Lab 3: Digital Sampling and Spectral Analysis

1. Introduction

Computers are often used as data measurement systems. This laboratory experiment will use the computer to measure a series of data points, referred to as samples. The three main concepts of computer data acquisition that will be explored in this lab are spectral analysis, sample rate, and sampling resolution. You will make additions to your data acquisition GUI from Lab 2 and investigate the effect of sample rate and resolution.

1.1 Lab Objectives

- Learn how to deal with the effects of aliasing
- Learn how resolution effects the sampling of a signal
- Learn how to perform spectral analysis using the Fourier transform
- Practice MATLAB GUI skills

1.2 Project Objectives

- Provide a detailed presentation of your design and strategy

1.3 Lab Hardware

- Function Generator
- Oscilloscope
- National Instruments DAQ
- Potentiometer

1.4 Project hardware

- N/A

2. Laboratory Concepts

2.1 A/D Resolution

The resolution of an A/D (analog-to-digital) converter indicates the number of discrete values it can measure over the range of analog values. Values are usually stored digitally in binary form, so the resolution is usually expressed in bits, as shown above. In consequence, the number of discrete values available, or "levels", is a power of two. For example, an A/D with a resolution of 8 bits can encode an analog input to one in 256 different levels, since $2^8 = 256$. The values can represent the ranges from 0 to 255 (i.e. unsigned integer) or from −128 to 127 (i.e. signed integer), depending on the application.

Resolution can also be defined electronically, and expressed in Volts. The minimum change in voltage required to guarantee a change in the output code level is called the least significant bit (LSB) voltage. The resolution Q of the A/D is equal to the LSB voltage. The voltage resolution of an A/D is equal to its input voltage range divided by the number of intervals, where M is the number of bits:

$$Q = \frac{V_{HI} - V_{LOW}}{2^M}$$

2.2 Aliasing

Another important characteristic of an A/D is the sampling rate, or speed, of the A/D. An A/D does not convert voltages to numbers continuously at every instant in time. Instead, an A/D periodically converts data samples at points in time determined by the sampling rate. The sampling rate, f_s, determines the number of samples the A/D measures per second:

$$f_s = \frac{\#\, samples}{second}$$

Because the A/D samples only at discrete points in time, changes in the signal voltage between any two sampling instants are not measured. For example, if the sampling rate is 1000 samples/sec (or Hz), then the signal is read by the A/D every millisecond and the computer has no knowledge of the signal value between samples.

It is very important that the sampling rate is higher than the frequency of the signals to be measured by the A/D. The Nyquist frequency f_N is defined as half of the sampling frequency:

$$f_N = \frac{1}{2} f_s$$

To prevent information loss from the sampling process, the actual frequency of measured signals should be below the Nyquist frequency:

$$f_{actual} < f_N$$

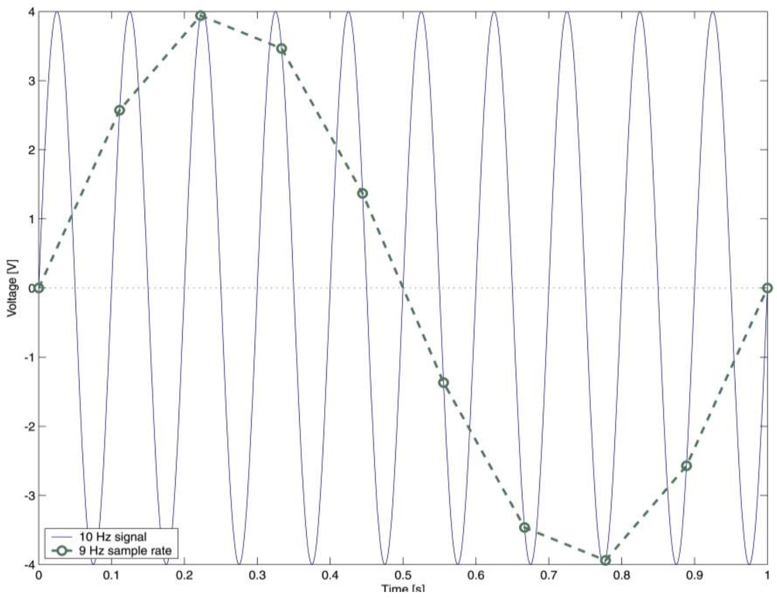
If the sampling rate is too slow, a phenomenon known as aliasing will occur. Aliasing is a misrepresentation of the measured signal at a slower frequency. The aliased frequency, f_{alias}, can be predicted using:

$$f_{alias} = |f_{actual} - M \cdot f_s|$$

$$M = roundToNearestInteger\left(\frac{f_{actual}}{f_s}\right)$$

where M is an integer chosen such that $0 \le f_{alias} < f_s/2$. This equation illustrates that for a given sampling rate, all of the detected frequencies will lie between zero and the Nyquist frequency whether or not the actual frequencies are in that range.

