Hunter Shiells - 500946154

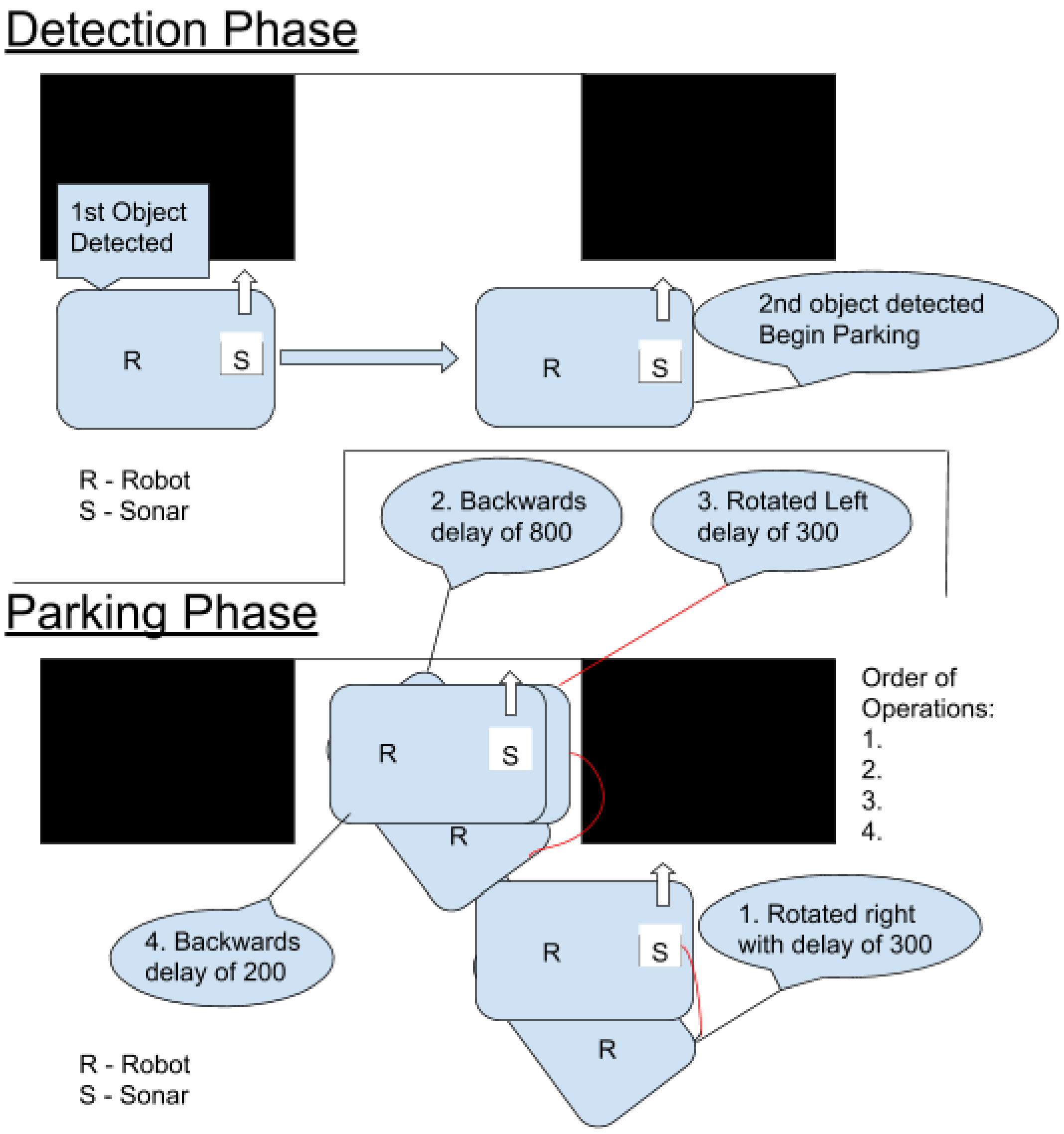
CPS 607 - Final

Enhanced Cruise Control

My strategy for Enhanced Cruise Control is to have all of the basic functions forward, backward, right, left, stop, and distance function. All of the defines are the same as the labs and the setup function is the same as well. The sonar is set to a basic position of 90 degrees meaning it is pointed straight forward as the object it would be following would be in front of it and not in a different lane. The robot was then set to forward and to take the distance, then stored in a middle distance integer variable. For staying in the lane and not going out of bounds the robot makes use of the left and right line sensors, and if the right sensor becomes true the robot is then told to make a slight left turn, or if the left sensor becomes true the robot is then told to make a slight right turn, doing this allows the robot to successfully follow and stay in the lane. If the middle distance is less than equal to 5 there is an object in front. It then enters an if statement, stopping and taking another distance measurement. There are then two if statements if the distance is less than 8 there is a stationary object in front and a lane change is needed, or if it is greater than 8 cruise control is needed. When the distance is less than 8, the robot makes use of the line sensors, when the robot is going to make a lane change it counts 2 lines and hits the third then it is in the correct lane. The robot starts in the rightmost lane and has boolean variables controlling what lane it is in and also if it has already made a lane change a boolean variable will become true telling the next if statement not to make another lane change cause one has already been made. The less than 8 if the statement has 2 if statements in the code to allow left to right lane changes, and within those, if statements it has a while true loop to count the number of lines it has crossed then it breaks once it has crossed 2 lines and hits the 3rd line indicating it is in the leftmost or rightmost lane. Then the normal code takes over. If the distance is greater than 8, it then needs to start cruise control and stay behind the object. It then enters a while true loop, still taking line sensors into account making sure it stays in the lane, it then continuously takes a distance. If the distance is less than 20 it slows the robot down and if it is greater than 21 it speeds up, but also at the same time has an if statement saying it goes too slow or too fast to set the speed much faster or slower. If the distance exceeds 200 it breaks from the loop setting the speed to 90, as the object in front would now be gone. All of these if statements allow lane changing and cruise control. also with the left and right sensor, it allows the robot to successfully stay in the lane. Every function and if statement allows the success of Enhanced Cruise Control.

Parallel Self-Parking

Parallel Self-Parking uses all of the same functions and definitions as Enhanced Cruise Control. For my loop function, first of all, set the sonar to the leftmost position, then the robot to go forward and to take a distance test and store it to a middle distance variable. The robot will start aiming at one of the objects to the left of it, this will activate an if statement saying if there is an object under 20 in distance then a boolean variable becomes true, but if the distance becomes greater than 30 then this boolean variable becomes false and a passed boolean variable becomes true to indicate it has passed the first car parked on the left. Once both of these boolean variables are true the robot is at the edge of the second parked car on the left it then begins the parking algorithm. To start the robot stops turns right at a delay of 300, Stops then goes backwards at a delay of 800. Then the robot is in the spot without hitting and cars but it is not straight but pointed to the right, to correct this I have the robot go left for a delay of 300. Once the robot is in the spot perfectly to make sure it isn't too close to the car in front it goes backwards for a delay of 200. The code is then over the robot is told to stop and the entire robot stops with the exit code of 0, ensuring that the robot will no longer move. This strategy allows my robot to successfully complete Parallel Self-Parking as shown in the drawing below.


Detection Phase
1st Object Detected
R
S
R
S
2nd object detected Begin Parking
R - Robot
S - Sonar
2. Backwards delay of 800
3. Rotated Left delay of 300
Parking Phase
R
S
R
Order of Operations:
1.
2.
3.
4.
R
S
1. Rotated right with delay of 300
4. Backwards delay of 200
R
R - Robot
S - Sonar