**Memo**

**To:** Dr. Steele and Dr. Li

**From:** Andrew Conners and Yer Yang

**Team #:** 408

**Date:** 2017-05-12

**Re:** Lab #7: Robot Show (Balancing Robot)

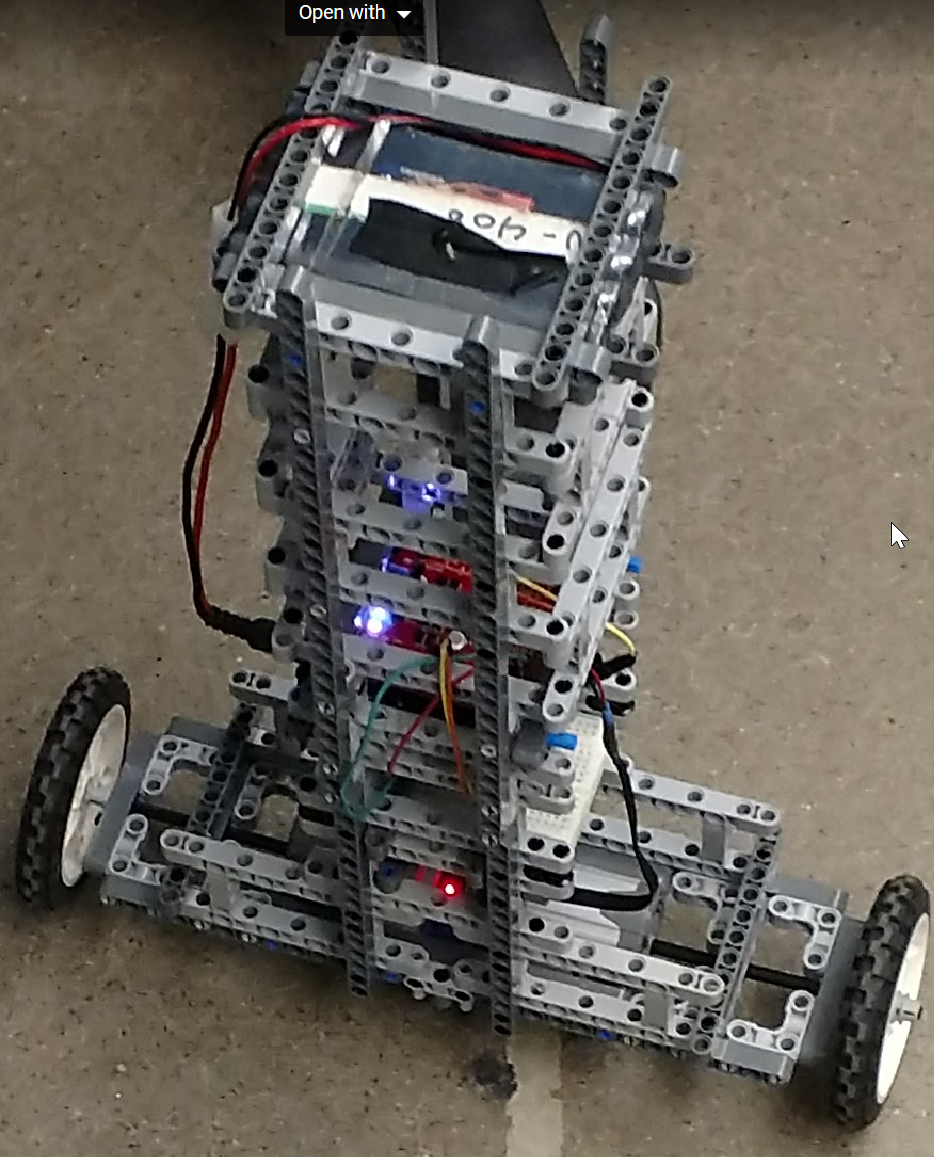
**Project Statement:**

Balancing robots are a challenge and fun way to demonstrate the principals learned in this class. In lab 7, a balancing robot was designed that utilized a 6-axis gyroscope and accelerometer to balance. The robot relied on a variety of techniques to maintain balance, including PID control and tuning of PID parameters. Overall this was a challenging and fun project with some great learning opportunity.

