PAT 451/551 INTERACTIVE MEDIA DESIGN I

SENSORS

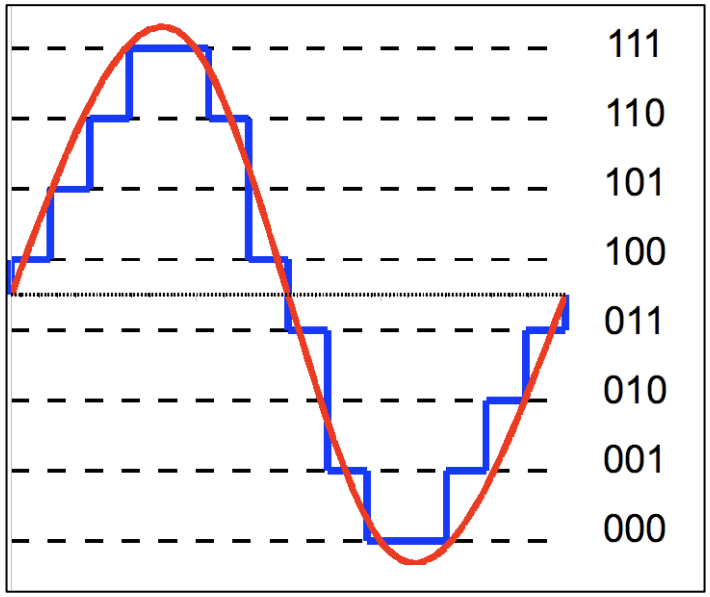
SENSING CONTINUOUS CHANGES

- Until now, we've been able to detect ***discrete*** changes in the environment—buttons that are on or off
  - Digital inputs
  - 0V or 5V signals

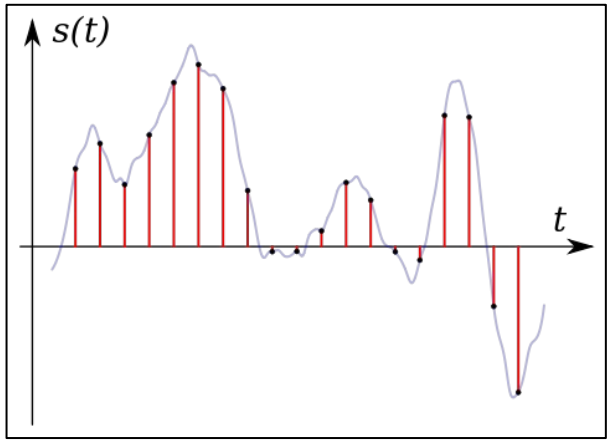
- Now, we want to also detect ***continuous*** variations
  - Analog inputs
  - Signals varying anywhere between 0V and 5V

ANALOG-TO-DIGITAL CONVERSION (ADC)

- **Quantization**
  - Since we have a digital computer, we can only represent a continuously varying voltage as discrete steps.
  - The more steps we have, the closer the approximation to the original signal.

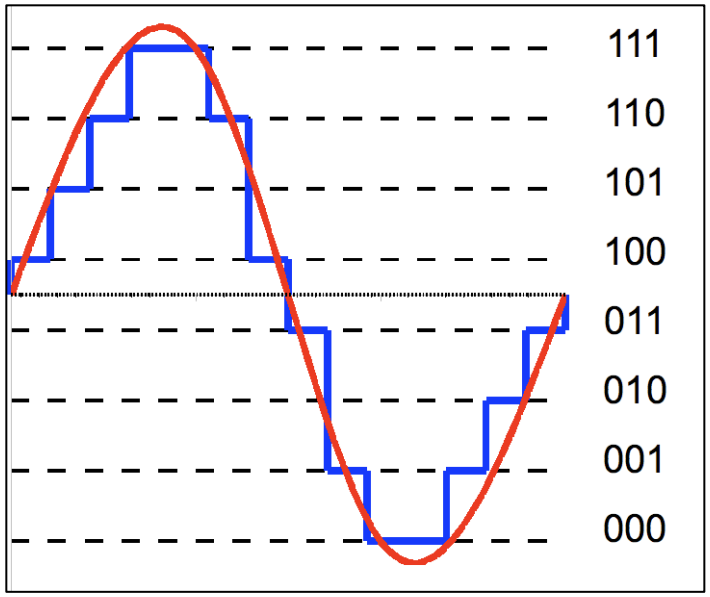


- **Sampling**
  - We can only measure at discrete time points
  - Unlike with audio systems, we don't have a fixed sample rate.



QUANTIZATION

- Quantization to an **n-bit** number gives 2^n discrete levels
  - Available values are **0 to** $(2^n - 1)$ if we're only using positive numbers
  - Audio standard: 16-bit (65536 levels)
  - Online image standard: 8 bits (256 levels) per color
  - MIDI data (pitch, velocity, etc): 7 bits (128 levels)
- The **Arduino** has a **14-bit** analog-to-digital converter, so we can represent **16384 levels**, with numbers from 0–16383



3-bit quantization:
8 discrete values,
0 to 7 (decimal)
000 to 111 (binary)

ANALOGREAD FUNCTION

```
int analogRead(pin);
```

By default, analogRead uses 10-bit resolution, so it returns a number from 0 to 1023, corresponding to the analog voltage on *pin.*

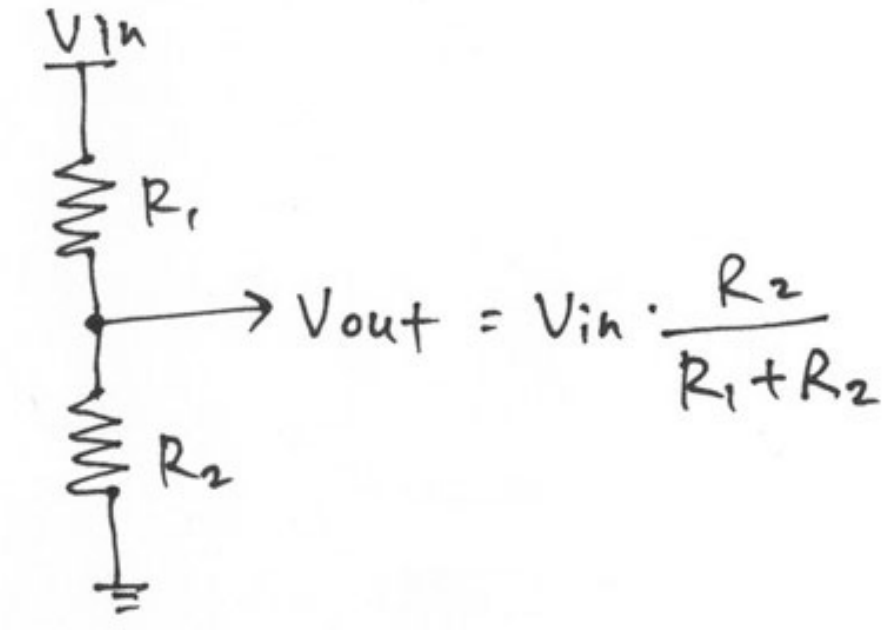
pin is a number from 0 to 5 on the Arduino Uno, corresponding to the numbered analog inputs.

