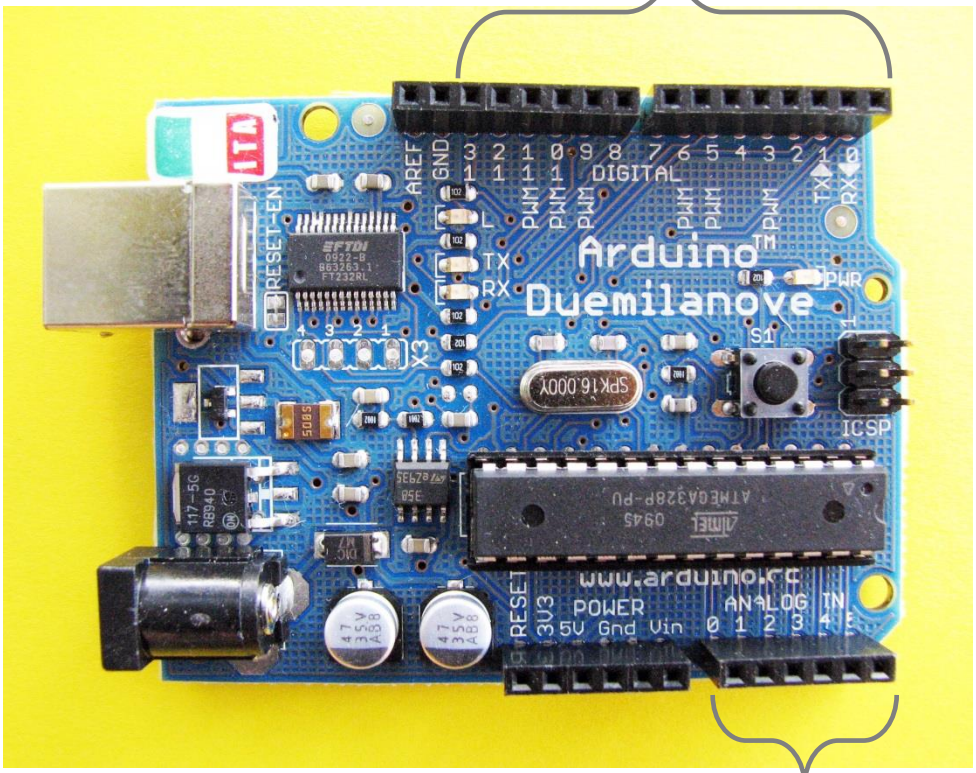
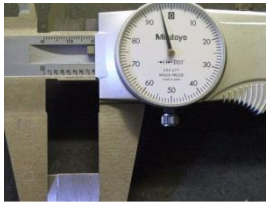
Analog and Digital Measurements

14 digital input / output pins

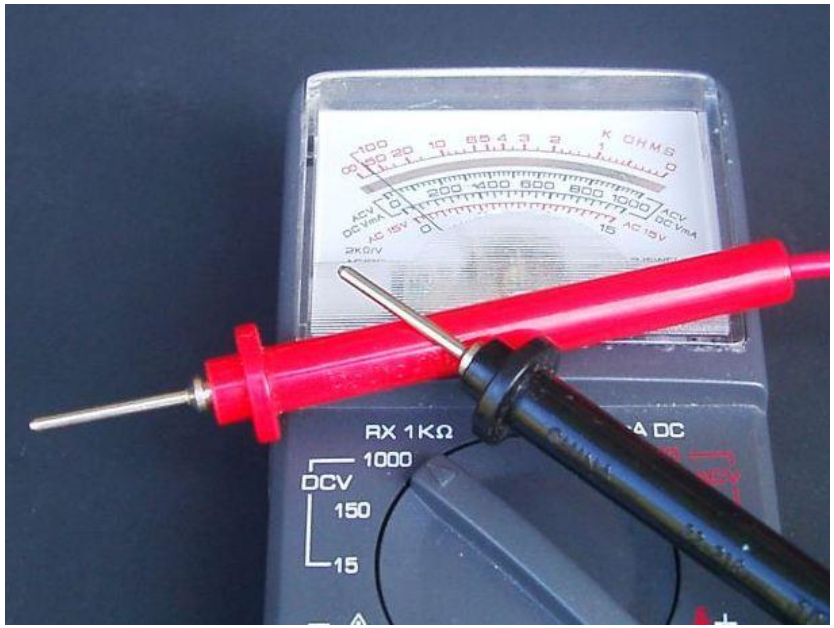


6 analog input pins



WIKIPEDIA
The Free Encyclopedia

A **digital** system is a data technology that uses discrete (discontinuous) values. By contrast, **analog** (non-digital) systems use a continuous range of values to represent information. Although digital representations are discrete, they can be used to carry either discrete information, such as numbers, letters or other individual symbols, or approximations of continuous information, such as sounds, images, and other measurements of continuous systems.

