USING AN ARDUINO TO MEASURE FREQUENCY RESPONSE AND CURRENT-VOLTAGE DEVICE CHARACTERISTICS IN ELECTRONICS LABS

Steve Weis
Department of Engineering
Texas Christian University
s.weis@tcu.edu

Abstract

Most of our students learn electronics in lab; comparing their design and analysis work with lab measurements allows them to understand concepts more completely. Using the Arduino microcontroller to semi-automate repetitive measurements accomplishes two of our educational goals. First, since we are trying to accomplish as much as possible in lab - streamlining data-taking is helpful. Second, it allows us to employ a microcontroller in a useful way. As embedded microcontrollers become ubiquitous, we want to include them throughout our curriculum. There are several options for software control of the Arduino's data acquisition process. We chose to use MATLAB since we have it available and our students use it. Three example instrumentation circuits and their associated MATLAB scripts are presented: (1) measurement of the I-V characteristic of a diode, (2) measurement of the common source I-V characteristics of a MOSFET, and (3) measurement of a simple filter's frequency response.

Introduction

The Arduino has been extensively used as a platform for teaching engineering, science and technology concepts to undergraduates. Several papers have been delivered at ASEE conferences describing different approaches. At the 2011 ASEE Annual Conference, Recktenwald and Hall described using the Arduino to teach programming, design and measurement to first year students [1] and Bird described his work using Arduinos to teach a microprocessors course [2]. At the 2012 ASEE Annual Conference, Tremberger et al., described their work using the Arduino for student projects in a community college [3]. Hochgraf described his experience of using the Arduino to teach digital signal processing at the 2013 Northeast Section Conference in March 2013 [4]. There are also two excellent webinars provided by Mathworks that describe using MATLAB and the Arduino to teach mechatronic concepts [5, 6]. Following these examples, this paper shares a few ideas of how the Arduino might be used as a data acquisition device in a third-year electronics lab.

Most people *learn* electronics in the lab. Theory and circuit analysis are certainly important, but putting concepts to work in the lab seems to be the most meaningful to most students. Since there are so many things we wish to teach in the lab, using the Arduino to streamline some of the data taking is helpful. Other benefits of using the Arduino to acquire data include the ease of visual comparison with simulation and

analytically predicted results, and further integration of the microcontroller in the undergraduate engineering curriculum.

Although the Arduino's analog-to-digital converter resolution is only 10 bits with a dynamic range of 0 to 5V, it is sufficient for many undergraduate electronics lab measurements. The Arduino is certainly not a replacement for any of the conventional electronics laboratory equipment but is merely another tool that may be used to teach. The three lab examples included here are measurements of device current-voltage (I-V) characteristics and measurement of amplifier and filter frequency response characteristics.

We chose to use MATLAB (http://www.mathworks.com/hardware-support/arduino-matlab.html) with the Arduino since we have it available and the data is easily plotted and saved. However, there are several other Arduino-based means to acquire and store data and later plot it. One way is to use the Adafruit data-logging shield for Arduino (http://www.adafruit.com/products/1141) and an open-source plotting utility to plot the data stored on the SD card. Another way is to capture the data using a serial port data capture utility like RealTerm (http://realterm.sourceforge.net/), save the data and then plot it.

MATLAB and the Arduino

Giampiero Campa from Mathworks developed an Arduino Input/Output (IO) package that allows the user to program in MATLAB on the host PC and communicate with and control the Arduino. In his words, "Arduino + ArduinoIO package + MATLAB = inexpensive and interactive Analog and Digital IO from the MATLAB command line."

The Arduino IO Package contains the driver software to allow the Arduino to appear as a virtual serial port to the operating system and may be used to perform analog and digital input and output as well as motor control from the MATLAB command line. A script of command line commands can be written to use the Arduino to sample signal voltages and write them back to the host PC within the MATLAB environment.

The required software (MATLAB support package for Arduino - also known as ARDUINO IO) is freely available at http://www.mathworks.com/matlabcentral/fileexchange/32374. The package enables an Arduino connected to the computer via USB to perform analog and digital input and output from MATLAB. The readme.txt file within the package describes how to install the software.

