Exploration Badge

Custom Controller Shenanigans

My experience before this...

Project inspiration

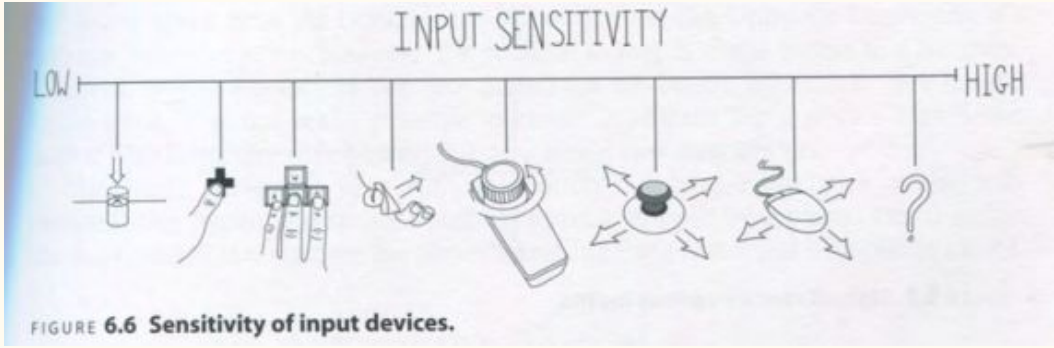


FIGURE 6.6 **Sensitivity of input devices.**

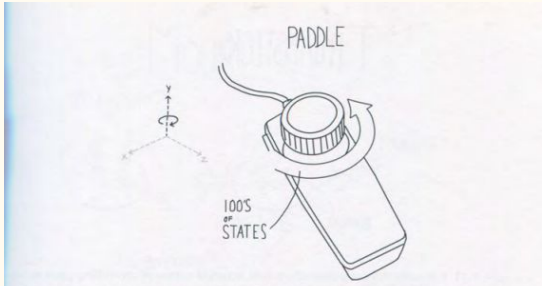


FIGURE 6.10 **An old school paddle controller: bounded rotational control in one dimension.**

one-axis rotation. There was one spinner input on the front of the controller (Figure 6.10). It was gripped between the thumb and forefinger and could be rotated left or right a certain amount before it reached a defined hard boundary point and the plastic would catch and stop it. Through a combination of factors, this input type fell out of vogue, but it was quite a sensitive input, with hundreds of possible states between fully left rotation and fully right.

A paddle controller returns a float value, in a range from −1.00 to 1.00. When the paddle knob is centered, it's at 0.00. As it is rotated left of center, it goes negative (something like −0.26) and as it is rotated right it goes positive (something like 0.41).

