Class Syllabus

Instructors:
Dr. Mark Palmeri, M.D., Ph.D.
(mark.palmeri@duke.edu)
Office Hours: Friday 11:00-12:00 (258 Hudson Hall Annex)

Suyash Kumar, CTO kelaHealth, Former Uber Software Engineer
(suyash.kumar@duke.edu)
Office Hours: Thursday after lecture (125 Hudson Hall)

Teaching Assistant: Arjun Desai
(arjun.desai@duke.edu)
Office Hours: 11:30-13:00 (First Floor Teer)

Lecture: Tues/Thurs 15:05–16:20, 125 Hudson Hall

Course Overview

Software plays a critical role in almost all medical devices, spanning device control, feedback and algorithmic processing. This course focuses on software design skills that are ubiquitous in the medical device industry, including software version control, unit testing, fault tolerance, continuous integration testing and documentation. Experience will be gained in both dynamically- (Python) and statically-typed (C/C++) languages.

The course will be structured around a project to build an Internet-connected medical device that measures and processes a biosignal, sends it to a web server, and makes those data accessible to a web client / mobile application. This project will be broken into several smaller projects to develop software design fundamentals. All project-related work will be done in groups of 3 students.

Prerequisites: Introductory Programming Class (e.g., EGR103)

Course Objectives

- Define software specifications and constraints
- Agile project management
  - Team roles
  - User stories
  - Backlog, To Do, In Progress, Completed, Blocked "Issues" (Kanban)
  - Weekly sprints
  - MVPs
- Device programming fundamentals
  - Review of data types
  - Analog-to-digital / digital-to-analog conversion
  - Python (v3.6): numpy, scipy, pandas, scikit
  - C/C++
  - Simplified Wrapper and Interface Generator (SWIG)
  - Data management (variables, references, pointers, ASCII/Unicode/binary data)
  - Regular expressions (regex)
  - Compilation, make, cmake
  - Use of a programming IDE
  - Debugging
- Backend Software Development in the Cloud
  - HIPPA Compliance in the cloud
  - Databases
  - HTTP & RESTful APIs

  - Leverage scalable compute infrastructure in the cloud via Remote Procedure Calls (RPCs)
  - Call web services from Matlab & Python
  - Design & Implementation of a biomedical web service (Python Flask)
  - Docker and dependency management intro
  - SSL and Encryption
  - Internet of Things (IoT) and cloud connected biomedical device design
  - Load Balancing and throughput bottlenecks

TBD Sockets and streaming data over networks

- Software version control (git, GitHub)
- Documentation
  - Docstrings
  - Markdown
  - Sphinx / Doxygen
- Testing
  - Unit testing
  - Functional / System testing
  - Continuous integration (Travis CI)
- Fault tolerance (raising exceptions)
- Resource profiling (cProfile)

Fall Semester Class Schedule

The course schedule is subject to change depending on progress throughout the semester. Specific lecture details, along with deliverable due dates, will be updated on Sakai and this syllabus (which is hosted in a GitHub repository). New due dates will be announced in lecture and by Sakai announcements that will be emailed to the class. The following is a summary of activities this semester:

Date	**Lecture**	**Assignment**
Tues Aug 29	Class Introduction, Objectives and Logistics; Git Demo	A01: Git(Hub), Python
Thurs Aug 31	Git: Repo Setup, Issues, Branching, Pushing/Pulling	In-class exercise
Tues Sep 05	Python virtualenv (conda/pip), Unit Testing	In-class exercise
Thurs Sep 07	Continuous Integration (Travis CI), Python Types/Exceptions	Comlete unit test exercise
Tues Sep 12	Python: Dictionaries, Numpy Arrays, Binary Data	A02: Heart Rate Monitor
Thurs Sep 14	Python: Project Structure (approaching HRM)	
Tues Sep 19	Data Types, Application Program Interfaces (API) & JSON	
Thurs Sep 21	Modules, Classes, Composition Classes: Inheritance & Composition	
Tues Sep 26	Python: docstrings, Sphinx	A03: TBD
Thurs Sep 28	Python: try/except, logging	
Tues Oct 03	Python: Read/Writing Data (CSV, JSON, HDF5, MATv5)	A04: robust, logging
Thurs Oct 05	Regular Expressions	
Tues Oct 10	FALL BREAK	
Thurs Oct 12	Installing SWIG & C/C++ Compiler	
Tues Oct 17	Python $\rightarrow$ C/C++ & Static-Compilation	
Thurs Oct 19	Binary Data Files & Bit Operations	
Tues Oct 24	Continuous Integration Testing	
Thurs Oct 26	Regular Expressions	
Tues Oct 31	TBD	A05: TBD
Thurs Nov 02	Simplified Wrapper & Interface Generator (SWIG)	
Tues Nov 07	TBD	A06: TBD
Thurs Nov 09	TBD	
Tues Nov 14	IEC 62304	Project Unit Tests

