Lecture 5

Computers & Using Your Arduino

Roadmap

We have almost everything we need to make our useless box more flexible. But we need two learn a few more things before we can start. The first is how a computer internally represents numbers, images, etc, and then how to physically connect external switches and motors (the stuff of the useless box) to our Arduino. Finally we will discuss how do we program the Arduino to do what we want. We will cover these issues in this lecture.

Learning Objectives

- MOS transistors have resistance when they are on
  - The output of your Arduino driven by MOS transistors
  - Can't drive the motor directly
- How to write a simple Arduino program
  - Three parts: declarations; setup; loop
- Understand how to connect a switch to a chip
  - Chips like their inputs driven to Vdd or Gnd (not floating)

PREVIOUSLY IN E40M

Simple Model of an nMOS Device

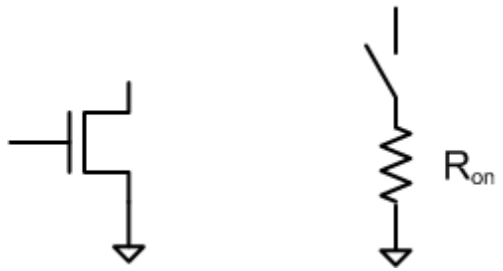
- We will model an nMOS device by components we know
  - Resistors
  - Switches

- NMOS

  Source = Gnd

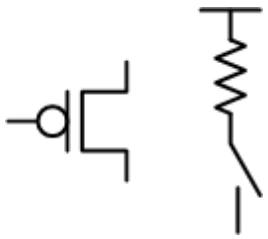
  Gate = Gnd => Off

  Gate = Vdd => On



pMOS Device Work Well With Vdd

- PMOS
  - Source = Vdd (+ supply)
  - Gate = Gnd => On
  - Gate = Vdd => Off

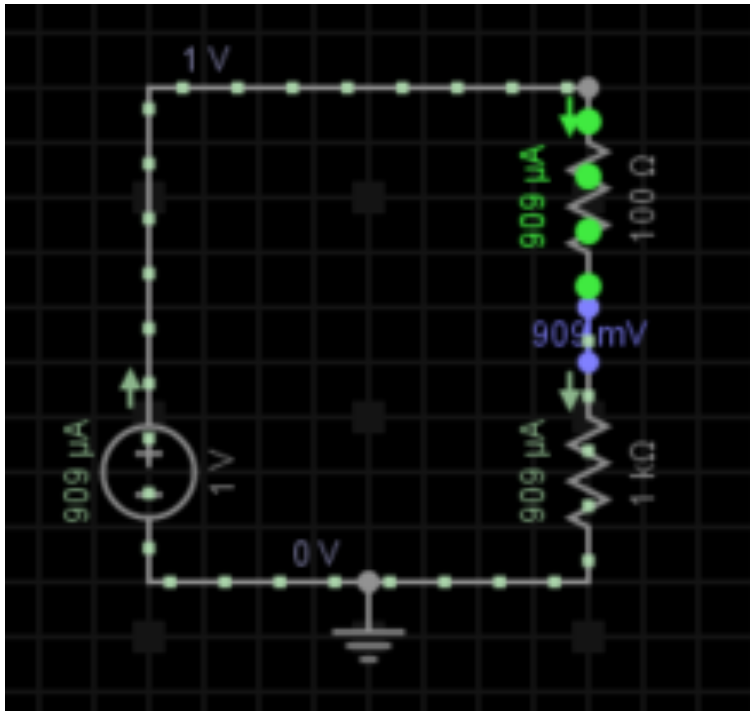


Building a CMOS NOR Gate

- Output should be low if either input is high (true)
- Output should be high if both inputs are low (false)

Every Circuit

- Simple simulator that we will use for circuits



- http://everycircuit.com/app/

HOW A COMPUTER WORKS

Gates Deal With Binary Signals

- Wires can have only two values
  - Binary values, 0, 1; or True and False
- Generally transmitted as:
  - Vdd = 1; Gnd = 0
    - Vdd is the power supply, about 1V
    - Gnd is the “reference” level, about 0V
  - Some signals can be negative true
    - 1 = 0V, these are generally indicated by a bar (or _b)
- So how do we represent:
  - Numbers, letters, colors, etc. ?