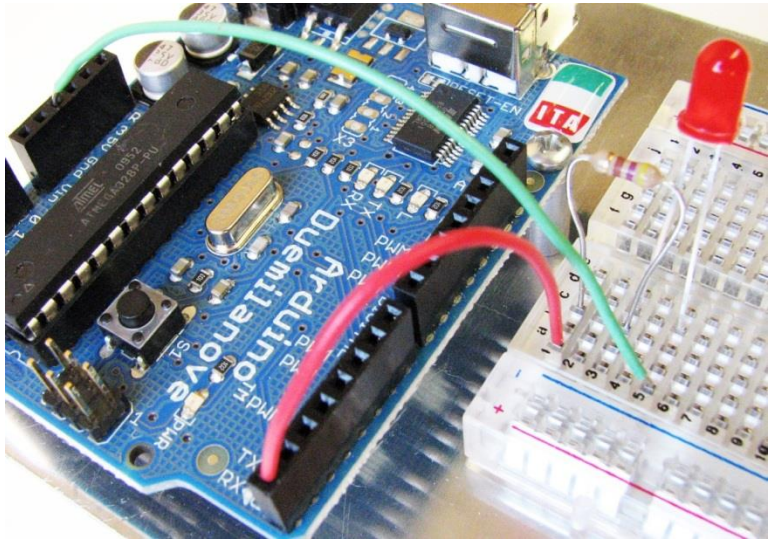


Inputs and Outputs

An **input** "receives" information or senses a voltage in the external world.
An **output** "delivers" information or makes something happen in the external world.

Below is an example from an earlier class where we made an LED flash on and off.

Are we using digital pin 0 as an input or an output? ⇨ **digital output**




```
void setup() {
  pinMode(0, OUTPUT);
}

void loop() {
  digitalWrite(0, HIGH);
  delay(1000);
  digitalWrite(0, LOW);
  delay(500);
}
```

Receiving Input from an Arduino

digital input

```
int val;

val = digitalRead(7);
```

val is either 0 or 1

- 0 = voltage sensed at digital pin 7 is LOW (0V)
- 1 = voltage senses at digital pin 7 is HIGH (5V)

analog input

```
int val;

val = analogRead(5);
```

val is an integer between 0 and 1023

- 0 = voltage sensed at analog pin 5 is zero volts
- 1024 would correspond to five volts at pin 5 (can only go to 1023 though, so a little top end range is lost)

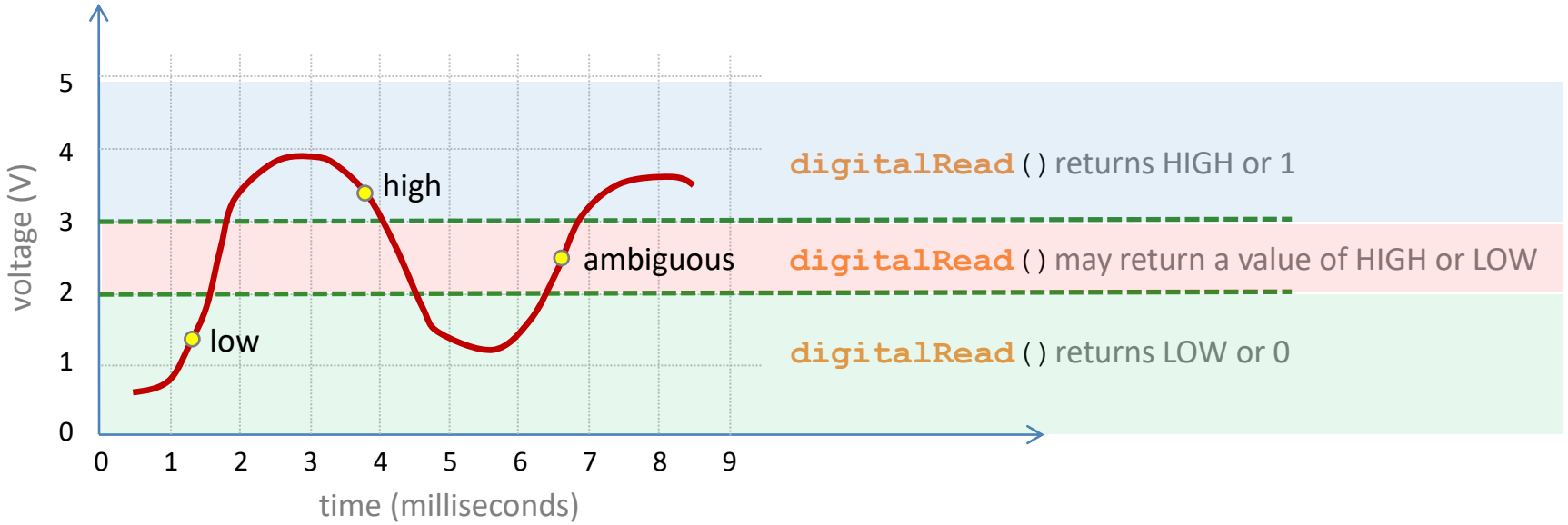
Guess what val would be if the voltage sensed at analog pin 5 was 2.5V?

