CMPT 433 News
Notes & Schedule
Assignments
Project
Course Info
Links

CMPT 433 Summer 2018 - Dr. Fraser

Course Project

Project Description

Form groups of 3-4 students and work on a course project. You can choose your own topic (see below for topic constraints). You will be evaluated as a group with some marks for individual effort (students who do not contribute much will get a low mark; minimum of 0 on the project). I strongly recommend you create a repository on SFU's GitLab server for your project.

Criteria for selecting a project

- Must make good use of an embedded Linux system, such as the BeagleBone.
  Your project must run on target hardware for this course.
- Should interface with an external system, such as having a webpage, or use Ethernet, USB, RS232, GPIO, I2C.... Perhaps audio.
- Should be focused on software (not on hardware design). OK to create application software (user space), or work in lower levels such as kernel, U-Boot, or bare metal.
- Must have a significant amount of code implemented in C or C++. May use other languages as well.
- Must be an impressive display of your group's embedded software development skills.
- List of project suggestions to help inspire you! Includes list of hardware in the student hardware packages.
- List of expected project elements to guide/inspire you.
- Posted list of hardware I have to lend.

Dates (tentative)

- June 7th (Thur): Project Proposal
- June 18th (Mon): Milestone 1
- July 9th (Wed): Milestone 2
- July 30th (Mon): Demo day (Noon-4pm, ASB North Atrium)
- Aug 3rd (Fri): Project Submission: Write up, DVD, How-To-Guide
- Aug 3rd (Fri): Individual Peer Evaluation

[0] Project Proposal

- Describe what you have in mind. You should have done some research already, and likely started on the first steps of the project. Even if your proposal is not accepted as is, it's likely a revision will make it acceptable, so don't wait to get started!
- You may later change what is in your project. This is just a plan; you will not be marked for hitting this plan.
- Length: 1.5 pages; may be longer if adding figures.
- **Sections:**
  - Group Information:
    - Group name.
    - Group members: names and SFU email's.
    - *Note: Create your group in CourSys when submitting.*
  - Topic description (1-2 paragraphs):
    - What will your system do, and how? Add a diagram if it is has many components.
    - Describe connections to any external systems.
    - Any required additional hardware (students must provide). If using special hardware, include mention of its Linux support (drivers, libraries, guides...).
  - Time-line/Project plan (10-20 lines/points):
    - For each of the two Milestones, provide an estimate of which features will be working by then. You won't be marked on hitting all these points, but it gives us a place to start discussing how it's going.
      - Put some thought into what tasks are required, and breaking down the project into manageable stages. Think of what the most basic features are, and what is needed to get this working. Focus your efforts on getting something working, and then add to it little by little. This is critical if a component take longer than expected to get working!
      - Hint: Work on the hard parts first, such as interfacing with hardware or external systems. If you leave this for last it's much more likely that problems will prevent you from getting it working.
    - List the high-level stages/tasks to complete (for the minimal part, and the full system).
    - What is the minimal configuration that will let your system work?
    - What tasks do you expect might be the most time-consuming?
- See due date above; submit via CourSys. For faster feedback, submit earlier (and email Dr. Fraser) and you'll receive feedback as soon as possible.
- If interested in any of the hardware I have to lend (or when applicable, having Computing Science buy additional hardware), email me and I'll figure out who gets what shortly after the proposal is due. Interesting sites to consider for hardware (consider Grove hardware for easy connectivity) (ensure parts are 3V, not 5V!):
  - AdaFruit
  - RobotShop
  - Digikey
  - RP Electronics (local)
  - Lee's Electronics (local)
  - Seeeds Studio
- It's not worth any marks, but it's where I'll give you feedback on your idea, its level of difficulty, and any other issues that come to mind. Putting more effort into the proposal will help your group pick a better project, and hit the ground running.
- Proposal marking guide (sort of).

[2] Milestone 1

- Submit a 1/2 page description of
  - 2 sentence description of your project (to refresh my memory of your project),
  - What has been accomplished so far, and
  - What has fallen behind.
- Briefly mention any expected project changes from your project proposal.
- Must have some progress to earn these marks!

