Arduino Programming :

Functions

Objectives

- Understand what a function is
- Understand why functions are important
- Understand how functions are implemented

What is a function?

"A function is a block of organized, reusable code that is used to perform a single, related action." *

Two basic types of functions:

- Built in: for example, delay() or setup()
- User defined: We write our own!

*https://www.tutorialspoint.com/computer_programming/computer_programming_functions.htm

Purpose of functions

Functions are used to do repeated specific tasks.

Example: Let's say in a program we want to flash an LED five times at different points in the program.

Every time we want this to happen we would have to write several lines of code to do this….

Flash LED five times code

```
for (x = 0; x<5; x++)
{
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
}
```

We have to copy the same code over and over…

```
// Do something

for (x = 0; x<5; x++)
{
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
}

// Do something else…

for (x = 0; x<5; x++)
{
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
}

// Do something else…

for (x = 0; x<5; x++)
{
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
}
```

Not only does this make the code long, it takes up more memory!

Instead we can create a function…

```
void flashFive (void)
{
  for (int x = 0; x<5; x++)
  {
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
  }
}
```

Name of function

Body of function

Curly braces define beginning and end of function

How do we use the function? Look at previous code…

```
// Do something
flashFive();                    // “Call” the function

// Do something else…
flashFive();

// Do something else…
flashFive();
```

The function is outside of loop!

```
void loop()
{
 delay(1000);
 flashFive();
 delay(1000);
}

void flashFive(void)
{
  for (int x = 0; x < 5; x++)
  {
    digitalWrite (LED, HIGH);
    delay(300);
    digitalWrite (LED, LOW);
    delay(300);
  }
}
```

So what does void mean??

```
void flashFive (void)
{
  for (int x = 0; x<5; x++)
  {
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
  }
}
```

Nothing is sent or "passed" to the function

Nothing is returned from the function

Sending a value to a function

```
void flashLED (int numFlashes)
{
  for (int x = 0; x<numFlashes; x++)
  {
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
  }
}
```

The variable numFlashes is "passed" to the function

Note that it's type is defined also

numFlashes is used in the function

Sending and receiving values to and from a function

Name of function

The variable *value* is "passed" to the function

```
int squareVal (int value)
{
int result;
   result = value * value;
   return result;
}
```

int defines the type of value the function will return

The variable *result* is returned to the main program

Defining internal variables

```
int squareVal (int value)
{
int result;
   result = value * value;
   return result;
}
```

The variable value is *local* to the function – it only exists while the function is active

The variable result is *local* to the function – it only exists while the function is active

These local variables are only accessible to the function, and NOT to loop. This is called *encapsulation*.

Example of using a function

```
int answer;
void loop()
{
   answer = squareVal(4);          // calls function squareVal
}


int squareVal (int value)
{
   int result;
   result = value * value;
   return result;
}
```