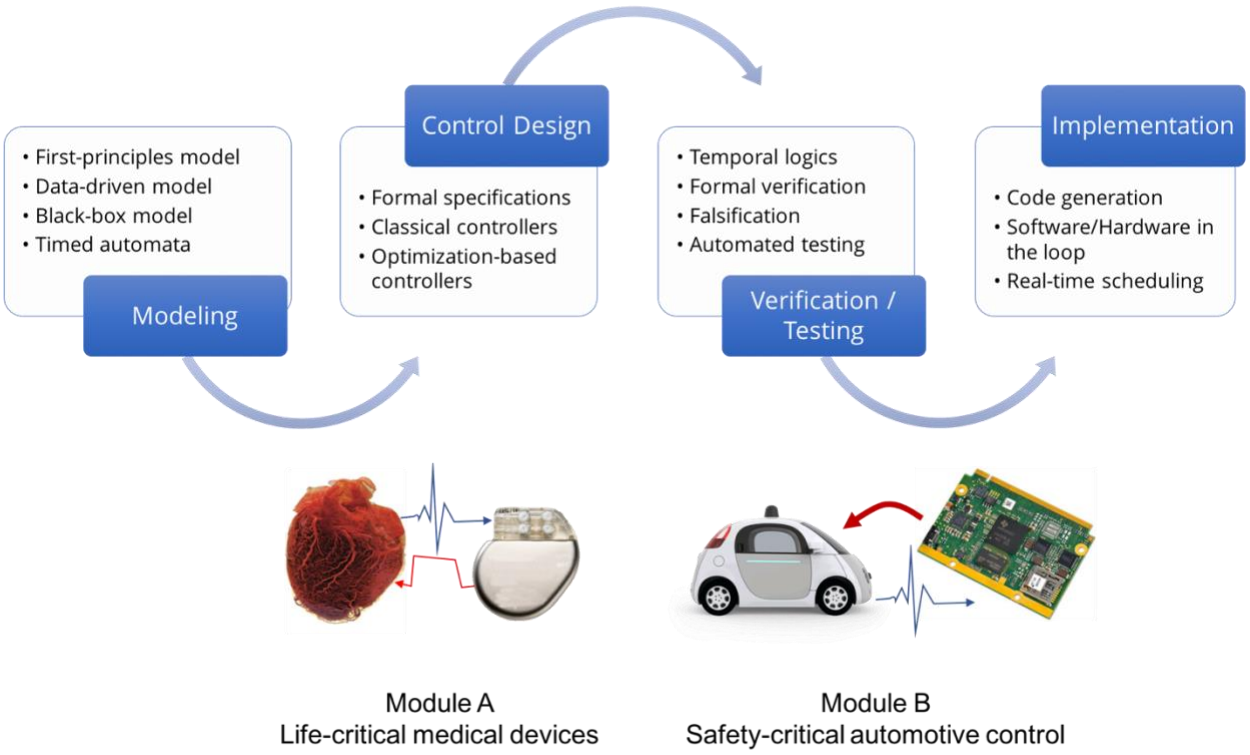
Cyber-Physical Systems

ECE 499/599 and CS 419/519

(Syllabus subject to change until the class starts)



What does it take to design and implement life-critical software in an implantable cardiac defibrillator?

How to certify that a car in autonomous cruise control mode will drive itself safely?

This course will lead you to modeling for verification and testing of such advanced embedded systems, known as *Cyber-Physical Systems*. The course is 50% theory covering the foundations of temporal logic, controls and falsification and 50% practical skill development with the use of industry standard tools in verification, testing and model-based development. In two month-long modules, we will cover exciting applications of cyber-physical systems with in-depth modeling of implantable cardiac medical devices and testing of advanced driver assistance software in automotive controllers. This course provides the foundations and tools for a career focus in model-based design of advanced embedded systems needed in the emerging domains of self-driving cars and autonomous medical devices.

Instructor

Dr. Houssam Abbas Houssam.abbas@oregonstate.edu

Office hours: Tuesdays and Thursdays 1:30-2:30 (after class) in KEC 3101

Teaching Assistant and office hours

TBD

Meeting times

Tuesday – Thursday 12-1:20pm

Course website

We use Canvas to assign worksheets, labs, deliver slides, readings and grades, and to communicate with each other: https://canvas.oregonstate.edu/

