

Individualized Laboratory Using Moodle

Brian Daku

University of Saskatchewan, brian.daku@usask.ca

Abstract - Practical laboratories are an integral part of an engineering education. The development of a challenging lab for a course requires a significant effort on the part of faculty and support staff. Thus labs, once developed, tend to have a relatively long lifetime. A problem with traditional labs is that student's lab results can be passed on to "next years" students. This reduces the effectiveness of the lab, since students are unlikely to put the same effort into the lab when working from a previous lab book. This paper describes a method to address this problem. This method makes use of a Moodle-based implementation to individualize the lab. The lab is structured to make use of the features in Moodle to automatically vary key parameters in the lab tasks. Thus, each lab group effectively has a different set of lab tasks. An added advantage of this approach is that the individualized lab task results are automatically marked. This approach has been used in a course that introduces students to a DSP (Digital Signal Processing) chip and the software used to build and debug DSP programs. Preliminary feedback indicates that the students like this Moodle-based approach.

Index Terms – Digital Signal Processing, Individualized Task, Laboratory, Moodle.

INTRODUCTION

This paper documents the use of Moodle [1] in creating engineering labs. The Moodle implementation is designed to make the labs unique for each lab group, or in other words, individualized for each lab group. The reason for individualizing a lab, is that in traditional labs student lab books are usually passed on to “next years” students. This reduces the effectiveness of the lab, since these students are unlikely to put the same effort into the lab if they have a lab book complete with results.

The approach described in this paper was used in developing a set of new labs for an introductory Digital Signal Processing (DSP) course. A curriculum change resulted in a significant laboratory component being added to this DSP class. This change provided an opportunity to explore this new Moodle-based approach to engineering laboratories.

The paper is organized as follows: the next section describes the content of the DSP laboratory implemented in Moodle. This is followed by a section that describes the structure used for the Moodle-based laboratory. Since the individualized tasks are based on Moodle quizzes, there is a section describing the quiz format, focusing on the

calculated and random question types. This is followed by a section titled, Lab Tasks, that describes and gives examples of how the task questions were implemented in Moodle. The final section is titled Discussion and Summary.

DSP LABORATORY

The purpose of the DSP course laboratory component is to introduce the students to the basics of using a DSP processor chip and some interface hardware. Specifically, the Analog Devices ADSP-BF533 EZ-KIT Lite is used in the lab. The DSP processor chip is the Blackfin BF533 and the interface hardware on the board that is used in the lab includes an ADC (analog to digital converter), DAC (digital to analog converter), push button inputs and LED outputs. The laboratory component also includes an introduction to: the integrated developing and debugging software tools (called VisualDSP++ for the Analog Devices processors) needed to build and debug DSP programs; assembly language programming; efficient hardware constructs, such as zero-overhead loops and circular buffers; and hardware interrupt service routines. A summary of the laboratory content is provided through the objectives and learning topics for each of the four parts of the lab, which are listed below.

A. LED Blink Program: The primary objective of this part is to learn how to use VisualDSP++ to build, monitor and debug a DSP assembly program. The learning topics are:

1. How to connect the EZ-KIT board to a computer.
2. How to start and use VisualDSP++ with the EZ-KIT board.
3. How to use zero-overhead loops.
4. How to turn the six user LEDs on and off.
5. How to single step a program.
6. How to use breakpoints.
7. How to access and change register contents using VisualDSP++.

B. Push Button Program: The primary objective of this part is to learn how hardware interrupts and interrupt service routines work and how VisualDSP++ can be used to debug them. Secondary objectives include learning how to use conditional breakpoints in VisualDSP++ and conditional assembly statements that use the CC (Control Code) bit. The learning topics are:

1. How to use the Programmable Flag (PF) pins to generate hardware interrupts.
2. How to generate and use an interrupt service routine (ISR).

3. How to use conditional breakpoints in VisualDSP++.
4. How to use conditional assembly statements (ie. IF CC JUMP label).
5. How to implement a simple counter.

