

ME220 Lab #2 Experiments with a Phototransistor

Due: 4/22/08, before 5:00 pm

Goals of this Lab:

We'll be hooking up a phototransistor (light sensor) and an Infrared (IR) light-emitting diode (LED). Some of you may have used these before. The signal will be transmitted from the IR LED to the phototransistor. There is some building to do, and the main things that are new here are the generation of physical output waveforms for the LED using LabView and gathering input signal waveforms, generated by the phototransistor into LabView.

Once all the "hardware" and "software" of this lab is running, we'll do some experiments to measure the time response of the phototransistor/LED combo. In the time domain, we'll see that there are issues with the background electrical noise and sunlight.

In the frequency domain, we'll see that it is possible to separate an oscillating signal from the background and do some measurements that are a bit less influenced by the background.

In the report attach all spectral plots (frequency domain) along with a brief description.

Additional note: On some of the exercises when you're collecting data, you may run into the very real problem of running out of computer memory (especially in the last section where you want to collect data for 50 seconds at 10000 samples/second. If this happens, make a note of it and try a shorter time that doesn't cause your computer to complain.

Acknowledgement

Zachary Nelson of National Instruments orchestrated a donation of the LabView Software and the National Instruments Data Acquisition hardware that we'll be using in the lab this quarter. Zach may visit sometimes during the quarter, so please be nice to him!

Part 1 – Build some hardware

We're going to use the NI Analog Output Channels to turn an IR LED on and off. Unfortunately, the output current from the output channel (5 mA) is not enough to do this. We need more current!

So, we're going to use the analog output of the NI card as a logic signal to control a larger current with a Power Driver (DS3658 – familiar to people from 210 or 218). The first thing to do is to build the following circuit on your protoboard, or breadboard, and make connections to the connector block and the power supply. If you haven't used a breadboard before, see the tutorial in the "links" section of the ME220 web page.

The phototransistor is a LTR-3208E (The part with a dark black package—the data sheet is posted on the web site). It is a pretty nice device that responds quite linearly (see Figure 4 on datasheet) That is, more current can flow through the transistor at 5V with increase in irradiance. It does saturate at higher light levels, but we shall leave that and other limits of the phototransistor for another lab assignment.

The IR LED is the one in a clear package. It has a Forward Voltage drop of about 1.6-2V. Aside from the protoboard and the wires, the rest of these parts will be provided in an envelope at the lab bench. Go through the connections as shown in the figure. More details about connections are given in the following paragraphs.

