Final Project

Navigation Competencies & Wireless Communication

Reading: *Ch. 4 of the text*
Read this entire project theory and procedure before starting.

****

Purpose: The purpose of the final project is for the student team to demonstrate the integration of several concepts learned this quarter. The primary goal is to develop several navigation competencies on the CEENBot. The first task is to use topological and metric navigation to move the mobile robot from a start point to a goal point in the world. The localization task involves using sensor feedback with an exploration routine to determine the location of a lost or kidnapped robot in the world. The final task is to create a topological or metric map of the robot's world by using sensor feedback with a coverage algorithm. In order to achieve these tasks, the student team will use wireless communication in order to send and receive path planning, localization and mapping data. The student team will also use the Pixy camera for color tracking to improve robot navigation.

Objectives: At the conclusion of this project, the student should be able to:

- Implement topological path following on a mobile robot to move the robot from a start point to a goal location
- Implement metric path planning on a mobile robot by using wave front propagation or grassfire expansion to move the robot from a start to a goal location
- Use sensor data and an exploration algorithm to localize a robot in a world given an a priori map
- Use sensor data and a coverage algorithm to create an occupancy grid or topological map of the robot's world
- Create a world model that the robot can use for navigation
- Use the computer vision or color tracking for robot navigation
- Use wireless communication to transmit data between the robot and the computer

Equipment:
Base Robot
Pixy Camera
Masking Tape
Ruler
Various Wires
Red, Green, Blue LEDs
Green, Pink, Orange, Yellow Index Cards
Wireless Controllers:
  PlayStation Controller & BlueTooth Dongle (for teleoperation)
  2 nrF24L01 Wireless Transceivers (for bidirectional communication)
  Arduino Uno Microcontroller

References:
Arduino Reference, Getting Started
https://www.arduino.cc/en/Guide/HomePage
MATLAB Support Package for Arduino
http://www.mathworks.com/help/supportpkg/arduino/examples.html
Simulink Support Package for Arduino
http://www.mathworks.com/help/supportpkg/arduinoio/examples.html

SimuLink and Arduino Getting Started Guide:
http://makerzone.mathworks.com/resources/install-support-for-arduino/?s_eid=PRP_5807
Alternative Arduino IDEs
https://learn.sparkfun.com/tutorials/alternative-arduino-interfaces
http://playground.arduino.cc/Code/Eclipse
AccelStepper library
https://learn.adafruit.com/adafruit-motor-shield/downloads?view=all
Pixy Camera to Arduino
http://cmucam.org/projects/cmucam5/wiki/Hooking_up_Pixy_to_a_Microcontroller_(like_an_Arduino)

****

THEORY

****

Topological Path Following

Topological path following is based upon landmarks in the world. A *distinctive place* is a landmark where the robot can make a navigation decision. Typically, the robot will use one behavior to move in the world and then change when it gets close or in the neighborhood of the distinctive place. For the purpose of this project, the distinctive places in the world will be hallways, corners, t-junctions, and dead ends. The student team will design behaviors and perceptual schema for identifying gateways in an artificial environment. The robot will be given a list of navigation commands based upon the world's topology and use a parsing routine and a sequencer to move the robot from a start point to a goal point. For example, if the robot is given "SLRT" (S = **S**tart, L = go **L**eft, R = go **R**ight, T = **T**erminate) then it would start, follow hallway, turn left at the next gateway, turn right at the next gateway, then stop at the last gateway. The robot should continue to move forward until a gateway is encountered or until the stop command is encountered. Figure 1 is an example of topological navigation using the command "SLRT".