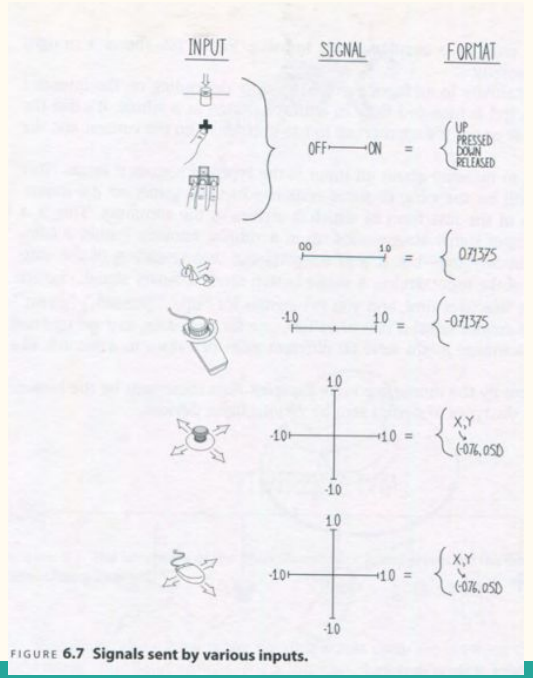


FIGURE 6.7 **Signals sent by various inputs.**

Paddle

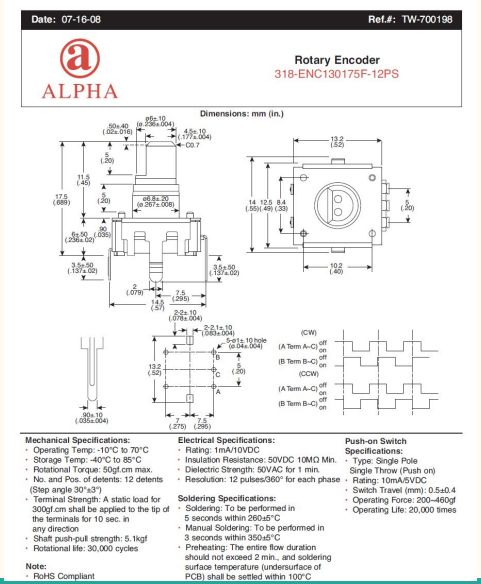
Though they're not in common use anymore, it's interesting to note that the paddle controllers sold with many of the first home consoles used a hard-boundary,

What is a potentiometer???

Think of a volume knob, where it goes from 0 - 10, and it has boundaries set that doesn't allow it to rotate all the way around. Old school paddle controllers and arcade devices used these.

What is a Rotary Encoder???

It is a knob you can rotate infinitely. It has no real state (0 - 10 value) associated with it, as it has no beginning, middle, or end. But you can keep track of the rotation in code and use it as a knob input by turning it up or down as much as you want. In other words...



Date: 07-16-08 Ref.#: TW-700198

ALPHA

Rotary Encoder
318-ENC130175F-12PS

Dimensions: mm (in.)

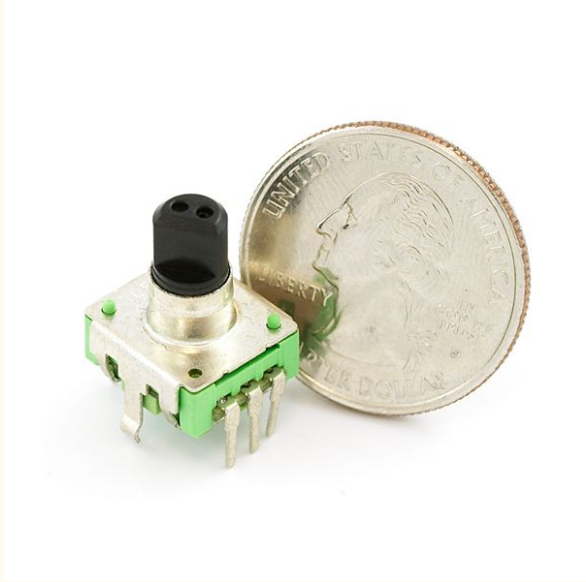
Mechanical Specifications:
- Operating Temp: -10°C to 70°C
- Storage Temp: -40°C to 85°C
- Rotational Torque: 50gf.cm max.
- No. and Pos. of detents: 12 detents (Step angle 30°±3°)
- Terminal Strength: A static load for 300gf.cm shall be applied to the tip of the terminals for 10 sec. in any direction
- Shaft push-pull strength: 5.1kgf
- Rotational life: 30,000 cycles

Note:
- RoHS Compliant

Electrical Specifications:
- Rating: 1mA/10VDC
- Insulation Resistance: 50VDC 10MΩ Min.
- Dielectric Strength: 50VAC for 1 min.
- Resolution: 12 pulses/360° for each phase

Soldering Specifications:
- Soldering: To be performed in 5 seconds within 260±5°C
- Manual Soldering: To be performed in 3 seconds within 350±5°C
- Preheating: The entire flow duration should not exceed 2 min., and soldering surface temperature (undersurface of PCB) shall be settled within 100°C

Push-on Switch Specifications:
- Type: Single Pole Single Throw (Push on)
- Rating: 10mA/5VDC
- Switch Travel (mm): 0.5±0.4
- Operating Force: 200~460gf
- Operating Life: 20,000 times

