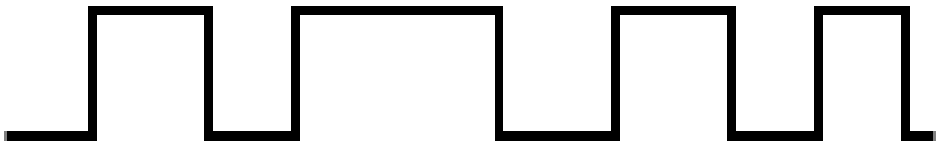
Basic Digital I/O

Digital vs Analog Signal

- *I/O* stands for “input & output,” This is how we will communicate information to and from the microcontroller
- Digital means that it comes in a discrete set of possible values
  - For the Arduino (and most microcontrollers), these are 0 and 1
- This is opposed to an analog signal, which can take any possible value with in some range
- IF you look at a signal as it varies with time
  - Analog will be a smooth continuous wave
  - Digital will be a series of square pulses with sharp, sudden changes

Analog Signal



Digital Signal

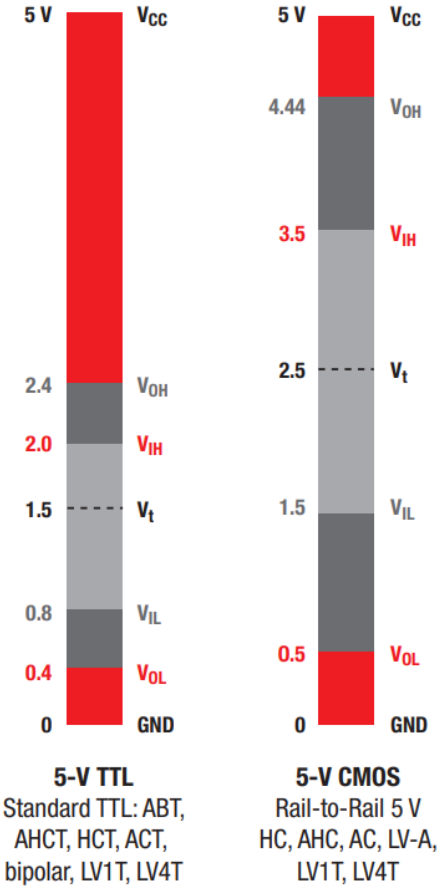
Characteristics of a Digital Signal

To use digital signals as input or output we need to specify some details

- How is the signal being transmitted?
  - What is a physical connection?
  - Is it a voltage, current, infrared light?
- What defines 1 and what is a 0?
  - These are logic levels, for the Arduino these are based on voltage
- What do the 1s and 0s mean
  - For anything other than a single, simple On/Off you need to define some kind of communication protocol

Logic Levels Defining 0 and 1

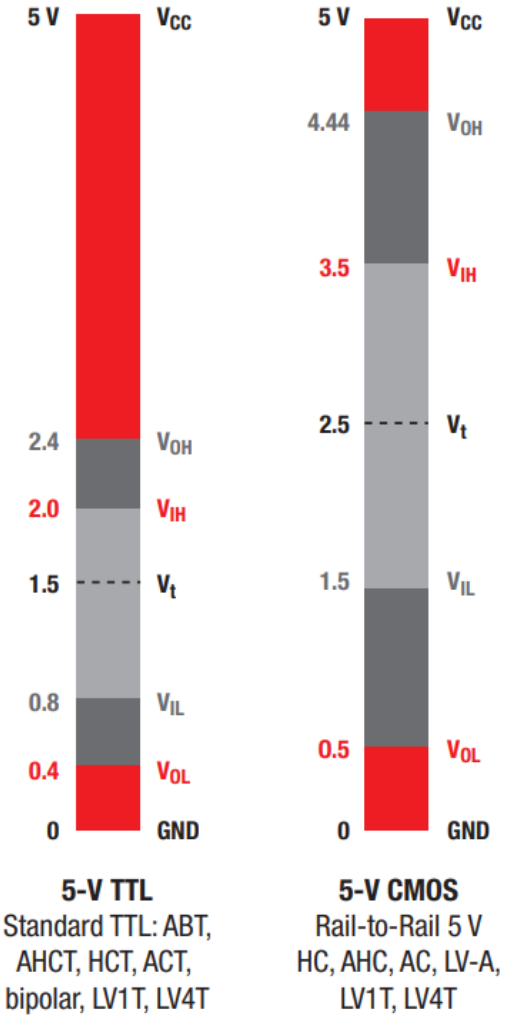
- The manufacturer of a digital device will give a set of specifications that define the voltages for High(0) and Low (0)
- These are called the “Logic Levels”
  - When discussing, you will often see terms like 5V TTL or 3.3V CMOS
  - This is the underlying technology (ex., TTL - Transistor to Transistor Logic) and nominal voltage level
- Devices will further group devices into “Logic Families” that share logic levels and additional specs
  - For example, HCT uses the TTL levels but has additional high-speed capabilities



