Analog Input in a Nutshell

- An Arduino UNO has 6 analog input pins that can be identified by integer values 0, 1, 2, 3, 4 and 5, or by the predefined constants, `A0`, `A1`, `A2`, `A3`, `A4` and `A5`.
- Analog input pins are used without prior use of `pinMode`. In other words *do not* use `pinMode` for an analog input because `pinMode` only applies to digital I/O pins.
- The following command reads the voltage level on Analog Input pin ***`pin`***.

  ```
  value = analogRead(pin)
  ```

  The result is a 10-bit integer stored in ***`value`***: $0 \leq$ ***`value`*** ≤ 1023.
- To convert an analog reading on an Arduino UNO to a voltage, multiply the reading by 5.0/1023.0. For example

  ```
  int reading;
  float voltage;

  reading = analogRead(A0);
  voltage = reading*5.0/1023.0;
  ```

  Including the decimal points in the conversion factor, i.e., using 5.0 and 1023.0 instead of 5 and 3, insures that the compiler will not precompute and store the result of an integer division of 5/1023 as zero. The following page explains why 5.0/1023.0 is better than 5.0/1024.0.

Practice with `analogRead`

Write Arduino sketches to perform the following actions. These problems build upon each other so chunks of the code can be reused. Save your solution to each problem as a separate sketch. Figure 1 has schematics of the circuits used in the programming exercises.

1. Read a photoresistor and print the raw (10-bit) reading to the Serial Monitor.
2. Read a photoresistor and turn on an LED when it is "dark".
3. Read a potentiometer and turn on an LED when the voltage is less than 1.0V and greater than 4.0V.

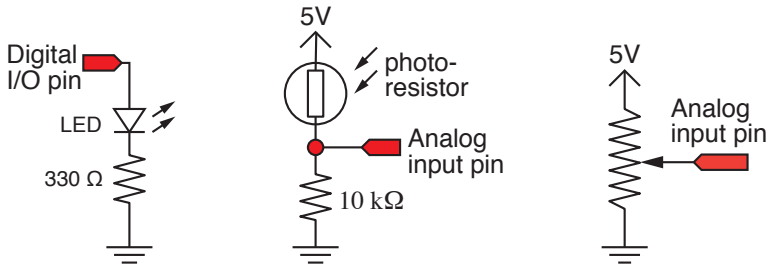


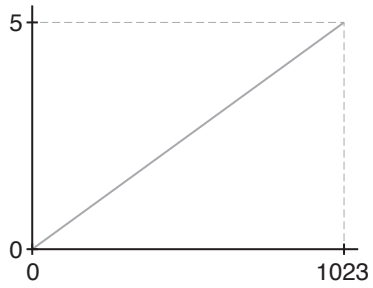
Figure 1: Circuits used for demonstration of `analogInput`.

Scaling of Arduino Analog Input Readings

The `analogInput` function on an Arduino UNO returns a value of 1023 when the input pin is at a voltage of 5V[1]. To convert a numerical value returned by the `analogRead` function to a voltage, should the scaling be 5V/1023 or 5V/1024? Answer: The correct scaling is 5V/1023.

Graphical Explanation

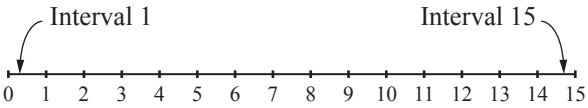
The relationship between voltage and the 10-bit scale can be visualized as the straight line in the plot to the right. The slope of the line is 5.0/1023.0 and since the line passes through $(0, 0)$, the conversion between the scales is simply the slope.



Technically, the straight line relationship is not correct because the horizontal axis only allows discrete values. Nonetheless, the plot establishes the conversion factor in a simple-to-understand image.

Analogy to a Lower Resolution Scale

An alternative explanation for using 5.0/1023.0 can be made by analogy. Suppose that the analog to digital conversion produced values on a 4-bit scale instead of a 10-bit scale. An unsigned 4-bit number can have the values 0, 1, 2, ... 15 as depicted by the following number line.



Although there are 16 numbers that label points on the line, there are only 15 equal-length segments. Therefore, the scale factor for a 4-bit analog-to-digital converter is 5V/15 because the 5V range is divided into 15 intervals. By analogy, the correct scaling for a 10-bit analog to digital converter having a maximum input range of 5V is 5V/1023.

There are 16 values on the 4-bit number line, but the first value, 0, just marks the beginning of the line. The other values, 1, 2, 3, 15 are labels for the right hand endpoints of the intervals.

Yet another way to think about this is to consider the question: How many years have you lived when you reach your nth birthday? For example, on your twenty-first birthday, you have lived 21 years. The day you were born is birthday 0.

[1] Unless the `analogReference` function is used to set another maximum voltage level for analog readings.