3Project 3PI Robot Challenge Report

To find the solution to the robot challenge, we used the three programs used in class, which are named “maze-solve.c,” “follow-segment.c,” “bargraph.c,” “turn.c,” and “main.c” to obtain the basic commands of the activity. These functions include setting the motors of the 3pi robot to an specific speed, printing the messages in the LCD screen of “Start,” “Stop,” “Traffic Light,” “Junction,” and “Finish,” hold the sensor values with the command “unsigned int sensors [5];,” the auto-calibration, and other general operations. We solved the challenge by reading carefully the aforementioned programs, using logical thinking to do modifications, testing the implemented commands, and taking notes of the results of the different adjustments. We adopted this method to find the solution to the 3pi challenge; we invested a significant amount of time solving it.

Some examples of the commands we declared using the provided programs are “int proportional = ((int) position) – 2000;” that means the proportional term should be zero when the robot is on the line, the commands “int derivative = proportional – last\_proportional;” and “integral + = proportional;” used to compute the derivative (change) and integral (sum) of the position. Next, we used the command “last\_proportional = proportional;” to remember the last position. With this commands declared, we computed the difference between the two motor power settings, m1-m2. If this is a positive number the robot will turn to the left, but if it is a negative number, the robot will turn to the right, and the magnitude of the number determines the sharpness on the turn.

The challenge consisted in making the 3pi robot follow a path marked with black tape. On the trail, there were three main obstacles marked; two gaps were the robot had to stop and one junction that it had to ignore. On these gaps, there was no visible line ahead or an intersection, so we described it as an end; we set the motors to zero, print “STOP!,” and we delayed the robot for 2 seconds. After the two gaps were passed and the robot did a left turn following the black tape, we faced the junction; we detected it declaring “if (sensors [0] > 100)” and “if (sensors [4] > 100)” in the “maze-solve.c” program, we printed “Junction!” on the LCD screen, and delayed the robot for 300 milliseconds. Finally, the robot kept on following the line until it hit the finish line, where we used the command “if (sensors [0] > 600 && sensors [1] > 600 && sensors [2] > 600 && sensors [3] > 600 && sensors [4] > 600);,” when the robot had this reading on its sensors, the motors will be set on zero, and it would print “Finish!” on the LCD screen.

Our degree of success was good, especially since we learned basic programming; the time we spent needs to be reduced to be more efficient in solving future academic and professional assignments. My lab partner and I are going to keep working and studying at the same quality level we used to solve this challenge to have success as future engineers in the professional field.