Department of Systems and Computer Engineering

SYSC 3010
Computer Systems Development Project

Calendar description

Development of expertise in designing, implementing and testing industrial-quality embedded systems through team projects. Applying modern programming languages, system design practices, current development processes (refactoring, iterative and incremental development) as well as current team-management tools (communication, version control) to medium-scale projects.

Includes: Experiential Learning Activity.

Lectures two hours a week, laboratory three hours a week.

http://calendar.carleton.ca/undergrad/courses/SYSC/

Prerequisites

SYSC 2100 and either SYSC 2003 or SYSC 3310 (may be taken concurrently), and enrolment in Computer Systems Engineering.

Precludes additional credit for SYSC 2101 (no longer offered), SYSC 3110 and COMP 2404.

Prior knowledge

Students should be:

- Proficient in Embedded C programming.
- Proficient in programming in an object orient language, either Python or Java, or both; including graphical user-interfaces.
- Proficient in Embedded Input / Output Programming.
- Conversant with Analog-to-Digital Conversion and Pulse-Width-Modulation.
- Competent in Digital Electronics (designing and wiring circuits involving sensors and actuators).
- Competent in Software Testing.

Course objectives

This course is an introduction to the software development life cycle and to team project management. From a technical perspective, the course is expected to draw from material covered in prior and concurrent courses, with lectures largely providing additional depth or considerations regarding scale and complexity. Complementary to the technical aspects and with equal importance are the challenges associated with working with a team to produce a project on schedule. Resulting from this course a student should see the technical importance of topics already covered in their degree program, have broader appreciation of the design process and aspects such as teamwork, discipline, scheduling and communication. Self-reflection is a key aspect of professionalism and this is encouraged and examined within the course. The expected result will be a deeper appreciation of the importance of technical knowledge, the design cycle and professional skills which will be beneficial for, co-op placements, the fourth-year project and final employment.

List of topics

- Hardware Introduction (Linux, Headless vs Desktop, Programming on RPi versus Downloading)
- Distributed Systems – Networking and JSON
- UML Expectations, Sample Proposals
- Adding Hardware to our RPi and/or Arduino
- SQLite and MongoDB
- System Architecture Diagrams
- Message Dictionaries [& Unit Testing of the System]
- Getting Started with Android
- Git Workflows
- Unit and Integration Testing
- What is a Code Inspection
- Review of Unit Test Plan by Tas
- Demo of Unit Test Plans
- The Bigger Picture:
    - Ethics for Computer Engineers
    - Indigenous Relations in Canada

Learning outcomes

By the end of this course, students should be able to:

- Use industry-standard processes and tools for working effectively in teams, specifically for communication, time management, feedback and version control.
- Describe concepts of iterative and incremental systems development processes.
- Describe the purpose of a proposal, a design document and test plan.
- Describe a system design solution using UML diagrams (architecture, deployment, class, sequence and message dictionaries).
- Explore new technical topics (new language, new computer, new task) independently, going beyond the background course material in their degree so far.
- Describe the importance of interfaces, message protocols as one differentiating aspect of embedded system architecture design as compared to software design.
- Create tests, at both the architectural level and the class level, and for hardware.
- Construct moderately complex systems composed both of embedded computing and web-enabled applications.

- Communicate their design effectively to their peers in both oral and written forms in a collaborative and comparative environment.
- Work cooperatively in groups, scheduling their own work within the time allocated and considering relevant design aspects as safety, performance, cost and product life cycle.
- Reflect on the work and challenges encountered.

Graduate Attributes (GAs)

The Canadian Engineering Accreditation Board requires graduates of engineering programs to possess 12 attributes at the time of graduation. Activities related to the learning outcomes listed above are measured throughout the course and are part of the department's continual improvement process. Graduate attribute measurements will not be taken into consideration in determining a student's grade in the course. For more information, please visit: https://engineerscanada.ca/.

Graduate Attribute	Learning outcome(s)
4.4 Design: Developed: Design solution(s)	3, 4
4.5 Design: Developed: Design implementation / task(s) definition	2, 3, 4, 6
6.1 Individual and Team Work: Developed: Personal and group time management	10, 11
6.3 Individual and Team Work: Developed: Leadership: initiative and mentoring, areas of expertise, and interdisciplinary teams	1, 9
7.2 Communication Skills: Developed: Professional documents: writing, design notes, drawings, attributions, and references	3, 4, 9

Accreditation Units (AUs)

For more information about Accreditation Units, please visit: https://engineerscanada.ca/.

The course has a total of 43 AUs, divided into:

- Engineering Science: 50%
- Engineering Design: 50%

Instructor and TA contact

Specific to course offering (tbd)

Textbook (or other resources)

Specific to course offering (tbd)

Evaluation and grading scheme

Specific to course offering (tbd)

Breakdown of course requirements

Specific to course offering (tbd)

Tentative week-by-week breakdown

Specific to course offering (tbd)

General regulations

Specific to course offering (tbd)