PAT 451/551 INTERACTIVE MEDIA DESIGN I

PULSE-WIDTH MODULATION

MORE I/O

So far, we've seen:

Digital Input

- Sense voltage state (High or Low; +5V or GND) at a point in a circuit
- Switches
- Digital Logic

Digital Output

- Set a voltage state (High or Low; +5 or GND)
- Turn on/off LEDs

Analog Input

- Sense a continuous voltage range between GND and +5V
- ADC
- Sensor data

WHAT ABOUT ANALOG OUTPUT?

For proper continually-varying analog output, you'd need a digital-to-analog converter (DAC). This is what your soundcard does to convert digital audio signals to something you can year.

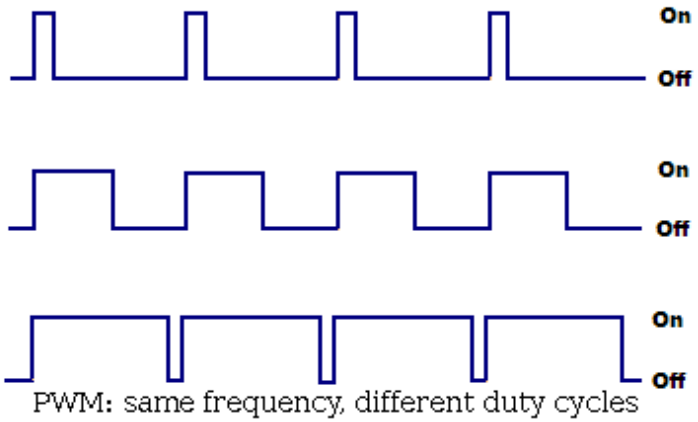
Arduino does have a DAC (but it's tricky to use)

But, it does have Pulse Width Modulation (PWM), which we can use to continuously control some types of displays.

PULSE-WIDTH MODULATION

As the name suggests, it involves a stream of pulses with variable width.

The frequency is constant*, but the "duty cycle" or pulse width changes.



PWM: same frequency, different duty cycles

Duty cycle is the proportion of HIGH time vs LOW time in each cycle. So 50% duty cycle is a true square wave.

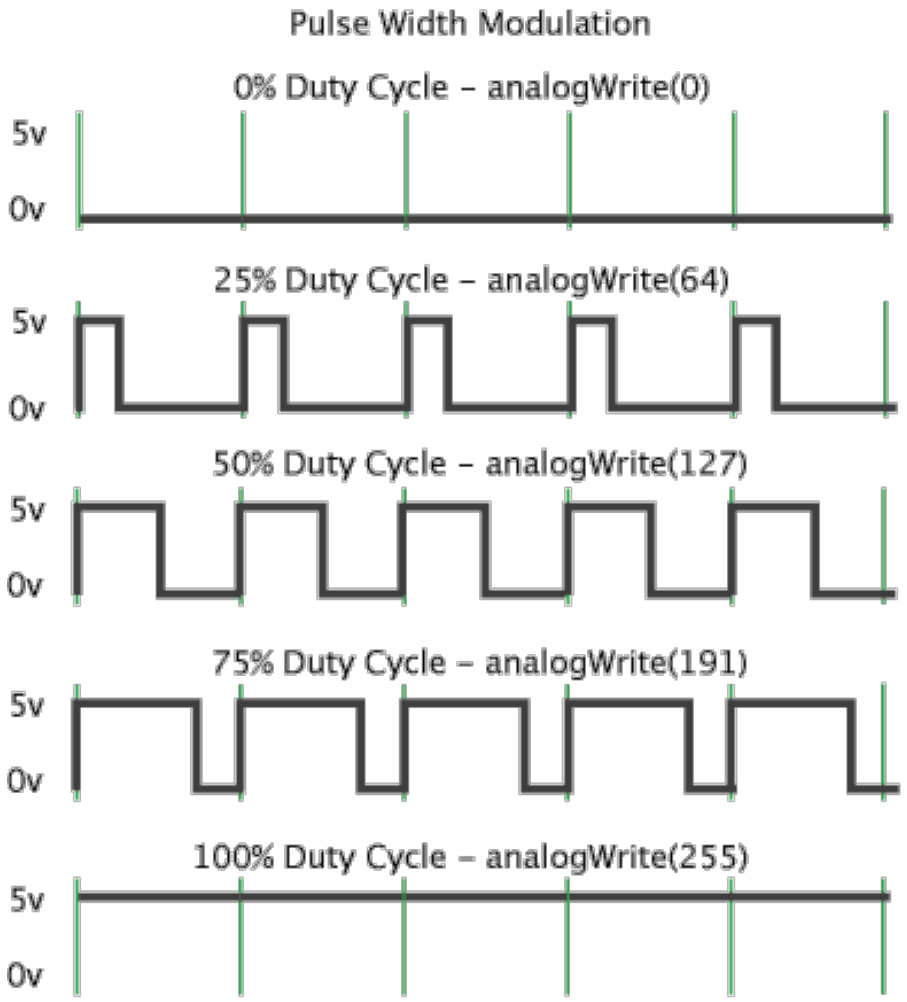
*(On our Arduino UNO, the PWM frequency is 490Hz on most pins. It is 980Hz on pins 5 and 6. There are ways to change this, but it isn't really important for most applications.)