5V TTL and CMOS Logic levels for TI Devices
https://www.ti.com/lit/sg/sdyu001ab/sdyu001ab.pdf?ts=1758714419432

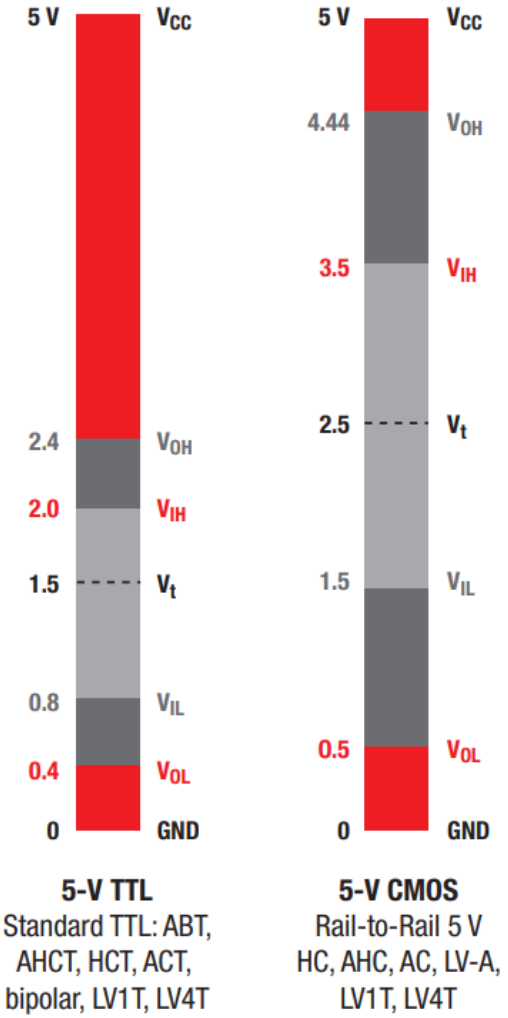
Logic Levels Output

- V_{OH} (**V**oltage **O**utput **H**igh) - Minimum voltage that will be seen when the output is high
  - So a digital 1 output will be between V_{OH} and V_{cc}
- V_{OL} (**V**oltage **O**utput **L**ow) - Maximum voltage that will be seen when the output is low
  - So a digital 0 output will be between 0 and V_{OL}



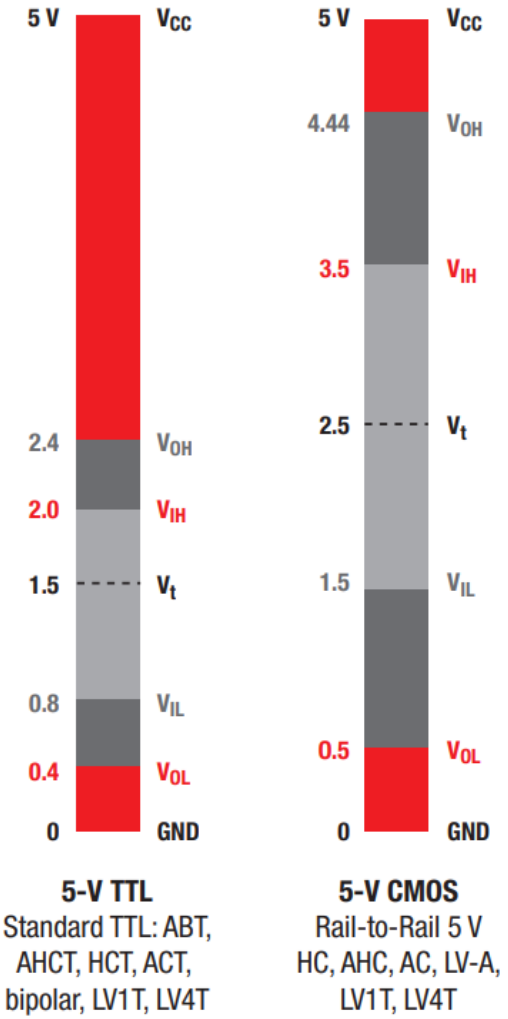
Logic Levels Input

- V_{IH} (**V**oltage **I**nput **H**igh) - Minimum voltage that will be considered high
  - Any voltage between V_{IH} and V_{cc} will be read as a 1
- V_{IL} (**V**oltage **I**nput **L**ow) - Maximum voltage that will be considered a low
  - Any voltage between 0 and V_{IL} will be read as a 0