The software installation process begins by using the Arduino IDE upload adiosrv.pde to the Arduino board. The adiosrv.pde (or srv.pde) is the "server" program that will continuously run on the microcontroller. It listens for MATLAB commands arriving from the serial port, executes the commands, and, if needed, returns a result. Running the install_arduino.m MATLAB script adds the relevant ArduinoIO folders to the MATLAB path and saves the path.

Once the software is installed, MATLAB scripts can control the Arduino and use it as a simple data acquisition device. Three example scripts and setups that may be used in electronics labs are provided in the next section.

Examples

Measurement of the I-V characteristic of a diode

The first example uses the Arduino to measure the I-V characteristic of a diode. This is a "bare-bones" script that requires the student to process the voltage data after it is captured. The student can modify the provided script to do the data processing or can use a spreadsheet or other plotting software later. A schematic diagram of the setup used is shown in Figure 1.

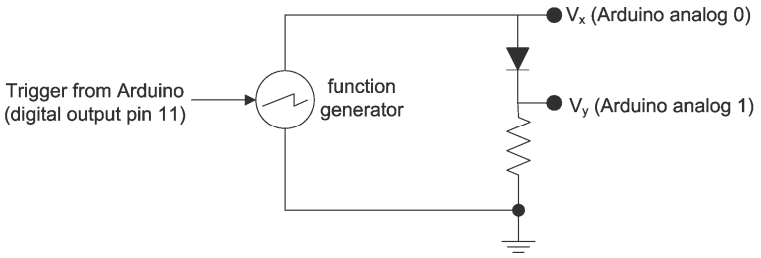

Figure 1: Diode I-V characteristic measurement setup

Figure 2 is a plot of the measured raw data much like that created as the points are measured.

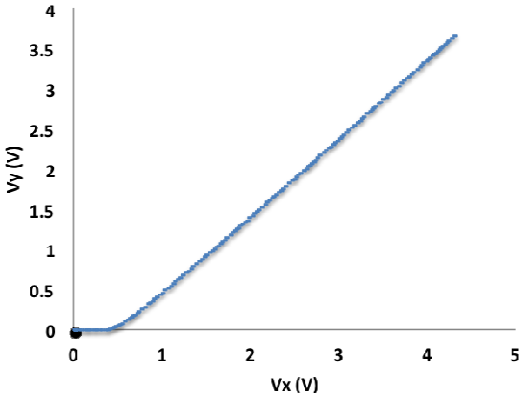

Figure 2: Raw data captured using setup shown in Fig. 1

The raw data is written to a text file in a comma-separated format. The user is prompted for the file name as part of the script.

To plot the diode's current-voltage characteristic from the raw data, the voltage across the diode and the current through it must be calculated. To determine the voltage across the diode, the user must subtract the voltage across the sampling resistor (Arduino analog 1) from the voltage across the function generator (Arduino analog 0). To calculate the current through the diode, one divides the voltage across the resistor (Arduino analog 1) by its resistance. This is easily accomplished using either MATLAB (with another script or by modifying the script below) or a spreadsheet.

The MATLAB script is provided in Appendix A.

I-V characteristics of an N-channel enhancement MOSFET

The second example uses an Arduino to measure the $i_D - v_{DS}$ characteristics of an enhancement-type NMOS transistor. This is a more complete script than the diode script; it calculates and plots the I-V characteristic as the points are being measured. A schematic diagram of the setup is in Figure 3.