The aliasing phenomenon can be a difficult concept to visualize, so the laboratory procedure that follows will investigate how aliasing can alter the measurement of a signal of known frequency. The figure bellow provides an example of a 10 Hz signal sampled at 9 Hz to intentionally cause aliasing. Notice that the aliased frequency that the computer detects is 1 Hz.



A 10 Hz signal and its 1 Hz alias, when sampled at 9 Hz.

Although the Nyquist frequency is the minimum sampling frequency necessary to avoid aliasing, the accuracy of measuring a signal will be improved substantially by using a sampling rate that exceeds ten times the highest frequency of the measured signal.

Aliasing can also be eliminated by applying a low pass filter to the signal before it is measured by the A/D converter. The low pass filter will remove high frequency content of the signal that would cause aliasing. The frequency of the filter is generally less than or equal to the Nyquist frequency.

2.3 Spectral Analysis using Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is the analysis tool behind all digital spectral analysis systems. It allows a waveform sampled at discrete time intervals to be broken down into its individual frequency components.

In general, any continuous signal can be represented by an infinite sum of sinusoidal terms, with frequencies ranging from 0 to infinity. The Fourier Transform allows us to find the amplitudes of each of those terms (so we know how much of each frequency is in our signal).

Similarly, a signal made up of a **finite** number of discrete samples can be represented by a **finite** sum of frequency components. The DFT allows us to find the amount (magnitude and phase) of each of these frequency components that make up the signal. Some important characteristics of the DFT are given as follows:

- The DFT is a transform between two discrete, complex data sets of length N: g_n, n = 0...N-1 and G_m, m = 0…N-1,

$$G_m = \sum_{n=0}^{N-1} g_n e^{-j2\pi mn/N} \qquad \text{forward transform}$$

$$g_n = \frac{1}{N}\sum_{m=0}^{N-1} G_m e^{j2\pi mn/N} \qquad \text{inverse transform}$$

- If the data set $\{g_n\}$ represents data taken over a time period T, then $\{G_m\}$ represents the amplitudes of the frequency components: 0, Δf, $2\Delta f$ …f_{max} that make up that data, where Δf = 1/T is the frequency resolution of the discrete spectrum. Since these amplitudes are complex numbers, they have both magnitude and phase.

$$G(m\Delta f) = \sum_{n=0}^{N-1} g(nT) e^{-j2\pi mn/N}$$

$$g(nT) = \frac{1}{N}\sum_{m=0}^{N-1} G(m\Delta f) e^{j2\pi mn/N}$$

- The minimum frequency Δf (or frequency resolution) represented in the DFT is the inverse of the total time of the data record T, which can also be expressed as a function of the sample rate and the number of samples: $f_{\min} = \Delta f = 1/T = f_s/N$.

- The maximum frequency $f_{\max}$ represented in a DFT is determined by the sample rate. While the DFT technically computes frequency terms all the way up to the sampling frequency f_s, the terms above the Nyquist frequency are just complex conjugates of the terms below the Nyquist frequency, and are redundant information. Thus when we plot the amplitudes of the frequency components, we typically merge the complex conjugates and only plot up to the Nyquist frequency, which can also be expressed as a function of the frequency resolution and the number of samples: $f_N = f_s/2 = (N/2)\Delta f$.

- There is an implicit assumption in the DFT, and Fourier Series in general, that the recorded data is periodic (i.e. the data set repeats itself over and over), which is not generally true in a practical situation. The result is that the DFT, or Fourier Series, calculations will say that there are frequencies in the measured signal that may not really be there in order to approximate the assumed periodicity of the data at the beginning and end of the recorded data. This is known as Spectral Leakage, since some of the power artificially "leaks" into other frequencies. Techniques known as "windowing" can help reduce leakage by multiplying the original signal by a function that makes the data record equal zero at the beginning and end of the time record. Leakage can also be reduced by recording longer periods of data (e.g. increased number of data samples) such that the effects of the beginning and end of the data is small compared to all of the data during the rest of the period.

