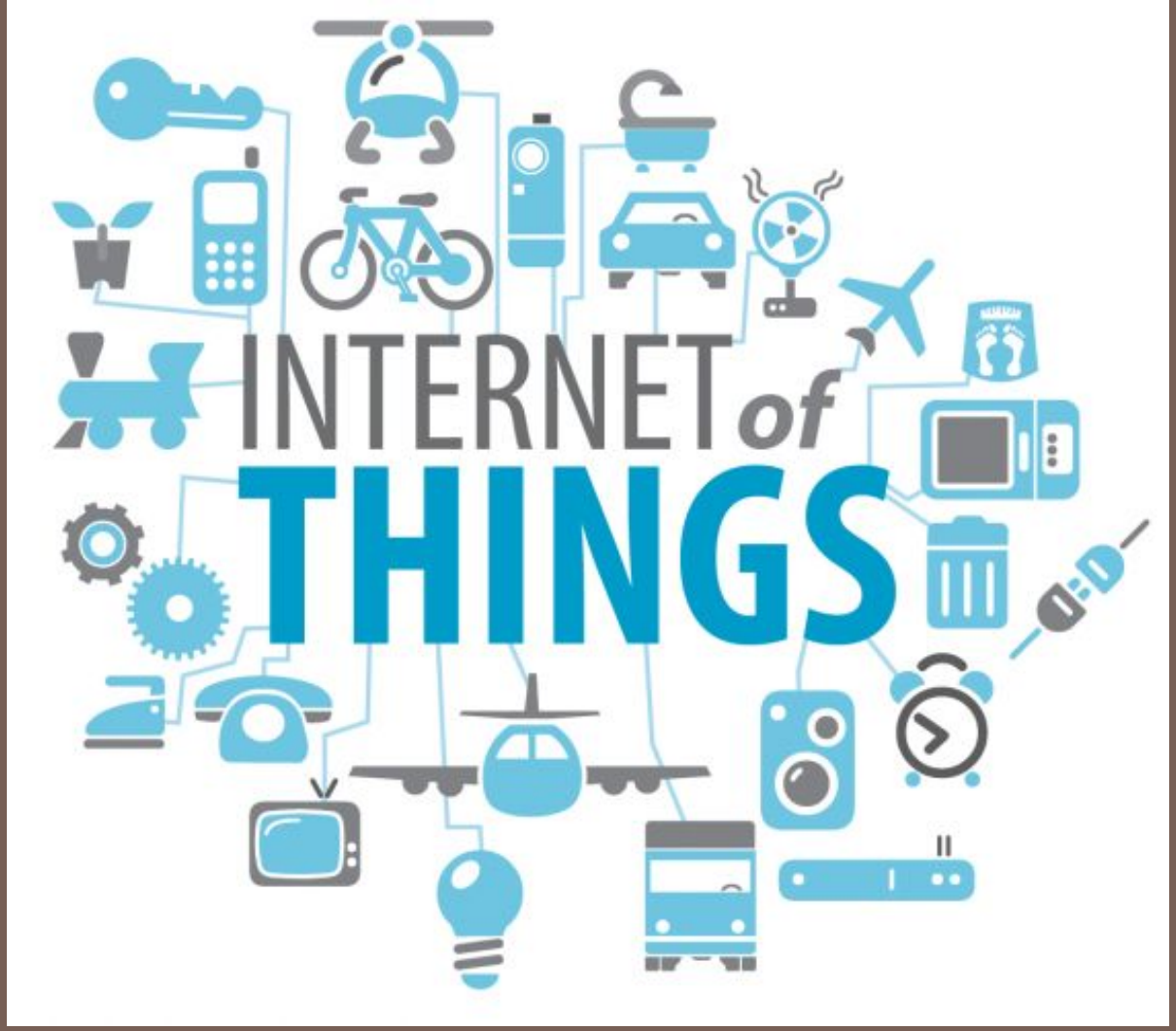


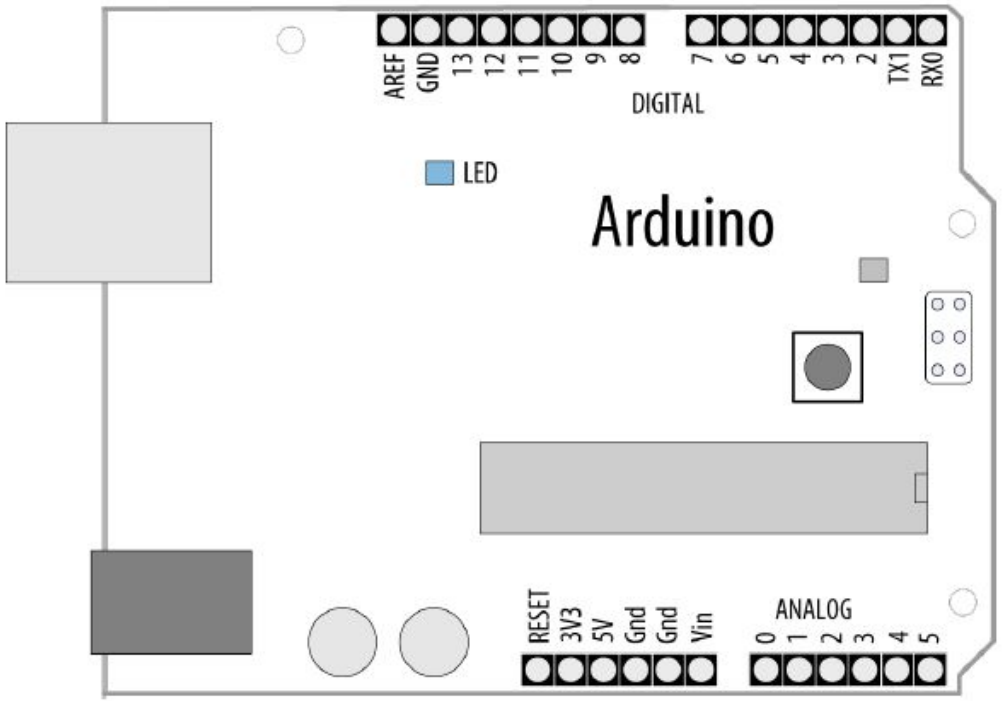
IMPLEMENTING IOE

Week 4

Assist. Prof. Rassim Suliyev - SDU 2017

Simple Digital and Analog Inputs

- The Arduino's ability to sense digital and analog inputs allows it to respond to you and to the world around you
- Digital input pins sense the presence and absence of voltage on a pin
- Analog input pins measure a range of voltages on a pin