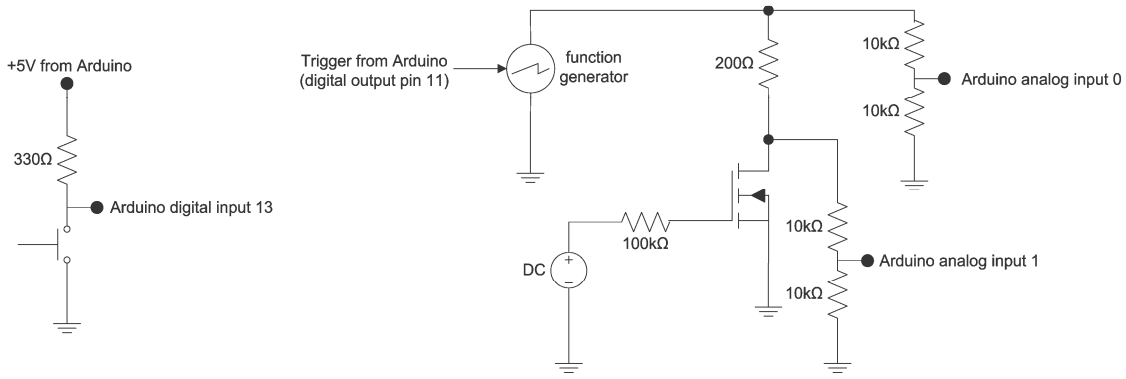

Figure 3: MOSFET I-V characteristics measurement setup

The two 10kΩ voltage dividers were necessary since the Arduino's input voltage range is limited to 0 to 5V and the signal of interest was between 0 and 10V. The resistor values were a compromise between loading the MOSFET under test and keeping the impedance "seen" by the Arduino analog-to-digital converter to less than 10kΩ as suggested in the ATmega48PA documentation.

The gate voltage is set using the dc power supply. The script prompts the student to adjust the voltage to a desired level and then push the button switch. When depressed, the PBNO switch pulls Arduino digital input pin 13 to ground. The function generator is then triggered using Arduino digital output pin 11.

A set of curves measured using the setup is shown in Figure 4. The gate voltages were 2V through 7V in 1V increments.

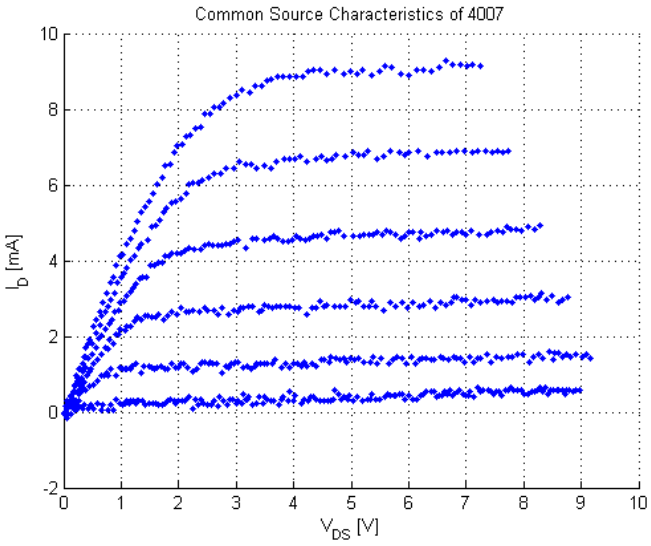

Figure 4: MOSFET I-V characteristics measured using the setup in Fig. 3

The script used to measure these curves is included in Appendix B.

Frequency response measurement

The third example uses an Arduino to measure the frequency response of a filter or amplifier. The Arduino is used to trigger the frequency sweep of a function generator that provides the filter/amplifier's input signal. The output signal is sampled, digitized, plotted and stored using the Arduino and a MATLAB script. A plot of the output voltage magnitude as a function of time is plotted during the data acquisition. At the completion of the process, the acquired data is replotted as a Bode magnitude plot. The sampling rate was approximately 20 samples/second.

A schematic diagram of the setup is shown in Fig. 5. The peak detector is composed of two op-amp buffers, a switching diode and a hold capacitor. The 1nF hold capacitor was chosen to optimize hold time for the sweep of 10Hz to 100kHz. A FET op-amp (LF412) was chosen to minimize leakage currents.

An Agilent 33220A function generator was used for the script presented here. The Agilent 33220A may be configured to provide a logic signal to its SYNC output terminal that goes from low to high at the beginning of the sweep and high to low at the end of the sweep. This signal was connected to digital input pin 12 of the Arduino to provide start and stop times and thus allow the calibration of the frequency component of the response data.

