BioE 101 Lab 0

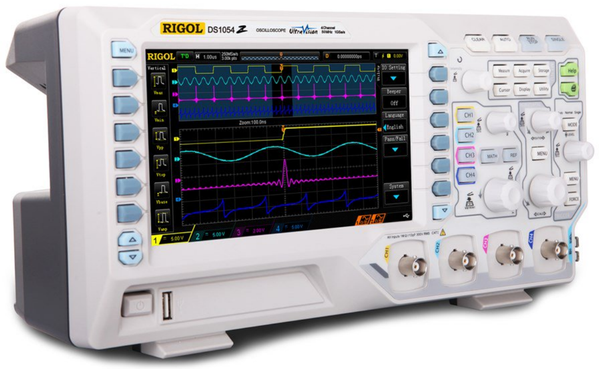
Table of Contents

I. Test Equipment

In an electronics lab, there are several pieces of equipment that are used for testing and validation. These include a power supply, signal generator, multimeter, and most importantly, an oscilloscope. These are the tools of the trade for debugging.

Oscilloscopes

An oscilloscope, when broken into its root words *oscillo* and *scope*, literally mean ”wave measure.” It is arguably the most important test equipment for analog engineers because a lot of signals we measure are time varying signals.

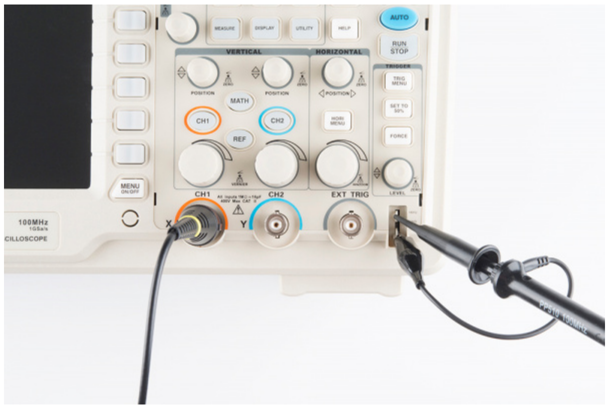


Why Use Oscilloscopes?

The oscilloscope allows us to zoom-in in time as well as amplitude so we can see small and fast signals. How fast of a signal we can see depends on the scope's bandwidth. In our lab, the scopes are useful up to 10s of MHz which is more than enough to analyze biomedical signals, but is super slow for some engineers who might need to see signals in the hundreds of MHz or even GHz range. For that kind of speed you will need specialized RF equipment such as a spectrum analyzer or network analyzer.

How To Use an Oscilloscope

To debug and test your circuits, you'll need to use oscilloscope probes to connect the test point on your circuit to the scope. As cautious debuggers and designers, we always want to eliminate any variable that might cause our circuit or system to not work. This is why we should first test our probes on the test points (usually square rings located on the bottom left of the scope). The connection should look like this:



Now to view the probe test results (which is a square wave), you can do one of two things.

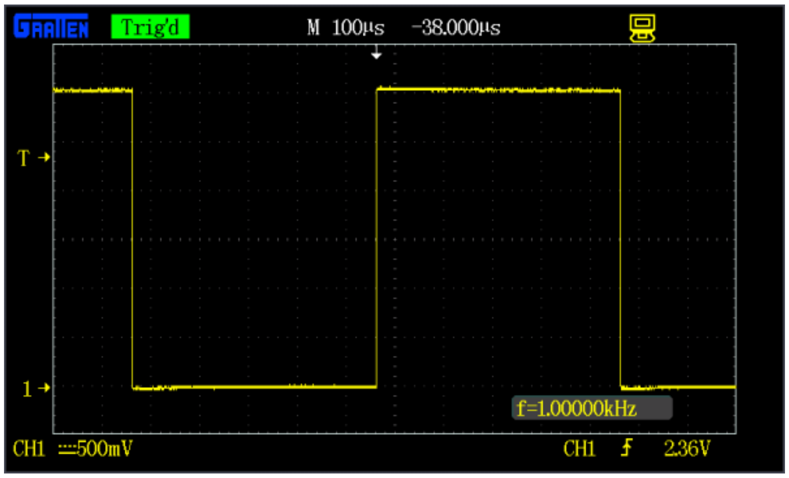
You can either:

- Press the **CH** button of the channel you're using (make sure it's lit), usually CH1
- Set the channel to **DC coupling**
- Adjust the **Volt/Div** knob until you can reasonably see the signal
- Adjust the **Time/Div** knob until you can reasonably see a period or two of the square wave

OR press the the **Auto** or **Auto-Adjust** button, which will automatically setup the scope so you can view the signal.