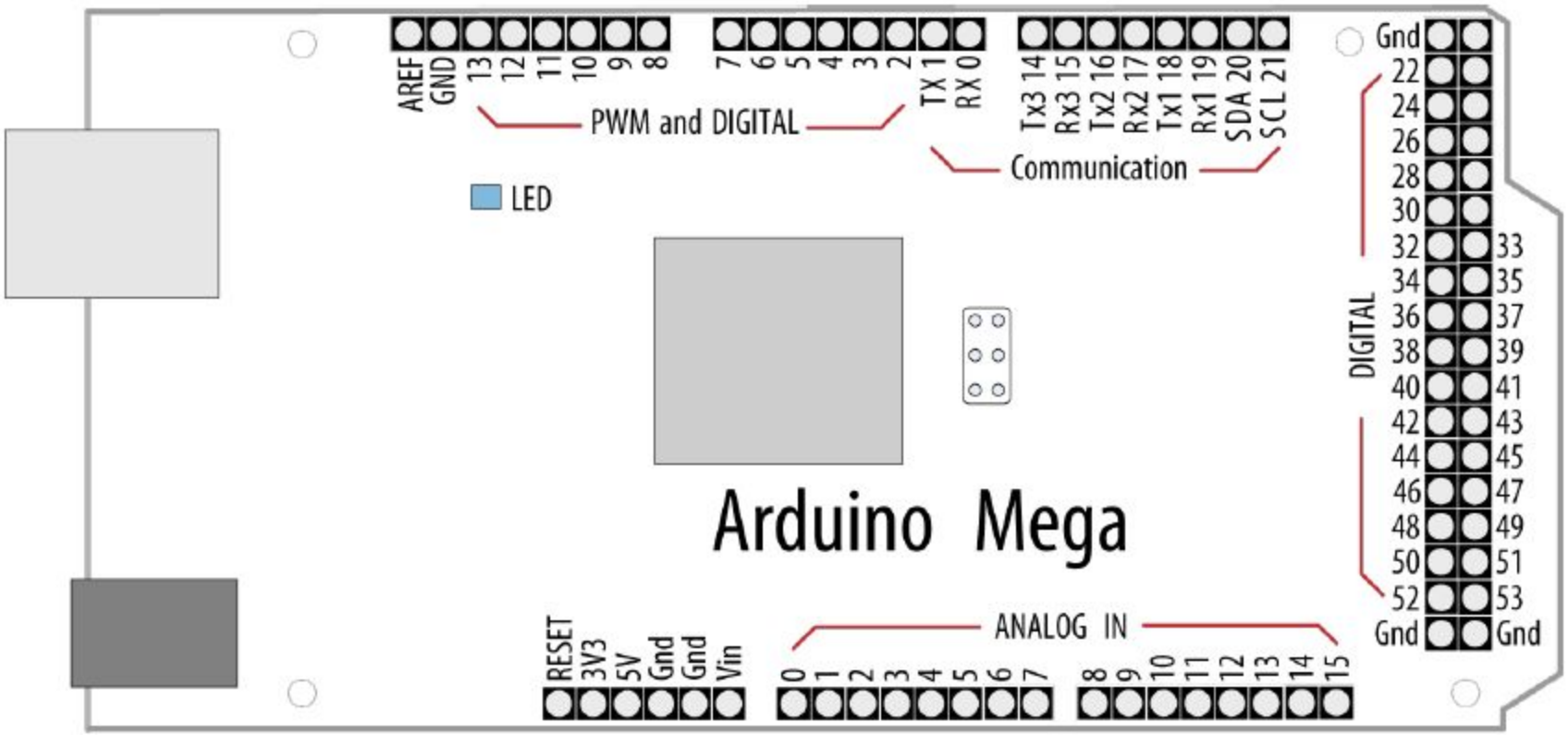


Simple Digital and Analog Inputs

- digitalRead(pin) - tells your sketch if a voltage on a pin is HIGH (5 volts) or LOW (0 volts)
- pinMode(*pin*, INPUT) – configure pin as an INPUT
- 14 digital pins (numbered 0 to 13)
- Pins 0 and 1 (marked RX and TX) are used for the USB serial connection
- Need more?
- Analog pins 0 through 5 can be used as digital pins 14 through 19

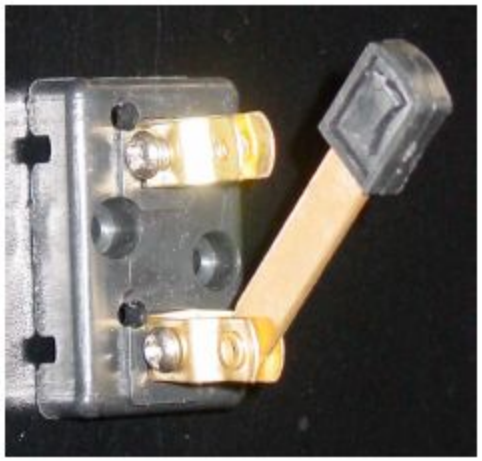
Simple Digital and Analog Inputs

- Still need more?
- Analog pins 0 through 15 are digital pin numbers 54 through 69



Sensors & Inputs

- Many sensors are variations on switches
- Switches make or break a connection



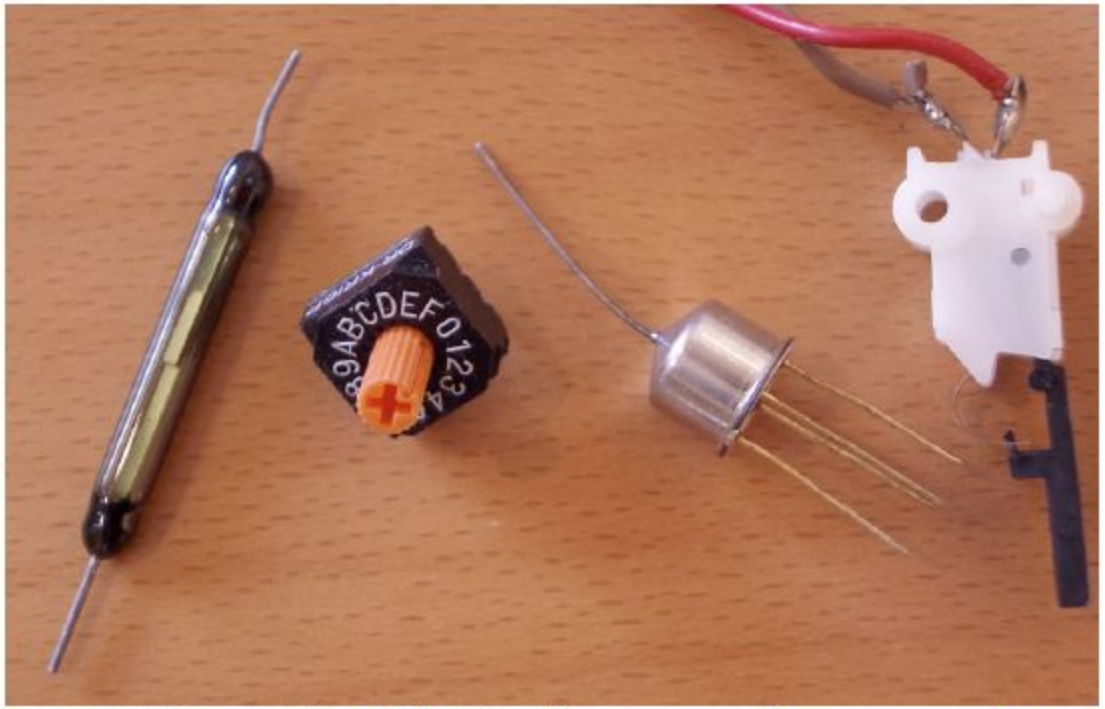
knife switch
(SPST)

toggle switch
(SPDT)

Single pole = only one circuit is being controlled
Double pole = two circuits are being controlled at once
Single throw = only one path for circuit
Double throw = two potential paths for circuit

Many Kinds of Switches

- Tilt sensor has a little ball inside
- Magnetic switches are delicate
- The hex switch is actually many switches in one, and outputs 4 signals

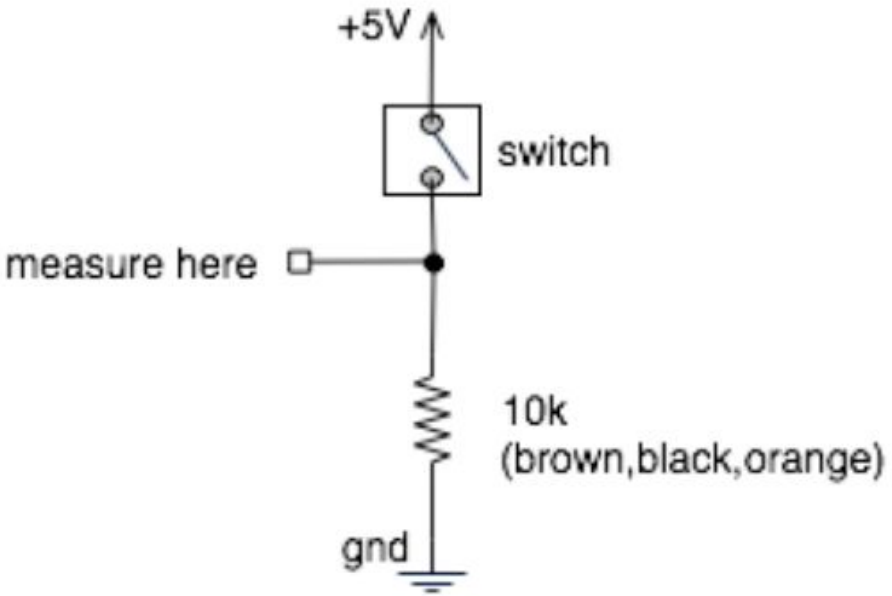


Digital Input

- Switches make or break a connection
- But Arduino wants to see a voltage
- Specifically, a “HIGH” (5 volts) or a “LOW” (0 volts)
- How do you go from make/break to HIGH/LOW?