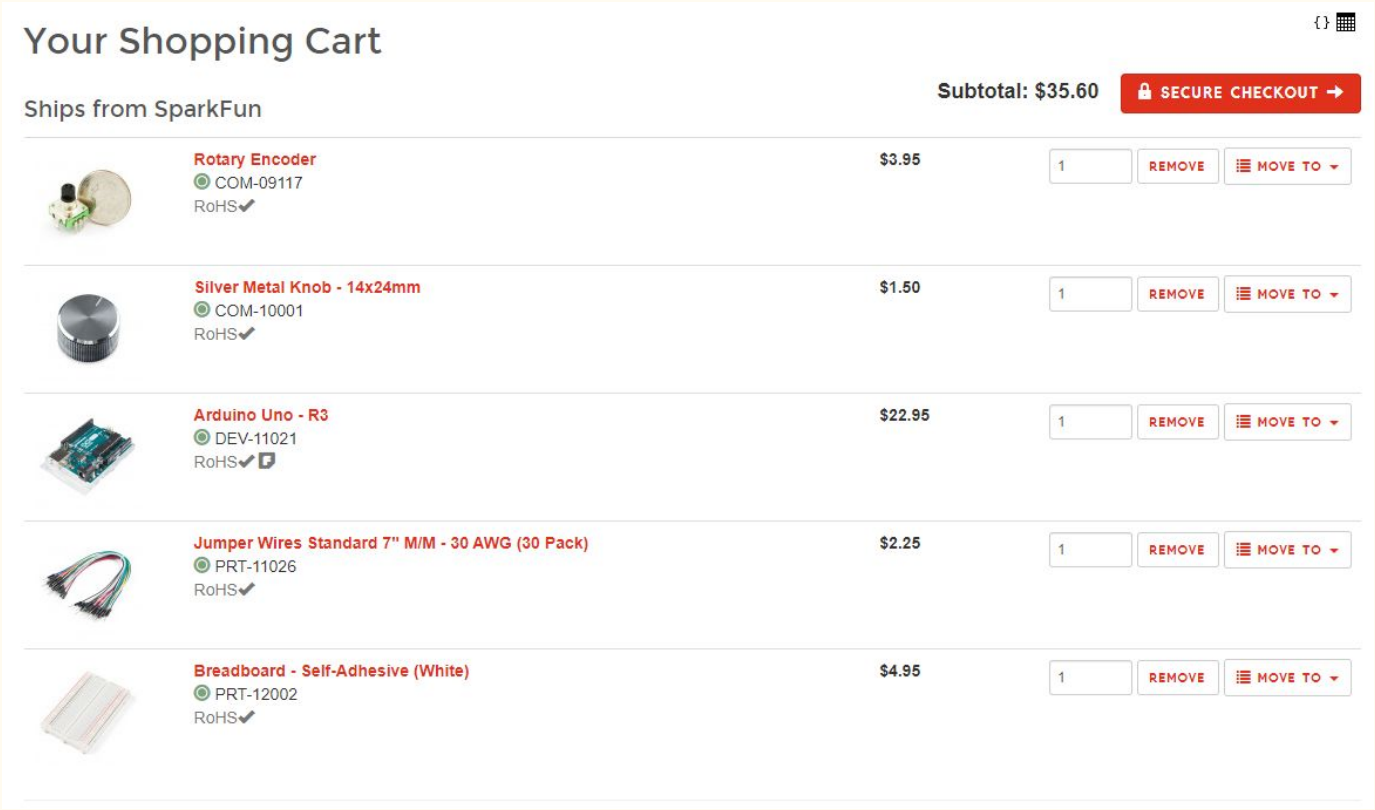


Spinnnnnn

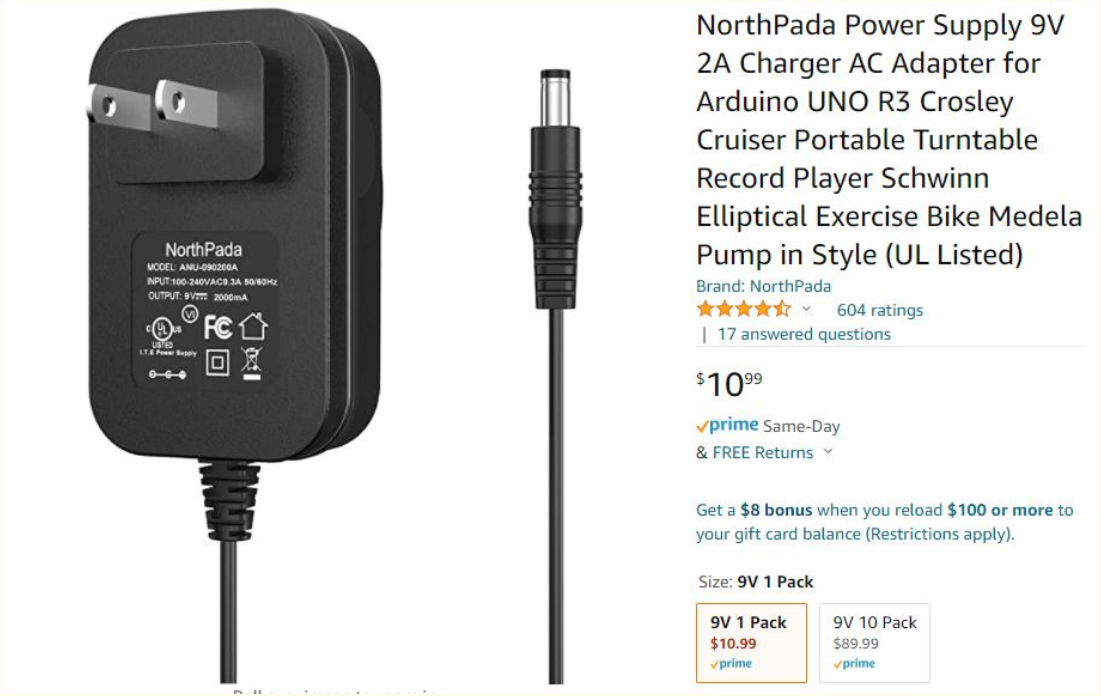
