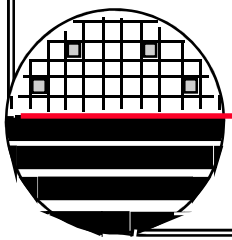
ROCHESTER INSTITUTE OF TECHNOLOGY
MICROELECTRONIC ENGINEERING

Micro Controller - Basics (for Microsystems)

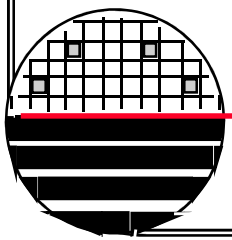
Dr. Lynn Fuller
Webpage: http://people.rit.edu/lffeee/
Microelectronic Engineering
Rochester Institute of Technology
82 Lomb Memorial Drive
Rochester, NY 14623-5604
Tel (585) 475-2035

Email: Lynn.Fuller@rit.edu
MicroE webpage: http://www.microe.rit.edu



OUTLINE

Definitions
Microsystem
Microcontroller
Arduino Hardware
Software for Arduino IDE
Software for Processing PDE
Processing Graphical Output
Output Data File
References
C++ Primer
Example: Display of Analog Signals
Homework Questions



DEFINITIONS

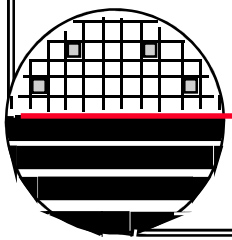
Arduino – refers to a project that provides open source hardware and software to learn by doing projects with micro controllers.

Uno – one of the several Arduino hardware platforms available containing a micro controller, power regulator, USB interface and interconnect pins and sockets.

Shield – an add on hardware board that plugs into the Arduino micro controller platform and provides additional capabilities such as blue tooth wireless, WiFi, etc.

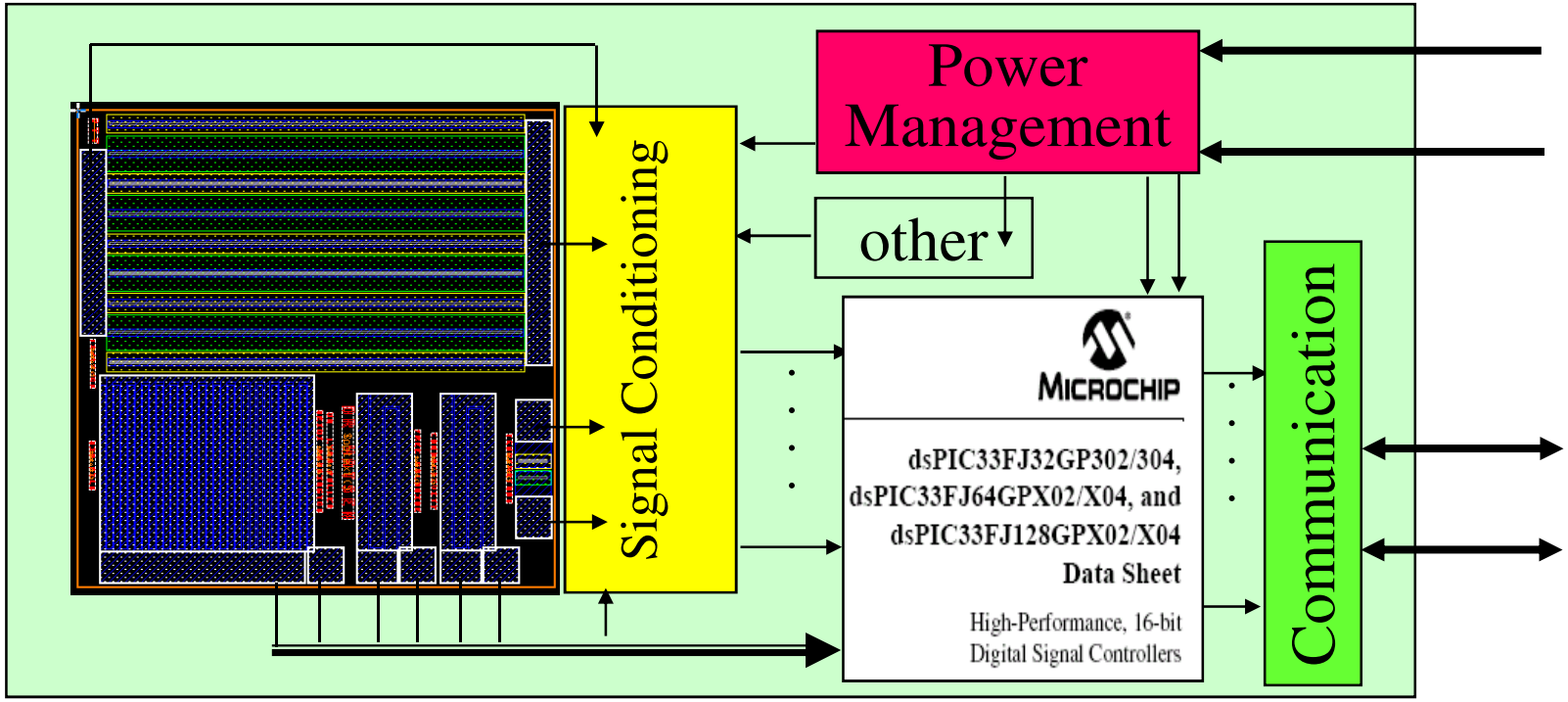
Processing – a "C" based software programming tool to create graphical output and communicate with hardware platforms such as the Arduino Uno.

Sketch – name for the "C" programs used by "Processing" and by "Arduino" software to make the hardware do something and to process the results.