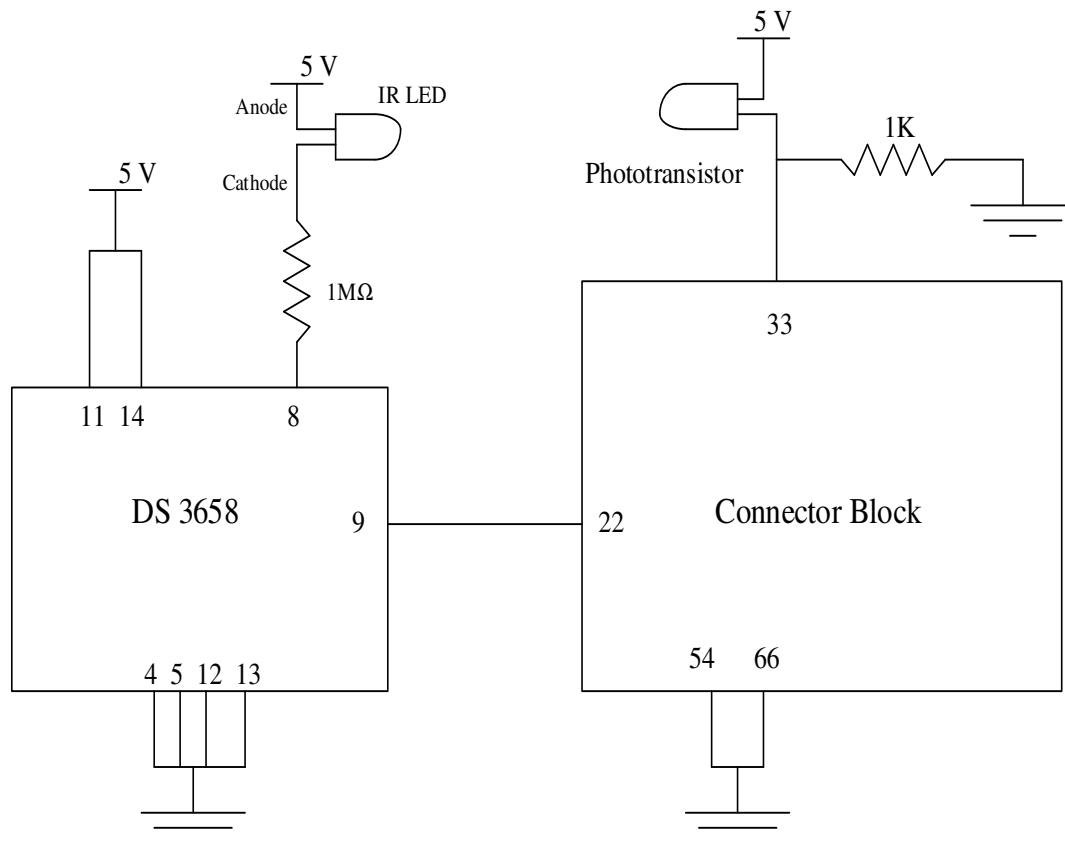


Figure 1: Illustration of circuit schematic for phototransistor experiments

In this setup, we're using the analog output signal from the NI card (pin 22) to activate the DS3658.

When activated, the DS3658 makes a connection from Pin 8 to ground. This allows current to flow from +5 through the $1M\Omega$ resistor and the LED (the anode is the long lead connected to the resistor. the cathode is the shorter lead) to pin 8, through the 3658, to ground. We intentionally use such a large resistor so as to generate a very weak signal from the LED.

Once the LED switches ON and the phototransistor senses this change in IR, a voltage appears across the $1K$ resistor connected between the emitter and ground. The collector (shorter lead is collector) is connected to power, and its emitter pin is connected to pin 33 of the NI card terminal block. We will look at these voltages with the A/D converter and data analysis functions of the LabView System.

Physically, the phototransistor and LED should be about 1 cm apart on your protoboard, with the axes of the plastic housings aligned (i.e., they are pointing at each other).

Once everything is plugged in, you can turn the power supply on and look at the phototransistor signal with the oscilloscope (measured across the $1k$ resistor).

The signal you see is quite noisy and also has a DC offset. Try covering the phototransistor with your hand and see the offset fall. Why do you think that happens? Hint: Would your offset change if you were working during the day versus during the night? If it is daytime, adjust the positions of the shutters on the windows to see how much difference that makes. In the report describe the observations from these experiments with the background light. For the measurements in the next sections, please be sure to minimize the amount of sunlight in the space around your setup, and especially to avoid direct sunlight on your lab bench.

Time to put generate some real signals and observe the response of the phototransistor in LabView...

Part 2 – Build some software.

Open the blank.vi (you can download it from the website, Lab1). Set things up so that you have the front panel on the left and the block diagram on the right.

Create a signal source (right click on block diagram to get the functions menu, and select input, and then simulate signal)

Set it for a square wave at 25 Hz, amplitude 2, and offset 2. Set the number of samples to 200 samples and the samples per second to 25 Hz. LabView displays an error. So now we know that the sampling rate must be at least twice the desired frequency (this is a fundamental requirement. If you want a good-looking output waveform, you'll need many more than 2 samples per cycle). Increase the sampling rate to 50Hz and 200 samples. This means that it will take approximately $200/50$ seconds to take the data. Now make a graph indicator and connect its input to the output of the signal source. Place the VI in a while loop as you did in the previous lab (Right click again, and select Execute Control, while loop, and drag this while loop around the Simulate Signal and DAQ Assistant variable).

Save the file with a new name in your folder. Run the VI and look at the graph. You might have to change the scale from 0-0.1. Doesn't really look like much of a square wave. Try increasing the sampling rate and number of samples until the wave looks square. Since you're sampling faster, you can take more data points without substantially increasing your measurement time. About 10000Hz sampling rate with 1000 samples should be about right for a very clean square wave.

In your report, briefly comment on why the sample rate should be at least twice the desired frequency. (Ask if you don't understand!).

