Final Project:

Objective: Design an algorithm that will allow the robot to continue moving forward while avoiding obstacles, and escaping corners. The idea is to just have the robot able to continuously drive around without hitting into walls or other obstacles. Use the PID\_v1 library to help the robot control it’s directional movement (forward, back, turning), use the HC\_SR04 and Servo to detect distance of objects within a cone in front of the robot (SODAR class is an option here as well), and the State class to easily facilitate motion control. Redefine the problem statement in your terms in TURN\_IN.docx

* Research: Spend some time researching online for conceptual obstacle avoidance algorithms to see what has been done in the past. Do we have sensors sufficient enough to implement some of these algorithms? Paste links to related research you have conducted, and write a short explanation of each of the links (have at least 4 sources).
* Ideation process: As a group, come up with at least 4 different conceptual algorithms. (Do not start writing them in code). Give each a name and a short description describing how it will work. Draw the software block diagram for each of the conceptual algorithms and include in the TURN\_IN.docx document.
* Decision Matrix: Using the engineering decision matrix, come up with 5 metrics to grade each of the algorithms. Grade each of the algorithms using intuition as a group and then pick the highest scoring algorithm. That is the algorithm you will start with.
* Pseudo Code: Looking at the software block diagram for the chosen algorithm, in the arduino IDE start writing “Pseudo Code”. Pseudo code is comment-based code that will not compile, but provides the designer a way to check the logical flow of the code inside the actual structures (if statements, while loops, for loops, functions, libraries etc…). Post your pseudo code in the TURN\_IN.docx. (Take a look at the end of the document to see some example Pseudo code.
* Implementation: Once completed the Pseudo code, and it seems to have all the necessary elements, start filling in the code with actual compilable code. This step will take the longest but take your time and eventually the code will compile and run on the robot. Does the robot respond as intended to obstacles? If not, it’s time to do some algorithmic debugging to see what is missing in your code, or what is something extra being done that is preventing the algorithm from working as expected. Discuss each of these problems you ran into and how you fixed them in TURN\_IN.docx
* If you have time left, take the algorithm that worked the best and run another group brainstorming session to see if there are any ways to improve performance in code. Try implementing these updates and see if the response is any better. (Remember an improvement could be as simple as slowing the robot down). Discuss your reasoning for this change and if it helped improve the obstacle avoidance capability of the robot.

Example Pseudo Code:

//global variables

//servo variable

//SODAR initialization

//distance variable array

Int distance[];

Void setup()

{

//attach and initialize servo

//initialize SODAR

//initialize motor\_setup()

}

Void loop()

{

//update SODAR

sodar.update();

//run obstacle detection function

obstacle\_detection();

//update motors based on State class

raw\_motor\_control(//stuff in here);

}

Void obstacle\_detection(){

//parse through the distance array and determine how motion should be controlled to

//avoid obstacle

//if left distance < threshold, turn right

if(dist[0] < threshold) { control\_state.setRotationState(-45); //set rotation to 45 deg/sec to the right.

}