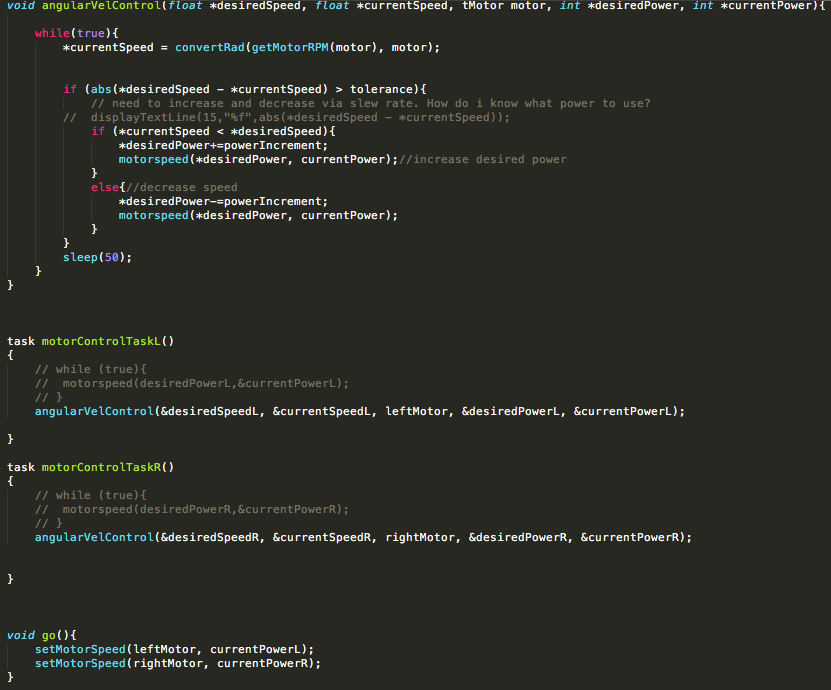
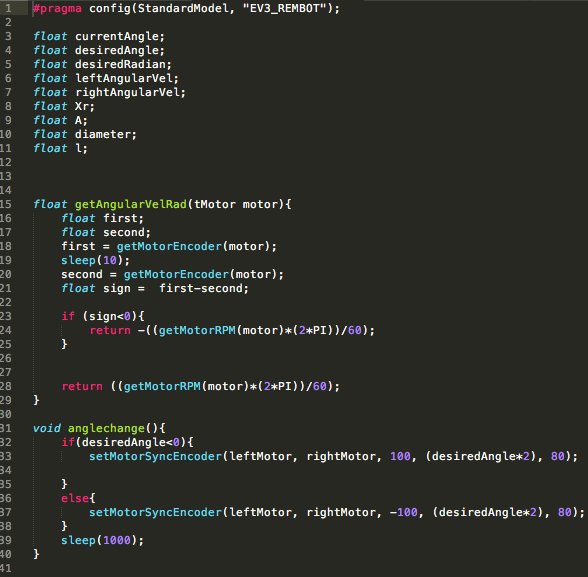
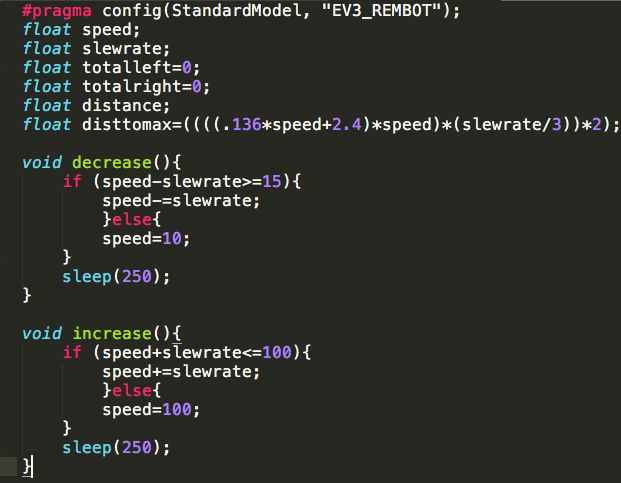
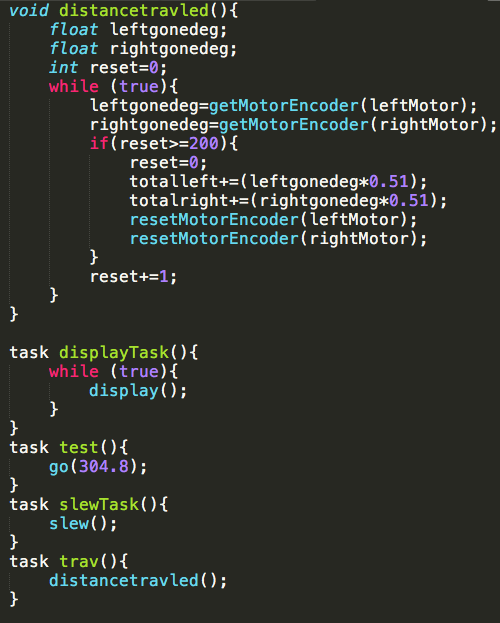
Matt Donnelly