C. Audio Codec Program: The primary objective of this part is to introduce circular buffers and use them to implement an FIR (finite impulse response) filter. The learning topics are:

1. How to use a Data Address Generation (DAG) unit.
2. How to use a DAG to implement a circular buffer.
3. How to use the analog to digital converters (ADCs) in the audio codec.
4. How to use the digital to analog converters (DACs) in the audio codec.
5. How to process the input signal sample-by-sample.
6. How to capture and plot a signal using VisualDSP++.
7. How to use VisualDSP++ to export a saved signal to a text file and then process it using MATLAB [2].

D. BPSK Program: The primary objective of this part is to use previously covered concepts, such as circular buffers and FIR (finite impulse response) filters to implement a BPSK (Binary Phase Shift Keying) communication system. The learning topics are:

1. How to generate a sinusoid using a look up table.
2. How to generate and use a square-root raised cosine pulse shaping filter.
3. How to implement a BPSK modulator.
4. How to implement a BPSK demodulator.

LABORATORY STRUCTURE IN MOODLE

The DSP laboratory is divided in four parts, as listed in the previous section, with each part consisting of the following sections:

1. Objectives and Learning Topics: the actual content is listed for each part in the previous section.
2. Materials: lists the equipment, files, and software tools required to complete the lab.
3. Background: provides introductory information required for completing the lab.
4. Procedure: a sequence of steps needed to configure the hardware and software in preparation for performing the following tasks.
5. Tasks: a set of tasks that are used to direct the student in exploring relevant DSP hardware and software concepts. Each task asks a question that the student must answer and enter into the response box. Multiple attempts are allowed, with a 10% penalty for each incorrect answer.
6. Upload Code: Any DSP software programs that are produced by the student are uploaded to the website for marking.

The first four items in the above list are implemented as a Lesson in Moodle. As presented in the Moodle help facility: A lesson delivers content in an interesting and flexible way. It consists of a number of pages. Each page normally ends with a question and a number of possible answers. Depending on the student's choice of answer they either progress to the next page or are taken back to a previous page. Navigation through the lesson can be straight forward or complex, depending largely on the structure of the material being presented.

The Tasks are implemented as a Quiz and quizzes are discussed in detail in the next section. Finally, the Upload Code item is implemented as an Assignment. As presented in the Moodle help facility: Assignments allow the teacher to specify a task that requires students to prepare digital content (any format) and submit it by uploading it to the server. Typical assignments include essays, projects, reports and so on. This module includes grading facilities.

All of the laboratory content (items 1 to 5 in the above list) was initially developed as a LaTeX document [3]. Once the lab has been debugged it is converted to an HTML document in Moodle, which involves copying and pasting the text and LaTeX equations into the Moodle documents. jsMath is used to display the LaTeX equations [4]. Note that in Moodle the jsMath filter must be enabled by an Administrator. jsMath basically includes a LaTeX equation parser in JavaScript, rendering the equation using special fonts and/or image based fonts.

MOODLE QUIZZES

The laboratory tasks are implemented using the quiz module in Moodle. The quiz module can be used to generate various question types, such as multiple choice, short answer, numerical, true-false, matching, embedded answers (cloze), random short-answer matching, calculated questions and essay (Moodle version 1.9). These quiz questions are placed in a categorized database, creating a pool of questions that can be re-used in the same course or between courses. Quizzes can be generated using specific questions from the pool or by using a random selection of questions. Quizzes can allow students to have multiple attempts at a question, with each attempt being automatically marked. There is also a versatile system for providing feedback and/or the correct answers. A description of Moodle quizzes and the logistics of generating a quiz are given in chapter 6 of [5].

The only quiz question type that can be directly used to individualize a task is the calculated question. A description of the calculated question, taken directly from the Moodle help facility with some modifications follows. A Random question can also be used to individualize tasks and this approach is discussed later in this section.

Calculated questions offer a way to create individual numerical questions by the use of wildcards that are substituted with individual values when the quiz is taken. Below is a shrunken view of the main quiz editing page with some example inputs:

Question: How much is {a} + {b} ?