Create an analog output (right click, select functions, output, DAQmx Assistant). This will take you through some menus. Select Analog Output, Voltage, "ao0" and finish. This will build an icon and automatically open the features window. Set output range for 0-10V. On the Timing Window, select "generate continuously", and check the box for "use timing from waveform data". Click the "OK" button on the lower right. LabView does a little work building a set of things that you don't need to see, and then things should be ready to go. Make sure everything stays inside the while loop.

(Note – there is a chart showing the relationship between the numbers on the connector block and the I/O functions for the DAQ board. This chart is a sheet of paper hanging from the shelf over each workstation).

Draw a wire from the output of the "simulate square" to the input of the DAQ assistant. You already created a graph indicator on this wire to look at the square wave signal.

When all this is there, try hitting the run button. You should see a square wave on the graph indicator from 0-4V. Turn on the oscilloscope and look to see if this same

waveform is showing up on the output signal from the hardware – this should be on pin 22 of the connector block, and ground is pin 54.

Experiment with some of the settings on the source and on the display until you have a good feel for how to operate this “Virtual instrument”. In the report, describe the setup you finished with in this section, and some of the features that had to be adjusted to get a setup that you liked best.

Part 3 – Build some more software

Create a Signal Input by right clicking, then select input and then DAQ Assist. Place this icon within the while loop from the last section so that everything will run together. Once again, this will take you through some menus. This time, select analog input, voltage, “**ai1**”, and hit finish. As before, this will open the features window. Set the range for -5V to +5V, differential. Set the Task Timing for “acquire N samples”, and set for 1000 samples to read and 10000 Hz rate. When you hit the OK button, this will do some work building the VI for you and then be ready to go.

Make a graph indicator at the data output of this DAQ Assistant, and set the properties for autoscale on the vertical axis, and ensure that it's 0-0.1 on the x axis. Set the previous graph indicator to the same settings.

So, we've set the signal acquisition system to read 1000 samples at 10000 Hz, which will take about 0.1 sec. We've set the graph indicator for 0-0.1 sec. We've also set the signal generator for 1000 samples at 10000 Hz (that takes 0.1 second), and we've set the signal output voltage generator to sample continuously. This is an important arrangement – the timing for the source and sampling must match if you want to run continuously.

Save the file.

When you hit the run button, you should see on the signal graph indicator something like this:

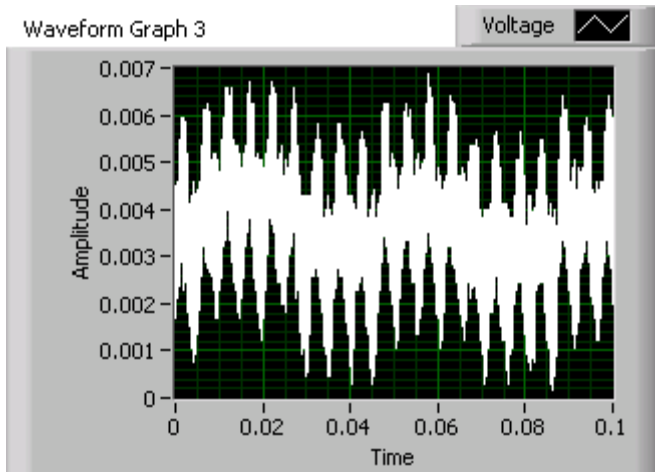


Figure 2: Raw signal from phototransistor.

What you're seeing is an extremely weak lower frequency signal (notice the amplitude) with a lot of higher frequency noise. Take a careful look at the noise. What do you see? Briefly comment on the amplitude and frequency of the signal and noise components. Do they each make sense? If you block the light with your finger, what changes?

We have a weak signal with a lot of added noise. Now we want to estimate the RMS amplitude of the noise (voltage units).

$$RMS = \sqrt{\frac{1}{n} \sum X_i^2}$$

To start, let's block the signal by putting something in the path between the LED and detector, so that the entire measured "signal" is all noise. Capture a plot of this for comparison with the plot with the real signal present.

We want to see how "random noise" varies with bandwidth. Connect a low-pass filter to the data terminal of the DAQ Assistant2 (Right click in the block diagram, select analysis, then select filter) and connect the output of the filter to a graph. Select a realistic 3rd order filter. How does this change the observed signal?

Think about the noise in the signal – Is it random, periodic, or both? At what frequency is it? What effect should your filters have on this noise contribution?

