

Name/NetID:

Teammate/NetID:

EXPERIMENT #4: Arduino as oscilloscope

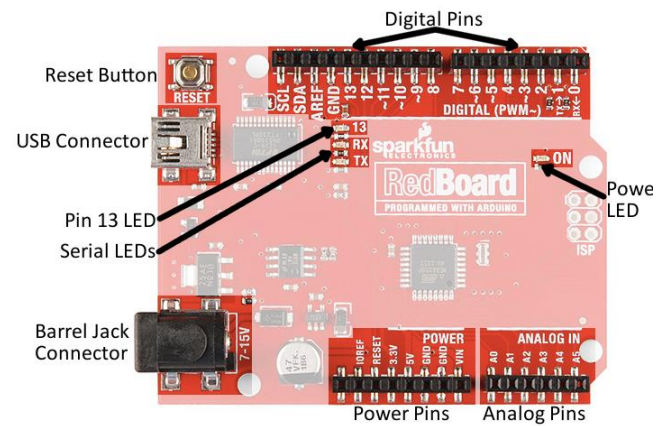
Notes:

Laboratory Outline

This module adds the ability to visualize time-varying signals to your portable bench equipment. If you have completed the module that shows the different ways you can use the Arduino/RedBoard as a voltmeter you know that the Arduino USB interface with the computer works both ways. Data can be sent back to the computer and you can watch a list of numbers appear in a window called the *Serial Monitor*. If the voltage actually varies more quickly and you want to capture and visualize the signal – like an oscilloscope – you can use a couple of methods. The first method, the brute force method has you cutting and pasting the data into either Excel or MATLAB. Recently, an additional feature was added to the Arduino IDE that allows you to visualize the serial data called the Serial Plotter. Rather than print out the values they are plotted.

The Analog Inputs on the Arduino

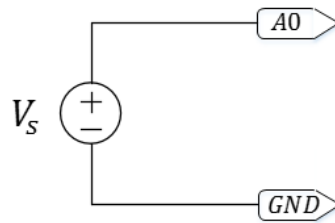
This module uses the same 6 pins labeled Analog Pins. Again, these pins accept an analog voltage – that can be time-varying – and convert the continuous voltage into a number from 0 – 1023 every 100 microseconds. Let's see if we can plot this data so that we can visualize the signal in MATLAB or Excel.




Retrieving Serial Data

Start with exactly the same set up as the module that uses the Arduino/ReBoard as a voltmeter.

- ✓ Set the power supply to +5V.
- ✓ Hook-up the power supply to the analog pin labelled A0 as shown in the schematic below. The arrow shapes indicate which pin on your Arduino/RedBoard to connect to the positive and negative terminals of the power supply.



- ✓ Enter the code that outputs either the 10-bit integer from the A/D or the computed voltage to the Serial Monitor. For the observant – yes there are examples of both under *File > Examples > 01.Basics*.
- ✓ After checking under the **Tools** menu that the software knows which board you are using (the RedBoard is a clone of the Arduino Uno) and which COM port you are using. When you plug the USB cable into the lab computer the associated COM port is usually the highest numbered port. For Mac users the USB communication ports are the device file names. Upload the code to the board by clicking the  icon at the top of the window.
- ✓ Open the Serial Monitor.

The problem now is how to get the data that is streaming from the Arduino/Redboard to the computer in a format so that you can plot the data.

The Brute Force Method

This method uses the very useful cut and paste feature included in the Arduino IDE. The data streaming to the Serial Monitor now is scrolling very quickly. Selecting the data you want to extract would be difficult. There are two ways to make this easier: i) slow down the display rate, and ii) specify the number of samples to print.

Slowing Down the Display Rate

The program that you loaded to the Arduino may include a statement `delay(int)`; where `delay` is a statement that uses the parameter `int` to suspend the program for `int` microseconds where `int` is an *unsigned long integer* value. Unsigned means that the value must be positive so that the extra bit for the sign is not needed. Long means that the integer is 32 bits rather than 16 bits long providing a range of 4,294,967,295 ($2^{32} - 1$).

- ✓ Change the delay statement to `delay(100)`;

Question 1: How many hours of delay can be specified?

