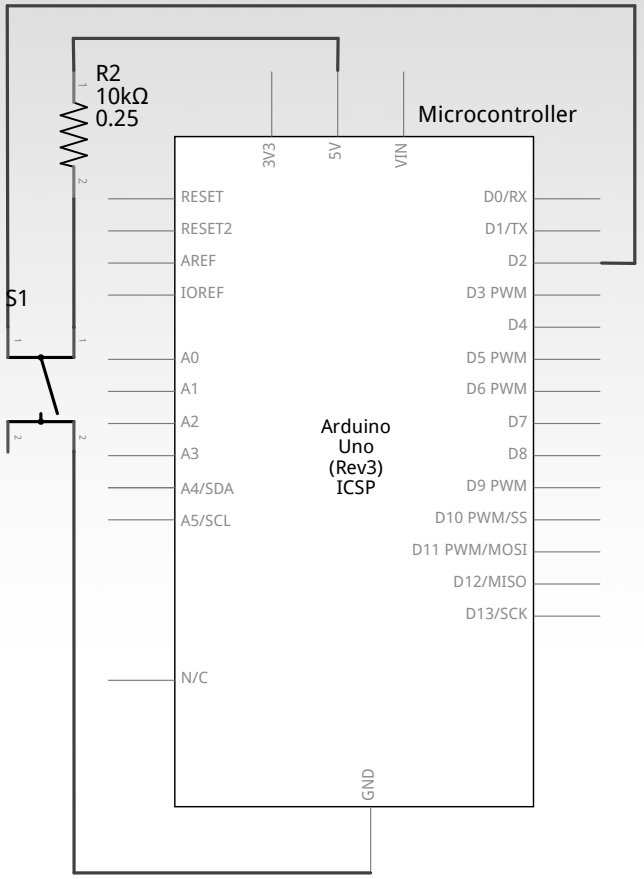


fritzing

roto

The Pull-Up Resistor!

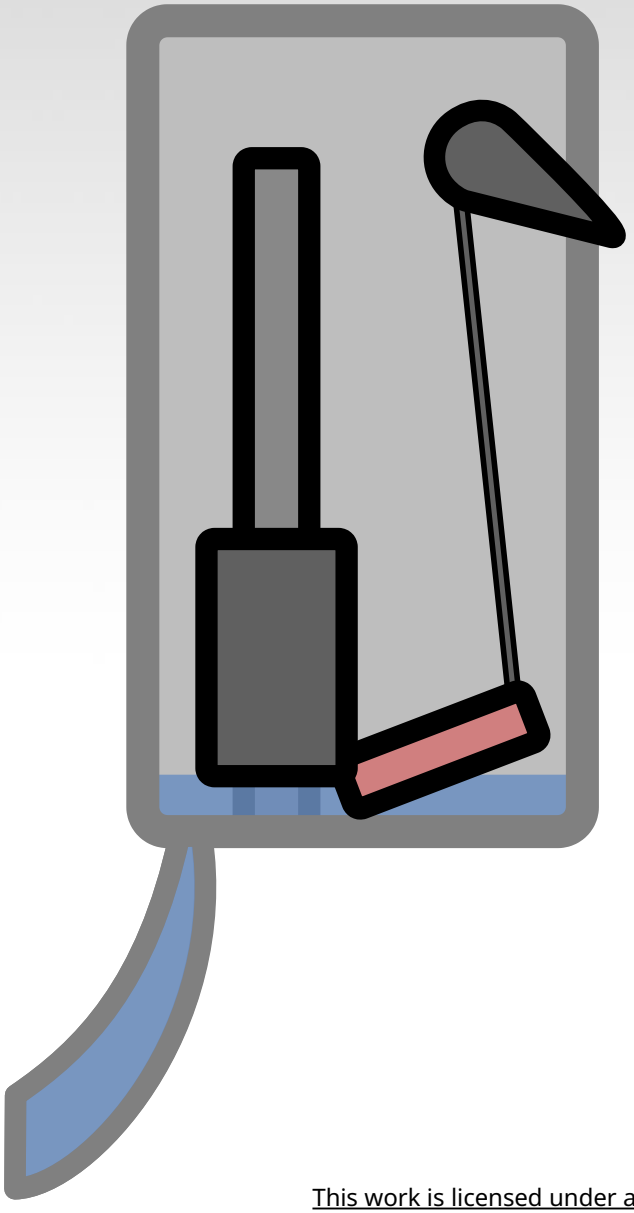


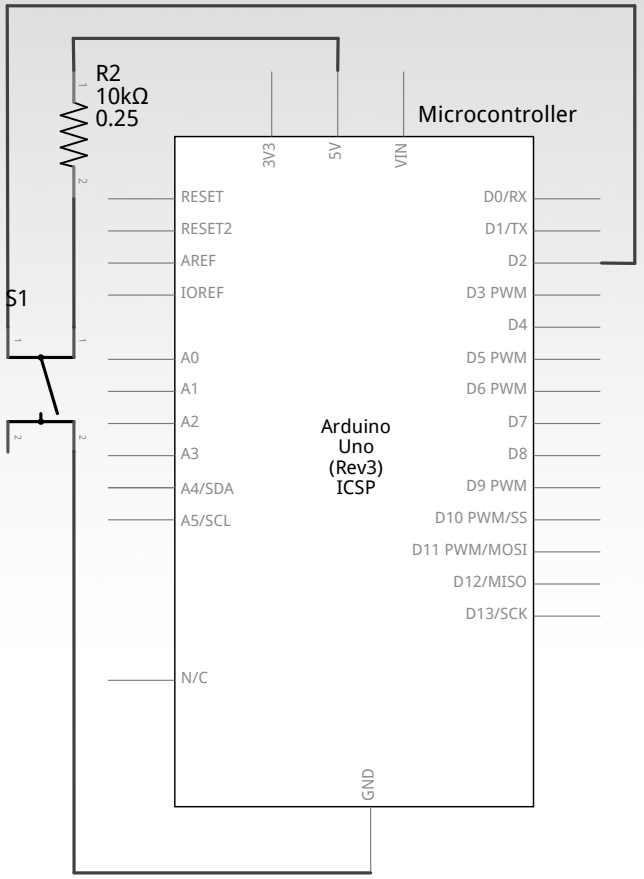


fritzing

roto

But There's A Trade-Off Between This...

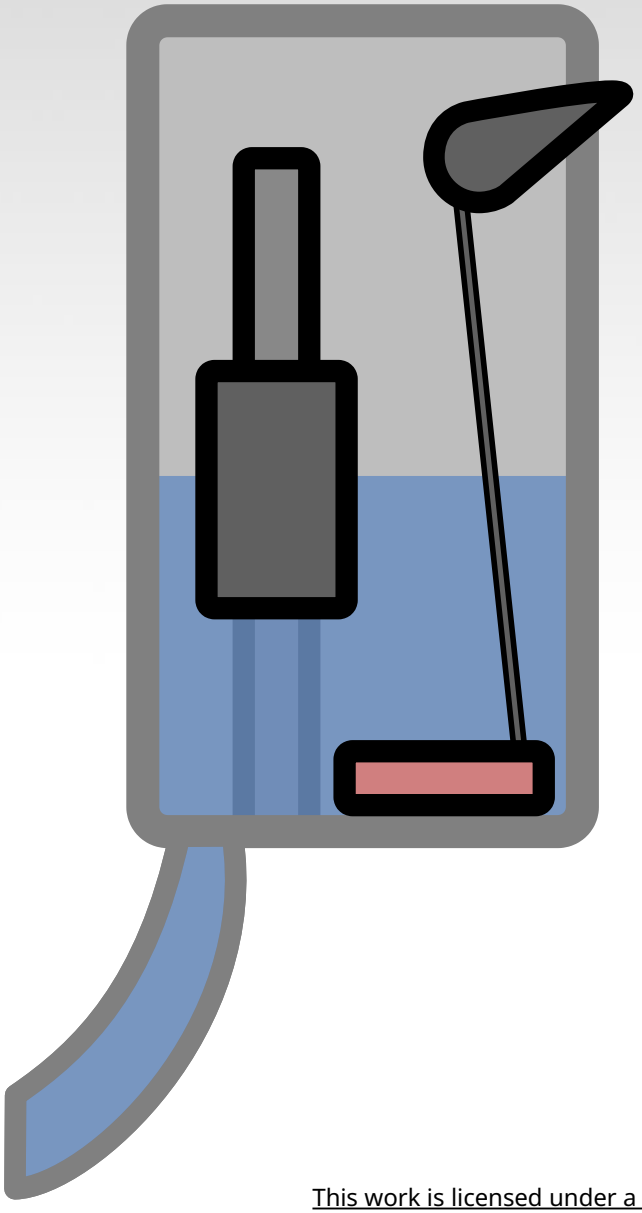


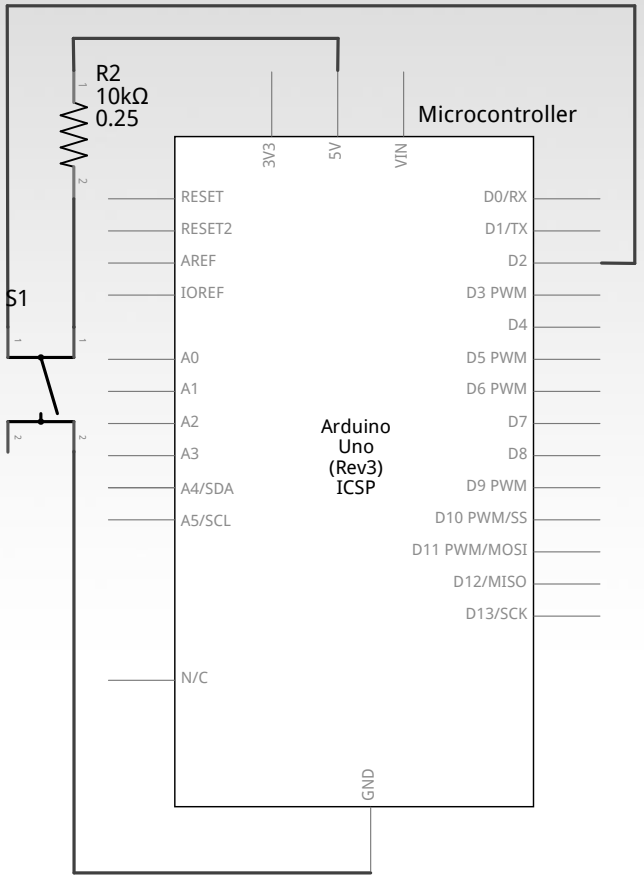


fritzing

roto

...And This





fritzing

roto

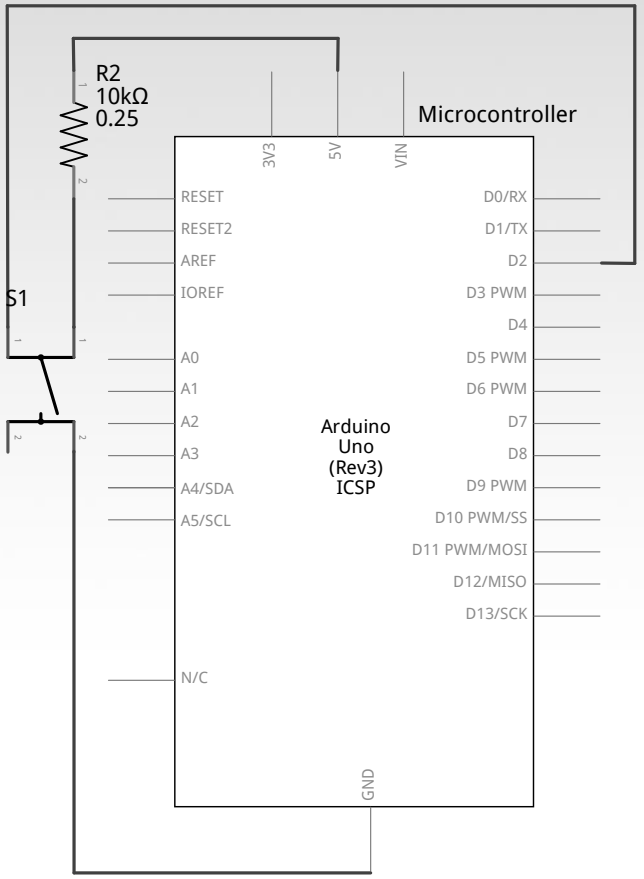
It's All About Compromise

10kΩ is a good value

(brown-black-black-red-gold)

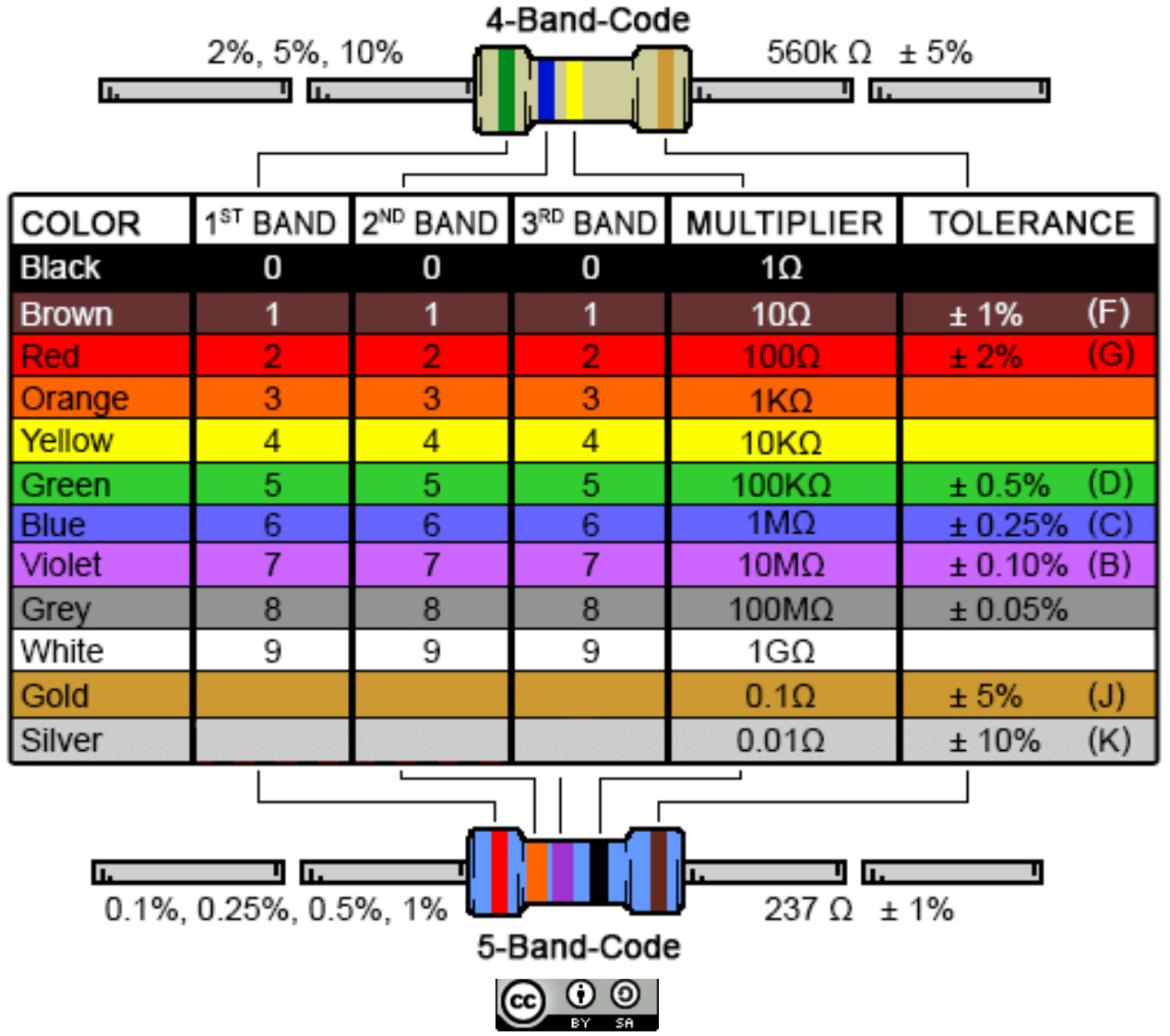
or

(brown-black-orange-gold)