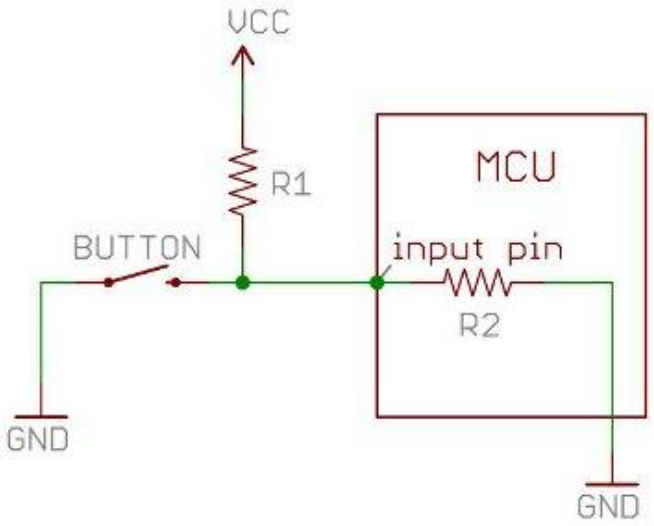
From Switch to HIGH / LOW

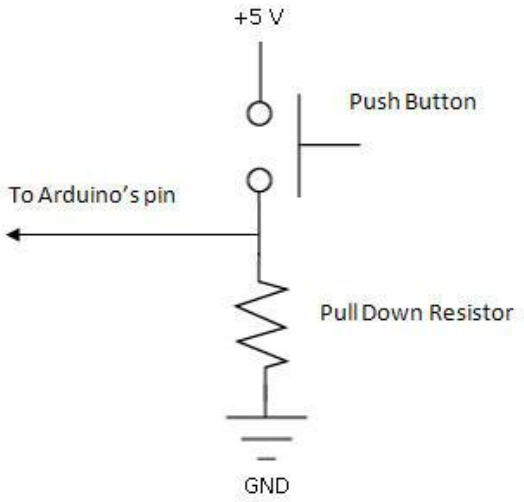
- With no connection, digital inputs “float” between 0 & 5 volts (LOW & HIGH)
- Resistor “pulls” input to ground (0 volts)
- Pressing switch “pushes” input to 5 volts
- Press is HIGH
- Not pressed is LOW
- Pull-down resistor



Pull-up and Pull-down

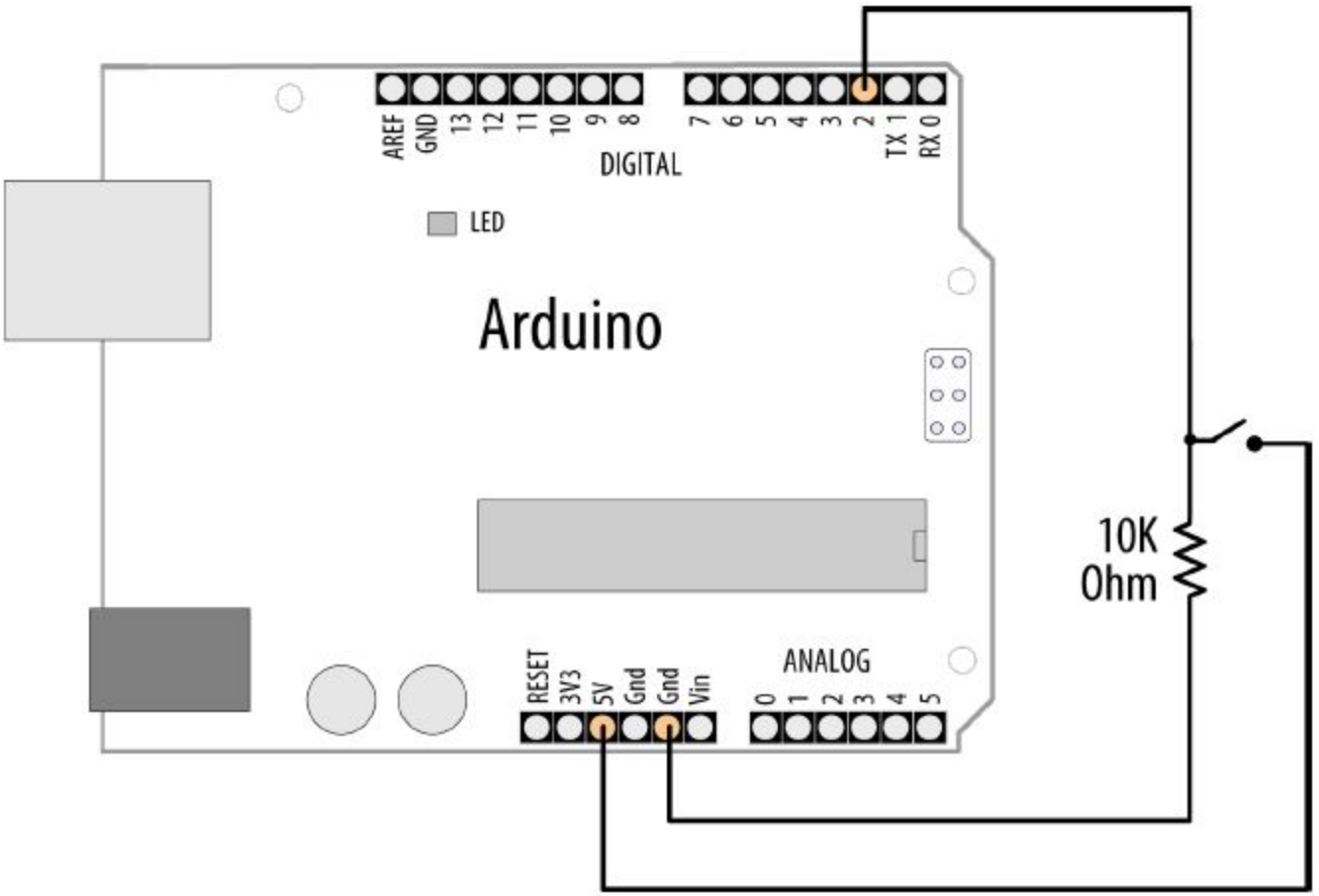
- *pull-up* resistors – pull the voltage up to the 5V line that the resistor is connected to
- *pull-down* resistors – pull the voltage down to 0 volts
- Although 10K ohms is a commonly used value, anything between 4.7K and 20K or more will work





Control the Blinking

- Connect a button to pin 2 with a pull-down resistor
- Turn on LED if button pressed and OFF if released



