**Dustin Maclay**

**Talen Phillips**

**Due date 3/5/2013**

**Experiment number 6**

**Lab title: Serial Monitor Input and Output**

**Date of the lab 2/27/2013**

**Source Code:**

/\* Talen Phillips and Dustin Maclay

\* EE107-01

\* 28FEB2013

\* "Arduino calculator"

\*/

int in = 0; **// input from incoming serial data**

int op = 0; **// operation variable**

int tot = 0; **// running total**

void setup()

{

Serial.begin(9600); // opens serial port, sets data rate

}

void loop()

{

**// send data only when you receive data:**

while (!Serial.available());

**// read the incoming byte:**

in = Serial.read();

while (in != '$') **// Exit the following loop when a $ is entered.**

{

if (in == '+') op='+'; **//set the operator**

if (in == '-') op='-';

**// If a number is seen, execute the following code:**

if (('9' >= in) && (in >= '0'))

{

**// If the operator is** '**-**'**, subtract the**

**// value of the number from the total:**

if (op == '-') tot -= (in-'0');

**// NOTE: in-'0' offsets the ASCII values coming**

**// through the serial bus**

**// otherwise add to the total.**

else tot += (in-'0');

**// NOTE: this defaults to addition.**

op = 0;

}

**// read the next serial input before reevaluating the loop:**

in = Serial.read();

}

**// print the total:**

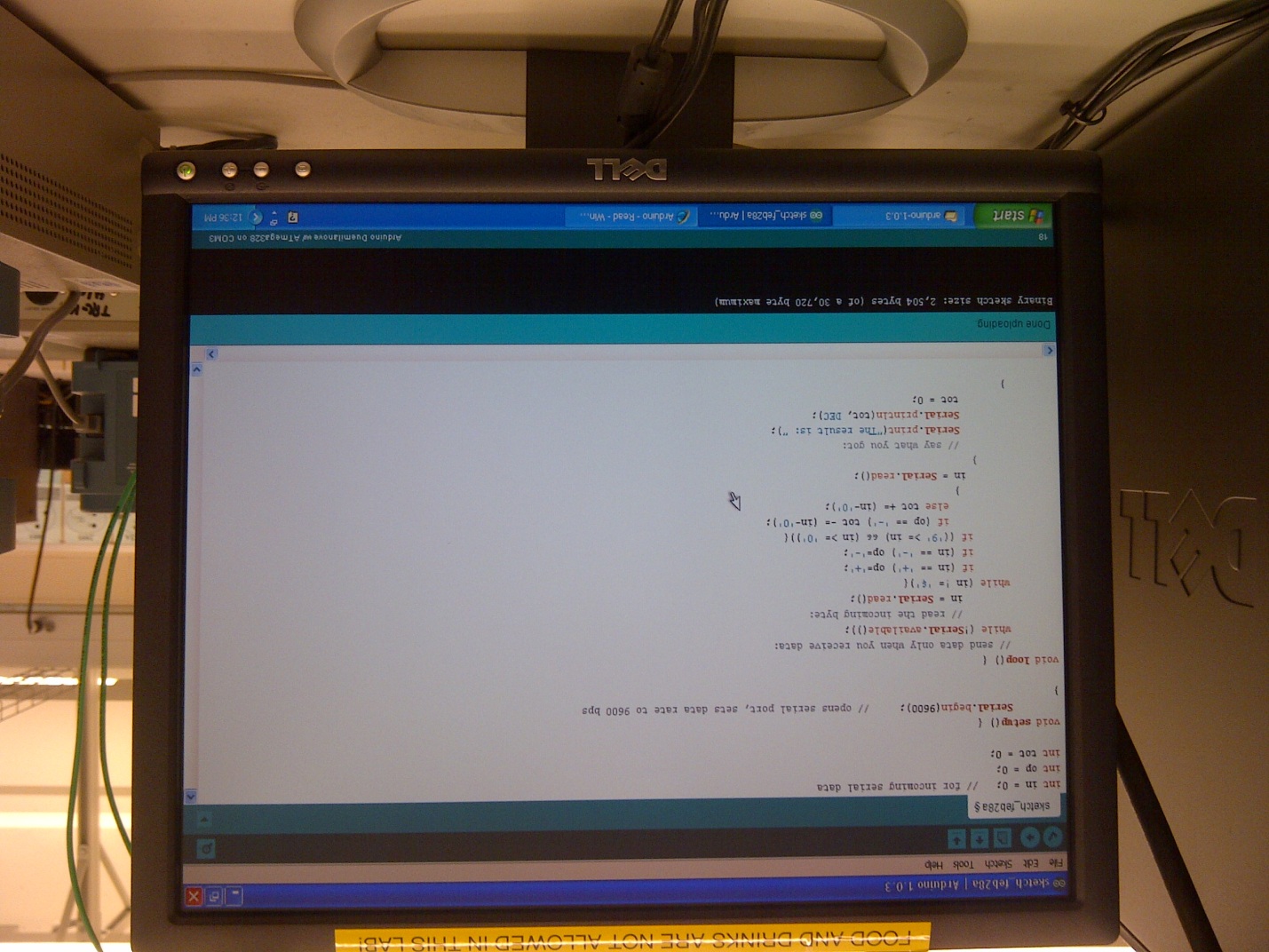
Serial.print("The result is: ");

Serial.println(tot, DEC);

**// reset variables:**

tot = 0;

}



**Please note:** Our screenshot with the serial monitor was lost. We also took this picture of the system out of habit. The only difference between this picture and the screenshot (other than the serial monitor) is the statement (op = 0;), which prevents subtraction operations from carrying over.

**Summary:**

In this lab students were asked to build a program that reads an equation in the serial monitor and uses the Arduino board as a calculator using only positive integers 0-9, “+”, “-“, and the “$” sign. Where the “$” sign signifies the termination of the equation.

Within the program was a while loop that continued as long as the output of the Serial.available function was false. The loop had no statements, and only served to prevent the program from continuing if there was no input from the serial monitor.

If the function returned false, the program would read the input, and use another while loop to evaluate it. That loop executed as long as the input was not $, and used a series of if statements to evaluate the character, and either alter the op (operator) variable (if it was a + or – symbol) or add/subtract the digit (if it was a number) from the tot (total) variable. The loop would then read the following character from the serial input, and restarted the loop.

If the Serial.read function ever returned a dollar sign, the loop would end, and the remaining code would print the final value of tot (total), reset tot, and return to the top of the void loop() function.

Students found that if we input a single digit positive integer that the output was the numeric equivalent of the ASCII character. In order to have a true equation, we had to subtract 48 (the ASCII equivalent of ’0’) from the output from the serial monitor.

Also note that the default operation for this program is addition. This is an arbitrary choice, but it seems necessary given the parameters of the lab exercise. It may be possible to use multi-digit numbers by making the default operation “multiply [a subtotal] by 10 and add the new digit”.

The code could also be easily modified to include additional operations (including, but not limited to ‘\*’ and ‘/’), by including additional if statements within the second while loop. For each additional function, one if statement would be needed to set the op (operation) variable, and one if statement would be needed to execute the operation itself when a number is received.