- The discrete power spectrum of the signal $|G_m|^2$ identifies the "power" associated with each harmonic in the spectrum. The power spectrum is used extensively in engineering practice.

- In practice, the DFT is implemented by the Fast Fourier transform (FFT) algorithm. The most common form of the FFT works with records of length $N = 2^n$, $n = 1,2,\ldots$ While the MATLAB FFT function works with any record length, it is most efficient when it is a power of 2.

In lecture, you were taught the theory of Fourier Transform. In this lab, you will use MATLAB to perform the Discrete Fourier Transform. The following functions are used to perform the DFT on a data set, `data`, taken at a sampling rate `samplerate`.

```
% Compute DFT using fft function
G = fft(data);

% Smallest measurable frequency
spectralResolution = samplerate/length(data);

% Largest measurable frequency
f_Nyquist = samplerate/2;

% Define range of frequencies
freq = 0:spectralResolution:f_Nyquist;

% Calculate magnitude in volts
Mag = abs(G(1:length(freq)))/length(data);
Mag(2:end-1) = 2*Mag(2:end-1);

% Calculate power of frequencies
Power = Mag.^2;

% Calculate phase
phase = angle(G)*180/pi;
```

3. Pre-Lab Exercises

1. The A/D of the data acquisition cards installed in the laboratory computers is 12-bit with an input range of ±10 V. What is the resolution of the A/D in milliVolts?

2. What is the minimum sampling rate to avoid aliasing if the actual frequency of a signal is 20 Hz? Write the specific sampling rate.

3. Again, consider sampling of a 20Hz signal. How high should the sampling rate be if you want to obtain a reasonable representation of the signal? Write the specific sampling rate.

4. Calculate the aliased frequency of the following signal frequency and sampling rate pairs.
   a. f_{actual} = 50 Hz, f_s = 10 Hz
   b. f_{actual} = 50 Hz, f_s = 40 Hz
   c. f_{actual} = 50 Hz, f_s = 70 Hz
   d. f_{actual} = 50 Hz, f_s = 100 Hz