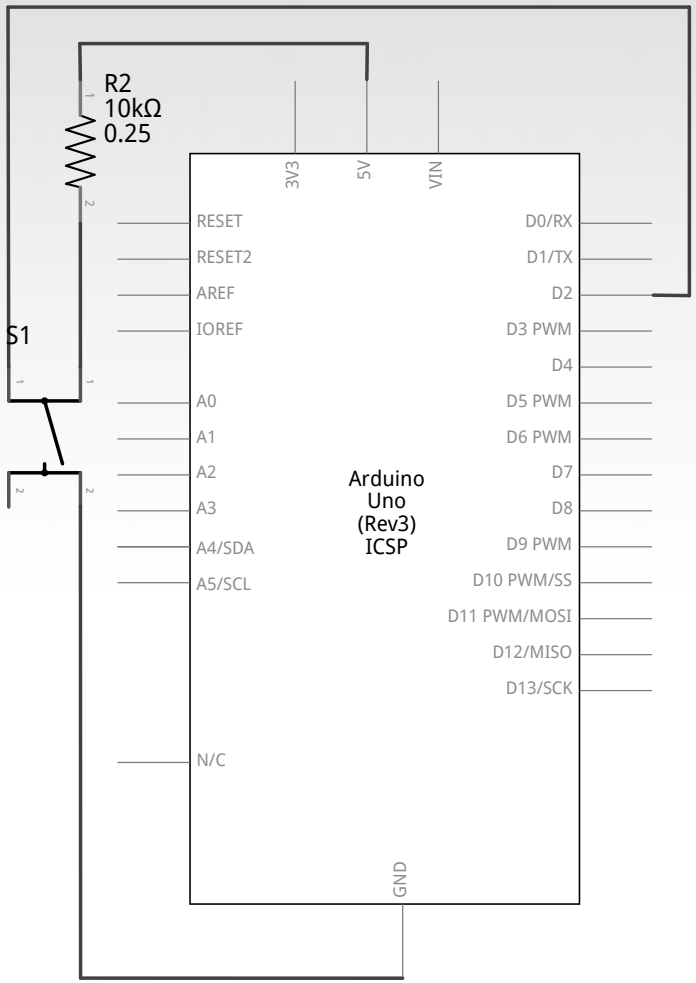
roto

Decoding Resistor Values



COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8	100MΩ	± 0.05%
White	9	9	9	1GΩ	
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)

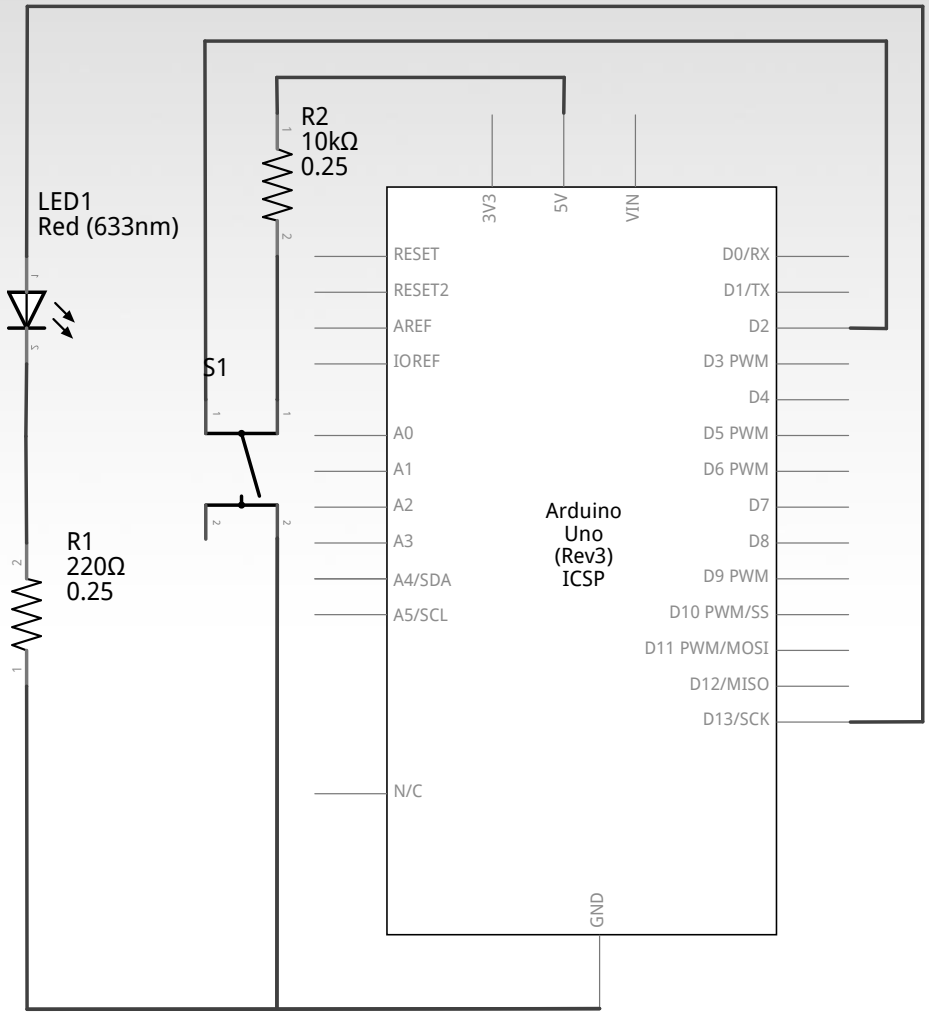
Project 4 – Button/Maintained Switch Switch Input Schematic



Schematic of Pull-Up Resistor and Switch

Project 4 – Button/Maintained Switch Switch

Full Schematic

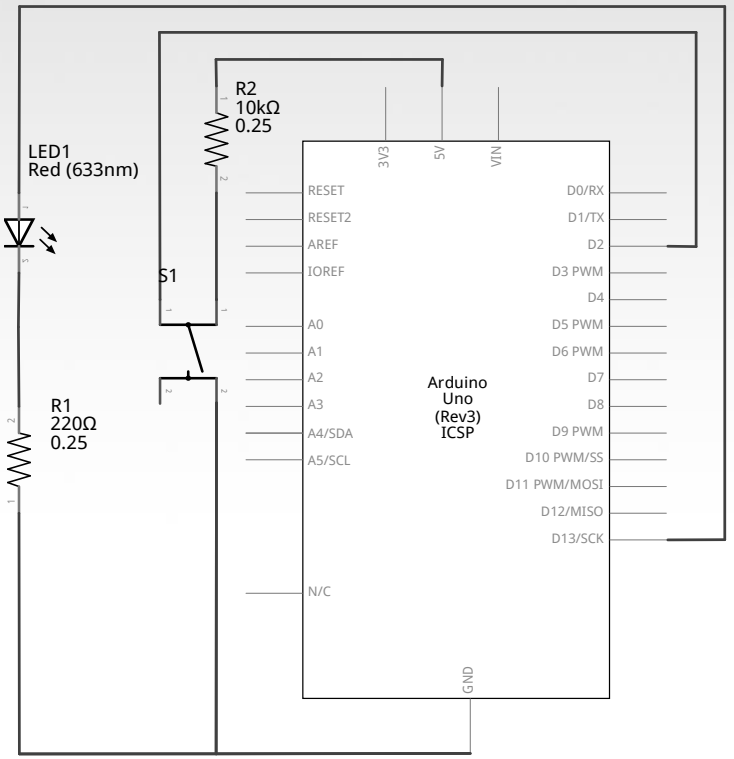


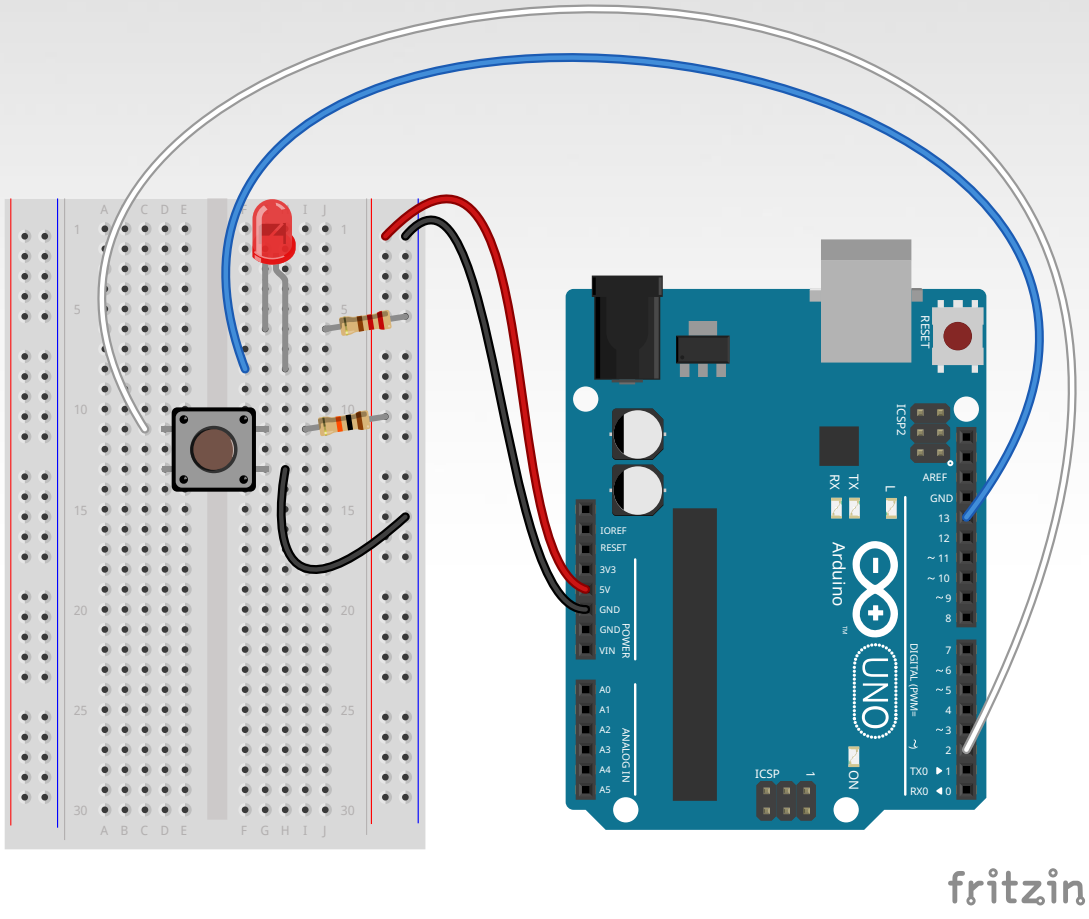
And the output is still the same as Project 1!

fritzing

roto

Project 4 – Button/Maintained Switch Switch Wiring Diagram





Project 4 – Button/Maintained Switch Switch
Our First Input Commands!

```
pinMode(pin, INPUT/OUTPUT);
```


roto

Project 4 – Button/Maintained Switch Switch
Our First Input Commands!

```
pinMode(pin, INPUT/OUTPUT);
  ex: pinMode(2, INPUT);
```


roto

Project 4 – Button/Maintained Switch Switch
Our First Input Commands!

```
pinMode(pin, INPUT/OUTPUT);
  ex: pinMode(2, INPUT);

digitalRead(pin);
```

Project 4 – Button/Maintained Switch Switch
Our First Input Commands!

```
pinMode(pin, INPUT/OUTPUT);
  ex: pinMode(2, INPUT);

digitalRead(pin);
  ex: digitalRead(2);
```

Project 4 – Button/Maintained Switch Switch
Our First Input Commands!

```
pinMode(pin, INPUT/OUTPUT);
  ex: pinMode(2, INPUT);

digitalRead(pin);
  ex: digitalRead(2);
  ex: int button_state = digitalRead(2);
```

Project 4 – Button/Maintained Switch Switch
Our First Input Commands!

```
pinMode(pin, INPUT/OUTPUT);
  ex: pinMode(2, INPUT);

digitalRead(pin);
  ex: digitalRead(2);
  ex: int button_state = digitalRead(2);
```

button_state will either be HIGH or LOW

HIGH if switch is off, LOW if switch is on

roto

Project 4 - Button/Maintained Switch Switch Code Review

button_switch.ino

```
/*
 * make a push button behave like a switch
 */

// assign pins based off circuit
int button = 2;
int led = 13;

int button_val = HIGH;         // the button starts off not pressed
int last_button_val = HIGH;    // and wasn't pressed before the sketch started
int pressed = false;           // will tell us if the button was just pressed
int led_val = LOW;             // LED starts turned off

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize pin 2 as an input and pin 13 as an output
  pinMode(button, INPUT);
  pinMode(led, OUTPUT);
}

