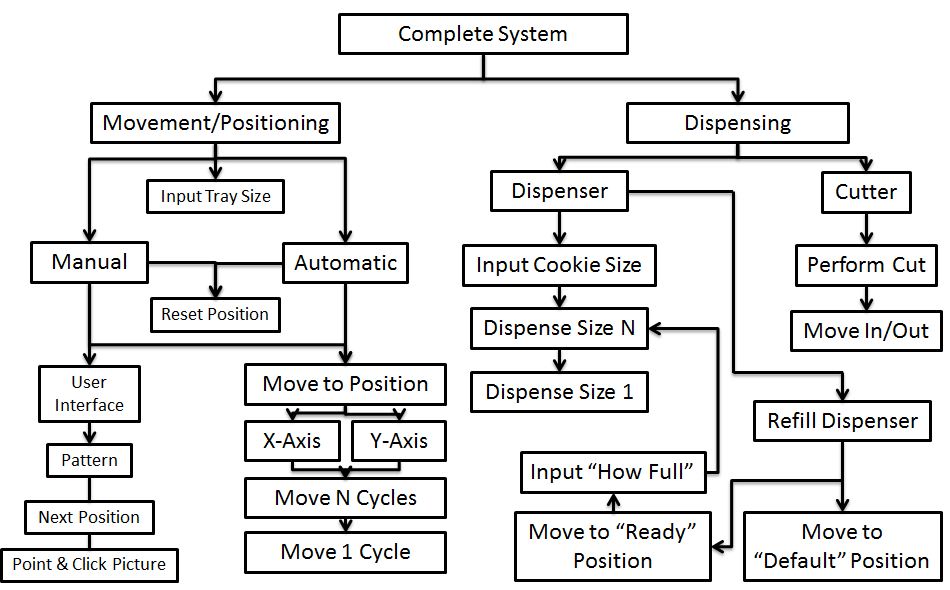
Like all PoE projects, we knew from the beginning that we needed a software component to our project. Because of the nature of Cookie Monster, it made sense that this software side should interact greatly with the user. In the beginning phases of ideation, we didn’t have a clear concept of how this interaction would go about; instead, we focused mainly on the physical aspects of our project.

As our first design review approached, we began to think of the specific interactions that would need to take place between the user and the machine. Having constructed code in the past for a much smaller PoE project, it wasn’t very difficult to decide on a list of key functions that the machine needed to recognize and perform. For our first design review, I used the list of functions that we had decided on to create a flowchart of sorts to help visualize the process that would we most likely take when building the software component of our project. Below is that very chart.



From the get-go, we knew we needed two complimentary subsystems: a movement and positioning system, and a dispensing system. This two subsystems were also present in the mechanical design, so it made sense that each subsystem needed its own control system.

On the movement and positioning side, we knew we wanted both manual and automatic controls. The aforementioned smaller PoE project forced us to include both control types; as it was still fresh in our minds at the time, we decided that it would be a good idea to include both. However, at the time, we weren’t quite sure what each would exactly do and how they would be different. I’ll discuss later why this particular choice was a fortuitous one. We also wanted to include multiple tray sizes at the time, as we believed our default tray size would be larger than the smallest tray possible.

Both modes obviously needed a way to move. Thus, we wanted to construct positioning software to move both the x-axis and y-axis of our CNC. We considered implementing a reset position function. However, our biggest unknown of the subsystem at the time was user input; how exactly would the user tell the machine where to place cookies, especially in manual mode where the user could potentially place cookies anywhere. A quick brainstorming session on the topic proposed three solutions: the user could tell the machine a set of coordinates which it would then place cookies at; the user could tell the machine one coordinate, which it would place a cookie at, and then ask the user for a new coordinate; there could be a picture on which the user could click to place a new cookie. At the time, the third idea sounded really awesome, but I was a bit unsure; I had never used GUIs extensively before, and I wasn’t sure how well our goal would translate to it. I decided to push the idea off to the side for a while, believing that constructing a very basic user input system on the command line would be sufficient to start out with. Little did I know this idea would form the foundation for our entire software component.

On the dispensing subsystem side, we wanted to perform two distinct actions. The easier of the two was clearly the cutting; that process was the same every time and only required a very basic action. More important was the dispensing. That didn’t seem very difficult to program at the time either, although it did require some level of user control, as we decided that in order to give our users more control than just movement, we would give them control over cookie size as well. Larger cookies means more dough, so the user had to be able to tell the system how larger he or she wanted the cookies. However, during this time, I had also realized something crucial: the user needed a way to refill the cookie dough when the dispenser was empty. This was a vital aspect of the project that could have been easily overlooked at first. At the time, we decided to give the user more choice: they could either refill to a “default” position, presumably at the top, or to any position on the piston, which they would need to clarify. Refilling would eventually become one of the bigger challenges I would have to face when writing the code.

These ideas for the software side of our project were all that we had going into our first design review. Because of this, we didn’t really have the opportunity to receive feedback; after all, most things look very good on paper, but building them is a different story. Once the design review had passed, I decided it was time to start building our software, even if it might potentially be done before there was a mechanical system to test it on.

As I previously stated, up until this point, my conception of the user interface consisted of only a command line where the user could input coordinates to place their cookies. After all, that’s all I had built in the previous project. But after thinking over our third positioning idea more, I realized that that’s what a user would really want: an easy-to-use representation of cookie placing. Although I hadn’t made a GUI before, I decided that there was no better time to try than now. And so, I dived right in.

My first set of obstacles was setting up, which including deciding how I was going to make the GUI. The two most recent languages I’d used were C and Python, and something told me that C wasn’t the best language to be writing beautiful GUIs in. Python it was! I knew Python could make some pretty good GUIs, since a team I was on last semester had made a nice GUI in Python. After some brief research, I decided that Tkinter in Python was my best option; it appeared to have a lot of documentation and support, which I felt was vital considering I had never written a GUI before. From there, it should have been simple to go straight into coding; unfortunately, it wasn’t. Apparently to use Tkinter I needed a certain version of Python, but I didn’t have it, so I needed to install it. But you can’t have more than one version of Python on your computer or everything breaks. But even then I had to reinstall it like three times. It was just a hassle. Oh, and I did I mention I was doing everything in Windows. In retrospect, this was a great idea, but at the time it felt like a bad decision. Anyway, after like two hours of trying to get Python to compile Tkinter, it finally worked!

Time to begin! First things first: I knew I needed to differentiate between manual and automatic mode. So I quickly made the starting screen two buttons, one for each mode. Simple enough. But then I had to make it so that when a button was pressed, a new window would open that would be the true interface. I also had to make it so the first window disappeared. It took a little bit, but I got this to work. Now I was left with a big blank window whenever I clicked either button.

Well that’s no fun! I needed some pizazz in my GUI! I went online to find a picture of a cookie sheet taken at the most birds-eye view possible. I found one, but it wasn’t completely perfect; I guess it would just have to do. Placing it within the GUI was really annoying though. I had to learn all about importing pictures and using canvases in Tkinter. Learning this was very valuable, however; I soon learned that canvases had a very simple way to obtain the coordinates of a mouse click. Perfect! That’s exactly what I needed to send to the Arduino.

Great! I had a cookie sheet! But what’s a cookie sheet without cookies to place on it? After obtaining a picture of tastiest cookie I could find, I attempted to place the cookie on the coordinates of a mouse click. But wait, where was my cookie! It didn’t show up! Apparently on a canvas you need to make sure that your pictures are on the top so that you can see them. Additionally, you also need to import your pictures in a very specific way that requires additional packages.

So great, I had a picture of a cookie on the sheet! But what’s this? Why was the cookie surrounded by a white square? I had thought I made the background of the picture transparent and saved it as a .gif? I supposed that wasn’t enough, but for the time being, I decided to put off the problem and just make the background grey to match the cookie sheet. I had other problems to worry about. Namely, whenever I clicked somewhere else on the sheet, the previous cookie would disappear. Or would it?! I also strangely couldn’t click on where previous cookies had been placed, even if nothing was there. That hinted to me that the cookie was still there, but there was no picture to represent it. To make it past this dilemma, I decided make an array of cookies, and each new cookie would be placed within the array. This entailed importing the picture every time I wished to make a new cookie, which I didn’t want to do since it’s clunky. However, I believed this is the only way to do this, as Tkinter references the specific instance of the picture when it updating the screen.

So after all that trouble, I finally could get multiple cookies onto the sheet at once! Horray! Around this time, the next design review was coming up shortly. But before that, there was a few more details I had to add. First was a size slider. Thinking back to the dispensing subsystem, we needed a way to control the size of the cookies we placed. I decided that a slider was the best representation of this. The slider I added had sizes 1 to 9 on it; sending one digit to the Arduino is easier than sending two, I thought to myself. I also made it so that the cookie picture linearly resized itself based on the current location of the slider, which I thought was pretty cool. On the more subtle side, I added several if statements to limit cookie placement to within the cookie sheet. We didn’t want anyone placing cookies on the rim of the sheet. I also added in a clear button to remove all the cookies from the sheet; this simply removed all of the cookies from the cookie array that Tkinter uses to update each frame with. Finally, I added a Quit button. Why? Why not? Becca and I also did a quick test between my code and her Arduino code; I had set up the sheet to send coordinates to the Serial Monitor for every cookie placed. It didn’t work very well at the time, which I later realized was caused by lack of appropriate consideration of all coordinate scenarios (i.e. sending a coordinate of (50,50) didn’t work because Arduino expected the coordinates to be a set of 6 digits). The size of each cookie was also sent to the Arduino after each set of coordinates.

The second design review went much better than the first. This time, I had something to actually show! Because of that, I also received a good amount of feedback. Most importantly, someone pointed out that there was no good way to fix mistakes; once you placed a cookie, it was sent to the Serial Monitor, and from there it was hard getting it out without the Arduino catching whiff of it first. Admittedly, I had thought about that issue before, but I wasn’t really sure how to solve it. Another person suggested that the user should be able to place cookies, and then click a button to send cookies only when he or she was finished placing them. Brilliant!

I got to work on implementing this new system immediately. I realized that the best way to accomplish this goal was to include an output buffer. The buffer would store the appropriate values to be sent to the Arduino and would only send them once the user told it to. This also made undoing very simple. I decided to split cookies up into two categories: sent and new. Obviously new cookies were cookies that hadn’t been send yet and were still in the output buffer. With this in mind, I made three new buttons: one button undid the last cookie, which was easily done by removing the most recent information from both the buffer and the cookie array; the second button cleared all of the new cookies, which worked in a similar way; and the third button wrote everything in the output buffer to the Serial Monitor.

I was very proud of myself at this point; I had constructed a fully functioning manual mode for our system. The next question was, “What exactly is automatic mode going to be?” Over the course of the previous several weeks, I had thought about the question only in passing. Perhaps automatic mode shouldn’t even have an interface; you could just press the automatic button and cookies would be placed in a 3x4 grid. That was my main idea. But when discussing cool things I could try to make the GUI do with my team, someone mentioned that the user should have the ability to place predetermined shapes. While that could be difficult, I thought, I took the idea and shaped it into something else. How about automatic mode have a series of buttons, each corresponding to a different preset cookie configuration. That way the user could choose between some basic cookie layouts that are very common.

Again, I got straight to work doing exactly that. What was very interesting about this process is that the framework was essentially the same as manual mode’s. In manual mode, each cookie was placed in the cookie array and sheet, and then a series of 7 digits comprising of location and size were sent to the Serial Monitor. Automatic mode did the exact same thing, but this time it didn’t read from the canvas, but merely placed each cookie from a preset place with a preset size and sent that information to the Serial Monitor. It was a great way to make both modes essentially the same internally but not externally. With some suggestions from my teammates, I eventually refined the system to use the same output buffer as manual mode and allow the user to switch designs at will until they sent their desired design to the Arduino.

At this point, I was very confident in what I had done: both manual and automatic mode seemed to be up and running. However, there were a few things that still needed to be done, the first of which was fixing the damn cookie pictures. As I said earlier, the pictures for some reason wouldn’t load with transparent backgrounds. This problem took me much longer to solve than it should have, but eventually I figured out that I had to convert the pictures to RGBA. Only then would the background be transparent. I was extremely happy the first time I placed cookies on the sheet and they actually looked like cookies being placed on a sheet. Additionally, I made it so the cookie pictures were rotated a random angle on placement, which I thought was a nice touch. Another problem I encountered was that there was no way for a user to tell which cookies have been sent and which ones were new. Therefore, I changed it so all cookies were tinted red when placed, and only when sent were changed to their normal color. Now, this sounds very simple, but I assure you, it wasn’t. Previously, I had no need to store the location of button presses and cookie sizes, but because I was unable to simply switch the picture of a cookie, I had to set up each cookie picture sent from scratch, which required me to store this information. I’m still certain there’s a way to only switch out the picture, but I just couldn’t find it.

With most of the minutia out of the way, it was time to tackle the biggest problem: refilling. I knew this day would come! In a similar vein, we needed to figure out how to tell the Arduino where the piston was at startup, since there’s no way for the Arduino to know. We decided to combine the two efforts into one. Instead of having the Arduino keep track of where the piston was and telling the Python to tell the user that the dispenser needs refilling, we thought it was best to have the Python do all of the thinking. It would keep track of how much cookie dough was initially there, how much cookie dough was placed with each cookie, and would appropriately calculate when the dispenser needed a refill. Additionally, I programmed it to not send cookie requests that there was not enough dough to make. Python sent the Arduino a request to activate refilling mode by sending the command 9999990, as the coordinates (999,999) were not in use by our canvas. The problem then arose on how to tell the Arduino when to go back to pressing dough. Additionally, it needed to know how much dough was placed in the machine. To do this, I created a pop-up window whenever refilling was done, which requested the user to input where the dough was filled up to. The user would then have to press the Done button. This send the command 9999991 to the Serial Monitor, which was then followed by the appropriate piston location in two digits. I made sure to warn users not to press Done until they have refilled the cookie dough and set its location.

This refilling process was constructed first for manual mode. When it came time to transplant it into automatic mode, it occurred to me that if the user had to refill the machine while it was placing a pattern of cookies, it wouldn’t exactly be “automatic,” now would it? Thus, in automatic mode, the user is prompted at start-up and multiple other times to only place cookies if the dough is up to a certain level, one that was more than enough to place the available patterns. Refills in automatic mode sent the command 9999992 to the Arduino, which doesn’t expect location information proceeding it. We firmly believed that forcing the user to abide by our rules for their convenience was a healthy tradeoff.

I forgot to mention, but I also added a slider in the same window as the manual and automatic buttons. This slider was used to give the Python a starting location of the piston so it could actually keep track of it. I also made it rather big in comparison to the buttons so that users would actually remember to set the slider before going into the main section of the GUI.

Finally, I was done! Refilling worked! Time for some testing. As it turns out, I wasn’t as smart as I had initially thought. Over the course of creating the numerous buttons, I lost track of exactly what each one did. The most important oversight was that when the coordinates were less than three digits (i.e. less than 100), I added a 0 to the output buffer so that the Arduino would get three digits for each coordinate like it expected. The problem was that when undoing cookie placements, it only removed the last three elements of the output buffer. But if this 0 was present, each set of seven digits (i.e. one cookie) could be either four or five elements long. I somehow missed this when making the initial change, but I soon corrected it after discovering it.

Now that the code worked perfectly (or at least I think it did), I decided to have to fun. I deserved it, right? First, on the more fun side, I added some secret dialog boxes. On manual mode, if the user hits the Clear New button twenty times without placing a cookie, a dialog box pops up that says “Make up your mind and place some damn cookies already!” On automatic mode, if you hit the pattern buttons in a specific order, a dialog box pops up that says “What?! How did you find this place? LEAVE, AND NEVER RETURN!” Not very funny, but I couldn’t really think of any references at the time. I thought it was fun making them at least.

On the more cosmetic side, I added in two headers to the GUI. The first one says “Cookie Monster” with a green background. I decided to place it in the starting window with the piston slider and manual and automatic buttons. That way, when people open it, they know what it is. The second one also says “Cookie Monster”, but it’s in this weird cartoony font with a black background. I placed it on the main GUI above the sheet canvas. I think both of them look fairly pretty, but what do I know?

Well, that’s everything I’ve done. The one thing that still remains to be done is calibration with the mechanical system. What I mean by this is that the Python is currently keeping track of how much cookie dough the dispenser can hold and how much dispensing a cookie of each size uses up this dough. However, I don’t actually know this information myself. Right now the GUI is programmed to store 20 units of dough. Each cookie uses up an equal amount of units as its size on the slider. Obviously this isn’t true, but I have nothing to test it with since calibration hasn’t been done yet. Additionally, I don’t know how much dough needs to be in the dispenser to place an entire pattern in automatic mode. Also as of right now, I don’t know how large each cookie size should be on the GUI sheet. Essentially all of the calibration issues on the GUI side relate to keeping track of the dough. There is some calibration with the movement that needs to be done, but that’s more on the Arduino side; all the GUI has to do is send the Arduino the proper coordinates and cookie size over the Serial Monitor.

And that’s the end of my journey! I’ve really learned so much during this process. Not only have I truly explored GUI creation for the first time in my programming career, but I also learned a great deal about user design. A lot of thought and effort had to go into each design decision I made, but in the end, I feel like I was able to accomplish every design goal I put forth. I definitely couldn’t have done it without the help of my team and others giving me suggestions along the way. I’m very glad I was able to have this experience, even if the project didn’t turn out as well in the end as we all would have desired. It’s my sincere hope that with the knowledge and expertise I’ve gained during this project, I’ll be able to go on and do bigger and better projects in the future. This is merely the first step on the stairs to success.