CentraleSupélec

Course catalogue

Courses on the Metz campus during the spring semester

Spring Semester

Metz Campus of CentraleSupélec

last update: July 15, 2026

Contents

1 Projects

SPM-PRJ-003	Software development project	84.0 h	8 ECTS
SPM-PRJ-004	Robotic and AI project	21.0 h	2 ECTS
SPM-PRJ-002	C++ programming project	18.0 h	2 ECTS

2 Programming

SPM-INF-007	C++ programming	42.0 h	4 ECTS
SPM-INF-004	Introduction to Quantum Computing	34.0 h	3 ECTS
SPM-INF-016	Programming paradigms	28.0 h	3 ECTS

Software development project

Course supervisor: Hervé Frezza-Buet

Total: 84.0 h , 8 ECTS **CM:** 2.0 h, **TD:** 6.0 h, **Projet:** 76.0 h

SPM-PRJ-003 *back*

Description: This module focuses on applying software engineering methods to develop a large-scale IT project. This project may involve industrial collaborations and joint work with other specialties. The aim of the module is to gain experience in designing and implementing a project in a situation that is as close as possible to a professional environment.

Learning outcomes: At the end of this course, students will have implemented a professional methodology for carrying out a significant IT development project as part of a team.

Evaluation methods: Report and defense

CM:

1. Serious Game Agile : Intro (2.0 h)

TD:

1. Serious Game Agile 1/4 (1.5 h)
2. Serious Game Agile 2/4 (1.5 h)
3. Serious Game Agile 3/4 (1.5 h)
4. Serious Game Agile 4/4 (1.5 h)

Robotic and AI project

Course supervisor: Hervé Frezza-Buet

Total: 21.0 h , 2 ECTS **Projet:** 21.0 h

SPM-PRJ-004 *back*

Description: The objective of this project is to integrate artificial intelligence algorithms into robotic platforms in order to confront the reality of situated artificial intelligence approaches. It will involve synthesizing the skills acquired in the seventh semester courses and in the robotics course. In addition, emphasis will be placed on experimental rigor (implementation, reporting) in order to prepare students to report on research experiments.

Learning outcomes: At the end of this course, students will have practiced AI outside of manipulating datasets, but on dynamic systems in interaction.

Evaluation methods: Evaluation based on the code deposited on the git and ongoing monitoring by the supervisors

C++ PROGRAMMING PROJECT

Course supervisor: Hervé Frezza-Buet

Total: 18.0 h , 2 ECTS **Projet:** 18.0 h

SPM-PRJ-002 *back*

Description: Programming project for practical application of the C++ language. The code produced will be versioned using Git.

Learning outcomes: At the end of this course, students will have had their first experience of collaborative development and will have learned how to move from a problem to the appropriate coding choices in C++.

Evaluation methods: Evaluation based on the code produced and uploaded to the git. Evaluation will also be based on ongoing monitoring by the supervisors.

External resources:

- C++ project website

C++ PROGRAMMING

Course supervisor: Hervé Frezza-Buet, Frédéric Pennerath

Total: 42.0 h , 4 ECTS **CM:** 15.0 h, **TP:** 24.0 h

SPM-INF-007 *back*

Description: The purpose of this course is to cover the main concepts of programming in C++ (C++20 and later). We will introduce the advantages of strong typing, the object-oriented approach (encapsulation and inheritance, operator overloading, etc.), generic programming (templates, concepts), and the functional approach (function manipulation and lambda functions). All of this will be illustrated using the STL (iterators, ranges) during practical work.

Learning outcomes: By the end of this course, students will have acquired more advanced skills in C++, based on object-oriented programming, functional programming, and generic programming, with the various features offered by the latest versions of the language (ranges, concepts, etc.).

Evaluation methods: 3-hour individual computer test, can be retaken.

External resources:
- C++ website

CM:

1. Types et mémoire (1.5 h)
2. Syntaxe (1.5 h)
3. Goodies (1.5 h)
4. Heritage (1.5 h)
5. Fonctions (1.5 h)
6. Exceptions (1.5 h)
7. Templates 1/4 (1.5 h)
8. Templates 2/4 (1.5 h)
9. Templates 3/4 (1.5 h)
10. Templates 4/4 (1.5 h)

TP:

1. Heritage (4.0 h)
2. Fonctions (4.0 h)
3. Finalisation (4.0 h)
4. Templates 1/3 (4.0 h)
5. Templates 2/3 (4.0 h)
6. Templates 3/3 (4.0 h)

Introduction to Quantum Computing

Course supervisor: Damien Rontani, Stéphane Vialle

Total: 34.0 h , 3 ECTS **CM:** 15.0 h, **TD:** 4.5 h, **TP:** 9.0 h, **Projet:** 5.5 h