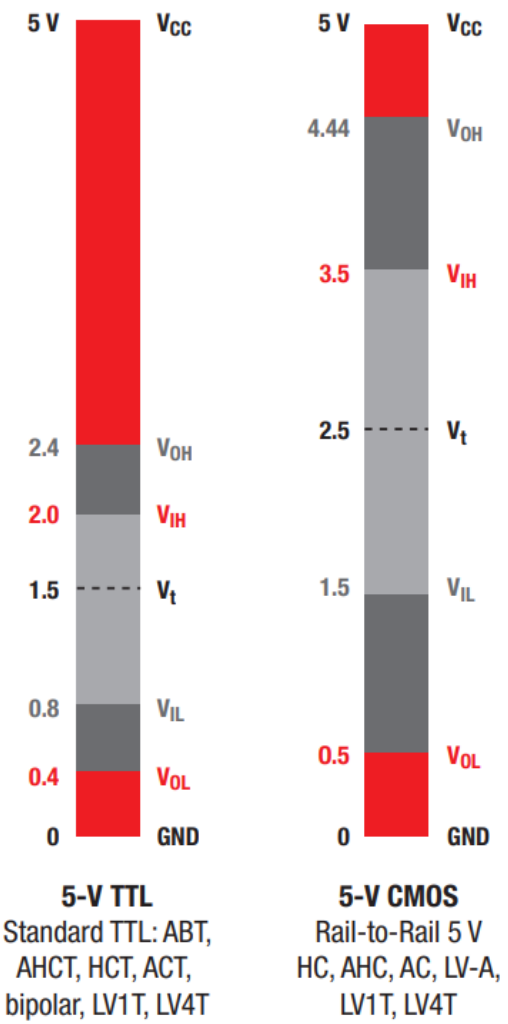
Intermediate Region

- Notice the gap between V_{IH} and V_{IL}? Because of this gap, there is a sharp distinction between 0 and 1
- This makes digital signals less susceptible to errors caused by "background noise," which would normally affect an analog signal
- Also notice the outputs are more restricted than the inputs, this allows for slight voltage drop or rise between the output and input



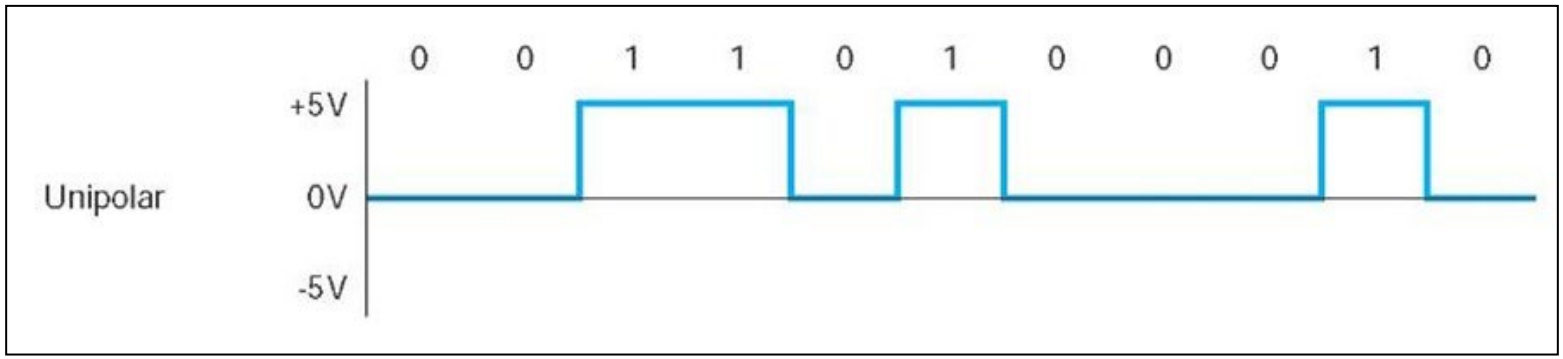
Mismatched logic levels

- Look more closely at the two levels to the right
  - Consider TTL output to CMOS input
  - The TTL High output could be 3.0V (>2.4V), but this will fall in the intermediate range for CMOS input (<3.5V)
  - So even though both are “5V Logic,” they may fail to work together
- This mismatch is even worse with differing voltage levels, ie. 3.3V and 5V
  - Devices called level shifters are used between the different logic levels