Image to display: None

Correct Answer Formula: {a} + {b}

Tolerance: 0.01

Tolerance Type: Relative

Significant Figures: 2

In both the text for the question and the formula for the answer, {a} and {b} can be seen. These and any other {name} can be used as a wildcard that is substituted with some value when the quiz is taken. Also, when the completed quiz is submitted by the student, the correct answer is calculated using the expression in "Correct Answer Formula". The answer is calculated as a numerical expression after the substitution of the wildcards. The possible wildcard values are defined on a later page in "editing wizard" for calculated questions.

The formula in the above example uses the addition operator, +. Other accepted operators are -, *, / and %, where % is the modulo operator. It is also possible to use some PHP-style mathematical functions (PHP is a popular scripting language). Among these there are 24 single-argument function: abs, acos, acosh, asin, asinh, atan, atanh, ceil, cos, cosh, deg2rad, exp, expm1, floor, log, log10, log1p, rad2deg, round, sin, sinh, sqrt, tan, tanh, and two two-argument functions atan2, pow and the functions min and max that can take two or more arguments. It is also possible to use the function pi() that takes no arguments. Similarly, the other functions must have their argument(s) within parentheses. Possible usage is, for example, sin({a}) + cos({b}) * 2. It is also possible to include functions within each other, such as cos(deg2rad({a}+90)). More details on how to use these PHP-style functions can be found in the documentation at the PHP website [6].

The numerical questions allow a margin within which all responses are accepted as correct. The "Tolerance" field is used for this. However, there are three different types of tolerances. These are Relative, Nominal and Geometric. These terms are defined in the Moodle help facility.

The field Significant Figures relates to how the correct answer should be presented. Examples: If it is set to 3, then if the correct answer is 13.333, it would be presented as 13.3; 1236 would be presented as 1240; 23 would be presented as 23.0.

It is also possible to approximate an individualized task using a Random question within a quiz. A Random question is replaced by a randomly-chosen question from the category that was set. The description of a Random question, taken with modifications from the Moodle help facility follows.

Random Questions may be added to any category. When you add a Random Question to a quiz it will be replaced with a randomly-chosen question from the same category. This means that different students are likely to get a different selection of questions when they attempt this quiz.

The same question will never appear twice in an attempt. If you include several Random Questions then

different questions will always be chosen for each of them. If you mix Random Questions with non-random questions then the random questions will be chosen so that they do not duplicate one of the non-random questions. This does imply that you need to provide enough questions in the category from which the random questions are chosen. The more questions you provide the more likely it will be that students get different questions on each attempt.

LAB TASKS

The majority of the DSP laboratory was implemented using the Moodle structure previously presented. The Tasks sections in Part A, B and C were completely implemented using the Moodle quiz module. The Part D tasks were implemented as a combination of a calculated question in a Moodle quiz along with a MATLAB script to produce unique BPSK modulated signals.

The breakdown of the quiz question types used to implement the task questions in Parts A, B and C and the number of questions of each type are given in Table 1. As shown in the table, there were a total of 65 task questions in Parts A, B, and C.

TABLE I
BREAKDOWN OF TASK QUESTION TYPES

Section	Matched	Numerical	Multiple Choice	Short Answer	Random	Calculated	Total
Part A		3	5	3	6	3	20
Part B		2	7	5	1	2	17
Part C	1	7	4	12		4	28
Total	1	12	16	20	7	9	65

The initial goal in developing the lab was to use individualized questions for all of the task questions. Unfortunately, this was not possible in the first year, thus as Table 1 shows, 25% of the questions are individualized (the Random and Calculated questions). The difficulty is in posing the question such that there can be a random component. This can be a time-consuming process and thus a decision was made to individualize the tasks over 2 to 3 years.

An example of an individualized question using the calculated type quiz question is displayed in Figure 1. The question in Figure 1 reinforces two VisualDSP++ features. First, the ability to change the contents of registers at a breakpoint and the method used to change the displayed number format of the registers. The question is individualized in the sense that the hexadecimal number is different for each lab group. This eliminates copying between groups and forces each group to perform the operations requested in the task question. The actual

question text used in the Moodle question is “Determine the unsigned integer equivalent of FE{a}{b} hexadecimal by writing the value to the data register R4 and then reading it in integer format”. Note that, as discussed in the previous section, the random component is defined by the two wildcard variables {a} and {b}. The correct answer is calculated using the expression $65024 + \{a\} * 16 + \{b\}$. In the Moodle quiz question a dataset is generated for each variable. In this problem, the variables can take on the integer values from 1 to 9, which gives a total of $9 * 9 = 81$ different combinations.

