Project Specifications

ECE 180D: Systems Design Laboratory

October 11, 2019

Contents

1 Introduction

As we have discussed extensively this year, we wanted to have an additional project during the first quarter of the design lab. You will be tasked to create a design for a large-scale system that allows for the display of large letters across graduation caps during commencement. In doing this project, we hope that you will be gaining the skills and experience to think about the development of your own clothing-based project that incorporates similar skills.

In this document, we will be helping you list some of the big points that you should consider in creating your design, as well as detailing some more information in the upcoming midterm presentation, where you will be presenting a proposed set of designs for the implementation of the project.

While it will be difficult to say if all implementations will work just from a presentation, but we want you to really consider the trade-offs in making system decisions and choose what you think is the best way to create this system. You can view this as a friendly "design competition" where the goal is for the most robust and most practical design.

2 Project

The project is the creation of a large-scale display system viewable by the audience from the crowd, where based on some method, students will be able to display their positive graduation messages to the audience. While this is a vague description as to the objective of the project, we will begin to discuss the details further in the document.

There are some assumptions that can be made about the project.

- All students sitting in commencement will orient themselves in a rectangular formation. The students may place their caps in one of many orientations (angled, flat, etc.) unless otherwise instructed to.
- As for the number of students, we can assume that there could be any number from 30 to 80 students participating. While you can potentially make some assumptions as to how many rows or columns are necessary for our display, do note that the number of students should be considered variable and seating is not fixed before the date (unless specially planned out).
- There could be spotty connectivity in such a large and populated environment, so robustness comes into play.
- It will definitely be noisy (volume-wise). Speech processing may be difficult to impossible in such an environment.
- Lights will be fairly bright in the room, making LEDs slightly less noticeable. Furthermore, remember that the audience will be sufficiently far away so that subtle differences will likely be unnoticed. Redundancy is good (but increases complexity and cost).

After making this assumption, we want to design some standardized device across the group of students seated at commencement that will be able to synchronize and display messages to the audience.

Your job is to create an optimal design that achieves this goal. Not every design will be practical, but with the combination of over 10 designs, we hope that some ideas will appear to be plausible.

2.1 Design Problems

There are some design problems that you will have to consider when making your design.

- (**Design of an individual physical device**) What materials will be used so that the display looks nice (how many LEDs)? How will one implement the circuitry onto their clothing? How much will it cost?
- (**Design of the system**) How will each device know where it is? How will messages look nicest on the display (flashes, scrolling left / up)? How will the display scale based on the number of people attending?
- (**The communication between devices**) How will devices be synchronized? How will each device communicate to neighboring devices (and in that manner, how will devices know which other devices are neighbors)? In what manner will messages be initiated? What is the message reliability?
- (**Power / Processing limitations**) How long can the typical device last on our current battery power? What needs to be on at all times? How long does the RPi need to do certain computations?
- (**Robustness**) What will you use as fail-safes in case something does not work? Is there any way to correct for individual mistakes through machine intelligence (height, angle, orientation)?
- (**Miscellaneous**) Will your design require major changes to "traditions" (e.g. will it remove the ability for tassel interaction during the graduation ceremony)?

2.2 Subtopic Considerations

In general, these are the important components that you can consider in these large-scale systems. These are the some of the major requirements used in past projects, and are still highly applicable to our project this year.

1. Localization. This can refer to either the determination of the position of each device within the larger formation or it could also mean the determination of the position of the graduation cap (incl. angle)
2. Speech Processing. We will discuss some resources next week for speech processing, but this can be used as a means to control your device without additional devices. However, it has been known to be relatively finnicky.
3. Gesture Recognition. This will be based primarily on the IMU. This can be implemented as nods or taps, for example.
4. Self-learning. We mean the learning of the device as to how the person is using the device, so as to either self-correct for problems or notify the users that they should change the way they are using it.

We will have at least discussed the basics of how to implement most of these topics in lab, but you will need to go further with what you have learned to properly implement your projects.

2.3 A Basic Sample Design

Since the task ahead is quite tough, we present a sample design that demonstrates some solutions to the design problems. Extensive experimentation has not yet been done, so it may not work in practice, but it seems like a plausible start.