There is a function for RMS measurement inside the same menus as all the other functions you've been using (functions—analysis—amplitude & level measurements). Select the RMS check box. Use a numeric indicator to display RMS. Connect the output of the low-pass filter to the RMS function. Do a series of measurements of the amplitude as a function of the cutoff frequency of the low-pass filter. How should the

RMS noise depend on this cutoff? A plot of the RMS noise as a function of the cutoff is a useful tool, and it might be better to plot it on a log-log scale. Make sure to include this in your report.

Increase the order of the filter (to about 15), so as to better remove any noise with frequency close to the cutoff frequency. This 15th order filter has a very sharp cutoff at the cutoff frequency, and is more ideal in that respect. It is easy to make this in a digital system, but very hard to build an analog 15th order filter.

Tabulate the total RMS noise values for cut-off frequencies of the 15th order low-pass filter at 100 Hz, 200 Hz, 500Hz, 1kHz, 2 kHz, 5kHz, 10kHz, 20kHz and 50kHz. A plot of this result should be in the report. Can you explain the shape of this plot?

Revert the order of the filter to 3 or 4 once finished.

OK, this is all an interesting exercise. However if we look at the graph of the waveform after filtering we can see that the signal is distorted (not a square wave anymore), and the amplitude has been affected. This is a fact of signal conditioning – filters can influence the noise and the signal, and it is hard to get one without the other.

Notice how the filtered output changes dramatically when the phototransistor is covered, and when the alignment between the phototransistor and LED is changed.

Another way to see how the filters are altering the signal is to look at all of this in the frequency domain.

Part 4: Some Frequency-Domain Analysis of the oscillating signal.

Create a Spectral Measurement function (right click, select analysis, and set the Spectral Measurement Option to “Magnitude RMS”). Also create a graph indicator at the output of this spectral measurement, and draw a wire connection from the output of the filter to the input of the spectral analysis. Save the file.

Go ahead and run. This new display shows a frequency spectrum with amplitudes as a function of frequency. Signals at 25 Hz are separated from periodic noise at other frequencies, and added to random noise that is spread over frequencies. Capture a plot of the spectrum and explain the various contributions that are displayed.

You’ll see that the spectrum that you generate does not have much resolution in the frequency domain - meaning that there are not many data points per unit frequency. The problem here is that you are not measuring long enough – the 0.1 seconds of data is only able to produce data points every 10 Hz on the plot. This is just not going to be very useful for measurements around 25 Hz.

This is another fundamental aspect of frequency domain analysis - there is a relationship between the measurement time for a scan and the lowest frequency that can be detected.

If for example, you want to measure the amplitude of a 0.1 Hz sinewave, you need to measure for at least as long as it takes for the sinewave to go through a max and a min. If you want 0.1 Hz steps in the spectrum, you need to have a 0.1 Hz point somewhere, and the time necessary to gather that point sets the necessary time for the measurement.

Go through the various menus for the source, data acquisition & spectral measurement, and set things up for 1 second of data (10000 samples and 10000Hz sampling rate). When you see the plot, you should see a clear peak at 25 Hz well above the noise. Now, this is not a big peak – the amplitude of the oscillation you estimated was pretty small, so this is not expected to be very large. Make a cursor legend (as in the last lab) to measure the actual peak height. Also measure heights of other dominant frequencies.

Change the total measurement time (in all the necessary menus) to 5 seconds of data, and repeat the measurement. In this case, you'll see that the spectrum changes. For instance, the shapes of the peaks have changed (describe what you see), and the amplitude of the background noise baseline has changed (describe what you see). Increase the time again to 50 seconds, and compare/explain again. What you should be seeing is some of the relationship between the measurement time and the bandwidth associated with each point in the spectrum and the resulting amount of random noise at each point.

Try to distinguish the various peaks observed in the graph and their nature/origin in the report. Which of these peaks belong to the signal and which of them are noise?

Throughout all of this, you're hopefully beginning to see and understand some of the relationship between the time domain and the frequency domain. In the time domain, you have some good intuitive experience and understanding with the effect of averaging, sampling, filtering, etc. These things all matter in the frequency domain as well. In this lab, you may be seeing some of this for the first time, and hopefully beginning to understand some of it. If not, no worries! We'll be doing more and more of this.