Teaching Statement

Rahul Mangharam
http://www.seas.upenn.edu/~rahulm/teaching/

My teaching goal mirrors my inter-disciplinary approach towards research: I aim to ensure that students cultivate a holistic view of life-critical and safety-critical system development by drawing stronger connections between real-time and embedded systems, control systems, formal methods and hands-on platform development. I design courses that have a significant experimental component, where students work in teams to develop sizable distributed embedded systems that integrate concepts across electrical and computer engineering, computer science, bioengineering and mechanical engineering. These include building test-beds for energy-efficient building heating/cooling systems, electric vehicle powertrain systems, Heart-on-a-Chip medical devices, biofeedback control systems and leader-follower quadrotor aircraft systems[1].

By emphasizing *learning by doing*, I aim to develop students' confidence in deconstructing a design problem, iterating over several inefficient designs, presenting intermediate results, and incorporating creativity in engineering solutions. In following this *design-driven approach*, every student is continually making decisions on how to partition functionality to different sub-systems, how to keep things simple, how to define clean and generic interfaces at hardware/software boundaries and finally how to demonstrate the core ideas within the operational system in an effective way. This emphasis on developing hands-on critical thinking skills forms the bedrock of my teaching approach.

In the rest of this statement, I will summarize the Autonomous Systems, Embedded Systems and Model-based Embedded Systems courses I have developed at Penn and discuss plans for a broader curriculum on Model-based Design of Systems. I will describe my efforts in developing cross-school collaborations within Penn, student and faculty based teaching events, and establishing Penn as a recognized force across global engineering communities.

A. Developing Embedded Systems Programs at Penn

Since 2008, I have been part of the founding committee for establishing the embedded systems curriculum at Penn through the development of the Computer Engineering (CMPE) undergraduate program and, separately, the Masters in Embedded Systems (EMBS) program (1st such program in the nation). Since 2015, I have served as the EMBS Program Director. Both programs have been tremendously popular, with high quality students. Embedded Systems graduates, who were under my supervision, during and after taking my courses, have gone on to pursue graduate studies in Stanford (Kevin Conley and Jeff Kiske), Georgia Tech (Matt Hale), Duke (Ashleigh Thomas), UCLA (Paul Martin), CMU (Utsav Drolia), MIT(Max Li), among others.

Prior to 2008, very few technology companies recruited from the ESE Department and that has changed tremendously with both programs. Students joined SpaceX, Tesla Motors, NASA (Langley), GoogleX Labs, BOSCH Research, Intel, Nvidia, Qualcomm, etc. Students graduating from the Autonomous Systems course were very successful in getting top placements in Tesla AutoPilot group, Honda's and Nvidia's Autonomous Vehicles group, Rivian's electric drivetrain group, Aurora's and Zoox's motion planning and control team, nuro's software infrastructure team, among others. This is a testament that the students are nationally competent with a broad and deep skillset of model-based and data-driven system design.

Through the 60+ undergraduates and 80+ Masters students that I have advised, I have come to appreciate the need to balance theory and skill development. As we refined the curriculum, our goal was to ensure students not only get solid foundations in becoming "Platform Architects" across the computing stack, but also became skilled in "Systems Thinking" for debugging and developing complex systems with perception, planning and control pipelines.

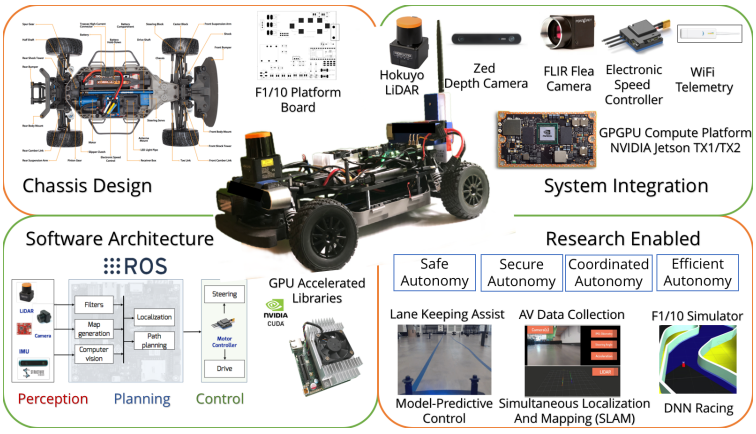
B. Autonomous Systems Curriculum Development

In order to facilitate research and education in autonomous systems, I developed an open-source research platform of high-performance autonomous racing cars that are 1/10th-scale of Formula-1 cars and can reach a top speed of 50mph. F1/10 [http://f1tenth.org/] enables a wide range of machine learning engineering with perception, planning, control and coordination modules. In addition to the platform hardware, my

[1] For a really exciting compilation of ESE350 project videos, see http://tinyurl.com/ese350videos

teaching team developed an AV software stack and a set of simulators as plug-and-play replacements for the 1/10th-scale platform itself. The course covers five modules and include three races for evaluation:

1. *Introduction to ROS, F1/10 & the Simulator:* Introduction to self-driving hardware and full software stack, automatic emergency braking, LiDAR, rigid body transformations, Laplace domain dynamics, and PID control for wall following.
2. *Driving using Reactive Methods & RACE!:* Build a vehicle, tune the electronic speed controller, and implement reactive driving methods such as 'follow the gap' and complete Race 1.
3. *AV Mapping & Localization:* Understand the key elements of SLAM by implementing scan matching and particle filters; use Google Cartographer SLAM; implement pure pursuit driving; complete Race 2 using maps.
4. *AV Planning:* Learn Moral Decision Making for autonomous systems; implement planning with rapidly exploring random trees (RRT), and understanding model-predictive control (MPC) for raceline optimization.
5. *Learning & Vision:* Design and implement algorithms for object detection and pose estimation, reinforcement learning and visual feature extraction.
6. *F1/10 Grand Prix:* Race 3 will include a project to implement racing strategies which combine fast perception, agile planning and aggressive control for race optimization.