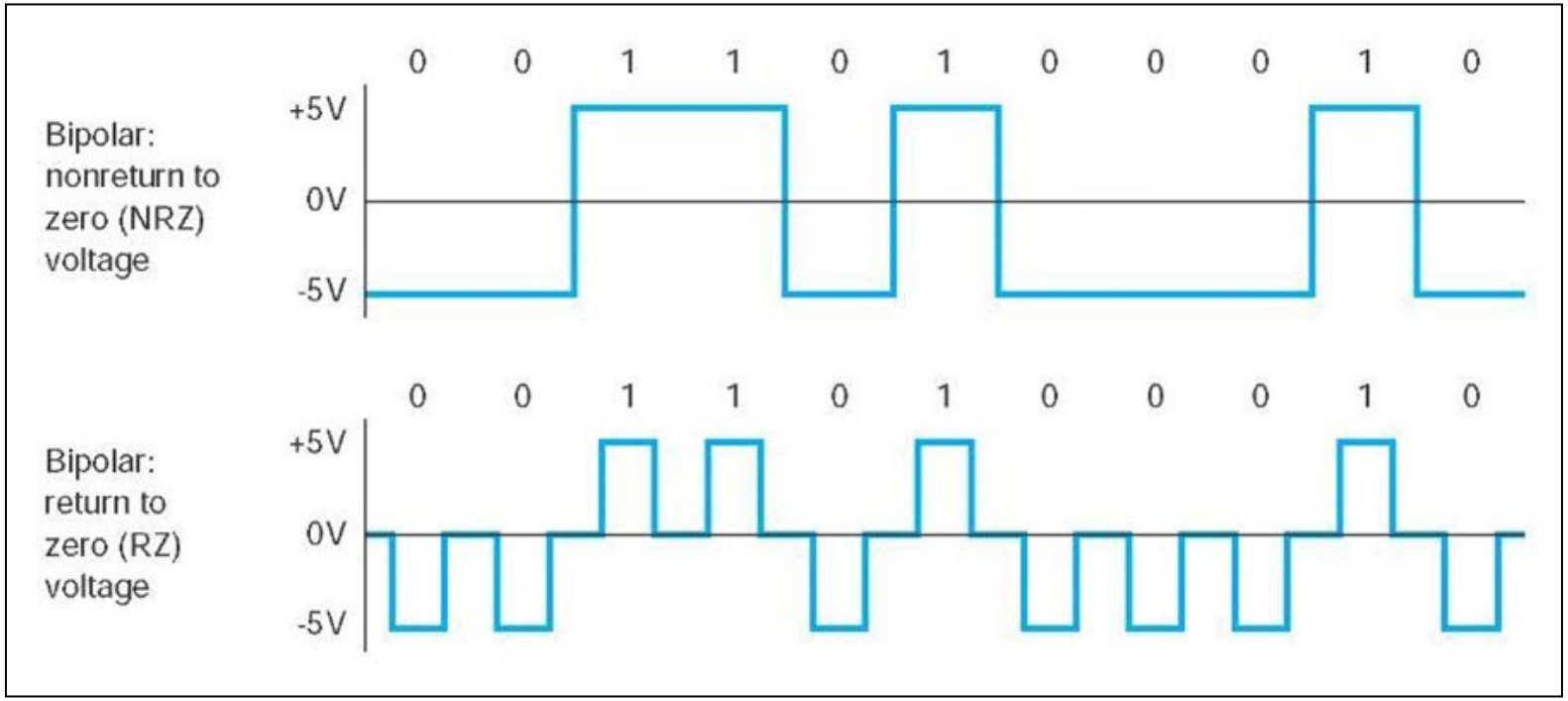
Unipolar Digital Signals

- The voltage levels we just looked at are examples of unipolar signals
- In unipolar, low is assigned to 0 voltage, a.k.a ground. The high is assigned to the output voltage
- The idle state (when not actively transmitting) can either be High or Low



