LEDs and Sensors Part 2: Analog to Digital

In the last lesson, we used switches to create input for the Arduino, and, via the micro-controller, the inputs controlled our LEDs when playing Simon. In this lesson, we will again use input to control output, but this time we will use *analog input* and *analog output*.

Analog Input

The real world is not digital. Consider temperature fluctuation as an example, it generally moves within some range of values without making large abrupt changes. We can measure aspects of our world like temperature, light intensity, forces, or whatever using *analog sensors*. In a digital device, the resulting signals are stored as sequential digital data.

Consider the analog signal depicted below.

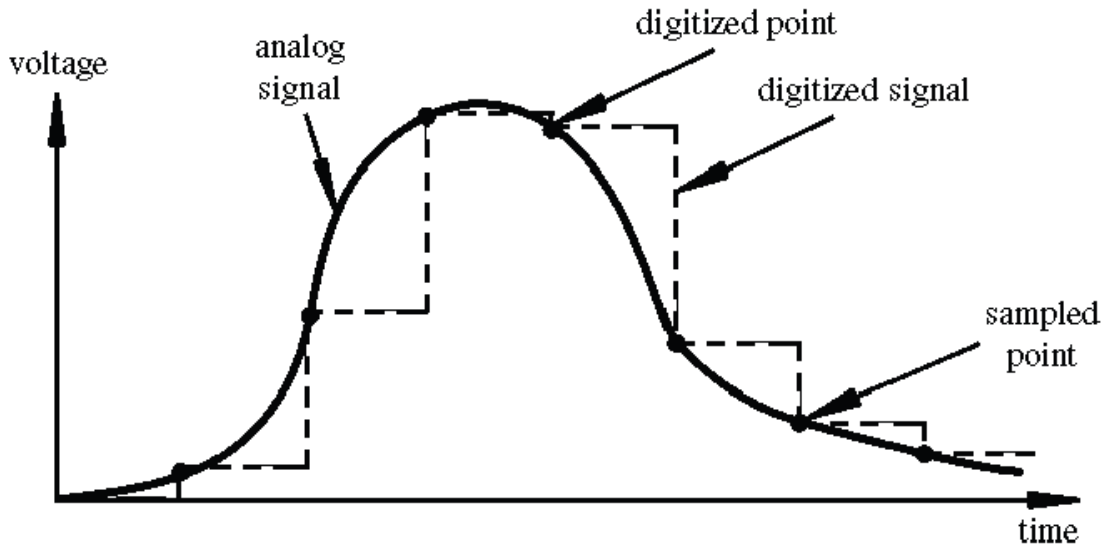


Image credit: Tod Kurt

In a digital device, we discretize the input signal range (i.e., 0 to 5 Volts for the Arduino) into different voltage levels or intervals called *states*. The number of states is the *resolution*. Common resolution values range from 256 states (i.e., stored in 8 bits) to 4,294,967,296 states (i.e., stored in 32 bits); the Arduino uses 1024 states stored in 10 bits. This means that 5V/1024 or 4.8 mV is the smallest voltage change that you can measure.

The translation of analog voltage values into different states is called *Analog to Digital Conversion*. One small part of the micro-controller on the Arduino board is dedicated to translating analog voltages into digital states; it is the *Analog to Digital Converter* (*ADC)*.

On the Arduino, the `analogRead()` command applied to an analog input pin is used to invoke the ADC process.

Analog Output

As you know, the digital pins on the Arduino board can output either high (5V) or low (0V) — on or off. But what if we want an output voltage in between 0V and 5V--- something simulating an analog signal? Well, we can do this using a process called *Pulse Width Modulation* (*PWM)*.

As you know, we can't use the Arduino's digital pins to *directly* supply say 2.5V, but we can *pulse* the output on and off really fast to produce the same effect. If you were to watch the output signal on an oscilloscope, you would see the output pulsing between high and low at regular intervals. Since this on-off pulsing is happening so quickly, the connected output device "sees" the result as a 50% reduction in the normal voltage (in this example).