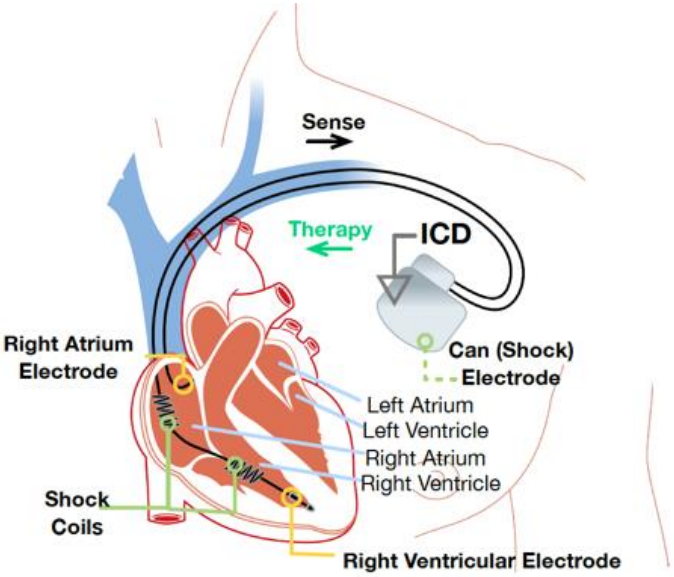
Module A: Modeling for Verification

Application: Life-critical implantable medical devices and their software

Objectives: *Learning the basics of temporal logic and timed automata, appreciating the utility and challenges of formal proofs and model-checking.*
Tools: *UPPAAL model checker, Simulink*
Models: *Timed automata*

Life-saving medical devices, like pacemakers and defibrillators, require a rigorous approach to verifying their safety. Testing, in which the device is fed different inputs and its behavior observed, cannot *guarantee* correctness and freedom from faults. Formal verification, on the other hand, provides such a guarantee.

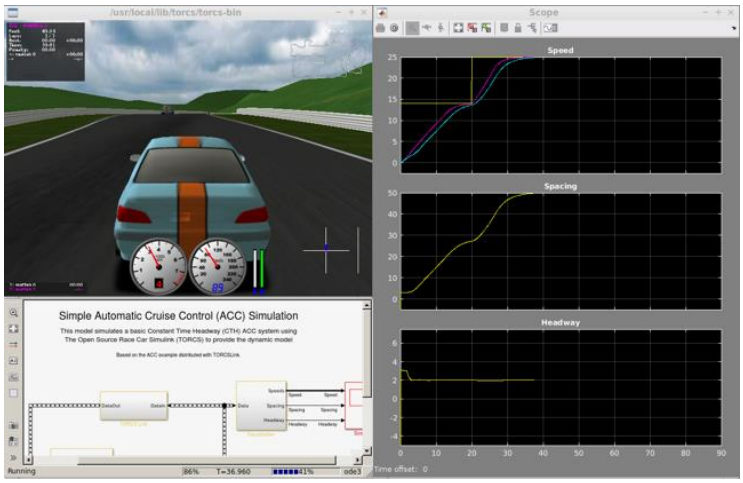
In this project, we will get an overview of the fascinating computer models of the human heart out there, and get a feel for their capabilities. Then we will focus on timed automata, which allow us to formally and rigorously prove freedom from faults. The lessons and perspective you learn in this project will be useful to you in almost any embedded systems project you tackle in the future.



Module B: **Modeling for Testing**

Application: Automotive testing

Objectives: *Learning co-simulation with black-box models, find bugs with automated testing, code generation.*
Tools: *Simulink, Simulink Coder.*
Models: *Simulation black-box model.*



You like to write code but you hate bugs. You're also not a fan of car accidents. In this project, you will learn how to automate the process of bug-hunting to look for what matters: specification violations. Safety and correctness have the highest priority, but are difficult to guarantee with formal verification due to the complexity and hidden nature of black-box models. We will employ an automated testing technique based on formal logics to find bugs in our controller design. Finally, we will generate C code of the controller automatically from Simulink and inspect the effect of real-time properties of the implementation on the control performance.

Course Prerequisites

(Please contact the instructor if you're not sure you have the pre-requisites.)
Introductory Ordinary differential equations.
Notions of automata and finite state machines.
Working knowledge of MATLAB and Simulink – if you've never used Simulink before we will give a one-hour tutorial in class, but then plan additional time to come up to speed.
Mathematical maturity: you are comfortable reading a proof, and have practiced proving minor theoretical statements (e.g., you took a class in Theoretical CS, Abstract Linear Algebra, Graph theory, Discrete Math, Real Analysis, Optimization, etc).