Find an inspiration

http://www.kanyezone.com/

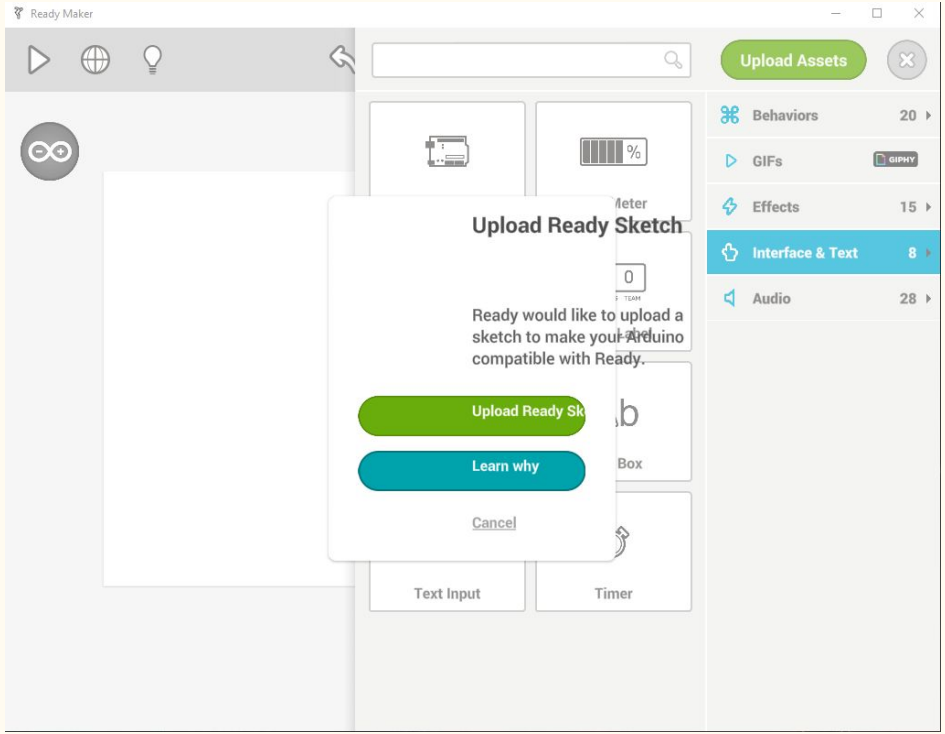
Time to go shopping!