PWM IN ARDUINO

`analogWrite(pin, val)`

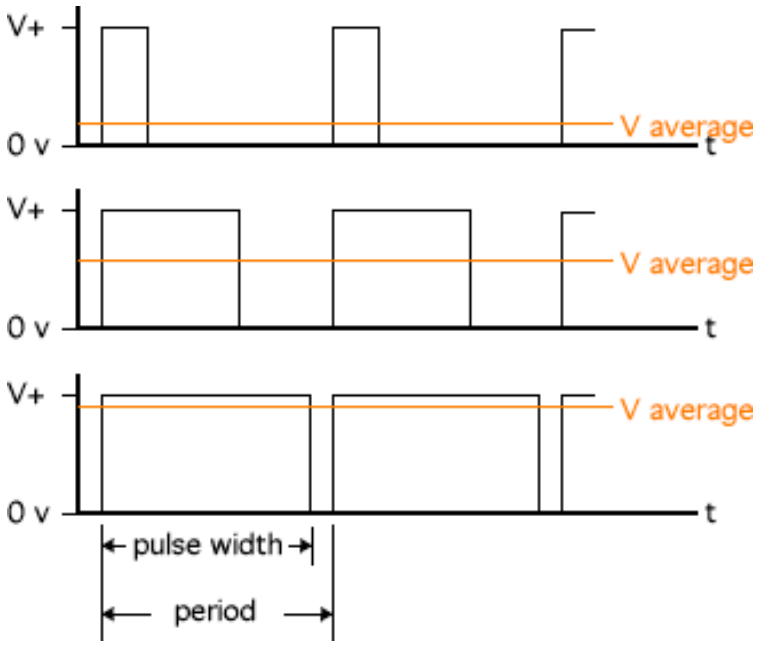
- *pin* must be a valid PWM pin.
  - on Arduino Uno, digital pins 3, 5, 6, 9, 10, 11
  - Indicated by a ~ on the Arduino board next to the pin number
- *val* is between 0-255
  - 0 -> 0% duty cycle (off)
  - 63 -> 25% duty cycle
  - 127 -> 50% duty cycle
  - 191 -> 75% duty cycle
  - 255 -> 100% duty cycle (on)

PWM SIGNALS


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

WHY IS PWM USEFUL?

- "Simulate" an analog signal whose voltage is the average value of the PWM signal over time
- If PWM signal is fast enough (high frequency), we don't notice the switching on some displays, e.g., LEDs
- You can also low-pass filter (smooth) the PWM signal
- Many devices have a "low-pass" response, they won't be able to follow the signal exactly, so the effect is the same as smoothing it.