```

Code Review Continued

button_switch.ino

```
21  // the loop function runs over and over again forever
22  void loop() {
23    button_val = digitalRead(button);                  // read the value of pin 2
24
25    if(button_val == LOW && last_button_val == HIGH) {  // if the button is pressed but
26      pressed = true;                                  // wasn't pressed last check,
27    }                                                  // set the variable 'pressed' to true
28    else {
29      pressed = false;                                 // otherwise set it to false
30    }
31
32    if(pressed) {                                      // if 'pressed' is true
33      if(led_val == LOW) {                             // and led is off,
34        led_val = HIGH;                                // turn led on
35      }                                                // if 'pressed is true
36      else {                                           // and led is on,
37        led_val = LOW;                                 // turn led off
38      }
39    }
40
41    digitalWrite(led, led_val);                        // write the LED state to the LED
42
43    if(button_val != last_button_val) {                // if the button has been pressed or
44      delay(50);                                       // released, give the circuit time
45    }                                                  // to settle
46
47    last_button_val = button_val;                      // update last_button_val for next loop
48  }
```

Code Review Continued

button_switch.ino

```
1  /*
2   * make a push button beha
3   */
4
5  // assign pins based off c
6  int button = 2;
7  int led = 13;
8
9  int button_val = HIGH;
10 int last_button_val = HIGH
11 int pressed = false;
12 int led_val = LOW;
13
14 // the setup function runs
15 void setup() {
16   // initialize pin 2 as a
17   pinMode(button, INPUT);
18   pinMode(led, OUTPUT);
19 }
20
```

button_switch.ino

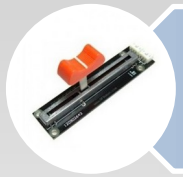
```
21 // the loop function runs over and over again forever
22 void loop() {
23   button_val = digitalRead(button);
24
25   if(button_val == LOW && last_button_val == HIGH) {
26     pressed = true;
27   }
28   else {
29     pressed = false;
30   }
31
32   if(pressed) {
33     if(led_val == LOW) {
34       led_val = HIGH;
35     }
36     else {
37       led_val = LOW;
38     }
39   }
40
41   digitalWrite(led, led_val);
42
43   if(button_val != last_button_val) {
44     delay(50);
45   }
46
47   last_button_val = button_val;
48 }
```

Project 4 – Button/Maintained Switch Switch Puzzles

Challenge 4a – Wire another LED. Make the button turn one off and the other on

Challenge 4b – Make an LED blink when you press the button (hint: look at File>Examples>Digital>Blink Without Delay)

Challenge 4c – Make an LED alternate between off, on and blink by pressing the button



analogRead()

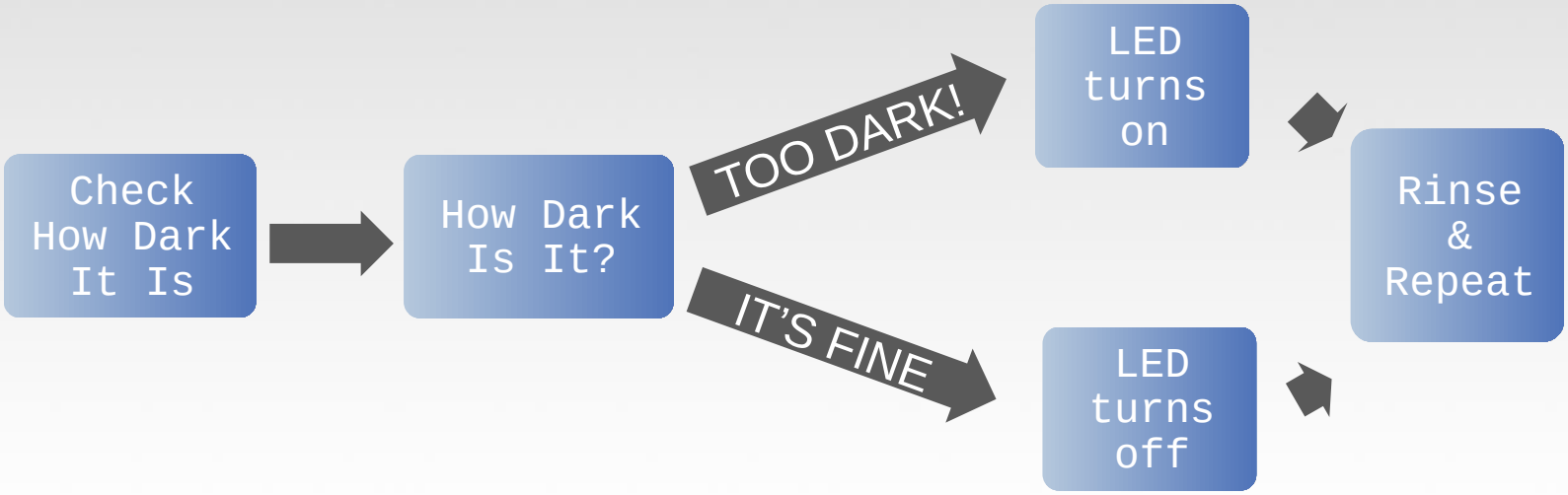
Project 4 – Night Light

How Dark Is Too Dark?

Psuedo-code – how should this work?

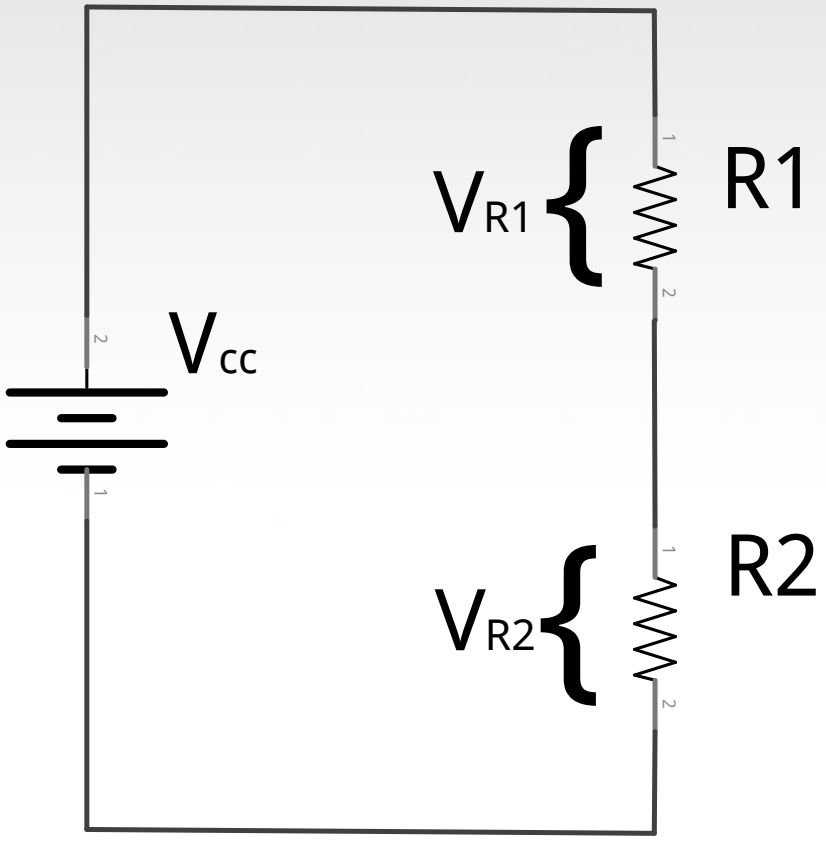
Project 4 - Night Light

Inputs and Outputs



Inputs	Outputs
Photoresistor	LED

The Voltage Divider

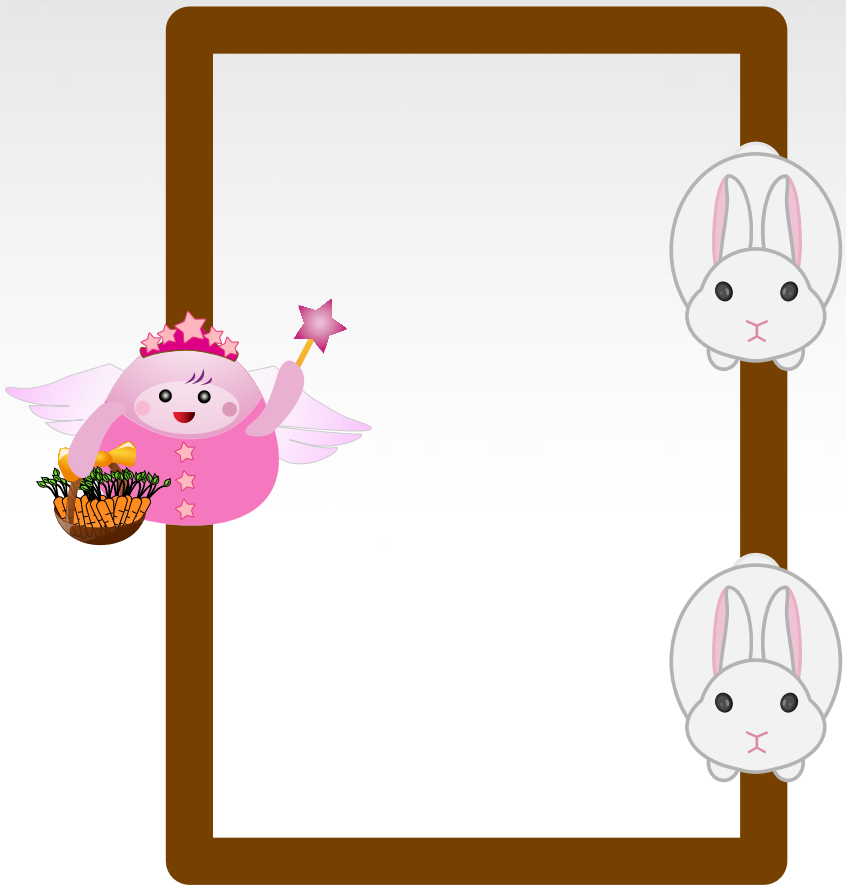


$$V_{R1} = V_{CC} \cdot \left(\frac{R_1}{R_{Total}} \right)$$

$$V_{R2} = V_{CC} \cdot \left(\frac{R_2}{R_{Total}} \right)$$

$$R_{Total} = R_1 + R_2$$

The Voltage Divider

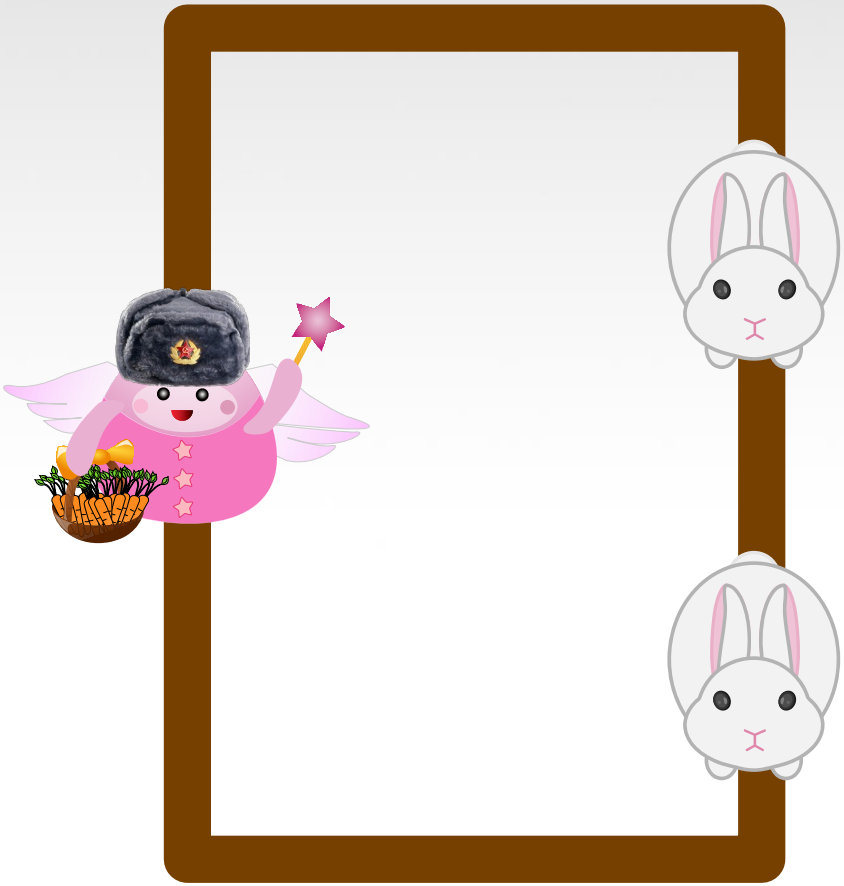


$$V_{R1} = V_{CC} \cdot \left(\frac{R_1}{R_{Total}}\right)$$

$$V_{R2} = V_{CC} \left(\frac{R_2}{R_{Total}}\right)$$

$$R_{Total} = R_1 + R_2$$

The Voltage Divider



"From each according to their ability, to each according to their need."

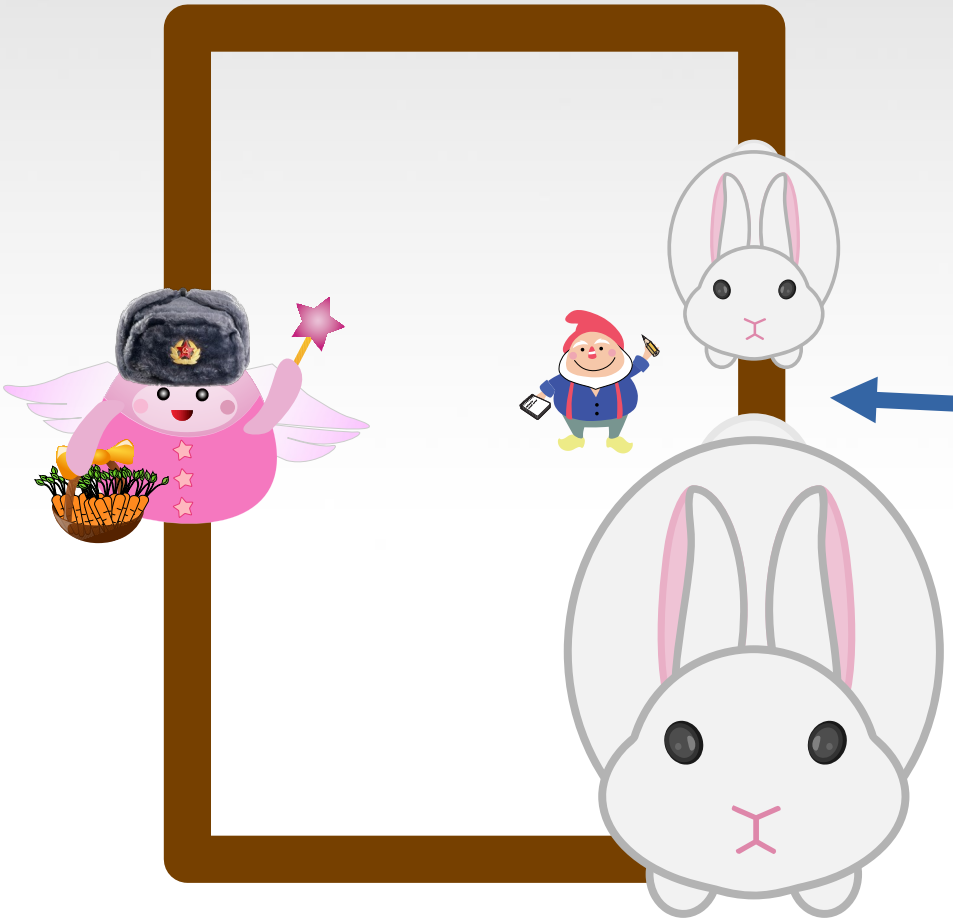
The Voltage Divider

How many carrots does the fairy have left at this point?

