BE/EE/MedE 189a: Design and construction of biodevices
Exercise packet

Background and Introduction

In this course, we provide you with two design challenges. First, you will design and build a pulse oximeter. After completing that project, you will tackle the more challenging project of designing and building a real-time polymerase chain reaction (qPCR) machine. We will provide basic specifications and construction materials for these devices, and it is your job to implement the specification and demonstrate a working prototype.

In the lab, we will use ELVIS II breadboards controlled using LabVIEW (both from National Instruments) on a personal computer. To get you up to speed on using the board with LabVIEW, you will complete the exercises in this document. Each exercise gives a very simple specification, and it is your job to design and execute a simple circuit and/or LabVIEW virtual instrument (VI) to meet the specification. In all cases, you should provide verbal explanations of the design choices you make.

You will have to be resourceful in doing these exercises. The course staff is around to help you, but you will likely need to ask Google for help on LabVIEW programming. This is part of learning how to use new tools. For each exercise, we give you some hints on what to look for.

While you can find lots of LabVIEW instructional resources and sample VIs online within the LabVIEW software, we have provided notes and sample VIs prepared by Changhuei Yang and Justin Bois.

1 Turn on an LED

This is the "`Hello, world.`" of hardware. Create a VI that will light an LED that is built-in on the ELVIS board.

Helpful concepts: DAQ assistant, Boolean constants.

2 Blink an LED with an front panel input

Create a VI that will blink an indicator LED on the front panel with a fixed blinking rate. (Note that this is a virtual LED; it's on the front panel of the LabVIEW VI, not

on the board.) You can decide how you want to set the blinking frequency.

Helpful concepts: Numeric constants, indicators on front panel, while loops, flat sequences, feedback of variables in a while loop.

3 Blink an LED with a software input

Create a VI similar to Exercise 2 but now have blinking rate editable from the front panel and use a physical LED on the ELVIS board. This is a physical LED, not a built-in one on the board; these LEDs and necessary resistors are in the lab. Use a numeric input type of your choosing on the front panel to set the brightness of the LED in the ON and OFF states. Limit your numeric choices to integers and whole numbers but you can represent them in any way, as a slider, a knob, etc. To test, vary the blinking rate and show the LED changing brightness.

Helpful concepts: Numeric inputs on the front panel.

4 Blink an LED with a hardware input

Use a DAQ assistant to read in an analog voltage when connecting the circuit shown in Fig. 1. Use the data that is read in to change the duty cycle of the LED. Then output to an LED on the board. In the end, if there is minimal delay in the read/write of analog voltage signals from the DAQ, you should be able to control the perceived brightness of the LED by tuning a potentiometer (provided by the TAs) and via pulse width modulation (PWM) signaling (turning the LED on for a set amount of time and turning it off for a set amount of time). *You may not be able to control brightness in this way, and the LED will still show perceptible blinking. What does this say about the time delay for DAQ I/O?*

Helpful concepts: DAQ assistant to read in signals, numerical operations like division, addition, etc., inverters and negations

5 Turn on and off an array of LED in sequence

Create a VI that will turn an LED array on and off in sequence. You may either use the built-in LEDs on the right side of the ELVIS board or you may construct your own array of LEDs on the board. The timing of how long to keep each LED on can be done in any way you want to design. Your light sequence should look like the lights

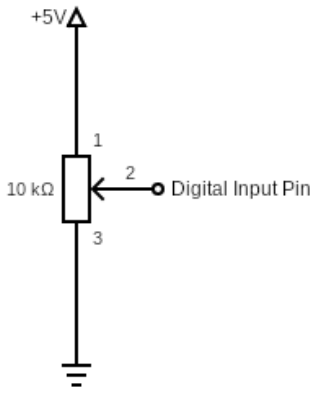

Figure 1: A simple circuit for reading analog voltage.

on the front of KITT (the talking car) in the hit 1982 TV show *Knight Rider*. See here and you can see a detail here.

Helpful concepts: For loops, arrays, indexing, feedback in the different control structures, case structures

Pro tip 1: You can use a DAQ assistant to output to multiple channels (digital pins) at once. To do so, add the extra pins in the editing window of the DAQ assistant after you have placed it in your VI. Then you can pass the DAQ an array with the number of elements equaling the number of channels for the DAQ and it will output all of them to the correct locations. You should try this before you get too deep into your program so that you know how it works.

