General Lab Policies and Procedures

ECE 341 — University of Idaho

Attendance

- Attendance is mandatory.
- Every minute you are late you will be penalized 5% of that lab's entire grade. This means that 20 minutes late will carry a penalty of 100%.
- If you satisfactorily demonstrate your lab to the TA before the end of the lab session, you may leave the lab session early.
- Important and helpful information will typically be presented at the beginning of the lab session- information such as what is required in the report, clarification on specifications for the lab, as well as the questions and topics you should address in the report conclusion.
- The lab section is your time to work on the lab with the TA available- take advantage of this fact!

Lab Preparation

- You are expected to come to your assigned lab section prepared to start writing code. This means that you have read and familiarized yourself with:
  - The lab handout.
  - The datasheet and/or reference manual for each new peripheral, protocol, component, and software libraries we are working with.
  - The relevant chapters in the course textbook.
- You are strongly encouraged to start the labs early as they can be deceptively difficult.
- These labs build on each other. If you don't get this week's lab working then next week's won't work either.

Deliverables

For every lab, there are three parts. Each part is due at a particular time, and worth a certain portion of the total lab grade. The prelab will be collected at the start of new lab and the demonstration must be performed by the end of the week. The lab report will be collected in lab the next week. **Late work is not accepted.**

Submission	Deadline	Weight
Prelab	Hardcopy before lab	20%
Demonstration	Friday, 3:00 PM	50%
Report	Hardcopy in lab the next week	30%

General expectations are described in the next three sections about the prelab, the demonstration, and the report. These are the expectations for every lab deliverable, unless otherwise stated on the webpage, in class, or in lab. Each lab will have a lab handout that explains concepts related to that week's lab; these handouts will also contain specifications that must be met for a satisfactory demonstration. When necessary, there will be an additional document posted along with the handout that will help clarify the handout and may modify the expectations below. This additional document will also contain topics and questions that should be addressed in the report.

Prelab

- The prelab is a planning tool to prepare you for the lab. It will consist of two parts: the first is the goal and background information, and the second is your plan to accomplish the goal.
- The goal must be consistent with the lab specifications and should be short, clear and concise.
- After the goal, the prelab should elaborate and expand upon subsystems mentioned in the goal, providing background information.
- The background information should be a summary (*not copied*) of the relevant prelab materials (lab handout, datasheets, textbook, and lecture).
- You must cite your sources in the background summary. It doesn't matter how you cite it (footnotes are fine), but it must be clear where the information comes from.
- Do not simply regurgitate (or worse, copy) the lab handout!
- Only write background information for new components or peripherals.
- The first part of the prelab should be copied into the report as the beginning of the introduction (after it has been updated and corrected if necessary).

- The second part, your plan, will form the groundwork for the new material added to the implementation section of the report.
- The plan details how you intend to accomplish the lab and meet specification.
- While working on the lab, it will be helpful to go back and modify your plan to document your code. This will make the inclusion of the new implementation details into the report trivial. However, don't let modifying your plan take priority over completing the lab and demonstrating it.
- The plan must contain at least two distinct diagrams: one that defines how data flows through the system (the data flow diagram), and another that defines the execution flow of the program (the control flow diagram). Additional information about the diagrams will be provided if necessary.
- The prelab grade will depend on how well the background material is understood, how mature the plan is, and the quality of the diagrams.
- Unless otherwise stated, the prelab shouldn't be more than a page and a half, excluding diagrams. While there is no minimum length, make sure it is long enough to satisfy the requirements above.

Demonstration

- You are responsible for proving your code meets specification.
- You must come up with a set of tests that you'll show the TA. Each test will show at least one specification being met.
- The tests you use to demonstrate your lab to the TA should be subset of the tests you describe in the report (see below).
- During the lab, the TA will test your design to ensure complete functionality.
- Partial credit will be awarded for demonstrations that don't meet specification, just make sure to note the parts that don't work both when you demonstrate and in the report.
- The only way to get a zero for the demonstration is if you don't demonstrate or submit your demonstration on time.