Limiting the Total Number of Points Measured

Limiting the number of data points sent to the Serial Monitor can also be a useful method to slow down the data rate. To do this you need to add a couple of statements to your code. There are many ways to program this feature using the Arduino IDE so the method described below was chosen so that you can see how to use a class of programming functions called **conditional** statements. These statements are designed to execute ONLY when certain specified condition(s) are met.

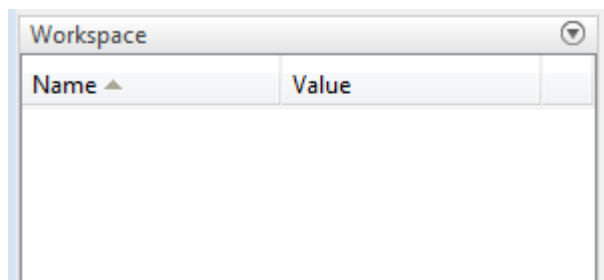
In the code:

1. Insert the statement `int length=100;` **before** the `setup{}` loop. This statement declares that an integer variable named `length` is created and can be used in the `setup{}` **and** `loop{}` portions of the code as well as any functions created. This type of variable or is termed a global variable because it can be accessed and changed from anywhere in the code. `length` is initialized to 100.

2. In the *loop* section insert a `while(condition){code segment}` statement. The condition inside the parentheses tells the Arduino when to do all of the statements that are between the curly brackets {}. Put `while(length>0){` just after the `loop{` statement.
3. Close the curly brackets so that the while statement includes ALL of the statements in the *loop* section.

Question 2: At this point nothing about the program has changed functionally so you can upload it, run it while varying the voltage and open the serial monitor to make certain. Now, insert a statement inside the while loop that will make the board take 3 data points, write them to the serial monitor and then do nothing. Hint: you will want to modify the variable `length` so that it changes. Eventually, the variable `length` will stop satisfying the condition (`length>0`). Print out or write down the code used.

- ✓ Paste them into Excel
 1. Open Excel and bring up an empty spreadsheet.
 2. Go to the serial monitor with should have 3 data points in it and highlight these points. Cntrl + c to copy them to the computer's clipboard.
 3. Click the first cell of the second row of the spreadsheet and Cntrl + v to paste the data.
 4. Starting at 0 you can enter the relative approximate time each point was taken since there is a .1 s delay before the loop section repeats.
- ✓ Paste the into MATLAB
 1. Open MATLAB.
 2. Find the workspace section – mine looks like this...



3. Click on the down arrow in the upper right corner and choose New or use Cntrl + n to have a new variable created. It shows up as a box where you can give it a name – anything you want. Pressing return completes the action.
4. Double-click on the new variables name to open up a window that looks a lot like a spreadsheet.
5. Click on the first cell and press Cntrl – v to paste the values into the new variable. Clicking the x in the upper-right corner minimizes the variable data window. You can double-click on any variable to view and/or modify its contents.
6. In the command window type `time=0:timeIncrement:NumPoints*timeIncrement` where `timeIncrement` is the time between sample in this case 100 ms or .1 s and `NumPoints` is the total number of data points which you specified as 3. Another method is to type the statement `time=linspace(0,.2,3)`

Now you have data in either Excel or MATLAB. As with all devices that sample time-varying signal there is a limitation on how fast the measuring device can sample. From the data sheets and forums the claim is a sample every 100 microseconds.

- ✓ In the Arduino code remove the delay function and set the number of samples to 100.
- ✓ Make certain that the signal generator is in HIGH Z mode – ask your TA for help checking this.
- ✓ Set the signal generator to output a sinewave with a frequency of 10Hz, peak-to-peak amplitude of 4V and an offset of 2V. IT is VERY IMPORTANT that the signal does not go negative.
- ✓ Using a BNC-to-BNC cable connect channel 1 of the oscilloscope to the signal generator and be certain that the sine wave stays positive by adding an offset.
- ✓ Disconnect the oscilloscope and connect the signal generator to the Arduino using a BNC-to-banana cable and the special termination wires. The black connector is connected to GND and the red is connected to pin A0.

Question 3: Set the Arduino to collect 100 points while varying the voltage at the power supply. Get the data into either Excel or MATLAB and make a plot of the data – you do not know the time increment so the x-axis corresponds to the numbers from 1 -100 for now.

Question 4: Find the periodicity of the sine wave and count the number of data points in one period. You can do this for several periods and average. Use the number of data points per period to determine the unadulterated sampling rate for a single analog input pin.