Example usage:

```
int reading = analogRead(0);
```

See **a2dtoSerial.ino** example program.

VOLTAGE DIVIDER

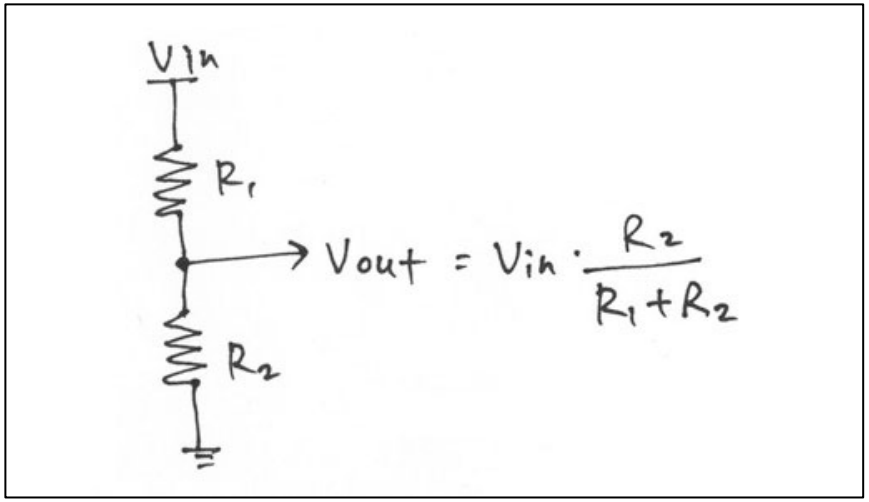


V is proportional to R, so the voltage drop across R_2 decreases as R_2 gets smaller.

By varying R_1 and/or R_2, we can vary V_{out} while V_{in} remains constant.

EXERCISE

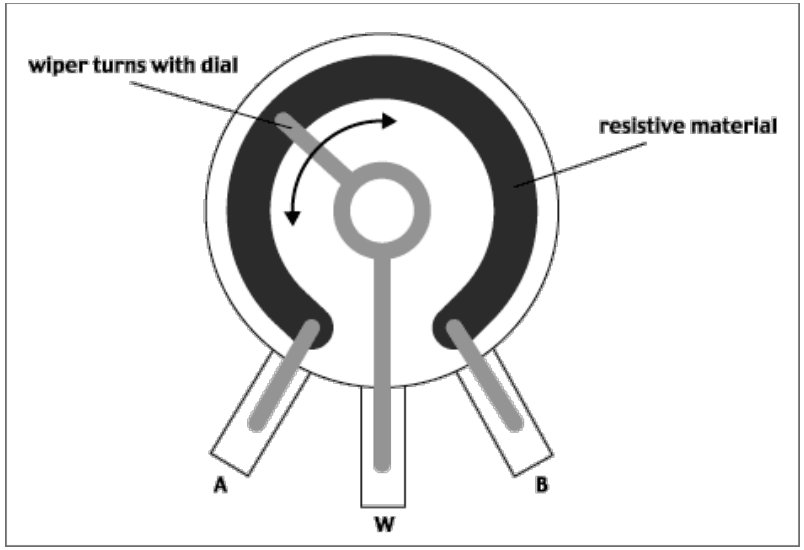
- Choose 2 different resistor values and build a voltage divider circuit.
- Measure your resistors.
- Measure your input voltage (should be around 5V).
- Predict what V_{out} should be based on the measured values.
- Connect V_{out} to analog input 0 on your Arduino.
- Predict what number should be returned by **`analogRead(0)`**:
  `Vout / Vin * 1024`
- Run the **`a2dtoSerial.ino`** example program and compare to your prediction

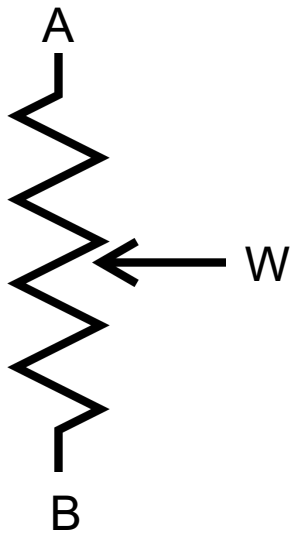


ROTATION SENSING

Potentiometer

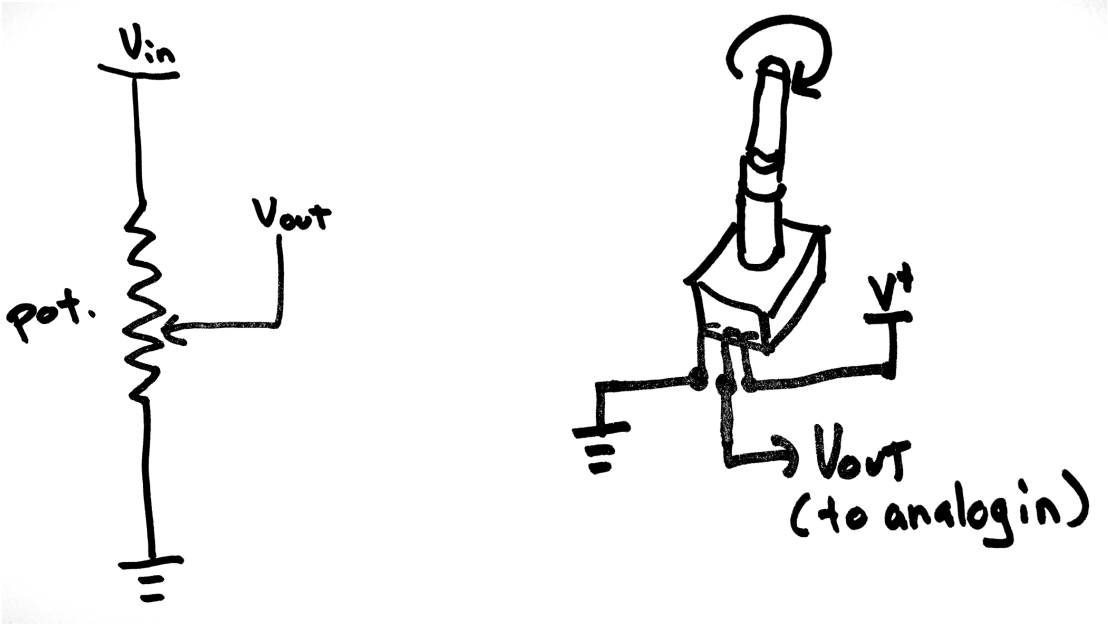
- Resistive material
- Wiper moves along material
- Resistance between terminal and wiper varies with distance
- Rotation range is limited by the physical material, usually a bit less than 360˚
- Some multi-pots can make more than 1 full rotation, using either a gear mechanism or a spiral resistive element.