An alternate script was written for the Fluke Philips PM 5138A function generator that provides a voltage output that ramps from 0 to 10V during the frequency sweep via its "Sweep" output terminal. The ramp voltage was limited to 0 to 5V using a voltage divider and was acquired by the Arduino through an analog input pin. This allowed the calibration of the frequency component of the response data.

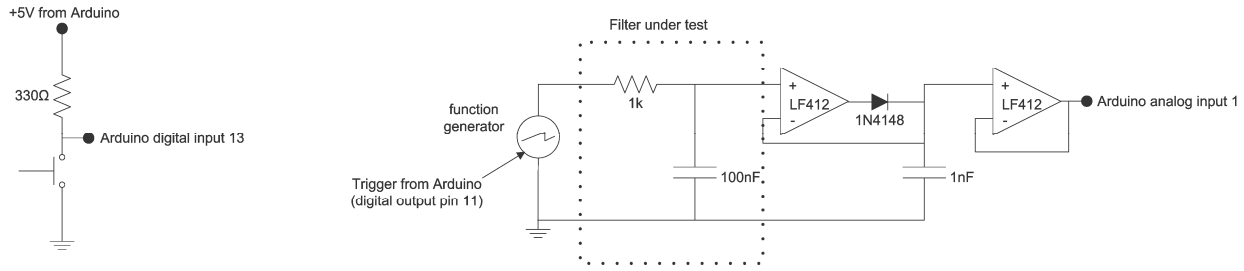

Figure 5: Frequency response measurement setup

A Bode magnitude plot of the measured frequency response of a low-pass RC filter is shown in Fig. 6.

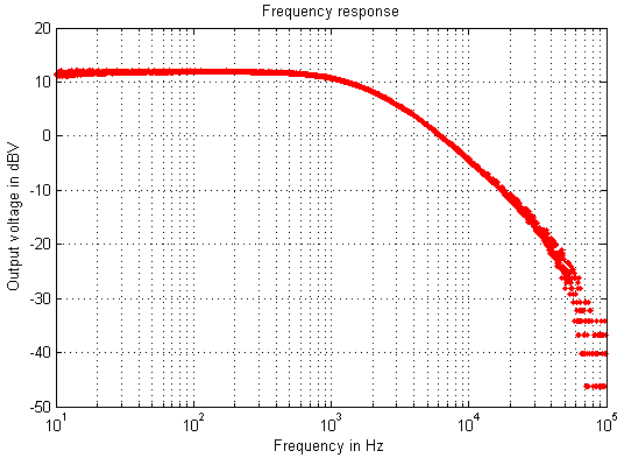

Figure 6: Low-pass filter frequency response measured using the setup in Fig. 5. Sweep time was 100 seconds.

The script used to measure the frequency response is provided in Appendix C.

Summary

The Arduino coupled with MATLAB provides a valuable addition to the electronics laboratory that will allow students to gain experience with a microcontroller, reduce some of the tedium of data acquisition, and allow the students to more easily plot their measured, calculated and simulated data.

References

[1] Recktenwald, G. W., and Hall, D. E., AC 2011-2642: "Using Arduino as a platform for programming, design and measurement in a freshman engineering

course," Proceedings of the 2011 ASEE Annual Conference, Vancouver, B.C., June 2011.

[2] Bird, N., AC 2011-437: "Use of the Arduino platform for a junior-level undergraduate microprocessors course," Proceedings of the 2011 ASEE Annual Conference, Vancouver, B.C., June 2011.

[3] Tremberger Jr., G., Armendariz, R., Takai, H., Holden, T., Austin, S., Johnson, L.P., Marchese, P.J., Lieberman, D.H., and Cheung, T., AC 2012-3977: Applications of Arduino microcontroller in student projects in a community college," Proceedings of the 2012 ASEE Annual Conference, San Antonio, TX, June 2012.

[4] Hochgraf, C., "Using Arduino to teach digital signal processing," Proceedings of the 2013 Northeast Section Conference, Norwich University, March 2013.

