DIY Electric Skateboard Write-up

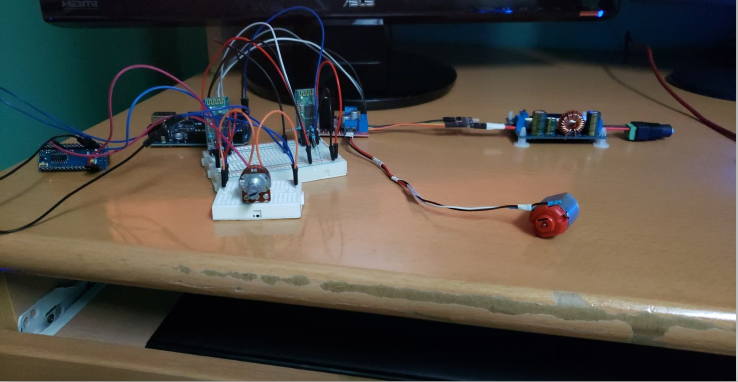
Austin Chua

The idea came as I was learning and getting familiar with the Arduino microprocessor. I was experimenting procedures for motor control along with peripherals when I discovered the potential that the Arduino could accomplish. I thought it would be a neat idea to fabricate my own electric skateboard using Arduino micro controllers.

The first step was to pair the two HC05 Bluetooth modules to communicate with each other through serial communication. This was accomplished by connecting each module to the Arduino through the TX and RX pins to allow to communicate with each module Electronics/Hardware

The skateboard end of the system is powered by two 3S LiPo batteries wired in series to step up the voltage to 22V. The remote end of the system is power by a 1.5V AA alkaline battery wired through a 5V step up booster to ensure ample power to the Arduino. The battery and Arduino housing were secured to the skateboard using bolts and nuts. The battery housing was padded with foam to ensure the safety of the batteries.

through the Arduino’s AT command mode. With AT commands, I was able to configure one module as a Master and another as a Slave. With a master and slave module, I set each module so that they will only connect to each other respectively by pairing the Bluetooth addresses.



Once paired, I began writing the scripts for the remote side and the board side. The remote script consisted of checking connectivity and writing an averaged potentiometer value through to the Arduino’s serial communication. Safety precautions were taken so that if the potentiometer on the remote was not dialed at zero on start-up, the system would not function until the dial was set to zero to avoid skateboard drive-offs. A blue LED is used to display system status to the user: a steady pulse indicates the need to dial the potentiometer to 0, a rapid pulse indicates no connectivity, and a solid LED indicates the system is functioning. The most common bugs I encountered were related to the system not functioning properly due to confusion between the system states. This was solved through incremental development, trialing bits of code at a time to diagnose the location of the issue.

The code for the skateboard end Arduino was less complicated than that of the remote end. The task of the Skateboard Arduino was to read a value and write it to an Electronic Speed Controller (ESC) functioning with the Arduino Servo library to control the speed of a brushless motor. A safety protocol was created to ensure a slow deceleration of the skateboard on the event of a loss of connection. This was accomplished by checking for connectivity and using a 10 ms delayed loop to decrement the speed value until 0.

Once the code was initially finished, I let the code run for one hour as a stress test. I realized after idling for about 5 minutes, the system would crash. After testing and research, I learned that allowing the two Arduino modules to communicate at max baud rate for a long period of time will eventually overload the processor. So a delay of 100 ms was put in place to ensure the continuity of the program.

Electronics/Hardware

The skateboard end of the system is powered by two 3S LiPo batteries wired in series to step up the voltage to 22V. The remote end of the system is power by a 1.5V AA alkaline battery wired through a 5V step up booster to ensure ample power to the Arduino. The battery and Arduino housing were secured to the skateboard using bolts and nuts. The battery housing was padded with foam to ensure the safety of the batteries.

Overall, this project was a thorough learning experience for me. I learned how to properly translate the programming skills I learned from class and apply it to a practical real world scenario where code is used. Learning how to properly use peripherals to control electronics for a user is a practical skill that I have gained from this project. Not only did I learn important programming and user design skills, I also learned about the Arduino microprocessor and its methods of communication that I can apply to more advanced processors.

Images and Videos of Progression.