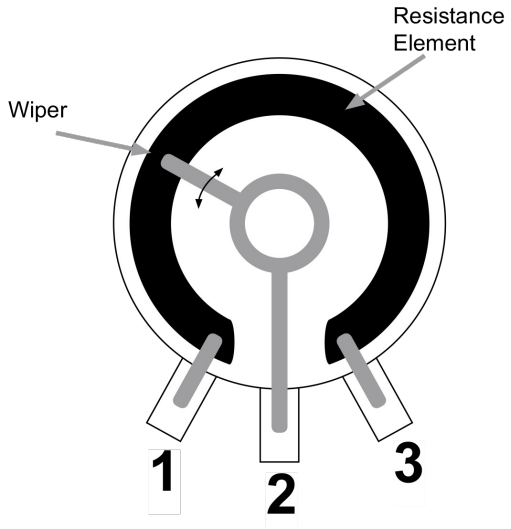


ROTATION SENSING

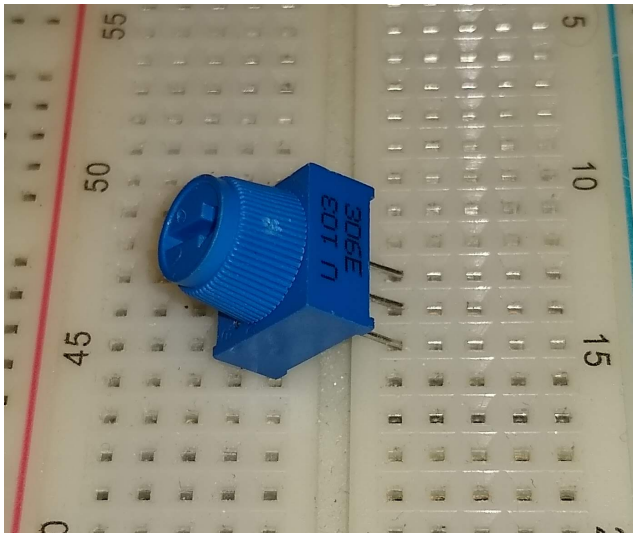
- The wiper “divides” the resistance into 2 parts. If we connect the outer legs of the pot to +5V and Gnd, we get a voltage divider.
- V_{out} changes as you rotate the pot.
- Assuming the change in resistance of the material is linear over its length, V_{out} will vary proportionally to the wiper angle.
- Connect V_{out} to an **analog input** of the Arduino to measure the voltage



POTENTIOMETERS

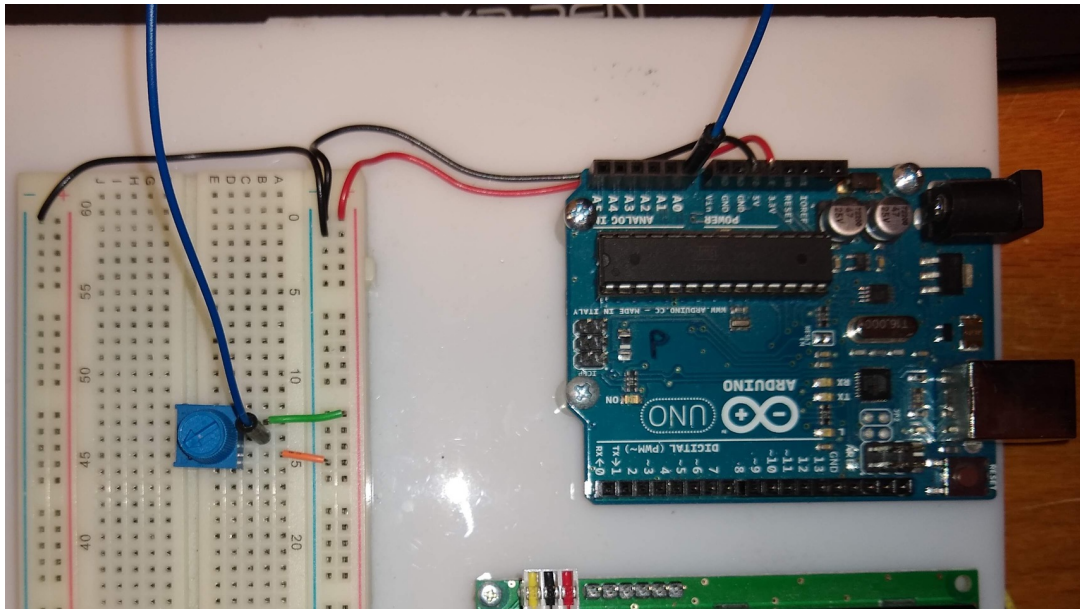


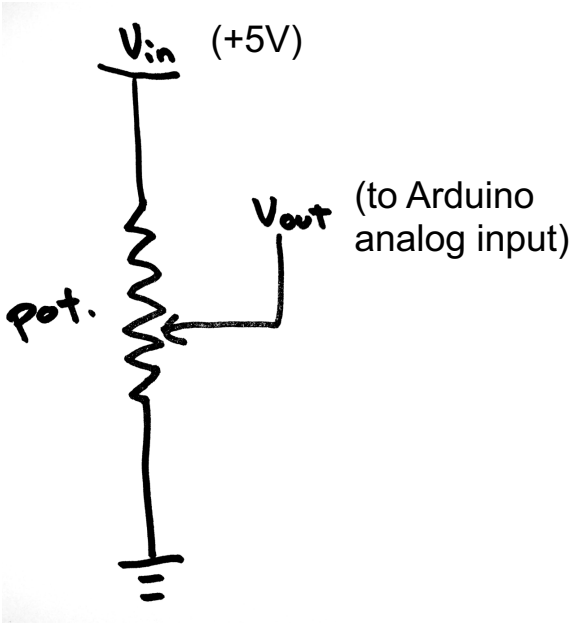
- Pin 2 (middle pin) is always the wiper.
- Pins 1 and 3 are the endpoints.
- The resistance between pins 1 and 3 is constant, and is the rated resistance of the potentiometer.
- Ours are 10kΩ.