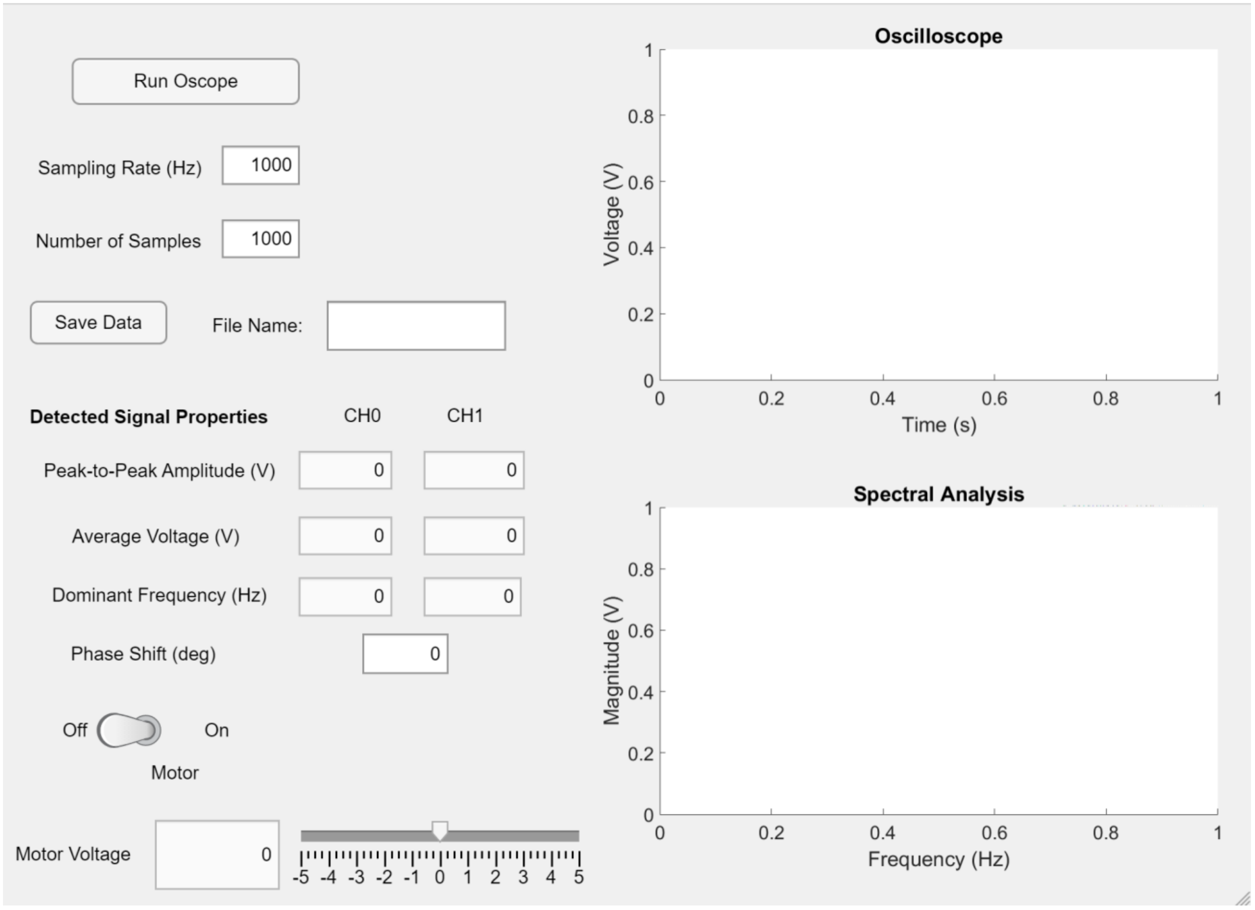
5. Create a function in MATLAB that takes the FFT and returns the spectrum (magnitude, phase, and frequency) for a given data set sampled at a certain rate. Attach your code for this function to the Prelab (.m file). The function definition should begin like this:
   ```
   function [ Mag, phase, freq ] = fft_sample( data, sample_rate )
   ```
   where `Mag` is the magnitude of the signal (volts) for each frequency in the signal, `phase` is the phase angle (`deg`) of every frequency, `freq` is the frequencies present in your signal (`data`), `data` is a single sample set of a signal, and `sample_rate` is the sampling frequency (Hz) that your data was taken at. Use this function for the next problem.

6. In the Lab 3 folder you will find the file **Lab3_prelab_6.mat** containing a signal data set sampled at 500 Hz for 1 second. (MATLAB variables data (V) and time (sec)). Using the function you created in the previous problem, perform the FFT in MATLAB and plot the Magnitude (y-axis) vs Frequency (x-axis). Attach a properly labeled plot and your labeled MATLAB code to your prelab. What are the primary frequencies (and their amplitudes) contained in the data sample?

4. Laboratory Exercises

4.1 Spectral Analysis and MATLAB GUI

In this lab exercise you will expand on the MATLAB GUI you created in Lab 2. You will use your function fft_sample() from pre-lab to add spectral analysis functionality. This will allow you to measure frequency and phase shift between two signals. **This will be used in further exercises in this lab.** When you are finished with exercise 4.1, your GUI will look like the following figure.



4.1.1 Create a Copy of your Lab2 GUI

- Open your GUI/App from Lab 2 and save as *Yourname_Lab3_GUI.mlapp*
- Put a copy of your function *fft_sample.m* from pre-lab 3 in the same working directory as your app

4.1.2 Add Spectral Analysis using FFT

- In the *Design View*:
    - Create a new Axes object with the name `FFT_Axes`
    - Title the Axes "Spectral Analysis"
    - Label the x axis "Frequency (Hz)"
    - Label the y axis "Magnitude (V)"

- In the *Code View*:
    - Add the following code to the bottom of the while loop in your `RunButtonValueChanged()` callback function:

```
% remove DC offset and compute FFT
[Mag0,phase0,freq]=fft_sample(app.Data(:,1)-ave0,rate);
[Mag1,phase1,freq]=fft_sample(app.Data(:,2)-ave1,rate);
plot(app.FFT_Axes,freq,Mag0,freq,Mag1);
drawnow; % this makes sure the graphics update
```