The Voltage Divider

How many carrots does the fairy have left at this point?

The Voltage Divider



How many carrots does the fairy have left at this point?

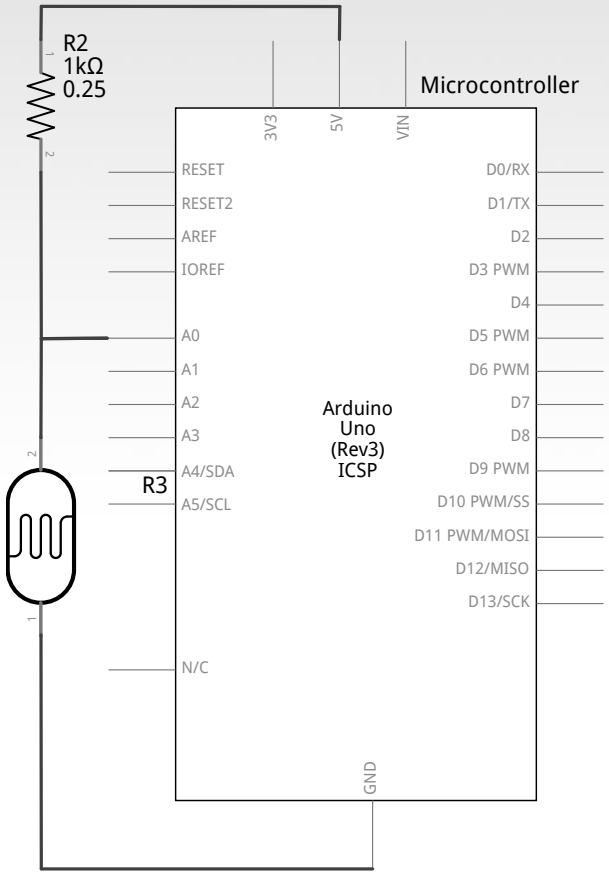
The Voltage Divider

How many carrots does the fairy have left at this point?

The Voltage Divider

How many carrots does the fairy have left at this point?

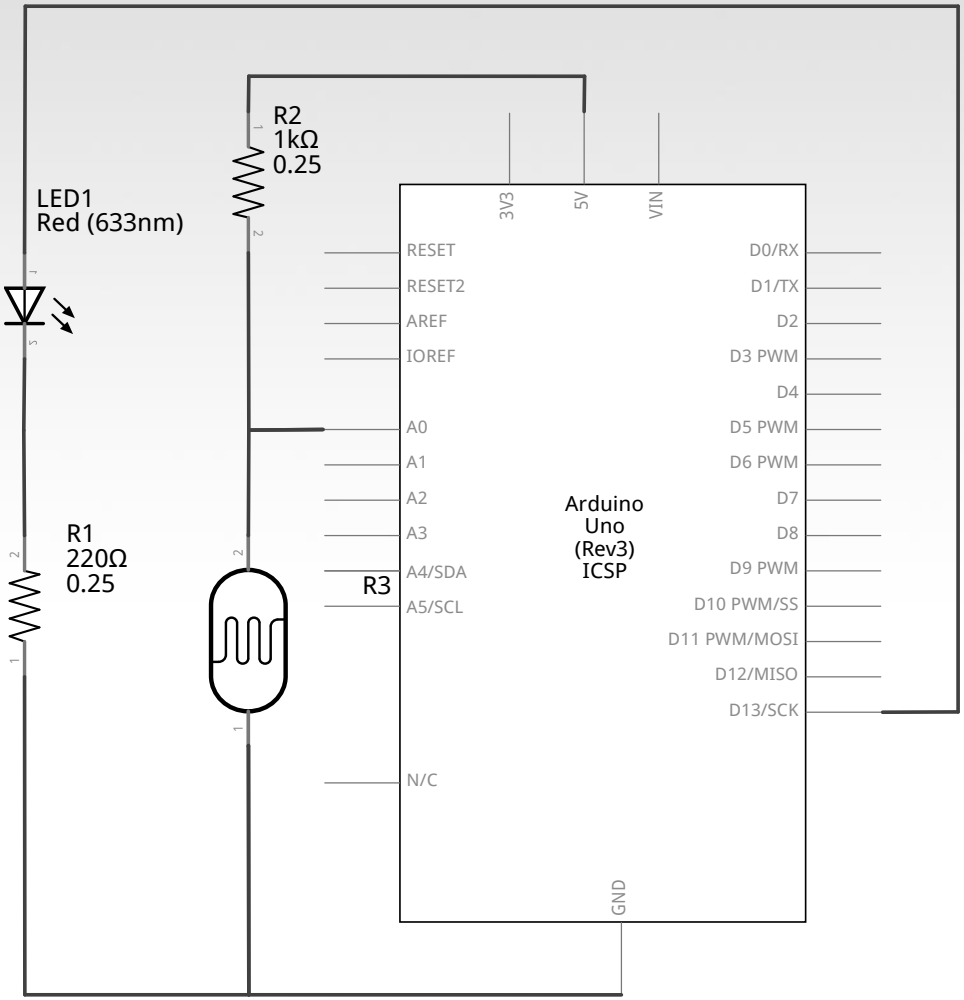
The Switch-Over



fritzing

roto

Project 4 – Night Light Schematic

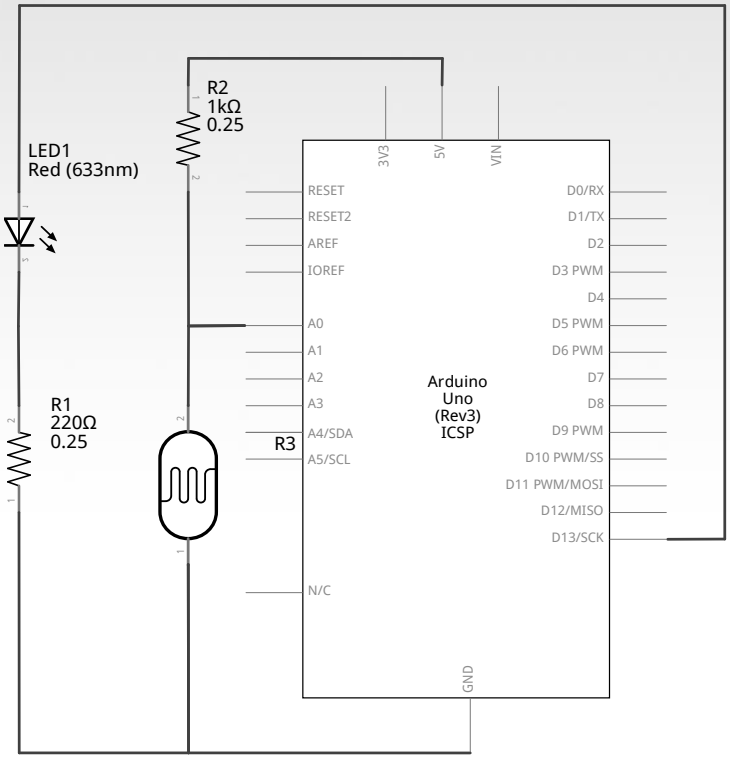


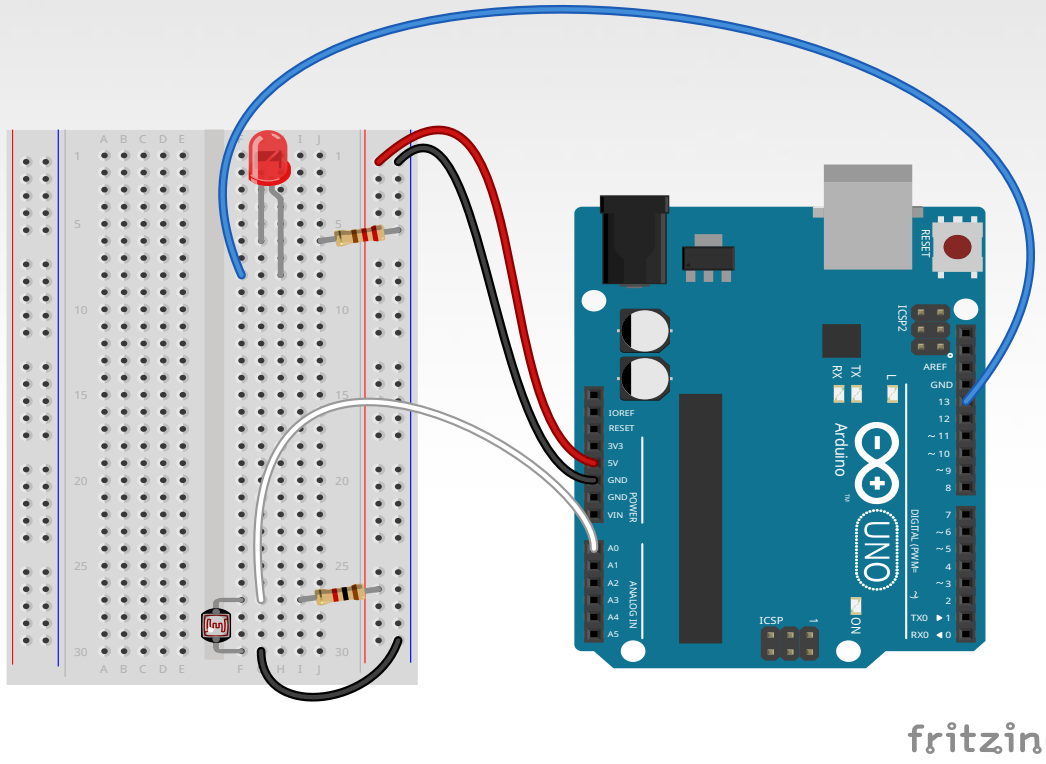
fritzing

roto

Project 4 – Night Light

Wiring Diagram





Project 4 – Night Light

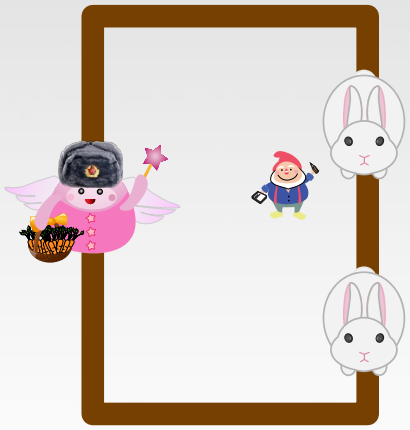
```
int sensor_val = analogRead(pin);
```

pin – refers to the analog input pin (limited to pins A0, A1, A2, A3, A4 and A5)

Arduino uses a 10-bit analog-to-digital converter (ADC):

- This means that `sensor_val` is always a number between 0 and 1023
  - Minimum Carrots → 0
  - Maximum Carrots → 1023

Determining Analog Values



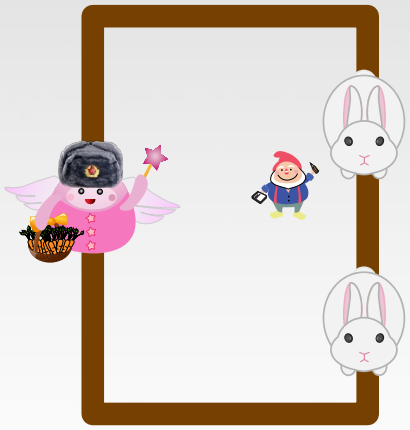
But the top bunny is always going to get some carrots, so we can't really get to 1023

And we can't really make our bottom bunny so small that it gets 0 carrots

So how do we determine the minimum and maximum numbers for our circuit?

- Do math

Determining Analog Values



But the top bunny is always going to get some carrots, so we can't really get to 1023

And we can't really make our bottom bunny so small that it gets 0 carrots

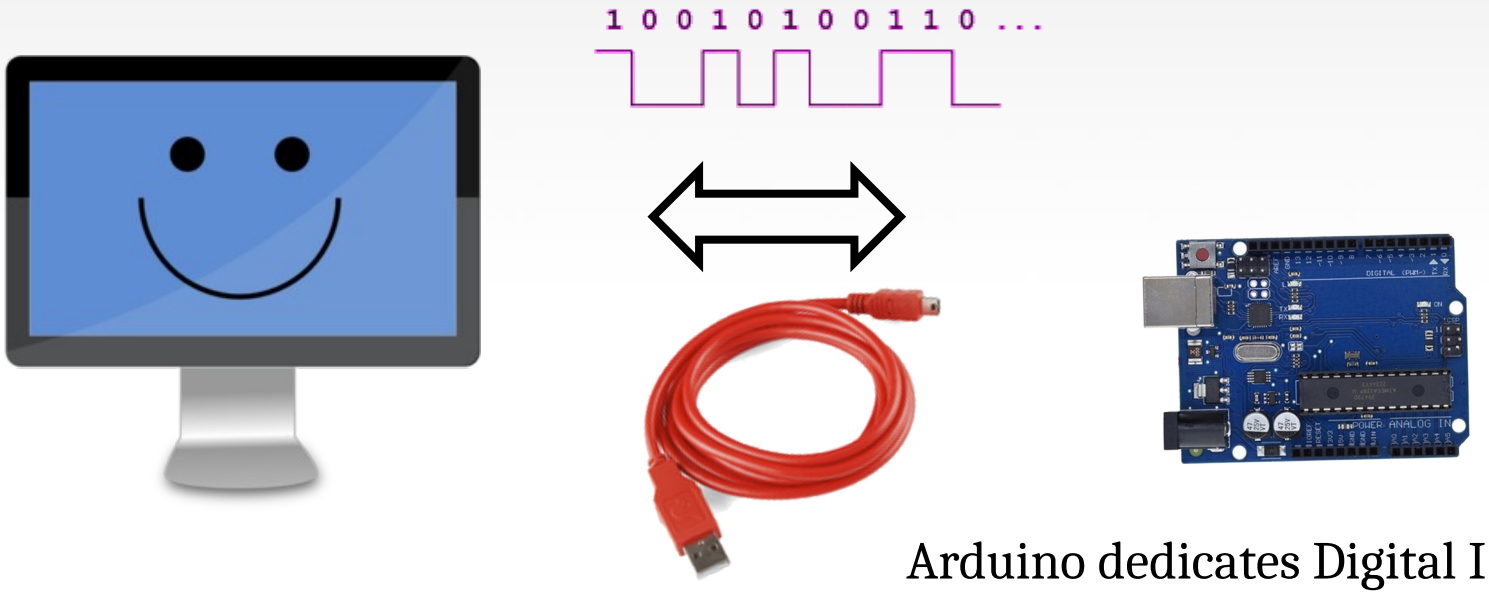
So how do we determine the minimum and maximum numbers for our circuit?