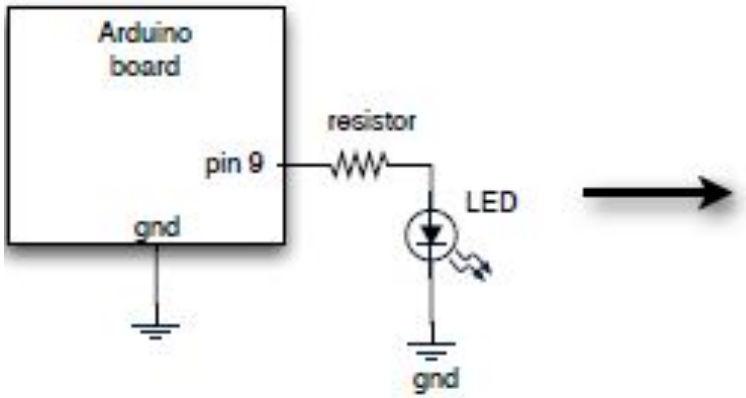
Control the Blinking

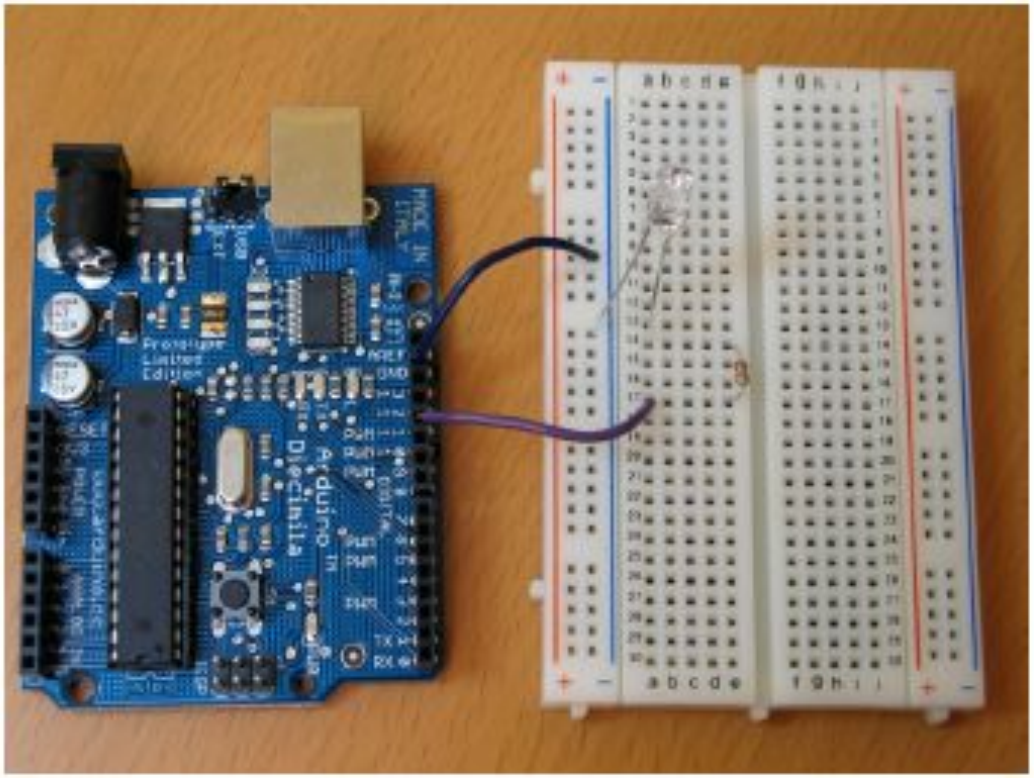
```
//  Pushbutton sketch a switch connected to pin 2 lights the LED on pin 13

const int ledPin = 13; // choose the pin for the LED
const int inputPin = 2; // choose the input pin (for a pushbutton)
void setup() {
  pinMode(ledPin, OUTPUT); // declare LED as output
  pinMode(inputPin, INPUT); // declare pushbutton as input
}
void loop(){
  int val = digitalRead(inputPin); // read input value
  If (val == HIGH) { // check if the input is HIGH
    digitalWrite(ledPin, HIGH); // turn LED on if switch is pressed
  } else {
    digitalWrite(ledPin, LOW); // turn LED off
  }
}
```

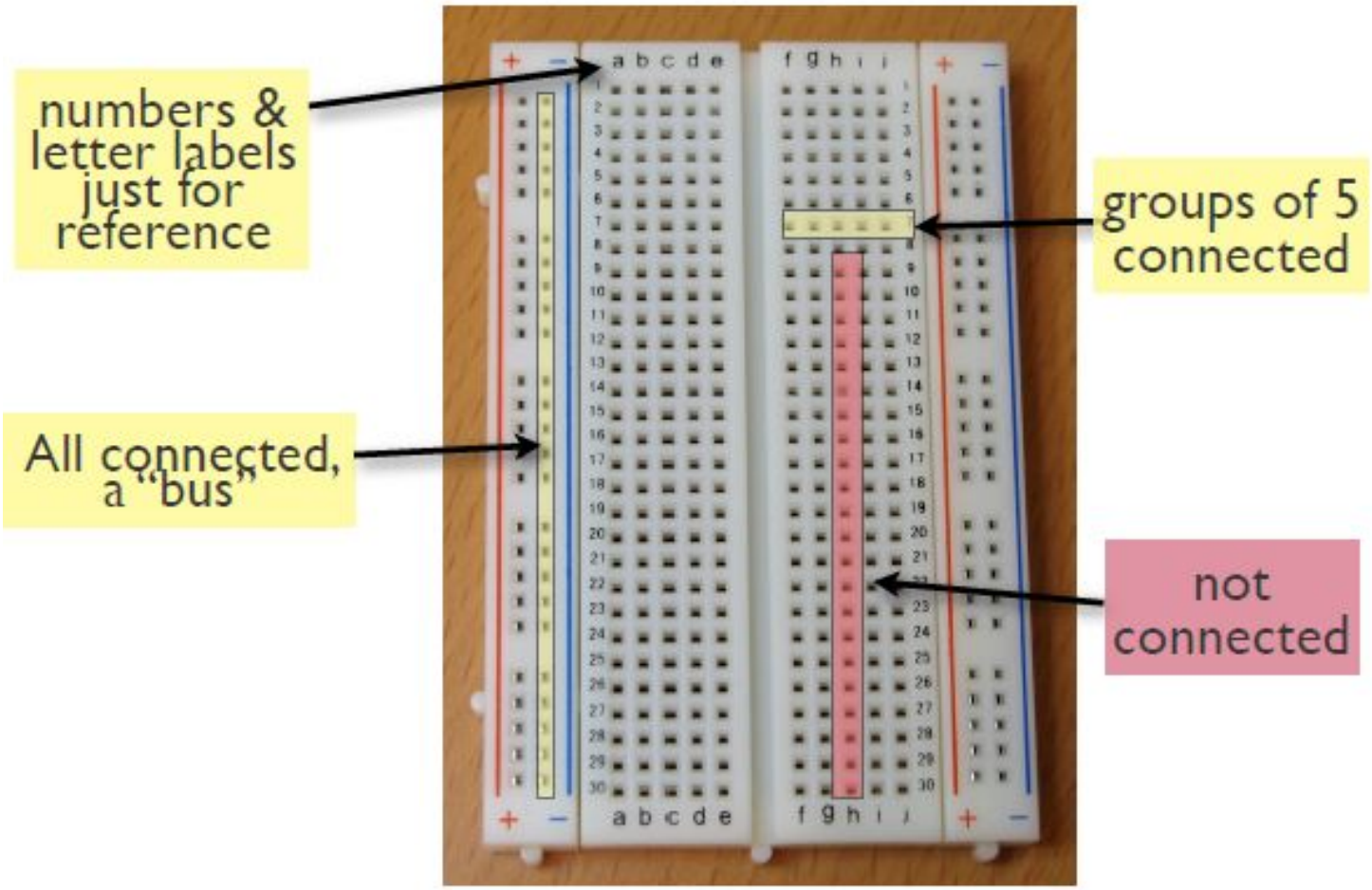
Let's Wire It Up

- Going from schematic to physical circuit.





