How to Succeed With This Lab

Before executing the procedure, skim-read through the whole lab handout.

Skim-read through the **Write Up** section so you know what is expected for the lab report. Work together as a team with your lab partners to solve problems and overcome any snags (e.g. syntax issues with C, fabrication of the sensor leads, writing up an explanation of results).

The number one problem students face doing this lab:

Incorrectly wiring the sensor/ not correctly reading the datasheet pinout. Not sure? Ask. On the datasheet, the sensor pins come out of the page towards you.

Objectives

1) Become familiar with the Arduino integrated development environment (IDE)
2) Become familiar with the Arduino hardware's input and output capabilities: `analogRead()`, `analogWrite()`, `Serial.println()`, etc.
3) Use the Arduino to read a temperature signal from the LM61 temperature sensor and display the sensor reading to serial terminal window on the host PC.
4) Use the serial monitor to collect data and plot the data with Excel.
5) Setup a interrupt routine to provide a regular sampling interval of 100 milliseconds, using MsTimer2.h.
6) Practice modifying the source code, uploading and debugging.

Reading

http://Arduino.cc

Intended Learning Outcomes:

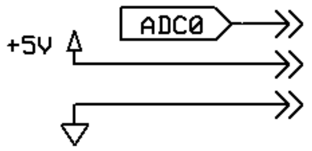
Utilize Arduino environment to edit, compile, load and run a program.	
Demonstrate understanding of how to use Arduino serial terminal monitor window to debug a program, display data and copy that data to Excel for post processing.	
Demonstrate use of software interrupts to establish a regular sampling interval.	

Overview of Arduino Uno Microprocessor

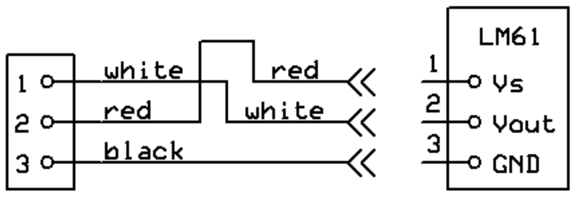
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Procedure

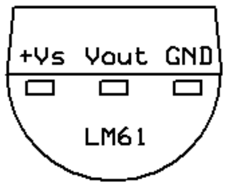
1) Setup the following items in the lab.
   a. Arduino Uno microprocessor board. **IF YOU HAVE THE RED YOURDUINO VERSION, SET THE OPERATING VOLTAGE SWITCH TO 5 VOLTS.**
   b. USB cable from Uno to host PC.
   c. Solder a three pin header to the TO-92 LM61 temperature sensor pins
   d. Plug the loose ends of the three wire cable harness onto the sensor header pins. Observe color code: +Vs – red, Vout – white, and GND – black. Getting this right will save you many hours in later labs.



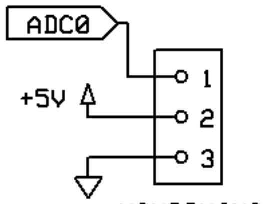
Protoboard mapping to Uno headers



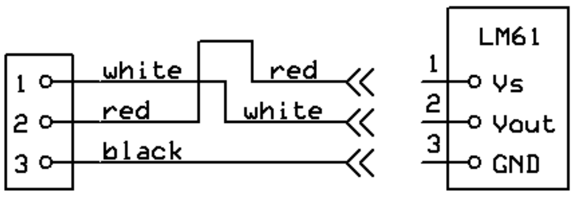
Cable header wires reordered to LM61 as shown.
ARDUINO UNO SEPARATE HEADER PINS



