



The Big Idea:

Computers are **binary**. Everything is on or off, high or low, one or zero. There are no gradations in between. But the real world is **analog**. A light may have an infinite number of brightness settings from off to full on. Sounds range from barely detectable through painfully loud.

In this lesson we learn how to connect the digital Arduino™ to the analog world such that we can detect where an input is within its natural range, and control an analog device by providing an output that matches its natural range.

Background:

Up to now all work with the Arduino™ has been with digital inputs and outputs. A button is either "pushed" or not. An LED is either on or it isn't. The `digitalRead()` statement returns either HIGH or LOW, while the `digitalWrite()` statement either sets a pin HIGH or LOW.

But the real world doesn't work that way. States have an infinite number of values. Here are some examples:

- The music was very loud.
- She spoke in a moderate voice.
- The piano played softly.
- Miracle Max, from *The Princess Bride* (1987), says, "It just so happens that your friend here is only *mostly* dead."

Analog input

In input mode a digital pin is either HIGH or LOW, never anything in between. This characteristic is referred to as binary, meaning it is either one thing or not that one thing. This is sufficient if what is to be measured is itself binary. Examples are a button that has been pushed or not, a sensor that sees a light or doesn't, and a switch that is either on or off.

But what if what needs to be measured is not binary but rather something that can have multiple values between the binary extremes? Examples of this: How far is the knob turned? How loud is the sound? How bright is the light? How salty is the water?

The Arduino™ Uno's analog pins, pin A0 through A5, can each measure a voltage between 0 and +5 volts to an accuracy of about 0.005 volts. This measurement is expressed as a number between 0 and 1023.

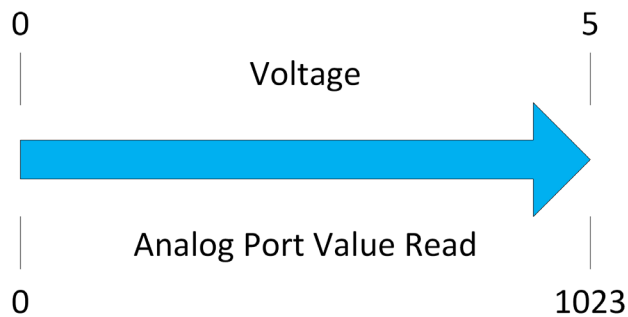


Figure 9-1. Analog port representation of voltage

The C language provides a tool for translating analog readings into other scales, from 0 to 100 for example. This is called *mapping*.

Analog output

While it is true that an analog port can accurately measure any voltage from 0 through +5, the Arduino™ Uno has no way of providing an output voltage in that range. The output voltage of an Arduino™ pin, be it an analog or digital, is +5 volts or zero, and nothing in between.

Yet the C language does provide a way of writing an analog value that approximates the range of 0 through +5 volts. It does this by switching rapidly between 0 volts and +5 volts.

If the output of a pin is at +5 volts half the time and 0 volts the other half, then the average voltage is 2.5 volts. Not only that, but an LED attached to that pin will produce half as much light as when the average voltage is +5 volts.

By modifying the ratio of the time the output pin is at +5 volts to the time it is at 0 volts, the average output voltage can be set to any value from zero through +5 volts.

The process of repeatedly setting an output pin HIGH for a fixed time then returning it to LOW is called *pulsing*. A single rise, wait, and fall cycle of the output voltage is called a *pulse*. The length of time the pulse is HIGH is called the *pulse width*. Finally, using pulses alternating with periods of LOW output is called *pulse width modulation*.