Obviously the second option sounds really nice to use all the time, but Auto-Adjust can sometimes make it so you won't be able to see some important features of your signal, so don't rely on it!

If you've done it right, you should see something like this:

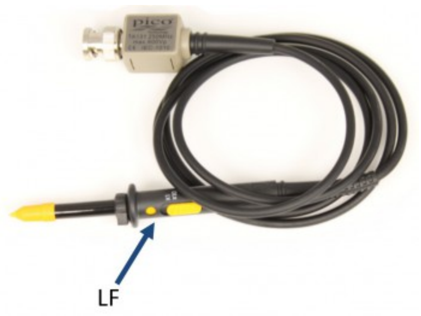


If your signal has a high DC component but you want to only look at its AC components, then you can set the scope to **AC coupling**.

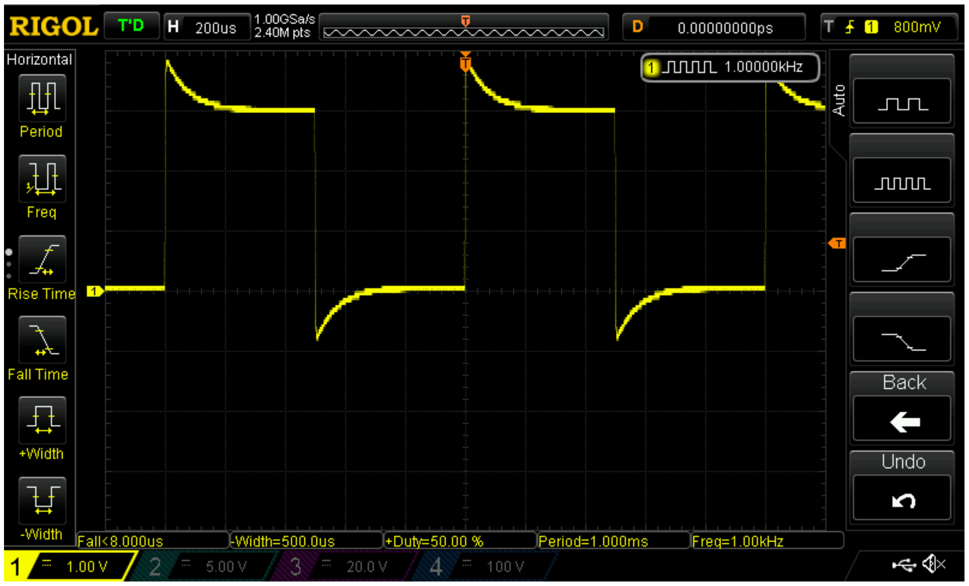
Oscilloscope Probes

There are two types of probes: active and passive - the ones we'll be commonly using in lab are passive. There are also 1x, 10x, and 100x probes that attenuate your signal by that amount.

There is also a place to tweak the probe located on the side of the probe. It might look like this:



This is used for what's called *probe compensation*, which you would do when testing your probe on the square wave test point on the scope. Untuned probes might cause the square wave test to look distorted like this:



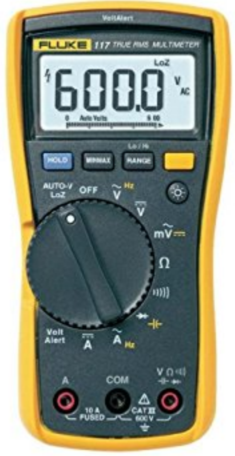
Fall<8.000us -Width=500.0us +Duty=50.00 % Period=1.000ms Freq=1.00kHz

To fix this, use a screwdriver to tune the probe until the square wave test yields a satisfactory result.

Multimeters

Multimeters are an invaluable tool for when you need a quick measurement or validation without having to to take the time to set up the scope.