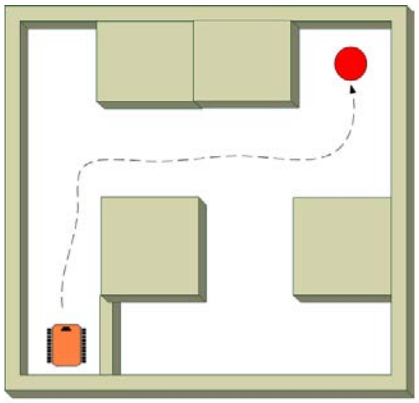
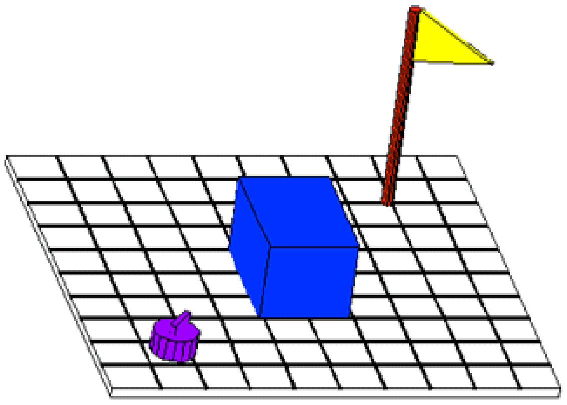


Figure 1: Topological Navigation ("SLRT")

Metric Map Path Planning and Execution

In this project, you will use a wavefront algorithm (or grassfire expansion) on an a priori map to create a path from the robot's start position to goal location. Use the wall following, obstacle avoidance and move to goal behaviors to move through the list of goal points until the robot arrives at the final destination. If you assume the algorithm uses an eight-neighborhood, the robot can move diagonally, otherwise a four-neighborhood would be used which may have less odometry error. The configuration space will be an occupancy grid divided into 18" x 18" squares, where free space is represented by 0's and occupied space by 99's. You should devise a scheme to represent the robot's start position and goal position. Your code should be flexible such that these values can be specified at run time. Figure 2 is an example of the world representation.



a. Real world

0	0	0	0	0	0	0	0
0	0	0	0	0	0	G	0
0	0	0	0	0	0	0	0
0	0	0	**99**	**99**	**99**	0	0
0	0	0	**99**	**99**	**99**	0	0
0	0	0	**99**	**99**	**99**	0	0
0	0	0	0	0	0	0	0
0	S	0	0	0	0	0	0

b. Configuration Space (8 x 8 matrix)

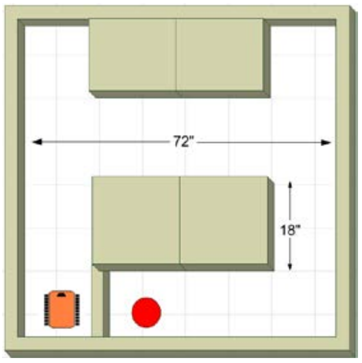
Figure 2: World Representation

The wavefront is created by starting at the destination and creating eight connected neighbors back to the start point. The numbers represent the number of steps to the goal point. The robot would then follow the numbers in the reverse order to arrive at the goal point (see Figure 3). The goal is for the robot to always move such that the steps to the goal position are reduced. This path plan has 9 steps from start to finish. Note that you may need to grow the obstacles by the robot's width or radius to avoid clipping them.

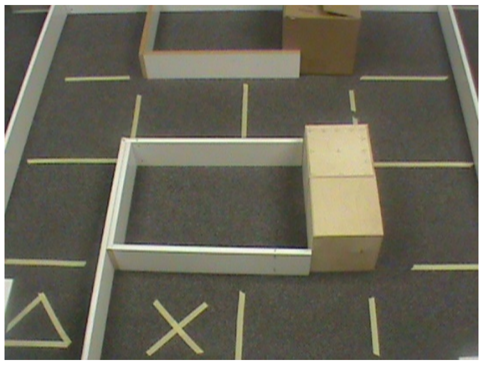
The test arena for the project demonstration will be 6 ft x 6 ft with 18" x 18" obstacles. This artificial world will be a 4 x 4 grid where the robot's start point is denoted by a '∆' and the goal point is marked by an "X". Figure 4 shows a sample test arena.