[5] Campa, G., "Learning Basic Mechatronics Concepts Using the Arduino Board and MATLAB," http://www.mathworks.com/company/events/webinars/wbnr43537.html

[6] Ferraro, D., "Teaching Mechatronics with Low Cost Hardware," http://www.mathworks.com/company/events/webinars/wbnr69522.html

Appendix A: Diode I-V characteristic measurement script

```
%% This is IV_meas
% The goal of this script is to retrieve two analog voltage inputs
% from the Arduino, then
% plot and save the data in a file. A button will determine start time.
%
% This is for measurement of current voltage characteristics
%
% Settings on the function generator: 0 to 5V ramp in 10 seconds
% triggered externally.
%
% Arduino analog 0 is tied to the voltage across both the device
% and the current sampling resistor
%
% Arduino analog 1 is tied to the node between the device
% and the current sampling resistor
%
% Arduino pin 13 is connected through a 330 ohm resistor to +5V and
% through a pushbutton switch (PBNO) to ground.
% When the switch is depressed pin 13 reads a logical 0
% and the Arduino triggers the function generator.
%
% Arduino pin 11 is tied to the external trigger input
$ of the function generator.

% Steve Weis, TCU Dept of Engineering

clear all;
%% create arduino object and connect to board
a=arduino('COM5');
%% setup
% specify pin mode for pin 13 that will be used for start button
a.pinMode(13,'input');
% specify pin mode for pin 11 that will be used to trigger fn generator
a.pinMode(11,'output');
%insure that the output is not floating
a.digitalWrite(11,0);
numpts=300; %number of points measured
xydata=zeros([numpts 2]); % preallocate
% ask for filename
xydataFileName=input('Filename for the x and y data? ','s');
%% basic analog and digital IO
disp('Push the button when you wish to start taking data');
dv=1;
% read digital input from pin 13
while dv
dv=a.digitalRead(13);
end
% send trigger to fn generator - trailing edge
a.digitalWrite(11,1);
pause(0.01);
a.digitalWrite(11,0);

for i=1:numpts
   xydata(i,1)= a.analogRead(0)/1023*5; %scaling factor, 5V full scale
```

```
   xydata(i,2)= a.analogRead(1)/1023*5;
   plot(xydata(:,1),xydata(:,2),'bo');  %plots blue points with
                                        % x =analog 0
                                        % y = analog 1
   grid
   drawnow;
end
dlmwrite(xydataFileName, xydata)
%% close session
delete(a)
```

Appendix B: MOSFET I-V characteristic measurement script

```
%% This is IV_meas for the MOSFET
% the goal of this script is to retrieve two analog voltage inputs
% from the Arduino, then
% plot and save the data in a file.
%
% A button will determine start time.
%
% This is for measurement of current voltage characteristics
%
% Settings on the function generator: 0 to 9.5V ramp in 10 seconds
% triggered externally
%
% Arduino analog 0 is tied to the voltage across both the device and
% current sampling resistor (function generator output)
%
% Arduino analog 1 is tied to the node between device and current
% sampling resistor
% (analog0 - analog1 is voltage across sampling resistor)
%
% Arduino pin 13 is connected through a 330 ohm resistor to +5V and
% through a pushbutton switch (PBNO) to ground. When switch is
% depressed pin 13 reads 0 and the Arduino triggers the function
% generator.
%
% Arduino pin 11 is tied to the external trigger of the function
% generator.

% Steve Weis, TCU Dept of Engineering


clear all;

%% create arduino object and connect to board
a=arduino('COM5');

%% setup

% specify pin mode for pin 13 that will be used for start button
a.pinMode(13,'input');

% specify pin mode for pin 11 that will be used to trigger fn
generator
a.pinMode(11,'output');

%insure that the output is not floating
a.digitalWrite(11,0);

numpts=150; %number of points measured

numcurves=6; % number of curves

RD=200; %drain resistance in ohms
```

```
xydata=zeros([numpts*numcurves 2]); % preallocate

% ask for filename
xydataFileName=input('Filename for the x and y data? ','s');


figure(1)
clf
hold on
grid on

xlabel('V_D_S [V]')
ylabel('I_D [mA]')
title('Common Source Characteristics of 4007')

 current_scale_factor = (1/RD)*2*1000*(1/1023)*5;
       %scaling factor, 5V full scale, 200 ohm R_D, current in mA,
       %splitter factor of 2

 voltage_scale_factor = 2 * 5 * (1/1023);
       %scaling factor, 5V full scale, splitter factor of 2


%% basic analog and digital IO
for curvnum = 1:numcurves
     offset = (curvnum -1)*numpts; % offset for data vector, one
long vector easier for Excel

    disp(['Curve number:  ',num2str(curvnum)]);
    disp('Push the button when you wish to start taking data');

    dv=1;
    % read digital input from pin 13
    while dv
    dv=a.digitalRead(13);
    end

    % send trigger to fn generator - trailing edge
    a.digitalWrite(11,1);
    pause(0.01);
    a.digitalWrite(11,0);

    if curvnum == 1
        numpts=numpts+25;
    end
    %this makes up for time lost setting plot when going through the
    %loop below the first time

    for i=1:numpts
       V_fn_gen = a.analogRead(0);
       V_device = a.analogRead(1);
       xydata(i+offset,2)= (V_fn_gen-V_device)*current_scale_factor;
```

```
       xydata(i+offset,1)= V_device*voltage_scale_factor;

% this if is to prevent miscalculation if the ramp voltage goes to
% zero between readings
      if V_device == 0
            xydata(i+offset,2)=0;
      end

       plot(xydata(i+offset,1),xydata(i+offset,2),'b.');
                 %plots blue points with x = voltage across device
                                  % y = current through device
       drawnow
    end
end
dlmwrite(xydataFileName, xydata)

%% close session
delete(a)
```