- ~~Do math~~
- Ask the microcontroller

Using Serial Communication

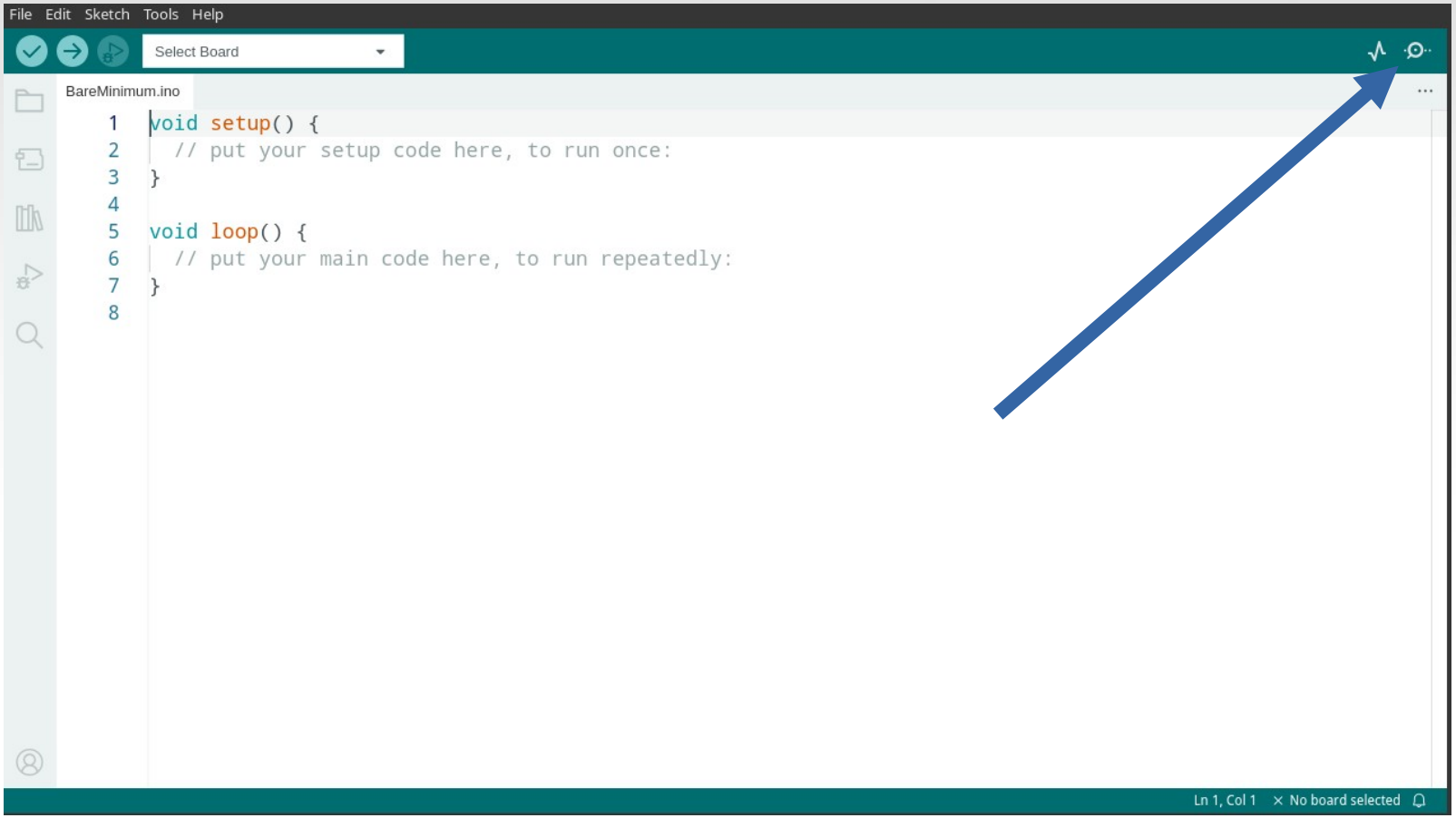
Method used to transfer data between two devices.

Data passes between the computer and Arduino through the Universal **Serial** Bus (USB!) cable. Data is transmitted as zeros (‘0’) and ones (‘1’) sequentially.

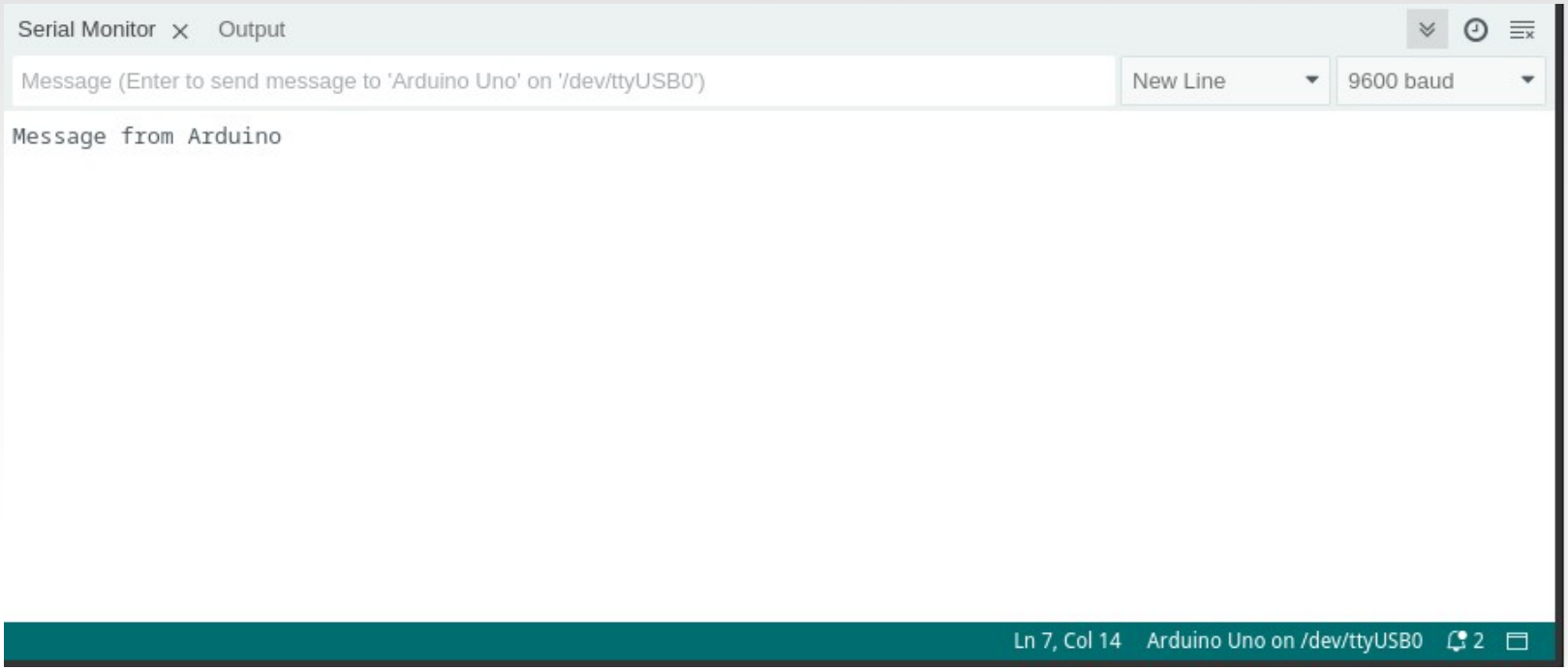


Arduino dedicates Digital I/O pin # 0 to receiving and Digital I/O pin #1 to transmit.