⇨ **512**

Digital Inputs

The Arduino reference indicates that **`digitalRead`**() will return . . .

- a value of **HIGH** if the voltage at the digital input pin is greater than 3 volts
- a value of **LOW** if the voltage at the digital input pin is less than 2 volts.



LOW or **HIGH**???

Analog Inputs

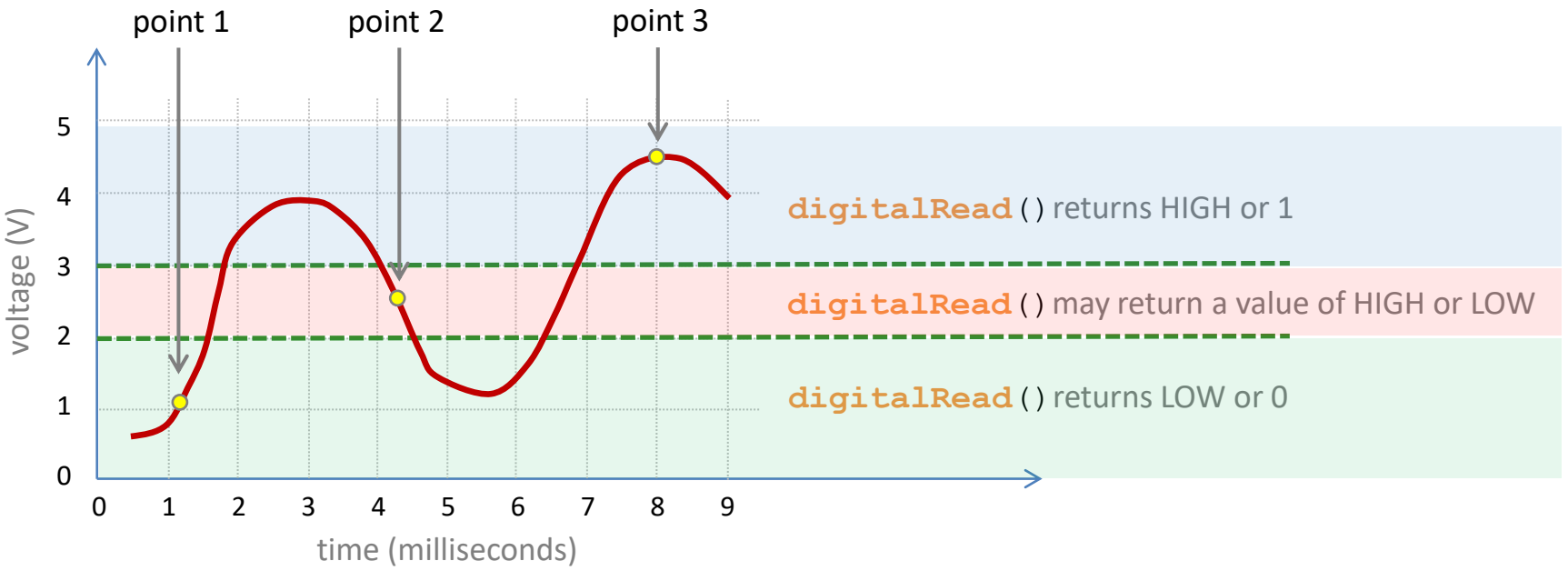
The analog input pins on your Arduino have 10-bit resolution and consequently measure in $(2)^{10}$ or 1024 increments. $(2)^{10} = 1024$