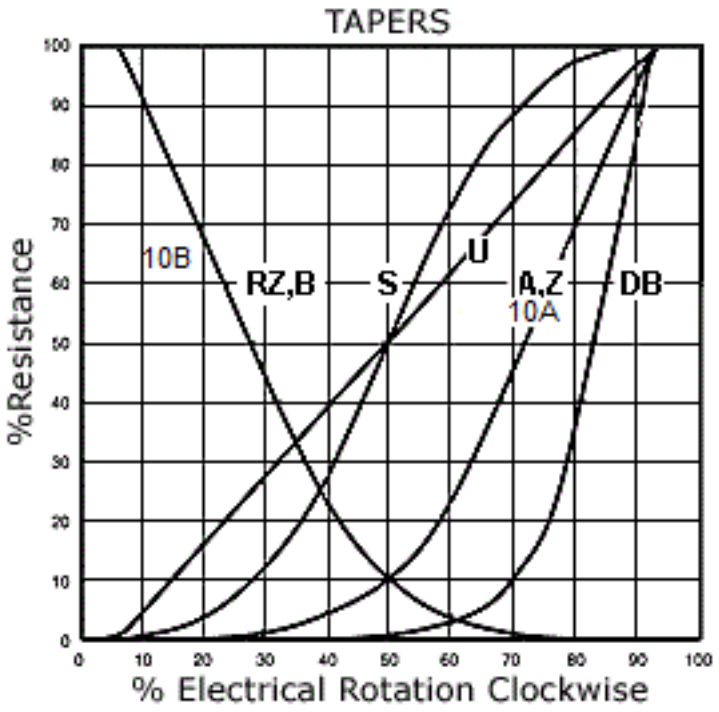
- These are known as “trimmers” or trimpots.” Handy because they fit into our breadboards.

POTENTIOMETER CIRCUIT



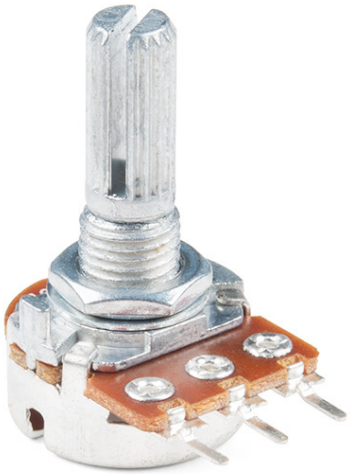


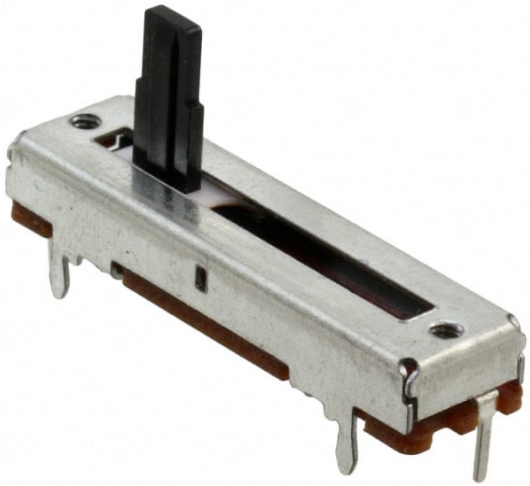
POTENTIOMETER TAPERS

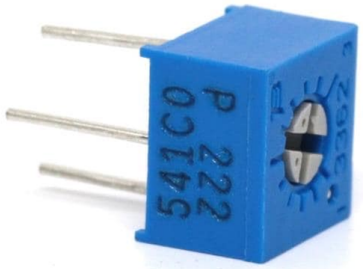


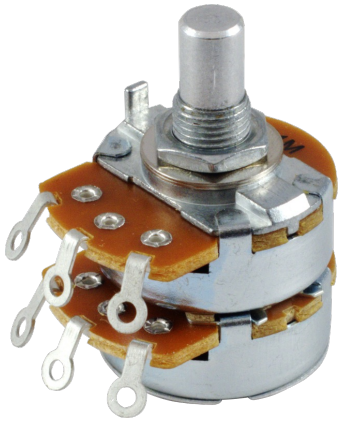
- ”Taper” indicates relationship between rotation and resistance in a potentiometer
- Visualized as a curve
- U curve here is linear
- “Audio Taper” or “Log taper” (A,Z) here usually means 10% of the resistance is at 50% rotation. 15% is also common.
  - Designed to represent how we hear changes in loudness for audio circuits.
- When using a pot as a sensor, it’s always good to check whether it’s linear first.

OTHER POTENTIOMETER FORMS



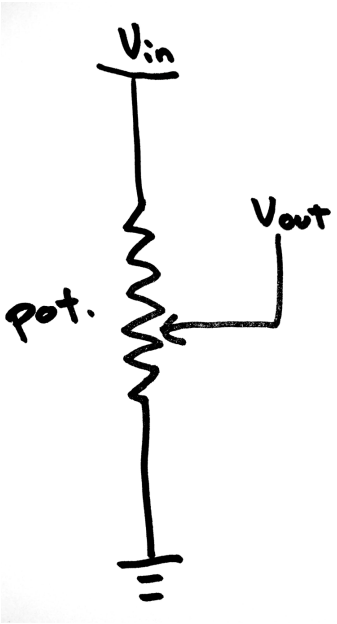


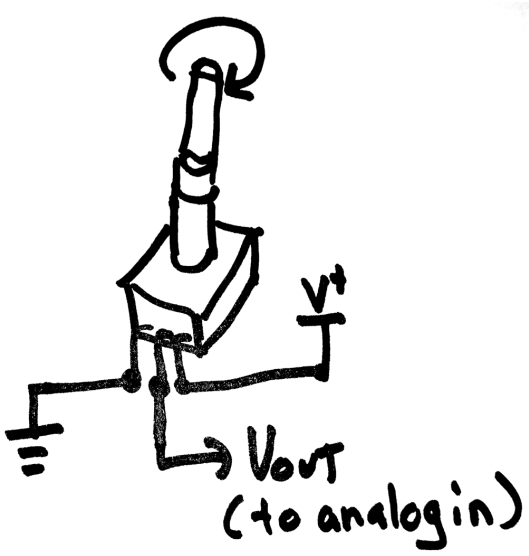




EXERCISE

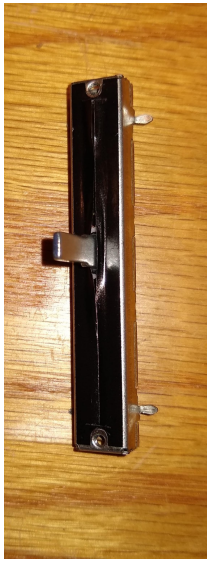
- Wire up your blue knob potentiometer as a voltage divider.
- Run the **`a2dtoSerial.ino`** example program and see the results.
- Does the potentiometer have a linear taper?