7	6	5	4	3	2	1	1	1
7	6	5	4	3	2	1	**0**	1
7	6	5	4	3	2	1	1	1
7	6	5	**99**	**99**	**99**	2	2	2
7	6	6	**99**	**99**	**99**	3	3	3
7	7	7	**99**	**99**	**99**	4	4	4
8	8	8	7	6	5	5	5	5
9	**9**	8	7	6	6	6	6	6
10	9	8	7	7	7	7	7	7

Figure 3: World Representation Wavefront



a. Artificial world



b. Real world

Figure 4: Sample Test Arena

Instead of representing the world map as an occupancy grid, it can be based upon the topology of the space. The salient features of the space are walls, hallways, corners and junctions. Each square will be represented by an integer between 0 and 15, dependent upon where walls are present around the square. The north (0001), east (0010), south (0100) and west (1000) walls represent one bit of that integer (see Table 1). Using the coding in Table 1, the maze shown in Figure 3 is represented by an 8 x 8 matrix of integers shown in Figure 5. Using the coding in Table 1, the maze shown in Figure 6 is represented by an 11 x 10 matrix of integers.

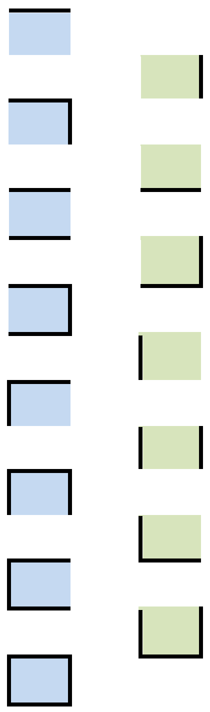
To use the topological map to plan a path from a robot start location to a goal point it is possible to use the wavefront algorithm again. However, instead of the robot moving to cells on the occupancy grid, the robot will use behaviors and rules such as move forward, turn left, follow wall, follow hallway, avoid obstacle, etc. The key difference between this path following and the first one described is there are walls in the world as opposed to occupied or empty cells. This navigation involves taking the list of actions and executing them.

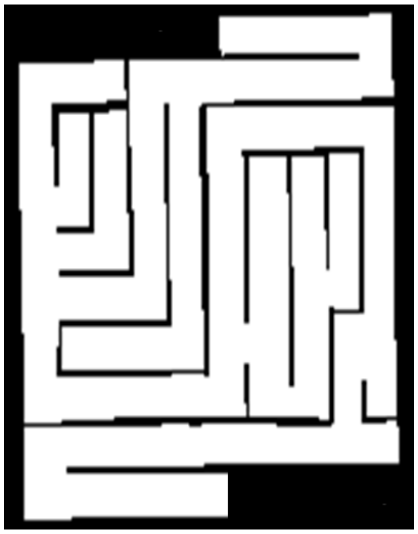
9	3	15	15	15	15	9	3
8	2	15	15	15	15	8	2
8	0	1	1	1	1	0	2
8	0	4	4	4	4	0	2
8	2	15	15	15	15	8	2
8	2	15	15	15	15	8	2
8	2	9	1	1	1	0	2
12	6	12	4	4	4	4	6

Figure 5: Sample Test Arena Topological Map

Table 1: Topological map coding

Integer	**Binary**	**Hexadecimal**	**Direction**	**Wall Location**
0	0000	0		
1	0001	1	North	
2	0010	2	East	
3	0011	3		
4	0100	4	South	
5	0101	5		
6	0110	6		
7	0111	7		
8	1000	8	West	
9	1001	9		
10	1010	a		
11	1011	b		
12	1100	c		
13	1101	d		
14	1110	e		
15	1111	f		





15	15	15	15	15	13	5	5	5	3
9	5	7	9	1	5	5	5	5	6
10	11	11	10	10	9	5	5	5	3
10	10	10	10	10	10	11	11	11	10
8	6	10	10	10	10	10	10	10	10
8	5	6	10	10	10	10	10	10	10
8	5	5	6	10	10	10	10	6	10
10	13	5	5	6	8	2	10	9	2
12	5	5	5	5	6	12	6	10	14
9	5	5	5	5	5	5	5	4	7
12	5	5	5	5	15	15	15	15	15

