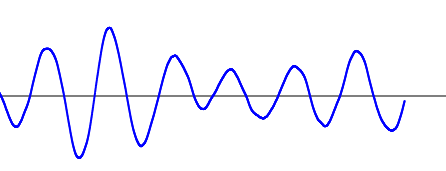
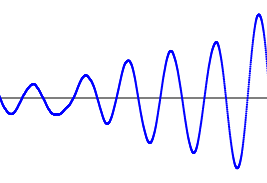
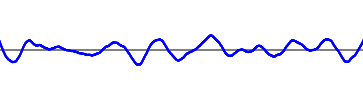
Our autonomous program uses a rotation sensor to determine where the robot is on the field. The particular sensor that we use is called the BNO055, and combines readings from a magnetometer, gyroscope, and accelerometer to get better rotation readings. We use PID algorithms to allow the robot to **make turns accurate to 2 degrees and move straight for up to ten feet with error of just on degree.** PID can be broken up into three parts: proportional, integral, and derivative. The proportional part of the algorithm looks at the distance between a robot's current and target rotations. The greater this distance, the faster the robot turns to reduce it. Although a purely proportional program works great for some applications, the actual rotation tends to oscillate around the target value. Using a sample program that we created, I plotted the actual value (in blue) and the target value (in grey) of a purely proportional program.



The integral part of PID looks corrects for past error. For example, if the value has been consistently under the target, the integral will try to increase the current value. When a program is created that uses just the proportional and integral algorithms, the actual value still oscillated but now the robot can correct for systematic error, such as heavy friction on a specific wheel. Here is a graph the PI (proportional integral) program:



Once the derivative is added to the proportional and integral, the algorithm becomes very effective. On its own, the derivative looks at the change in rotation over the last .1 seconds. If the robot is moving forward, the derivative part of the program tries to make the change in rotation 0. In a program that controls robot turning, the speed of the turn can be set by changing the target rate of change. The derivative constant counteracts the oscillation shown above in the PI program, and allows the program to react well to random error. Here is a graph of the full PID program with all of the three separate algorithms working together:



Although the actual value doesn't always match the target, it is much closer than in either the P or the PI programs. Once the full PID program has been written, the constant must be tuned. This process, although laborious, allows the robot to do a much better job holding its position. Once we tuned the sample program, the robot's rotation was much closer to the target, as is exemplified in this graph:

