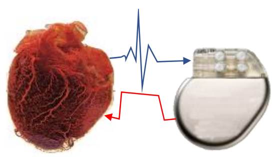
ESE 680: Digital Twins - Model-Based Embedded Systems

- First-principle model
- Data-driven model
- Black-box model
- Timed automata

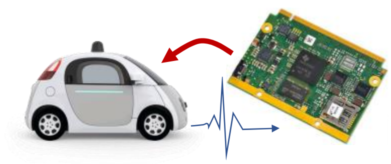
- Formal specifications
- Classical controllers
- Optimization-based controllers

- Temporal logics
- Formal verification
- Falsification
- Automated testing

- Code generation
- Software/Hardware in the loop
- Real-time scheduling



Module A
Life-critical medical devices



Module B
Safety-critical automotive control

Module C
Building modeling and control

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?

How do you develop and tune controls for skyscrapers with complex interactions with the environment, occupants and equipment?

This course will lead you to modeling for verification, testing and control of such safety-critical 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 software and testing of advanced driver assistance software in automotive controllers. The class will conclude with a research project around topics covered in class. This course provides the foundations and tools for a career focus in model-based design of embedded systems.

Instructors

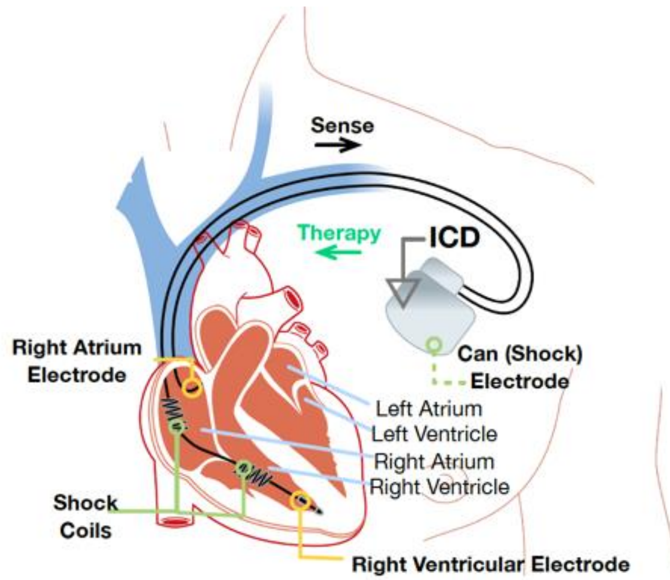
Dr. Houssam Abbas habbas@seas.upenn.edu
Dr. Rahul Mangharam rahulm@seas.upenn.edu

Module A: Modeling for Verification

Application: Life-critical implantable medical devices and their software

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



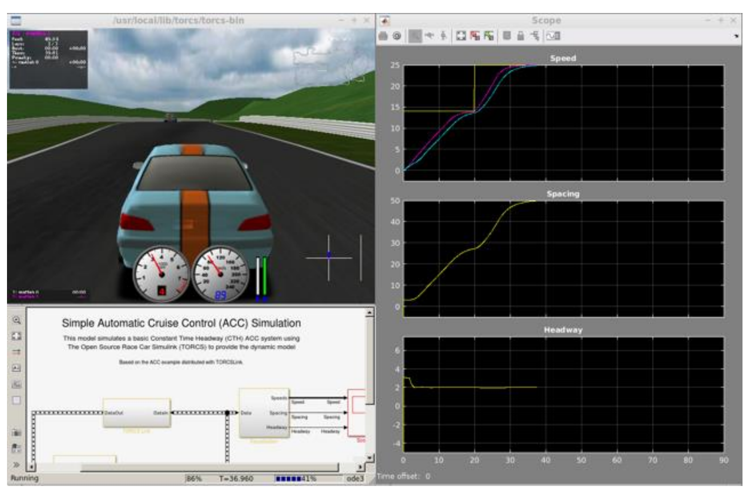
In this project, we will get an overview of the fascinating computer models of the human heart out there, and get a feel for their capabilities. Then we will focus on timed automata, which allow us to formally and rigorously prove freedom from faults. The lessons and perspective you learn in this project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

Module B: **Modeling for Testing**

Application: Modeling for Testing

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

You like to play games, you like to race, and you hate bugs. In this project, you will learn how to connect your favorite modeling and control design tool, Simulink, to your favorite car racing game, TORCS, to automate driving. 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.



Prerequisites

Ordinary differential equations.
Notions of automata and finite state machines.
Working knowledge of MATLAB and Simulink, and general comfort in programming
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, Optimization, etc)

Grading Criteria

In-class participation:	5%
Scribing	5%
Module A – Modeling for Control: Energy-efficient buildings The grade for this module breaks down as follows: Worksheet and Lab 1: 25% Worksheet and Lab 2: 25% Worksheet and Lab 3: 25% Worksheet and Lab 4: 25%	30%
Module B – Modeling for Verification: Cardiac implantable devices The grade for this module breaks down as follows: Lab 1: 25% Worksheet 1: 5% Lab 2: 25% Worksheet 2: 15% In-class presentation: 30%	30%
Research projects Subjects to be decided	30%

Schedule

Date	Case study	Topic	Assignment
January 11	Course Overview	Course overview. Syllabus and policies	
January 16	Modeling for Verification	Ad hoc pacemaker design: what could go wrong?	
January 18	Modeling for Verification	Principled modeling of the cardiac electrical activity	Worsksheet - PM Step by Step