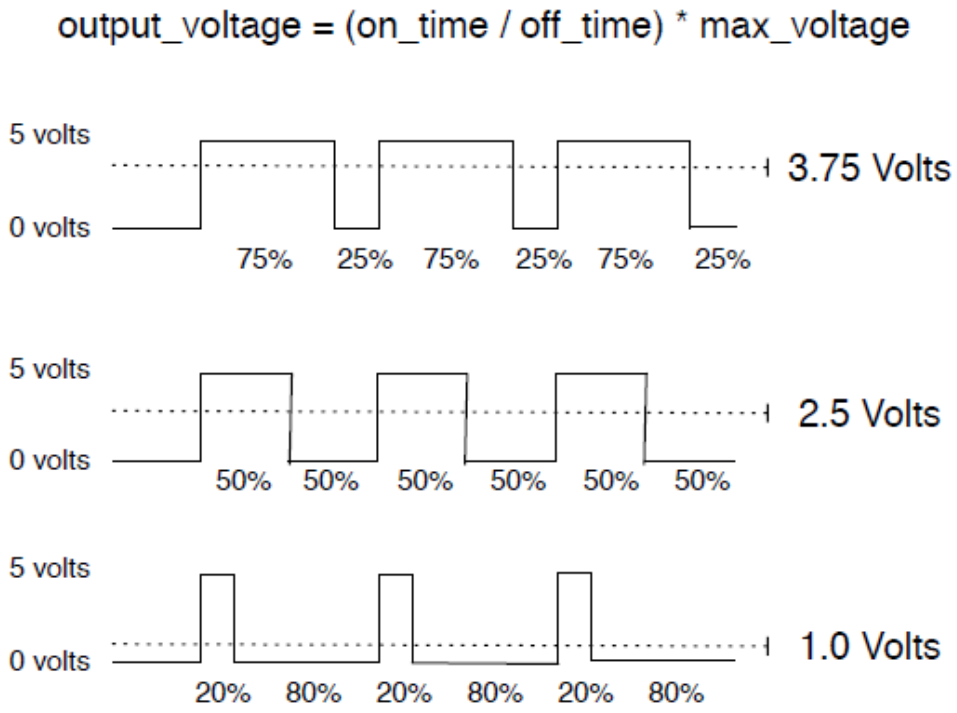


Image credit: Tod Kurt

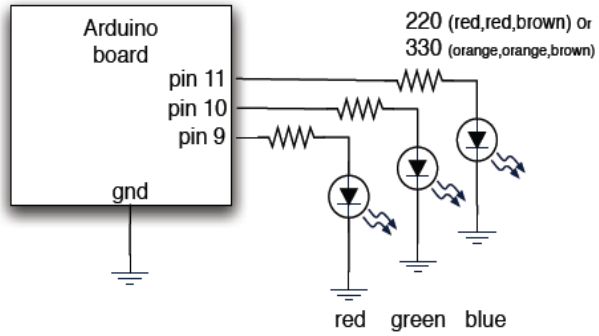
We can vary the output voltage (the "effective voltage") by regulating or "modulating" the width of the high pulse. For example, if we make the high pulse 25% as "wide" (in time) as the low pulse, the effective voltage will 25% of full voltage.

On the Arduino, the `analogWrite()` command applied to an analog output pin is used to invoke the PWM process.

In the next 2 exercises, you will use an old friend, the potentiometer, to generate analog input that is used to control PWM output.

Analog I/O Exercise 1: A LED mixer

In this exercise, we will make an LED fader/mixer using a schematic developed by Tod Kurt; create the circuit depicted below on your breadboard.



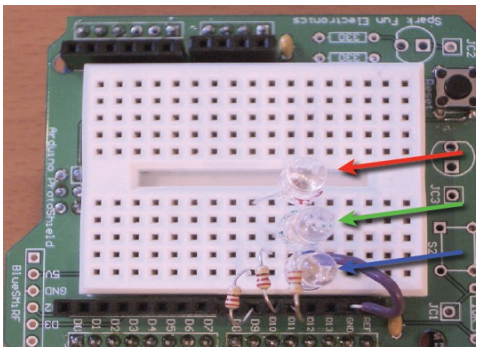


Image credit: Tod Kurt

With RGB you can make any color (except black)

The program to drive the fader/mixer is available as an Arduino tutorial. In this program, you use PWM to change the intensity of the light emitted by the 3 LEDs on your board. You can mix the light using a diffuser such as a pin-pong ball. Try it out!

Note that the program also outputs the current color values to the serial port if you turn on "DEBUG;" try that as well.

Now modify the program in some interesting way and be prepared to demonstrate and explain your modification to the class.

Analog I/O Exercise 2: A Theremin

In this exercise, we will use a potentiometer to generate analog input. DO NOT BREAKDOWN THE CIRCUIT you created in exercise 1, but rather add a speaker to pin 7 and potentiometer to pin 2. Now run the Theremin program by Tod Kurt.

Once you understand the program, modify it so that the input from the potentiometer not only drives the speaker but also drives the LEDs. Try to create a seasonally “spooky” sound and light device. Be prepared to demonstrate your device and explain the changes you made to the class.