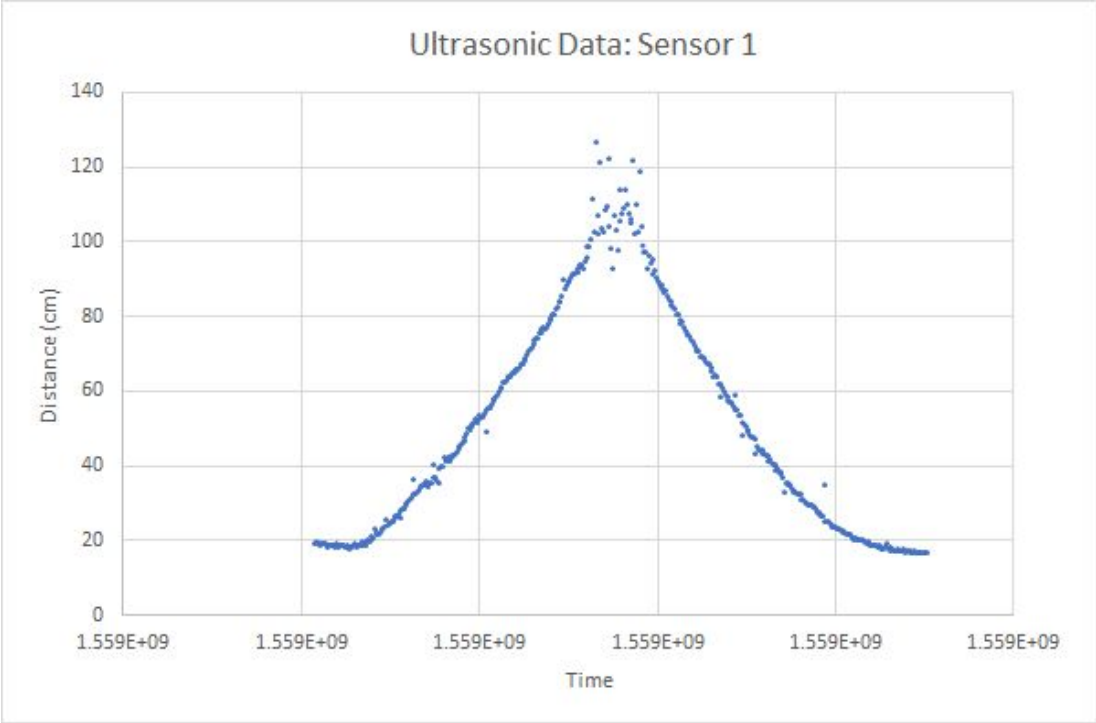
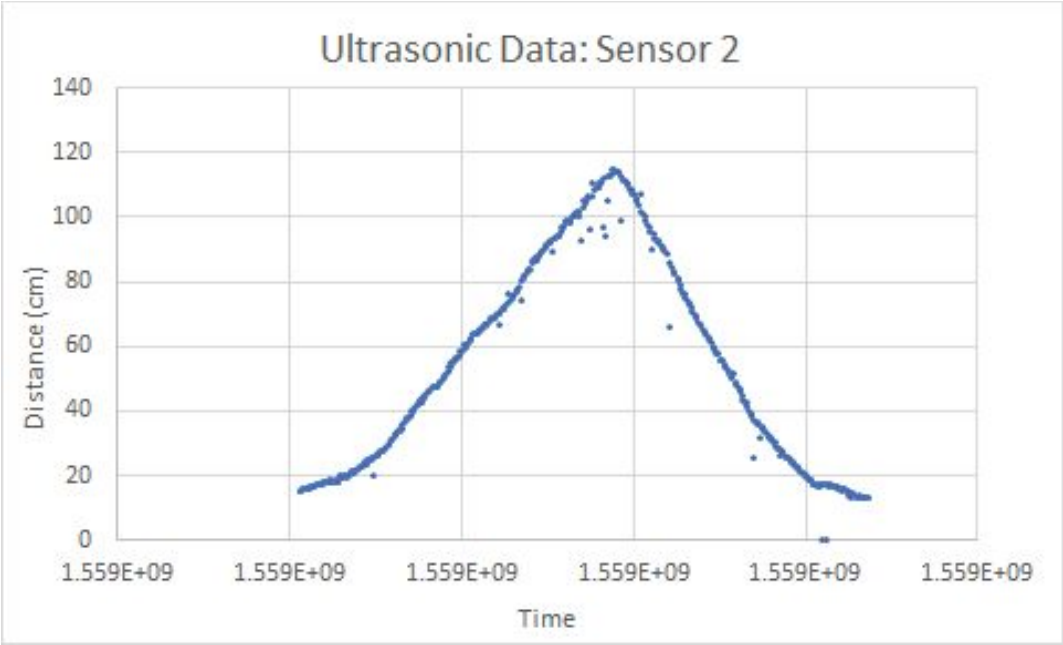
Assignment 4 Write Up Report
Tim Nguyen, Frank Ren, Dennis Zang

A. Visualize the sensor data





The sensor data is recorded from the ultrasonic sensor. We started to record the ultrasonic sensor data by placing an object around 15 cm away from the the sensor. We then move the object further away from the sensor and stop around 100 cm, and we then move the object

closer to the sensor and stop at around 15 cm. As we can see, the plots follows the pattern described above. The sampling rate is approximately 30 fps.

