Name:
Date:

9

Advanced Section

Name:
Date:

// 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);
```

Name:
Date:

// Useful Libraries

Libraries are a great and easy way to extend the functionality of your Arduino. They're basically collections of code that you can import into your sketch by going to 'Sketch'-> 'Import Library'. The Arduino software ships with several useful Libraries, some of which are discussed below. The full list of available libraries and documentation can be found at: http://arduino.cc/en/Reference/Libraries.

EEPROM: EEPROM stands for "Electrically Erasable Programmable Read-Only Memory", but you can think of it as the Arduino's permanent storage unit. Basically, if you want to keep track of something (say, sensor values) you can read or write them to the EEPROM, using the EEPROM library and the read() and write() functions. A complete example of reading in an analog sensor value and writing it to EEPROM can be found online at: http://arduino.cc/en/Tutorial/EEPROMWrite.

Ethernet: The Ethernet library, along with an Arduino Ethernet Shield, can enable your project to connect to the internet, and allows your project to act as either a server (accepting incoming connections) or a client (making outgoing connections). Check out the Arduino tutorial homepage under 'Ethernet Library' (http://www.arduino.cc/en/Tutorial/HomePage) for some code examples. You can find the Ethernet shield here: http://www.sparkfun.com/products/9026.

LiquidCrystal: This library is for controlling Liquid Crystal Displays (LCD's). It is designed for most text-based LCD's which use the Hitachi HD44780 chipset and driver – if that's greek to you, don't worry, there are many compatible screens out there. The basic wiring and code to get started are both pretty easy, and can be found here: http://arduino.cc/en/Tutorial/LiquidCrystal.

Stepper: Stepper motors are a type of motor that rotates continuously in small, precise steps. For this reason it makes a very useful component in any project that requires precise or continuous motion. The stepper library, in combination with a stepper motor board to control the motor (most steppers use more energy than the Arduino can provide) makes using stepper motors super easy. Stepper Motors typically look something like this:

Stepper motors can be used pretty easily with a stepper motor driver like the EasyDriver, which you can find here (along with schematics and example code): http://www.sparkfun.com/products/10267.
Always remember to check the datasheet for any component you plan on using so you don't burn out your board or the motor.

Name:
Date:

// Processing

Processing is a free, open-source programming language and environment created by Casey Reas and Ben Fry at the MIT Media Lab. Processing and Arduino are, in fact, built off some of the same code, and look very similar in their programming environments. However, while Arduino is built for interfacing with the physical Arduino boards, Processing specializes in visual-based programming on the computer. What's exciting about this is that Processing can read in values from Arduino on the Serial port and then manipulate visuals based on what the Arduino is doing. For example, you can make the turning of a potentiometer change the color of the screen in Processing. In order to do this, you need both Processing and Arduino set up properly.

You can download Processing from: http://processing.org/download/ (there are pretty good install instructions on the site, as well as a bunch of example code and references).

Now that you're all set with Processing and Arduino, let's set up a simple example where we read in the analog values from a potentiometer and change the background color in Processing in response.

First, set up Circuit #2 on your breadboard (we won't be using the LED though, so you can leave that out if you like). Next, open Arduino and type this code:

```
/*
AnalogReadSerial
Reads an analog input on pin 0, prints the result to the
serial monitor
This example code is in the public domain.
*/

void setup() {
 Serial.begin(9600);
}

void loop() {
 int sensorValue = analogRead(A0);
 Serial.print(sensorValue/4, BYTE);
}
```

Upload it to your Arduino board. We send the sensor value in byte form divided by four because we want Processing to be able to read in the values as integers from 0-255 (the full greyscale color range).

Now, open up Processing, and type in the following:

```
/* Analog Read from RedBoard
 Change Background color based on analog sensor value
 This code is in the Public Domain
*/

import processing.serial.*; //import serial library

Serial myPort; //get a Serial object

void setup() {
 println(Serial.list()); //print out the available ports
 //pick the first port in our list
 myPort = new Serial(this, Serial.list()[0], 9600);
 size(600,600); //set the canvas size
}

void draw() {
}

void serialEvent(Serial myPort) {
//when the serial port gets a byte do the code below

 int in = myPort.read(); //read in a byte from the port
 background(in); //set the background color to the value
 println(in); //print it out to Processing's monitor
}
```

Launch the Processing sketch. Voila! Turning the pot should fade the color of the screen from white to black. If you don't understand all of the code, don't worry. Processing was built with new users in mind. There's great documentation on the Processing site:
http://processing.org
as well as a bunch of resources listed in the references section. Remember, this is just the tip of the iceberg; the only limit of what you can do is your imagination!

Name:
Date:

// Further References

Arduino:

Main site: http://arduino.cc

Reference (for functions): http://arduino.cc/en/Reference/HomePage

Arduino's Getting Started Guide: http://arduino.cc/en/Guide/HomePage

Examples: http://arduino.cc/en/Tutorial/HomePage

Tutorials: http://www.arduino.cc/playground/Learning/Tutorials

Forum: http://arduino.cc/forum/

DIY Electronics Tutorials:

SparkFun: http://www.sparkfun.com/tutorials

Adafruit: http://www.adafruit.com/index.php?main_page=tutorials

LadyAda: http://www.ladyada.net/learn/arduino/

Instructables:
http://www.instructables.com/tag/type-id/category-technology/channel-arduino/

Tom Igoe's Physical Computing Pages: http://tigoe.net/pcomp/index.shtml

Fritzing & DIY Circuit Board Production:

Main Fritzing site: http://fritzing.org/

Learning Fritzing: http://fritzing.org/learning/

Fritzing Part Libraries: http://fritzing.org/parts/

PCB Production: http://fritzing.org/learning/tutorials/pcb-production-tutorials/

BatchPCB: http://batchpcb.com/index.php/Products

Search on Instructables.com for 'Circuit Board' or 'PCB' – there are dozens of tutorials, one of which will probably suit your particular situation.

Name:
Date:

Books:

SparkFun's List: http://www.sparkfun.com/categories/176

Electrical Engineering 101: http://www.sparkfun.com/products/9458

Getting Started With Arduino: http://www.sparkfun.com/products/9301

Make: Electronics: http://www.sparkfun.com/products/9600

Making Things Move: http://www.sparkfun.com/products/10394

Making Thing Talk: http://www.sparkfun.com/products/9300

Processing:

Processing Website: http://processing.org

Open Processing: http://openprocessing.org

Learning Processing: http://learningprocessing.com

Creative Applications: http://www.creativeapplications.net/category/processing/

Other Stuff:

Illustrated Guide to Soldering ("Soldering is Easy: Here's How to do it"):
http://mightyohm.com/files/soldercomic/FullSolderComic_20110409.pdf

Evil Mad Science (blog and store): http://www.evilmadscientist.com/

Make: Magazine online (also has projects and videos): http://makezine.com/

Making Things Move Companion Website: http://www.makingthingsmove.com/

SparkFun's Education Website: http://www.learn.sparkfun.com