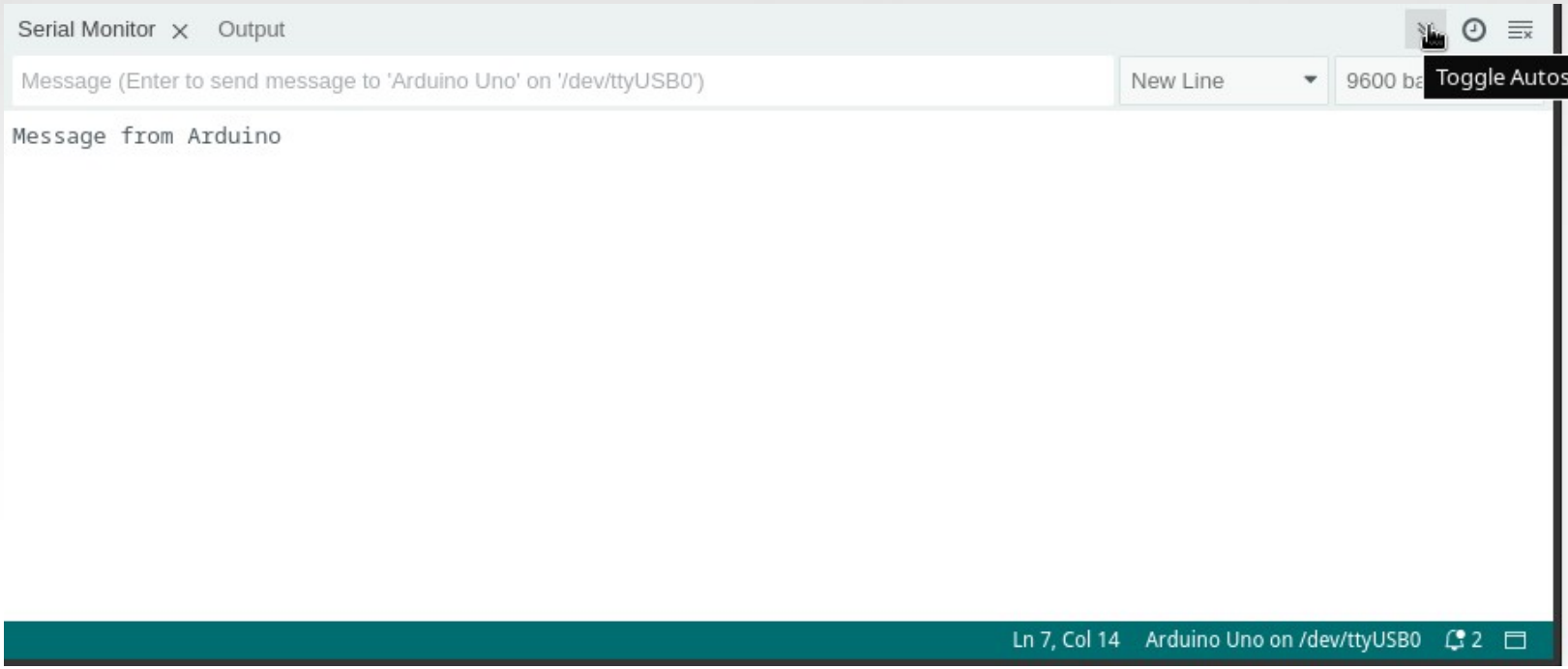
Displaying the Serial Monitor



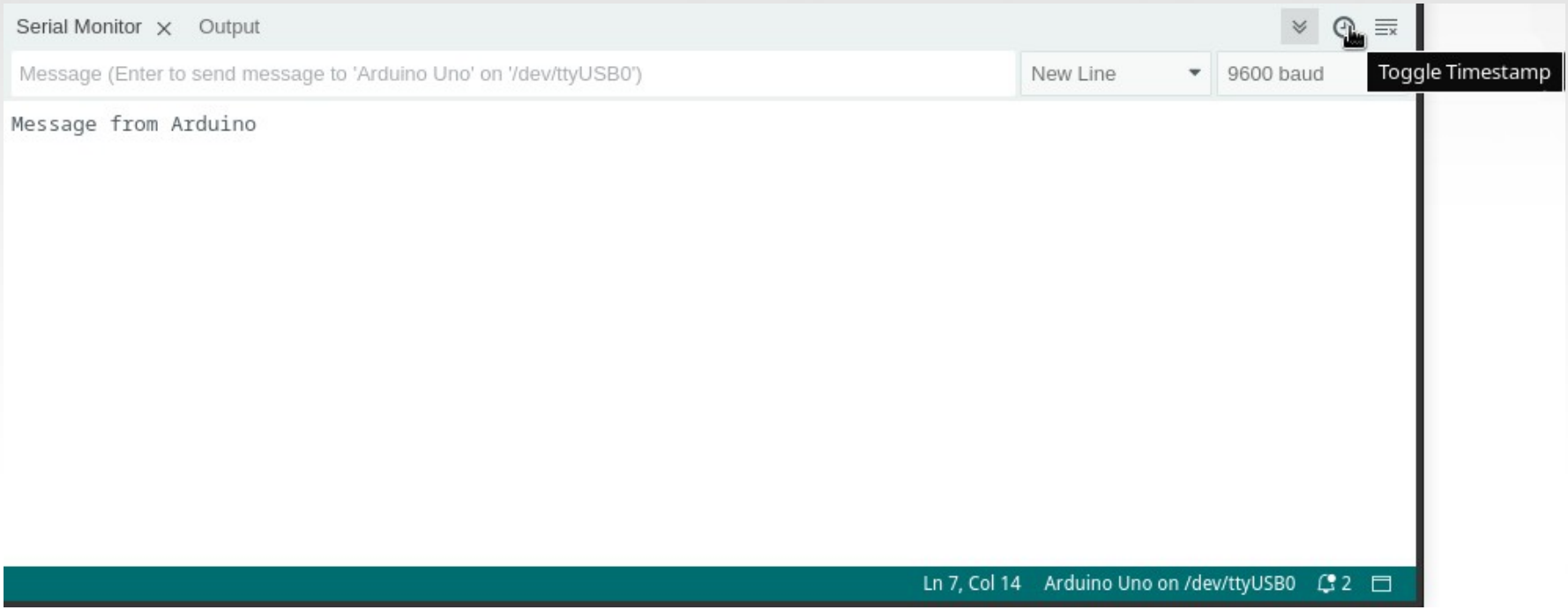
A Tour of the Serial Monitor



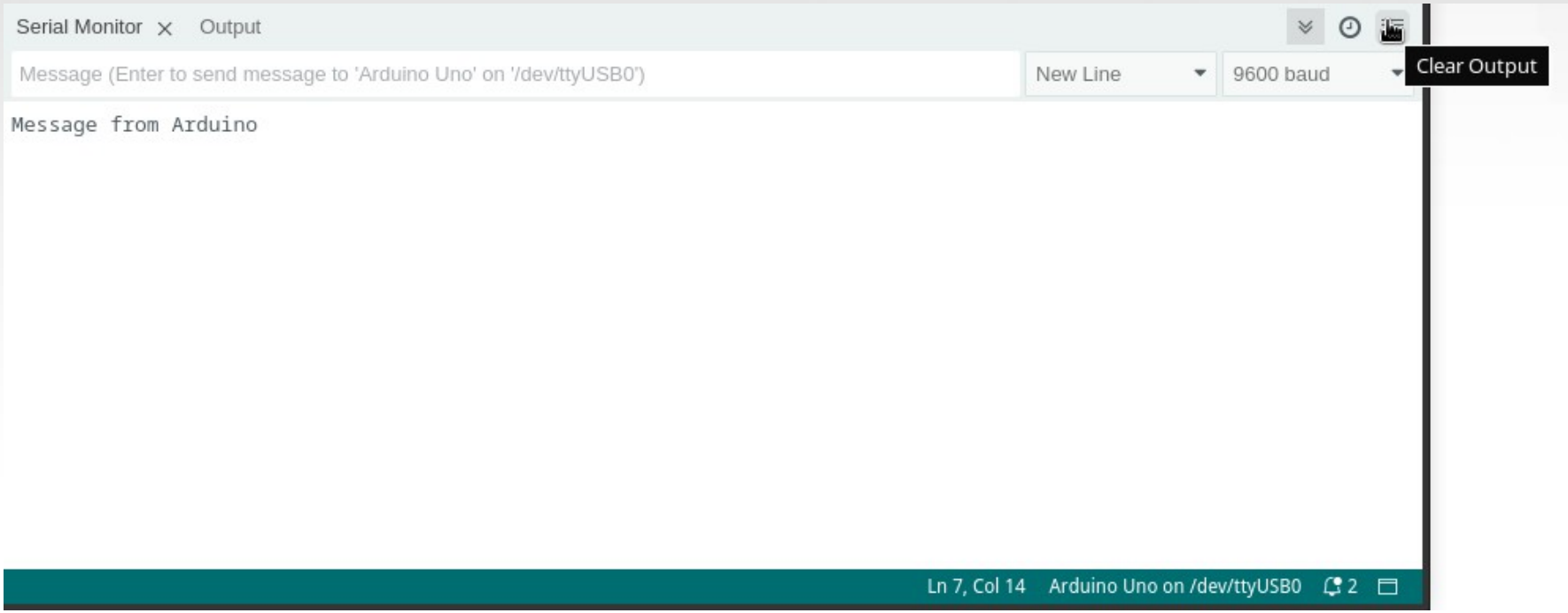
A Tour of the Serial Monitor



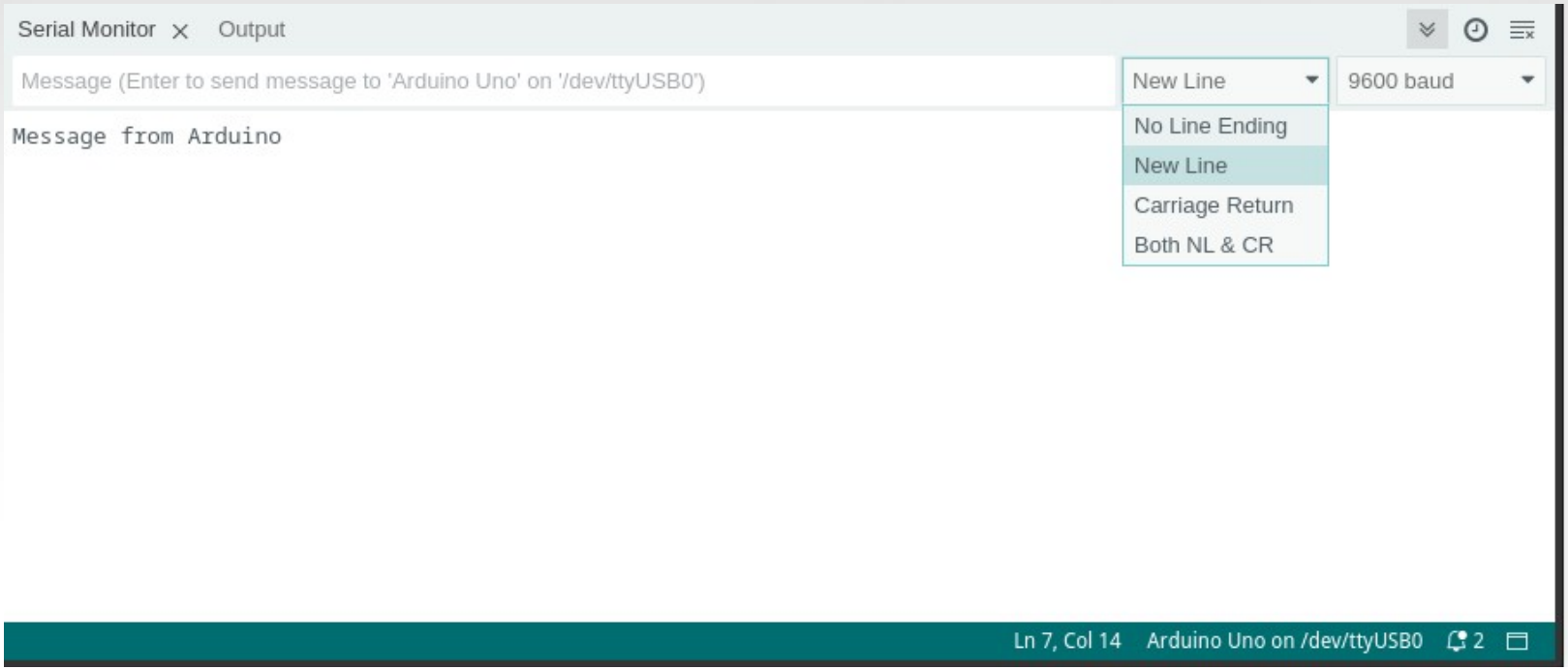
A Tour of the Serial Monitor



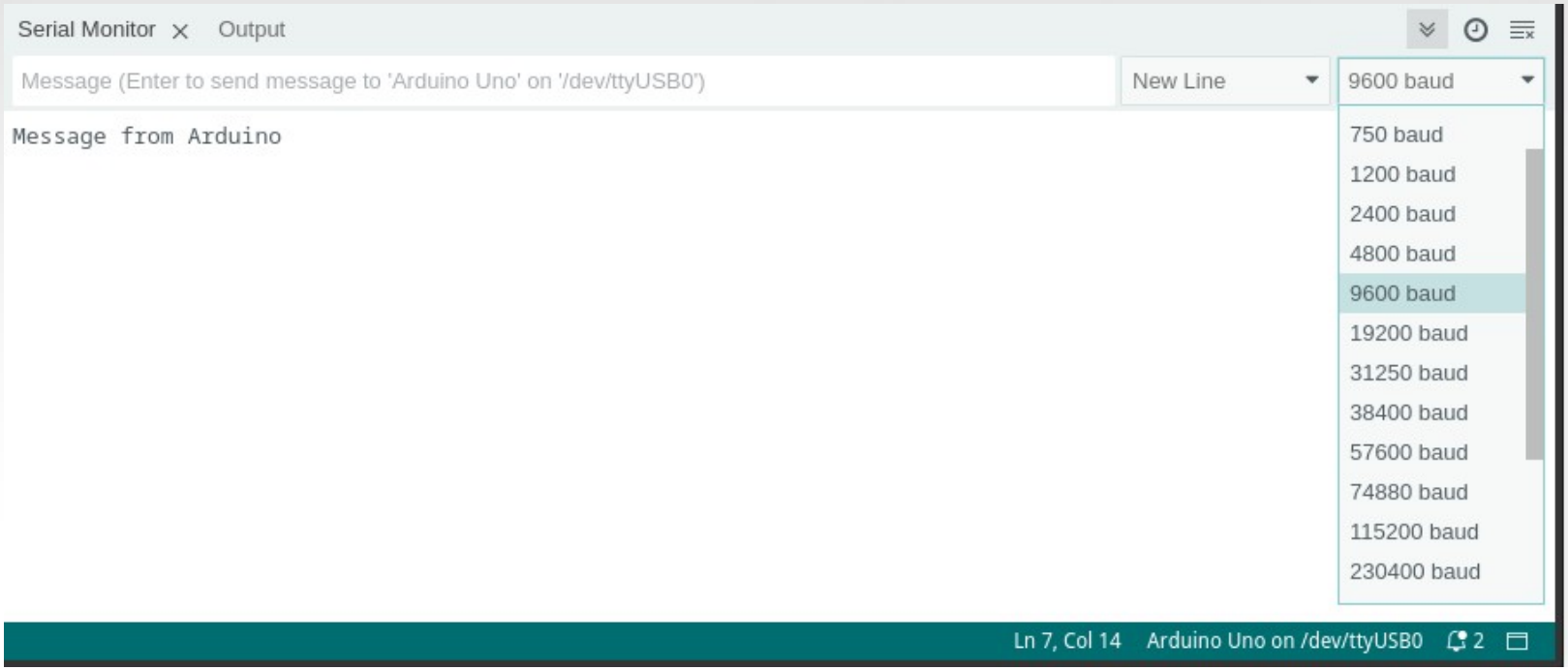
A Tour of the Serial Monitor



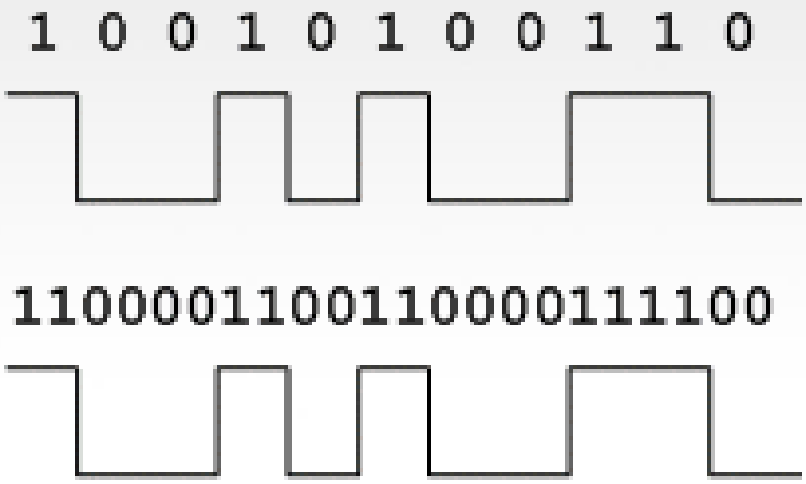
A Tour of the Serial Monitor



A Tour of the Serial Monitor



Baud Rate



How do we know how many 0's and 1's there are?

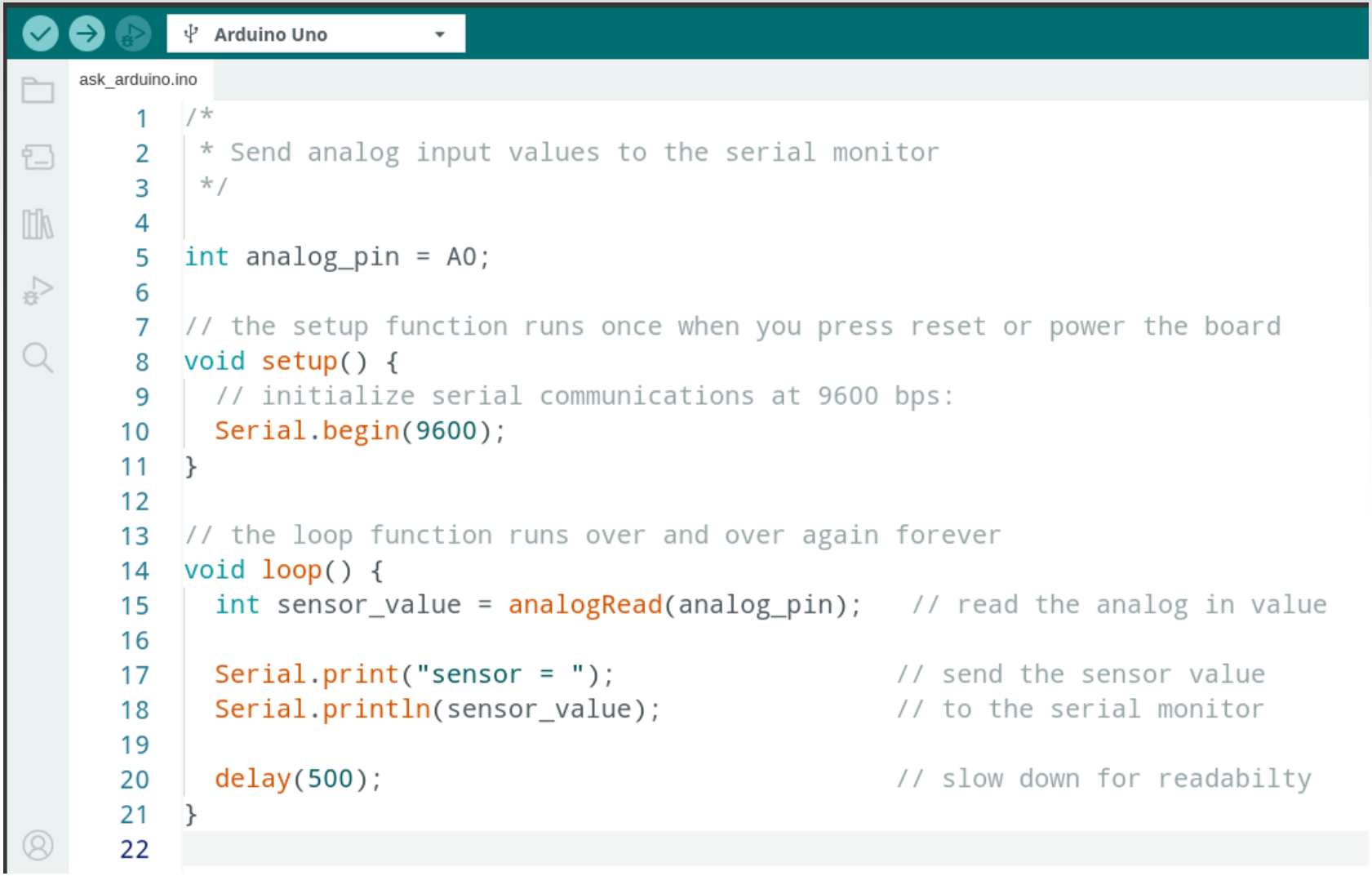
We tell how the computer and microcontroller how long a signal has to be high or low to count as a 1 or a 0 by setting the **baud rate**

For our purposes it doesn't really matter *what* the baud rate is, it's just important that the microcontroller and computer agree!