The number you got is not even close to $100\mu\text{s}$ but for slowly varying signals like most of the signals you will encounter when connecting the Arduino's analog input pins to the sensors in your kits it should be fine. The slow speed comes from several sources including the fact that there is only one A/D on the processor chip so all of the analog pins need to share. There are ways to increase the speed if needed that are not too difficult if you are comfortable with assembly and interfacing with Python or MATLAB.

Adding a Time-Stamp

A number corresponding to an onboard reference can be obtained and printed out to the serial monitor so that you know, approximately, when each sample was taken.

- ✓ Add the statement `String toprint;` in the statements before the setup section.
- ✓ Add the following statements just after the statement that reads the analog value so that the loop section looks like the code below.

```
void loop() {
  while(length>0){
    length=length-1;
    // read the input on analog pin 0:
    int sensorValue = analogRead(A0);
    //format the string that is output to the Serial Monitor
    toprint = " ";
    toprint += micros();
    toprint += ",";
    // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
    voltage = sensorValue * (5.0 / 1023.0);
    toprint += voltage;
    // print out the value you read and a time stamp giving a relative time
    Serial.println(toprint);
  }
}
```

- ✓ Upload and run the code, then open the serial window.
- ✓ Copy and Paste the data into Excel or MATLAB

NOTE: Notice the statements that print out the timestamp then a comma then the voltage. This is an alternative to using the Serial.print and Serial.println statements exclusively. The code below will print out the same information to the Serial Monitor.

```
int sensorValue = analogRead(A0);  
// Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):  
float voltage = sensorValue * (5.0 / 1023.0);  
// print out the value you read and a time stamp giving a relative time  
Serial.print(micros());  
Serial.print(',');  
Serial.println(voltage);
```

Question 5: The data should include two numbers. The first is a time stamp or a number provided by the processor indicating the number of milliseconds from an arbitrary starting point. We are just interested in the interval. Plot the data using the time stamp as the x-axis. Is the interval between samples the same as the value you computed in Question 4?

- ✓ Remove the statement that computes the voltage and print out sensorValue instead of voltage.
- ✓ Upload and run the program.

Question 6: Did the removal of this statement change the sampling rate of the signal. You can prove this either by inspection of the data or by plotting it.

Question 7: (Optional +2 points) Using only the *setup()* section and a different conditional statement - the *if* statement. Again, take 100 data points and print the results to the serial monitor. Provide the code and an explanation of your code. If you do not know the syntax or functionality of the *if* statement refer to the reference section on the Arduino website.

Using the Serial Plotter

The Arduino IDE has a serial plotter that has the same functionality as the Serial Monitor except it is a plot of the data rather than a list.

- ✓ Set the frequency of the sine wave provided by the signal generator to be 1Hz. Don't change the amplitude or offset.
- ✓ Make a copy of the code that writes out a voltage and a time stamp. Using this copy remove the *while* statement and references to the variable *length* so that the code writes to the serial monitor continuously.
- ✓ Upload the code and open the Serial Monitor.
- ✓ Under the menu heading **Tools** open the Serial Plotter. You should get an error – the Serial Monitor and the Serial Plotter cannot be open together. Close the Serial Monitor and open the Serial Plotter.

Question 8: What is being plotted? Why can't you see the sine wave?

- ✓ Close the Serial Plotter and re-open the Serial Monitor. The Serial Plotter will plot up to 6 variables if the each value is delimited by a comma or a space. To see the sine wave you need to remove the time stamp because the Plotter is also plotting the time stamp whose value is so much larger than the voltage.
- ✓ Modify the code to **ONLY** print out either the voltage or sensor value.

Question 9: Use the Snipping tool to grab a picture of the sine wave (do your best).

Question 10: The Serial Plotter has some serious limitations. List at least 2. Hint: time axis? You can browse some of the websites and blogs where the limitations are lamented. So close and yet so far.

Without some work it seems that the Arduino analog pins and sampling is not very useful for high frequency signals. But for our purposes – observing how a sensor responds to a stimulus or how the signal you will use to control the turning on and off of the motor – it is fine. This brute force method is a simple procedure to capture what you observe in the serial monitor. Including visual aids will help with documentation of the different things you try on your own when evaluating sensors and sub-circuits to use or not – it is important to include all the things you tried in your final design project. What if you want to pick up where you left off?