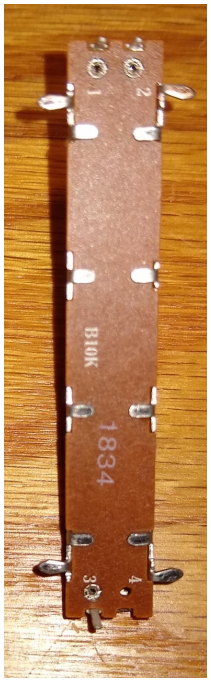




SLIDE POTENTIOMETER

- Works the same as a rotary pot, except resistive material is arranged in a straight line.
- Can often have audio taper too (think mixing boards)

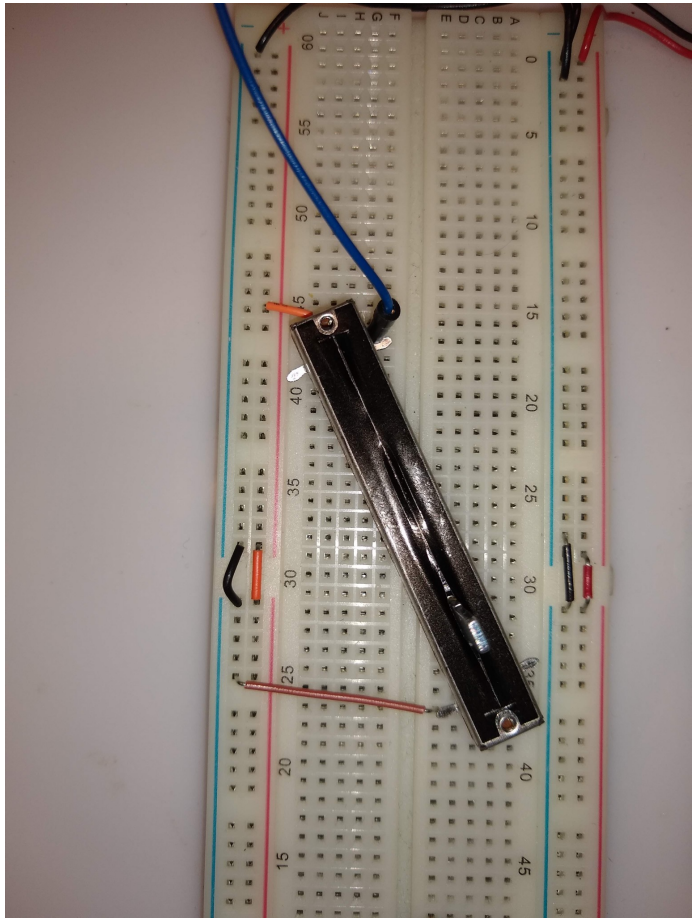




- Our slide pots have 3 numbered pins on the bottom.
- Pin 2 is the wiper
- Pins 1 and 3 are the fixed endpoints

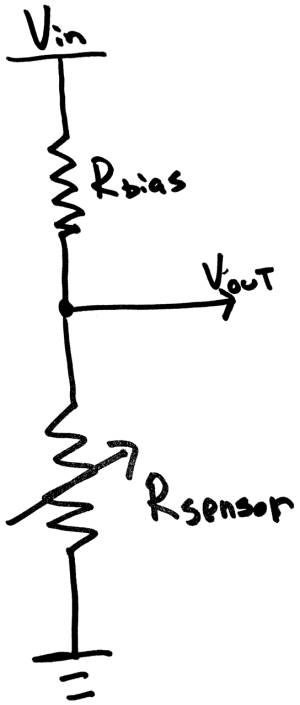
SLIDE POTENTIOMETER

- Very carefully bend the mounting tabs on the sides of the slide pot upward
- Then you can *very carefully* insert the 3 legs of the pot onto the breadboard, if you mount it at an angle like this.
- Ensure that pins 1 and 2 are in separate rows.
- You might need to *gently* bend one of the pins.



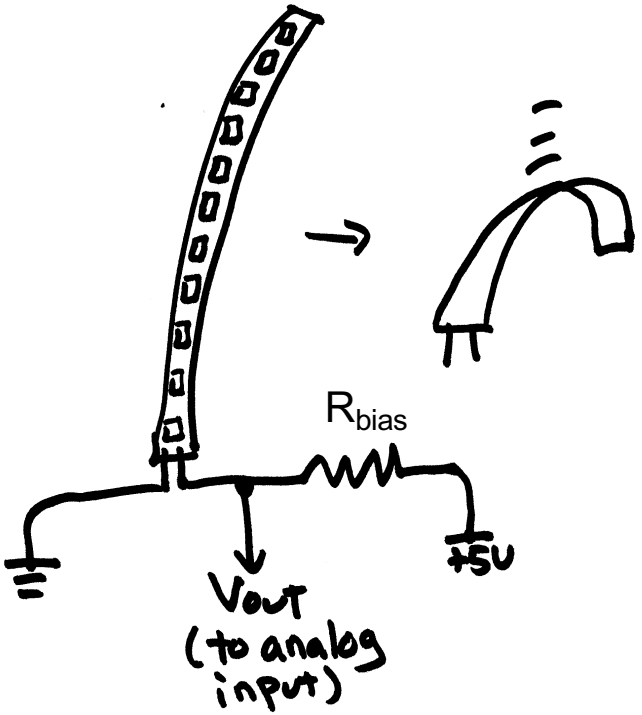
OTHER RESISTIVE SENSORS

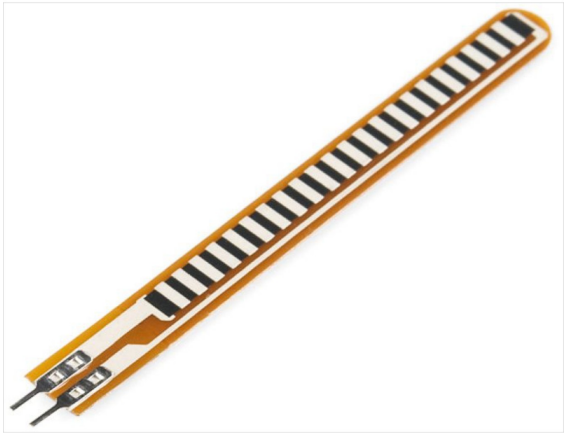
- Some sensors have a **varying resistance** between their 2 leads, depending on some **change in a physical property**
- Need a **fixed resistor** to make a voltage divider: "bias resistor"
- Resistance of the fixed resistor affects **range of output voltage**
- Unless R_{sensor} goes from 0 to infinity, V_{out} won't go exactly from 0–5V.