Serial Commands

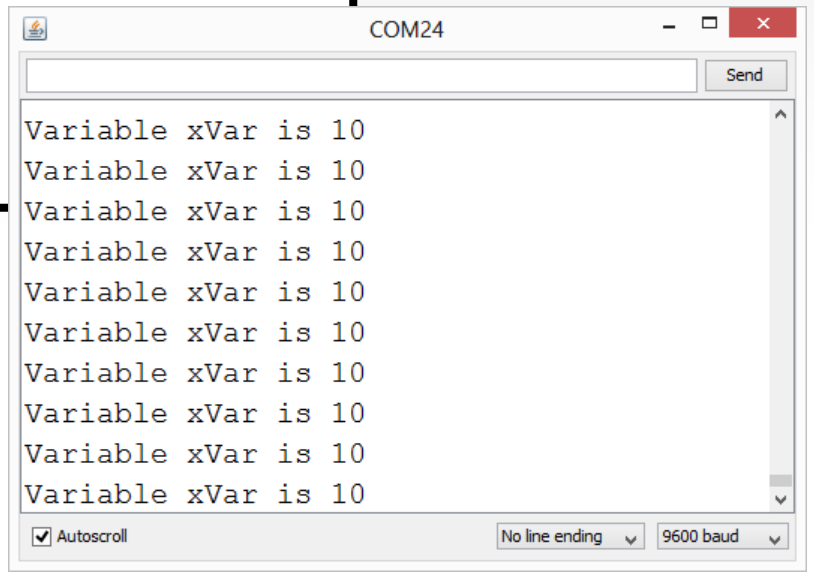
```
Serial.begin(baud_rate);
  ex: Serial.begin(9600);

Serial.print("Text");
Serial.print(variable);

Serial.println("Text");
Serial.println(variable);
```

Ask Arduino - Code Review



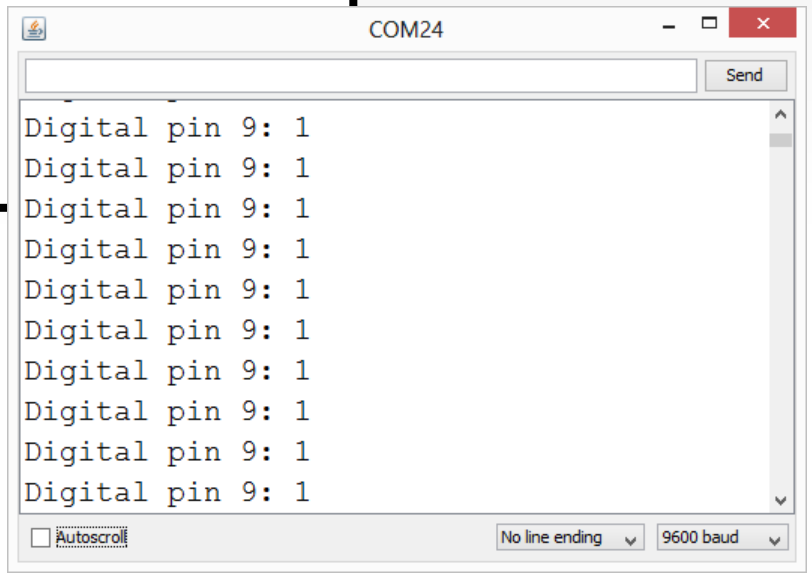
Serial Communication: Serial Debugging

```
void loop()
{
   int xVar = 10;
   Serial.print ( “Variable xVar is “ ) ;
   Serial.println ( xVar ) ;
}
```



Serial Communication: Serial Troubleshooting

```
void loop ( )
{
Serial.print (“Digital pin 9: “);
Serial.println (digitalRead(9));
}
```



Night Light - Code Review

night_light.ino

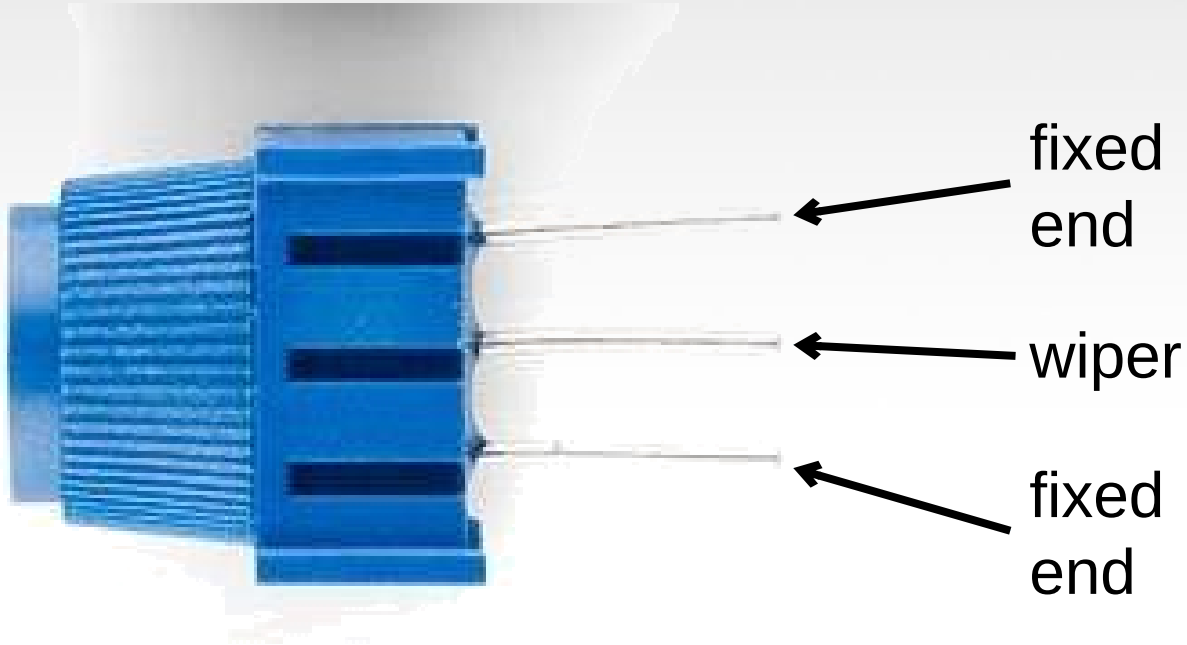
```
/*
 * Turns on a light if a photoresistor says it's too dark
 */


// assign pins according to the circuit
int photoresistor = A0;
int led = 13;

int threshold = 300;  // We find this by running the 'Ask Arduino' sketch

// the setup function runs once when you press reset or power the board
void setup() {
  // note that we don't have to set up the analog pin A0 as an input
  pinMode(led, OUTPUT);   // set led pin to output
}

// the loop function runs over and over again forever
void loop() {

  int light_level = analogRead(photoresistor); // read the value from the photoresistor

  if(light_level > threshold) {               // if sensor value is greater
    digitalWrite(led, HIGH);                  // than our threshold, turn
  }                                           // the light on
  else {
    digitalWrite(led, LOW);                   // otherwise, turn the light
  }                                           // off

  delay(5);                                   // pause to let ADC settle
}

```

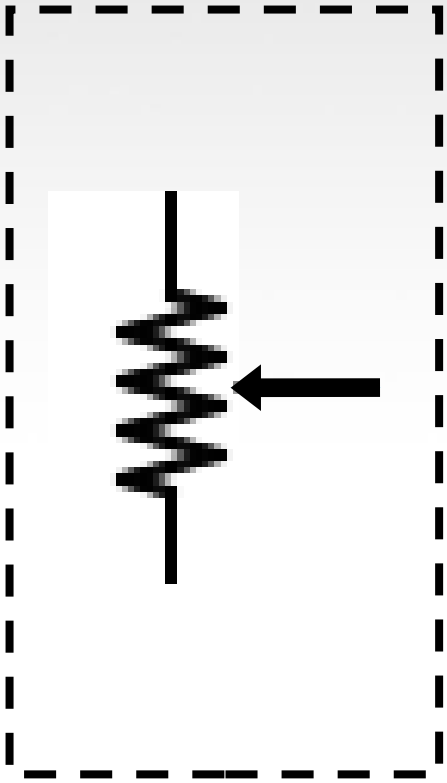
Analog Sensors

Examples:

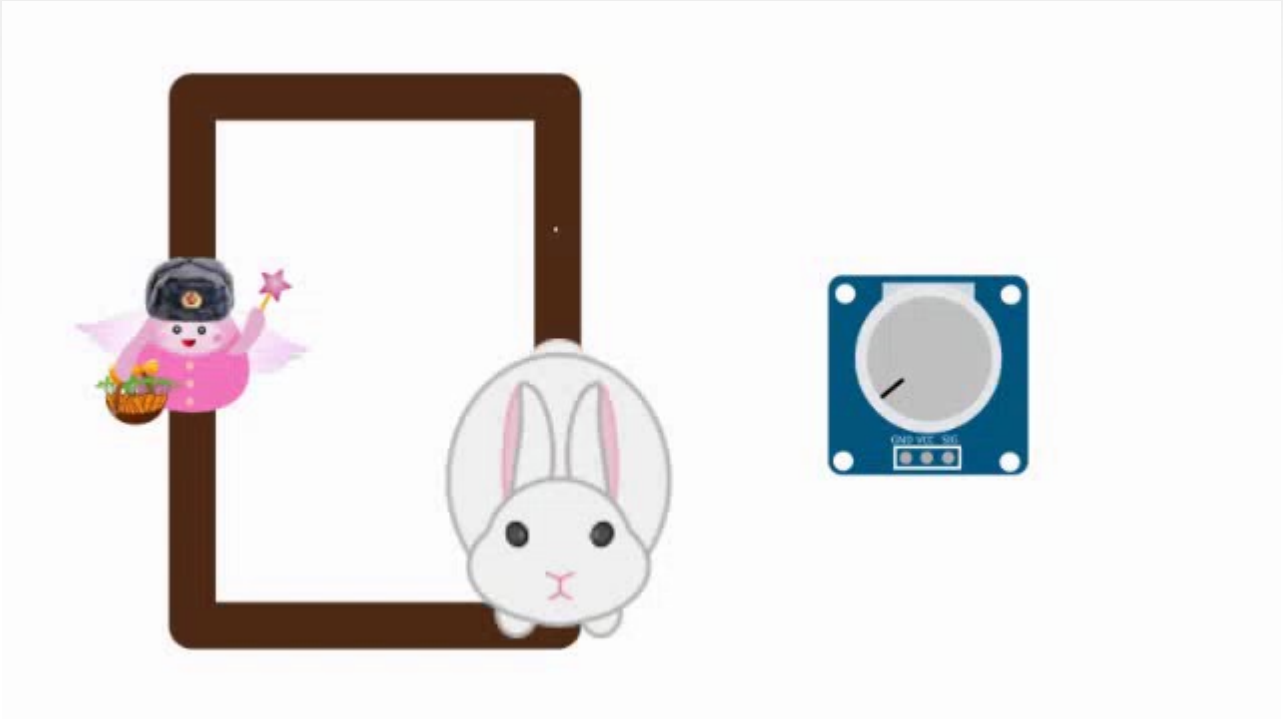
Sensors	Variables
Photoresistor	light_level
Microphone	sound_volume
Temp Sensor	temp
Flex Sensor	bend
Accelerometer	tilt, acceleration

Trimpot (Potentiometer)
Variable Resistor





Trimpot (Potentiometer)
Variable Resistor



Trimpot (Potentiometer)
What does it mean?

Since we're controlling the size of *both* bunnies:

- We can make either bunny so small it gets zero carrots
- We get the full range of 0-1023!
  - This makes potentiometers excellent controllers for physical computing "settings"
  - Can we think of any examples?
  - But what if the setting we want to control has less (or more) than 1023 values? Or if we want it not to start at 0?
    - Do math

Trimpot (Potentiometer)
What does it mean?