Pro tip 2: Be careful when bringing am array into a for loop. Sometimes it does auto indexing which means every time the loop is run, it automatically takes in the next value of the array. This can cause problems if this is not what you intended and are trying to access the array inside the for loop. To find if this is what is happening, you should look at the box where you bring in the wire from outside to inside the for loop. If it has [ ] in the box, it is auto-indexing. To change from either one to the other, right click on the box and select the appropriate option.

6 Low pass filter

Develop a virtual low pass filter using DAQ analog input and output. Output a triangular waveform with a peak-to-peak voltage of 2 V and a frequency of 100 Hz. Connect the analog output on the NI ELVIS II system to one of the analog inputs

and use the DAQ assistant to acquire the voltage signal. Apply a moving average filter and then output the filtered signal to a waveform chart. You should be continuously generating and acquiring signals. Write the filtered voltage values to a file after you stop the experiment. To verify that it wrote correctly, you can plot the signal using software of your choice.

Pro tip: You can test the input side of your VI by filtering the output of a triangle wave from the FGEN port on the NI ELVIS II system. You can control the FGEN port using the **NI ELVISmx Function Generator**, which can be accessed via `Start >> All Programs >> National Instruments >> NI ELVISmx Instrument Launcher`. The **NI ELVISmx Instrument Launcher** contains other instruments that could be useful for your design projects.

The moving average filter is defined by the following equation

$$y[n] = \sum_{k=0}^{N-1} \frac{x[n-k]}{N} \tag{6.1}$$

where $x[n]$ is the raw signal, $y[n]$ is the filtered signal, and N is the filter window size. It is an example of a finite impulse response (FIR) filter.

Helpful concepts: FIR filter, write to measurement file, simulate signal

7 Design an oscilloscope

Design a VI to simulate an oscilloscope using a waveform chart. Generate a sine waveform that varies from -5 to 5 V to serve as input. An example of a virtual oscilloscope is shown in Fig 2. You don't need to write every feature and use the same user interface as shown in the figure. You only need to implement the following features.

1. One channel input.
2. The scale of the y-axis (Volt/Div) can be tuned.
3. The time base (Time/Div) can be tuned.
4. The display offset can be tuned.

Design the user interface and use any controls or indicators that you think are suitable.

Helpful concepts: Graphs, charts, chart properties, graph properties, numerical operations, waveforms