Binary Numbers

- To represent anything other than true and false
  - You are going to need more then one bit
- Group bits together to form more complex objects
  - 8 bits are a byte, and can represent a character (ASCII)
- 24 bits are grouped together to represent color
  - 8 bits for R, G, B.
- 32 or 64 bits are grouped together and can represent an integer

Place Values For Binary Numbers

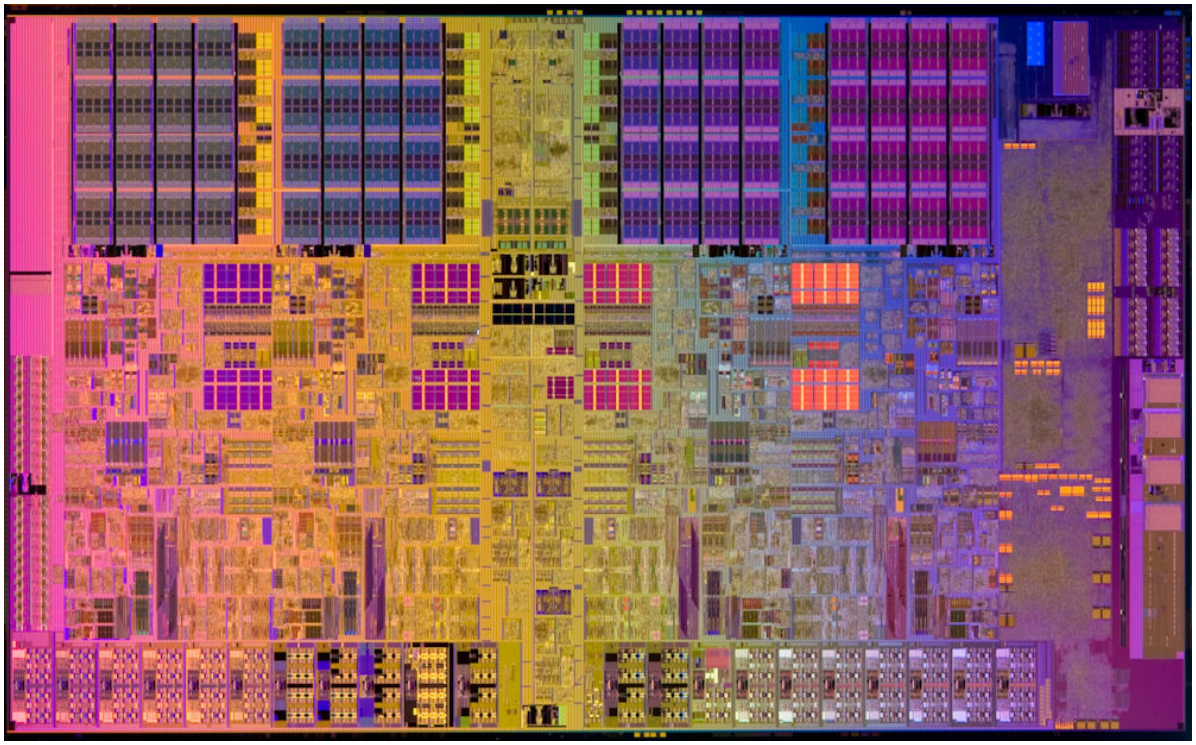
- For decimal numbers
  - Each place is 10^n
    - 1, 10, 100, 1000 …
    - Since there are 10 possible values of digits
- For binary numbers
  - Each digit is only 0, 1 (only two possible digits)
- The place values are then 2^n
  - 1, 2, 4, 8, 16, …
- It is useful to remember that $2^{10} = 1024$

Can Represent Operations in Boolean Logic Like Add

- If I have two binary numbers
  - A and B, both 32 bits
- And I want to get the sum (A + B)
  - I can represent that as a logical equation
- For each bit position compute
  - $Sum_i = A_i \text{ ^ } B_i \text{ ^ } Cout_{i-1}$
  - $Cout_i = A_i \text{ \& } B_i \text{ | } A_i \text{ \& } Cout_{i-1} \text{ | } B_i \text{ \& } Cout_{i-1}$

If You Look At Your Computer Chip

- It is just billions of transistors
  - Creating many logic gates, and memory



- Take EE108A if you want know how they do that..

ARDUINO MICROCONTROLLERS

Micro-controllers

- These are simpler processors built to control stuff
  - There are tons of different types of these computers
- Typically contain more than just a processor
  - Contain RAM, and FLASH on the die
  - So you have your disk, and memory too
  - All on chip, so the pins are free to control stuff
- But the amount of storage can be small
  - 32KB of “disk”
  - 2KB of RAM
- Common architectures: PIC, TI 430, Atmel AVR

Arduino

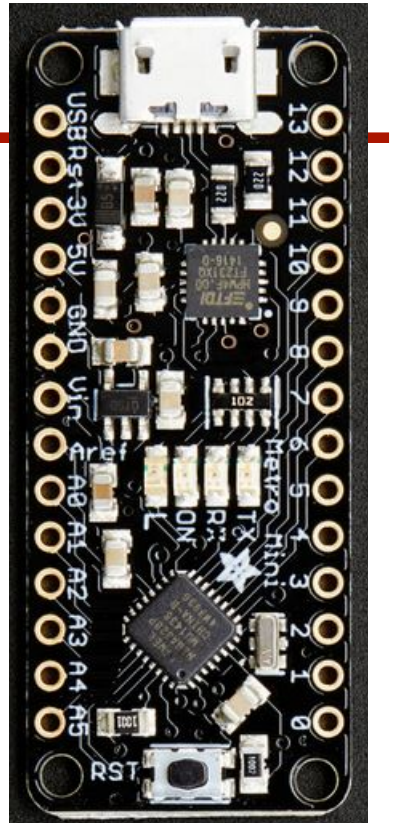
- All these machines had different compilers/software tools

- Arduino is a software development system
  - It is pretty basic, but it is open source
  - Has a very simple way of controlling I/O
  - And lots of people have been using it.
    - So there are many code examples to look at / leverage

- Was initially developed with the Atmel AVR architecture
  - But now it has been ported to others (ARM, Intel)

- It is what we are going to use in this class.

