Debugging Your Code

If you have programmed before, you have debugged code. Your first draft of code will almost always have errors, whether they are syntax, computational, logical, or something else. The Arduino IDE will not compile if you have syntax errors in your code, so Arduino will locate those bugs for you. To make sure the rest of your code works correctly, however, you will have to locate the errors yourself.

As you write your code and test it, you will naturally find bugs and figure out ways to fix them. For example, if the temperature youre displaying isnt what youre expecting, think about possible places that the bug could have occurred. Maybe the temperature reading you got from the thermocouple was inaccurate. Maybe it wasnt stored or loaded correctly from memory. Maybe the calculations you performed on it messed up the value. Maybe your program is skipping a part of the code. Maybe you just made a typo and are displaying the wrong variable. When you realize that something is wrong, trace how that variable changes as you move through your code.

So how do you figure out the value of a variable at a certain point in your code? This tutorial provides a few tips as you try to locate your errors. The Arduino IDE is very lightweight and thus does not have a dedicated debug tool that a more complicated IDE such as Eclipse or Visual Studio might have, so were going to have to do it manually.

Serial Communication

The most common way you will probably debug is using serial communication. This will allow you to print out variables to the serial monitor to see if they are what you expect. To set up serial communication (communication between the Arduino and your computer), initialize it in the void setup() function:

```
Serial.begin(9600)
```

9600 is the **baud rate**. It is the number of data packets sent to your computer per second. Now that you have initialized it, you can use the following command to print strings and variables.

```
Serial.print(Temperature: ); //print a string
Serial.print(x); //print a variable
```

If you want to print something on its own line, you can use:

```
Serial.println(x);
```

Play around with the serial communication to see what you can print. You can stick these statements where you want to check the value of a variable and simply run the Arduino while connected to your computer, open the Serial Monitor, and you can view the print out.

LCD

At times you might want to debug how the LCD is working. Simply print values to the LCD-refer to the LCD tutorial.

Toggling Debug Code

You will often find that you are adding debug code, removing it, only to find that you need it again a few days later. Of course, you can simply comment it in and out every time. A cool trick you can do to quickly

toggle between your debug modes is to use define statements. Add something to this effect to the beginning of your program.

```
#define DEBUG 1
#ifdef DEBUG
  #define DEBUG_PRINT(x)  Serial.println(x)
#else
  #define DEBUG_PRINT(x)
#endif
```

Remember that `#define`will replace all instance of `DEBUG_PRINT(x)` with whatever is behind it (in this case either a Serial print statement or simply nothing). Simply add `DEBUG_PRINT(Temperature)` to your code and you can toggle all of them on or off simply by changing the value of `DEBUG` to 0 or 1.