SPM-INF-004 *back*

Description: This Quantum Programming and Algorithms course offers a comprehensive dive into the fundamentals and practical applications of quantum accelerators. Students will explore contemporary quantum architectures, including the principles of analog and digital architectures, as well as innovations such as hybrid CPU-QPU, NISQ, and LSQ architectures. The course will cover the formalism of qubits and digital quantum programming, highlighting the importance of superposition and entanglement for quantum computations. The principles and methods of measuring results will also be discussed. An introduction to quantum circuits, including basic gates and early circuits, will enable students to understand the practical basics of quantum programming. Using tools such as QFT, Grover, QPE, and QMC, students will explore classical circuits and their applications, while examining the limitations on NISQ architectures. Finally, students will delve into variational circuits and algorithms, including QAOA and Vxx circuits, and study runtime and performance models for QPUs as well as CPU-QPU loops, familiarizing themselves with the orders of magnitude of current runtimes.

Bibliography:

- Ref. [1] : R. Hundt, Quantum Computing for Programmers, Cambridge University Press (2022)
- Ref. [2] : P. Kaye, R. Laflamme, M. Mosca, An Introduction to Quantum Computing, Oxford University Press (2006)

Learning outcomes: AA1: Identify basic gates and build initial quantum circuits – AA2: Effectively use tools such as QFT, Grover, QPE, and QMC algorithms to solve engineering problems – AA3: Understand and be able to assess the limitations of NISQ architectures and propose appropriate solutions – AA4: Design and implement variational circuits and algorithms, such as QAOA and Vxx – AA5: Evaluate the performance and execution times of quantum algorithms in various contexts, including QPUs and CPU-QPU loops.

Evaluation methods: Assessment of the mini project

CM:

1. Rappels Mathématiques pour l'informatique Quantique (1.5 h)
2. Formalisme le calcul quantique (1/2) (1.5 h)
3. Formalisme pour le calcul quantique (2/2) (1.5 h)
4. Synthèse de circuits et contraintes matérielles (1.5 h)
5. Structure des Algorithmes Quantiques (1/2) (1.5 h)
6. Structure des Algorithmes Quantiques (2/2) (1.5 h)
7. Algorithmes pour les architectures NISQ (QAOA,VQE) (1/2) (1.5 h)
8. Algorithmes pour les architectures NISQ (QAOA,VQE) (2/2) (1.5 h)
9. Algorithmes variationnels et application au machine learning quantique (1/2) (1.5 h)
10. Algorithmes variationnels et application au machine learning quantique (2/2) (1.5 h)

TD:

1. TD de formalisme et d'analyse de circuits quantiques (1.5 h)
2. TD de conception d'algorithmes quantiques sur QPU (1.5 h)
3. TD de conception d'algorithmes variationnels sur CPU+QPU (1.5 h)

TP:

1. TP de mise en oeuvre de circuits quantiques en qiskit sur simulateur et machines quantiques (3.0 h)

2. TP de conception et mise en oeuvre d'algorithmes quantiques à partir de circuits connus (3.0 h)
3. TP de conception d'une méthode d'optimisation par algorithme variationnel sur CPU+QPU (3.0 h)

Programming paradigms

Course supervisor: Cyril Grelier

Total: 28.0 h , 3 ECTS **CM:** 14.0 h, **TP:** 14.0 h

SPM-INF-016 *back*

Description: This course covers different programming paradigms by introducing different languages that are strongly influenced by these paradigms. We will draw on functional languages such as Haskell and Ocaml for functional approaches (the concepts of type inference and lazy evaluation), Prolog for logic programming, and LISP for construction through the calculation of executable expressions...

Learning outcomes: At the end of this course, students will have learned to think about programming in ways other than traditional methods. They will be able to recognize these paradigms when they arise implicitly and apply appropriate solutions (for example, recognizing a functional approach learned in Caml in recursive template designs in C++, recognizing a logical programming paradigm in makefiles, etc.). The idea is also to be able to propose elegant solutions to certain problems without being dependent on the structure of the language used (e.g., implementing a lazy approach in C++).

Evaluation methods: Assessment based on participation in experiments (TP) and results returned

---

CM:

1. Cours-1 (2.0 h)
2. Cours-2 (2.0 h)
3. Cours-3 (2.0 h)
4. Cours-4 (2.0 h)
5. Cours-5 (2.0 h)
6. Cours-6 (2.0 h)
7. Cours-7 (2.0 h)

TP:

1. TP-1 (2.0 h)
2. TP-2 (2.0 h)
3. TP-3 (2.0 h)
4. TP-4 (2.0 h)
5. TP-5 (2.0 h)
6. TP-6 (2.0 h)
7. TP-7 (2.0 h)