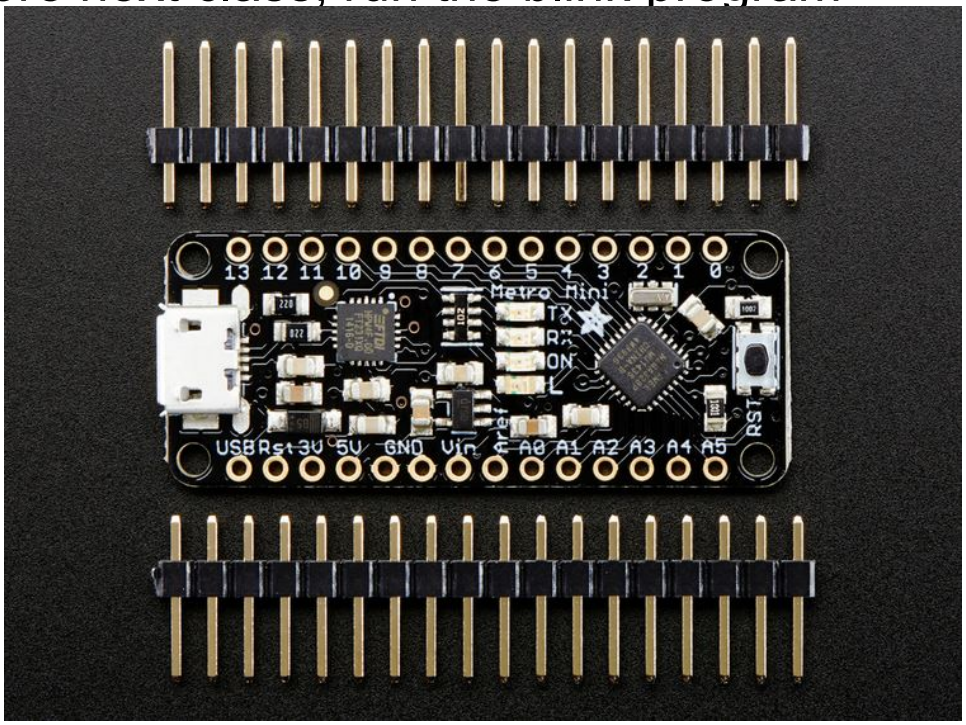
Arduino Metro Mini



- We are going to use the Arduino Metro Mini
  - It is based on the ATmega328
- It has 20 digital input/output pins
  - Some can be used for analog input too.
- Runs on a 16MHz crystal
- Programs through a USB connector to your computer
  - Powered by the USB
  - Fits nicely on your breadboard.
    - But you will need to solder the pins on the board!

Download the Software

- http://arduino.cc/en/Main/Software
  - Set the board type to “Uno”
  - Before next class, run the blink program

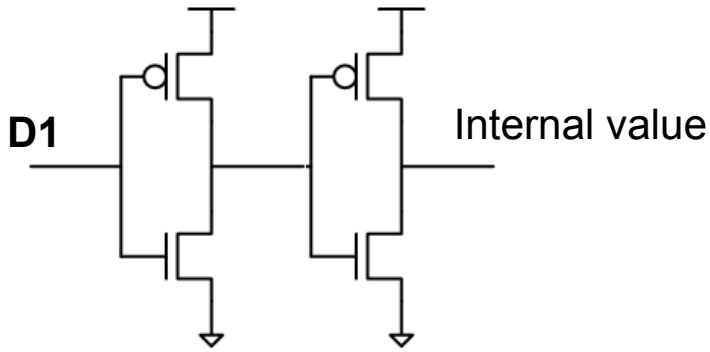


MAKING BREAK

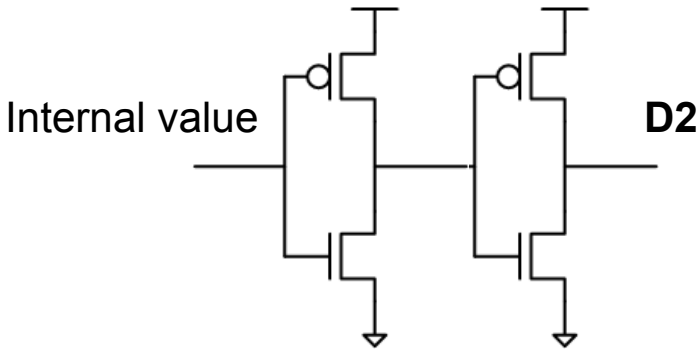
Getting Signals In/Out of the Arduino

- The Metro uses a CMOS chip
  - Which means it is made from nMOS and pMOS transistors
- Each pin can be an input pin, or an output pin