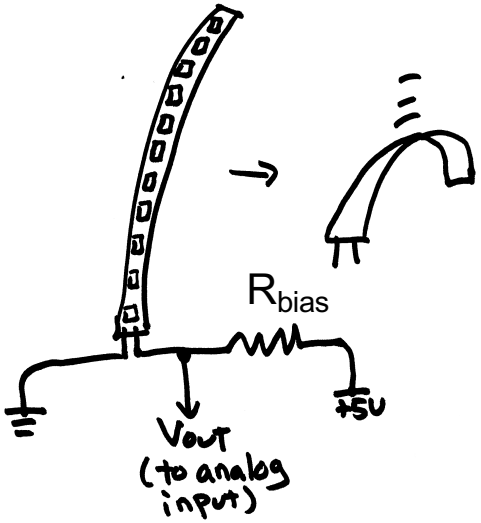
FLEX SENSOR

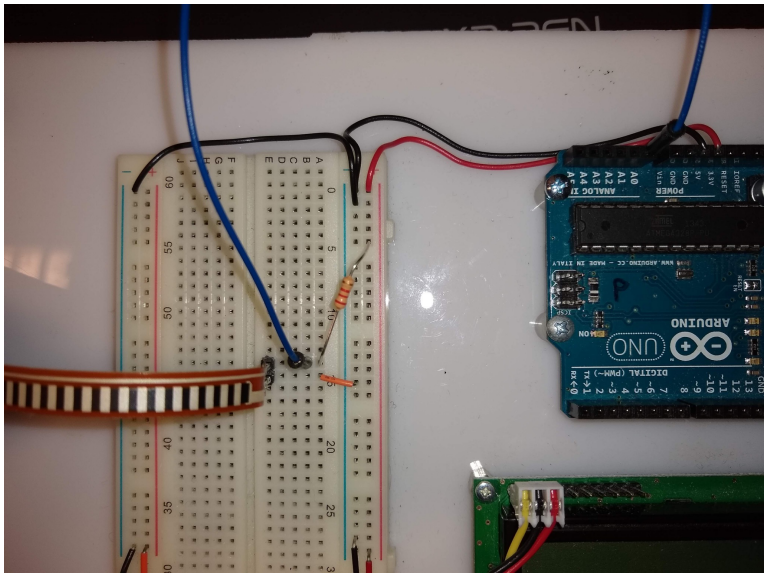
- A flex sensor changes resistance as you bend it
- Most sensors only work in 1 direction
- Used in data gloves; sensor is usually finger-length
- Costs about $10/ea
- Pins are somewhat fragile





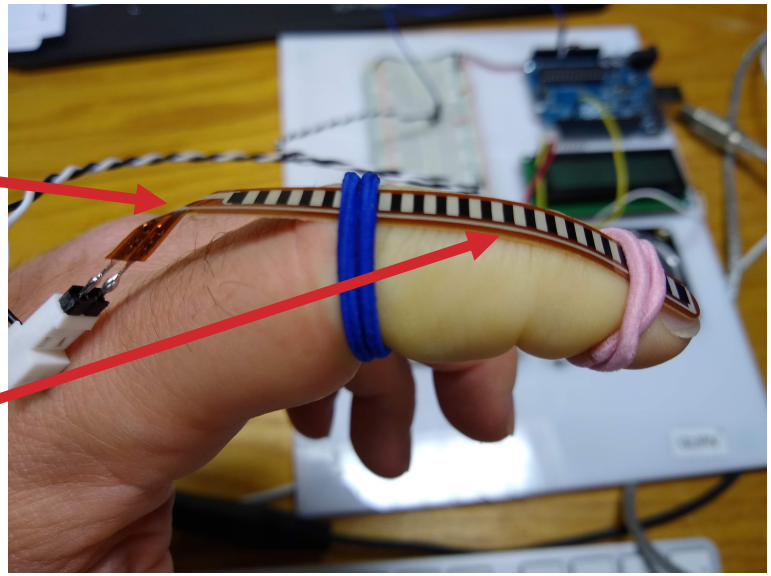
FLEX SENSOR



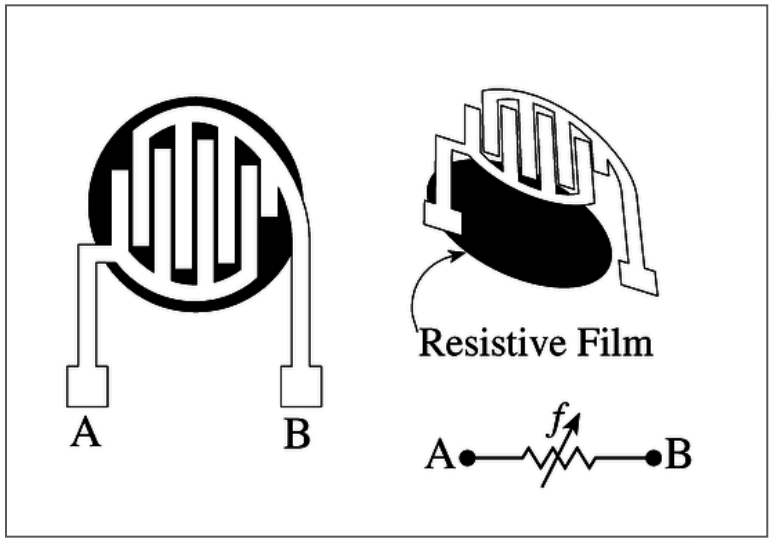


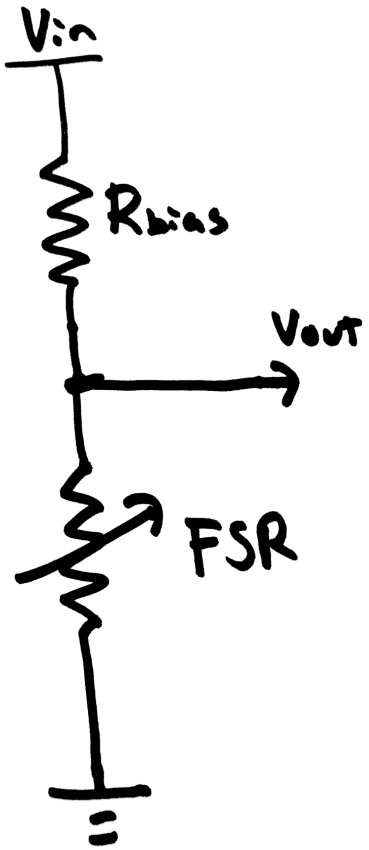
Try not to bend this part (pin mounting is fragile)

Bend this part (in the direction shown)



FORCE SENSING RESISTOR (FSR)