Don’t forget the power cable! (jk you actually don’t need that lol)

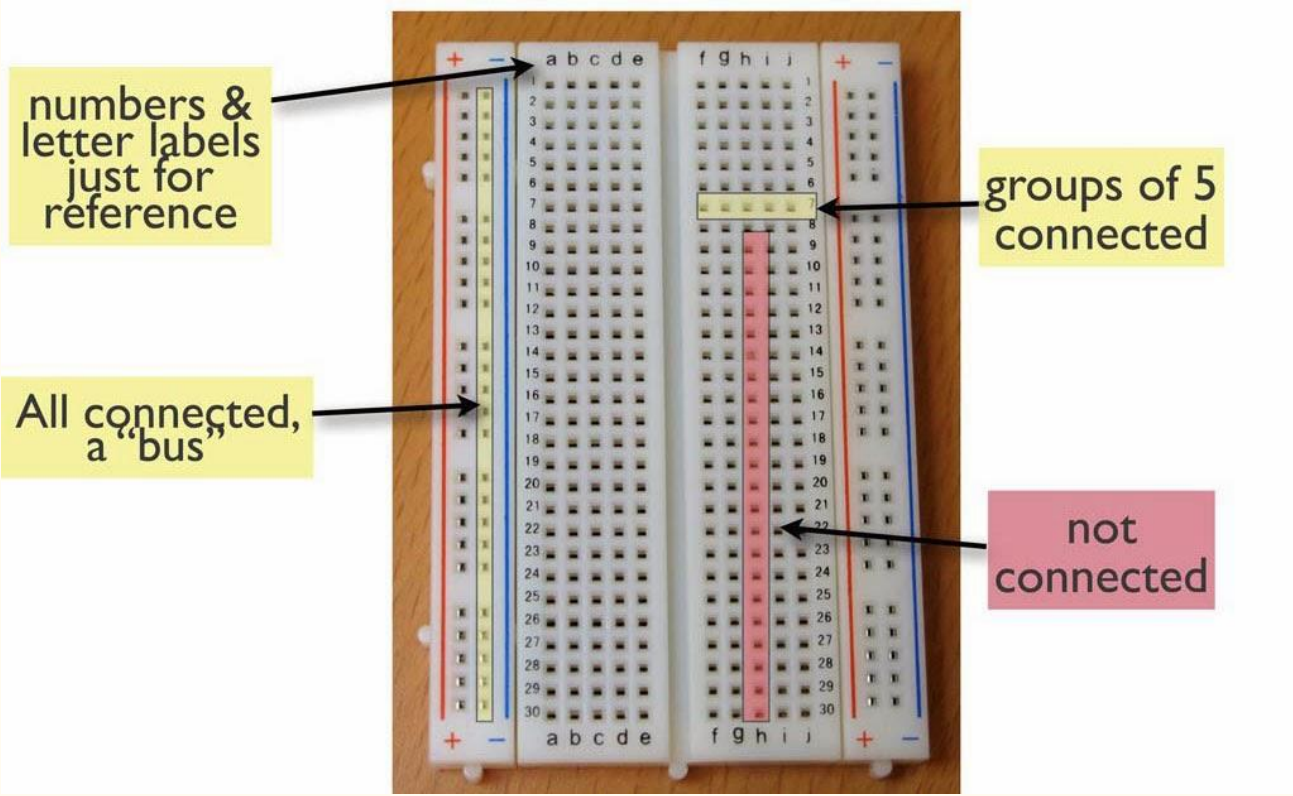


That was week one, now that we got the parts, now what? Download broken third party apps