Input Pin



Output Pin



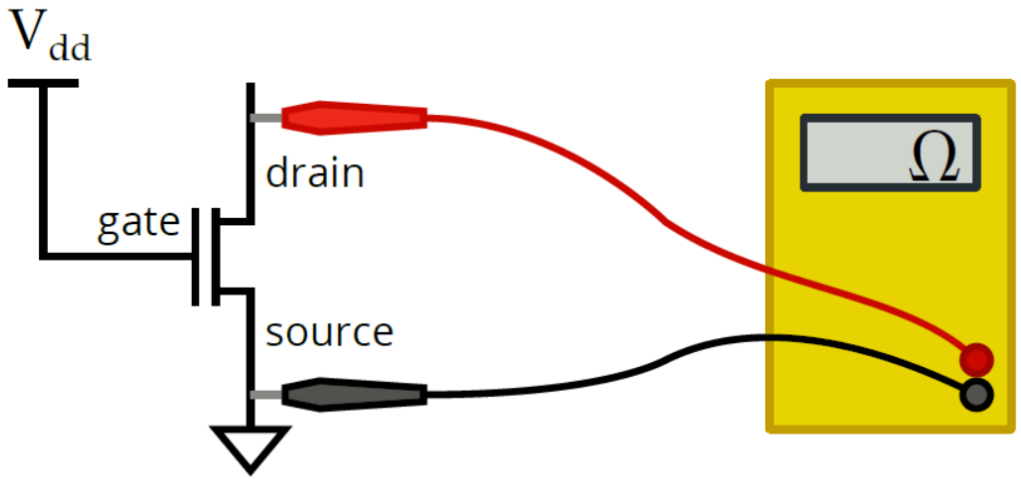
Careful With Arduino Outputs

- The output is just a CMOS inverter
  - And remember that MOS transistors have some resistance
- The resistance will limit the amount of current you can drive
  - One of the prelabs is to measure this resistance
- It will turn out to be too large to directly drive you motor

Measure Transistor Resistance

Turn the nMOS transistor on by making $V_{GS} > 1V$
Note that the drain doesn't need to be connected

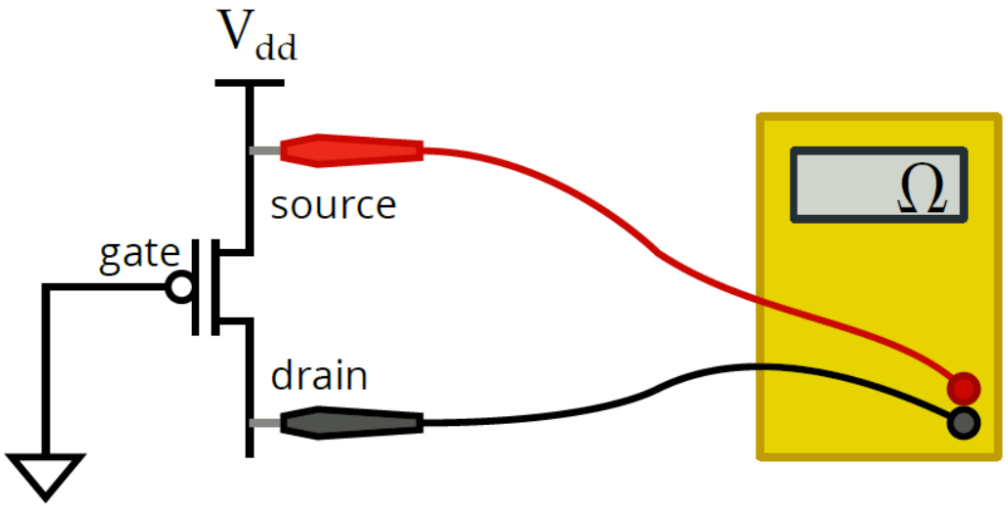
Measure the resistance between the drain and source by using your meter in resistance mode.