The **analogRead**() functions returns a value between 0 and 1023, where 0 is zero volts and 1024 would be 5 volts.

$$voltage = \texttt{analogRead}\, output \cdot \frac{5\ volts}{1023}$$

smallest increment of voltage that can be read = 0.00489 volts

Examples



data point from plot above	If a digital pin is used to sample voltage	if an analog pin is used to sample voltage	
	output of digitalRead()	hypothetical analogRead() output	voltage computed from analogRead() output
1	0	217	$217 \cdot \frac{5V}{1023} = 1.061V$
2	ambiguous	526	$526 \cdot \frac{5V}{1023} = 2.571V$
3	1	964	$964 \cdot \frac{5V}{1023} = 4.711V$

generalizing A/D conversion

For a simple binary converter, the digital output of an A/D converter is . . .

$$D_o = int\left[\frac{V_i - V_{rl}}{V_{ru} - V_{rl}}(2^N - 1)\right]$$

and the corresponding voltage for a given digital output is . . .

$$V_i = D_o \cdot \left(\frac{V_{ru} - V_{rl}}{2^N - 1}\right) + V_{rl}$$

where

V_i = analog input voltage
V_{ru} = upper value of input range
V_{rl} = lower end of input range
N = number of bits
D_o = digital output

quantizing error

An error exists because the A/D inversion process return an integer.

$$\text{input resolution error} = \pm 0.5 \cdot \frac{V_{ru} - V_{rl}}{2^N - 1} \text{ volts}$$

where

V_{ru} = upper value of input range
V_{rl} = lower end of input range
N = number of bits

Example Problem A 12-bit A/D converter has an input range of -10 to +10 V. Find the resolution error of the converter for the analog input.

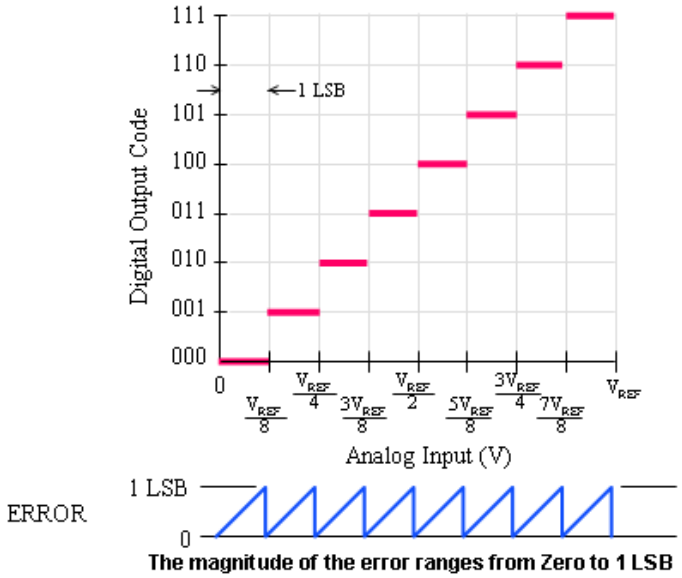
$$\text{input resolution error} = \pm 0.5 \cdot \frac{10V - (-10V)}{2^{12} - 1} = \pm 0.00244V$$

The resolution uncertainty of $\pm 0.00244V$ is the best that can be achieved.

http://www.national.com/AU/design/courses/259/

quantizing error

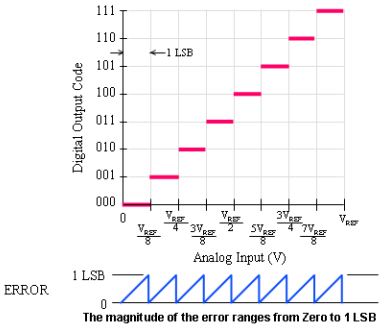
- Our simple 3-bit ADC can also illustrate **Quantization Error**.