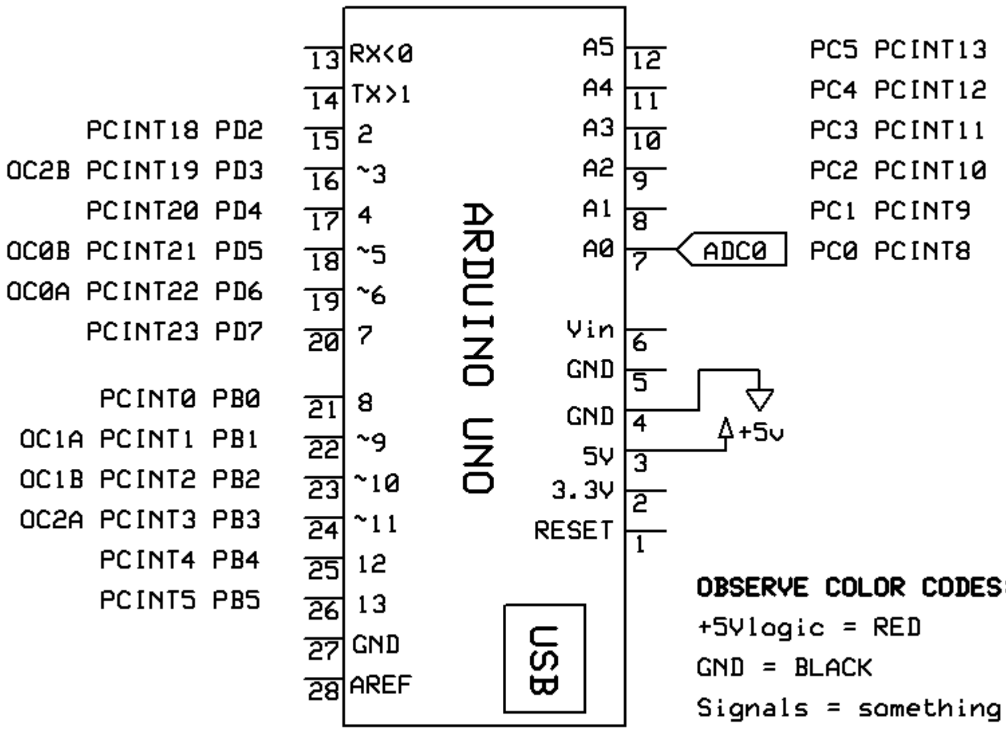
BOTTOM VIEW



YOURDUINO HEADER

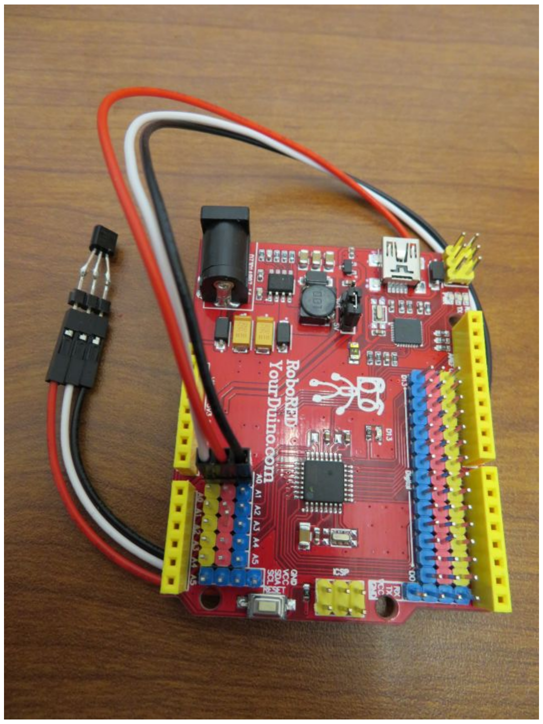


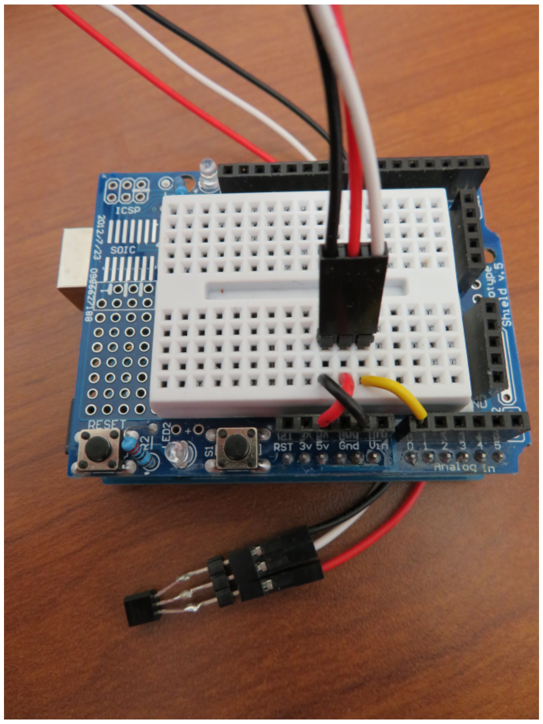
Cable header wires reordered to LM61 as shown.
YOURDUINO UNO INTEGRATED HEADER PINS