Since we're controlling the size of *both* bunnies:

- We can make either bunny so small it gets zero carrots
- We get the full range of 0-1023!
  - This makes potentiometers excellent controllers for physical computing "settings"
  - Can we think of any examples?
  - But what if the setting we want to control has less (or more) than 1023 values? Or if we want it not to start at 0?
    - ~~Do math~~
    - Get the microcontroller to do it for us

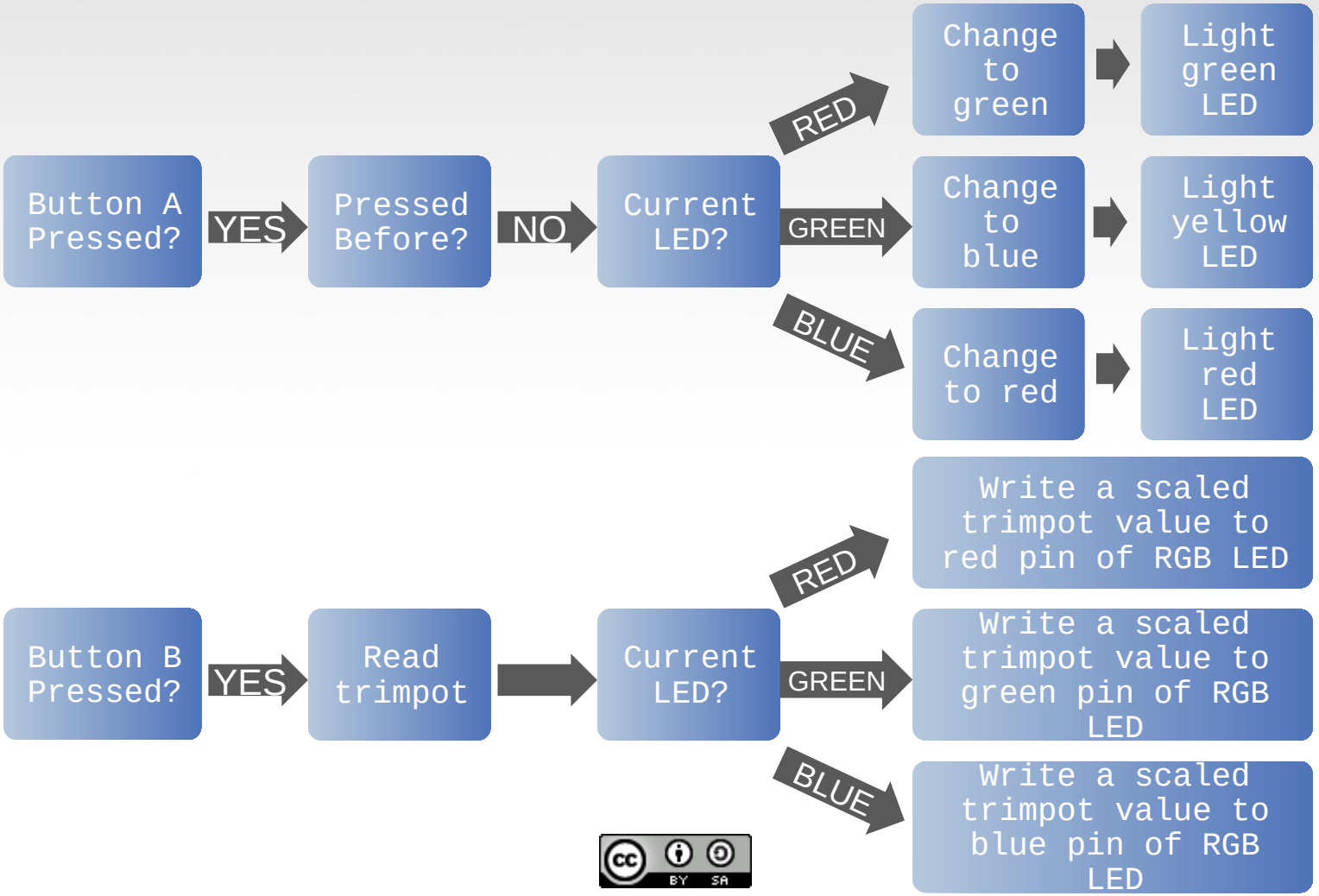
Map Command

```
int new_var = map(old_var, old_low, old_high, new_low, new_high);

    ex: offset_val = map(val, 0, 100, -50, 50);

    ex: reverse_val = map(val, 1, 100, 100, 1);

    ex: write_val = map(read_val, 0, 1023, 0, 255);
```

Project 5 – Real Time Color Mixer

Tying it all together...

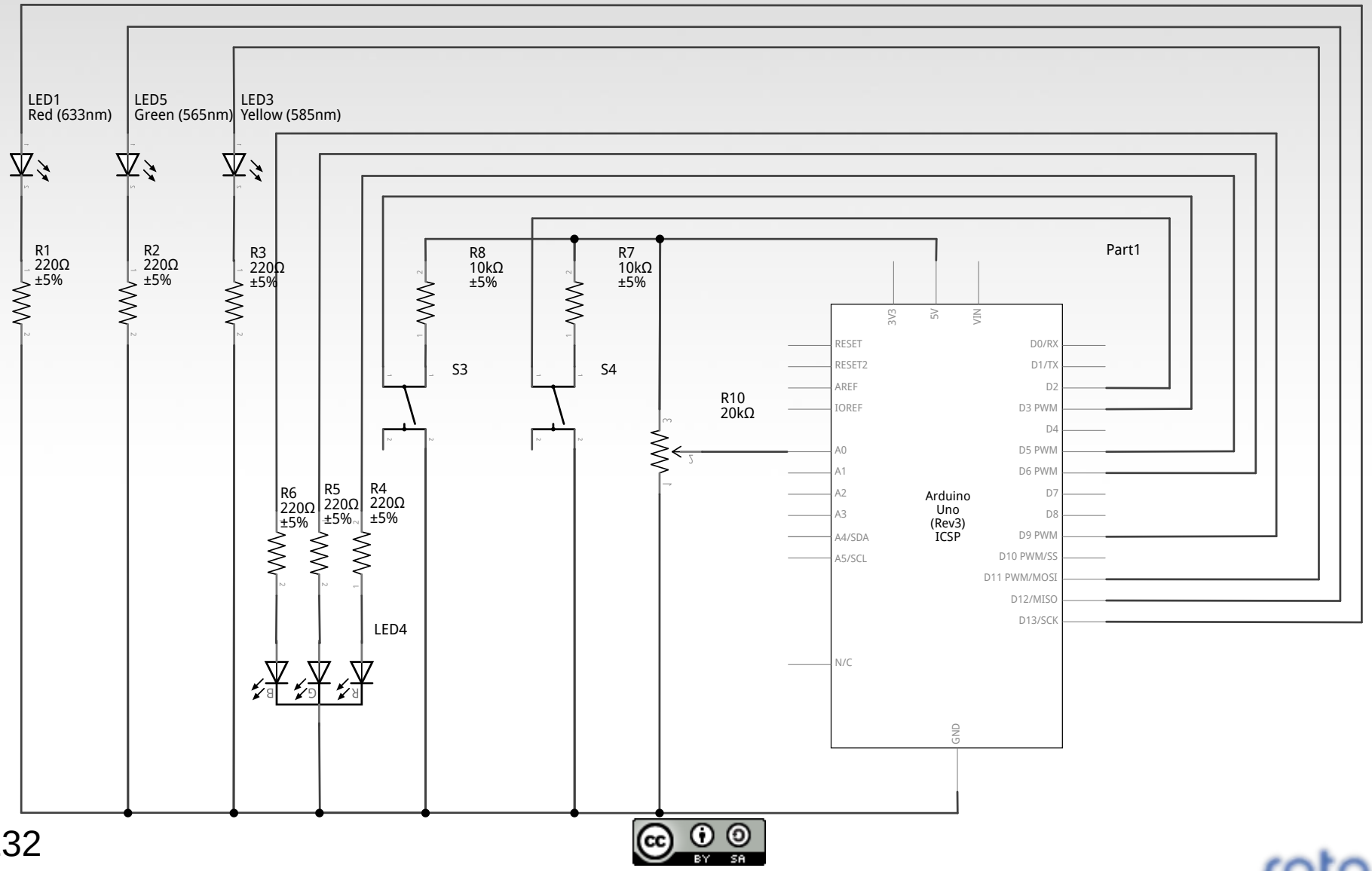
Psuedo-code – how should this work?



Project 5 – Inputs and Outputs

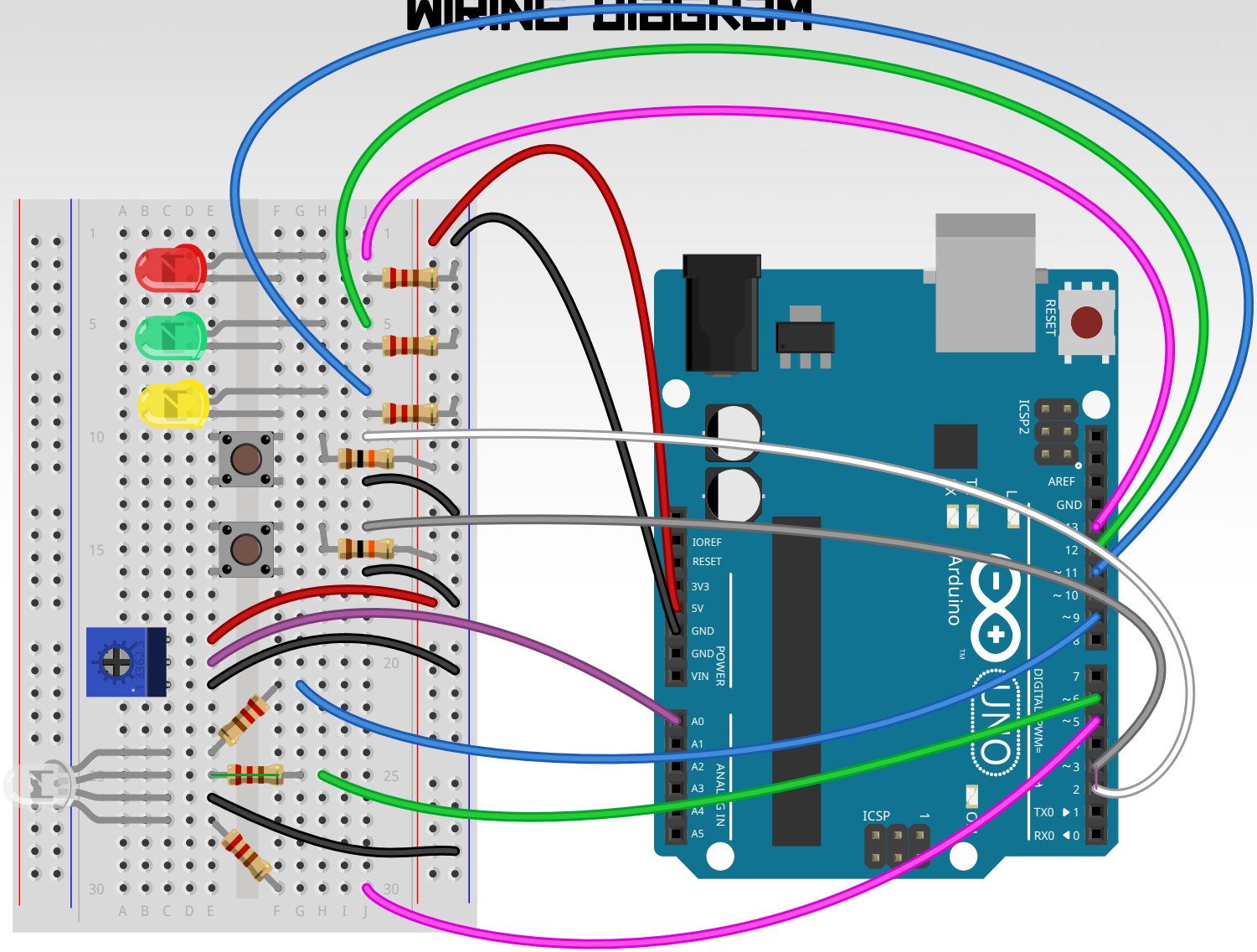
Inputs	Outputs
a “Select” Button	Red Indicator
an “Adjust” Button	Green Indicator
Trimpot	Blue Indicator
	RGB LED

Project 5 – Real Time Color Mixer Schematic



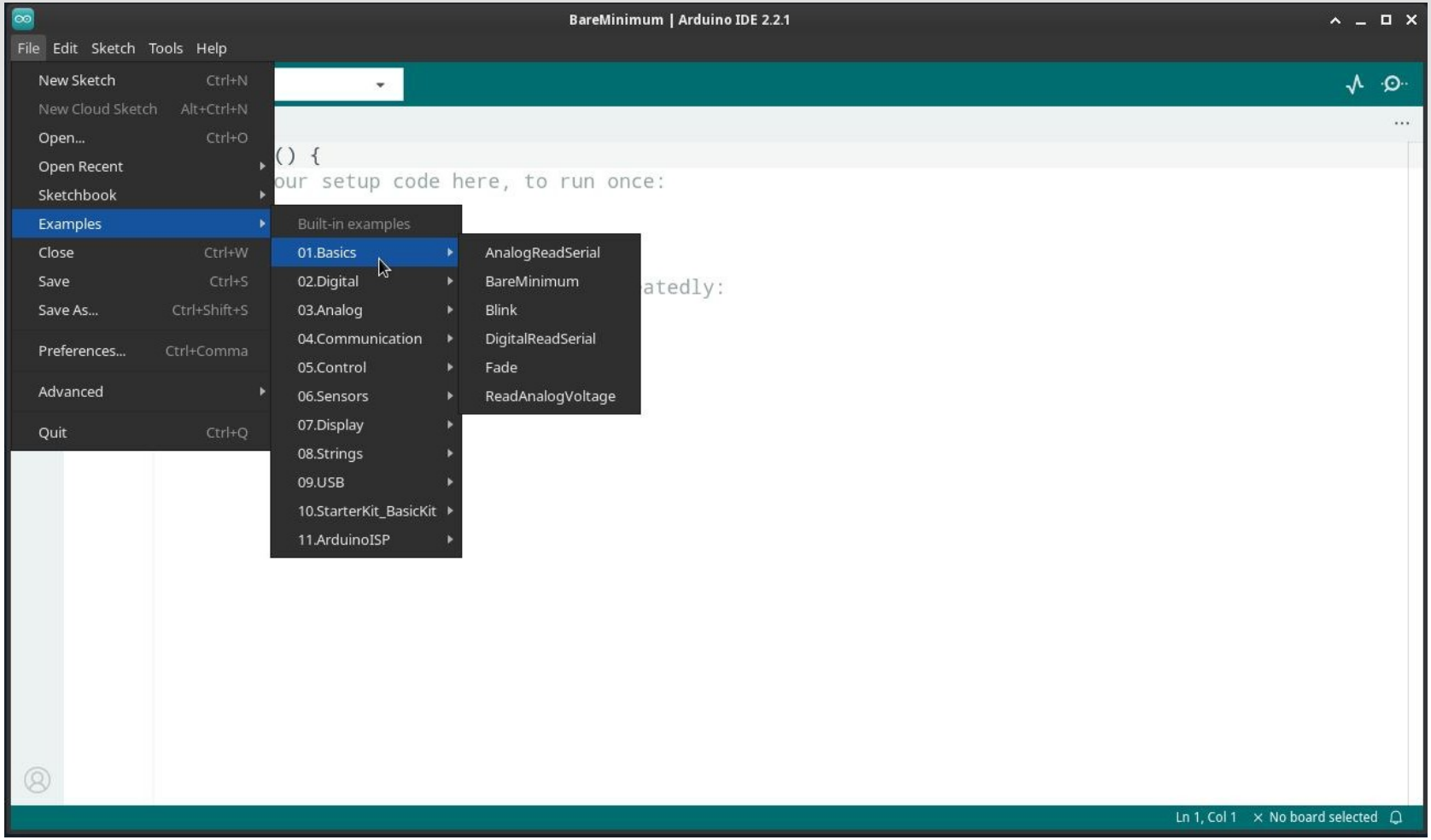
Project 5 - Real Time Color Mixer

Wiring Diagram



fritzing photo

Where To Go From Here:



Special Thanks:

roto