Appendix C: Frequency response measurement script

```
%% This is freq_sweep_response
% The goal of this script is to measure the frequency response
% of a filter or amplifier using the Arduino to do the A to D
% conversion.
% The digital data received by MATLAB will be plotted and saved
% in a data file.
%
% A button switch will determine start time of the freq sweep.
%
% Arduino analog 1 is tied to the buffered peak detector output.
%
% This script uses the Agilent 33220A function generator

% Steve Weis, TCU Dept of Engineering

clear all;

%% create arduino object and connect to board
a=arduino('COM5');

%% setup

% specify pin mode for digital pin 13 that will be used for start button
a.pinMode(13,'input');

% specify pin mode for digital pin 12 that will be used for detecting the
%                                       start and end of the freq sweep
a.pinMode(12,'input');

% specify pin mode for digital pin 11 that will be
%                                             used to trigger fn generator
a.pinMode(11,'output');

%insure that the output is not floating
a.digitalWrite(11,0);

% ask for filename
xydataFileName=input('Filename for the frequency response data? ','s');

% ask for starting frequency
f_start=input('Start frequency? ');

% ask for stop frequency
f_stop=input('Stop frequency? ');


%% analog and digital IO
disp('Push the button when you wish to start taking data');

dv=1;
% read digital input from pin 13
while dv
dv=a.digitalRead(13);
end
```

```
% send trigger to fn generator - trailing edge trigger
a.digitalWrite(11,1);
pause(0.01);
a.digitalWrite(11,0);

% read digital input from pin 12 (sync pulse from generator)
sync = 0;
while (sync == 0)
    sync=a.digitalRead(12);
end

i=1;
while sync
   xydata(i,1)=i;
   xydata(i,2)= a.analogRead(1)/1023*5; %scaling factor, 5V full scale
   plot(xydata(:,1),xydata(:,2),'b.');  %plots blue points with x =i
                                        % y = analog 1 input
   grid
   drawnow;
   sync=a.digitalRead(12);
   i=i+1;
end

%% close session with Arduino
delete(a)
disp('all done with Arduino!');

%% calculate frequencies, plot and save

numpts = i-1;

D=log10(f_stop/f_start); % # of decades

multiplier = 10^(D/numpts);

f(1)=f_start;
for j=2:numpts
    f(j)=f(j-1)*multiplier;
end

% plot Bode plot
semilogx(f,20log10(xydata(:,2)),'r.');
grid
title('Frequency response')
xlabel('Frequency in Hz')
ylabel('Output voltage in dBV')

% save output file
fvdata=[f(:) xydata(:,2)];
dlmwrite(xydataFileName, fvdata)
```