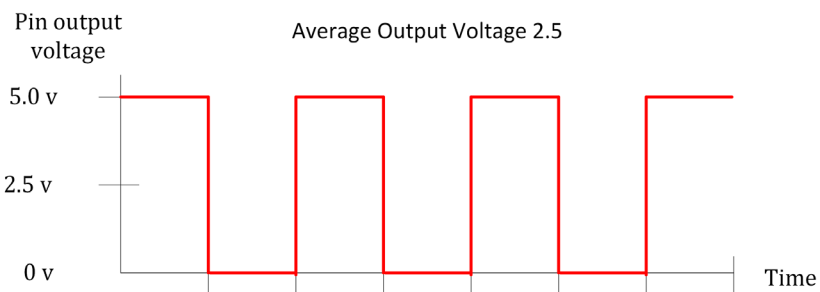


Figure 9-2. Pulse width modulation with average voltage +2.5 v

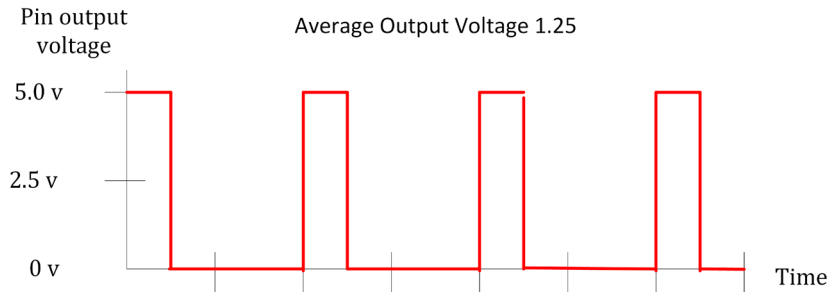


Figure 9-3. Pulse width modulation with average voltage +1.25 v

Not all digital pins of the Arduino Uno can be pulsed. Those that can be are indicated on the Arduino™ by a tilde mark (~) before the pin number. There are six such pins: 3, 5, 6, 9, 10, and 11.

Table 9-1. Vocabulary

Term	Definition
analog	A thing that is representative of the state of another thing. In this lesson, the read value for a potentiometer will be a number between 0 and 1023. The actual number is analogous to the rotation of the shaft of the potentiometer. The farther it is turned the higher the read number.
duty cycle	The length of time that a specified pin is HIGH vs. LOW; duty cycle is expressed as a percentage.
mapping	The determination of a number within a range based on another number and its range. If the number 8 is between 1 and 12 the mapped number between 100 and 160 is 140. The proportions are the same.
pulse	The setting of a pin from LOW to HIGH for a specific time then returning it to LOW.
pulse width	The length of time a pulse is HIGH.
pulsing	Repeatedly setting an output pin HIGH for a fixed time then returning it to LOW.
pulse width modulation	Using pulses alternating with periods of LOW output.
potentiometer	A variable resistor, usually disk shaped with a knob that may be turned to select the resistance.

Description:

Although they are related, reading an analog value from an analog pin is different from writing an analog value, which is sent to a digital output pin capable of sending pulses.

Programming statements

The three C-language commands that work with analog input and output are `analogRead`, `analogwrite`, and `map`.

The command `analogRead()` returns a number from 0 through 1023 that is proportional to the voltage from 0 through +5 volts on the read pin. The command `analogwrite()` sets an output pin's average voltage between 0 and +5 volts by writing an integer in the range of 0 through 255.

The command `map()` remaps a number from one range to another.

`analogRead`

`analogRead(pinNumber)`

`value` Integer between 0 and 1023

`pinNumber` Integer that specifies which analog port is to be accessed. On the Arduino™ Uno the analog ports are 0 through 5.

example `int val;`

`val = analogRead(A2); // returns 0 to 1023`

`analogwrite`

`analogwrite(pinNumber, value)`

`value` Integer from 0 through 255 that specifies how much time the pin specified by `pinNumber` is HIGH vs. LOW. Expressed as a percentage of 255, this value is called the *duty cycle*. For example, a value of 153 specifies a duty cycle of 60%, meaning the pin is switching rapidly between HIGH and LOW in such a way that it is HIGH 60% of the time.

`pinNumber` Integer that specifies which pin is to be accessed. Use of pins 5 and 6 are not recommended.

example `analogwrite(3, 100);`

`map()`

`map (value, fromRangeLow, fromRangeHigh, toRangeLow, toRangeHigh)`

<code>value</code>	The number to be mapped. It is expected to be between <code>fromRangeLow</code> and <code>toRangeLow</code> .
<code>fromRangeLow</code>	Integer representing the low end of the range being converted from.
<code>fromRangeHigh</code>	Integer representing the high end of the range being converted from.
<code>toRangeLow</code>	Integer representing the low end of the range being converted to.
<code>toRangeHigh</code>	Integer representing the high end of the range being converted to.

example	<pre>int newVa1ue; newVa1ue = map (15, 0, 50, 0, 100);</pre> <p>Converts the value of 15 as it appears between 0 and 50 to the equivalent value for the range 0 to 100. newVa1ue will be 30.</p>
---------	--

Electronics

This lesson introduces the *potentiometer*. A potentiometer is a resistor, much like those used to limit current for LEDs and to pull a digital input pin to +5 volts. But it is mechanically different. The resistor itself is a flat, curved surface. A slider controlled by a knob moves across this surface. Pins are connected to each end of the surface with a third pin connected to the slider. The slider, then, becomes a variable resistor.