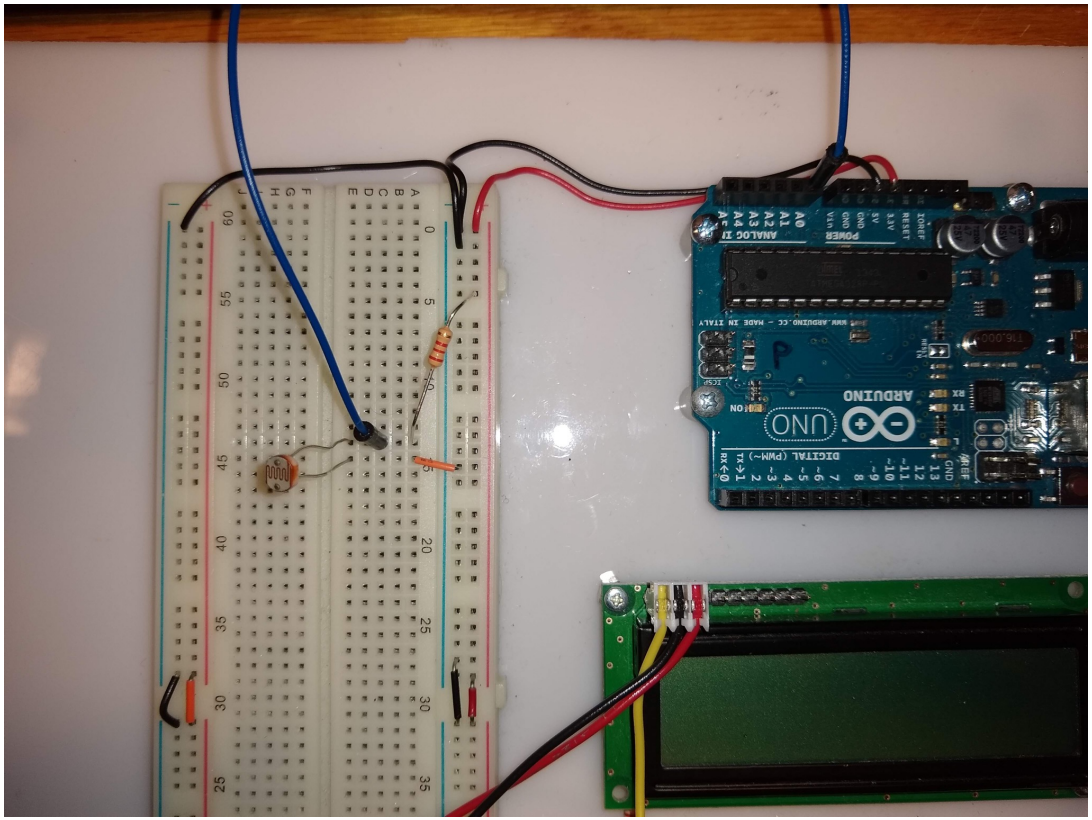


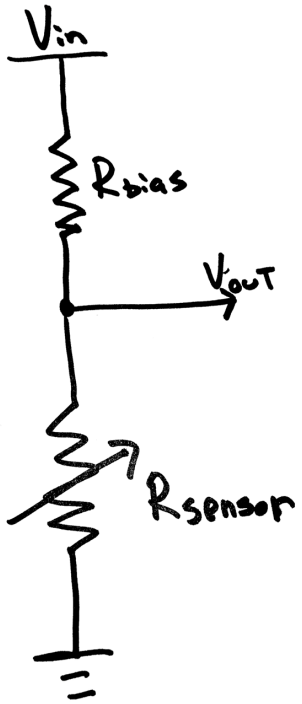
- Resistive polymer film
- Interdigitating contacts
- Resistance inversely proportional to force (resistance goes down as you press harder)

PHOTOCELL

- AKA: photodetector, photosensor, photo resistor, Light Dependent Resistor (LDR)
- Sometimes known by the semiconductive chemical used: Cadmium Sulfide (CdS)
- Resistance decreases with increasing light intensity
  - Brighter light->less resistance
- Commonly used as a sensor for street lights, garden lights, camera light meters, laser beam-break detector

PHOTOCELL CIRCUIT





FINDING THE OPTIMUM BIAS RESISTOR

- Measure the minimum and maximum resistances from your sensor.
- Calculate the geometric mean:

$$R_{bias} = \sqrt{R_{min} \cdot R_{max}}$$

- Find the closest resistor you can.
- If you need something precise, you can always use 2 or more resistors in series.
- Calculate the minimum and maximum voltages you expect:

$$V_{min} = 5 \frac{R_{min}}{R_{min} + R_{bias}}$$

$$V_{max} = 5 \frac{R_{max}}{R_{max} + R_{bias}}$$

EXAMPLE

- Measure the minimum and maximum resistances from your sensor.
- Calculate the geometric mean:

$$R_{bias} = \sqrt{1502 \cdot 18270}$$

$$R_{bias} = 5238\Omega$$

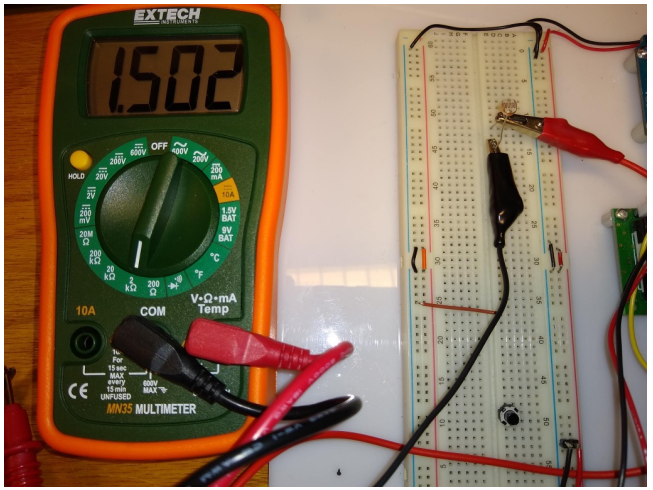
- Find the closest resistor you can:

Green-blue-red-gold: 5.6 kΩ

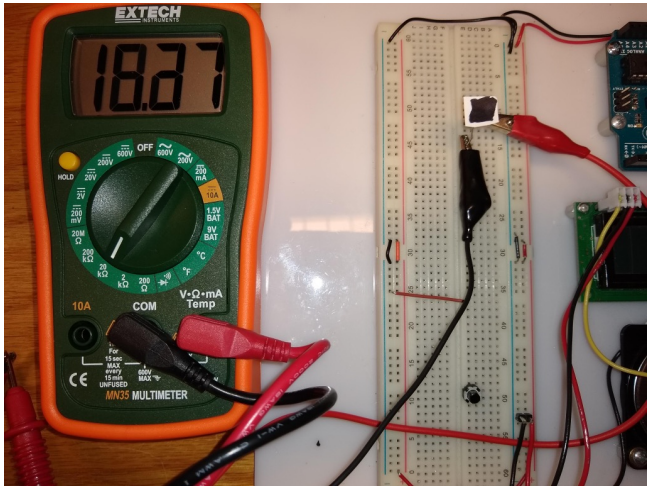
- Calculate the minimum and maximum voltages you expect:

$$V_{min} = 5\frac{R_{min}}{R_{min} + R_{bias}} = 5\frac{1502}{1502 + 5600} = 1.06\text{V}$$

$$V_{max} = 5\frac{R_{max}}{R_{max} + R_{bias}} = 5\frac{18270}{18270 + 5600} = 3.83\text{V}$$



R_{min} = 1.502kΩ



R_{max} = 18.27kΩ

EXAMPLE

- Now calculate predicted values returned by analogRead() :

$$1024 * \frac{1.06V}{5V} = \mathbf{217}$$

$$1024 * \frac{3.83V}{5V} = \mathbf{784}$$

- So we should expect to see values in the range of 217–784 in the Serial monitor.
- Take note: in the future, when we want to control things (i.e. music parameters) with these data, it's useful to know what range of values we should get, and not just guess.