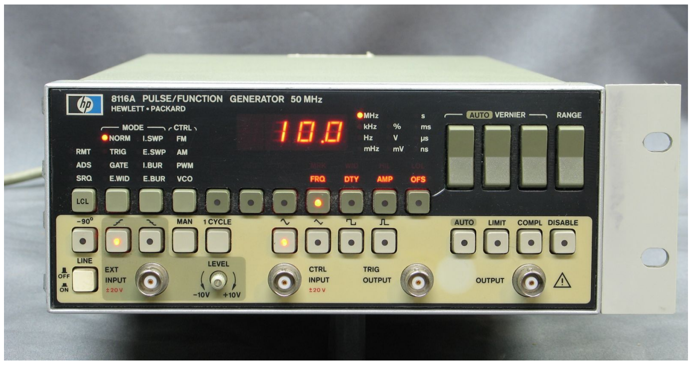
One of the most useful functionalities is the multimeter's continuity test, or short test. I usually refer to this test as "ringing the circuit out" since during this mode, the multimeter will ring or



beep if there is a short circuit between the points that you touch with the two test leads. Testing if there is a short somewhere in your circuit without having to actually turn it on prevents the parts from frying, since short circuits draw large currents. A *short circuit* is when there is a low resistance connection between two points. If these two points have a non-zero voltage difference between them it could causing large amounts current to flow due to Ohm's law.

Signal Generators

Signal generators (also called function generators) create various voltage waveforms of different shapes, amplitudes and frequencies. For example, I can create a 10 kHz sine wave with 1V peak-to-peak amplitude. This is invaluable for testing how our circuits behave at a certain range of frequencies and amplitudes.



Power Supplies

As the name suggests, power supplies supply power to our circuit. We can specify either the voltage or current to our circuit. One very important thing to do before hooking the power supply up to our circuits is to limit our current!!! This is so that in the event of a short circuit, the maximum current that the supply can deliver is the current limit that was set, by default most supplies can deliver 2-3 Amps of current, which can definitely fry your circuit and you!

II. Microcontrollers

Microcontrollers are small computers that have programmable input/ouptut. They contain a CPU, some memory, and Analog-Digital Converters (ADC) for inputs Digital-Analog Converters (DAC) for outputs. We will be using the built-in ADC's of our microcontroller units (MCU) to receive our signals.

Microcontrollers vs. Oscilloscope

Oscilloscopes are nice! Why don't we just stick to only using them? What's nice about MCUs is not only that receive our signals like scopes can, but we can program the microcontroller to process our signals or send them to software that can (like MATLAB or Python).

Arduino

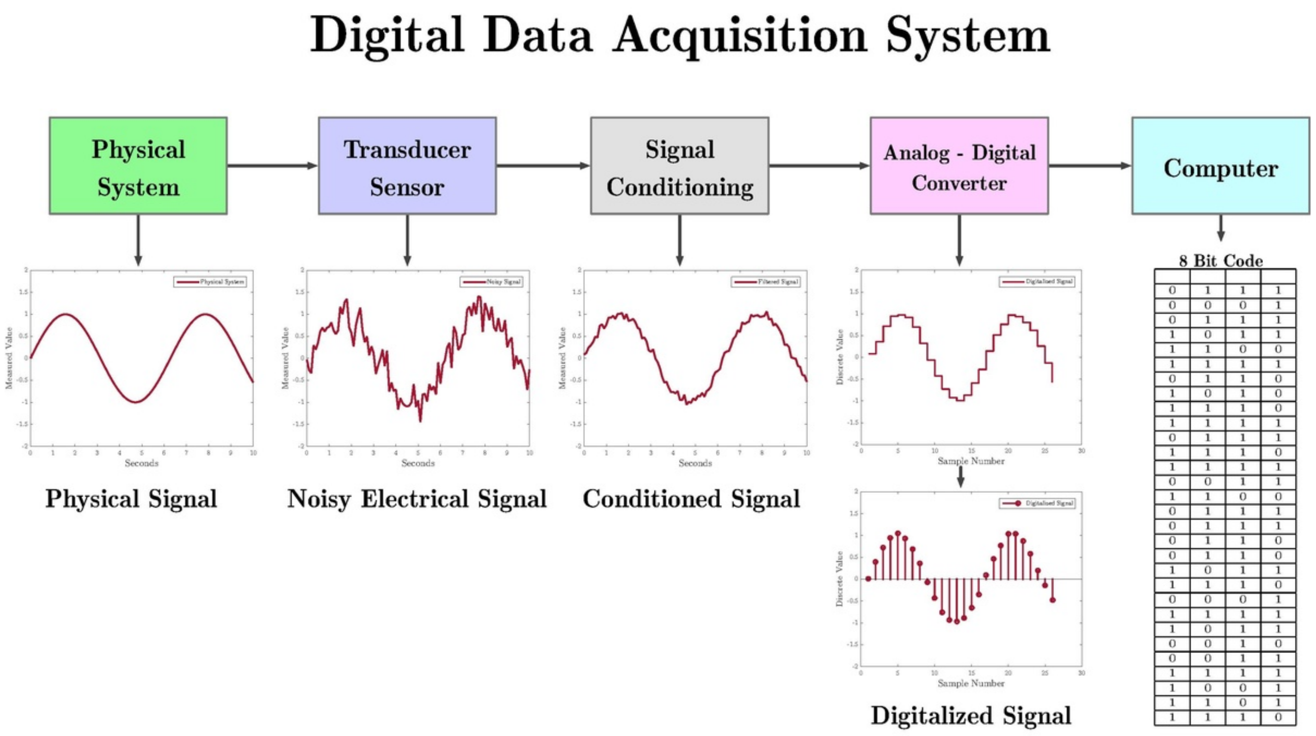
The Arduino is a great entry-level electronics hobbyist tool for easy data acquisition and processing. It's IO (input/output) pins have a built-in Analog to Digital Converter (ADC) with 10-bits of resolution for the Uno and 12-bits for other boards. It is programmed by uploading code to the device via the Arduino IDE. As it is intended for entry-level, it can't really be used for high-performance applications such as sampling a signal that has 100 MHz bandwidth. That would require a beefier (and more expensive) microcontroller and ADC system.