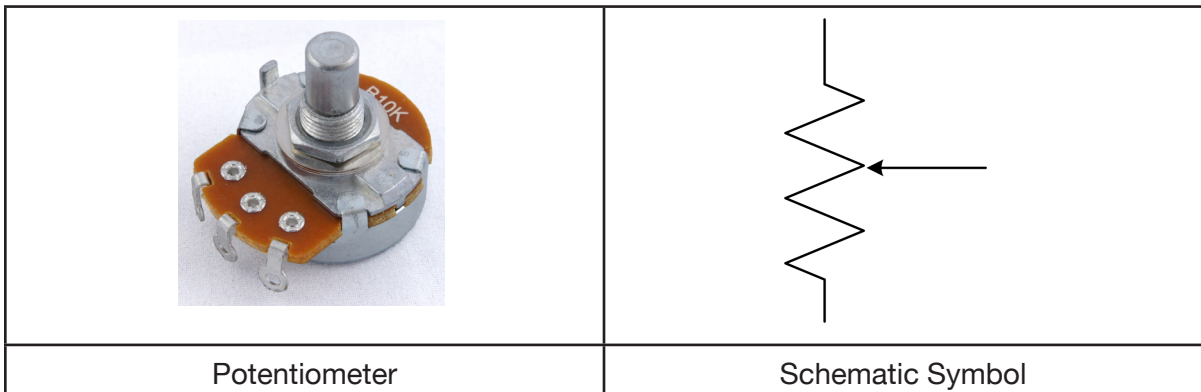


Figure 9-4. Photograph of and schematic symbol for potentiometer

As Figure 9-5 shows, the total resistance is between pins 1 and 3. The resistance between pin 1 and 2 increases as the potentiometer is turned clockwise and the wiper moves farther away from pin 1. Also, the resistance between pins 2 and 3 decreases as the wiper gets closer to pin 3.

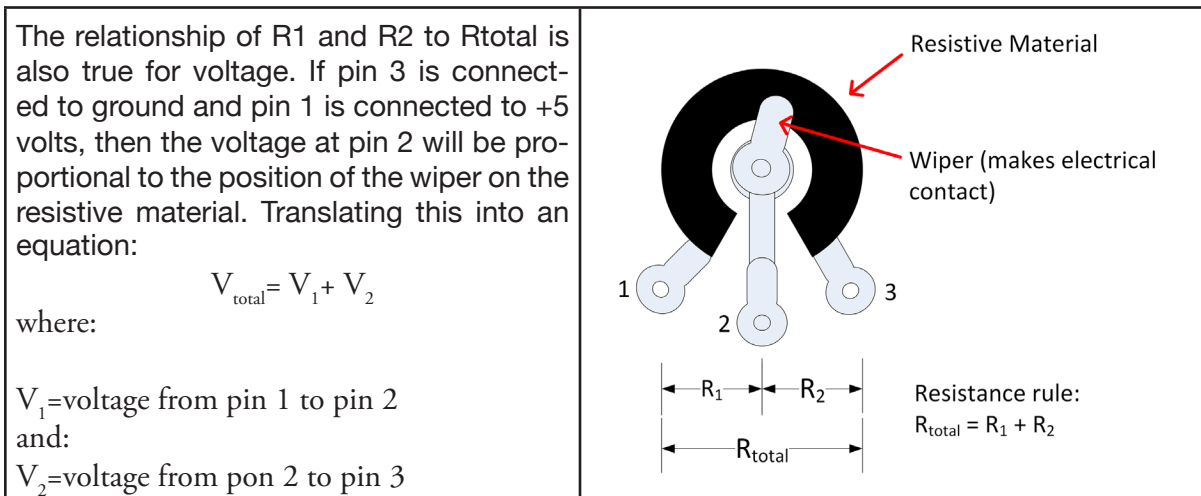


Figure 9-5. Resistance in a potentiometer

What this means for this lesson is if pin 1 is connected to +5 volts and pin 3 is connected to ground (GND), then the voltage on pin 2 can be adjusted to any value from 0 through +5 volts.





The potentiometer, in this case, is called a voltage divider. Pin 2 is an ideal input to an Arduino™ analog pin.




Goals:

By the end of this lesson readers will:

1. Be able to identify a potentiometer and know that it is merely a resistor whose value can be adjusted.
2. Write an Arduino™ sketch that can use an analog pin to read the relative position of a potentiometer and translate that into an integer between 0 and 1023.
3. Write a sketch that generates a series of pulses. Use these pulses to set the brightness level of a light-emitting diode (LED).
4. Recognize that the brightness of the LED is proportional to the width of these pulses.
5. Write a sketch that allows a user to control the brightness of an LED by turning a potentiometer.