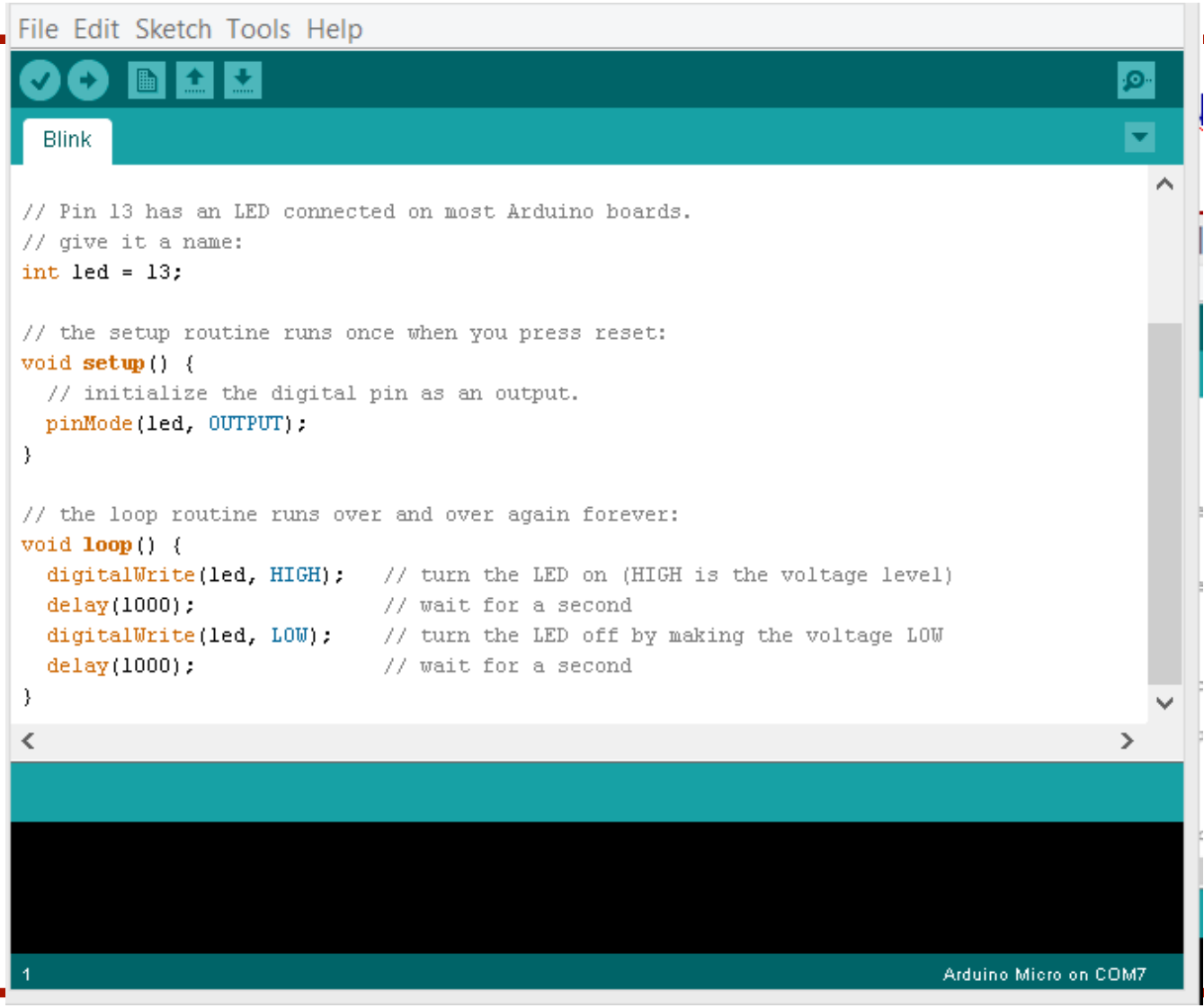
Measure pMOS Resistance

Turn the pMOS transistor on by making $V_{GS} < -1V$
Again, the drain doesn't need to be connected

Measure the resistance between the drain and source by using your meter in resistance mode.



Arduino IDE


File Edit Sketch Tools Help
Blink
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}
1
Arduino Micro on COM7

Anatomy of an Arduino program

```
// You can put global variables here

void setup() {
  // This runs once on power up or reset
}

void loop() {
  // This runs over and over again, forever
}
```

pinMode(PIN, MODE)

PIN - the digital pin number

MODE - one of:

INPUT - Pin is used to read voltages
INPUT_PULLUP - Input with internal 20k pullup
OUTPUT - Pin sets voltages

http://arduino.cc/en/Reference/PinMode

```
digitalWrite(PIN, VALUE)
```

PIN - the digital pin number

VALUE - HIGH or LOW (1 or 0)

http://arduino.cc/en/Reference/DigitalWrite

`Serial.begin(BAUD_RATE)`

Baud rate must match when you use the serial console. 115200 is typical.

`Serial.print(STUFF)`

Can print strings, characters, and numbers.