4.1.3 Display Dominant Frequencies and Phase Shift

- In the *Design View*:
    - Add a new pair of *Edit Fields (Numeric)* displaying the dominant frequencies in CH0 and CH1. Then add a single *Edit Field (Numeric)* for displaying the phase shift between the dominant frequencies in CH0 and CH1. Name them `CH0FreqField`, `CH1FreqField`, and `PhaseShiftEditField`. Arrange them and add labels as shown in the figure above.
- In the *Code View*:
    - Add the following code to the bottom of the while loop in your `RunButtonValueChanged()` function.

```
% compute and display dominant frequencies
[~,index0]=max(Mag0);
[~,index1]=max(Mag1);
app.CH0FreqField.Value = freq(index0);
app.CH1FreqField.Value = freq(index1);

% compute and display phase shift between CH0 and CH1
phase_shift = phase1(index1)-phase0(index0);
if phase_shift > 180
    phase_shift = phase_shift - 360;
elseif phase_shift < -180
    phase_shift = phase_shift + 360;
end
app.PhaseShiftEditField.Value = phase_shift;
```

4.1.4 Test your new GUI

- Connect the function generator to DAQ channels AI0 and AI1 (using a BNC splitter) with a frequency of 100 Hz and amplitude of 3 V. Run the GUI. Check to make sure the frequencies are showing up correctly.
- The phase between the two channels should be 0 deg.
- Adjust your sample rate as needed to acquire a good sampling of the signal
- If you have issues your fft_sample() function may be to blame.

4.1.5 Now that your spectral analysis works, take a sample of data from a <u>square wave</u> at 10 Hz with a sample rate of 1000 Hz. Save the data using the functionality of the Lab3_GUI. <u>Verify that the data was saved.</u> <u>You will need this for the Post lab.</u>

Have a TA check your progress and have them sign off.

4.2 A/D Resolution

In this exercise, you will investigate the effects of sampling resolution on the observed signal. Using the function generator and a voltage divider, create a 20 Hz, 3 mV peak to peak sinusoid signal. The function generators alone are not capable of providing a 3mV signal. This is where the voltage divider using the potentiometer is used.

- Get a potentiometer from the electronics drawers.
- Feed the output of the function generator to the input of the potentiometer; use a BNC to test-clip and connect the red clip to port 1 of the potentiometer and the black to port 3.
- Connect the wiper output of the potentiometer to the AI0 on the DAQ. Don't forget to make a common ground.
- Run your **Lab3_GUI** to view the waveform of the signal.
- Adjust the 'Amplitude' knob on the function generator so that you get the smallest possible output amplitude.
- Then turn the wiper on the potentiometer until you get a 3 mV signal.
- Show your TA what effect resolution has on sampling analog signals.

Have a TA check your progress and have them sign off.

4.3 Aliasing

In this exercise, you will investigate the effect of sampling frequency on measured frequency of a signal.

4.3.1 Run your Lab3_GUI.

4.3.2 Using BNC cables connect the function generator to the AI0 of the DAQ.

4.3.3 Using a BNC-to-test clip cable, connect the red and black leads to each other. Connect the BNC end to AI1. This grounds out CH1 and will eliminate cross talk between CH0 and CH1.

4.3.4 Turn on the function generator and adjust the output to a 20 Hz, 4.0 V sine wave. Verify this signal using a sampling rate of 1000 Hz. **NOTE:** The function generator does not have the ability to display the amplitude of the signal. You will have to adjust the amplitude by looking at the measured signal in the GUI and adjusting the AMPL knob as needed. Also make sure the amplitude knob is pressed in.

4.3.5 Using **Table 4.3.5 in section 4.4,** take sampled data at each of the listed sampling frequencies, calculate the expected aliasing frequency, record the observed frequency, and make a comment or two describing the shape in the space provided.
NOTE: Collect 1 second worth of data.

Have a TA check your progress and have them sign off.

4.3.6 Set the sampling frequency of the GUI to 1000 Hz.

4.3.7 Using **Table 4.3.7 in section 4.4,** below record the measured frequency when you increase the actual frequency of the signal from 125 Hz to 2000 Hz. In the postlab you will answer questions related to these measurements.

Have a TA check your progress and have them sign off.

Clean up your area and have the TA sign you off.