What are these parts??? Firstly... Breadboard

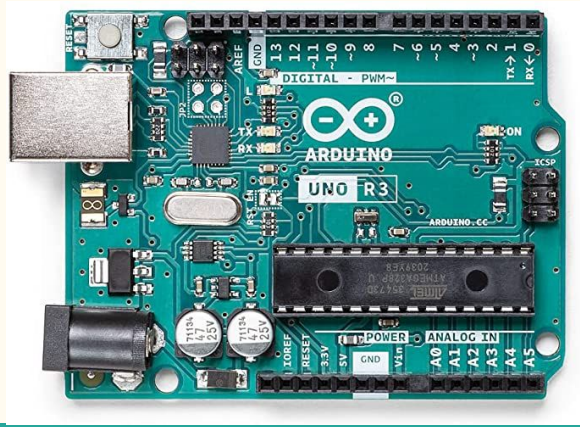
Nah just kidding, here's a real bread board



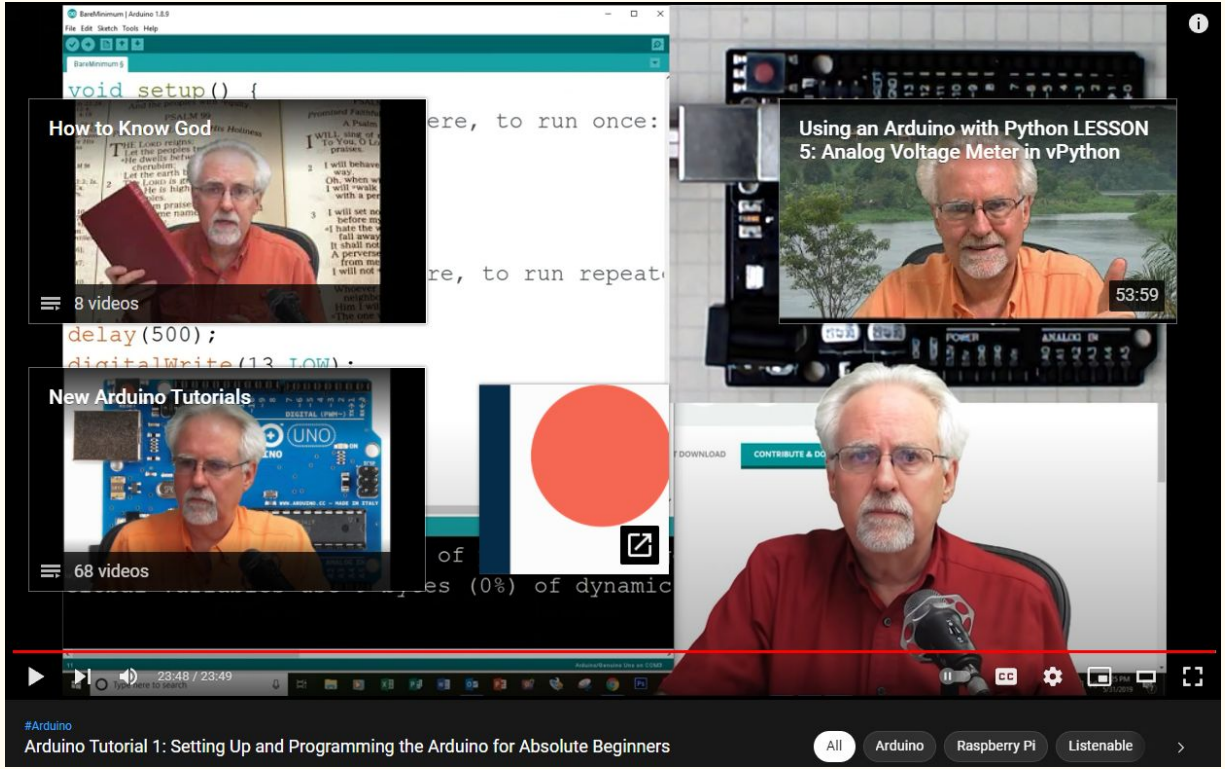
What is an Arduino Uno?