- In terms of the basic design, each cap would be decorated with 4 RGB LEDs, one on each corner of the cap. If brightness is a concern, then additional LEDs may need to be added. The idea behind this decision is that from the audience a cap will appear to be similar to a single pixel. As such, too many additional LEDs may be additional drain on cost and power resources.
- The communication will be done with a separate token device. The token device will act as the communication server / master that will be doing the bulk of communication scheduling. The goal of this device is for it to passed down throughout the people participating, so that it can determine the seating arrangements and orientation.
- A gesture may be used to indicate that a person has received the token, which can turn on a special communication link between the token device and a personal display device. This communication link will give the person his or her position (row/column) in the formation.
- Then, using messaging, requests can be sent to the token, which will elect to choose a message to send to all caps. Given its positioning and the knowledge of a message (and the time that the message request is received), each device will operate the message to its completion.
- Power and processing limitation is unknown. However, given we are not using the camera or anything complex, we expect this design to be one of the more power efficient designs.
- Would require some level of connection to work well. As such, would also need to test the communication robustness.

2.4 Baseline Requirements

Of course, for this project, there will be some baseline requirements. However, due to the nature of how constrained the project is, we are not enforcing that every aspect we have discussed must be there. However, we will still enforce some baseline requirements for every group to have

- We want you to use git. Version control is supremely important when you move into the larger groups at companies and, thus, we want to see you trying to take steps to integrating this fundamental skill into your repertoire. Additionally, it helps us track the balance of work from the software side for group members and allows us to see how much progress is made per week.
- We want something working. With how much freedom you have to implement this project, there are countless ways to approach the problems at hand. However, at the end of the day, we want to see something working, even if some additional assumptions have to be made to get it to work.

3 Midterm Presentation 1

For the midterm, you want to propose your project. While the presentation is relatively free-form, you should definitely include some of the following

1. Overall Design of the project, with reasons for the choices. Include block diagrams if possible (See Figure 1 for an example).
2. Explain how your project incorporates each of the design problems that we have brought up previously (plus any additional problems that we may bring up in class!)
3. Have you found any relevant research and coding segments? Have you done preliminary experimentation on some of the components? What has worked and what hasn't worked, if you've tried anything?
4. How will you test your project? How will you collect data? Do you see any difficulties going forward?
5. Do you have backup plans? Do you have any stretch goals, if things go smoothly?
6. Workload Partition. How will you breakdown the work so that different group members can work on things in parallel?
7. Proposed Timeline (in more detail than just the deadlines). Set target deadlines for all of your partitions. What are your targets and goals for the project?

One thing that is important about your midterm presentation is to keep an open mind - see what your classmates have as input toward your project and look at what other teams are doing to mold your own group decisions. Look at other's ideas to improve your own. Remember that even though you are proposing a design week 4 that this is in no way binding. It is a way to express what you think is a great design and see if there may be difficulties that may arise from them. In fact for week 5, the weekly report will likely be a re-evaluation of your own design after considering the other presentations.

Each student should present for roughly equal amounts of time. Use good slide practices - avoid too many slides that are words only. Use figures to support your points, if possible. Expect that your presentation should be up to 10 minutes with up to 5 minutes of questions. Your goal for this midterm presentation should be to sell your design to the class, allow you to organize your ideas, and to share the ideas you have about implementing your solutions.