OBSERVE COLOR CODES:
+5Vlogic = RED
GND = BLACK
Signals = something else

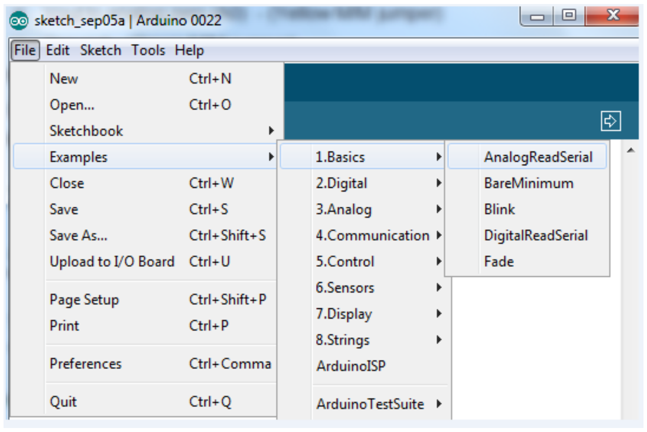
e. Using jumpers from your kit and the protoboard, jumper wires from the LM61 cable to +5V, ground and analog input zero (A0) on the microprocessor board. If you are using a Yourduino Uno (red board) the free end of the sensor cable may be plugged directly into the header adjacent to A0. If you are using an Arduino Uno, map the free end of the sensor cable through a three element male/male header into a proto board and then with jumpers to the Uno headers.





2) Launch the Arduino.exe program located in the C:software folder.

3) Load the AnalogReadSerial program.
   a. File-> Examples->1.Basic->AnalogReadSerial



```
//  AnalogReadSerial: public domain code
//  Reads analog input on pin 0, prints to console.

void setup() // runs once when you press reset
{
  Serial.begin(9600);
}

void loop() // runs over and over forever
{
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  delay(1); // delay in between reads for stability
}
```

   b. This code sets up serial communication at 9600 baud to the PC, then reads an analog value from pin zero (A0), and prints the decimal value of the ADC converter reading to the serial port.
   c. Upload the program to the microprocessor File->Upload, or Ctrl+U. This will compile the code, check for errors, and upload the file.
   d. If you see an error message that says file did not upload, go to Tools-> Serial Port and change the serial port e.g. select com6 instead of com1. Try uploading again
   e. Open a serial terminal window and view the data coming back from the microprocessor. Tool->Serial Monitor or Ctrl+Shift+M. Typical values will be around 170.
   f. Put your finger on the temperature sensor to warm it up and watch for the A/D reading values to increase slightly.

4) Modify the code so that there is a delay between display updates:

   a. Increase the loop delay to 500 msec. This slows the loop down to two samples per second, making it easier to see changes. Add console header to `setup()`. Add loop counter `nSmpl`, and indexing outputs.

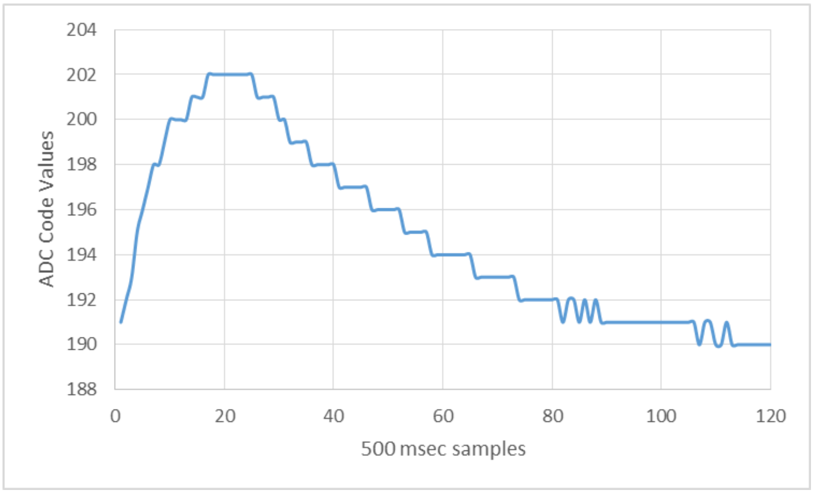
```
int nSmpl = 1, sample; // global variables

void setup()
{
  Serial.begin(9600);
  Serial.print("\nsmpl\tADC\n"); // column headers
}
void loop()
{
  sample = analogRead(A0);
  Serial.print(nSmpl); Serial.print('\t'); // tabbed output
  Serial.println(sample);
  ++nSmpl;
  delay(500);
}
```