RIT MICROSYSTEM CONCEPT

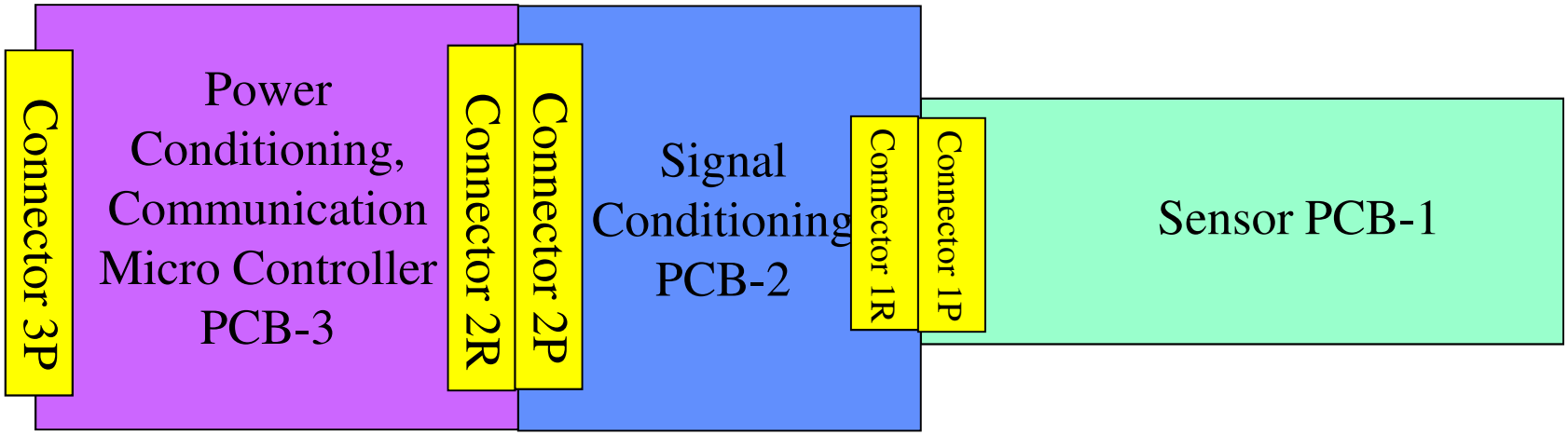
Multi-Sensor MEMs Chip



Micro Controller

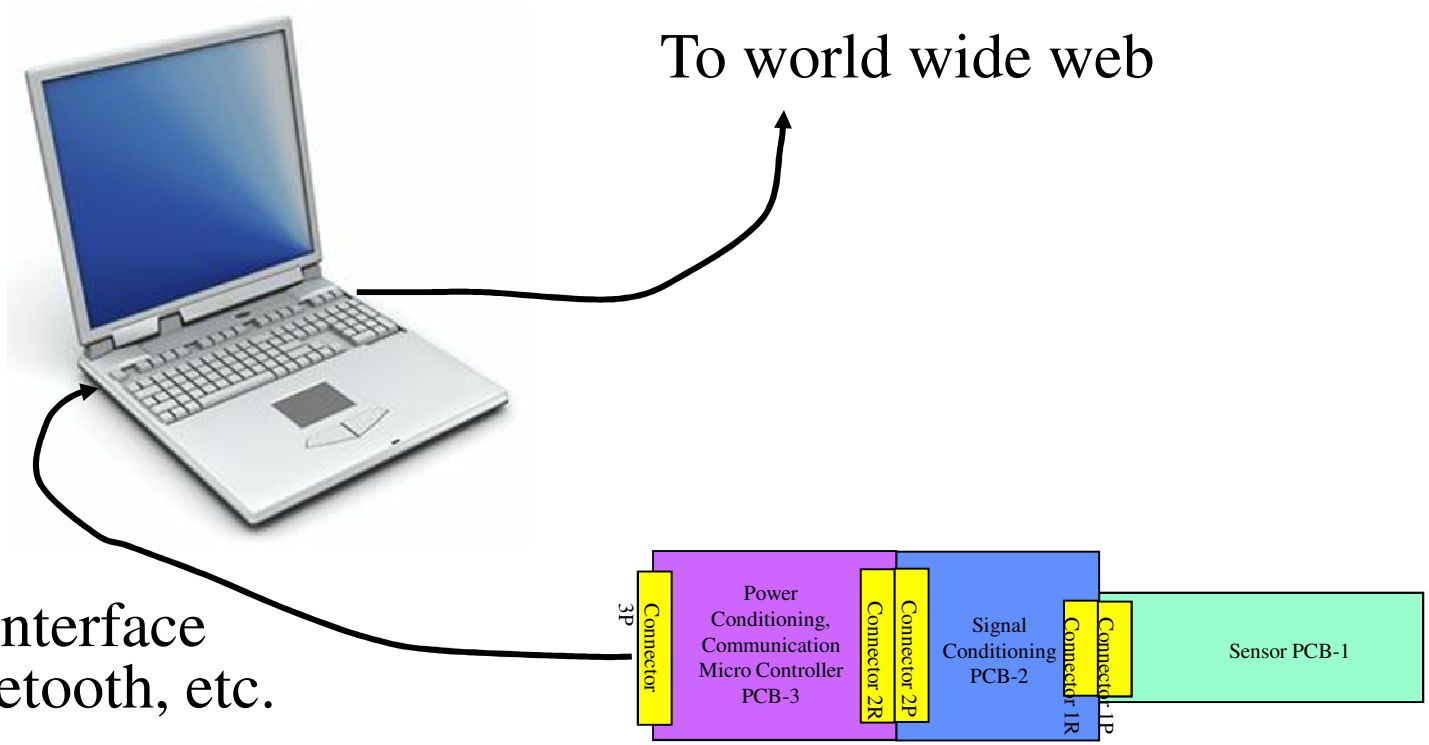
Signal Conditioning Electronics

MODULAR SYSTEM CONCEPT



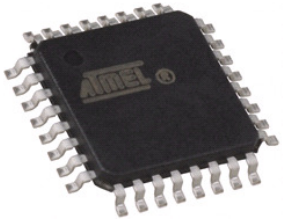
Eventually PCB-2 and PCB-3 should be combined into one PCB. The Sensors and PCB-1 will need replacement in the field after an unspecified time yet to be determined.

COMPLETE SYSTEM



MICRO CONTROLLERS – MADE BY

Analog Devices
Atmel
Cirrus
Cypress
Fairchild
Freescale Semiconductor
Intel Maxim
Microchip Technologies
National Semiconductor
Silicon Laboratories
STMicroelectronics
Texas Instruments
Zilog
Others



ATmega329
~$5

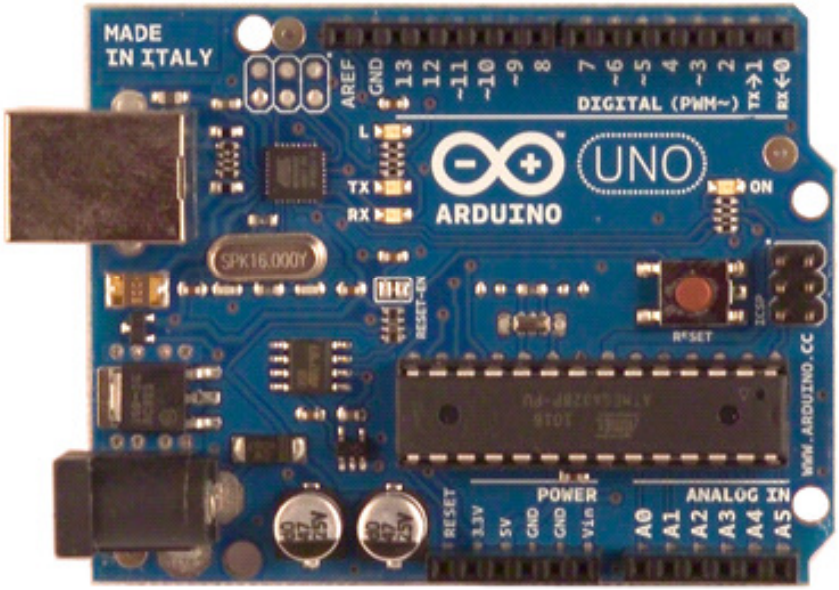
Packages:
Surface Mount
Through hole
Chip Scale

Digikey carries ~6,000 microcontroller products
from 25 companies (4-bit to 32-bit)
8-bit micro controllers start at under $1

ATMEL MICRO CONTROLLER

The Arduino Project uses the Atmel ATMega328 micorcontroller. It is a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996. This was one of the first microcontroller families to use on-chip flash memory for program storage.