Figure 6: Maze Topological Map

Localization

Localization is the process a robot uses to determine its location in a world given an a priori map and sensor readings. A localization algorithm should process the map to identify key features such as the gateways or distinctive places [corners (C), hallways (H), dead-ends (D), t-junctions (T)]. Although it is a not a gateway, a series of 10's or 5's or neighboring 1's and 4's or 8's and 2's indicate a hallway (H) in the world. The numbers (1, 2, 3) indicate a gateway where the robot can make a navigation decision. The distinctive features or gateways are the nodes in the world and the hallways can be used with a local control strategy such as follow center or follow wall to move between nodes. An example of coding a map for topological path following is shown in Figure 7.

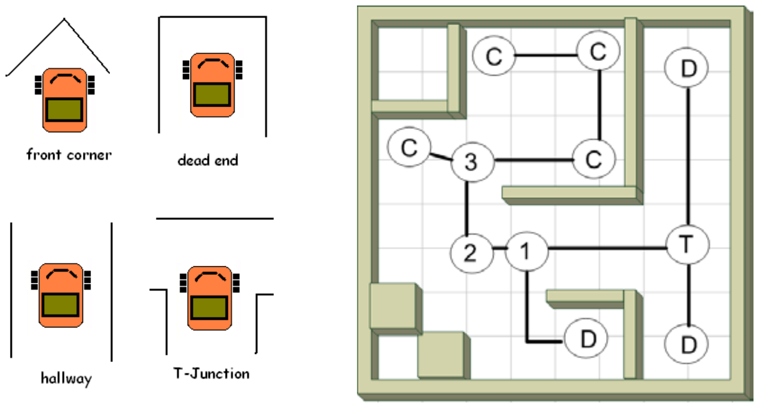


Figure 7: Feature Extraction (Map Coding)

Once the robot is placed in the world, the robot should then use some motion algorithm such as random wander, follow center, or wall following to explore the world and identify gateways. It should keep track of the gateways passed and the order in which they were passed. Although there will be some odometry error, it would also aid the localization algorithm to keep track of odometry such as distance traveled and turns. Within three to four iterations of this process, the robot should be able to use a probabilistic method such as the Partially Observable Markov Decision Process (PMDP) to localize itself. Note that if the robot also uses directional information such as it starts out facing north in in the world, it can reduce the number of steps required to localize. Figure 8 provides an example of this localization process with the proposed robot locations after each step marked on the map.

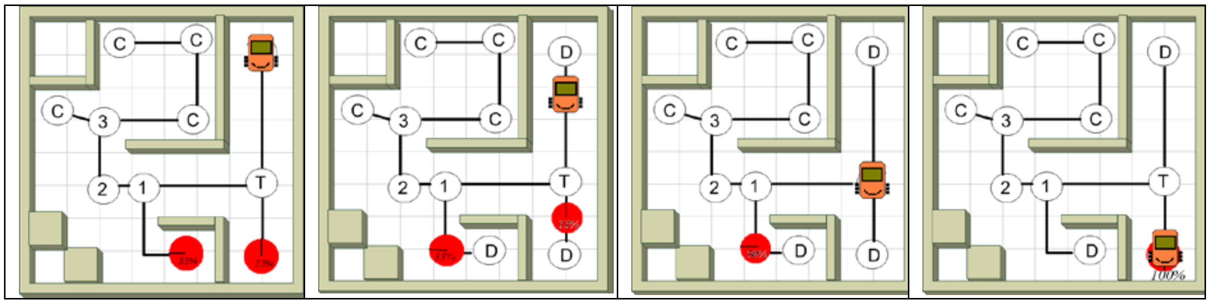


Figure 8: PMDP Localization

Mapping

Mapping is the process that a robot uses with sensor feedback to create a map of an unknown environment. It is possible to use Histogrammic in motion mapping (HIMM) to identify objects in the environment. Although your textbook states that this algorithm was created for a sonar sensor, it is reasonable to use it for the primary axis of the IR sensor. For this method when the cell value exceeds some threshold it is coded as occupied or used to create a topological map. Alternately, you can create an occupancy grid with 0's and 99's based upon the free and occupied space. Figure 9 provides an example of using a Generalized Voronoi Graph (GVG) and HIMM to create a world map.

