EEE-201 Chapter 1

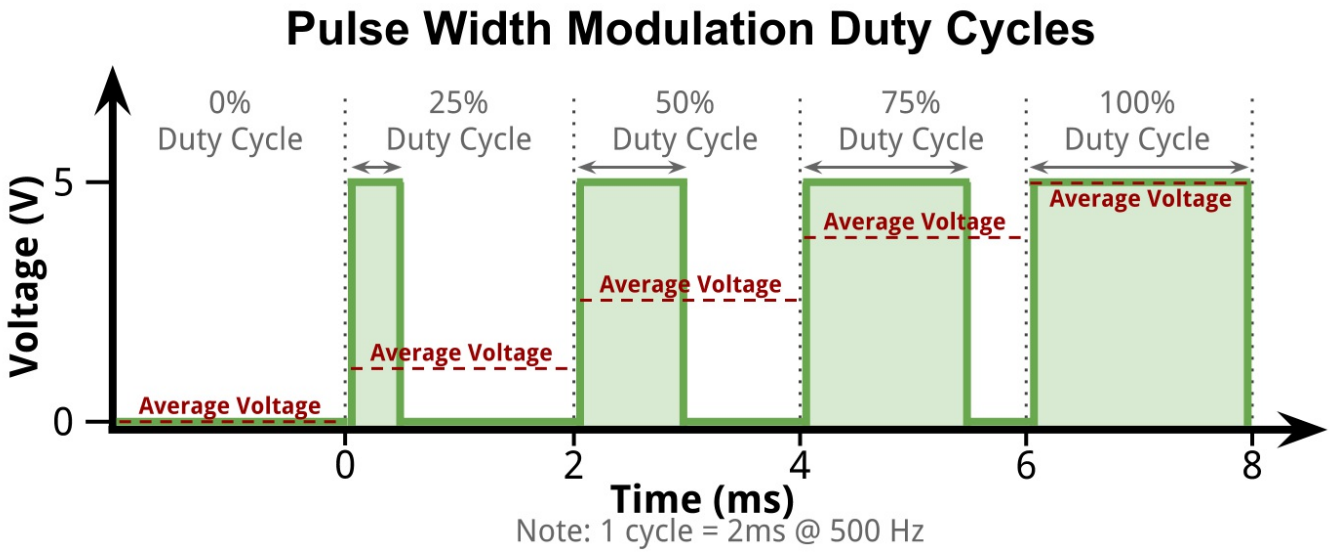
In the Arduino 1 class, we covered the topic of pulse width modulation (PWM); we will revisit that concept in this chapter and see how it is used to control servos. We'll be covering the following concepts:

- pulse width, voltage and period
- duty cycle
- servo control

Table of contents

Part 1 - pulse width modulation

We previously learned that PWM is a method of using a digital output pin (which can only be `LOW` or `HIGH`) and turning it into a quasi-analog output pin which can have almost any value between 0V to 5V; the figure below shows how this is achieved:



A PWM is a signal (square wave) made up of a repetitive pattern pulses or cycles. As the name implies, a square wave as a square pattern with a voltage being either `LOW` (0V) or `HIGH` (5V in this case). The start/end of the pulse period in designated by the voltage changing from 0V to 5V. So, in the example above, we can see that the pulse period is set to 2ms (or 0.002s) - that is, every 2ms the signal repeats itself.

These pulses are often designated in units of frequency which is just the inverse of the pulse period in seconds - in this case, 1/0.002s = 500 Hz.

The amount of time the signal is at `HIGH` (or at 5V in this case) defines the duty cycle or pulse width; for example, if the pulse is `HIGH` for 1ms, the duty cycle equates to 50% (1ms/2ms). At a 50% duty the cycle, the average voltage during the entire cycle will be 50% of the pulse voltage (5V * 50% = 2.5V). In this way, by adjusting the pulse width, we can adjust the voltage on the output pin. We had used this technique in EEE-105 to control the brightness of a LED, but in this class we will use the same technique to control a servo!

Part 2 - controlling a servo

We can use a signal generator to show how to use PWM to control a

servo. The majority of hobby servos requires a 50Hz signal (0.02s or 20ms) with a pulse width in the range of 0.5-2.5ms; if you do that math, that equates to a duty cycle range of 2.5-12.5%.

We will use the Rigol DG1022A signal generator to generate a square wave using the `Pulse` functionality. Note that we can't use the `Square` function on the signal generator because the smallest duty cycle it allows is 20%, which is out of the range of the servo - so instead, we use the `Pulse` function.