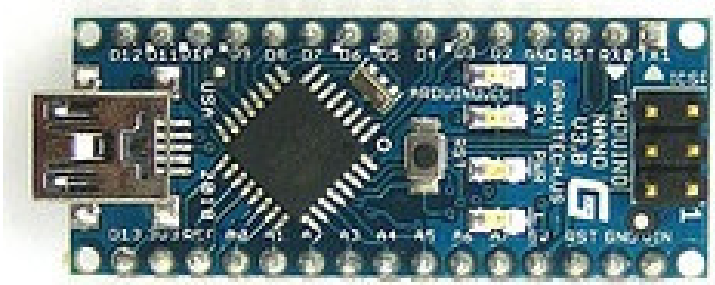
ARDUINO UNO



Uno Development Board
2” x 2 ¾”
~$29

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

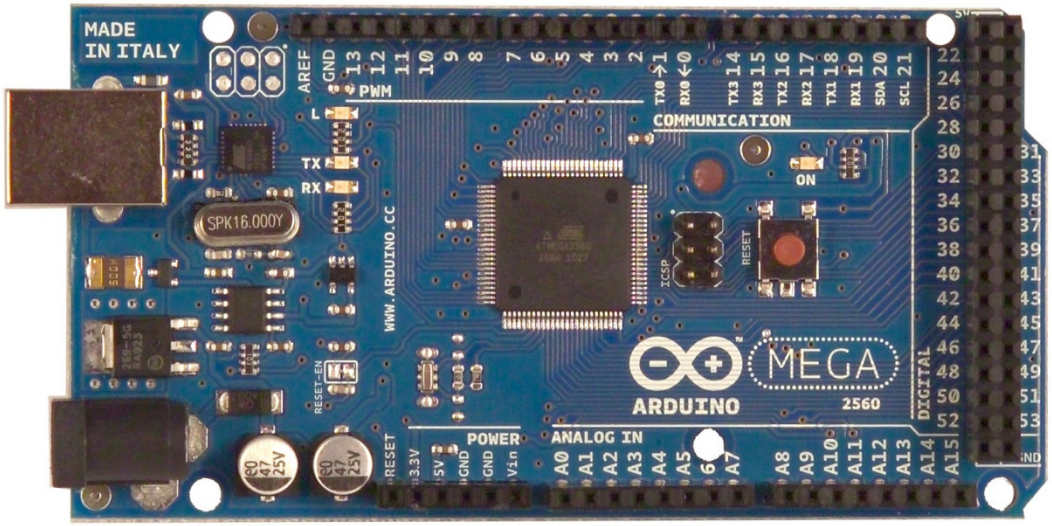
OTHER ARDUINO HARDWARE



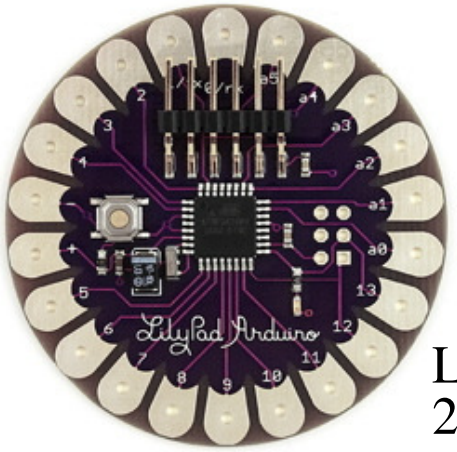
Nano (smaller 0.73” x 1.70”) ~$35

Bluetooth ~$150

http://arduino.cc/en/Main/Hardware

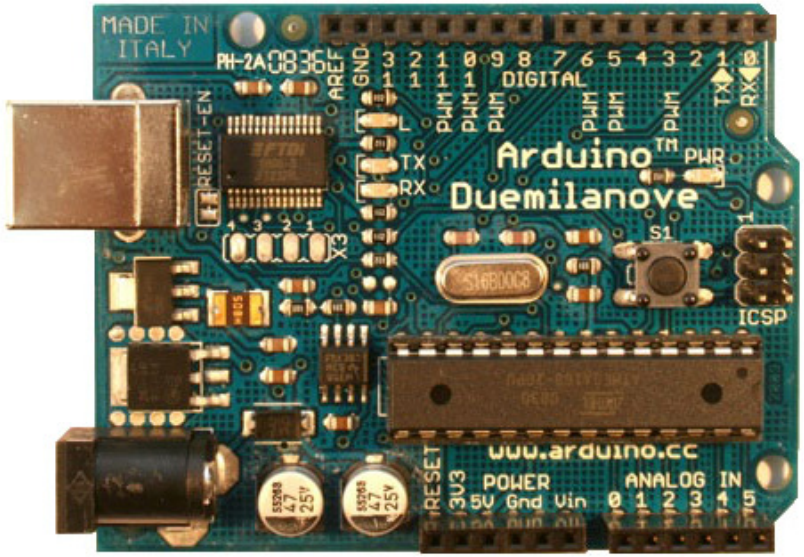


Mega ~$65



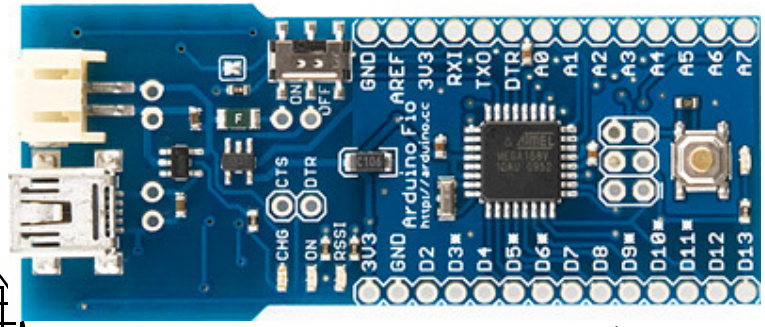
Lily Pad
2” diameter
~$22

MORE ARDUINO HARDWARE

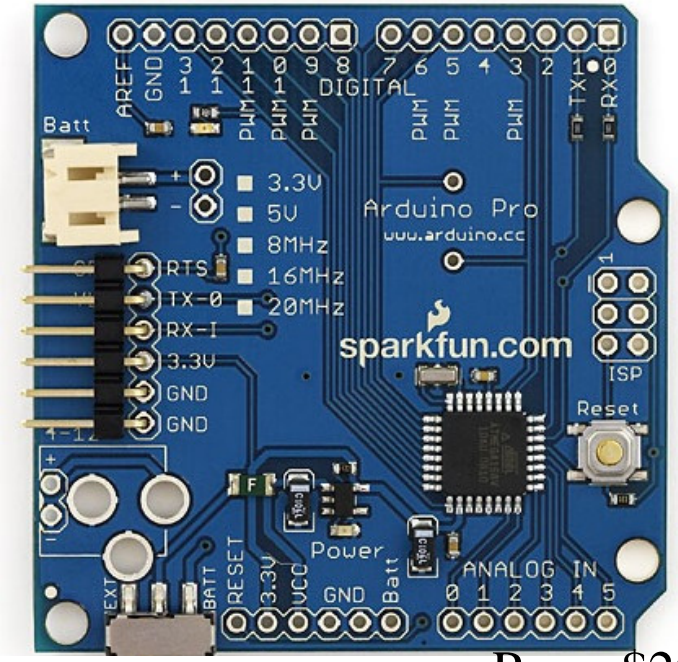


Duemilanove (replaced by Uno)

Pro Mini (smallest)
0.7” x 1.3” ~$19



Fio (Xbee) ~$25



Pro ~$20

http://arduino.cc/en/Main/Hardware

INTRODUCTION TO THE SOFTWARE

Each company that sells micro controllers provide software to create, compile and upload programs to the micro controller. The software to work with output from the micro controller may be third party. Lab View is an example of a third party software.

USER INTERFACE TO "ARDUINO" SOFTWARE

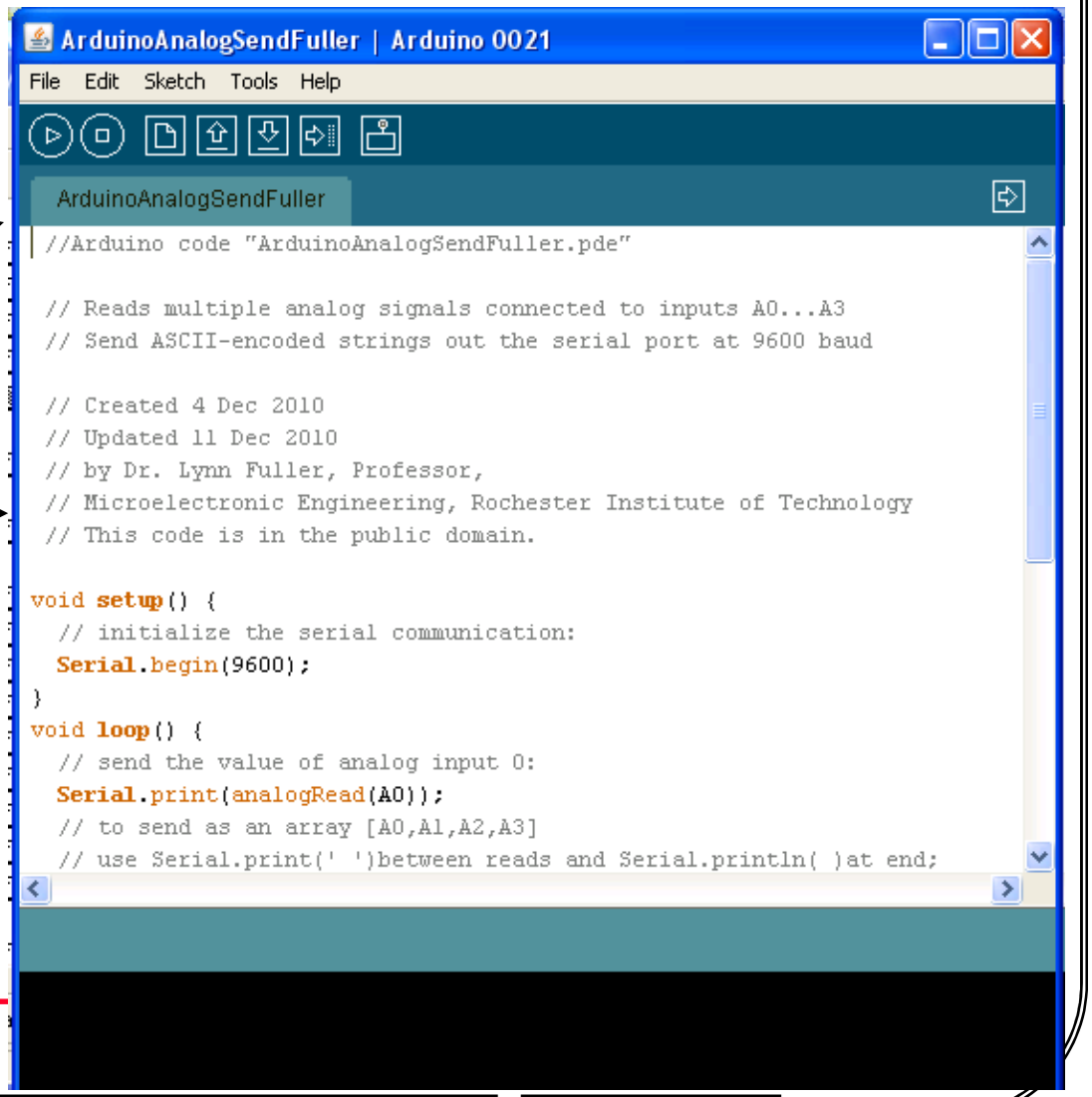
Tool Bar - Run, Stop, New, Open, Save, Upload and Serial Monitor buttons

Tabs

Text Editor Space to create and edit "C" code

Message Area

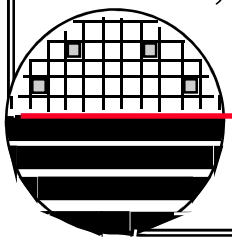
Dialog Text active during run



PROCESSING DEVELOPMENT ENVIRONMENT

The Processing Development Environment (PDE) consists of a simple text editor for writing code, a message area, a text console, tabs for managing files, a toolbar with buttons for common actions, and a series of menus. When programs are run, they open in a new window called the display window.

Software written using Processing are called sketches. These sketches are written in the text editor. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by Processing programs including complete error messages and text output from programs with the print() and println() functions. The toolbar buttons allow you to run and stop programs, create a new sketch, open, save, and export:



USER INTERFACE TO "PROCESSING" SOFTWARE

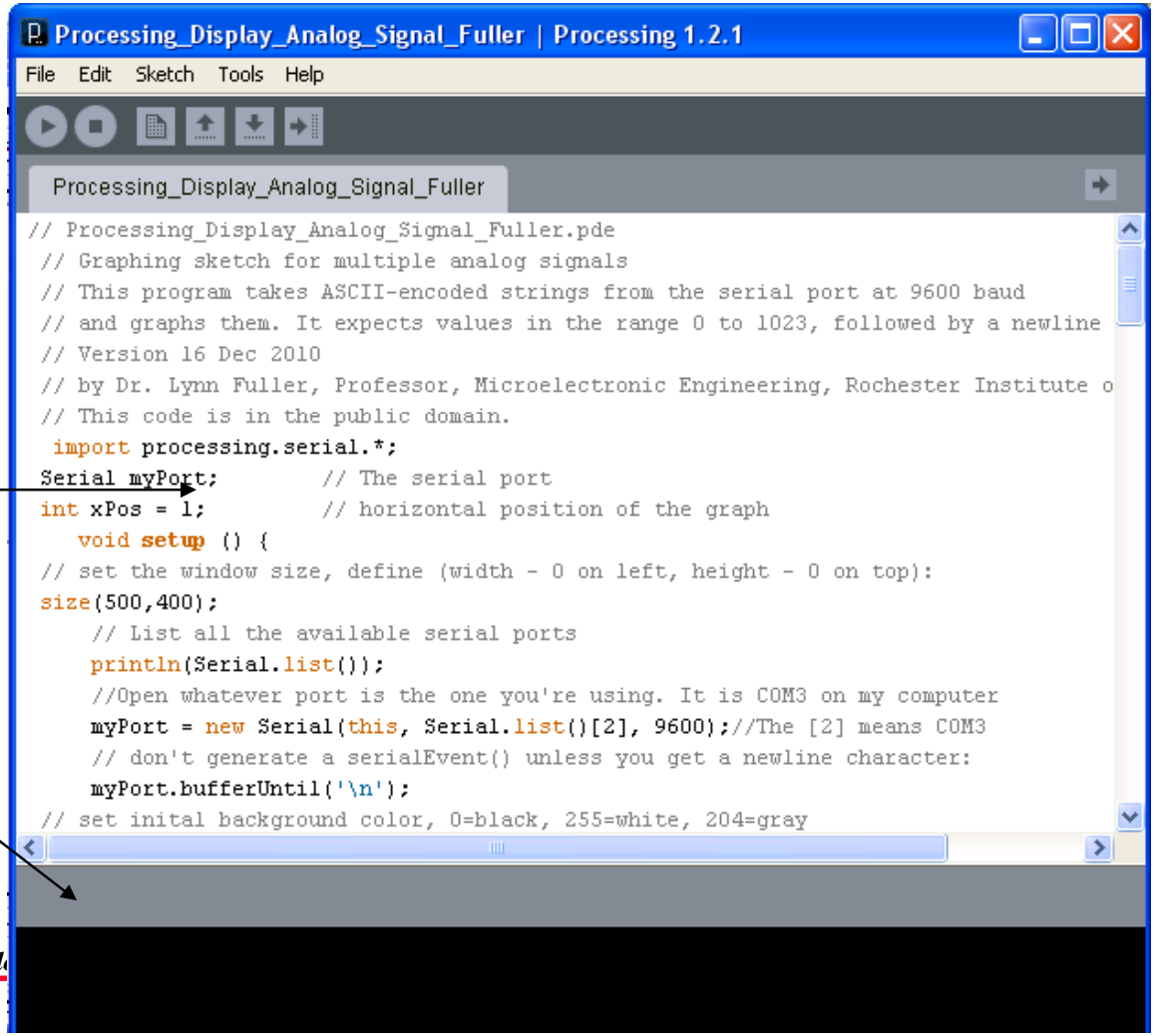
Tool Bar - Run, Stop, New, Open, Save, Export buttons