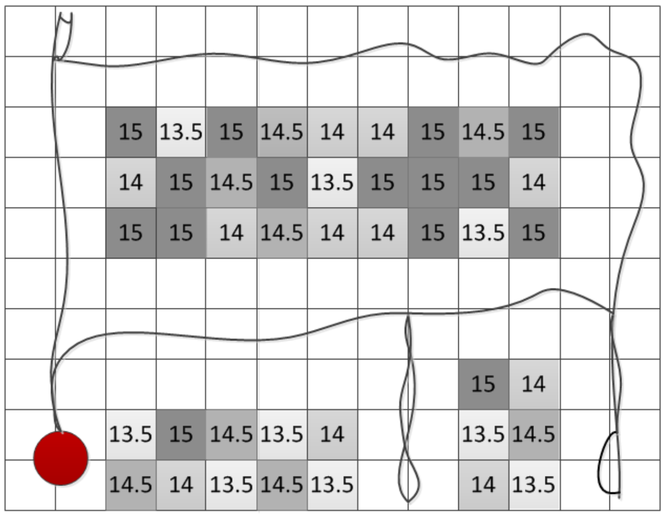


Figure 9: GVG with HIMM

****

PROCEDURE

****

PART A – PATH PLANNING & FOLLOWING

Part I - Topological Navigation and Execution

1. The student team should place the robot in several corners and intersections of hallways in the artificial environment and determine the perceptual schema to identify these gateways and distinctive places.
2. It would be advisable to use a combination of the IR and sonar for sensor redundancy to identify these locations in the world. There will be some sensor error, this is a standard problem in mobile robot navigation and the program should be designed in such a way to minimize the effect of the error. For example, average 5 or 10 readings and filter out spurious readings. Figure 13 provides descriptions of the possible world landmarks.
3. Include a table similar to Table 2 that includes the test data and perceptual schema for each of the landmarks. You should have a minimum of 4 sensors around the perimeter of the robot in order to get sufficient data for mapping and localization. You may also have a combination of IR and sonar.
4. Program the robot to identify the gateways and make navigation decisions such as follow hallway, turn left or turn right based upon a topological path plan.
5. Finally, input the robot a command such as "SRLT" in your program and place it in the world so that it can execute the path. Use the LEDS to indicate when the robot arrives at the destination.