`Serial.println(STUFF)`

Same as print(), but appends a newline.

http://arduino.cc/en/Serial

Print example

```
// Start counting up from 0, printing the
// next number every second.
void setup() {
  Serial.begin(115200); // Initialize the port
}

void loop() {
  static int count = 0;
  Serial.print("The count is: ")
  Serial.println(count);
  delay(1000);
  count = count + 1;
}
```

```
int digitalRead(PIN)
```

PIN - the digital pin number

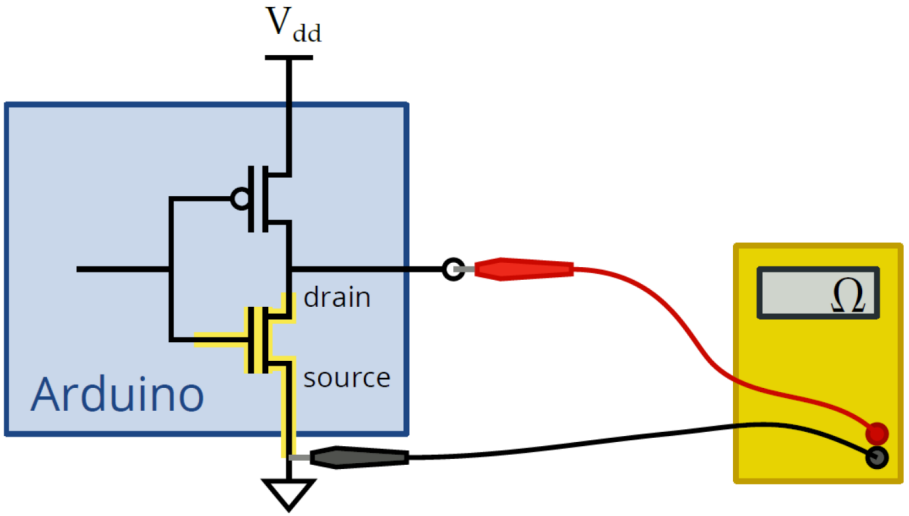
Returns HIGH or LOW (1 or 0)

http://arduino.cc/en/Reference/DigitalRead

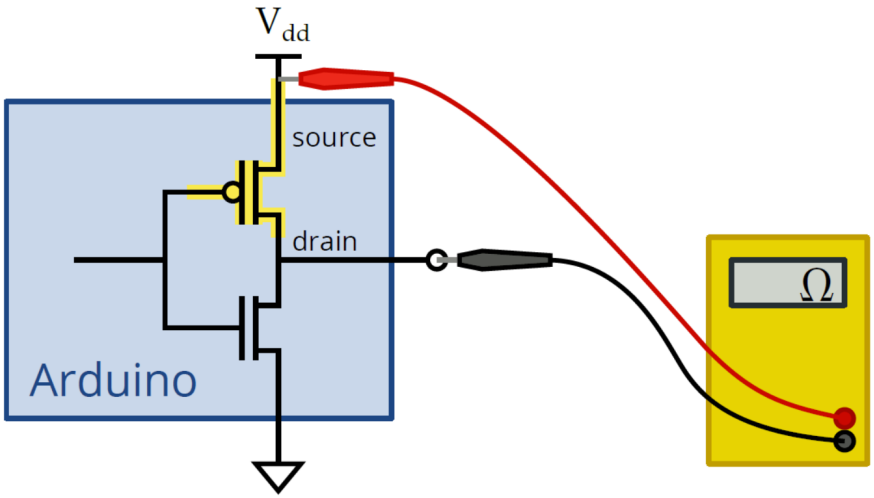
Measuring Arduino Output Resistance

In the Arduino, we don't have direct access to the gate, but we can turn the nMOS on by setting the output pin low in software: `digitalWrite(2, LOW)`

Then we measure resistance like before:



To measure the pMOS inside the Arduino pins, we have to turn it on by setting the output high:
`digitalWrite(2, HIGH)`

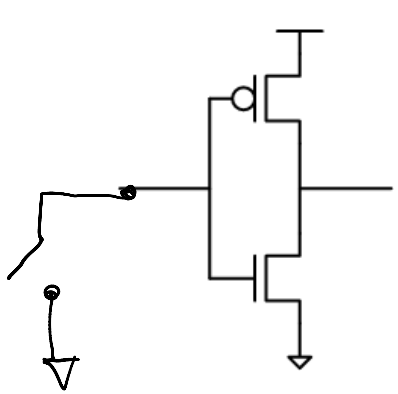


USING YOUR ARDUINO

Connecting Switches To Computers

- For an input, you are connecting to the gate of two transistors
  - But the gate terminal doesn't take any current
    - It is a capacitor, a device we will explain a little later in the class
  - It just measures the voltage that is presented to it

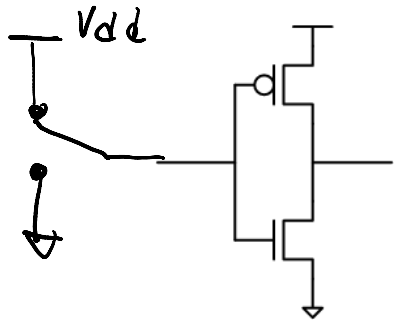
- So this doesn't work



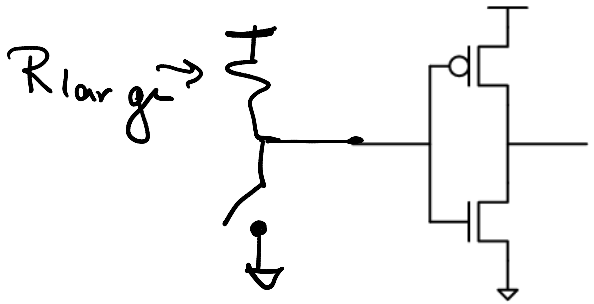
Why?