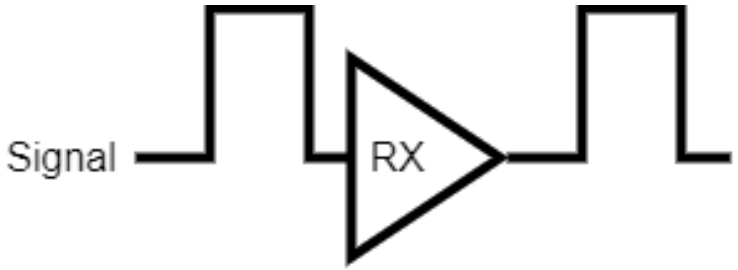
Bipolar Digital Signals

- In some cases, you will see both positive and negative voltages used for a signal, called bipolar
- The signal may or may not return to 0V when idle

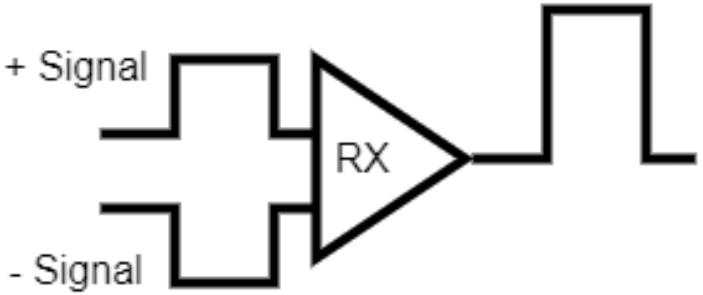


Single Ended Vs Differential

- Signal can use a single wire and be measured compared to the device's 0V (GND)
  - This is called **Singled Ended**
  - This requires a common ground between the transmitting device and the receiving device
- An alternative is to use two lines and measure the difference
  - The signal is split and transmitted with opposite polarity, giving x2 the effective signal when received