			Lab 1: Modeling the heart in Simulink
January 23	Modeling for Verification	Simulink hands-on	
January 25	Modeling for Verification	Introduction to Linear Temporal Logic.	
January 30	Modeling for Verification	Introduction to Linear Temporal Logic (Cont'd)	Worksheet-LTL problems
February 1	Modeling for Verification	Basics of Model Checking finite-state systems.	
February 6	Modeling for Verification	A guided tour of UPPAAL for model-checking timed automata	Lab 2 UPPAAL
February 8	Modeling for Verification	A review of Lab 2	
February 13	Modeling for Verification	Various extensions of timed automata and their tools (parameter synthesis, SMC and cost-optimal reachability).	Worksheet - Seminar
February 15 Note: Drop period ends Feb 16	Modeling for Verification	Student Seminar	
February 20	Modeling for Verification	Student seminar	
February 22	Modeling for Testing	Introduction to automotive control: car dynamics and Adaptive Cruise Control	Worksheet: Understanding ACC Lab: Running ACC
February 27	Modeling for Testing	Quantitative satisfaction of requirements	
March 1	Modeling for Testing	Quantitative satisfaction of requirements	Worksheet
March 3-11	Spring Break		
March 13	Modeling for Testing	Falsification as optimization	
March 15	Modeling for Testing	S-Taliro tutorial	Worksheet: Alternative optimization methods Lab: Specification-guided testing
March 20	Modeling for Testing	Student seminar: other optimization methods	
March 22	Modeling for Testing	Student seminar: other optimization methods	
March 27	Research projects	Project topics	

March 29 NOTE: Last day to withdraw March 30	Research projects	Medical platforms	
April 3	Research projects	Other heart models	Research plan due
April 5	Research projects	Guided research	
April 10	Research projects	Guided research	
April 12	Research projects	Mid-project report back	
April 17	Research projects	Guided research	
April 19	Research projects	Student seminar	
April 24	Course conclusion	Wrap-up	

Assignments

Most weeks, you will have 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). Because this is a graduate-level class, some assignments will require the students to read papers or perform small research projects and present the results in class later.

Scribes

Every week will have two assigned scribes. As a scribe your responsibility is to take good 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. Then you meet after the lecture to compare notes and have one version. I strongly recommend you do this right after class, while everything is still fresh. It is also a great way for you to determine whether you understood everything.

A portion of the grade will be assigned to scribing, and will depend on the quality of the notes. A week's scribe notes are due on the Sunday of that week.

Course website

We use Canvas to assign worksheets, labs, deliver slides, readings and grades: https://canvas.upenn.edu/

You can use the course's Piazza to answer each other's questions. The instructors may chime in from time to time.

Computing

Virtual Machine

For the Buildings and Automotive case studies (weeks 6 onwards) you will need to use the Virtual Machine (VM). We have installed all the necessary software on the VM and made the appropriate setups, which saves you time.

- Download and install VirtualBox: https://www.virtualbox.org/. Select the right installation file for your OS.
- Link to the Virtual Machine appliance: https://www.dropbox.com/s/5n3getp22ui6gi9/MBES2017.ova?dl=0
- Download the VM appliance file (.ova extension) to your computer. Open VirtualBox. Then choose the menu File/Import Appliance... and follow the instructions to select the .ova file you downloaded and create a virtual machine on your computer.
- Start the Virtual Machine. The default password is "mbes2017" without the quotes.
- To use Matlab, you will need a unique hostname (your ID) for the network license. We will assign your hostnames later once we know who will stay in the course.
- To change the hostname of your VM, open the text file named "How to change hostname" on your VM's desktop and follow the instructions therein.
- Make sure that you successfully change the hostname of your VM to your assigned hostname; otherwise you will cause conflicts when you start Matlab and cannot do your labs.

Install Matlab and UPPAAL on your own computer

For the Medical case study, you don't need the VM. Rather, you can use your own installs of Matlab and UPPAAL. If you have your own machine:

- follow the instructions in this link to download, install, and activate your free Penn student Matlab (including Simulink and Stateflow): https://www.seas.upenn.edu/cets/software/matlab/student/
- Download and install UPPAAL on your computer from this link: http://www.uppaal.org/ (use the stable version).

Computer Lab

The latest Matlab and UPPAAL are also installed in the Linux labs at Penn. Those are Towne M70 and Moore 100A. After you login using your PennID:

- start a terminal: click on the icon in the lower-left corner (similar to the Windows Start button) and type terminal. Hit enter.
- In the terminal, execute the appropriate command
  - >> matlab
  - >> uppaal

Also, Biglab and Speclab are remote Linux lab environments with special usage outlined below:
Biglab: http://www.seas.upenn.edu/cets/answers/biglab.html

Speclab: http://www.seas.upenn.edu/cets/answers/speclab.html

Policies

Collaboration

You are allowed and 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. In addition, each student must write on their homework the names of the people with whom they collaborated.

On each assignment, you should try to work with a different set of people, to maximize learning, but this is optional, not mandatory.

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. Cheating will be punished according to university regulations as determined by the Office of Student Conduct.

If one student shares code with another on a different team, both the donor and the recipient of the code are in violation of the Penn honor code and **will** be referred to the Office of Student Conduct.

Late Policy

Assignments are, as a general rule, assigned on the first class of the week, and are due a week after that, at midnight. You are allowed to turn in up to two late assignments in this semester. 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.