B. Transfer Function/Mapping

We used ultrasonic sensor to control the paddle. The ultrasonic sensor works by vibrating at a really high frequency and send a pulse, after some time, t, we receive the echo back when the pulse hit an object and bounces back. After getting the time elapsed between sending and receiving the ultrasonic pulse, t, we use speed of the sound wave times the time elapsed to determine the distance between the sensor and the object. The unit for the distance, in our implementation, is centimeters. The sensor sent the data to Raspberry Pi using GPIO ports. Then we then send the distance data from the Raspberry Pi to the local PC using UDP connection. After getting the distance data, the PC translate the distance data into the direct position of the paddle within the display window.

We ensured that the paddle will not move out of the display window size by having if statements for the distance we get from the Raspberry Pi before we update the paddle position, and by checking the distance value beforehand and set a range, we can always keep the paddle in the display window. In the end, if we want to move the paddle up, we simply move the object further away from the sensor, and vice versa.

The mapping is done linearly. First, it checks if the distance received is outside of the range 25-75 cm (and set the position to the top and bottom of the screen, respectively). Otherwise, in this range, the Python Pong script will map the distance to a height from the lowest possible to the highest possible according to a linear function (along the lines of "<highest_position> - ((<screen_height> / 50) x (<distance> - 25))") (note that in the actual program, the height position is mapped from top to bottom).

C. Latency

The latency is small but not negligible in this project. The lag between when the sensor reads the data and when the paddle actually moves on the screen is due to three reasons. First, we need to convert the data from elapsed time to distance in centimeter. The data is sampled at at least 20 fps (considering the 2 ms latency of measuring 75 cm delay and the 25 ms delay per measurement loop), so the latency per sample here would at most be about 30 ms. Second, the time needed to send UDP packets need to be considered. Due to the low processing power of raspberry pi, we can't really run the Pong game at high enough frame rate. So, we sent UDP to minimize latency and maximize the Pong frame rate, but the latency would be neglectable (about 0.15-0.20 ms in network at time of writing this report) (values derived from round trip times of pinging the laptop from the raspberry pi). Third, after receiving the data, we need to run the Pong game itself, including parsing the distance into position. In our case, we only had the script read the most recent UDP packet for position, so latency wouldn't matter too much here (as the game is mandated to run at 25 fps by the pygame delay). So, the minimum delay would be 30 ms plus

the delay of sending the UDP packet over the local network (neglectable) plus the delay of calculating each frame of the Pong game (about 40 ms), which totals to a maximum of about 70 ms. All three cases added up to small latency, and it doesn't affect the overall experience of the game.

D. Accuracy

Main obstacles of this assignment were learning how to set up the hardware to connect the raspberry pis to the ultrasonic sensors, writing the software to read the sensor data as float values by the raspberry pis, how to send the data to our computer's Pong script via UDP, and then how to convert the float values into byte arrays that could be used by the Pong script to map the location of the paddles. These obstacles were overcome via trial and error, and research over the Internet. One significant obstacle we had was in our implementation of reading data for a two-player game: we initially wanted to use two raspberry pis, each connected to a ultrasonic sensor, and using one for each paddle. We wanted to run the same script on each pi to read data from one sensor, but after complications with sending data from two pis via UDP, we changed our script to read data from two sensors and used only one raspberry pi.

Also, note that the second sensor may have been broken at time of demonstration, as when we tried running the sensor again immediately after the demonstration (while still in discussion), it wouldn't work (interchanging sensors works).

E. Resolution

The movement of the in-game paddles could sometimes be shaky and erratic when the real-life paddles were held outside the 25-75 cm range. The data was also being constantly read at high frequencies and picking up every little movement, and was also possibly receiving interferences from stray objects like a hand, thus leading to the in-game paddles sometimes appearing unstable. However, we learned that using large enough paddles like a notebook while also finding the correct distance to stand at would mitigate these effects. As one can see in the sensor data graphs from part A, the curves formed are mostly smooth except when the sensor is farther away, where there is a little bit more noise. But overall, the data received is consistent, forming straight lines as we moved our objects farther than closer to the sensors.