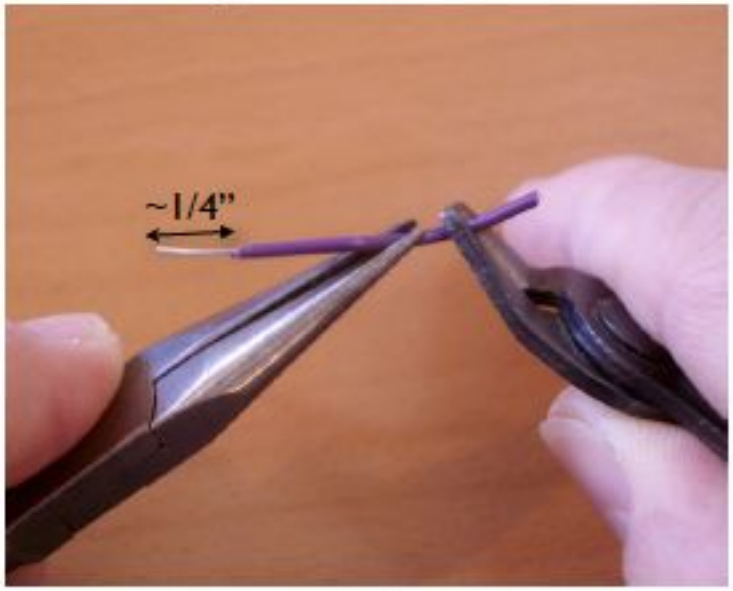
Solderless Breadboards



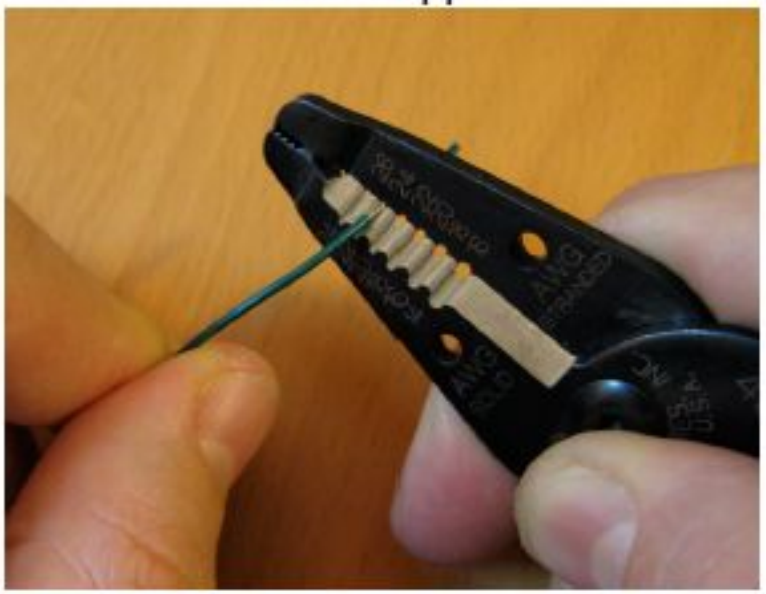
Useful Tools

Making Jumper Wires

pliers & cutter

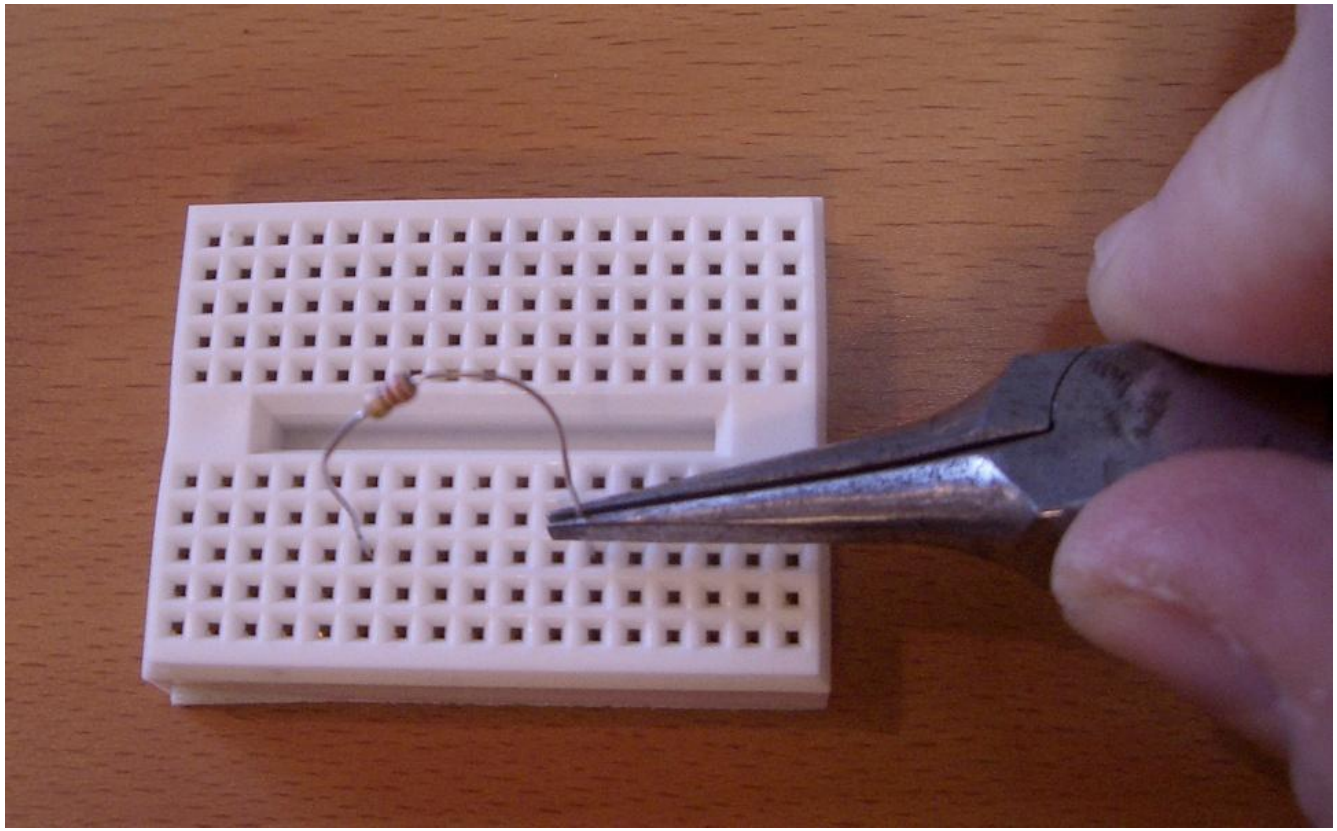


wire stripper

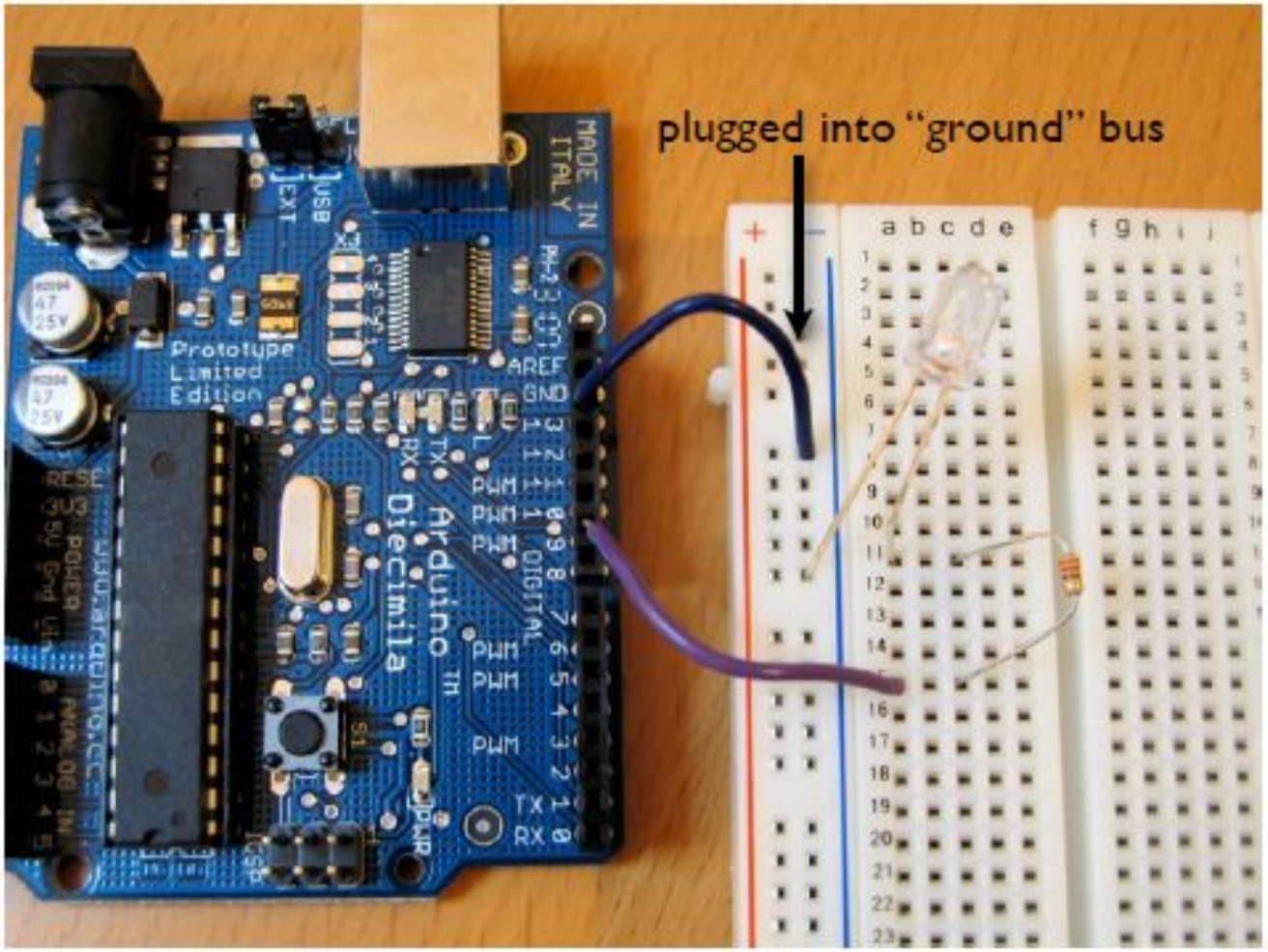


Using Solderless Breadboards

- Using needle nose pliers can help push wires & components into holes



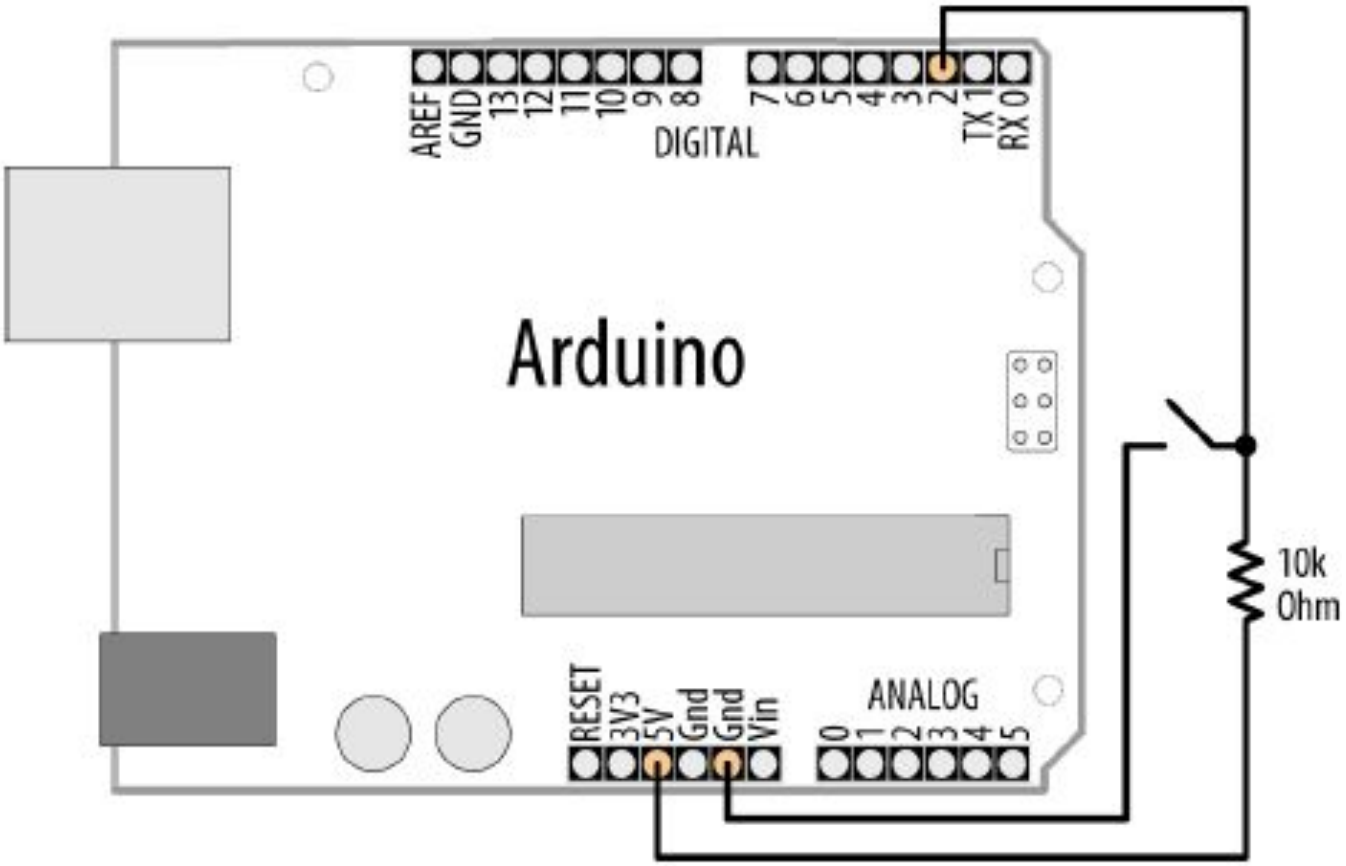
All Wired Up


plugged into "ground" bus
Prototype
Limited
Edition
Arduino
Diecimila
MADE IN
ITALY
a b c d e
f g h i j

Using Switches to Make Decisions

- Often you'll want to choose between actions, based on a data obtained from switch-like sensor
  - E.g. "If motion is detected, turn on the lights"
  - E.g. "If flower pot soil is dry, turn on sprinklers"
- Define actions, choose them from sensor inputs
- Let's try that with the actions we currently know
  - E.g.: If button is pressed send "Hello!" to serial port, and if released send "Goodbye!"