Materials:

Quantity	Part	Image	Notes	Catalog Number
1	Arduino™ Uno		Single-board computer. This board is delicate and should be handled with care. When you are not using it, keep it in a box or plastic bag.	3102
1	USB Cable		This is a standard USB adapter cable with a flat connector on one end and a square connector on the other.	2301
1	Computer with at least one USB port and access to the Arduino™ website, http://www.arduino.cc .	---	The operating system of this computer must be Windows, Macintosh OS/X, or Linux.	---
1	Light-emitting diode (LED)		Single color, about 0.02 amps rated current, diffused	1301
1	220 ohm resistor		¼ watt, 5% tolerance. Color code: red-red-brown.	0102

Quantity	Part	Image	Notes	Catalog Number
1	Potentiometer		10k ohm.	0301
1	Bread-board		Used for prototyping.	3104
As req'd	Jumper wires		Used with bread-boards for wiring the components.	3105

Procedure:

Part 1: Explore analogRead()

In this lesson only schematics are supplied to guide wiring the circuits on the bread-board.

1. Construct the circuit shown in Figure 9-6. Connect the potentiometer to the Arduino™ using clip leads and short jumpers.

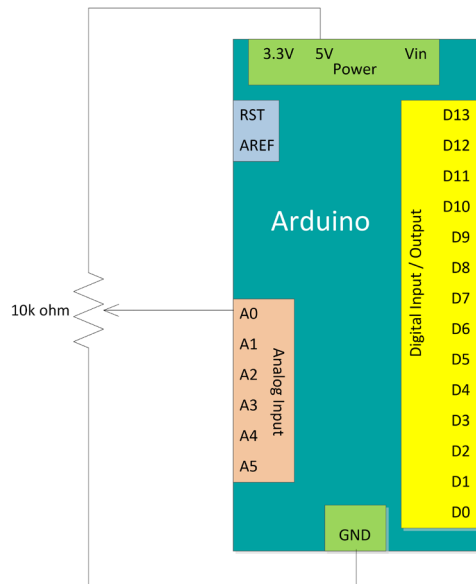


Figure 9-6. Schematic diagram of potentiometer connected to Arduino™ Uno

2. Open the Arduino™ IDE and create a new sketch. Name the sketch `Lesson9AnalogRead`. This sketch will read the value of an analog pin and display the reading on the Serial Monitor.
3. Enter the sketch as shown in Sketch 9-1.

Sketch 9-1. Complete listing for Lesson9AnalogRead

```

/* Lesson9AnalogRead
   by W. P. Osborne
   6/30/15
*/

void setup(){
  Serial.begin(9600);
}

void loop(){
  int readValue;
  readValue = analogRead(A0);
  Serial.println(readValue);
}

```


4. Upload the sketch to the Arduino™ and open the Serial Monitor. A number somewhere between 0 and 1023 should appear about 10 times per second. Adjusting the potentiometer should change this number. At one end of the potentiometer rotation the number will be 0. At the other extreme it will be 1023.
5. Map the read value to the range 0 through 255 by adding a `map` programming statement. The new `loop()` method should be as shown in Snippet 9-1.

Snippet 9-1. `loop()` method demonstrating mapping

```
...  
void loop(){  
  int readvalue;  
  readvalue = analogRead(A0);  
  
  int mappedvalue;  
  mappedvalue = map(readvalue, 0, 1023, 0, 255);  
  
  Serial.println(mappedvalue);  
}
```

The numbers displayed now should range from 0 through 255.

6. Save the sketch and close it.

Part 2: Explore `analogWrite()`

The potentiometer can remain connected to the Arduino™, but it won't be used for now.

7. Add an LED and its current-limiting resistor to the Arduino™, connecting it to pin 9. Note the short wire of the LED is going to GND. Also notice pin 9 is capable of generating pulses, as indicated by the tilde ahead of the pin number.

The schematic diagram is now as shown in Figure 9-7.