[8] Milestone 2

- Submit a document containing the following:
  - 2 sentence description of your project (to refresh my memory of your project).
  - Discussion of significant accomplishments (~0.5 to 2.0 pages). Three points must have proof submitted showing the accomplishment (see below). For each of these proofs, describe what is being shown in the "proof" and how it demonstrate the accomplishment. You can include the "proof" in the document or add it to the ZIP file you are submitting.
  - Discussion of any roadblocks or unexpected challenges (<0.5 page).
  - Discussion of significant differences between what has been accomplished and the plan in your project proposal (<0.5 page).
- Proof of accomplishments:
  - Pick the three best accomplishments so far and submit some from or proof to show their state.
  - Many things acceptable for "proof", such as: screen shot, photo of something in operation, text-capture of application running, directory listing of cross-compiled library, ... For videos, please upload to YouTube (or the like) and include a link in a text file or your write up.
  - Each of these "proofs" must be referenced and explained in your document, as described above.
- Possible in-class discussion of project with instructor and/or other students. TBA if happening and the date.
- Must have good progress to earn these marks!
- **Individual report on progress**
  - Each student must use the Peer Feedback JSON Generator to provide very brief feedback on team members.
  - Each student must submit the JSON file it generates to CourSys.
  - This feedback will be used by the instructor to identify group problems; it will not be distributed to team mates.

[10] How-To guide

- Group submits a step-by-step guide explaining one interesting task/topic/aspect you figured out for your project.
- Examples:
  - How to connect a USB camera to the board, compile & install drivers, and use it via a simple C program.
  - How to smoothly display video on an LCD.
  - How to write a driver to control an external IO board over I2C.
  - How to download, cross-compile, and install/use some useful tool/library.
  - How to cleanly interface to a library or tool.
- Should be 3-5 pages long.
  May be longer if includes many screen-shots.
- Should guide the reader through the task.
- Marked based on topic choice (must be non-trivial) and quality of explanation.
  - Must be a step-by-step guide.
  - Include screen shots, and sample output as required/reasonable.
  - May assume only what the average CMPT433 student might know.
  - Cannot replicate any guides provided by the instructor.
  - May not plagiarize online guides.
    It's OK to cite other guides, but your guide must be your own work.
  - Include troubleshooting steps to cover some likely errors.
  - Must be clearly written and easy to follow.
- Must give copyright permission to the instructor for your guide to be posted online for future students to access. You retain copyright.
- May be submitted as an HTML or PDF document. If HTML, include all necessary images, sample code and resources in the submitted zip file. Submit via CourSys.
- Must *not* include SFU student numbers! Will be posted online.
- How-To guide marking guide.

[10] Demo

- Demo session to be held in SFU Burnaby's Applied Science Building (ASB) Atrium.
- Time: Noon - 4pm, with ~8 minutes for instructor to view your demo.
- Must have a sign with:
  - Content large enough to read from 3 feet away. One or two pieces of paper an OK size.
  - Displays project title, group name, and student names (no student numbers).
  - State your project's goal/topic.
  - Explain system with block diagrams (physical connections, and/or software design such as libraries or other programs). You must make it clear what work your group completed and what parts are provided by a 3rd party.
- Demo your embedded system (~5 min).
- Instructor and TA will ask questions about the system and its functionality. This will help evaluate the difficulty and quality of the product.
- Marking:
  - Marked for general quality of demo (and sign content).
  - Each group must have at least one group member present through the demo session. (Special arrangements possible with Dr. Fraser if no group members can be present for some portion of the session).
- Bring your own power-bars and possibly an extension cord. There are power-plugs near each demo space. Pick any free space and be setup starting at noon.
- Demo marking guide.

[110] Project Work

- Project write up (5-8 pages):
  - **System Explanation:** What does your system do? How does it do it? Include screen shots, pictures, and diagrams where possible. State important things not working well (such as video streaming at .5 FPS, or bad sound, or flaky web