**Methods:**

**Sensor Configuration, PID Control, and Inverse Pendulums**

The robot sensor configuration consists of one MPU6050 6-axis gyroscope and accelerometer. Initially, balance was attempted with the sharp GP2D120 sensors but this proved to be ineffective due to interference between sensors. The MPU6050 gyroscope was mounted directly on the axle of the balancing robot to take advantage of the inverse pendulum principal of balancing robots. See figure 1 for a layout of the robot configuration. Additionally, PID control was implemented into the robot via way of three potentiometers, one for each of the “P”, “I”, and “D” parameters. Originally, the balancing robot was tuned without the potentiometer but the addition of an LCD screen and manual PID tuning made for a more fun learning robot.



IMU Location

Figure 1.1. Balancing Robot.

**Sensor Setup**

The gyroscope needed to be calibrated initially in order to ensure the balance point was accurate. There were various gyroscope tuning libraries available in the Arduino IDE. For the tuning of the MPU6050, the I2Cdev.h and MPU6050\_6Axis\_MotionApps20.h libraries were used. Additionally, the Wire.h library was required for communication with the I2C gyroscope. The potentiometers were mapped to the kP, kI, and kD values and the PID\_V1 library was utilized for updating the PID values.

**PID Tuning**

The PID tuning was done initially at various trial-and-error points until a balanced robot was achieved. The tuning parameters (gains) for this robot were 70 for proportional control, 200 for the integral control and 0.1 for the derivative control. The changes in these tuning parameters were evident as they were adjusted. To make the project more interesting, manual tuning parameters were mapped to potentiometer inputs to the analog pins and the manual gains were output on an LCD screen. This allowed for manual tuning of the PID parameters and real-world output on an LCD screen. In other words, the PID gains could be adjusted in real time.

**Results:**

After much trial and error (mainly in adjusting the PID gains and setting balance angle), a stable robot was achieved. The Lego structure used on the robot proved to be quite durable and even survived quite a few falls. The PID integration and manual tuning also proved to be a success and was very useful for visualizing the real-world output of various tuning parameters as they were adjusted.

**Challenges:**

One challenging aspect of this project proved to be the drift on the 6-axis gyro and accelerometer. One could set the correct parameters one day and come back the next day, only to find the robot no longer balances. To overcome this challenge, the robot was re-calibrated if the gyro appeared to be malfunctioning. This proved to be quite effective to resolve any drift issues in the gyro.

Additionally, the placement of the gyro proved to be critical in finding the correct balance. Initially, the gyro was located at the top of the robot (near the weighted point on the inverse pendulum.). However, after some research and some trial and error, the gyro was re-located to the axle of the robot where the small changes in the pose of the robot could be measured and adjusted as needed.

**Conclusions**

**How well did the robot perform?**

Overall, the robot performed well and was able to maintain upright balance almost indefinitely (at least until the battery wears out). Some additional challenge were put on the robot as well, such as pushing the robot to see if the balance could be compromised. Small pushes to the robot were correct by the programming and the robot was able to maintain balance. Overall, we achieved our goal.

**Biggest Surprise?**

The biggest surprise was how challenging the balancing robot proved to be. Initially, it seemed a 6-axis gyro would be the magic bullet with plug and play capability. However, factors such as robot design and placement of sensors were unexpected surprises.

**What Was Done Well?**

The coding of the robot and the design of the structure was done well. The robot survived multiple drop tests and still maintained balance. Additionally, the code was easy to update and adjust as the gyro went out of calibration.

**What Was Not Done Well?**

The manual PID tuning, although a great learning demonstration, proved to be quite sensitive to change. Even small adjustments in the potentiometer values led to large changes in the output, making it quite difficult to manually tune the gains. A bit more time with the code would resolve these issues.

**What You Could Do to Improve Next Time?**

On the next design the biggest area for improvement would be making the robot mobile. Balancing robots are cool but mobile balancing robots are the next level and also would provide a great challenge. Additionally, some of the PID manual tuning parameter mappings could have been adjusted to make the changes in potentiometer position less sensitive.

**References**

"Self Balancing Robot." *Wikipedia*. Wikimedia Foundation, 14 Apr. 2017. Web. 14 Apr. 2017.

"Navigating a Simple Maze." *Navigating a Simple Maze - ROBOTC API Guide*. N.p., n.d. Web. 14 Mar. 2017.