Single Ended Signal



Differential Signal

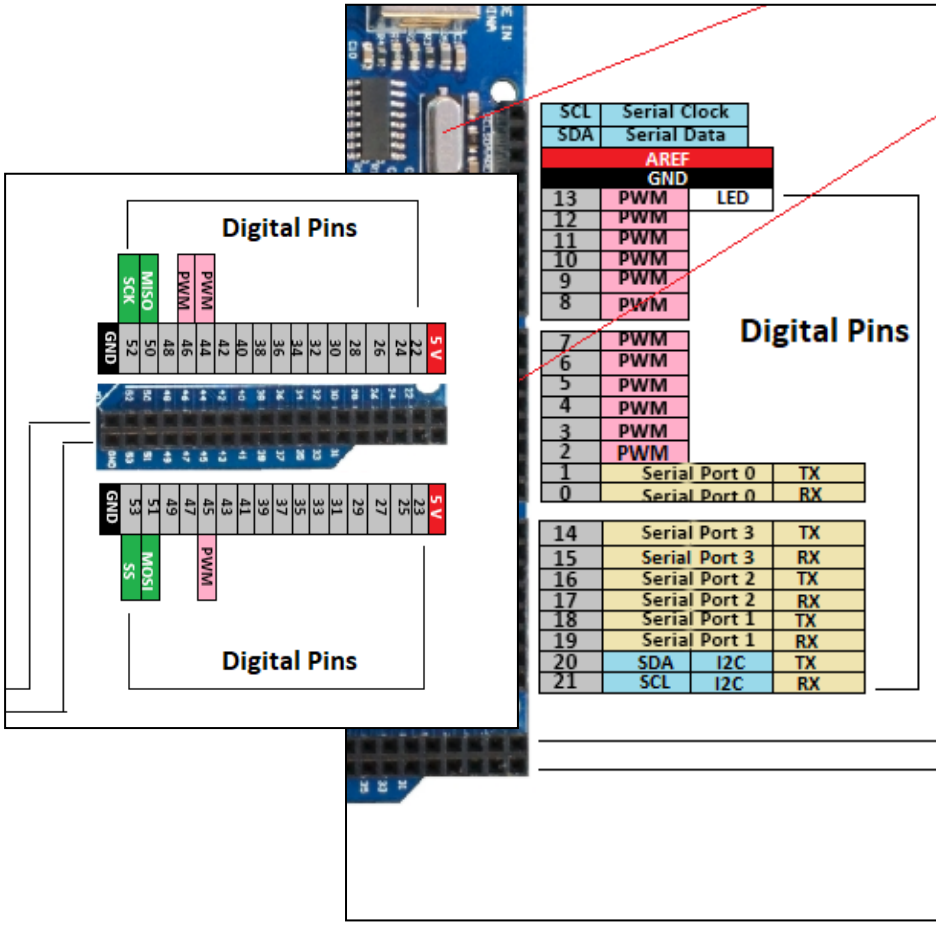
Arduino Digital Signals

The Mega2560 Digital IOs have these characteristics

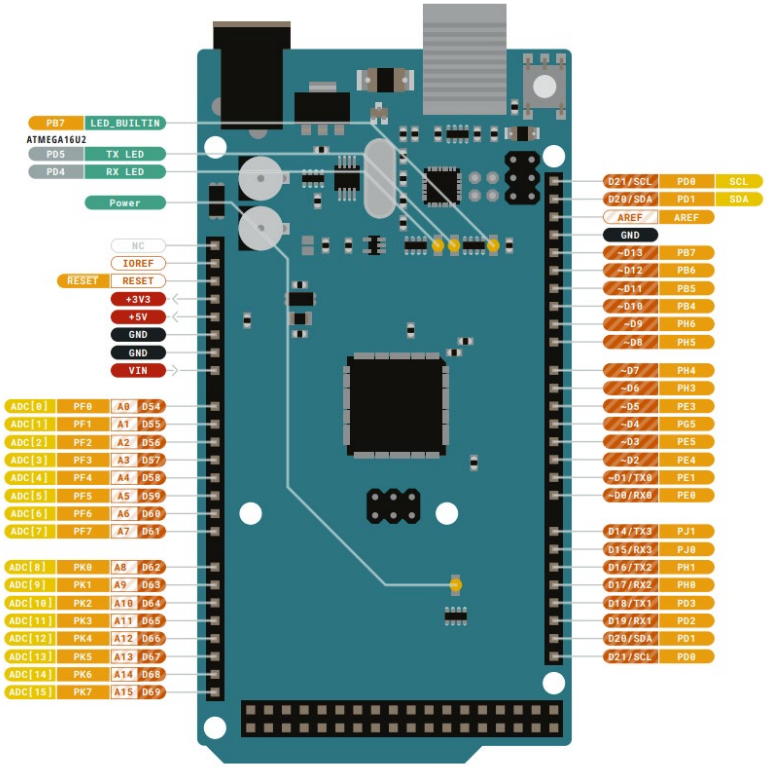
- Voltage Signals
- Unipolar
- Single Ended
- 5V TTL Logic

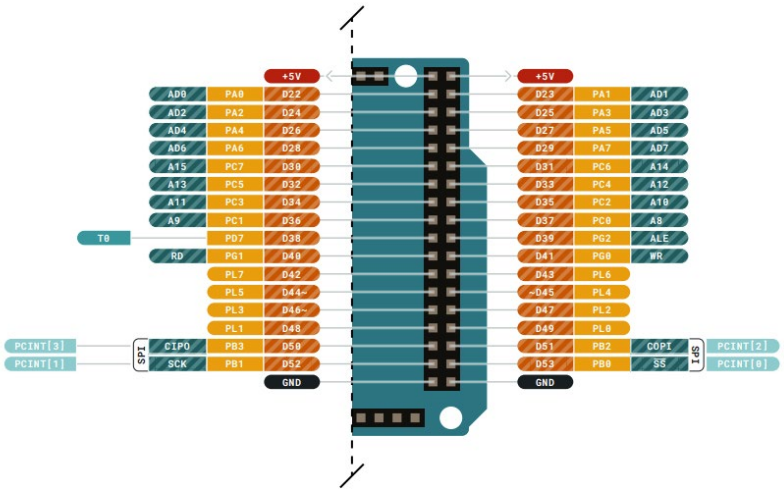
Digital Pins

- Except for the power pins, all of the pins on the Arduino headers can be used for basic digital input output
- Some of the pins have additional special functionality we will cover in later lectures
  - Pins A0-A15 can be used for analog input
  - Pins 0-1, 14-19 are for Serial Communication
  - Pins 20-21 I2C Communication
  - Pins 50-53 SPI Communication
  - Pins 2-13 can be used for analog output via a Technique Called Pulse Width Modulation (PWM)



Arduino Mega Pin Layout



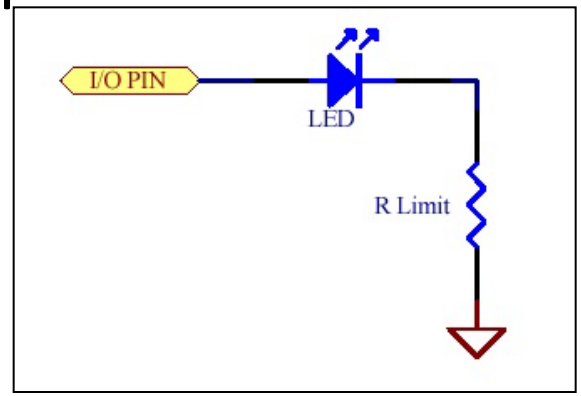


The full detail pinout can be found at: https://docs.arduino.cc/resources/pinouts/A000067-full-pinout.pdf