In addition to teaching the course at Penn, we have organized 5 international autonomous racing competitions for multi-vehicle racing. Three more races have been scheduled for 2020 at CPSweek (Sydney in April), IFAC World Congress (Berlin in July) and Embedded Systems Week (Shanghai in October). The course had been open-sourced and now is being taught at several universities including: Oregon State University, UVA, N. Arizona University, Clemson University, UT Austin, TU Vienna, etc. F1/10 has over 59+ university community partners for research and education. It is supported by a $1.5MM NSF CISE Community Research Infrastructure award and industrial partners such as National Instruments, Nvidia, Intel, etc. Driving at the limits of vehicle performance helps us accelerate the development of safe autonomous vehicles.

C. ESE350 Introduction to Embedded Systems

This course introduces the use of microcontrollers, sensors and actuators in building real-time embedded systems that interact with the physical world. With an approach focused heavily on learning by doing, the labs are designed to be interactive, fun, yet challenging. The students build:

- *Arduino from Scratch* – in the first 2 weeks students build the entire Arduino on a breadboard and are introduced to power management, bootloaders, memory layouts, and circuits with over 36 components
- *Build a musical instrument with Sensors and actuators* - to learn timers, analog-to-digital conversion interrupts, analog circuits
- *Interactive Hand-held Pong Game System* – learn to interface touchscreens, LCDs and motion sensors to make a fun and interactive game system
- *Balance Bot* - develop a self-balancing bot using stepper motors, motor drivers, an IMU and interface it with a Zigbee wireless remote. Implement controls for stability and position control and race!

The lab exercises begin with well-defined instructions where the students initially program the 16-bit microcontrollers hardware in C at the bare metal. The labs progressively get more open-ended and the students' progress to use programming libraries and eventually work with embedded operating system (using 32-bit ARM multiprocessors). This ensures the basics of hardware-software interfaces are learned, and allow the students to gradually abstract away the low-level details for more *system-wide thinking*.

The final 4 weeks feature a project where students are asked to build creatively on the foundations they have learned by designing and developing an embedded system of their own. Examples of such systems include tele-operated leader-follower quadrotors, body sensor games based on hacked Nintendo/Sega console games, electro-mechanical chess, connect-4 and basketball shootout machines, spherical robots, etc. Projects from ESE350 have won over *16 national and international awards*.

(b) ESE519 Real-Time and Embedded Systems

This core graduate course covers the concepts, theory, and tools necessary to understand, design, and build real-time and concurrent embedded systems. The course is spread across five major modules starting with an introduction to networked embedded systems, real-time scheduling theory, concurrent programming and distributed systems theory, life-critical systems and case studies in embedded system failures. Five labs introduce programming with a real-time operating system (RTOS); networked operation across distributed embedded systems; development of routing protocols; and design of safety-critical systems. Each group is given a set of wireless embedded nodes, 32-bit embedded multi-processor boards (700MHz and 1GHz) and a variety of sensors and actuators. Final projects have included:

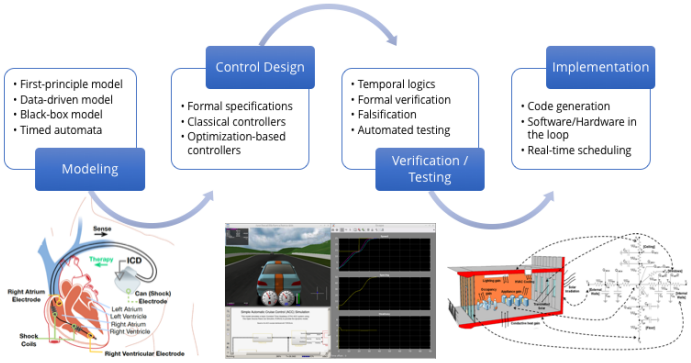
- Energy-efficient building automation systems, which integrate sensing, distributed controls and building automation algorithms for heating and cooling in HVAC systems.
- Automotive embedded systems that integrate control systems for traction, stability, anti-lock braking, and adaptive cruise control with a hardware-in-loop test-bed.
- Robot-soccer with multi-robot consensus protocols for defense/offense, using computer vision
- Implementation of new wireless protocols (ISA100.11a) for industrial control and automation

Student projects from ESE519 have resulted in over 12 international conference publications and demos in venues such as ACM/IEEE Cyber-Physical Systems Week, ACM Embedded Systems Week, ACM Building Systems Symposium and the IEEE Real-Time Systems Symposium (RTSS).

D. Model-based Systems Curriculum Development

I routinely experiment with new course concepts. To translate our research into teaching, I developed a course on foundations for Model-based Design for Cyber-Physical Systems. This course allows the student to journey from specifications to models to implementations to integration for rigorous system design of controls, computation and communication to answer the following questions:

- *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 that have complex interactions with the environment, occupants and equipment?*



This course focuses on 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 (Matlab/Simulink, EnergyPlus, UPPAAL). In three month-long modules, we cover in-depth modeling of implantable cardiac medical device software and systems, testing of advanced driver assistance software in automotive controllers and data-driven modeling and control of buildings. This course provides the foundations and tools for a career focus in model-based design of embedded systems.