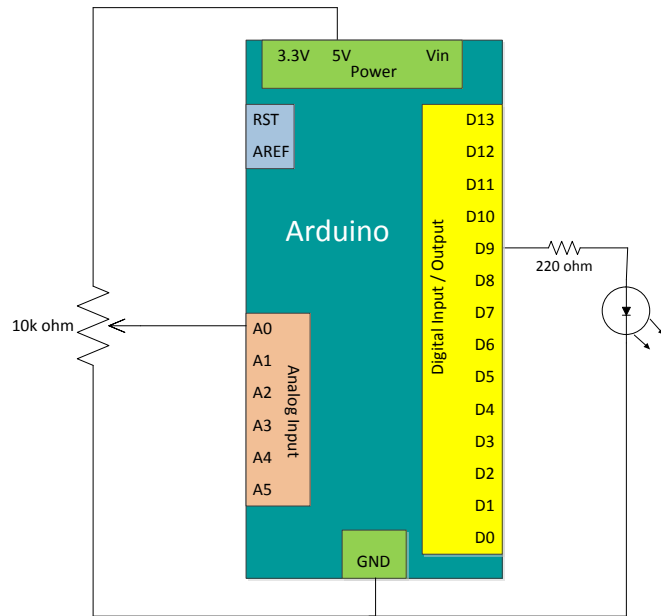


Figure 9-7. Schematic diagram of control of LED by potentiometer

8. Create another Arduino™ sketch. Name it `Lesson9Analogwrite` and enter the code as shown in Sketch 9-2.

Sketch 9-2. Complete listing for Lesson9Analogwrite

```

/* Lesson9Analogwrite
   by W. P. Osborne
   based on sample from www.arduino.cc
   6/30/15
*/

#define pinLED 9

int brightness = 0;
int fadeAmount = 5;

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

void loop(){
  // use analogWrite to set the average output
  // voltage.
  analogWrite(pinLED, brightness);
}

```

```
// change the brightness for the next loop
brightness = brightness + fadeAmount;

// If brightness is equal to or greater than
// 255 the diode is as bright as it is going
// to get because the output voltage is
// averaging 5 volts. So, start subtracting
// the fadeAmount.
//
// The same is true if the brightness is less than
// or equal to zero. Start adding the fade amount.

// The following program statements reverse the
// sign of fadeAmount by effectively multiplying
// by -1.

if(brightness >= 255 || brightness <= 0){
    fadeAmount = -fadeAmount;
}

delay(30);
}
```

Note: Sample from www.arduino.cc and modifications to the sample, as used in Sketch 9-2, used under a Creative Commons Attribution ShareAlike 3.0 license (<http://creativecommons.org/licenses/by-sa/3.0/>).

9. Save the sketch.
10. Notice that a new logical operator is used in the test of the brightness range. It is the pair of vertical lines, `||`. This operator is called **OR**. By putting it between two logical tests a new one is created. This new one evaluates to true if either or both of the logical tests is true.

Part 3: Explore map

For this part, no additional components are required. Rather, a new sketch combines `analogRead`, `analogWrite`, and `map` to use a potentiometer to set the brightness of an LED.

11. Create a new Arduino™ sketch. Name it `Lesson9AnalogReadAndWrite`.
12. Enter the sketch as shown in Sketch 9-3.

Complete listing 9-1. Using analogRead, analogwrite, and map to set brightness of an LED

```
/* Lesson9AnalogReadAndwrite
   by w. P. Osborne with some content from
   arduino.cc
   6/30/15
*/

#define pinLED 9

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

void loop(){
  int readValue = analogRead(A0);
  int brightness;
  brightness = map(readValue, 0, 1023, 0, 255);
  analogwrite(pinLED, brightness);
}
```

Note: Sample from www.arduino.cc and modifications to the sample, as used in Sketch 9-3, used under a Creative Commons Attribution ShareAlike 3.0 license (<http://creativecommons.org/licenses/by-sa/3.0/>).

13. Save the sketch then upload to the Arduino™. Turning the shaft of the potentiometer should now adjust the brightness of the LED from completely off to full brightness.

Exercise:

Exercise 9-1. Controlling two LEDs

Connect a second LED to your Arduino™. Don't forget to include the current-limiting resistor (220 ohms, red-red-brown). Write an Arduino™ sketch that increases the brightness of one while decreasing the brightness of the other as the potentiometer is turned. When the potentiometer is turned fully clockwise, only one LED will be on, and at full brightness. The other will be off. Halfway turned, both LEDs will be at half-brightness. Fully counterclockwise, the pattern will be the opposite of fully clockwise.

Name this sketch `Lesson9Exercise1`.



Analogwrite only works with Arduino™ pins marked with the tilde character (~). For the Arduino™ Uno, these are pins 3, 5, 6, 9, 10, and 11.