Digital Output Applications

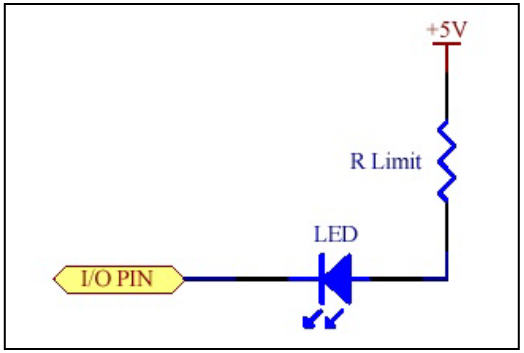
To use a pin as an output, the Arduino controls the voltage of the output

- Driving an LED with I/O pin as a current source.



HIGH pin turns LED on.
LOW pin turns LED off.

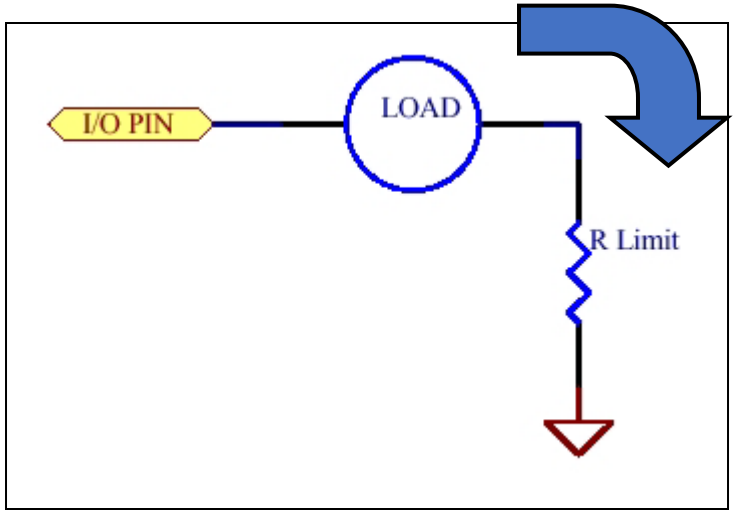
- Driving an LED with I/O pin as a current sink.



LOW pin turns LED on.
HIGH pin turns LED off.

Output Pin as a Current Source

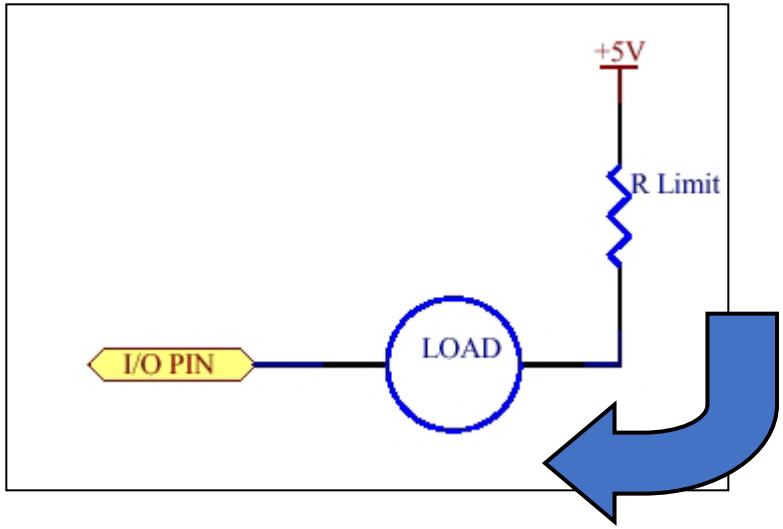
- In this case, the pin supplies the current to the load
- When a pin is HIGH, current flows from the pin through the load to GND
- When a pin is LOW, no potential difference between the pin and GND, so no current flows
- The Arduino can safely supply up to 40 mA



A current-limiting resistor may be needed in some applications to ensure the microcontroller is not damaged