Grading Criteria

In-class participation:	5%
Scribing	10%
Module A – Modeling for Verification: Cardiac implantable devices The grade for this module breaks down as follows: Worksheet and Lab 1: 50% Worksheet and Lab 2: 40% (graduates) or 50% (graduates) In-class presentation for graduates: 10%	45%
Module B – Modeling for Testing: Automotive Systems The grade for this module breaks down as follows: Worksheet and Lab 1: 50% Worksheet and Lab 2: 50%	40%

Schedule

Date	Case study	Topic	Assignment
September 26	Course Overview	Course overview. Syllabus and policies	
October 1	Modeling for Verification	Ad hoc pacemaker design: what could go wrong?	Worsksheet: PM Step by Step reading
October 3 (Last day to withdraw 100% refund is Oct 6)	Modeling for Verification	Principled modeling of the cardiac electrical activity	
October 8	Modeling for Verification	Simulink hands-on	Lab 1: Modeling the heart in Simulink
October 10	Modeling for Verification	Introduction to Linear Temporal Logic.	
October 15	Modeling for Verification	Introduction to Linear Temporal Logic (Cont'd)	Worksheet: LTL problems
October 17 (Last day to withdraw 50% refund is Oct 20)	Modeling for Verification	Linear Temporal Logic (Cont'd)	
October 22	Modeling for Verification	Timed automata.	Lab 2 UPPAAL
October 24	Modeling for Verification	A guided tour of UPPAAL for model-checking timed automata	
October 29	Modeling for Verification	Model-checking timed automata	
October 31	Modeling for Verification	A review of Lab 2	Graduate students assignment
November 5	Modeling for Verification	Graduate student seminar: Statistical model checking in UPPAAL	
November 7	Modeling for Testing	Introduction to automotive control: car dynamics and Adaptive Cruise Control	
November 12	Modeling for Testing	ACC requirements and specification-guided testing	Worksheet 3: Understanding ACC Lab 3: Running ACC
November 14 (Last day to withdraw, no refund)	Modeling for Testing	Quantitative satisfaction of requirements	
November 19	Modeling for Testing	Quantitative satisfaction of requirements	Worksheet 4: finite trace semantics and quantitative satisfaction
November 21	Modeling for Testing	Breach tutorial	Lab 4: Specification-guided testing

November 26	Modeling for Testing	Falsification as optimization	
November 28 – no class	Thanksgiving		
December 3	Modeling for Testing	Lab review	
December 5	Modeling for Testing	Code generation	

Assignments

The assignments are either a graded Worksheet and a graded Lab. The worksheet is a warm-up on the week's material, and is most like a homework where you have to solve pen-and-paper problems, including some proofs. You will occasionally have an oral examination component, where you meet with the TA or the instructors. The Lab is the bigger component of the assignment (time-wise and grade-wise). Unless noted otherwise, the assignments are

Scribes

Every week two students will be assigned the role of scribes. As a scribe your responsibility is to take excellent notes during the lectures, then transcribe these into Latex and produce lecture notes that you and other students can use.
Both scribes should take notes independently during class. Do *not* count on the other person. Then you meet after the lecture to compare notes and consolidate one version. You should do this right after class, while everything is still fresh. It is also a great way for you to determine whether you understood everything. Then prepare an outline for your notes and meet with me during Thursday office hour so we can agree to it. Finally, create the lecture notes. A week's scribe notes are due on the Sunday of that week.
A portion of the grade will be assigned to scribing, and will depend on the quality of the notes and their timeliness.

Computing

<u>If you have your own machine: Install Matlab and UPPAAL</u>

- As an OSU student you should already have access to Matlab – try the latest version. I suggest you download it to your own machine if you have one rather than using the network version which is slower.
- Download and install UPPAAL on your computer from this link: http://www.uppaal.org/ (use the stable version).

<u>Computer lab</u>

We have installed MATLAB and UPPAAL on the Linux machines in the computer labs under /usr/local/apps/uppaal/current

Policies

Collaboration

You are encouraged to work together.

You may discuss the homework with other people to understand the problem and reach a solution. However, each student must write down the solution independently, without referring to written notes from others. I.e., you must understand the solution well enough in order to reconstruct it by yourself. Basically, you know what I'm talking about. In addition, each student must write on their homework the names of the people with whom they collaborated.

Honor code

The purpose of problem sets in this class is to help you think about the material, not just give us the right answers. You are encouraged to use online resources for learning more about the material covered in class; however, you should **not** look for or use found solutions to questions in the problem sets. Specifically, you must not look at any code that has been created to solve the assignment, including solutions found on the internet to questions in the problem sets, code created by a student in a previous class or code created by a current classmate. (Though frankly, I seriously doubt you'll find anything, this is a brand new class). Cheating will be punished according to university regulations as determined by the Office of Student Conduct.

Late Policy

Assignments are, as a general rule, a week after the day they are assigned, at midnight. You are allowed to turn in up to two late assignments in this term. Each of these two times, you can be 2 days late at the most. After that, every late day is penalized by 20% of the full grade. Thus if the assignment is due on Tuesday, and you turn it in on Thursday, you don't lose any points. If you turn it in on Friday (=Tuesday + 2 days of grace + 1 day late), your final grade will be Your Earned Grade - 20. On Saturday, it will be Your Earned Grade - 40. Etc. The third assignment you turn in late is penalized by 20 points each day starting with the first. This applies to labs and worksheets.