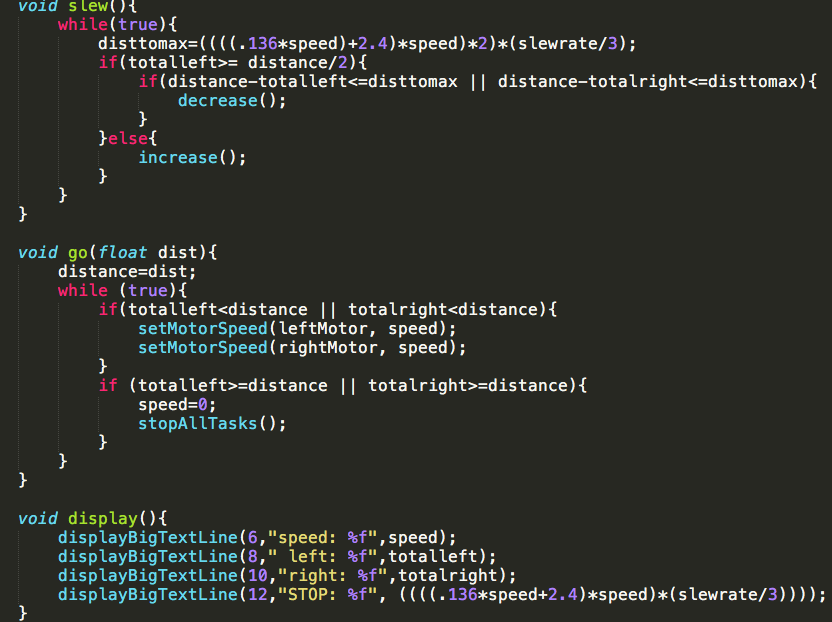
Daniel Bjorklund

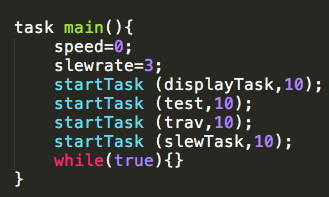
Lab 3 report

1. This program relies on our slew rate program from lab 2 that we created. Using that as a base, all we added was a way to calculate how fast it was going (done by the convertRad function) and a function that acts as a controller for our slew rate (angularVelControl). This simply checks if we are above, below, or at the desired speed and then calls motorspeed (our slew rate function) with the correct arguments for the situation. It supports the individual motor speeds through the two tasks motorControlTaskL/R.
2. This question most of the code is actually related to the screen display instead of the actual angle change. Since setMotorSyncEncoder does all of the heave work for you (since you can just pass it the angle and it’ll turn the desired amount) the only movement task is go, which just calls anglechange. The updateAngle task just constantly gets the angle gone from the right motor encoder. getAngularVelRad is essentially convertRad from the previous program but we get the sign manually since we were told that there was an error with the recommended way. The display task just calculates the various screen elements and displays them. When we tested with the recommended angles we found that this was pretty accurate, it might be off by a degree or so because of the sudden stop but it was different every time, sometimes it would stop dead on, other times it would be just slightly off. For a fairly crude method it was more accurate than I would have guessed.
3. Qsd
4. This program was developed to move the robot forward or backward while using a slew rate. This helps preserve the gears and the motor by slowly increasing and decreasing the speed. We programmed our robot to analyze the speed, distance the robot has gone, and distance left to go. The robot accelerates all the time, until it reaches 100, The way we chose to do this was by taking some sample data by testing how long (distance) it took the robot to slow down at a certain speed. We used the speeds 100, 75, 50, and 25. We then looked at the best distance it took to slow the robot down using a slew of 3. Once we got all our distances recorded, we analyzed the data to develop a formula: .136\*speed+2.4\*speed\*(3/slewrate). This tells the robot if you are going “X” speed, at this distance you need to slow down by the slew. This keeps going until the robot comes to a stop.









1. This program is very simple and really relies around two main functions, forward and turn. Forward is an earlier iteration of question 4 before we added slew rate. For this function all we do is convert the distance we need to go to degrees and then pass that to setMotorSyncEncoder with the correct turn ration. We calculated the circumference of the wheel in millimeters, divide that by 360 to get how many millimeters it goes per degree. Then we divide the desired distance by the millimeters per degree and give that to setMotorSyncEncoder. For turn, we give setMotorSyncEncoder the angle directly. Then all we do is calculate the desired LCD values and display them.