Tabs

Text Editor Space to create and edit "C" code

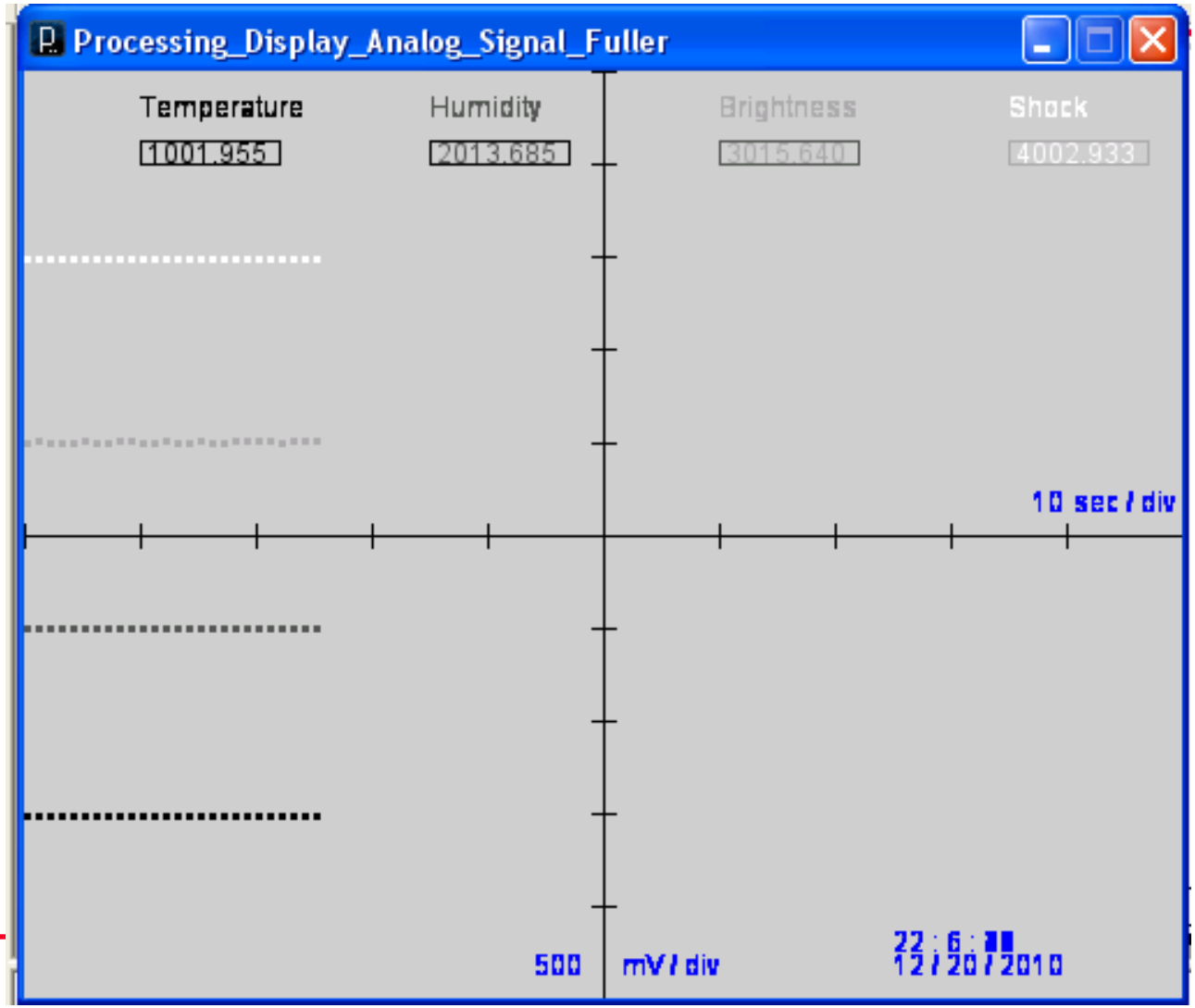
Message Area

Dialog Text active during run



GRAPHICAL OUTPUT USING "PROCESSING"

Display Window



OUTPUT TEXT FILE

Output_Data_File.txt - Notepad

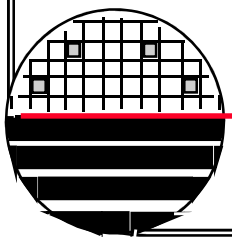
File Edit Format View Help

```
A0  Fixed          12  31  2010   18 : 54 : 35  259.62854
A1  Force          12  31  2010   18 : 54 : 35  73.9003
A2  Temperature    12  31  2010   18 : 54 : 35  310.45944
A3  Variable       12  31  2010   18 : 54 : 35  218.96384
A0  Fixed          12  31  2010   18 : 54 : 35  274.8778
A1  Force          12  31  2010   18 : 54 : 35  73.9003
A2  Temperature    12  31  2010   18 : 54 : 35  310.85043
A3  Variable       12  31  2010   18 : 54 : 35  218.96384
A0  Fixed          12  31  2010   18 : 54 : 36  258.4555
A1  Force          12  31  2010   18 : 54 : 36  74.68231
A2  Temperature    12  31  2010   18 : 54 : 36  310.85043
A3  Variable       12  31  2010   18 : 54 : 36  218.96384
A0  Fixed          12  31  2010   18 : 54 : 36  256.5005
A1  Force          12  31  2010   18 : 54 : 36  74.68231
A2  Temperature    12  31  2010   18 : 54 : 36  310.45944
A3  Variable       12  31  2010   18 : 54 : 36  218.96384
A0  Fixed          12  31  2010   18 : 54 : 36  272.9228
A1  Force          12  31  2010   18 : 54 : 36  74.68231
A2  Temperature    12  31  2010   18 : 54 : 36  310.45944
A3  Variable       12  31  2010   18 : 54 : 36  218.96384
A0  Fixed          12  31  2010   18 : 54 : 36  259.23755
A1  Force          12  31  2010   18 : 54 : 36  74.29129
A2  Temperature    12  31  2010   18 : 54 : 36  310.85043
A3  Variable       12  31  2010   18 : 54 : 36  218.96384
A0  Fixed          12  31  2010   18 : 54 : 36  256.5005
A1  Force          12  31  2010   18 : 54 : 36  73.9003
A2  Temperature    12  31  2010   18 : 54 : 36  310.85043
A3  Variable       12  31  2010   18 : 54 : 36  218.96384
A0  Fixed          12  31  2010   18 : 54 : 36  271.74976
```