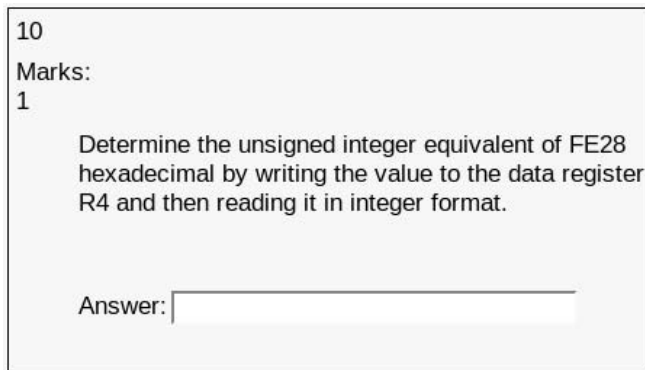


FIGURE 1
EXAMPLE OF AN INDIVIDUALIZED TASK QUESTION

Another task question, which focuses on DSP concepts, is shown in Figure 2. In this question the number 0.38 is generated using a wildcard variable named {a}. The expression used to calculate the correct answer is $\text{abs}(\sin(4 * \pi * \{a\}) / (\sin(\pi * \{a\}) * 4))$. The defined data set for {a} ranges from 0.26 to 0.49 in steps of 0.01. In this problem the dataset could be expanded by increasing the range (it is limited to a maximum range of 0 to 0.5, which is the maximum range for positive frequencies) or increasing the number of decimal places to 3 from 2, for example. Note that the question in Figure 2 has a Submit button, since this task question employed an adaptive mode, which is described later in the paper.

Digital signal processing, as the name implies, involves processing signals. Individualizing tasks that involve generating and processing signals is not possible in Moodle because of its limited signal processing capabilities. This can be overcome by combining the calculated question in a Moodle quiz with the signal generation features of MATLAB. The key is defining a suitable interface that uses the random feature of calculated quiz questions and the signal generation capabilities of MATLAB, but still employ the automatic marking feature of Moodle. The approach used here is to use a predefined function to encrypt a key that can be used in generating a signal. This encrypted key is presented to the student through a Moodle calculated question. The student uses this encrypted key as an argument for a MATLAB pcode function, which generates the signal for processing on the EZ-KIT board. Note that a pcode function is a function that has been compiled to generate a binary version, whose content is not viewable by

the student. The student can then process the MATLAB generated signal using the DSP hardware to produce an answer that can be entered into the Moodle calculated question for marking. The signal can be transferred to the DSP board by connecting the computer sound card output to the DSP board ADC input, if the signal is an audio signal. Or, the signal can be saved to a text file and imported into the DSP program using VisualDSP++.

