## Documentation for Workshop 5 - Wireless Interfacing

Yixuan Xu

Manqiu Wu

Xinghao Li

Part 1:Sensor Readout

* To reduce the number of readouts to the Serial Monitor, we could either change the baud bit from 9600 bps to a lower frequency or change the line “Serial.print(millis());” to “Serial.print(seconds());”. Note that if we were to change the baud bit on the code, we need to change it for the Serial monitor setting as well. Otherwise, the transmission of data will be messed up. We have tested this during the lab and our proposal worked.
* For setting up the 4 ultrasonic sensors, we used one pin as the trigger pin for all four ultrasonic sensors to save the number of pins used.
* We then ran into a problem as only one of the four ultrasonic sensors provided reading while the other sensors read 0. Soon, another problem arose - the Arduino board broke. The LED indicator only blinked and then turned off immediately. We debugged it by realizing that there could have been a short circuit and turned out that there was one in one of the short circuits.
* We then ran into another problem: often, at least one of the sensors will read “zero” each time when there is nothing in front of the sensors. We conclude that all the sensors are properly working, but will become finicky when there is nothing

Part 2: Software Serial

* We did not change the code for the Rover Uno
* For the code for the Sensor Uno, we incorporated the code from the first task and the second task together, and the code was successfully compiled.
* When connecting to Bluetooth, we ran into a problem with the connectivity of the sensor. Essentially, we know that the sensor was already paired before, and thus we expected it to connect automatically to our Andriod phone. However, the sensor failed to connect. When we tried using another app, the “Arduino Bluetooth” app, the app suggested me to restart the sensor, and it worked. The app was also pretty good in the sense that it has a Terminal mode, which has a monitor and a keyboard, but it also has a controller mode, which allows me to configure each key of a virtual controller, and drive the rotor with the virtual controller as if it were a game.

Part 3 and 4: Readout and Visualization

* When attempting to read and visualize sensor data using MATLAB, the code that we used couldn’t compile because it could not recognize the COM port. Upon further investigation, with the help of Google, we noticed that the COM port cannot be properly closed, and therefore cause it to be unreadable. We couldn’t find a way to close the COM port using “fclose” or “clear all’. In the end, we fixed the issue by restarting the MATLAB interface, and it worked.