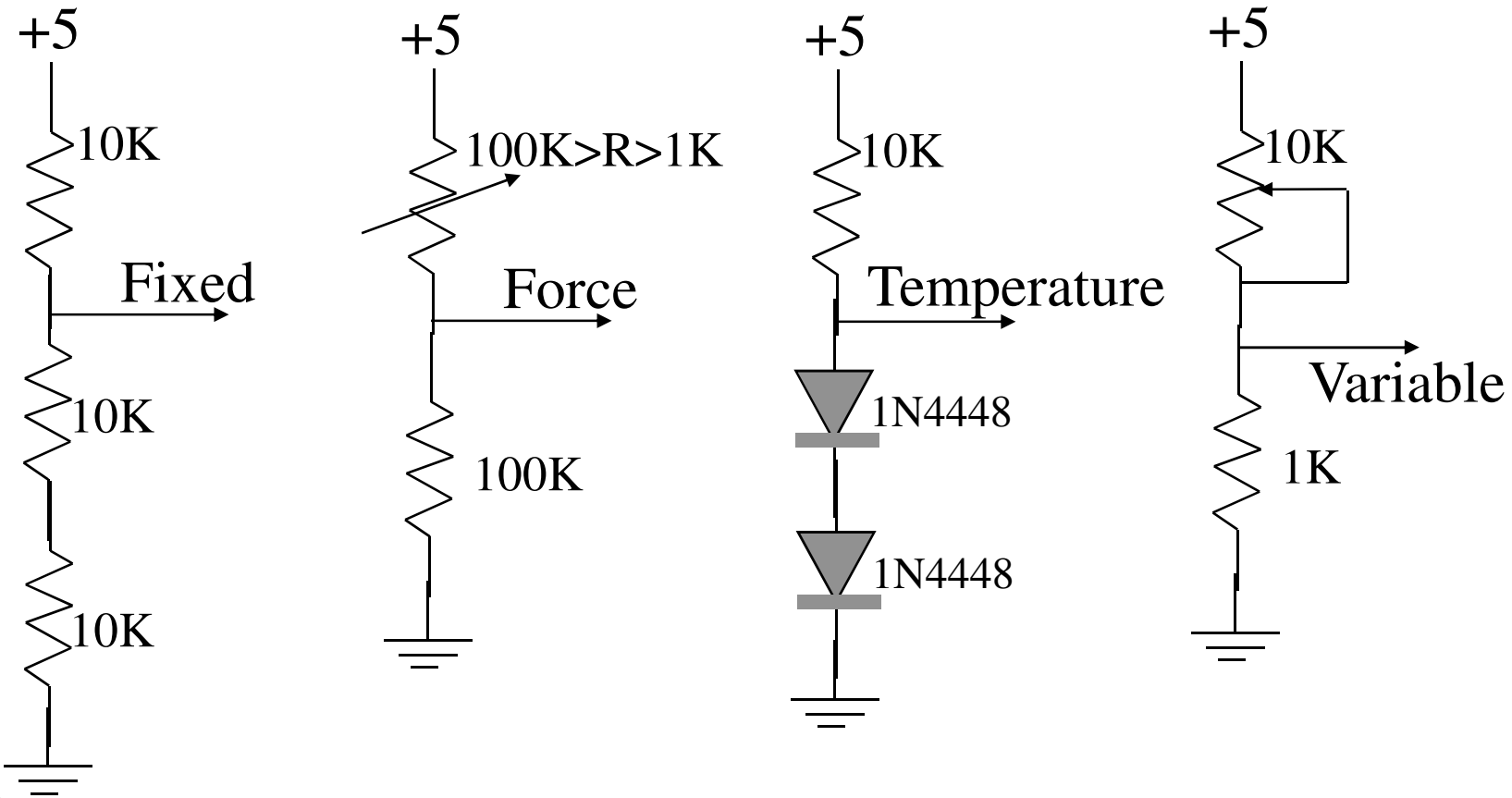
OUTPUT EXCEL FILE – BEFORE & AFTER SORTING

	A	B	C	D	E	F	G	H	I	J	K
1	A0	Fixed	12	31	2010	18	:	54	:	35	259.62854
2	A1	Force	12	31	2010	18	:	54	:	35	73.9003
3	A2	Temperature	12	31	2010	18	:	54	:	35	310.45944
4	A3	Variable	12	31	2010	18	:	54	:	35	218.96384
5	A0	Fixed	12	31	2010	18	:	54	:	35	274.8778
6	A1	Force	12	31	2010	18	:	54	:	35	73.9003
7	A2	Temperature	12	31	2010	18	:	54	:	35	310.85043
8	A3	Variable	12	31	2010	18	:	54	:	35	218.96384
9	A0	Fixed	12	31	2010	18	:	54	:	36	258.4555
10	A1	Force	12	31	2010	18	:	54	:	36	74.68231
11	A2	Temperature	12	31	2010	18	:	54	:	36	310.85043
12	A3	Variable	12	31	2010	18	:	54	:	36	218.96384
13	A0	Fixed	12	31	2010	18	:	54	:	36	256.5005
14	A1	Force	12	31	2010	18	:	54	:	36	74.68231
15	A2	Temperature	12	31	2010	18	:	54	:	36	310.45944
16	A3	Variable	12	31	2010	18	:	54	:	36	218.96384
17	A0	Fixed	12	31	2010	18	:	54	:	36	272.9228
18	A1	Force	12	31	2010	18	:	54	:	36	74.68231
19	A2	Temperature	12	31	2010	18	:	54	:	36	310.45944
20	A3	Variable	12	31	2010	18	:	54	:	36	218.96384

	A	B	C	D	E	F	G	H	I	J	K
100	A1	Force	12	31	2010	18	:	54	:	38	74.68231
101	A1	Force	12	31	2010	18	:	54	:	38	74.29129
102	A1	Force	12	31	2010	18	:	54	:	38	74.68231
103	A1	Force	12	31	2010	18	:	54	:	38	74.68231
104	A1	Force	12	31	2010	18	:	54	:	38	74.68231
105	A1	Force	12	31	2010	18	:	54	:	38	74.29129
106	A1	Force	12	31	2010	18	:	54	:	38	74.29129
107	A1	Force	12	31	2010	18	:	54	:	38	74.68231
108	A1	Force	12	31	2010	18	:	54	:	39	73.9003
109	A1	Force	12	31	2010	18	:	54	:	39	73.9003
110	A1	Force	12	31	2010	18	:	54	:	39	73.9003
111	A1	Force	12	31	2010	18	:	54	:	39	74.29129
112	A1	Force	12	31	2010	18	:	54	:	39	74.29129
113	A1	Force	12	31	2010	18	:	54	:	39	74.29129
114	A1	Force	12	31	2010	18	:	54	:	39	74.29129
115	A1	Force	12	31	2010	18	:	54	:	39	74.68231
116	A1	Force	12	31	2010	18	:	54	:	39	74.29129
117	A1	Force	12	31	2010	18	:	54	:	40	74.68231
118	A1	Force	12	31	2010	18	:	54	:	40	74.29129
119	A1	Force	12	31	2010	18	:	54	:	40	73.9003
120	A1	Force	12	31	2010	18	:	54	:	40	74.29129
121	A0	Fixed	12	31	2010	18	:	54	:	35	259.62854
122	A0	Fixed	12	31	2010	18	:	54	:	35	274.8778
123	A0	Fixed	12	31	2010	18	:	54	:	36	258.4555



