ENGR 190

2 December 2010

# Segway Robot Final Report

The design of the SEGWAY robot is fairly simple. The motors were attached to either side of the robot using long blue connector pieces and wired to the NXT in MOTOR_A and MOTOR_C. The light sensor was then attached to the bottom behind the SEGWAY using a small LEGO_Technic piece that allowed the sensor to point vertically downward. We chose the light sensor because it is more sensitive to changes in light than the color sensor and the ultrasonic sensor is probably the least accurate sensor of all three since its smallest denominator is a centimeter.

The challenge of the design was how to make it easier to balance. The initial design involved many pieces in the kit and turned out to be quite bulky. The balance point became very far off center and impractical to balance. Later we decided that simplicity was best and simply attached the wheels to the controller and set the light sensor in front. We used as little pieces as possible and it worked for our purposes.

After concluding the project we came to the realization that there could have been a slight workaround to the high latency of the robot (a topic discussed later in the paper). If we had built the robot to be tall, thin, and have a center of mass near its middle then it would have tilted much slower than with our simple design which had a center of mass near the wheels. This would have given the robot time to realize it was not balanced and adjusted itself before it

was too late. This is a working theory but in essence it should have helped. Maybe it wouldn't have done enough to compensate, but there should have been a noticeable difference.

The labs used to help us learn how to program the robot using MATLAB were very useful. In fact, we used a similar script from one of the labs to program the robot for our final assessment. We enjoyed the hands off approach to this creative inquiry where we were given assignments with the instruction included. This allowed us to work at our own pace which was probably much quicker than the other students during the project, and this allowed me to balance my classes effectively without rushing to complete any work.

The programming used for the SEGWAY script was done through MATLAB as instructed using the motor control kit supplied. We decided on a script that allowed the SEGWAY to alter its movement based on the light reading given by the touch sensor. The NXT is started at its equilibrium point of balance and the script is run. Once ran, the light sensor registers this point as the target level of brightness to maintain. If the light is brighter than the target level, it moves forward to compensate. If it is darker than the target level it moves backward. This way the NXT is always upright and hopefully staying in place where it should be.

One of the problems associated with this project was lighting. Although the light sensor is the best sensor for the job, it requires no altercations in the environmental lighting. Since the sensor is fixed to the front of the SEGWAY and not free to move, the rate at which light increases or decreases when the SEGWAY tilts is not constant. As a result, the movements of the SEGWAY are very jerky. This did not seem to make the Segway impossible though. We

simply adjusted the programming to have a maximum power output of -100 and a minimum of 100 which was more than enough power to correct its offset balance.

The response time and lighting was the major issue in this project. Even connected to the computer through USB the response was still too slow and the NXT is too far off balance before it realizes it is off of its center. This *could* have been worked around if we had a supply of larger wheels to raise the center of rotation but we did not have these at our disposal. We also attempted to program the Segway using DirectMotorCommand by TA's suggestion as it reduced the latency between computer and machine. This had no visible effect on the robot's poor performance as the difference between DirectMotorCommand and our original script was negligible.

In the end the Segway was limited to balancing with the need of outside assistance. By keeping a finger ready to provide some resistance when the robot began to fall, it was given enough time to react and subsequently corrected its positioning. This goes further to conclude that latency was indeed the reason the robot could not stand on its own two wheels. There are videos online of other robots built in the same fashion balancing on their own, but they ran off of code downloaded onto the robot, which resulted in zero latency. Some of them also had a gyroscope sensor, which is a much more accurate device for gauging displacement from the balance point.

We learned through this project how to use MATLAB, logical processes necessary to overcome some obstacles in an engineering setting, and simple design procedures. MATLAB will be useful in the future to me as an aspiring electrical engineer as it is a universal programming

language used on the field as well as in my future Clemson classes. Logic is also essential to an engineer, because if you couldn't proceed to analyze a problem and be able to plan the steps to fix it you'd be in huge trouble. Since we both have a background of building toys in our day this project was a fun way to incorporate the process of experimental design for a Segway that had a practical application in the field of schematics or plans. Overall we were dissatisfied that our Segway was limited to balancing with assistance, but we felt that this project was entertaining and we learned a lot from the design process. We wish to continue with this creative inquiry next semester in building the Rubric's Cube.