Two Ways to Fix this Problem

- Use a SPDT switch instead
  - Connect the input either to Vdd or to Gnd



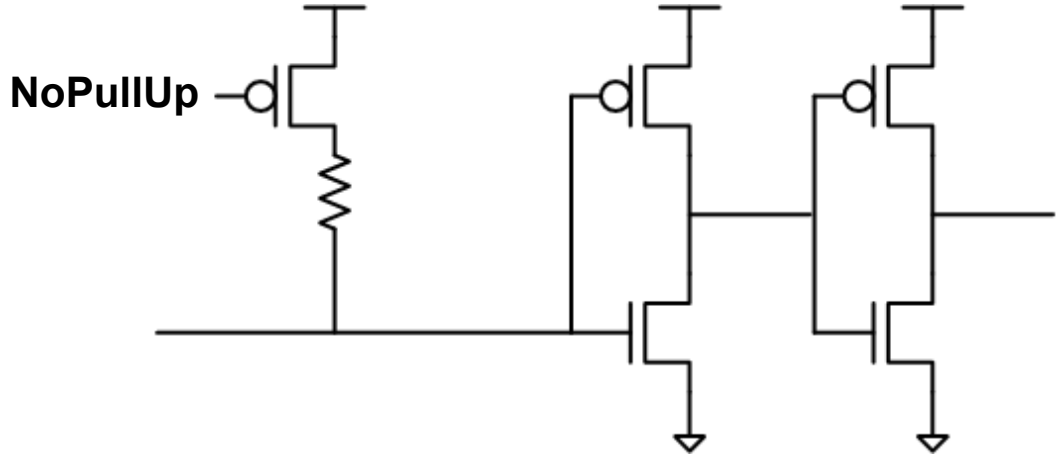
- Use a resistor pullup
  - Have a resistor that always lightly pulls the output high



Careful With Pullup Inputs

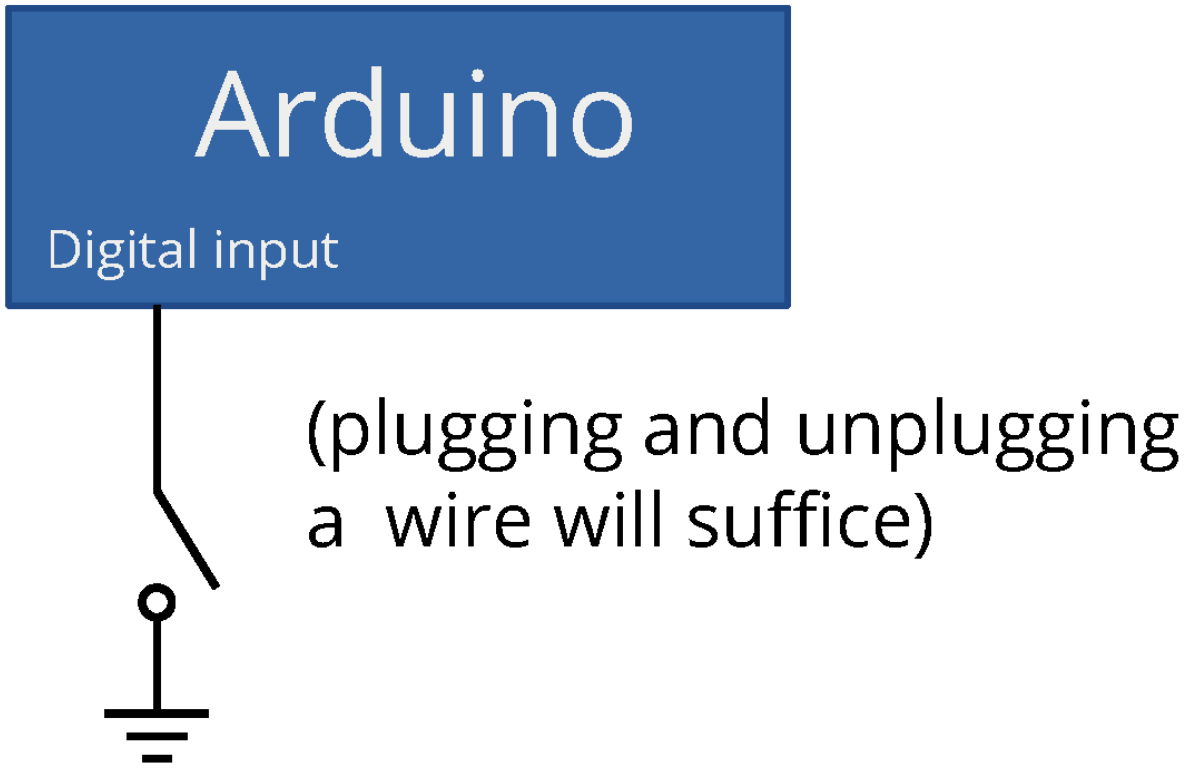
- Remember when the switch is **not** connected
  - The input is 1
- When the switch is connected
  - The input is connected to Gnd, and the input is 0