ANALOG-TO-DIGITAL CONVERSION: SUCCESSIVE APPROXIMATION

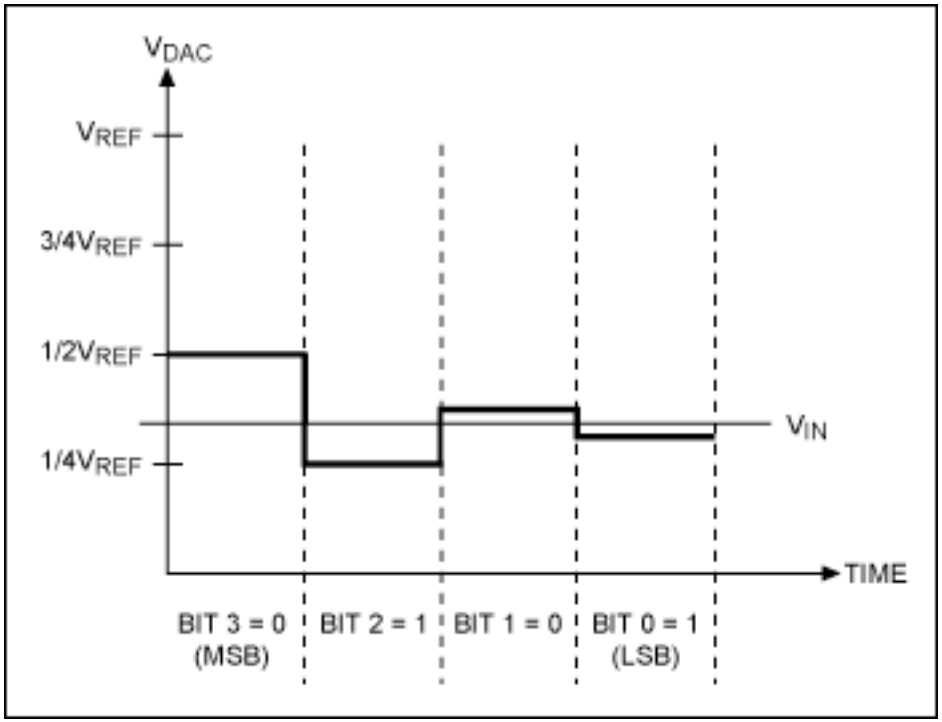
- Define n bits B_0 to B_{n-1}
  - Each bit can have the value 1 or 0
  - B_{n-1} is the MSB; B_0 is the LSB
- Initialize all bits to 0
- **For each value of k from 1 to n:**
  - **Temporarily set B_{n-k} to 1**
  - **Calculate V_{COM}:**

$$V_{COM} = V_{REF}\left[\frac{B_{n-1}}{2^1} + \frac{B_{n-2}}{2^2} + \frac{B_{n-3}}{2^3} + \ \ldots\right]$$

(Initially, this is just Vref / 2)

  - **If $V_{in} > V_{COM}$: keep B_{n-k} at 1**
  - **If $V_{in} < V_{COM}$: set B_{n-k} to 0**
  - **Repeat**

ANALOG-TO-DIGITAL CONVERSION



- Successive Approximation 4-bit ADC example

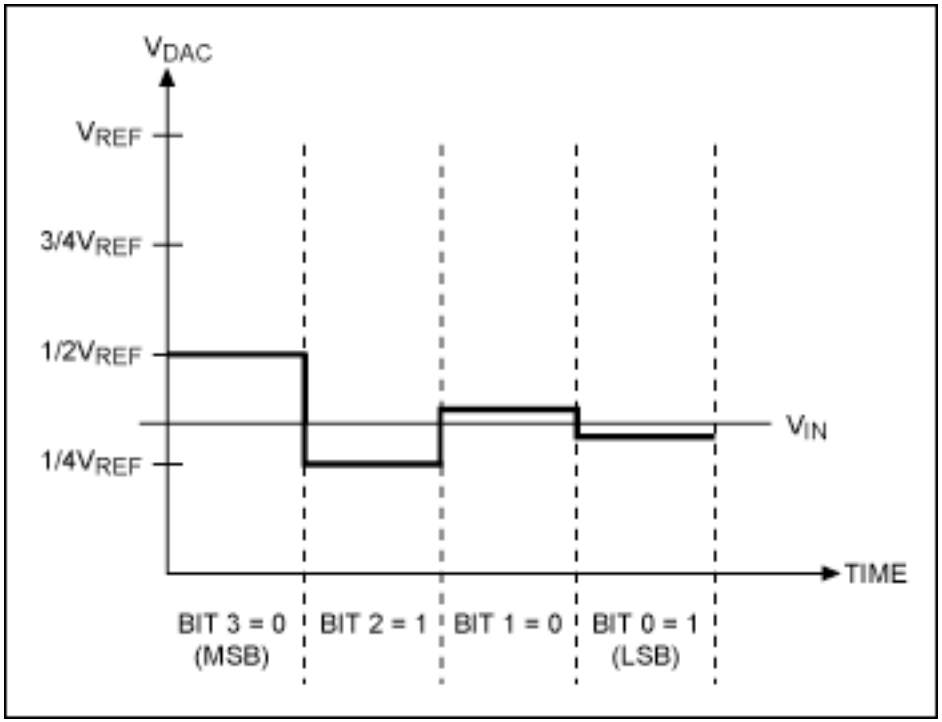
k = 1:
- **$V_{COM} = V_{REF} / 2$**
- **$V_{in} < V_{COM}$**
- B_3 is 0

k = 2:
- **$V_{COM} = V_{REF} / 4$**
- **$V_{in} > V_{COM}$**
- B_2 is 1

k = 3:
- **$V_{COM} = V_{REF} / 4 + V_{REF} / 8$**
- **$V_{in} < V_{COM}$**
- B_1 is 0

k = 4:
- **$V_{COM} = V_{REF} / 4 + V_{REF} / 16$**
- **$V_{in} > V_{COM}$**
- B_0 is 1

ANALOG-TO-DIGITAL CONVERSION



The Resulting Binary Number Is:

0 1 0 1

$= 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0$

$= 0 + 4 + 0 + 1$

$= 5$

Maximum value is $2^4 - 1 = 15$

- Successive Approximation 4-bit ADC example

ANALOGREAD RESOLUTION

```
int analogReadResolution(int);
```

By default, analogRead uses 10-bit resolution. This can be increased to 12-bit or 14-bit on several Arduino devices by passing the argument 12 or 14.

analogRead() will then return numbers in the range of 0–4095 for 12-bit and 0–16383 for 14-bit resolution

MAP FUNCTION FOR LINEAR SCALING

```
long map(value,fromLow,fromHi,toLow,toHi);
```

map linearly scales the number *val* in the range of *fromLow – fromHi* to a number in the range of *toLow – toHi*

It does not limit the input or output to be in the ranges specified.

It uses integer math, so values may be truncated or subject to rounding errors.