   b. Upload the code. Close the old serial monitor. Open a new monitor window. Observe data values as you warm the sensor with your finger.

   c. Copy and paste data from the Serial Monitor window into Excel. Plot temperature data for 60 seconds or so. Capture temperature rise and fall.

   d. P1-1Include an Excel scatter plot of the temperature response in your report. Label axes and entitle graph:

   P1-1 Plot of LM61 temperature sensor responding to finger touch



SECTION 2 – Setting up a interrupt to provide a uniform sampling interval

Overview:

In the last section, you made the microprocessor take two samples per second, but the actual sampling period was not exactly 0.5 seconds. Rather it is the sum of the explicit delay and the execution time of the rest of the `loop()` code. In this section loop timing is established with a hardware timer driven interrupt routine. The interrupt used is defined in the header file MsTimer2.h, which stands for millisecond timer2. It is a library which contains an interrupt timer routine.

```
#include <MsTimer2.h>

const int TSAMP_MSEC = 100;
volatile boolean sampleFlag = false;
int nSmpl = 1, sample;

//*********************************************************************
void setup()
{
  Serial.begin(9600);
  MsTimer2::set(TSAMP_MSEC, ISR_Sample); // Set sample msec, ISR name
  MsTimer2::start(); // start running the Timer
}

//*********************************************************************
void loop()
{
  while (sampleFlag == false); // spin until ISR trigger
  sampleFlag = false;          // disarm flag: enable next dwell

  sample = analogRead(A0);

  // Display results to console
  if (nSmpl == 1) Serial.print("\nsmpl\tsmpl\n");
  Serial.print(nSmpl);  Serial.print('\t');
  Serial.print(sample); Serial.print('\n');

  ++nSmpl;

} // loop()

//*********************************************************************
void ISR_Sample()
{
  sampleFlag = true;
}
```

Program Description:

The first line includes the library MsTimer2.h. Global variable `sampleFlag` is used to signal that it is time to take a new sample. It is declared as `volatile boolean` to ensure that when an interrupt routine is called, it is not over-written or misplaced when execution jumps from the main loop to the interrupt routine and back.

In Setup, the serial port is started at 9600 baud. Then the MsTimer2 is configured to jump every 100 milliseconds to the interrupt routine called `ISR_Sample`. In `ISR_Sample`, the variable `sampleFlag` is set true. This is a message, or semaphore, to the main loop that it is time to read a new voltage. The voltage is not read by this interrupt routine, it just says that it is time to do so. After the MsTimer2 is configured, the next line starts the timer that will jump to the interrupt routine.

In the main loop, a `while()` loop blocks progress until `sampleFlag` is set true by `ISR_sample`. Once the flag is detected true, execution escapes the `while()` loop. A new analog voltage is read and sent out over the serial port.

By setting up the code this way, a sample taken every 100 milliseconds regardless of the execution time of the sampling and console output code. This regular sampling interval is critical for low noise, high performance sampled data system operation.

Procedure

1) Download the MsTimer2.h library from the Arduino.cc website and place it into the **libraries** folder under the Arduino folder under program files. The library must be in its own folder named MsTimer2. After you do this, the Arduino IDE must be quit and restarted so that it recognizes the new library files.
2) Copy and paste the file code shown, upload, and run the program. Open the serial port and watch the data come in at a faster rate.
3) Collect 60 seconds or so of data, starting from ambient temperature, then putting your finger on the sensor for 10 seconds and then letting go of the sensor. Plot the data in Excel.
4) P1-2 Include an Excel plot of the temperature response over in your report. Label axes and entitle graph as:

   P1-2 Plot of LM61 temperature sensor responding to being touched

SECTION 3a – Controlling start of data collection

Now that the temperature samples are taken at regular time intervals, enable user control of data collection with the following code addition. Execution spins in setup() until a 'g' character is entered in the console.

```
void setup()
{
  Serial.begin(9600);
  Serial.println("Enter 'g' to go .....");
  while (Serial.read() != 'g'); // spin until 'g' entry

  MsTimer2::set(TSAMP_MSEC, ISR_Sample); // Set sample msec, ISR name
  MsTimer2::start(); // start running the Timer
  }
```

SECTION 3b – Collecting exactly 256 samples of data

A counter can be used to collect a certain number of samples and then stop the program. For ease of maintenance, define the sample count as a global constant variable in setup(). At the end of the loop, replace the simple increment of `nSmpl` with an increment and comparison to the target count. When the target is achieved, disable `ISR_Sample()` calls. The loop then freezes in the `while(sampleFlag)` loop.

```
...

const int NUM_SAMPLES = 256;

...

if (++nSmpl > NUM_SAMPLES) MsTimer2::stop();

...
```

P1-3 Include in your report an Excel plot of the temperature response for exactly 256 samples. Label axes and entitle graph as:

P1-3 Plot of LM61 temperature sensor responding to being touched

<u>Write Up:</u>

Include the cover sheet (next page) on top of your lab report.

<u>For a lab report grade of B, include the following.</u>

1) Your lab report must be submitted in MS Word using the IEEE Journal Transactions Format. A template Word document is posted on MyCourses for you to download. Write your report directly into the template document. Use the two column format as given.

3) Include a simple title. Include the names of all your lab partners as co-authors.

4) Include the three requested plots.

   a) Fully label the vertical and horizontal axes of graphs. The vertical axis is the A/D converter reading as an integer from 0-1023 which represents temperature. The horizontal axis is the sample number. You could convert both axis labels to engineering units (e.g. temperature in degrees C, or time in seconds), however that is not required for this report.

   b) Place a descriptive caption <u>below</u> each figure. The description that you write in the caption should sufficiently explain what is going on in the figure that a person who could not see the graph (e.g. a blind person using a text-to-speech program) would still be able to understand what the graph is showing.

   c) For Tables, the title of the table goes <u>above</u> the Table. See the example in the IEEE journal paper template.

5) Include a very short discussion of your results. In addition to a general discussion of your results, address the following:

   a) When, and how often is `sampleFlag` set true?

   b) Why is `sampleFlag` set false every time the main loop executes? (hint: What would happen if it was not set false?)

<u>For a lab report grade of A, also include the following</u>

6) Write code that runs in the Arduino to calculate the running mean and the running standard deviation. You can adapt the BASIC code that is given in chapter 2 of Smith. Include a text box that contains your statistics code. Include plots of mean and standard deviation running statistics and an associated discussion section that explains why the graphs look as they do.

7) Your lab partner believes when a person breathes on the temperature sensor (causing the temperature signal to go up and down) that the mean statistic will show this better than the standard deviation statistic. <u>Write a one paragraph response</u> to your lab partner than explains why you think the standard deviation statistic will be more responsive to temperature variations than the mean statistic. Support your argument with either data from your experiments or a rationale based on the formulas for mean and standard deviation.

CoverSheet: Lab #1 Intro to Arduino Microprocessor

Name:_______________________ Date: ____________________

LABORATORY GRADE	
Title, Name of Lab Partners, Two Column Format	/20
Plots for grade B paper - complete set, correct labels, brief discussion	/40
Plots for A paper - complete set, correct labels, brief discussion	/20
Lab participation	/20
Final Grade	/100

P1-1 Plot of LM61 temperature sensor responding to being touched (~60 seconds@ Tsample~500ms).

P1-2 Plot of LM61 temperature sensor responding to being touched (~60 seconds@ Tsample=100ms).

P1-3 Plot of LM61 temperature sensor responding to being touched (256 samples@ Tsample=100ms).

P1-4 Plot of mean (running statistics) of LM61 temperature sensor signal in response to being touched (256 samples@ Tsample=100ms).

P1-5 Plot of standard deviation (running statistics) of LM61 temperature sensor signal in free air (256 samples@ Tsample=100ms).