What The Arduino Input Really Looks Like



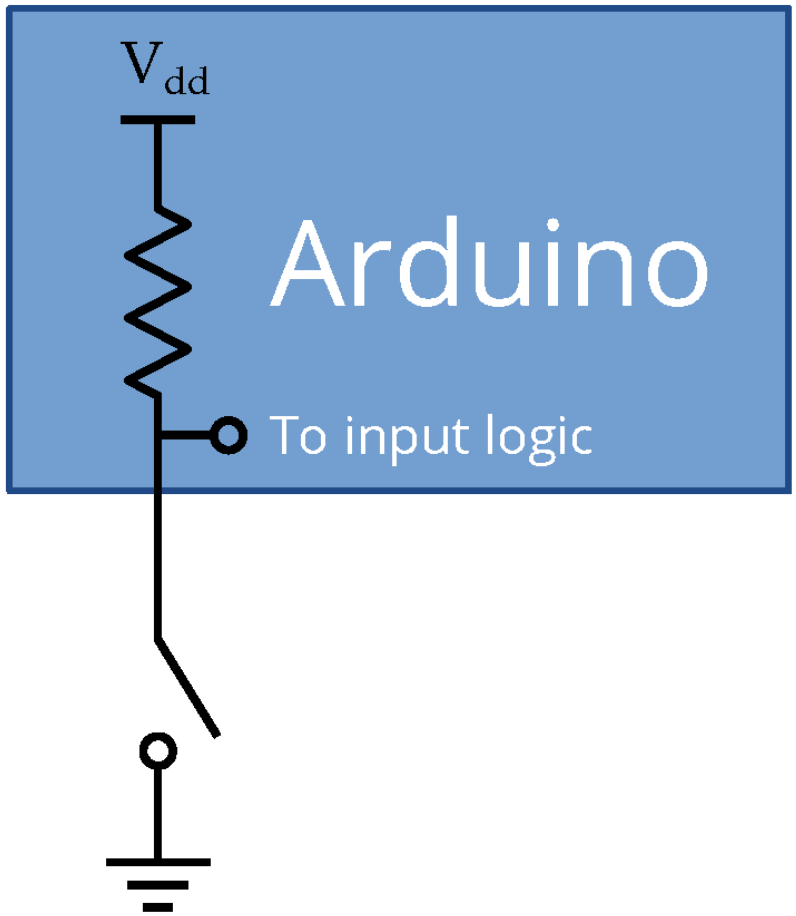
- It already contains a pullup resistor
  - You need to specify that you want to use it
    - pinMode(d2, INPUT_PULLUP)

Build this:



(plugging and unplugging a wire will suffice)

Behind INPUT_PULLUP



Read a switch

```
const int SWITCH = 4;
void setup() {
  pinMode(SWITCH, INPUT_PULLUP);
  Serial.begin(115200);
}

void loop() {
  Serial.println(digitalRead(SWITCH));
  delay(500);
}
```

We're using input_pullup because our switch only connects the pin to ground.

Read and write

```
const int LED = 2;
const int SWITCH = 4;
void setup() {
  pinMode(LED, OUTPUT);
  pinMode(SWITCH, INPUT_PULLUP);
}

void loop() {
  if(digitalRead(SWITCH) == HIGH){
    digitalWrite(LED, HIGH);
  }else{
    digitalWrite(LED, LOW);
  }
}
```

A word on data types

The Arduino Nano is a slow computer: 16 MHz

It only does math with 8-bit integers

And it doesn't have much RAM: 2kB

So code efficency is often important:

- Use `char` (8 bits) or `int` (16 bits) instead of `long`
- Avoid floating-point if at all possible
- And be aware that simple bits of code take time to run

Controlling brightness

```
const static int LED = 3;
void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, HIGH);
  delay(5); // Time LED is on
  digitalWrite(LED, LOW);
  delay(5); // Time LED is off
}
```

We can vary the on/off ratio to control the brightness.

```
analogWrite(PIN, VALUE)
```

PIN - the digital pin number

VALUE - 0 to 255

Be warned that this only works on some pins.
Read the Arduino documentation for details.

http://arduino.cc/en/Reference/AnalogWrite

Learning Objectives

- Understand how binary numbers work
  - And be able to convert from decimal to binary numbers
- MOS transistors have resistance when they are on
  - The output of your Arduino driven by MOS transistors
  - Can't drive the motor directly
- How to write a simple Arduino program
  - Three parts: declarations; setup; loop
- Understand how to connect a switch to a chip
  - Chips like their inputs driven to Vdd or Gnd (not floating)