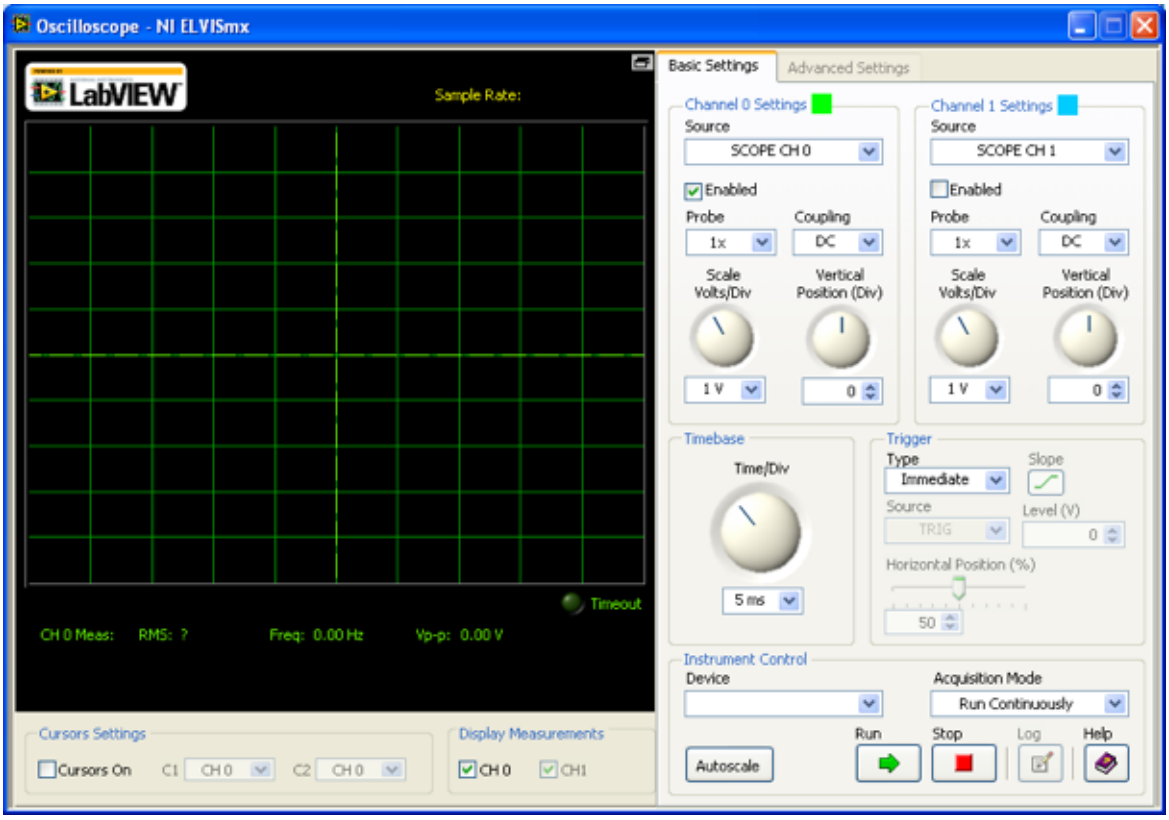

Figure 2: A design of a virtual oscilloscope.

8 ECG graph display

Design a VI that processes an electrocardiogram (ECG) signal. A typical one-cycle ECG signal is shown in Fig. 3. The following figure, Fig. 4 shows an example of the electrocardiogram (ECG) signal. Design a filter to remove the baseline wander and the high-frequency noise that can be observed in the figure. After denoising, design and implement an algorithm to find the R-peaks, and calculate the heart rate according the peak positions. An ECG trace can be found in the file `ecg.txt`, which includes 2000 data points. The entire trace is taken within a 5-second time window.

Helpful concepts: Pre-built VIs that read and write to text files, filters, summation of arrays, numerical operations on arrays, peak finder

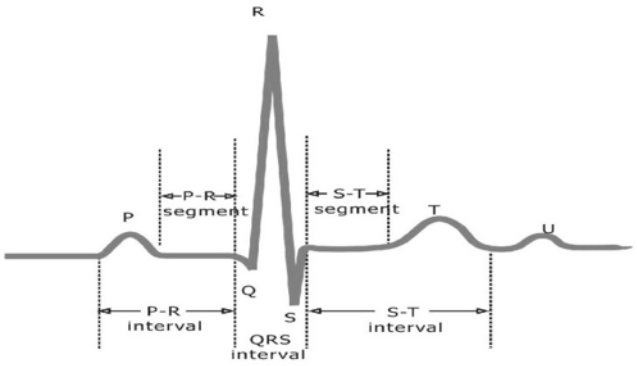

Figure 3: Single cycle ECG trace and its components.

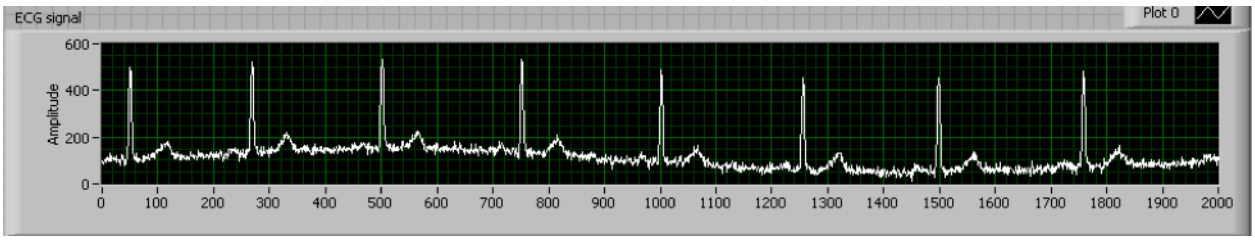

Figure 4: Sample ECG trace.