map()

- Re-maps a number from one range to another.
- That is, a value of **fromLow** would get mapped to **toLow**, a value of **fromHigh** to **toHigh**, values in-between to values in-between, etc.

Note: The "lower bounds" of either range may be larger or smaller than the "upper bounds" so the map() function may be used to reverse a range of numbers, for example

```
y = map(x, 1, 50, 50, 1);
```

The function also handles negative numbers well, so that this example:

```
y = map(x, 1, 50, 50, -100);
```

is also valid and works well.

The map() function uses integer math so will not generate fractions, when the math might indicate that it should do so. Fractional remainders are truncated, and are not rounded or averaged.

Syntax

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

Parameters

value: the number to map.

fromLow: the lower bound of the value’s current range.

fromHigh: the upper bound of the value’s current range.

toLow: the lower bound of the value’s target range.

toHigh: the upper bound of the value's target range.

Arduino Data Types

Byte - Byte stores an 8-bit numerical value without decimal points.They have a range of 0-255.

```
byte someVariable = 180;   // declares ' someVariable ' as a byte type
```

Int - Integers are the primary data type for storage of numbers without decimal points and store a 16-bit value with a range of 32, 767 to -32, 768.

```
Int someVariable = 1500;   // declares ' someVariable ' as a integer type
```

Note: Integer variables will roll over if forced past their maximum or minimum values by an assignment or comparison. For example, if x = 32767 and a subsequent statement adds 1 to x, x = x + 1 or x++, x will then rollover and equal -32, 768.

Long **-** Extended size datatype for long integers, without decimal points, stored in a 32-bit value with a range 2,147,483,647 to -2,147,483,648.

```
long someVariable = 90000; // declares ' someVariable ' as a long type
```

Float **-** A data type for floating-point numbers, or numbers that have a decimal point. Floating-point numbers have greater resolution than integers and are stored as a 32-bit value with a range of 3.4028235E+38 to -3.4028235E+38.

```
float someVariable = 3.14  // declares ' someVariable ' as a floating-point type
```

Note: Floating-point numbers are not exact, and may yield strange results when compared. Floating point math is also much slower than integer math in performing calculations, so should be avoided if possible.