Arduino Installation

To install the Arduino IDE and drivers, head to https://www.arduino.cc/en/Main/Software and download the appropriate installer.

III. Signal/Data Acquisition

In BioE 101, we'll be focused on learning what's called a *Data Acquisition System* (see diagram below).



0	1	1	1
0	0	0	1
0	1	1	1
1	0	1	1
1	1	0	0
1	1	1	1
0	1	1	0
1	0	1	0
1	1	1	0
1	1	1	1
0	1	1	1
1	1	1	0
1	1	1	1
0	0	1	1
1	1	0	0
0	1	1	1
0	1	1	1
0	1	1	0
0	1	1	0
1	0	1	1
1	1	1	0
0	0	0	1
1	1	1	1
1	0	1	1
0	0	1	0
0	0	1	1
1	1	1	1
1	0	0	1
1	1	0	1
1	1	1	0

The **Physical System** is usually the biomedical signal we want to measure (e.g., an EKG signal). The **Transducer Sensor** is the device used to convert (transduce) the biomedical signal into a voltage. These include EKG electrodes, pulse oximeters, accelerometers, etc. **Signal Conditioning** is where we amplify and filter our signal to clean it up and make it nice and pretty to send to our **Analog-Digital Converter**, which digitizes the analog signal into discrete voltage levels that translate to binary for the **Computer** to read and analyze.

Analog to Digital Conversion

ADCs have a set resolution (number of bits) and full-scale (voltage range). Since most biomedical signals are small (in the microvolt to millivolt range), we have to amplify our signal until it is reasonably within the voltage range that the ADC operates in. You'll learn more about ADCs soon.

IV. Scientific Computing/Digital Signal Processing

We've finally captured our signal digitally. Perfect! Now our jobs might seem over, but there is so much more to be done with our data.

Digital Signal Processing and Machine Learning

There are a lot of digital signal processing (DSP) techniques we can employ. We can transform our data from the time domain to the frequency domain. We could apply digital filters to our data. We could also use learning/regression techniques to make decisions or classify our data.

I'm at the risk of sounding buzzwordy, *but* we are the domain experts of our newly acquired biomedical data - this means that any features of the data that can be extracted during processing might make more sense to us since we know its physical origins and we can take advantage of that when employing machine learning techniques. Examples of this include trying to classify patients' EKG data to predict certain heart conditions or performing analytics on a time series of patient data.

Anaconda Installation Instructions

For this class, we'll be using Anaconda to do all this. If you'd like to download it to your computer yourself, please use the Python 3 version. If you don't like the thought of Anaconda messing with your Python path, you can also use virtualenvs to install the same packages that Anaconda has.

To install Anaconda, download the installer from https://www.anaconda.com/download/

Check to see if Anaconda is installed correctly by opening up a terminal or command prompt and type `python --version`. Check that it is some version of Python 3 (ideally 3.5 or later) and that the version is Anaconda.

It should look something like this (don't worry if your Python version is not exactly the same):

```
~$ python --version
Python 3.6.3 :: Anaconda custom (64-bit)
```

Also download and install the packages PyAudio, PySerial and PyQTGraphs.

To install PyQTGraphs, head to http://www.pyqtgraph.org/ and download and install the appropriate installer for your OS. To verify that the package has been installed correctly, open up the python interpreter by typing `python` in your terminal and run `import pyqtgraph`. If the interpreter goes to a new line you should be good.