Is it like a raspberry pi? Kind of, but not really.

It is a microcontroller board. It has only 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, etc. You just need to connect it to your computer or to a power cable and bam it works. Uno means one in Italian, obviously in reference to the hit game Uno, and not to mark the release of Arduino Software version 1.0.



Ok, hardware stuff out of the way, what about programming?



A New coding language?!?!?

Yeah this kinda caught me off guard. It is technically a c++ dialect, but my main background is C#. I had to look up literally every single line that was gone over in the tutorial for the rotary encoder.

Basically, I felt like a beginning coder again, plus hardware stuff!

Unity stuff

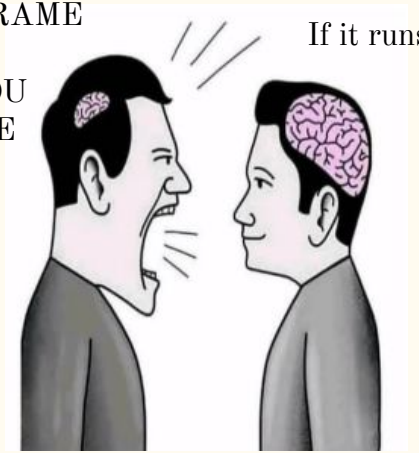
```
Unity Message | 0 references
private void Start()
{
    if (serial != null)
    {
        if (!serial.IsOpen)
        {
            serial.Open();
            //serial.ReadTimeout = 16; //give refresh rate of 60.5 fps
        }
    }
    //ii = 0;
    //serial.Write(ii.ToString());
}

Unity Message | 0 references
private void Update()
{
    currentInput = int.Parse(serial.ReadLine()) * factor;

    // Use this block for linear increase or decrease
    if (calcInput < currentInput - margin)
    {
        calcInput += Time.deltaTime * speed;
    }
    else if (calcInput > currentInput + margin)
    {
        calcInput -= Time.deltaTime * speed;
    }
    else // calcInput is very close to currentInput so we are ready to update currentInput
    {
        // Get input from arduino
        currentInput = float.Parse(serial.ReadLine()) * factor;
        //print("Arduino input: " + currentInput);
    }
    //print("Calculated input: " + calcInput);
    transform.localEulerAngles = new Vector3(0, 0, calcInput);
}
```

Basically, Arduino and Unity are talking to each other. Arduino outputs a value, and Unity tries to match that.

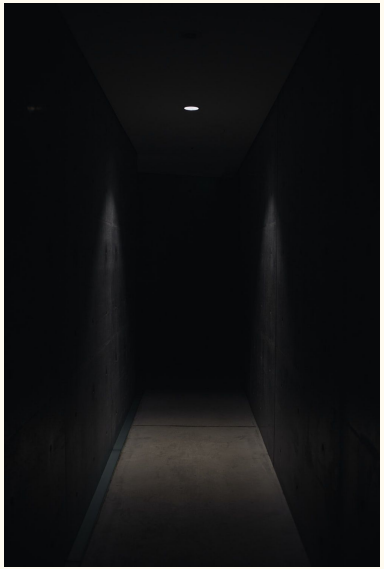
WHY ARE YOU RUNNING THIS ON EVERY FRAME THAT'S BAD FOR PERFORMANCE YOU NEED TO OPTIMIZE



If it runs, it's done

Does it work?

Kind of. It definitely needs more polish, but the general idea is there. If given more time, I'd learn more about Arduino, but I don't know if it would be useful to this specific project. There were many things I just had no idea how to do or even know what to ask.



Bad news...

No build was able to be had, and My Macbook Pro couldn't handle Arduino's constant stream of numbers.

But, I still got it from Editor!

My experience after this...