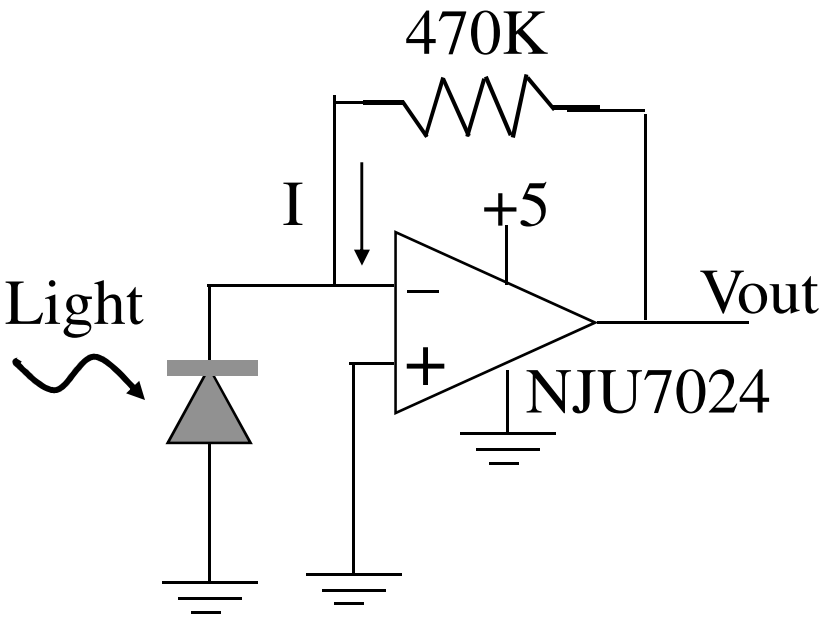
FIXED, FORCE, TEMPERATURE, VARIABLE



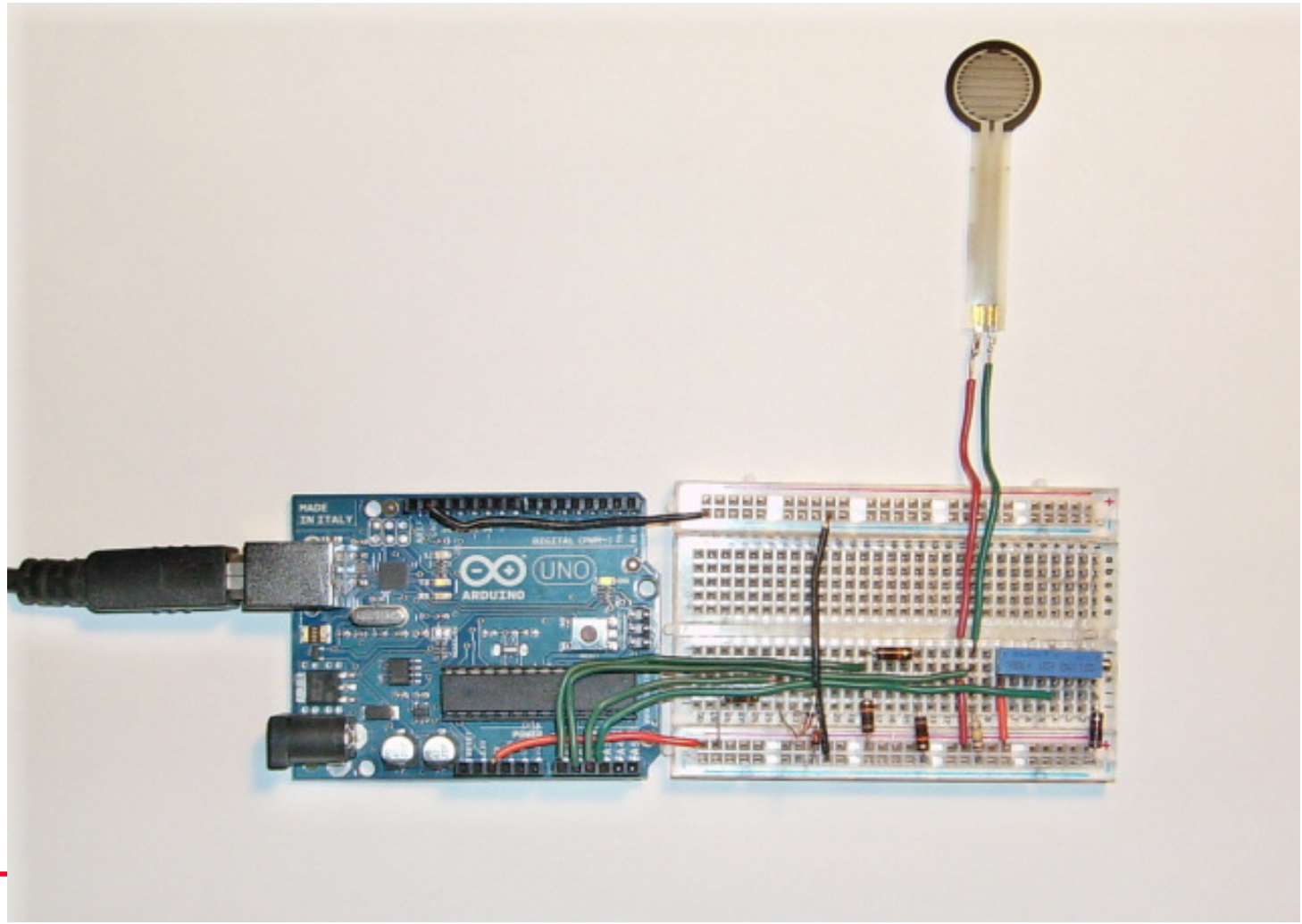
Circuits used to give four analog inputs

SIGNAL CONDITIONING

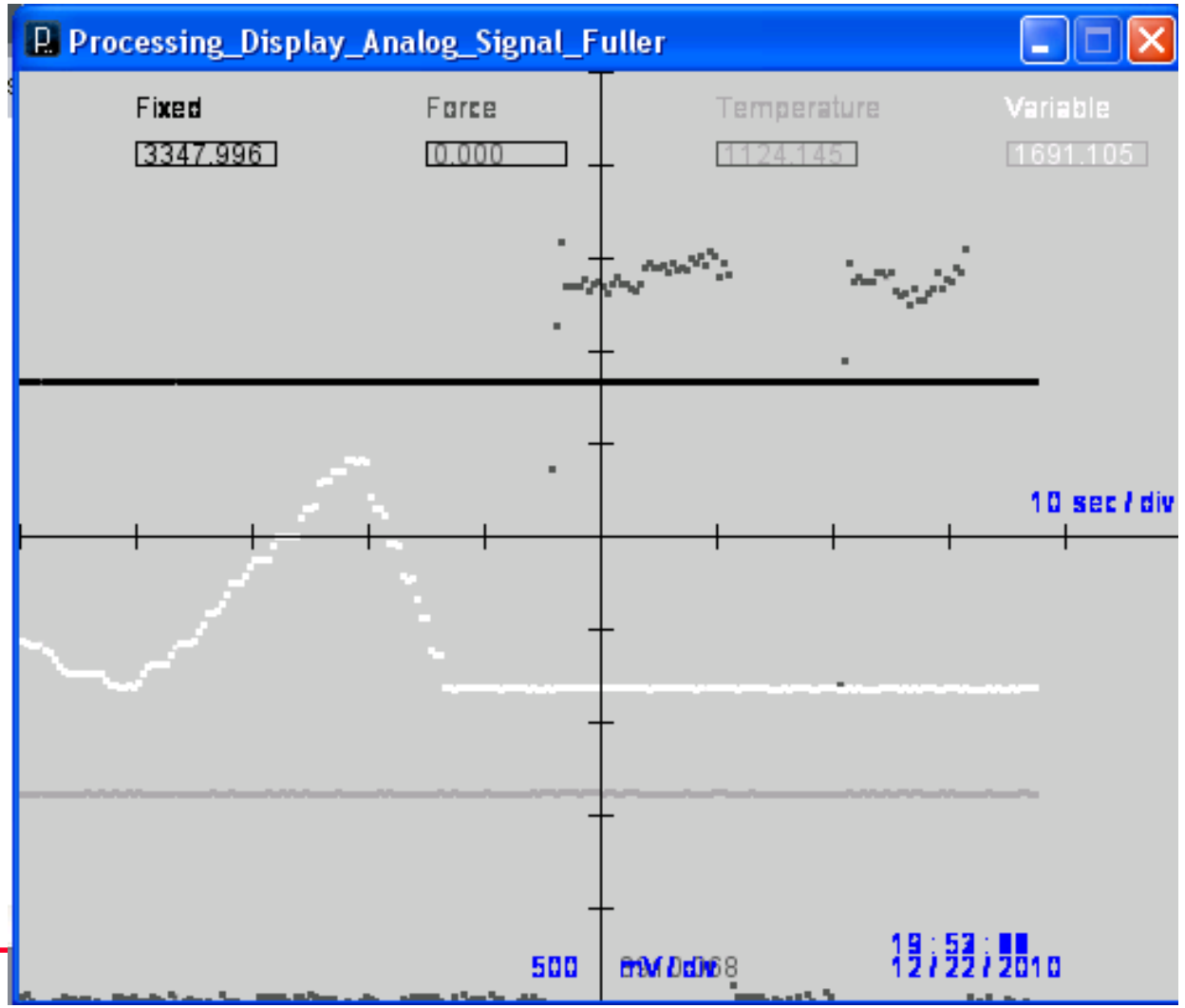
Vishay BPW46
Digikey No. 751-1017-ND



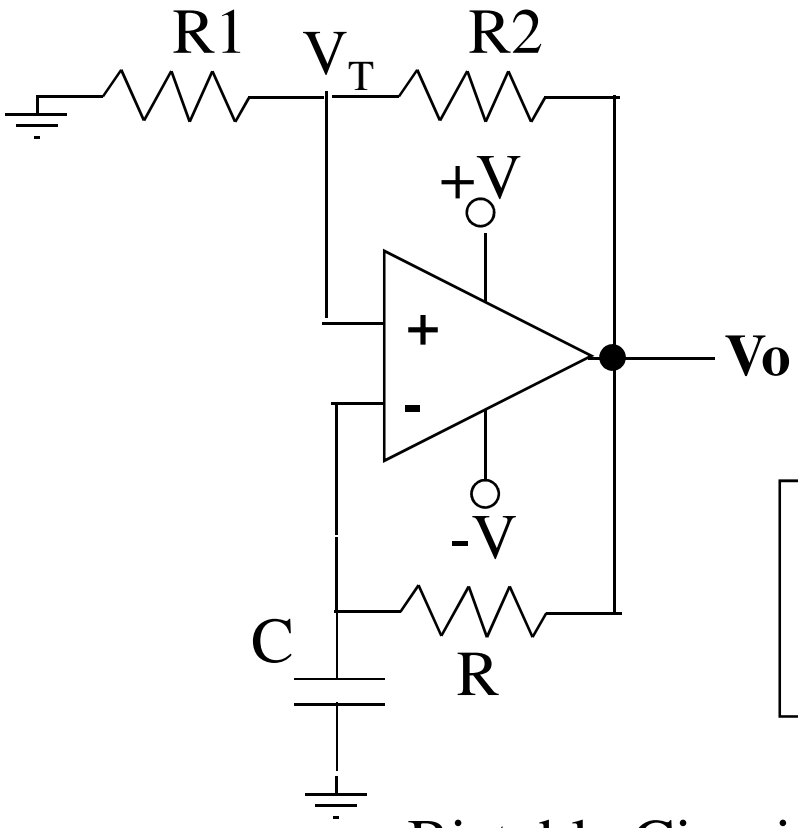
FIXED, FORCE, TEMPERATURE, VARIABLE

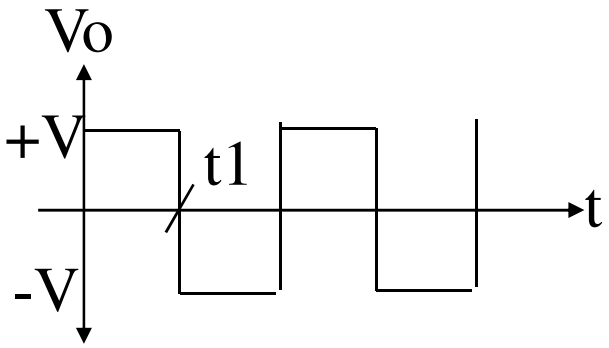


USING DIFFERENT SENSORS



OSCILLATOR (MULTIVIBRATOR)