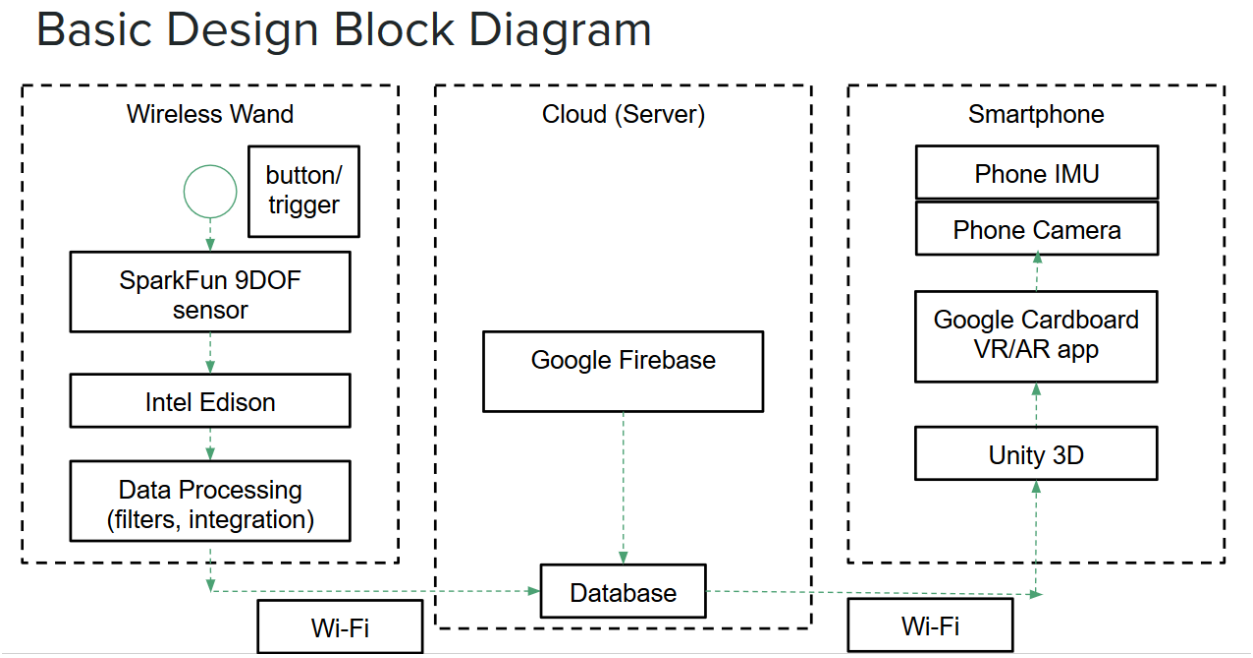

Figure 1: Consider including an idea as to how your project design will work

3.1 Updated Tentative Timeline

Expect the official project deadlines to look like Table 1. However, keep in mind that future dates after the first midterm presentation remain tentative.

TABLE 1 Project Timeline

Fall W3	Lab Tutorial (IMU / Speech)
	Finalize Teams
Fall W4	Design, Research, & Experiment
Fall W5	Midterm Presentation - Design Proposal
Fall W11	Final Presentation + Report
	Project Proposal (Report)
Winter W5	Midterm Presentation - Project Progress Report
Winter W10	Group Open House Pre-presentation
Winter W11	Final Presentation + Report

3.2 Weekly Reports

From week 5 onward (starting the week after the midterm presentation), we want to ensure that every week has a plan and has progress. As such, we will be enforcing weekly reports that you (i.e. *every individual*) must submit a document to CCLE every week, other than presentation weeks. You should also be prepared to have the document on your computer so that you can show the instructors your progress during the weekly checkup.

The requirements for these weekly reports will be:

1. Blurb about what was the plan for this week.
2. What *actually* happened this week. What did you learn about the topic that you are working on? Is there a way to use this in your own project in the future.
3. Based on what you've done, what is your plan for next week.

As these are individual reports, keep in mind that you only need to write about what your progress was in your progress report. A team progress report is not required.

Weekly reports will be judged mostly on specificity and progress. Not everyone should have the same plan and people should also express that they have made different progress. The more specific your plans are and the more specific your results are, the more apparent you spent time researching and working on your project. This means, you can add diagrams / graphs about what you have done, summaries of data you have collected, etc.

4 Group work

Group work is honestly one of the aspects of the UCLA undergraduate curriculum that is not emphasized enough that is incredibly important moving forward into your careers. These are just a couple tips from personal experience that may be useful to think about in this project.

4.1 Resource Allocation

Everyone has different strengths, and so, assigning people to their strong areas is an important part of group work. For this project, two of the most important things to consider when working in these groups is the idea that you will have to schedule times where people can meet.

In this project (generally with teams of 4), it is typically a good idea to split into two teams of 2, allowing for checking of work and such to happen.

If groups of 2 don't seem to work, one additional way that you can split work is to have each task be assigned a *primary* and a *secondary*. The primary will work on the code and the secondary can act as support for checking / debugging / etc. This way, people can work more independently but still share group aspects.

Remember that this project, while coding heavy, is not always about coding. There are plenty of things to do such as:

1. Research.
2. Data collection.
3. Testing.
4. Hardware setup / creation / 3D printing / etc.
5. Coding.
6. Art?

4.2 Documentation

Honestly, documentation is the one part of coding that no one wants to do for their coding assignments, with good reason. That is because coding assignments typically have a different goal for you - you do it in one sitting (or a small number of consecutive sittings), test if it works, and then you submit it, hoping you never have to look at the code again (which is not actually a terribly good mentality to have, but is often the experience that a student has). As such, it feels useless to do documentation because

1. It is a waste of time. You know how the code works because you have worked on it for so long.
2. There is no need to check code that you deem "done" because it already satisfies the project specs, meaning you don't have to go back and look at the code anymore.

However, as you start working more in groups and on large projects, it becomes more and more apparent what the difference between good documentation and bad documentation is. When you are working on more free-form (and larger) projects, in particular in groups, the lack of documentation can become detrimental to a project's success. Notably, without proper documentation, you will inevitably waste time understanding and debugging your code. This is as clear as

1. Other people may look at your code, not knowing what to expect. Without proper documentation, there may be wasted time in having others understand your code.
2. You may want to add some functionality to your code afterward. In most cases, you will forget how code works very quickly (especially complicated sections).

Despite that, this does not also mean that you should be documenting every single line of code that you have ever written. As such, here are just a couple tips as to what are effective ways to document.

- Document your name, the date, and what software version / OS you used for the code. This allows for a reader to more easily troubleshoot in case something goes wrong. If the code is adapted from some open-source code, link the open source code in your documentation.
- Use descriptive variable names and avoid name reuse within the same function. If you are using certain programming languages, also keep in mind that *variable typing* (e.g. `int` or `double`) also serve as valuable documentation information. You might find it useful to list important / confusing variables, their typing, and such, if you end up using an untyped language (e.g. python, Matlab). In python, you can consider using the typing library just to provide (unchecked) type documentation.
- Always document functions and classes. Include information of their purpose and where they should be used. For functions, you should always include what you should expect as inputs and outputs, along with their descriptions. This ensures that you always know how to use this function without going in to read the code itself, keeping your code "abstract" in the sense that you can just assume black box functionality.

  In addition, you can also include what are the limitations and potential bugs of the functions that have been found, if and how this function can be improved in the future to provide more functionality or better overall usage. This will be useful to help understand how to fix code and change code in the future.
- For any complicated section of code, provide a description as to how it works (or provide a link to see how it may work or where the code may have come from).
- For any procedures (compilation, execution) or overall goals and problems, keep these in a README file or something similar to make sure that you remember how to run your code and you know what you can continue to work on. This is useful for code written before a break (e.g. Winter break). Furthermore, please also try to document how you were able to install the libraries / sections of code that you use for your project.

Finally, one great thing about adding documentation is that it also provides insight and aid in writing your reports and presentations. It gives a good idea to you and your group as to what is working and what isn't working. It provides insight to how the code should work and makes sure that you understand why you are writing the code that you are writing. It also allows it to be easier to share your results and get help from other groups.

5 Tips and Recommendations

Here are a couple more tips and recommendations that I have for you guys based on personal experience and ways to help make a cool project.

5.1 Use all the open source code you can

Open source code is an incredibly common part of the culture of programming these days and you will find open source equivalents for almost everything proprietary (although likely to be of lesser quality and definitely to have less support). There are obvious benefits of open-sourced things, notably, that you have more control over your code. In addition, if many people are working on the open-source code, you will find that the code is often more likely to be bug-fixed than proprietary code. In your case, what you want with your project is to create something new from fundamental concepts rather than to recreate the fundamental concept, so you want to use as much open source code as you can.

However, do note that using open source code can be a double edged sword for your project. Most notably, you don't actually know whether the open source code that you have found was developed for the same purpose as you want to use it for. As such, you may find that you will have to modify the code. However, if these code segments are undocumented, this may prove to be a problem. Understanding some of the code, especially with respect to hardware, if undocumented, may become incredibly time-consuming, if not impossible. Finding code that can be easily edited and changed to suit your purposes will be a great place to start.

Just note when considering any open source option:

- What OSes does the code work under? Are there hardware restrictions (e.g. GPU requirement)?
- Is it difficult to install? Are there any dependencies to use it?
- Does the code provide good documentation? Is everything present or are there black boxes in the code?

Keeping these thoughts in mind, you should still search for all the code you can. Github is a great place to start looking, since it's designed to support open source code. Of course, you can always use Google as well.

5.2 Creating a Design Before Implementation

One important point to make is that you always want to create a design before you start implementation. While our midterm presentation is trying to facilitate this, this also holds for smaller cases as well. Before you write any code, you should know why you are writing it and how it fits into the grand scheme of things. The worst feeling would be to scrap large amounts of code after writing it because you haven't properly designed your code. It helps with debugging and documentation if you write your documentation before you even start coding.

While thinking up a design, keep the following in mind:

1. **Always Try to Simplify at First**.

   Getting something done is always superior to getting nothing done. Try to design your project in a way that incremental improvement can be seen every couple of weeks. Don't just design for a single end goal - design for many end goals, each of which seem possible from the previous step.

2. **Anticipate Problems - Always have a Backup Plan**.

   Turns out, life doesn't always work the way we anticipate. As almost a converse, you also want to anticipate potential problems that you might have in taking steps forward in the procedure. You want to keep them in mind and see if there are any alternate solutions or other features that you can potentially use if something won't work.

3. **Design for Testing**.

   Finally, when creating a clear picture of your design before implementation, you can also ensure that your design is easy to test. We won't stop emphasizing the need to test your work over and over because it is so important in this large scale project.

6 Additional Resources

6.1 Hardware Resources

1. **Other Sensors**:

   Clearly, one of the things that you can do is to consider other sensors. While we have not provided them here in this lab, the Raspberry Pi is one of the premier IoT systems, meaning that a lot of hardware and software has been developed to use a wide array of sensors. These include, but aren't limited to, pressure sensors, numpads, joysticks, touchscreens, moisture, and more. Here is a preliminary list of ideas for you to check out.

2. **Cellphones**:

   Of course, if you have any team members that understand how to do simple mobile app development, you always have the resource to use your cellphone as well as an additional sensor. Equipped with an IMU of its own and a camera, you definitely get some pretty good value for using your cellphone. Tilted (code is poorly documented, see below) is a project that used the smartphone camera to try to make a 3D drawing application on the phone. If anyone is interested in seeing more of the code, there is more that is resting in a Google Drive.

   However, the use of a cellphone may be discouraged for the larger commencement project, since the incredible number of phone OS versions across the graduating class would make it difficult to account for all situations.

6.2 Software Resources

Your capstone project will be a project that will take everything that you have learned in your undergraduate career and apply them in one way or another in some system. That said, it will require you to explore the possibilities of the techniques used outside of just schoolwork. As we have already seen with just image processing and IMU tracking, there are many different ways to approach how your project will work. As we collect more data as to what your projects will be like, this section may be expanded with more tips and aid with some research from the instructors' parts as well. In addition, there is a caveat that not all of these have been tested fully, but they look like they are plausible and are good places to start. If you happen to know any great solutions for any of these problems that are better than anything listed, please share with the class to better everyone's projects.

1. **Databases**

   Databases is where you tie all your data collection and processing together. Databases are one way to integrate multiple devices to see similar data. While message passing may suffice for you, if you need to do data storage or passing to other applications, you may want to consider some databases.

   In this project, you may not need to use a database to constantly store sensor data. However, maybe one consideration you can make is to store the messages and their respective timing in a database.

   (a) Not only a database, but contains a whole platform of app development options, one of the things that Firebase can do is to act as a database / server that can be used for things to access values.

   (b) List of IoT databases. While you aren't going to be making a huge-scale IoT application, you will still need to consider what kind of sources can read from which databases. This list gives you pros and cons of a couple of open-source databases.

6.3 Project Ideas and Previous Examples

Here, we will give a description for three of the projects from last year and list some of the technologies that they used (along with post their entire folder / report / codebase that they used). We cannot guarantee that the code will work on every computer and we can only hope that the documentation is good enough (we enforced that they did document their code) to have you understand everything. However, they exhibit some of great designs and techniques that will definitely prove to be useful for you going forward. We will consider posting every project if interest in looking at the previous projects is present.

1. **GTUber**.

   Description: This game was inspired by GTA (Grand Theft Auto), but with an emphasis on the player learning how to be an Uber driver. Interestingly, the group made controllers to somewhat accurately simulate driving - with a Frisbee as the steering wheel and some cardboard with a stressball as the gas pedal. It features an extensive tutorial and gameplay that involves picking up and driving virtual humans in a fixed city. Drivers had to abide by a certain set of rules (speed limit, red lights, etc) or they would be stopped by a cop to check for sobriety.

   Technologies used:

   - Unity
   - Communication - UDP.
   - Speech - PyPI
   - Image Processing - OpenCV.

2. **Synchro**

   Description: This game was inspired somewhat by Mario Party. It is designed to be a game where players will race to the finish, trying to complete multiple tasks along the way. For example, each player had to move to dodge lasers while simultaneously answer trivia questions. This project uses slightly more sophisticated software than some of the other groups, so while their concept was less developed, they had a great start to looking into a large amount of software. They even developed a full-multiplayer experience where two players could race on two separate machines.

   Technologies used:

   - Unity
   - Communication - MQTT.
   - Speech - Unity Speech.
   - Image Processing - OpenPose.

- Others - MySQL database.

3. **Zombie Archer**

   Description: This game is simply a "stand your ground and shoot until you get swarmed" zombie game. The main thing that makes this game special is that they incorporated the use of their own "bow" controller to make the game more like archery.

   Technologies used:

   - Unity
   - Communication - UDP.
   - Speech - Porcupine Library.
   - Image Processing - OpenCV.
   - Others - MySQL database.