I/O Pin as a Current Sink

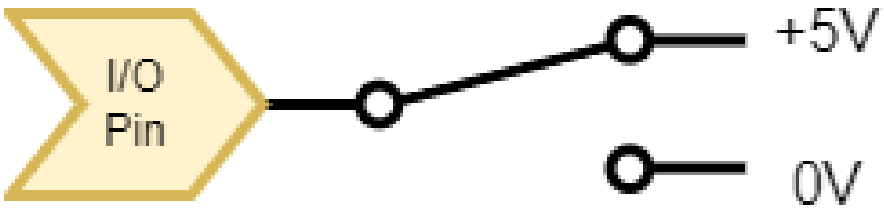
- Sink in current sink refers to the pin's ability to receive current
- The current comes from some other source (External Power Supply, Arduino 5V pin)
- When pin is LOW, current flows from the +5V power supply through the load into I/O pin
- When the pin is HIGH, no potential difference between the pin and +5V, and no current flows
- The Arduino can safely sink 40 mA, the same as the source limit



Again a current-limiting resistor may be needed in some applications to ensure the microcontroller is not damaged, especially if the external source is greater than 5V

Input Pins

- Digital Inputs are even simpler
- The Arduino measures the voltage at the pin and returns a 1 for 5V and 0 for 0V
  - Similar to a multimeter, in voltage mode, the input pin will have a high resistance, so very little current will flow when the pin is being read



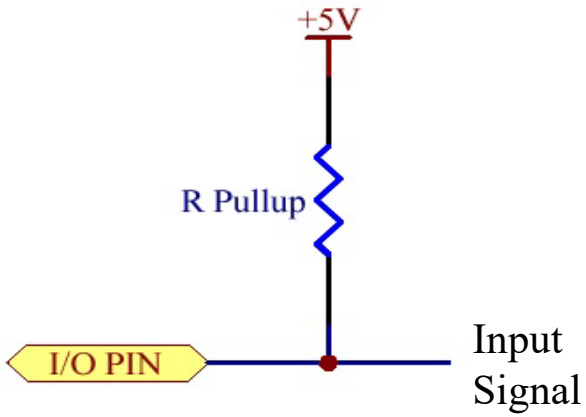
Floating Inputs

- So what happens if you try and read a pin that is not connected to anything
  - The unconnected pin is called “Floating”
- The pin will be in an unpredictable HIGH or LOW state depending on small charge or discharge paths on the PCB, Radio Frequency noise, temperature effects, etc.
- In most digital input applications, you want to remove the random effect and give a defined state by using a “Pull Up” or “Pull Down” resistor

Pull Up/ Pull Down

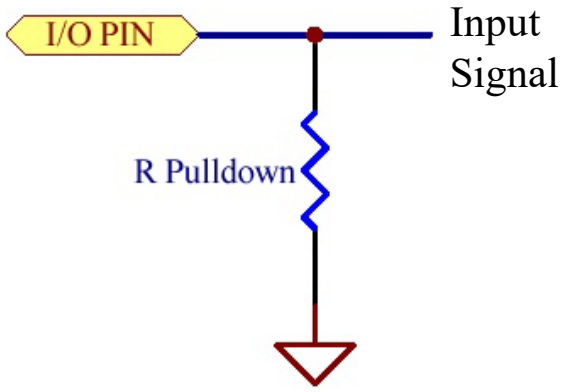
Pull Up resistors:

- Connect the I/O Pin to +5V to “pull up” the voltage to the 1 value when the input is idle



Pull Down resistors:

- Connect the I/O Pin to 0V to “pull down” the voltage to the 0 value when the input is idle



Basic Digital I/O Functions

- We must set the pins as Input or Output
- The digital pins on the Arduino Mega are set to input as default (since this high resistance it is the safe state)
- To use digital pins we have 3 different functions
  - *pinMode(pin, mode)* sets the pin as an input or output
    - **INPUT** and **OUTPUT** are defined as variables so they can be used in mode
  - *digitalWrite(pin, value)* is used set a pin to 5V or 0V
    - **HIGH** and **LOW** are again defined variables so you can use them for value
  - *digitalRead(pin)* reads the pin and returns either a 1 (for 5V) or a 0 (for 0V)
  - For all, pin the number of the pin as labeled on the Arduino

Configuring Pins Examples

- Here is an example of setting pin 13 as an OUTPUT, and then “turns on” the pin for 1 second and then turns it off.

```
void setup()
{
  pinMode(13, OUTPUT);           // sets the digital pin 13 as output
}

void loop()
{
  digitalWrite(13, HIGH);        // sets the digital pin 13 on
  delay(1000);                   // waits for a second
  digitalWrite(13, LOW);         // sets the digital pin 13 off
  delay(1000);                   // waits for a second
}
```