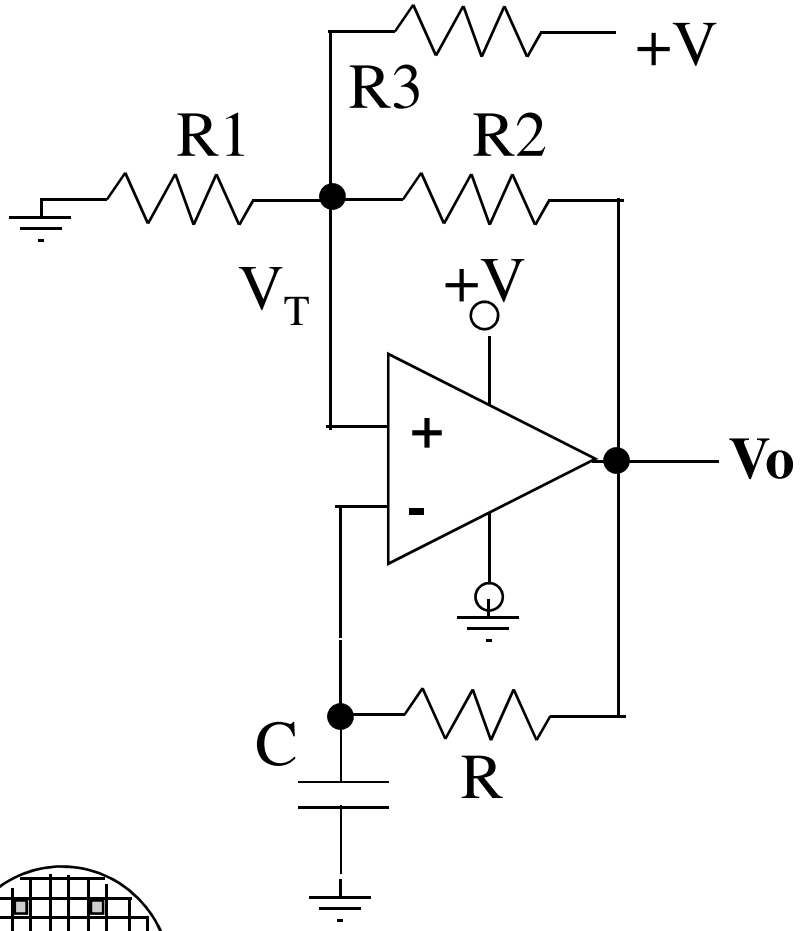


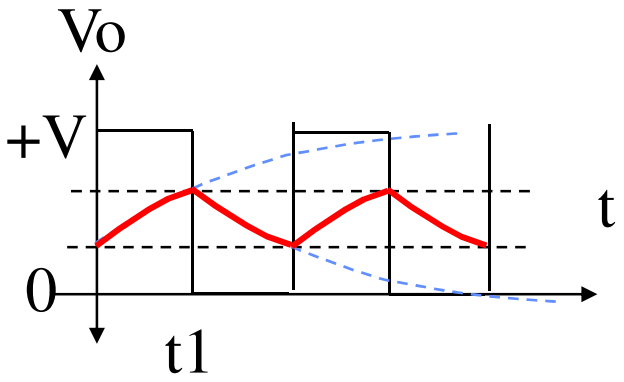


$$\text{Period} = T = 2RC \ln \left(\frac{1+Vt/V}{1-Vt/V} \right)$$

Bistable Circuit with Hysteresis and RC Integrator

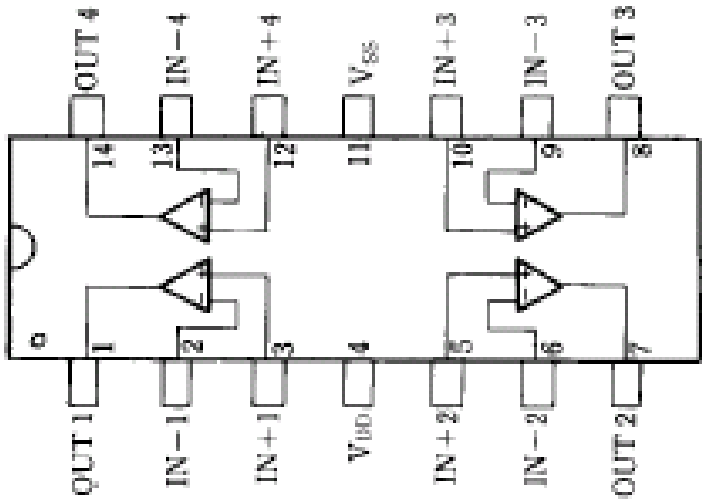
SINGLE SUPPLY OSCILLATOR (MULTIVIBRATOR)





Let R1 = 100K, R2=R3=100K
and +V = 3.3
Then $V_T = 2.2$ when Vo = 3.3
$V_T = 1.1$ when Vo = 0

OSCILLATOR


OUT 4
IN−4
IN+4
V_{SS}
IN+3
IN−3
OUT 3
14
13
12
11
10
9
8
OUT 1
IN−1
IN+1
V_{DD}
IN+2
IN−2
OUT 2
1
2
3
4
5
6
7
NJU7024D
NJU7024M
NJU7024V

C++ PRIMER - Page 1 of 4

Arduino Programming in Brief: The Arduino is programmed in the C language. This primer is for people who have a little bit of programming experience and just need a briefing on C and Arduino IDE. For more help see www.Arduino.cc , especially the Reference link.

Structure: Each Arduino program (sketch) has two required functions (routines).
void setup( ) { } All the code between the two curly brackets will be run once when the Arduino program first runs.
void loop ( ) { } This function is run after setup has finished. After it has run once it will be run again, and again, until power is removed.

Syntax:
// (single line comment) everything after the double slash to end of the line.
/* */ (multi line comment) everything between /* and */ is treated as a comment.
{ } (curly brackets) used to define when a block of code starts and ends, used in functions as well as loops.
; (semicolon) each line of code must end with a semicolon.
Also: commands are case sensitive, space and tabs are ignored, lines can only be 64 characters long.

C++ PRIMER - Page 2 of 4

Variables:

int (integer) stores a number in 2 bytes (16 bits), has no decimal places, number is between -32768 to +32768

long (long) used when an integer is not large enough, 5 bytes (32 bits), number is between –2,147,483,648 and +2,147,483,648

boolean (boolean) simple True or False, uses one bit

float (float) floating point math (uses decimals) 4 bytes (32 bits), number is between -3.4028235E+38 and + 3.4028235E+38

char (character) stores one character using ASCII code, 1 byte (8 bits)

Math Operators:

=	assignment, makes something equal to something
%	modulo gives the remainder, eg 12 % 10 gives 2
-	subtraction
+	addition
*	multiplication
/	division

more see www.arduino.cc especially the Reference link

C++ PRIMER - Page 3 of 4

Comparison Operators: used for logical comparison

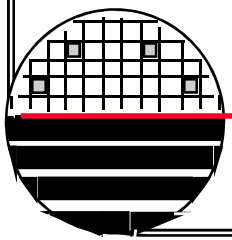
= =	equal	<	less than
!=	not equal	>	greater than
< =	less than or equal to		
> =	greater than or equal to		

more see www.arduino.cc especially the Reference link

Control Structure:

if (condition) { }	this will execute the code between the curly brackets if
else if (condition) { }	the condition is true, and if not it will test the else if
else { }	condition if that is also false the else code will execute

for (int i = 1; i < #repeats; i++) { } used when you want to repeat a chunk of code a number of times



C++ PRIMER - Page 4 of 4

Digital:

pinMode (pin, mode); // pin is the pin number, 0-19, (analog 0 to 5 are 14-19) mode is either INPUT or OUTPUT

digitalWrite(pin, value); //once a pin is set as an OUTPUT, it can be set either HIGH (pulled to +5 volts) or LOW (pulled to ground)

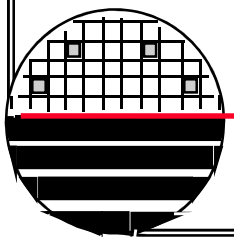
int digital Read (pin); //once a pin is set as an INPUT, you can use this function to return whether it is HIGH or LOW

Analog:

int analogWrite (pin, value); // some of the Arduino's pins support pulse width modulation (3,5,6,9,10,11). The value is any number between 0 (0% duty cycle) and 255 (100% duty cycle)

int analogRead (pin); // when analog pins are set to input you can read their value between 0 (zero volts) and 1024 (5 volts)

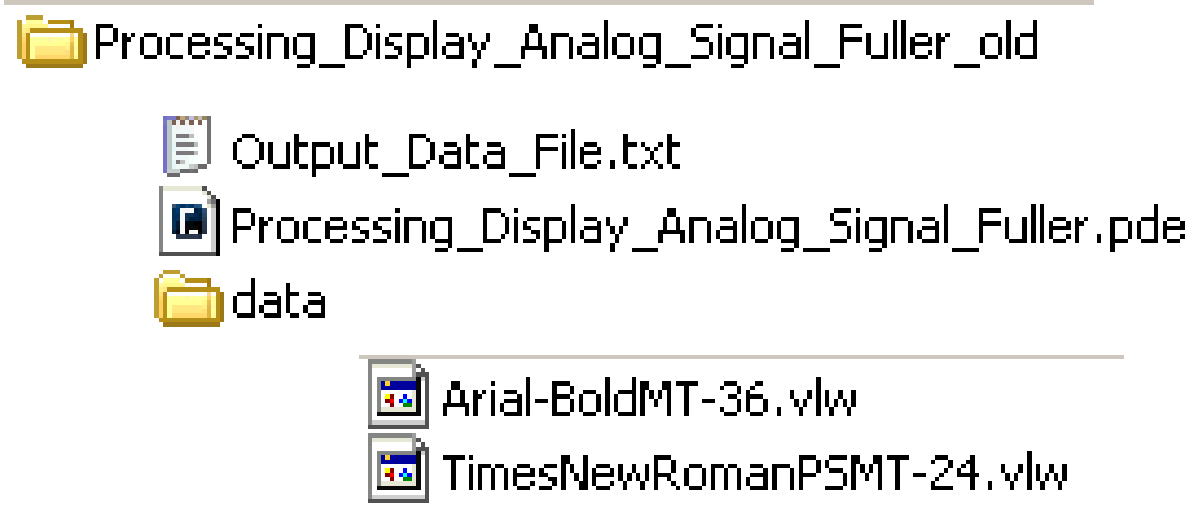
see www.arduino.cc especially the Reference link



PROCESSING PROGRAMMING

Processing Programming:

The sketch file must be in a folder with same name. Other associated files and data are also in that folder. For example font data folder, output files, etc.



