———————————————————README FILE
———————————————————

All code is combined in the main.c or lcd.c codes. Please read instructions carefully to run game code for each part of the lab.

PART 1

LCD initialization functions listed in the lab manual were written as described in the lab manual and saved as *lcd.c* (see attached code). These were based on knowledge from lecture and lab manual. Additionally, code/functions already included with this file were used as reference to figure out variables for the needed functions. The functions written are called in the *main.c* code to create the pong game. A demo was done to show the TA's these functions all worked on the LCD.

PART 2

As shown in the figures above, the touchscreen was added on top of the LCD and then interfaced using ADC as instructed in the lab manual. Calibration code was written to read ADC values and then scaled to fit the LCD for both x and y values (see attached code for comments). Pixels were displayed in demo for the TA's after a specific point was pressed. This was done separately from the pong game as per the TA's instructions because it was not necessary for the pong game to work. A setup and 2 different calculate functions were created to scale/calculate ADC values within the main function and draw a filled circled on the corresponding touchscreen area. First, all variables and libraries were included/defined followed by the functions mentioned above. These were all implemented in the main function to continuously run such that registers/ports and LCD were set up first and each calculation was done starting with the x coordinate. The idea behind this part was implemented in the next section, but is not necessarily the same in the code.

PART 3

The pong game was written to work following the criteria in the lab manual and TA's instructions. All criteria were met as described in the comments of the code (see attached code). Criteria that was unclear was clarified by the TAs during office hours. For example, we were told that proper gridlines did not necessarily mean physical lines, but areas for mapping, or that there were no specifications for number of rounds wanted. Touchscreen mode was achieved using the same calibration from part 2 of the lab. Please note that due to calibration sensitivity, touch screen only works when pressing inside the board boundaries drawn on the LCD. (DO NOT PRESS OUTSIDE OF THESE BOUNDS) Additionally, sounds were generated based on different pitches due to different OC1A values because sound criteria other than when to sound were not specified. Different LED colors were achieved turning certain backlights on and off. Game and touchscreen functionality was shown to the TA's for the demo. Similar to part 2, the code first starts by defining and including all libraries and variables. This included initialization for many of the global variables. Functions to setup the game were then written using functions from part 1 of the lab, and used to create other functions to run the game. The main code uses these functions to first set up all registers/ports, timers, and initialize variables. As long as the criteria are met, the game keeps going using the aforementioned functions constantly updating and checking to make sure when to end the game. Because the code is meant to be run sequentially, this could be grouped as the start or set up of the game, adding the necessary functions, running the actual game, and ending/restarting the game. Before using, please make sure nMode is set to 2 before running as explained in the next part of the lab.

PART 4

The pong game was added to such that the accelerometer was used rather than the

touchscreen feedback (see attached code). Rather than writing two separate codes, we kept our code for Parts 3 and 4, but with the ability to switch between the two modes based on a variable that is hard-coded into the overall *main.c* code (nMode=1). The only difference for the ADC values was the accelerometer used the values output from the x-axis of the accelerometer rather than the touchscreen values. Therefore, there was no need to scale in a calibration code. While trying to complete this part of the lab, we found our breadboard was uneven and caused a bias towards a certain side. However, this was alleviated if hand held rather than just placed on the table and tapped. Sound and light changes were integrated in this mode similar to part 3. As mentioned before, the logic of the code follows the same format of part three (touchscreen) section of the code. Please refer to Part 3 for code logic.