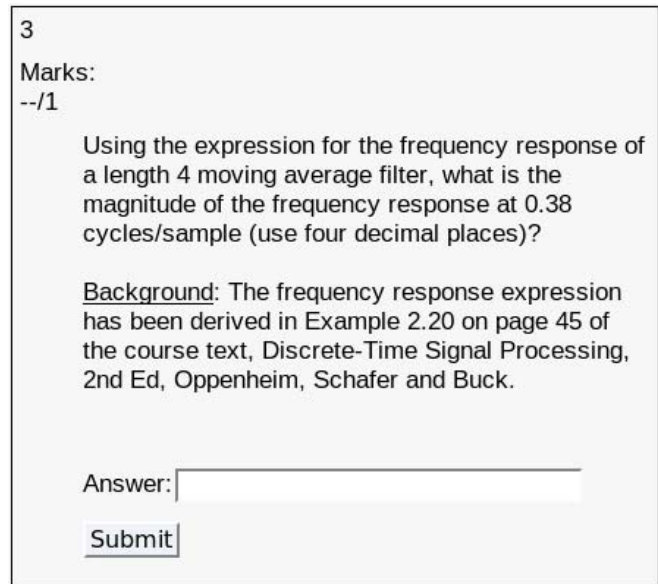



FIGURE 2
EXAMPLE OF AN INDIVIDUALIZED DSP TASK QUESTION

An example of this approach, which is used in Part D of the lab, is given here. Part D involves an implementation of a BPSK (Binary Phase Shift Keying) modulator and demodulator. The Moodle question for the example task is given in Figure 3. In this task, the student generates a BPSK signal using the provided MATLAB pcode function `gen_bpsk_sig.p` with the key value specified in the task question. The key is an encrypted decimal value that is converted to a binary number using `gen_bpsk_sig.p`. In the pcode function, this binary sequence is used to modulate a carrier waveform and the signal generated is saved in a file. In this implementation a SRRC (square-root raised cosine) pulse shaping filter is used. The student must modify the assembly program implementation of a demodulator to estimate the transmitted binary sequence from the signal in the saved file. The decimal equivalent of this binary number is entered as the answer to the task. Once the Submit button is pressed, Moodle compares the answer with the number that was randomly generated.

DISCUSSION AND SUMMARY

A new DSP laboratory was developed and implemented using Moodle. Initially, the lab was written in LaTeX to provide a single printed document that could be quickly accessed. This document was converted to the Moodle documents relatively painlessly by copying and pasting. The

advantage of LaTeX is that mathematical equations are easy to develop and they can be directly used in the Moodle documents.

3 

Marks:
--/1

A BPSK signal is generated by executing the following command in MATLAB,
`sig=gen_bpsk_sig(key);`
 where key contains the value 752329 (note that `gen_bpsk_sig.p` is a pcode function). This signal, which is saved in the file `bpsk_sig.dat`, has a bit rate of $R_b = 960$ bits/second, a sampling rate of $F_s = 48000$ samples/second and a carrier frequency of $\omega_c = 2\pi/24$ radians/sample. Modify the BPSK demodulator implementation for the Blackfin to decode the binary sequence. Convert the binary sequence to a decimal number and enter the answer below.

Answer:

FIGURE 3

EXAMPLE OF AN INDIVIDUALIZED SIGNAL GENERATION QUESTION

The initial intent was to generate individualized questions for all of the task questions. It turned out that posing individualized task questions using the available features for Moodle calculated and random quiz questions was very time consuming. Thus, the decision was made to individualize the task questions in stages over 2 to 3 years. In the first year 25% of the tasks were individualized.

The student feedback to the Moodle implementation of the task questions was generally positive. One of the multiple choice survey questions was: I like the way the Tasks are organized, where you have to provide an answer to each task. Students could select one of five response, Strongly Disagree, Disagree, Neutral, Agree and Strongly Agree. Of the 27 students that responded, 67% Agreed or Strongly Agreed, 30% were Neutral and 4% Disagreed. The two significant complaints the students had with the Moodle implementation were: not being able to view the task questions after the tasks were submitted, and not having any feedback on whether the submitted answers were correct. The problem of viewing completed tasks was solved by providing the task questions in a separate HTML document. The problem of answer feedback was the result of the decision not to provide answers, since only 25% of the questions were individualized. Though a change was made

in Part C tasks to address this complaint. This involved using the adaptive mode in the task questions. In adaptive mode, students can submit an answer and the system informs the student whether it is correct or not. If it is not correct, the student is penalized 10% of the question mark, but he/she can submit another answer.

Overall, this was a successful project. Moodle can be used to individualize labs, such that the labs can be used a number of years without losing their effectiveness due to copying of lab book results. Though, developing individualized tasks is time consuming and is best completed in stages.

REFERENCES

- [1] The Website for Moodle, <http://moodle.org>
- [2] The Mathworks Website for MATLAB, <http://www.mathworks.com>.
- [3] The Website for Latex, <http://www.latex-project.org>
- [4] The Website for jsMath, <http://www.math.union.edu/~dpvc/jsMath/>
- [5] Cole, Janson and Foster, Helen, Using Moodle: Teaching with the Popular Open Source Course Management System, O'Reilly, 2007 (online version available at docs.moodle.org/en/Using_Moodle_book)
- [6] The PHP Math Function Website, <http://www.php.net/manual/en/ref.math.php>