PROCESSING CODE – PART 1 of 5

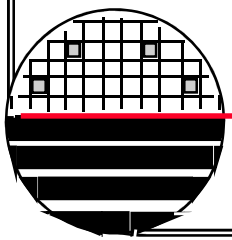
```
// Processing_Display_Analog_Signal_Fuller.pde
 // Graphing sketch for multiple analog signals
 // This program takes ASCII-encoded strings from the serial port at 9600 baud
 // and graphs them. It expects values in the range 0 to 1023, followed by a newline
 // Version 16 Dec 2010
 // by Dr. Lynn Fuller, Professor, Microelectronic Engineering, Rochester Institute of Technology
 // This code is in the public domain.
 import processing.serial.*;
 Serial myPort;        // The serial port
 int xPos = 1;         // horizontal position of the graph
 PrintWriter output;
   void setup ( ) {
 output=createWriter("Output_Data_File.txt"); //file name in sketch directory for output
 // set the window size, define (width - 0 on left, height - 0 on top):
 size(500,400);
    // List all the available serial ports
    println(Serial.list( ));
    //Open whatever port is the one you're using. It is COM3 on my computer
    myPort = new Serial(this, Serial.list( )[2], 9600);//The [2] means COM3
    // don't generate a serialEvent( ) unless you get a newline character:
    myPort.bufferUntil('\n');
 background(204);// set inital background color, 0=black, 255=white, 204=gray
 loadFont("Arial-BoldMT-36.vlw"); // Load Font used on graph
                        }
```

PROCESSING CODE – PART 2 of 5

```
void draw (  ) { ;
 // make axis, color is set by stroke(v1), v1=0 is black, or stroke(R,G,B)
 stroke(0,0,0);  // Black
 strokeWeight (0); // thin line for x and y axis
   line(0,height/2,width,height/2); // x-axis, line(x1,y1,x2,y2)
   line(width/2,0,width/2,height); // y-axis, line(x1,y1,x2,y2)
     int tics=10; // tic marks, tics is the number of tic marks on y axis
     for (int k=0; k<tics; k=k+1) {
               line(width/2-5,k*height/tics,width/2+5,k*height/tics);
               line(k*width/tics,height/2-5,k*width/tics,height/2+5);
                int Scale=5000/tics;// Full scale is 5 volts or 5000 mV for Arduino A to D
               // Scale in mV / div
               fill(0,0,250);// Blue
               text(" mV / div ",width/2+5, height-10);
                text(Scale,width/2-30,height-10);// print vertical scale on graph
               int pix=5;// increment the horizontal position by "pix" pixels after reading data pts
               text(" sec / div",width-50,height/2-10);
               text(width/pix/tics,width-65,height/2-10);
               text(month(  )+" / "+day(  )+" / "+year(  ),width-125,height-10);// Date Stamp
               text(hour(  )+" : "+ minute(  )+" : "+second(  ),width-125,height-20);// Time Stamp
                              }
               }
```

PROCESSING CODE – PART 3 of 5

```
// everything happens in the serialEvent( )
 // The data collection/display rate is set by the delay in the Arduino code
   int N=4; //the number of different analog signals to plot, can be up to 6
   int i=0;
   void serialEvent (Serial myPort) {
     String inString = myPort.readStringUntil('\n');// get the ASCII string
     if (inString != null)          {
          inString = trim(inString);//trim off any whitespace
          // convert to an int and map to the screen height
          float inByte = float(inString);
          inByte = map(inByte, 0, 1023, 0, height);
          fill(204); // same color as background color
          rect(50+(width/N)*i,30,60,10); //blank out previous displayed number
          stroke(253/(N-1)*i,253/(N-1)*i,253/(N-1)*i);
          fill(253/(N-1)*i,253/(N-1)*i,253/(N-1)*i);
```

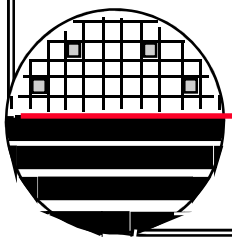


PROCESSING CODE – PART 4 of 5

```
// give names for plots
        String [ ] names={"Temperature","Humidity    ","Brightness ","Shock      "};
        print("A"+i+"  "+names[ i ]+"    ");//print to dialog box
        println(height-inByte);//print to dialog box
        output.print("A"+i+"  "+names[i]+"    ");//print to file
        output.print(month( )+"  "+day( )+"  "+year( ));// Date
        output.print("   "+hour( )+" : "+ minute( )+" : "+second( )+" “);// Time
        output.println(height-inByte);//print to file plus new line
        if (keyPressed == true){
        output.flush( );// writes the remaining data to the file
        output.close( );// Finishes the file
        exit ( );
          }
        // print text ( names, x position, y position)
        text(names[i],50+(width/N)*i,20);
        text(inByte*5000/height,50+(width/N)*i,40);
      //draw rectangle data point at x=xPos, y=height-inByte, size=2x2 pixels
      rect(xPos,height-inByte,2,2);
```

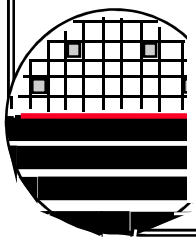
PROCESSING CODE - PART 5 of 5

```
if (i<N-1){
        i=i+1;
            }
        else {
          i=0;
          int pix=5;// increment horizontal by "pix" pixels after reading data pts
          xPos=xPos+pix;
             }
               }
  // at the edge of the screen, go back to the beginning:
    if (xPos >= width) {
          xPos = 0;
          background(204);
                 }
             }
             // all done with processing code
```



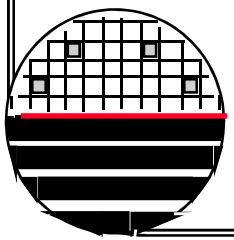
ARDUINO CODE

```
//Arduino code "ArduinoAnalogSendFuller.pde"
// Reads multiple analog signals connected to inputs A0...A3
// Sends ASCII-encoded strings out the USB serial port at 9600 baud
// Created 4 Dec 2010
// Updated 11 Dec 2010
// by Dr. Lynn Fuller, Professor, Microelectronic Engineering, Rochester Institute of Technology
// This code is in the public domain.
void setup() {
 // initialize the serial communication:
 Serial.begin(9600);
                }
void loop() {
 // send the value of analog input 0:
 Serial.print(analogRead(A0));
 // to send as an array [A0,A1,A2,A3]
 // use Serial.print(' ')between reads and Serial.println( )at end;
 // send the value of analog input 1:
 Serial.println( );
 Serial.print(analogRead(A1));
 Serial.println( );
 // send the value of analog input 2:
 Serial.print(analogRead(A2));
 Serial.println( );
 // send the value of analog input 3:
 Serial.print(analogRead(A3));
 Serial.println( );
 // wait a bit for the analog-to-digital converter
 // to stabilize after the last reading:
 // use the serial monitor 7th icon to see what sent
 delay(100); //this delay sets how often the analog data is sent (in milli-seconds)
                }
```



ARDUINO CODE FOR OSCILLATOR FREQUENCY

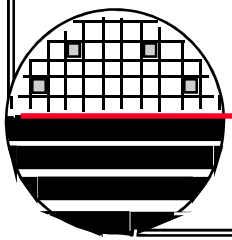
```
/*
  Capacitive Measurement Tool via RC Oscillation Circuit
  Measure frequency generatred by a 741 Op Amp and read
  the period using the Aurduino pulseIn() function
     Created by:    Dan Smith, Masters of Microelectrics Student
  Rochester Institute of Technology
  Version 1.0 | 06 Feb 2011
*/
int freqPin = 7;  //Input pin for frequency
int period = 0;   //Initilize period measuremtn to zero
int cap = 0;      //[pF]
int res = 10.3;   //[Mohm]
//Might want to make cap/res into long
void setup() {
  Serial.begin(115200);     //baud rate
  pinMode(freqPin, INPUT);  //set pin for input
}
void loop() {
  Serial.print("Period = ");
  period = pulseIn(freqPin,HIGH);
  //Reads time [us] for high square wave to go LOW
  Serial.print(period*2);
  //Only half the period, do double it
  Serial.println(" us");
  cap = period/(res); //T=2RC
  Serial.print("Capicator is: ");
  Serial.print(cap);
  Serial.println( "pF");
  delay(1000);
}
```



ARDUINO MULTI-SENSOR MOVIE

REFERENCES

1. Spark Fun Products, http://sparkfun.com
2. Arduino Home Page, www.arduino.cc
3. Processing Home Page, http://processing.org



HOMEWORK – MICRO CONTROLLERS

1. Modify the Arduino code to alarm (turn on a red LED) if one of the analog signals is above 4 volts.
2. Modify the Arduino code to increase the data rate if one of the analog signals is above 2 volts.
3. Modify the Processing code to place a small round indicator on the graph. It should be green if one of the analog signals is below 3 volts but red if above 3 volts.
4. Design an operational amplifier circuit to give a 0 to 5 volt analog output from a photodiode.