PWM APPLICATIONS

Dim an LED

Mix colors of an RGB LED

Drive a motor with variable speed

DIM AN LED

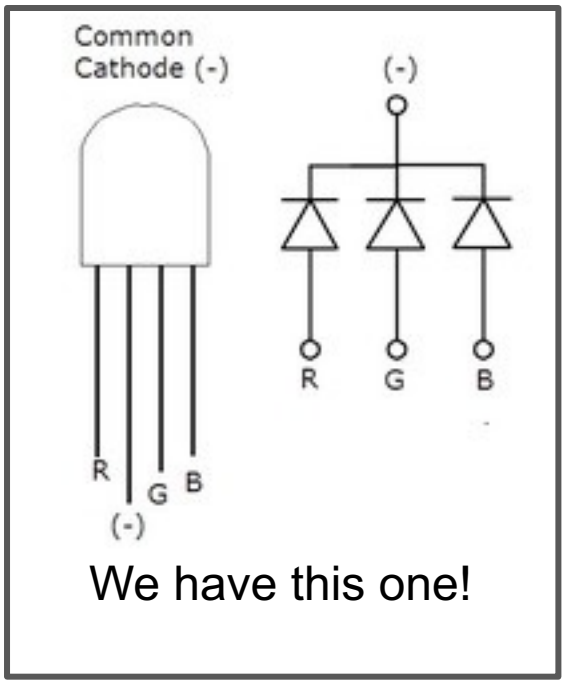
```
// see ledDimmer.ino example

int ledPin   = 3;
int analogPin = 0;

void setup() {
  pinMode(ledPin,OUTPUT);
}

void loop() {
  int reading = analogRead(analogPin);
  analogWrite(ledPin,reading>>2);
  // bit-shift right by two (divide by 4), to
  // convert from 0-1023 to 0-255 range
  delay(10);
}
```

RGB LED

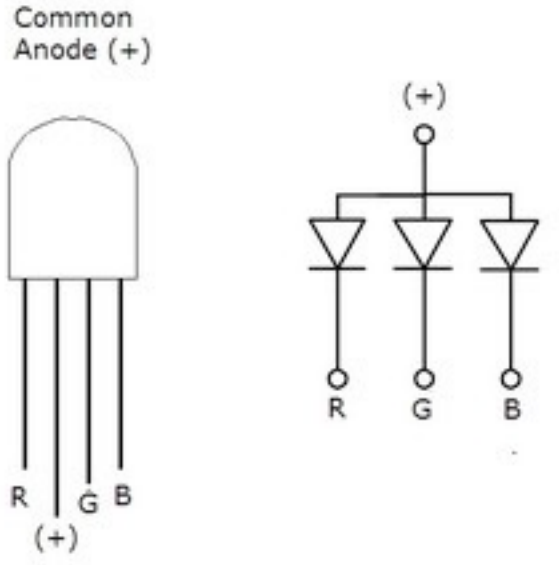
Simply: 3 LEDs (red, green, blue) in one package.

2 varieties: Common Anode (+) or Common Cathode (-)

We stock Common Cathode. They are a bit more intuitive to use with an Arduino.



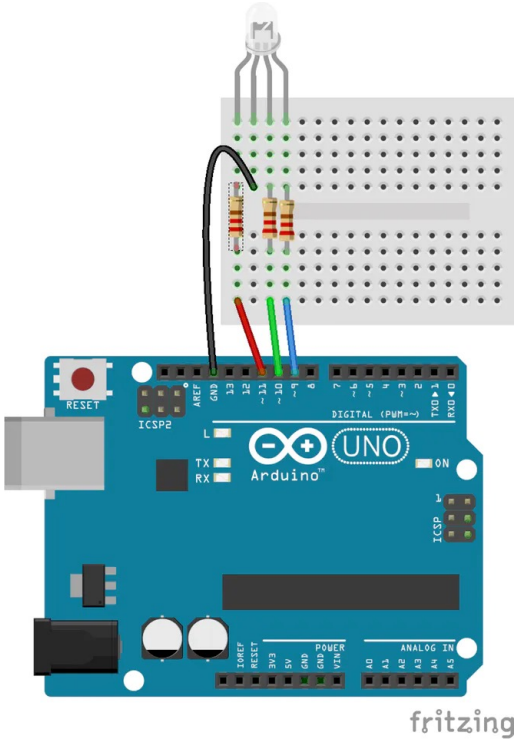
We have this one!



RGB LED

To use it:

- Connect each of the R, G, B pins through a resistor to separate PWM outputs of your Arduino.
- Connect the common (longest) pin to ground.
- Control the PWM duty cycle of the 3 outputs with **`analogWrite()`** to set each color channel and vary the overall color.



- See RGBFade.ino example for fancy color cycling.

MORE INFO

- https://www.arduino.cc/en/tutorial/fade
- https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM