# Week 10:

**Date:** 10/28/2022

**Hours:** 8

**Description of design efforts:**

**Game Logic:**

This week, I spent some time planning out how to better interact with the game. For example, we will want to have a “gameplay” mode, a start menu, a pause menu, and other various overlays, such as a “game over” display, with possibly different button inputs. I determined that the handleBounce function (taking the microcontroller packet’s bounce flags as input) will only be different if playing a different game mode, handled likely by a GameObject object. Perhaps it should be refactored as “handleScore”. Each game mode will be handled by a GameObject object and will have their unique classes that inherit GameObject. They can then have their own handleScore function, as this process will vary by game mode.

The handleButton function, rather, was the subject of most of my planning, as Jack plans to tackle most of the object-oriented programming framework of the game logic. Here, it feels more sensible to instead use an enumeration of values for the certain menus, and based on the current menu, we will enable or disable particular button actions. For example, we won’t want to change the game mode during the game, or in the pause menu; only from the main menu. I began by devising each menu and what buttons should be applicable there. That information is shown in Figure 1, below.

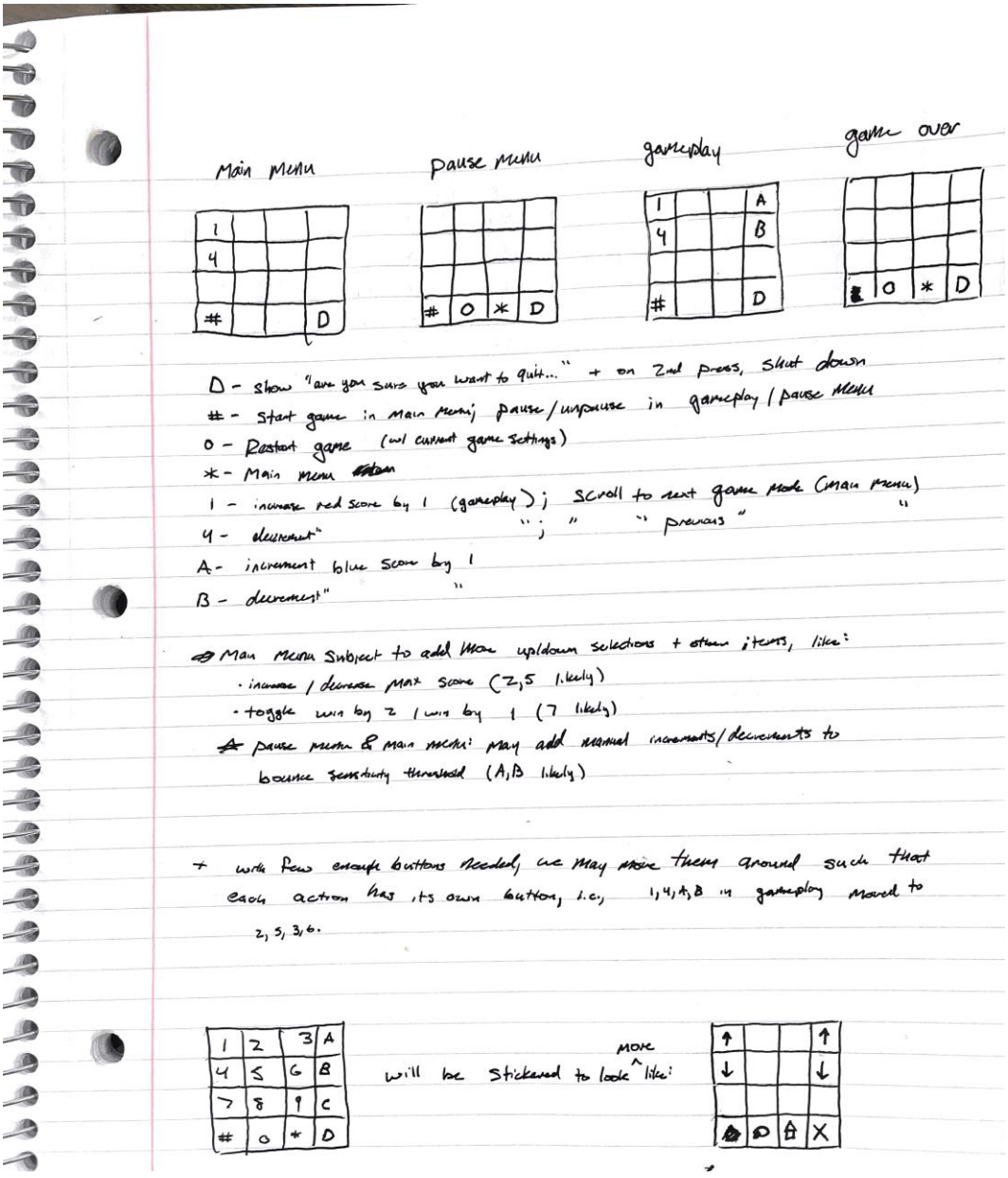


Figure 1. Button matrix design.

**Microphone Array Trilateration:**

I spoke with Dr. Walter this week about my algorithm, and he agrees that it should be feasible, at least in theory. To test feasibility, I set the ADC sample rate to 200 kHz, effectively sampling each of 4 sensors at 50 kHz. With this setup, I store 200 samples for each sensor after the first threshold cross on any sensor. I found that we can safely detect which microphone on either side received the signal first – always catching the cross at the $0^{th}$ position in the array. The second microphone, about 10 inches away, detects a bounce at around the $20^{th}$ position of the array, if the bounce were to be directly on the first sensor, or somewhere beyond it, on the intersecting line (as the first bounce will always be detected at 0, and a bounce on the intersecting line should have a constant delay behind the first sensor).

Thus, if we were to sample each sensor at 50 kHz, we have the capability to detect distance almost to the inch! Of course, this will rely on the algorithm running correctly as well. And it may not be feasible to sample each sensor that quickly if we will be using 8 of them. A lot of the progress made in determining the best sample rate, such that we maximize our ability to detect the ball's position, without harming performance of other processes. There are some methods we could use to simplify this:

1. If we just detected a bounce on the red side, we only need to sample microphones on the blue side, for the purposes of positioning. Of course, we wouldn't receive signal that the ball re-bounced on the red side, but we already have a built-in time-out feature if the ball is not detected to bounce again, and that software can cover this issue. This would allow us to effectively double the ADC sample rate.
2. We don't need to sample the sensors as fast as we do for position checking if we aren't expecting a bounce. The sample rate can be reduced when the game is not in "gameplay" or could be shut off altogether. It is possible for the sample rate to be low even until we detect a bounce, but the delays in switching the sample rate may be problematic.

Beyond basic testing, I did not spend time programming the algorithm yet. Instead, I spent a significant amount of time working on the hardware of the system – planning out the location of the microphones, attaching them to the table, and wiring them in a reliable fashion. Once I test the new microphone setup with the current code, to ensure they all work, I will begin development of the trilateration algorithm.

**Next Week:**

Next week, I intend to pursue completion of the following tasks:

1. Debug the game logic to properly score a game of ping pong, beyond any table-moving or intricate rules, such as a let.
2. Continue work on trilateration algorithm.
3. Work as a “freelancer”, filling in man hours where needed to assist my teammates.