Course Project Guidelines

Projects will be performed in teams of three to four people. And all projects will be evaluated on the basis of the quality of the design plan, documentation, and execution with particular attention on the methodology used. It is neither necessary nor sufficient to have a "cool demo" to get a good grade.

Good projects usually have:

- A good technical plan with realistic timelines.
- Effective use of models. Models could cover physical and/or cyber components, dynamic behavior of the system, safety and reliability, and timing.
- High quality software architecture: Modular, well-defined components with well-defined interfaces. Code should be well documented, with procedures clearly labeled with their functionality, with the names of variables and procedures carefully chosen to reflect their role, etc. Modularity implies, for example, that a software component interfacing a sensor is distinct and separable from another software component controlling an actuator.
- Good choices of hardware, considering cost, capability, reliability, and availability (watch out for long lead times on ordering parts!). Again, modular designs with clean architectures are better. Each component should have a clearly defined role and a well-chosen interface, and should be of the right scale and capability for the task at hand.
- Attribution. At every opportunity, when presenting your project or writing about it, you must credit your sources of inspiration and technology. Who wrote the sample code that you downloaded and modified? What is its copyright? What YouTube video inspired you? Who created the web page describing the project that you are emulating? Everything we do as engineers depends on what other people have done before us. It is extremely important to give credit to those people.
- Effective use of platform-based design and model-based methodologies you have learned in the class. Define clearly how you will use models for design, validation, analysis, and testing, and what software tools you used for modeling, simulation, and software development.

Deliverables:

3. Project Mini Updates: In lab sessions, week of November 17.
5. Project Presentations: December 17. Each team will have 10 minutes for a presentation and demo.
8. **Peer Evaluation Form.** Due December 19.

**Notes:**
1. Always identify you and your team by name
2. Always credit the work of others that you are building on (youtube videos, open-source software, etc.).
3. Giving demos early in a presentation, or even right at the beginning, is better than at the end.

**Reports**
- One report per team. Hard deadline: December 19, 11:59am PST. Submit on bCourses.
- Recommended length: four pages, two columns, 10 point font.
- Illustrate algorithms and models with diagrams, include pictures of your hardware, screenshots, etc.
- *Keep the report short and precise; avoid lengthy and verbose descriptions!*
- Project reports should articulate how the project involves two or more of the key concepts in the course:
  - concurrency,
  - modeling of physical dynamics,
  - reliable real-time behavior,
  - modal behavior governed by FSMs coupled with formal analysis,
  - real-time networks,
  - simulation strategies, and
  - design methodologies for embedded systems design.

**Suggested Project Topics**
Each team should self organize. You may choose to work on either research-oriented projects or embedded system application projects. If you want to pursue a project outside of the list, please see the instructors.

**Embedded System Applications:**
Most embedded system platforms consist of four basic components: sensors, computing units, communications, and actuators. Creative and interesting projects usually employ all of these components and solve a real-world problem. In this project, you should define your own application and IMPLEMENT it.

Examples of innovative applications follow:
- **H2O_IQ** uses a moisture sensor for monitoring plants, a Raspberry Pi as the computing unit, an XBee radio to communicate, and a servo actuator to control irrigation.
- **LaundryQb** monitors the status of your laundry and sends a text message when it is ready; this eliminates useless waiting periods staring at a washing machine.
- **SmartThings** offers a set of sensors and actuators so that people can easily build their own automated smart home.
- A recent Kickstarter project **blink(1)** connects the Cloud with an LED for notification, offering “glanceable notice”.

We recommend to use in your application individually addressable LED lights such as the NeoPixel, as they offer opportunities for interactive, reactive, decorative, wearable, or other creative displays. Examples of projects based on the LED lights are: an advanced version of **blink(1)** with sensor-driven interactive lights, for example as holiday decorations, scientific displays (e.g. illustrating wave motion), assistance for the hearing impaired, fire and safety. For inspiration, see this [YouTube video](https://www.youtube.com). Interesting variants might use wirelessly networked sensors (e.g. on an 802.15.4 networks, perhaps leveraging the **SmartThings** platform), for example to detect doors opening, motion, temperature, etc.

**Research Topics:**

1. **Real-Time Unit (RTU):** Many I/O functions in embedded systems (such as driving the NeoPixel lights in the third project category below) require extremely precise timing. Software environments such as Linux, which make programming and debugging much more convenient, cannot typically achieve such precise timing reliably. In the future, computer architectures may combine a real-time unit (RTU) with a general-purpose microprocessor in a manner similar to the way a graphics processing unit (GPU) is used today. At Berkeley, we have developed a prototype RTU based on the open-source Berkeley RISC-V architecture and can be realized as a soft core on the myRIO platform provides such a platform. This project, which will require an adventurous, research-oriented team with an interest in hardware designs, will develop this platform and use it to drive test cases such as the most demanding NeoPixel displays. A graduate student mentor is available to guide projects in this category.

2. Recently, a plethora of small, open-source embedded computing platforms such as Rasberry PI and Arduino have emerged, competing with more proprietary designs. A rich ecosystem of applications and add-on hardware have emerged that enable hobbyists and serious designers to quickly prototype designs, and even to design and deploy commercial products. The software development environments for such processors are fairly conventional IDEs and C programming environments, with libraries provided for peripheral devices. The purpose of this project to design and construct a model-based design environment for software for such devices. Instead of constructing C programs, application designers will build models representing state machines, dataflow diagrams, and discrete-event systems, that describe the application, and a code generator will produce the embedded C code.
Hardware
Your project can use the hardware in 204 Cory. You can also leverage the partnership with Citris Invention Lab, which can make the Invention Lab Inventory available to the projects for more components. Each team will be given a $200 budget for hardware (this includes the parts you get from the lab and Invention Lab). If you have a compelling need that exceeds this budget, please see the instructors. Your lab GSI is your primary point of contact for obtaining hardware.

Sensors:
Sensors that are readily available include Accelerometers, Gyroscopes, Photocells, Flex sensors, Force Sensitive sensors, Temperature sensors, Switches, Gas sensors, Infrared proximity sensors, moisture sensors, and microphones. Interfaces, timing characteristics and models will be provided.

Computation platforms:
mbed
Beaglebone Black
Arduino Mega 2560 Rev3

Communication:
ZigBee
Bluetooth
BLE
Wifi (CC3000)

Actuators:
NeoPixel LEDs
Step Motors
Solenoids
Electric Switches (Relays)
Buzzers/Speakers
iRobot Create

Some Useful Resources:
mbed in a nutshell
mbed C++ code