To install PySerial, type in `conda install pyserial` in your terminal. To verify that the package has been installed correctly, open up the python interpreter by typing `python` in your terminal and run `import serial`. If the interpreter goes to a new line you should be good.

To install PySerial, type in `pip install PyAudio` in your terminal. **For MAC users**, to install PyAudio you must install homebrew at https://brew.sh/ and then follow the instructions at https://stackoverflow.com/questions/33851379/pyaudio-installation-on-mac-python-3. To verify that the package has been installed correctly, open up the python interpreter by typing `python` in your terminal and run `import pyaudio`. If the interpreter goes to a new line you should be good.

If you've done everything correctly you should be able to open the interpreter and import all three modules like this (if your version isn't exactly the same that's okay):

```
Quincy@QUINCY-Y510P ~ $ python
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:27:45)
Type "help", "copyright", "credits" or "license" for more information.
>>> import serial
>>> import pyaudio
>>> import pyqtgraph
>>>
```

Common Installation Issues

Windows: After you download and install Anaconda, if typing in `python --version` or `conda --version` doesn't work and it tells you something like "command not found" this might be a PATH issue. Search in the start menu for "Edit the Environment Variables", click "environment variables" if needed, and head to PATH under either user variables or system variables. Add `C:/.../Anaconda3` and `C:/.../Anaconda3/Scripts` to the PATH. Replace ... with the actual folder path to your Anaconda installation. For example, mine is `C:/Users/Quincy/Anaconda3`

Mac: When installing PyAudio, a lot of students had errors when trying to use pip install. To fix this, install homebrew at https://brew.sh/ and then follow the instructions at

https://stackoverflow.com/questions/33851379/pyaudio-installation-on-mac-python-3.

V. Pre-Lab Report Questions

Try to answer all of these. You should be able to answer most by using CTRL-F in the lab document, but feel free to also use Google or other resources. There won't be anything to turn in for this week's lab, however.

1. Why do electronics engineers need to use oscilloscopes?
2. What does a 10x probe do?
3. What should I always do before using an oscilloscope probe?
4. My square-wave test looks kind of distorted, what should I do?
5. I have a large DC on my signal but I only want to view its AC components, what should I do?
6. Why do we bother with microcontrollers instead of just using oscilloscopes?
7. What is the resolution of the Arduino Uno's ADC?
8. Why do we need to amplify signals before digitizing? That is, why don't we just stick the ADC right after the sensor?
9. What are some examples of how we can process our data?

V. Lab Procedure

Sometimes the best way to get proficiency in electronics is to just go ahead and do something. Today's procedure will be short and sweet. There will be no post-lab report. Cables (BNC, banana plugs and USB) are in the back corner of the room near all the components.

- Follow the insrtructions in the Arduino section (II) to install the Arduino IDE.
- Follow the instructions in the Python section (IV) to install Python. Be sure to check the common issues section if you experience any bugs in installation. If your bug is not in the common issues, call over a GSI.
- Perform a squarewave test with an oscilloscope probe and view a nice square wave.
- View a 100mV, 10kHz with 1V DC offset sinewave using the signal generator on the scope. For the older function generators, the output is the right most BNC connection. The best way to connect the function generators to the scope is with a BNC-BNC cable.
- Measure 3V from a power supply using a multimeter. These are old power supplies that cannot set a current limit so be very careful when turning them on/off and turning the knobs to set a voltage (the knobs can be sensitive, causing you to overshoot the voltage you want to set).
- Connect your Arduino to your computer. In the Arduino IDE, open up the example sketch under Files > Examples > Basics > Blink and blink the LEDs at 200 ms intervals. Under Tools > Board, choose Arduino/Genuino Uno. Under Tools > Board, choose the one that has COMX (Arduino/Genuino Uno) where X can be any number. Then click the arrow button next to the check mark to compile and upload the sketch onto the Arduino. Play around with the delays and the process of uploading a sketch to get comfortable with the IDE (however, you're not expected to know how to code up a sketch, all sketches will be provided to you).
- Now upload the adc_sampling.ino sketch (in bCourses Lab 0 folder) to the Arduino. Apply a the 200mV, 100 Hz, 1V DC offset sinewave to pins A0 and GND of the Arduino and view the waveform in the serial plotter (which you can access via Tools > Serial Plotter). You may want to use jumper wires located in the component shelf area as well as the BNC to alligator clips cable to connect to pin A0 and GND.

Feel free to do any of these out of order and then get checked off by the GSI.