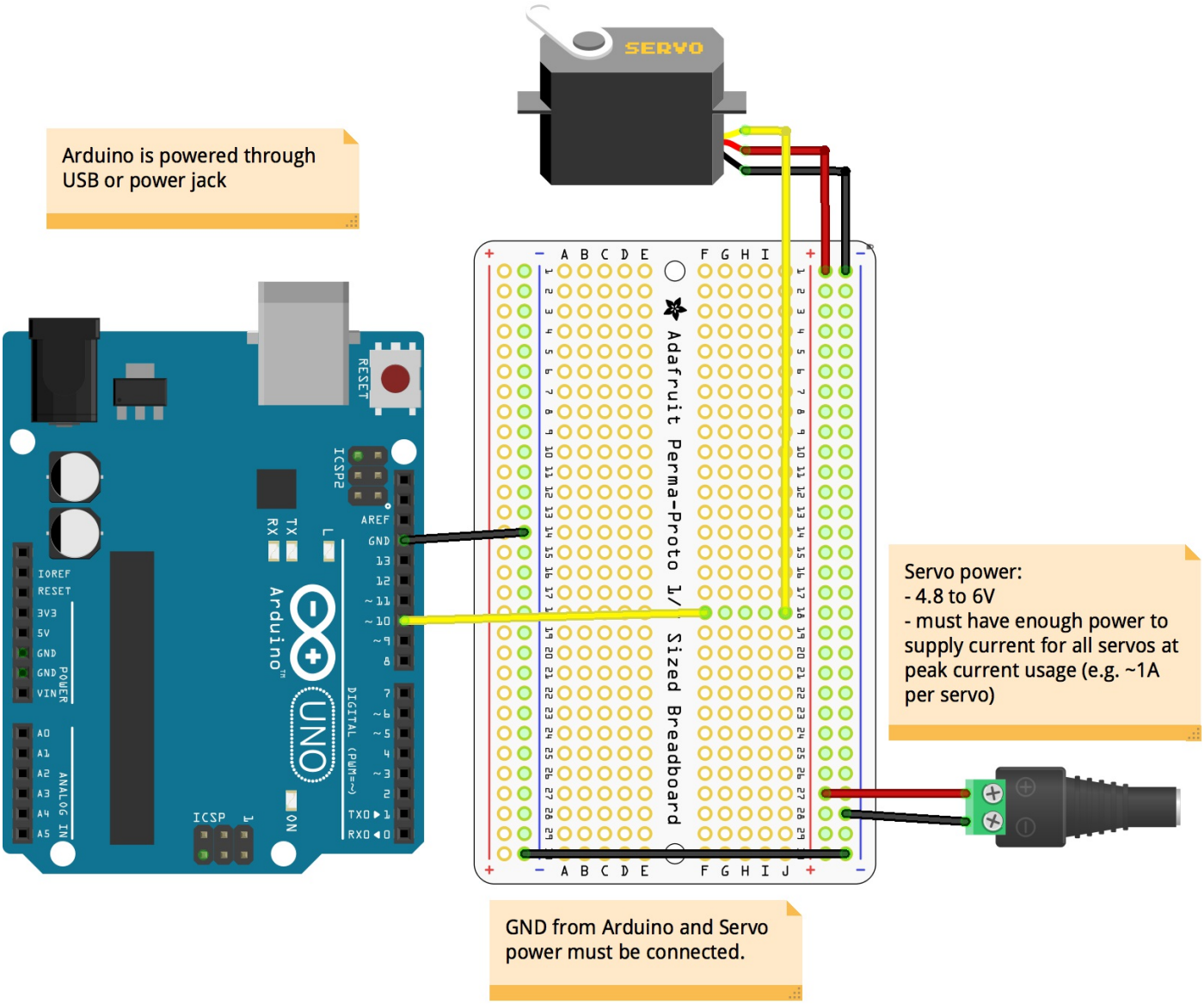
By hooking the servo up to the signal, we can see that by adjusting the duty cycle, the position of the servo also changes. It's important to note here that the pulse width can only be changed at a frequency of 50Hz; if it is changed more rapidly, you can get erratic behavior in your servo.

Part 3 - important note about servo power and wire color

In this class, we will be powering the servos directly from the Arduino. What that means is that the red (+V) wire of the servo is hooked up to the +5V pin of the Arduino and thus the power is being drawn through the voltage regulator of the Arduino. This voltage regulator has a current limit of 0.5A when powered through USB and 1A when powered through the external power jack.

Depending on the type of servo and how it's being used, servos can pull a large amount of current (~1A or so depending on the servo). The way we are using the servos in this class, the current draw is not very large (perhaps 40mA). So, when running two servos, we may experience 40mA * 2 = 80mA which is well within the 0.5A voltage regulator limit.

However if you were going to use a servo in a high load case (such as driving a robot wheel), you risk drawing too much power through the voltage regulator of the Arduino and then burning it up. The way of getting around this is, instead of attaching the red +V wire to the Arduino, attach it to an external voltage supply like this:



For reference, servos come with various wire colors; the table below details the function of each colored wire.

Futaba	JR (*this class*)	HiTec	Function
Red	Red	Red	+V (typically 4-6V)
Black	Brown	Black	GND

White	Orange	Yellow	DATA