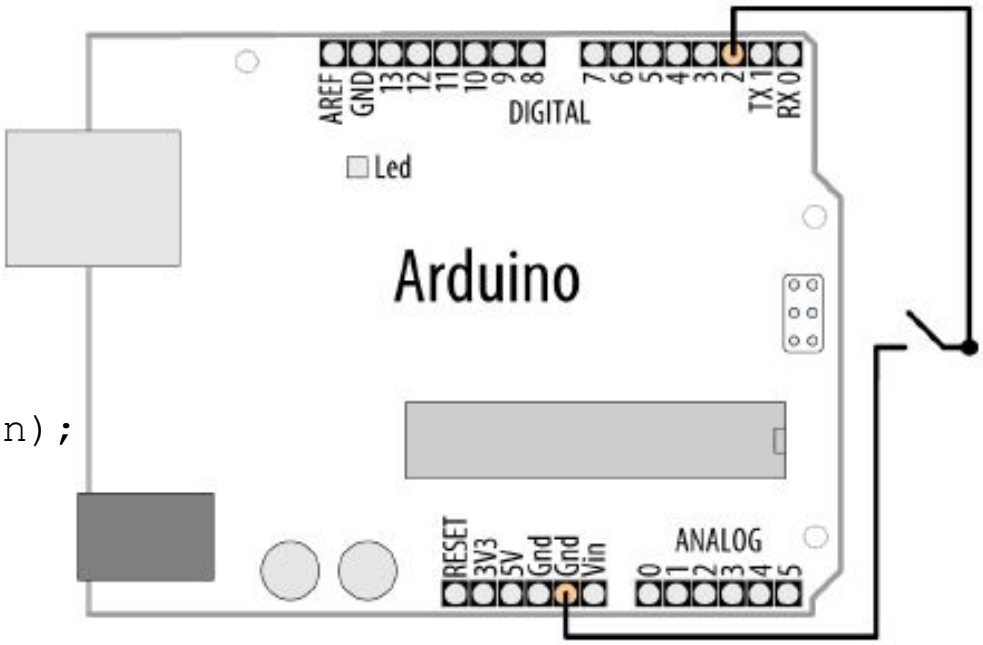
Control the Blinking (pull-up)



Switch Without External Resistors

- Arduino has internal pull-up resistors that can be enabled by writing a HIGH value to a pin that is in INPUT mode

```
const int ledPin = 13;
const int inputPin = 2;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(inputPin, INPUT);
  digitalWrite(inputPin,HIGH);
  // turn on internal pull-up
}
void loop(){
  int val = digitalRead(inputPin);
  if (val == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```



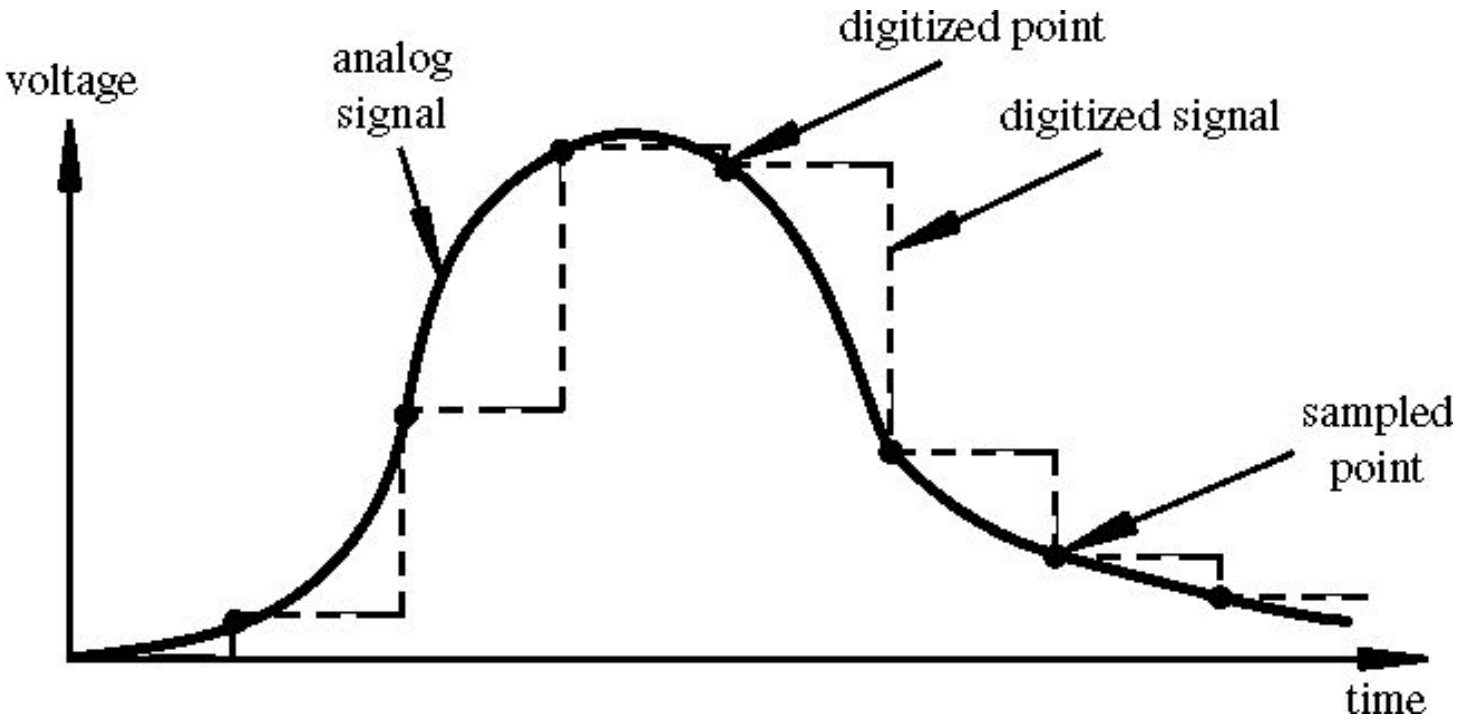
Reliably Detecting the Switch State

- contact bounce produces spurious signals at the moment the switch contacts close or open
- avoid false readings due to contact bounce - debouncing

