Lab Overview and Policies CSE/ECE 474

University of Washington 10-Apr-2023

Spring 2023

Vikram Iyer[1]
Kyle Johnson
Vicente Arroyos
Joe Breda

Introduction to the Labs

The Lab assignments are (see 474 main spreadsheet for links):

- Lab 1 Getting Started with the Arduino Mega
- Lab 2 Digital I/O and timing of outputs
- Lab 3 Round Robin Scheduling and multitasking
- Lab 4 FreeRTOS and Project

This document contains general advice and policy for the labs.

LAB ASSIGNMENTS

The lab projects are a significant part of your grade in the course. Each lab **builds on the previous lab**, so it's important that you keep up with assignments and also ensure that your designs are robust and well tested.

Project Teams

An important part of working as an engineer is working with different people to design, develop, then bring together all the elements of a project into a working system. To this end, we will be working in 2 person teams as we design and develop our embedded systems. When selecting a partner(s), please choose someone you can work with. Remember, different people may like to work different hours or have different work habits … check this out early, not halfway through the final project. We expect each person to contribute equally to the final design and implementation of each lab.

[1] Thanks for valuable guidance from Prof. Blake Hannaford, Dr. Jennifer Vining and Prof. Rania Hussein

Lab Demos

Your team must perform a demo for each project. There will be a specific "turn-in and lab demo requirements document" for each lab. Each team member must briefly describe what he or she did on the project and present the items from the lab demo. Point distribution for each lab demo depends on the specific lab and is given in the individual lab's Lab Demo sheet. Lab demos may include some or all tests specified in the turn-in requirements, but could also include other tests at the option of the TA or instructor. Demos may include asking the students to perform oscilloscope tests, or measurements of voltage, current, or resistance.

Q&A

Instructors/TAs will ask questions during the demo of all team members. Be prepared to quickly bring up the code for any part of the lab at the request of the instructor or TA. **All team members must be able to answer any question about any part of the lab work**. It is not OK to say "I can't answer because my partner did that part of the project." Of course, we expect that the team member who did the most work on any project part will have a better, more in-depth answer, and the team member who did little or no work on the part will have a less detailed answer.

Lab Reports

Each lab will require a lab report prepared jointly by your team (see requirements below). Point distribution for lab reports is given in the Lab Report Rubric (below), which remains the same for every lab report. Lab report due dates are posted in the course schedule on Canvas.

Lab Grading

See the syllabus for late grading policies. The final lab grade is a combination of Lab Demo points and Lab Report points. You must show all of your work for all problem solutions. Solutions will be marked down if work is not shown. Work will be given a 0 grade if the answer has been copied from a solution manual, someone else's report, or some website without attribution. Incidents of plagiarism such as these will be reported to the College of Engineering via its official policies at the sole discretion of the instructor.

In addition to deliverables identified in Lab Report Rubric, each project must include:

1. A significant contribution by each team member
2. Program source code submitted to Canvas in a zip file
3. The Lab Report (in **.pdf** format only)
4. All code must be readable and written in a consistent style!
   See the Embedded Systems Coding Standard document.

Lab Report Requirements

The lab report must be completed by the team and turned in via both student's canvas pages. The requirements for the lab report are:

1) The document must be a PDF and must follow the CSE474 Document Template.
2) We suggest using Google docs to prepare the PDF but you may use any software as long as the Template is followed.
3) Do not repeat any material from the lab assignment document in your report.
4) Your report should contain the following sections in this order.
   Pages must be single-spaced.
   a) **Introduction: (about ¼ page)**
      Summarize the lab's learning objectives and key activities in your own words.
   b) **Methods and Techniques (½ page)**
      Explain the activities, skills, instruments, and tests you used to achieve the learning objectives.
   c) **Experimental Results (1-3 pages as necessary)**
      Describe specific test results which show how your code and hardware worked together to meet the lab requirements. Include 1-3 photos of your hardware setup.
   d) **Code Documentation (1-3 pages as necessary)**
      Write a short paragraph for each task and each initialization process in your software. Identify the line number ranges for the tasks, function and struct names, etc in your paragraph. If some code is re-used from a previous lab, do not repeat its documentation. Reference the specific paragraph of your previous report.
   e) **Overall performance summary (½ page)**
      Describe the success and failures (if any) of your demo session or video demo. And how well you accomplished the learning objectives.
   f) **Teamwork Breakdown (¼-½ page)**
      Describe which team members worked on which parts. List their specific activities including architecture, coding, debugging, hardware integration.
   g) **Discussion and conclusions (½ page)**
      Describe the most challenging parts of this lab for your team. Describe parts you are especially proud of. Describe things you learned *in addition* to the official learning objectives.
5) **Coding Standard.**
   You must turn in your C source code files (turn-in format to be announced). Your code must follow the CSE474 coding standard.
6) **Code.**
   Turn in ALL `.ino`, `.h` and `.c` files in the project. Submit a zipped folder containing all code. Comment out (but do not delete) code which might be required for intermediate steps but is not required for the final code. This will help us give you partial credit!

Lab Grading Rubric

Criterion	Weak	Fair to OK	Excellent	Grading Weight
Demonstrates all lab functions	Less than 80% working	80-95% working	95-100% working	40%
	"Working" means correct logic, correct timing, consistent (glitch-free) operation.			
Q&A by both team members	Many gaps or confusion is evident.	Some gaps demonstrated by at least one team member	Clear explanations of all code and hardware	20%
Report Tech completeness	Missing required information. Errors.	Some missing information, sloppy measurements or reporting	Follows correct outline and template. Clear test results and facts.	25%
Code Standard Compliance	Minimal. Code is hard to read.	Some gaps in code standard compliance.	Complete code standard compliance.	10%
Report Professional Communication	Sloppy, >2 spelling and grammar errors per page.	~1 spelling grammar error per page.	Clear, proof-read, coherent explanations.	5%

Pro Tips for CSE474

TIME MANAGEMENT

Software development often takes a lot more time than we expect. Frustration can be high as we develop things and learn new tools; please try to hang in there and make sure to ask for help rather than spend countless hours making no progress. Remember the old saying, “We had better get the coding done quickly because we have a lot of debugging to do.” This is a poor approach we see all too often even with professional programmers. We strongly encourage you to start each design by thinking it through at the architectural level first. Then develop, test, and integrate the individual pieces. The people who spend time at the start of each lab thinking through their design before writing a single line of code finish well ahead of those who don't.

Here is a BAD STRATEGY that frequently trips up 474 students:

1) Divide the lab assignment into two parts
2) Go off and code
3) Put the parts together 24 hours before the due date.
4) Tank the lab.

The main difficulty of these labs is *making the parts work together smoothly as a whole.*
A better strategy is:

1) Get together (in person or Zoom) and plan the architecture of your completed code: Example questions to answer first: **exactly** how will the `loop()` function or scheduler work? What is the exact function prototype of each task? What data types or data structures will be used for shared information between the tasks? Work together on your `#defines` to make sure you use the same names for stuff.
2) Divide the lab assignment into two parts
3) Go off and code
4) Merge and test your code (adding one task at a time) at least 3-4 days before the due date.
5) Debug and prepare your demo.
6) Get a great grade on your lab!