Thurs Nov 16	Project TDD	
Tues Nov 28 Thurs Nov 30	Debugging Presentations: Profiled Code	Working Project Code
Tues Dec 05 Thurs Dec 07	Project "Lab" Final Project Due	Refactor Project Code

Attendance

Lecture attendance and participation is important because you will be working in small groups most of the semester. Participation in these in-class activities will count for 15% of your class grade. It is very understandable that students will have to miss class for job interviews, personal reasons, illness, etc. Absences will be considered *excused* if they are communicated to Dr. Palmeri and Mr. Kumar at least 48 hours in advance (subject to instructor discretion as an excused absence) or, for illness, through submission of Short Term Illness Form (STIF) **before** class. Unexcused absences will count against the participation component of your class grade.

Textbooks & References

There are no required textbooks for this class. A variety of online resources will be referenced throughout the semester. A great resource for an overview of Python:

`https://github.com/jakevdp/WhirlwindTourOfPython`

Distributed Version Control Software (git)

Software management is a ubiquitous tool in any engineering project, and this task becomes increasingly difficult during group development. Version control software has many benefits and uses in software development, including preservation of versions during the development process, the ability for multiple contributors and reviewers on a project, the ability to tag "releases" of code, and the ability to branch code into different functional branches. We will be using GitHub (`https://github.com`) to centrally host our git repositories. Specifically, we will be creating student teams in the Duke BME Design GitHub group (`https://github.com/Duke-BME-Design`). Some guidelines for using your git repositories:

- *All* software additions, modifications, bug-fixes, etc. need to be done in your repository.
- The "Issues" feature of your repository should be used as a "to do" list of software-related items, including feature enhancements, and bugs that are discovered.
- There are several repository management models that we will review in class, including branch-development models that need to be used throughout the semester.
- Instructors and teaching assistants will only review code that is committed to your repository (no emailed code!).
- All of the commits associated with your repository are logged with your name and a timestamp, and these cannot be modified. Use descriptive commit messages so that your group members, instructors, and teaching assistants can figure out what you have done!! You should not need to email group members when you have performed a commit; your commit message(s) should speak for themselves.
- Code milestones should be properly tagged.

- Write software testing routines early in the development process so that anyone in your group or an outsider reviewing your code can be convinced that it is working as intended.
- Modular, modular, modular.
- Document!
- Make commits small and logical; do them often!

We will review working with git repositories in lecture, and feedback on your software repository will be provided on a regular basis.

Project Details

Project details will be discussed in lecture throughout the semester.

Grading

The following grading scheme is subject to change as the semester progresses:

Participation	15%
Midterm project deliverables	55%
Final project	30%

Duke Community Standard & Academic Honor

Engineering is inherently a collaborative field, and in this class, you are encouraged to work collaboratively on your projects. The work that you submit must be the product of your and your group's effort and understanding. All resources developed by another person or company, and used in your project, must be properly recognized.

All students are expected to adhere to all principles of the Duke Community Standard. Violations of the Duke Community Standard will be referred immediately to the Office of Student Conduct. Please do not hesitate to talk with your instructors about any situations involving academic honor, especially if it is ambiguous what should be done.