```
boolean debounce(int pin) {
  boolean state;
  boolean previousState;
  previousState = digitalRead(pin); // store switch state
  for(int cnt=0; cnt < debounceDelay; cnt++) {
    delay(1); // wait for 1 millisecond
    state = digitalRead(pin); // read the pin
    if( state != previousState) {
      cnt = 0; // reset the counter if the state changes
      previousState = state; // and save the current state
    }
  }
  return state;
}
```

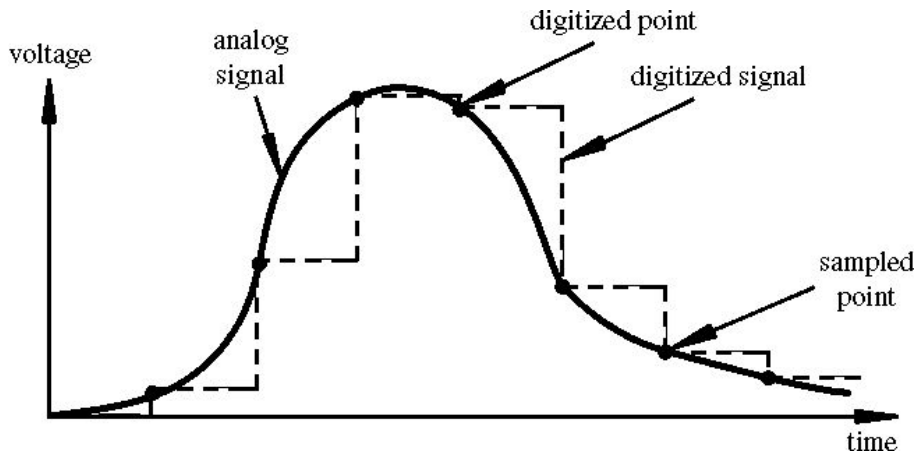
Analog Input

- To computers, analog is chunky



Analog Input

- Many states, not just two (HIGH/LOW)
- Number of states (or values, or "bins") is *resolution*
- Common computer resolutions:
  - 8-bit = 256 values
  - 16-bit = 65,536 values
  - 32-bit = 4,294,967,296 values

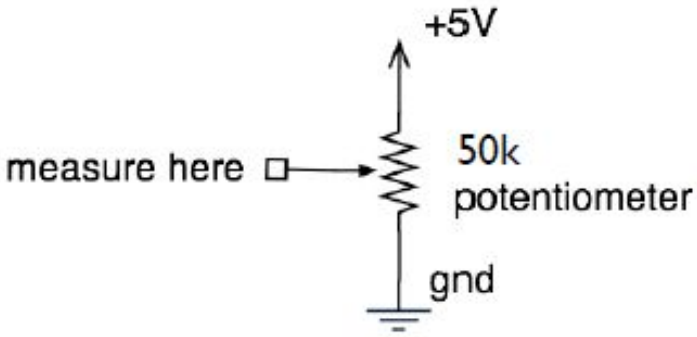


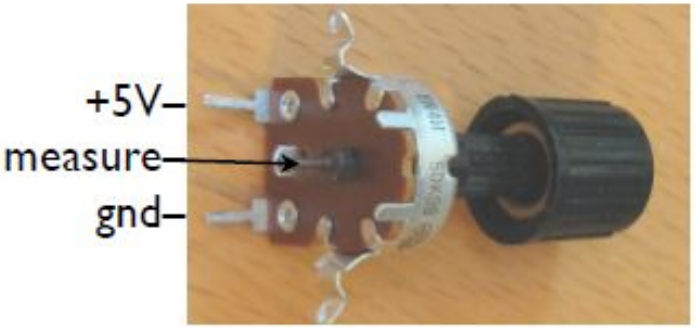
Analog Input

- Arduino (ATmega168) has six ADC inputs
- (ADC = Analog to Digital Converter)
- Reads voltage between 0 to 5 volts
- Resolution is 10-bit (1024 values)
- In other words, 5/1024 = 4.8 mV smallest voltage change you can measure

Analog Input

- *Sure sure, but how to make a varying voltage?*
- *With a potentiometer. (pot)*



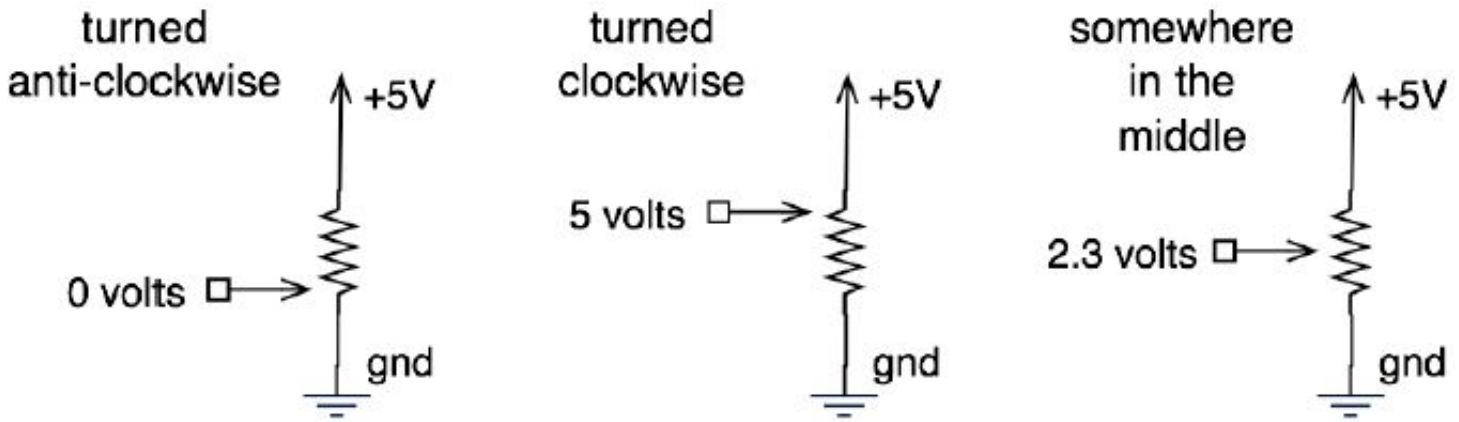


The pot you have

pots also look like this

Potentiometers

- Moving the knob is like moving where the arrow taps the voltage on the resistor
- When a resistor goes across a voltage difference, like +5V to Gnd, the voltage measured at any point along a resistor's length is proportional to the distance from one side.



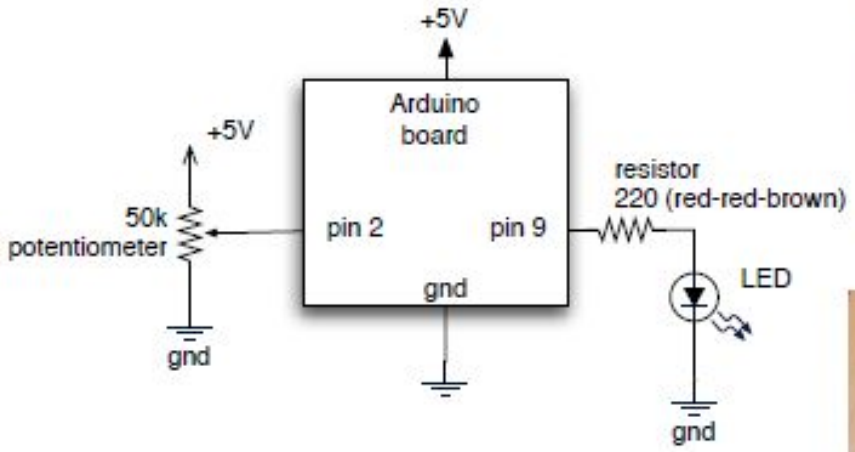
What good are pots at?

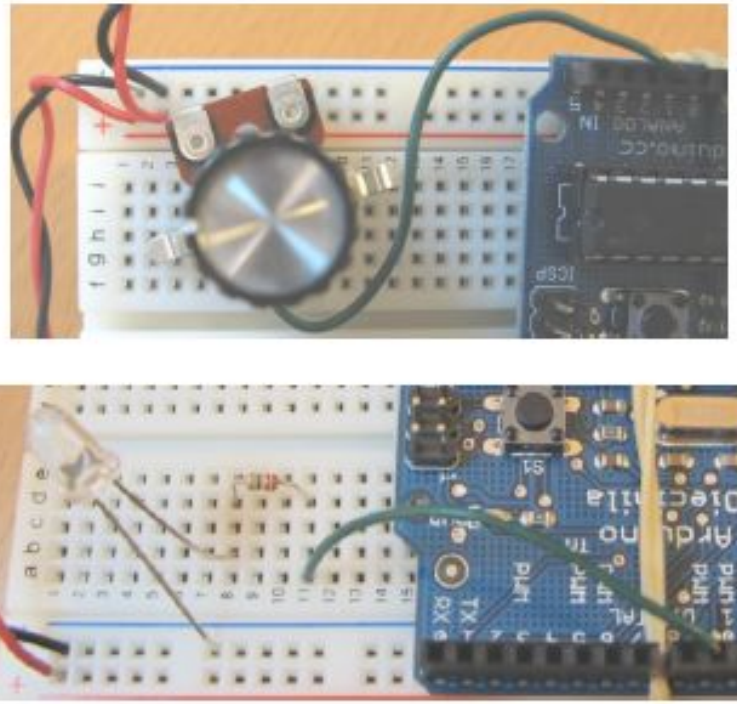
- Anytime you need a ranged input
- Measure rotational position
- steering wheel, robotic joint, etc.
- But more importantly for us, potentiometers are a good example of a resistive sensor

Arduino Analog Input

- Plug pot directly into breadboard
- Two “legs” plug into +5V & Gnd (red + & blue -) buses
- Middle “post” plugs into a row (row 7 here)
- Run a wire from that row to Analog In 2

Pot & LED Circuit





Pot Blink Rate

```
/*
  Pot sketch
  blink an LED at a rate set by the position of a potentiometer
*/
const int potPin = 0; // select the input pin for the
potentiometer
const int ledPin = 13; // select the pin for the LED
int val = 0; // variable to store the value coming from the sensor
void setup(){
  pinMode(ledPin, OUTPUT); // declare the ledPin as an OUTPUT
}
void loop() {
  val = analogRead(potPin); // read the voltage on the pot
  digitalWrite(ledPin, HIGH); // turn the ledPin on
  delay(val); // blink rate set by pot value (in milliseconds)
  digitalWrite(ledPin, LOW); // turn the ledPin off
  delay(val); // turn led off for same period as it was turned on
}
```