page). (I realize that a project can still be great while having some remaining challenges to solve).
    - **Feature table:** Create a table of important features with columns of
        1. Description: *briefly* describe each feature in < ~15 words
        2. Completeness: rate the feature's completeness from 1 to 5; see list below for explanation of values
        3. Code: state the type(s) of code used to implement the feature (C, C++, Java, Perl, Script, systemd, ...). If multiple languages used then list % for each language.
        4. Notes: If not 100% your own code, state what you used of others'. Also if feature not 5/5 complete, explain what works or what is broken. Ex: "Does not detect new USB devices. 20% sample code, 80% us; cross compiled".
    - **Feature Completeness Number Guide:**(pick the number best representing each feature (fractional OK, like 3.5)
        5. Feature 99+% completed and well tested. Feature works as desired almost all the time. Code is clean and fully integrated into the overall project. No (few) problems are known.
        4. Feature is mostly done and integrated into the overall project, but may lack some polish. Tested and works for normal cases but may fail on some unusual cases.
        3. Feature is partly complete and integrated into main project. It is usable, but only at a proof of concept level. May have many bugs, but still demonstrates it is a good workable idea and implementation. If given one more week to work on the project, the team could likely complete and fully test the feature (no extra time will be given, however!).
        2. Feature has a proof-of-concept implementation, but not integrated into the project. For example, have a program which can display to an LCD screen, but it has not been integrated into rest of project. Or, extensive research was done into the feature but it never worked out (explain work done in "note" section).
        1. Feature had some research or preliminary work done on it. No working code in project, or the code was scrapped because it really did not work as required. Group did not invest a large amount of time on the feature.
        0. No (real) work done. It's not actually a feature, it's just a dream. *So don't even list it!*
    - **Extra Hardware & Software Used:** Beyond the BeagleBone, Zen Cape, and the course's guides/notes, what hardware or software libraries did you use (if any)?
    - **(optional) Explanation of Challenges:** What slowed you down or took up a lot of time?
    - **Acknowledgements:** Who helped you, and what guides did you use?
- Code Submission (via CourSys, or CD/DVD/Flash-Drive):
    - All your code.
    - A README file to describe how to build and run the project using provided pre-compiled files.
    - Should include automated build tool/script (Makefile or the like) as much as possible. However, it is not likely that we will try and build and run your project.
- Project and Write-up Marking Guide contain more details.

Marking Support/Feedback

The following deliverable is to give each student peer feedback and to assist in grading each student. Generally, each student in a group will get a very similar mark; however, a student who is not an asset to his/her group may get 0 on the project.

[0] Individual Peer Review

Each group member submits their own self and peer review. High-quality comments are required; should be in the form of constructive feedback and supported by evidence. Students who do not submit this review may get a 0 on the *project*.

Each group member provides a score and comments about each other group member's skills. The *comments* are distributed to the other group member, but the individual *scores* are not. Students will see the adjustment factor applied to their work (see below) and so may be able to calculate the average score assigned by peers.

Use the Peer Feedback JSON Generator to write your comments and generate the JSON file to submit. **Read the directions at the top of the tool.** Your comments should address the following elements:

- **Team Contribution**
    - Did they contribute their fair share to the project?
      If not, did they only do 75% of the expected work? 50%? 10%?
    - What role did the person play in the group?
    - Were they easy to work with?
    - How can they improve?
- **Technical Skill**
    - Did the quality of their work meet your expectations?
    - How good was their code? How good was their work on documents?
    - Did they do anything that impressed you?
    - How can they improve?
- **Communication**
    - Were they good at communicating with the group?
    - How quickly did this person respond to messages?
    - Did you know what this person was doing almost all the time?
    - Were any difficulties resolved quickly?
    - How can they improve?

Scaling from Peer Feedback

In general, students will have their overall project mark scaled by between 0.85 (15% penalty) and 1.15 (15% bonus) in accordance with the following graph. This usually has a limited impact on most students' project grades, but does recognize some differences between individual contributions. Exceptional cases will be handled manually by the instructor in order to better ensure student contribution to the project is strongly reflected in individual student grades.
(Attempts to game the system with unfair evaluations will be handled manually, and unfavourably!)

- Let A be the average of all group member evaluations of you (excluding your evaluation of yourself).
- A is clamped to be between 10 ("half effort") and 40 ("double effort").
- A student's score on the entire project will be multiplied by the following weighting factor:
  $\text{weight}(A) = 0.65 + 0.0225A - 0.00025A^2$

- This formula gives the desired properties of A=10 gives a weight of 0.85; A=20 gives weight 1.0; A=40 gives weight 1.15.
- If you feel that extenuating circumstances apply to you, please submit that in the private notes section of your peer-review feedback (above).
- This approach is based on the work of Bill Gardner, and used by Ted Kirkpatrick.