Report

- The report should be a formal, professional report that is clear and concise. It is not required, but students are encouraged to consider using LaTeX.

- The report will consist of four major parts: the introduction, implementation discussion, testing and validation, and the conclusion:
  - Introduction:
    * Will be primarily copied from your prelab, as long as you've kept your prelab plan updated and corrected any misunderstandings in the background information and the goal.
    * Should only contain background information about new peripherals or components.
    * Should transition smoothly into the implementation.
  - Implementation
    * The implementation will document and justify the code you have submitted for your demonstration.
    * You will integrate new modules that are introduced each week into the report as well as update old modules if necessary.
    * Must include both a data flow diagram as well as a control flow diagram that describes your code.
    * The diagrams will become complex; use hierarchy to keep them readable.
    * Must include the majority of your code from the entire lab in the form of "listings" (similar to the lab handouts). The listings generally shouldn't be longer than a half page, should be captioned, and discussed in the text surrounding it.
    * Must describe the role each listing has in the system (its purpose).
    * While this section will become quite large, each week you will only need to add documentation about the new code you wrote for that lab each week.
  - Testing and Verification
    * You are responsible for testing your own project.
    * You must come up with tests that can show if your code meets the specifications of the lab.
    * You should describe your tests in such a way that they are easily repeatable (by the TA).
    * For each test, you should perform that test on your design, document the result, and explain the implications of the results.
    * Remember the instrumentation you have and use the appropriate type for the test. For example, don't use a LED to determine if a delay is exactly one millisecond.
    * The oscilloscope is a very powerful tool and screen captures are great for showing signal timing. After Lab 1, new screen captures from the oscilloscope should be included with every lab report (unless stated otherwise).

    - Each test should prove or disprove at least one specification. Examples will be discussed in lab if there are questions.
    - These tests should be developed while writing code for the lab. For example, if you write a function that writes a character to an LCD, you should test that function to determine if it works before using that function in another that writes a string.
  - Conclusion
    - This will vary the most from lab to lab, but it will always contain a discussion about the limitations of your design and the microcontroller. For example, if we are working with a timer, what is the maximum interval we can measure? What is the minimum interval we can measure? Always take parameters to the limit and ask, "What is the limiting factor here and why?"
    - Must address specific topics and questions specific to the lab which will be posted with the lab handout.
    - The answers should flow together in a few paragraphs with full sentences. Sentence fragments will not be accepted.

Academic Honesty

- The labs are not a team-based effort, they are individual labs and you are expected to work alone and come up with your own designs.
- If you copy code/reports/diagrams/prelabs and it is the first offense, you will receive a zero for that particular lab.
- If you copy code/reports/diagrams/prelabs again, you will receive a failing final lab grade.
- If you are ever in doubt about what constitutes plagiarism, ask the TA. If you ask the TA and it turns out it is plagiarism, you will not be punished *as long as you don't submit the code.* Plagiarism is only punishable if it is submitted for grading.
- You **are allowed to**
  - Look at code in the book, in the datasheets, in handouts, and from the lecture and use that code as a guide for you to write your own code. If you do, informally cite the source in a comment.
  - Refer to code on the web **only** if it is for referencing the syntax or semantics of the C programming language (how to write a switch-case block, for example) and standard libraries (the function `printf`, for example).

  - Help your classmate with general semantics of the C programming language. For example helping a friend with pointers is just fine; telling a friend how to correctly initialize the I2C bus is not.
  - Help each other by fixing syntax errors

- You **are *NOT* allowed to**
  - Share or copy someone else's code
  - Look at someone else's code or from the internet
  - Look at code pertaining to the lab from the internet. There are past labs that are available online; if you are caught looking at, referring to, or otherwise using those labs, it will be considered cheating and these rules will apply.
  - Copy code from the book
  - Write code you don't understand
  - Help your classmates with lab-specific details

- You must understand all the code you write for these labs.

- If some code looks suspect, you will be confronted and asked about it. A discussion will take place that will essentially assess how well you understand the code in question. If you don't understand the code you turned in, it will likely be considered plagiarism.