// Advanced Section

This section will discuss a few things that we didn't cover in the rest of the binder; more advanced programming concepts, additional data types and data structures, powering your projects, and interfacing with the Processing environment.

Arrays: If variables can be thought of as buckets that hold a single piece of information, then arrays can be thought of as a collection of buckets, or a big bucket with a lot of little buckets inside. Arrays are extremely useful for a lot of different programs – basically, any time you want to perform a similar operation on several variables (of the same type) – you should consider putting the variables in an array. For example, if I want to blink eight LED's at the same time, I could put them in an array, and then use a for loop to iterate over the array, like so:

```
/* this is the array that holds the pin numbers our LED's
would be connected to */
int ledPins[] = {2,3,4,5,6,7,8,9};

// in setup( )we can set all the pins to output with a simple
// for loop
// 8, because we have 8 elements in the array
for( int i = 0; i < 8; i++) {
// sets each ledPin in our array to OUTPUT
 pinMode(ledPins[i], OUTPUT);
}
```

The ledPins[i] part is important; it allows us to reference each element in our array by its place in the array, starting with 0 (which can be confusing). So, in our example above ledPins[0] == 2, since 2 is the 1st element we put into the array. This means that ledPins[1] == 3, and ledPins[7] == 9, and is the last element in our array. If this doesn't make sense, don't worry.

See http://arduino.cc/en/Reference/Array for further explanation.

Float: Data type for floating point numbers (those with a decimal point). They can range from 3.4028235E+38 down to -3.4028235E+38. Stored as 32 bits (4 bytes). A word of advice: floating point arithmetic is notoriously unpredictable (e.g. 5.0 / 2.0 may not always come out to 2.5), and much slower than integer operations, so use with caution. Some readers may be familiar with the 'Double' data type – currently, the Arduino implementation of Double is exactly the same as Float, so if you're importing code that uses doubles make sure the implied functionality is compatible with floats.

Long: Data type for larger numbers, from -2,147,483,648 to 2,147,483,647, and store 32 bits (4 bytes) of information.

String: On the Arduino, there are really two kinds of strings: strings (with a lower case 's') can be created as an array of characters (of type *char*). String (with a capital 'S'), is a String type object. The difference is illustrated in code:

```
Char stringArray[10] = “SparkFun”;

String stringObject = String(“SparkFun”);
```

The advantage of the second method (using the String object) is that it allows you to use a number of built-in methods, such as length(), replace(), and equals().
More methods can be found here: http://arduino.cc/en/Reference/StringObject

Increment (++): Increment is an easy way to tell a number variable to add 1 to itself. So instead of writing *variable = variable + 1;* all you have to write is *variable++;*. It is commonly used in for loops like so:

```
for(int i = 0; i < 10; i++) {
//will increment i by 1 each time through the loop
}
```

Decrement (--): Basically the inverse of increment. Writing *variable--;* is the same as writing *variable = variable - 1;*
Compound Notation: Compound Notation is similar to increment and decrement in the sense that it provides a shorter way of performing arithmetic on a variable. Compound Notation can be used with addition(+=), subtraction(-=), multiplication(*=), and division(/=). For example:

```
float f = 10;
f += 10; // f now equals 20
f -= 5   // f now equals 15
f *=2;   // f now equals 30
f /=3;   // f now equals 10
```

Other Useful Arduino Functions:

delay(): The delay() function is very useful in programs where you want (you guessed it) a delay between actions (such as an LED blinking on and off). delay() takes its argument in milliseconds, so delay(1000); would be a 1 second delay.

millis(): The millis() function returns the number of milliseconds since the program started running (up to about 50 days, at which point it resets to 0).

Name:

Date:

This can be useful if you sketch requires a timer or reset. For example, if I needed to blink an LED every five minutes, I could write:

```
if (millis() % 300000 == 0) {
 digitialWrite(ledPin, HIGH);
 delay(1000);
 digitalWrite(ledPin, LOW);
}
```

random(): The random() function will give you a pseudo-randomly generated number, with either a maximum number, or a maximum and minimum. So, if I wanted random numbers between 1 and 100, I would write:

```
int randomNumber = random(1,100);
```