- With an ADC input of zero, the output code is zero (000b) and there is no error.
- As the input voltage increases towards $V_{REF}/8$ ($V_{REF}/2^3$), the error increases because the input is no longer zero, but the output code remains at zero. This is because an input voltage *range* is represented by a single output code, as is necessary when interfacing between the analog and digital worlds.
- When the input reaches $V_{REF}/8$, the output code changes from 000b to 001b, where the output exactly represents the input voltage and the error reduces to zero.
- As the input voltage increases past $V_{REF}/8$, the error again increases until the input voltage reaches $V_{REF}/4$, where the error again drops to zero. This process continues through the entire input range and the error plot is a saw tooth.

http://www.national.com/AU/design/courses/259/

quantizing error



- The fact that a *range* of input voltages, or *quanta*, is represented by and converted to a single code is what we call **quantization**. The maximum error we have here with the ideal ADC is 1 LSB.
  - This range or errors is known as **quantization uncertainty** because it is the range of analog input values that could have caused a given code and when we look at the digital word we are **uncertain** as to exactly what the input voltage was that was converted to that code.
  - Since this error results from the quantization process, another name for this maximum error due to the quantization process is **quantization error**.
  - Since the ADC can only *resolve* the input into 2^n discrete values, the converter resolution is 1 in 2^n.
  - The quantization error can never be reduced below ½ LSB
  - For an 2 Volt reference (with a unity gain factor), a 3-bit converter *resolves* the input into $V_{REF}/8 = 2V/8 =$ 0.25 Volt steps. We can say that the converter "resolution" is 8 bits, or we can say, in this example, that it is 0.25 Volt. Quantization error is a round off error.
- An error of 0 to 1 LSB is not as desirable as would be an error of ±½ LSB because an error range of 0 to 1 LSB means that the converted word has a maximum error of 1 LSB from the actual input value, whereas an error of ±½ LSB means that the converted word has a maximum error is just ½ LSB from the actual input value.
  - To get the ±½ LSB error rather than the 0 to 1 LSB error, we introduce an offset into the A/D converter to force the error range to be ±½ LSB.

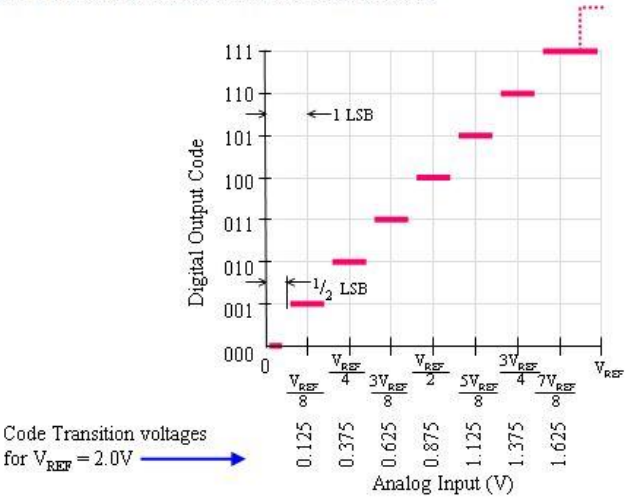
http://www.national.com/AU/design/courses/259/

offset makes error ± 1 LSB

Adding ½ LSB Offset to the ADC input

Getter a little more accuracy

- Rather than the transition from the code of zero to the first code being at 1 LSB, we add an offset of -½ LSB and the first code transition point is at an input level of ½ LSB, causing the output code to change when the input level is ½ LSB less than it would if there were no added offset.



  - The output changes from 000b to 001b with an input value of ½ LSB rather than 1 LSB and the last code transition (from 110b to 111b) is at 1.5 LSB below V_{REF}.
  - Note also that the center of the maximum output code (the "no error" point) corresponds to an input of one LSB less than the reference voltage. The implication here is that a higher resolution ADC will have a maximum output code that corresponds to an input level that is closer to the reference than would a lower resolution ADC, so better full-scale accuracy for higher resolution converters.

- The maximum error *range* we have here is 1 LSB. This -½ LSB to +½ LSB range or errors is still the same **"quantization uncertainty"** range of 1 LSB.

Adding a -½ LSB offset to the ADC input provides a little more accuracy because, in the ideal case, the maximum error is ½ LSB rather than 1 LSB.

other errors

ADCs are also subject to the following errors:

- linearity
- zero
- gain
- thermal stability
- sampling rate

ADCs only work within their intended range.