4.4 Student work and Tables

Name________________________

Table 4.3.5

Sampling Rate (Hz)	Nyquist Frequency (Hz)	Calculated Alias (Hz)	Measured Frequency (Hz)	Description
10				
12.5				
15				
17.5				
20				
22.5				
25				
27.5				
30				
40				
50				

Table 4.3.7

Sampling Rate (Hz)	Approximate Frequency (Hz)	Actual Frequency (Hz)	Measured Frequency (Hz)
1000	125		
1000	250		
1000	375		
1000	500		
1000	625		
1000	750		
1000	875		
1000	1000		
1000	1125		
1000	1250		
1000	1375		
1000	1500		
1000	1625		
1000	1750		
1000	1875		
1000	2000		

5. Post-Lab Exercises

1. Given your understanding of aliasing, how could you be sure to correctly determine the frequency of an unknown sinusoidal signal using digital sampling?

2. Using the square wave saved data from 4.1.5, plot the original signal (single channel) and plot the spectral analysis. From the spectral analysis you should see harmonic peaks. Using the ifft() function, reconstruct the signal with data up through 5 harmonics, then 10 harmonics, and then all harmonics. Plot the three reconstructed signals on the same axes. Include plots in the Post-lab submission.
   HINTS:
   - think about how the magnitude is calculated from the raw data.
   - Be careful of what data you include and what data you set to zero
   - If G = fft(V) then V = ifft(G)

3. Using the data from 4.3.7, create a plot of Measured Frequency (y-axis) vs Actual Frequency (x-axis). Attach plots to the Post-lab submission.

6. Project Milestone 3

The purpose of Milestone 3 is to detail the design of the mechatronic solution of your team's robot and wireless transmitter. For this PM you will present your design in a detailed PowerPoint (15 minutes) during Lab 4. A shared PowerPoint document through some cloud-based service is recommended since they will allow your team members to collaborate on the presentation. The presentation should provide a mid-level description of your selected design. Use pictures or diagrams for clarity. Your presentation should include the following:

- A detailed CAD model of your robot
  - A rendering of your model
  - Subassemblies models when necessary (mechanisms, mobile platform, etc)
  - Include actuators, wheels, mechanisms, fasteners, microcontrollers, sensors, shields
- A list of all sensors, actuators, shields, microcontrollers, mechanical hardware, and raw materials
  - Sensors and motor gear ratios (Pololu DC motors) will be finalized in later milestones. Students should be thinking of what they need to sense, but they do not have to decide on specific sensors for this milestone. Students will decide on specific gear ratios in later PM.
  - Provide a cost estimate for your robot. Be sure that it does not go over the allowed budget. See the "Available Parts in Mech Lab.pdf" for prices.
- Describe how your solution will complete the following objectives
  - Driving/navigating around the playing area
  - Acquiring and sensing blocks from the dispensers
  - Attaching blocks to the chassis
- State transition diagram
  - Provide a complete state transition diagram of your control scheme
  - Your diagram should provide a high-level view of each separate action/sub-routine the robot will perform and how the robot transitions between different actions.
    - States are continuous actions that repeat until an event causes a service
    - Event/transitions are what cause the robot to change states and what discrete action happens in between
  - Give a brief description of how your wireless communication and robot work with your given state diagram.
- Provide a preliminary pin table
  - The table should include all known connections to each microcontroller. Examples are shown below. As you construct your pin table, make sure to allocate all the pins required for each motor/driver and sensor (each motor driver and sensor may require multiple pins, both digital and analog). Refer to the *Robot Construction Guide* and the *Pin Considerations when using the Arduino Mega* documents on Canvas.

Arduino Mega pin	**External Connection**	**Purpose**
Digital pin 2	Xbee shield RX	Wireless communication - Receive
Analog pin A0	Hall effect sensor	Sense magnet
Digital pin 10	Motor shield M2PWM	Control wheel motor 2
…	…	…

Arduino Uno pin	**External Connection**	**Purpose**
Digital pin 0	USB serial RX	Laptop serial communication - Receive
Digital pin 3	Xbee shield TX	Wireless communication - Transmit
…	…	…

Be sure to provide enough detail so that your TA can fully and easily understand each concept. You will get feedback from your TA on your designs when you present. Provide concise concluding statements regarding your design and team member responsibilities. As design is an iterative process, it is not expected that the final design will match exactly with the descriptions presented here. However, you should have given significant thought to all questions above and be ready to begin manufacture of prototypes.