Arduino and Servo Motor Tutorial
By: Matthew Jourden
Brighton High School

The following program will also use a function to help consolidate the code that will be used to turn the ultrasonic sensor on/off and convert the sound wave value to measureable unit in inches

# Functions

Functions "Encapsulate" a task (they combine many instructions into a single line of code). Most programming languages provide many built in functions that would otherwise require many steps to accomplish, for example computing the square root of a number. In general, we don't care **how** a function does what it does, only that it "does it"!

When a function is "called" the program "leaves" the current section of code and begins to execute the first line inside the function. Thus the function "flow of control" is:

1. The program comes to a line of code containing a "function call".
2. The program enters the function (starts at the first line in the function code).
3. **All instructions** inside of the function are executed from top to bottom.
4. The program leaves the function **and goes back to where it started from.**
5. Any data computed and **RETURNED** by the function is used in place of the function in the original line of code.

---

## Why do we Write Functions?

1. They allow us to conceive of our program as a bunch of sub-steps. (Each sub-step can be its own function. When any program seems too hard, just break the overall program into sub-steps!)

2. They allow us to reuse code instead of rewriting it.

3. Functions allow us to keep our variable namespace clean (local variables only "live" as long as the function does). In other words, function_1 can use a variable called i, and function_2 can also use a variable called i and there is no confusion. Each variable i only exists when the computer is executing the given function.

4. Functions allow us to test small parts of our program in isolation from the rest. This is especially true in interpreted langaues, such as Matlab, but can be useful in C, Java, ActionScript, etc.

## Why do we Write Functions?

1. They allow us to conceive of our program as a bunch of sub-steps. (Each sub-step can be its own function. When any program seems too hard, just break the overall program into sub-steps!)

2. They allow us to reuse code instead of rewriting it.

3. Functions allow us to keep our variable namespace clean (local variables only "live" as long as the function does). In other words, function_1 can use a variable called i, and function_2 can also use a variable called i and there is no confusion. Each variable i only exists when the computer is executing the given function.

4. Functions allow us to test small parts of our program in isolation from the rest. This is especially true in interpreted langaues, such as Matlab, but can be useful in C, Java, ActionScript, etc.

---

## Steps to Writing a Function

1. Understand the purpose of the function.
2. Define the data that comes into the function from the caller (in the form of parameters)!
3. Define what data variables are needed inside the function to accomplish its goal.
4. Decide on the set of steps that the program will use to accomplish this goal. (The Algorithm)

Arduino and Servo Motor Tutorial
By: Matthew Jourden
Brighton High School

## Parts of a "black box" (i.e., a function)

Functions can be called "black boxes" because we don't need to know **how** they work. Just what is supposed to *go* into them, and what is supposed to *come* out of them.

When defining a program as a black box, we must describe the following attributes of the function.

***Note:*** *most documentation systems are just this, the attributes of a function with no code associated with it.*

1. The Name - describes the purpose of the function. Usually a verb or phrase, such as "compute_Average", or just "average".
2. The Inputs - called parameters. Describe what data is necessary for the function to work and gives each piece of data a **Symbolic Name** for use in the function.
3. The Calculation - varies for each function
4. The Output - Usually one (but sometimes zero or sometimes many) values that are calculated inside the function and "returned" via the output variables.

---

## Function Workspace

Every function has its own **Workspace**. This means that every variable inside the function is only **usable** during the execution of the function (and then the variables go away).

Having a separate "workspace" for each function is critical to proper software engineering. If every function shared every variable in an entire program, it would be easy to inadvertently change the values of variables that you shouldn't. Further, it would be hard to remember what "names" have been used elsewhere, and coming up with new names to represent similar ideas would be challenging.

A side-effect of function variables not existing after the end of the function is that the only way to get information "out" of a function is by "returning" that information via the output of the function.

Additionally, the function can only "see" the information that is "passed" to it via parameters. Thus the only way information can get "in" to the function is by using parameters.

***Note:*** *In certain object oriented languages (e.g., C++, Java, ActionScript), a function can also see all of the variables associated with its containing object.*

Tutorial Program Function:

```
void setup() {

inches = ultrasonic (x);
}

long ultrasonic (long microseconds)
{
  Ultrasonic Sensor Code

  return microseconds
}
```

Variable x is passed to Function ultrasonic

Microseconds is returned back to Void Setup where the function was called and overwrite the variable x to what microseconds equals