EE 152 / BME xx / ME xx / CS xx
Real-time embedded systems
Fall 202x

Class meeting time: ??
Instructor: Joel Grodstein. Office hours: ??
Teaching assistant: ??

What is a real-time embedded system, anyway?

Like many terms, this one does not have a precise definition. But here's what we meant when we named this course, at least 😊.

We use the term *embedded system* as a system whose customers do not think of it as a computer – but that still requires substantial amounts of computing for its functionality. Your laptop has lots of computing power, but is not an embedded system since most people consider it to be a computer.

Many refrigerators are embedded systems. While nobody thinks of their refrigerator as a computer, it still has computation to, e.g., keep a temperature setpoint and even to use RFID tags to tell you when it's time to order more milk.

However, a refrigerator is not a *real-time* embedded system. None of its responses have a very stringent time requirement. On the other hand, the anti-lock brakes in a car have a very stringent time requirement! So does a pacemaker. This course looks mainly at these quick-response devices, focusing on the hardware and software needed to meet those requirements.

***Course Objectives*:**

Upon completion of the course, students will be familiar with:

1. The basics of embedded systems; hardware and software
2. The basics of real-time software; how do you write software that has to respond in a specific amount of time.
3. The pros and cons of various embedded-device environments (Arduino, Raspberry Pi, bare metal, FreeRTOS).
4. At least one specific real-time embedded system, such as one to process ECG waveforms or EMG waveforms.

Learning outcomes

- Students will learn how to use a real-time operating system.
- Students will learn multiple APIs for embedded systems programming, and the pros and cons of each.
- Students will learn debugging techniques suitable for real-time embedded systems.

What you will learn, in more detail

How can you build systems that must perform under a time constraint? E.g., "my ABS system must respond to a skid within 20ms" or "my signal-processing system processes samples exactly every 2ms?" We'll discuss several ways to deal with this problem: e.g., spin-wait loops (not a great answer), the use of a *real-time operating system* (RTOS), and the use of hardware timers.

How do you program peripherals when time matters? In the hobbyist world, we often use Arduino or Raspberry Pi, and accessing peripherals is easy because the libraries do all of the heavy lifting. In the industry, shippable products rarely use these hobbyist libraries, but instead use a Byzantine collection of vendor infrastructures – almost every microcontroller company has (at least) one proprietary API! We'll learn why these infrastructure wars exist, and how to pick a path with a reasonably low effort for the performance level your product needs. In the labs, we'll use a popular industrial API to implement parts of the Arduino API in an efficient manner.

How do you write embedded software when your application consists of many independent tasks, each with their own (potentially different) inputs, outputs and timing rhythms? We'll build lots of examples with FreeRTOS (one of the most popular RTOSs) and learn by doing.

How do you debug an embedded system when it is processing non-repeatable embedded data streams that fly by at hundreds of samples per second? Many real-time embedded systems combine many elements that make traditional programs difficult to debug, and furthermore do so in a small processor that doesn't have a keyboard or a monitor! We'll learn techniques for dealing with this by debugging a real-time heart-rate monitor.

Textbook and other resources

There is no textbook. There's plenty of material on the web about programming STM32 processors. Here's some:

- https://vivonomicon.com/category/stm32_baremetal_examples starts from the ground up, without using PlatformIO or VSCode. It eventually gets to some higher-level tutorials (I like the one on DMA).

Lab work

- Build small real-time systems to prep us for the real thing.
- Build small systems with FreeRTOS.
- Build a basic ECG monitoring system.
- Potentially work with sEMG signals also.

Workload

- We will have roughly 10 labs. Each group should hand in only one official lab report. The labs build on each other, and the final few labs create a functioning ECG-based heart-rate detector.

Grading

The course grade will be computed roughly as follows:

- labs = roughly 90% of the total grade
- class participation = roughly 10%
- potentially a quiz or two as well.

Late Assignments:

Late assignments will be penalized by 10% per day. Any extensions due to extenuating circumstances (illness or family emergencies) must be arranged ahead of time with the instructor before the original due date.

Prerequisites

- This is an intentionally interdisciplinary course, covering parts of biology, computing, electrical engineering and programming. It is expected that very few people will enter the course being

competent in all of these areas, and we thus are not being strict about prerequisites. Obviously, the more areas you must learn, the more work you will have. With that in mind…

- The labs will expose you to a fair amount of programming, especially real-time programming. We will use C, since it is by far the most common language used in the industry. Our code will not be complex; CS 11 or the equivalent is more than enough, and really all you need is the basics of C. On the other hand, neither CS 11 nor most CS courses at Tufts really prep you for multi-threaded real-time programs, which is what many real-time devices are. So you will hopefully learn a lot 😊.
- We will work with digital signal processing. Some background on working in the frequency domain (e.g., EE 20, BME 10 or ME 30) would be useful.

ABET

This course may help fulfill ABET objectives as follows:

1. *an ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics*. Our labs do this. Especially, the final labs use a combination of software, hardware, DSP (which involves math) and algorithms.
2. *an ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives*. Essentially all of your work in this course will be as part of a roughly-three-person team.
3. *an ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions*. You'll be doing lots of debugging for this course – which will require you to do all the above.

Additional resources

Tufts University values the diversity of our students, staff, and faculty, and recognizes the important contribution each student makes to our unique community. Tufts is committed to providing equal access and support to all qualified students through the provision of reasonable accommodations so that each student may fully participate in the Tufts experience. If you have a disability that requires reasonable accommodations, please contact the Student Accessibility Services office at Accessibility@tufts.edu or 617-627-4539 to make an appointment with an SAS representative to determine appropriate accommodations. Please be aware that accommodations cannot be enacted retroactively, making timeliness a critical aspect for their provision. Tufts and the teaching staff strive to create a learning environment that is welcoming to students of all backgrounds. If you feel unwelcome for any reason, please let us know so we can work to make things better. You can let us know by talking to anyone on the teaching staff. If you feel uncomfortable talking to members of the teaching staff, consider reaching out to your academic advisor, the department chair, or your dean.