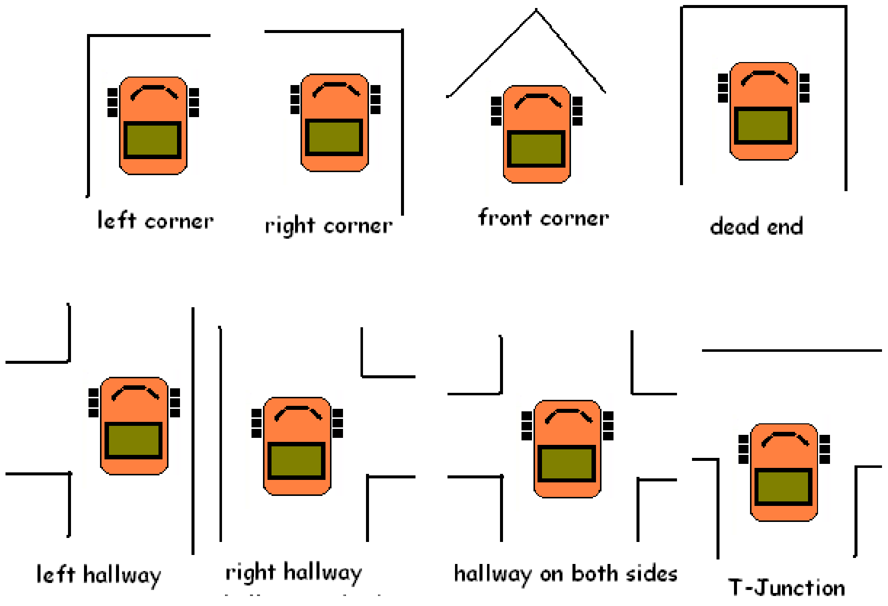


Figure 13: Distinctive Places

6. In the demonstration, you will be required to use this method so that the robot follows several different paths.

Table 2: Perceptual Schema Data Table

Sensor-> Landmark	**Front sensor**	**Left sensor**	**Right sensor**	**Back sensor**	**Front Left**	**Front Right**	**Back Left**	**Back Right**
Corner in front								
Corner on left								
Corner on right								
Hallway on both sides								
Hallway on left								
Hallway on right								
T-junction								
Dead End								

Part II - Metric Path Planning and Execution

1. Download the world map from the Resources project folder in Moodle. The world map is a text file that includes a 4 x 4 array of 0's and 99's that represents free space and obstacles (occupancy grid) or a 4 x 4 array of numbers 0 through 15 that represents world features (topological map). You must devise a method to encode the a priori map into your program. See Figures 2 and 3 for examples of occupancy grids and see Figures 5 and 6 for examples of topological maps. You should also display the map on the serial monitor to confirm that it matches the real world.

2. At the beginning of the demonstration, you will be given the robot's start position and goal position. Input the start and goal position into your code.

3. The robot should then plan a path from the start position to goal position by using the wavefront algorithm. The serial monitor should display the path planned by the robot. One suggestion for doing this is to show a list of cells that the robot will traverse from the start point to goal point. You can devise a way to indicate this information by using LEDs.

4. You should then place your robot at the start position and it should move to the goal point. You can display the robot's progress by using LEDs to indicate gateways and when the robot has completed the path.

5. Your algorithm should include a combination of odometry and reactive behaviors to avoid hitting the walls while executing the planned path. *(Note that one technique to prevent the robot from hitting walls and obstacles is to select the path that maximizes the distance between walls and obstacles or follow the center line (i.e. Voronoi diagram, follow hallway).)*

6. In the demonstration, you will be required to show that the robot can follow various distinct paths. You will be graded on how well your algorithm works; the efficiency of the path chosen by the robot, the ability of the robot to reach the goal point while also avoiding obstacles.

PART B – WIRELESS COMMUNICATION

The next step in the project is to establish wireless communication between the robot and your laptop or a game controller. Wireless communication can be used to send and receive data from the robot on the serial monitor. It will also be used to command the robot to drive remotely (teleoperation). Recall that teleoperation is the most basic level of robot control.

Part I – Attach Wireless Transceivers

1. The nrF24L01 wireless transceiver uses SPI communication and must use pins 50 (MISO), 51 (MOSI), and 52 (SCK) on the Arduino Mega 2560. The following link provides detail on the microcontroller pin mapping: https://www.arduino.cc/en/Hacking/PinMapping2560
2. On the robot microcontroller, move the wire on pin 50 to pin 44.
3. On the robot microcontroller, move the wire on pin 52 to pin 46.
4. On the robot microcontroller, move the wire on pin 51 to pin 49.
5. Note that you will need to change these pin numbers to make your robot move.

```
//define stepper motor pin numbers
const int rtStepPin = 44; //right stepper motor step pin (was 50)
const int rtDirPin = 49;  // right stepper motor direction pin (was 51)
const int ltStepPin = 46; //left stepper motor step pin (was 52)
const int ltDirPin = 53;  //left stepper motor direction pin
```

6. Figure 14 shows the wiring diagram to attach the wireless transceiver to the Arduino Mega 2560 on top of the robot.
7. You will connect an Arduino Uno microcontroller is connected to your laptop as a transmitter and/or receiver. The nrF24L01 wireless transceiver uses SPI communication and must use pins 12 (MISO), 11 (MOSI), and 13 (SCK) on the Arduino UNO. The following link provides detail on the microcontroller pin mapping: https://playground.arduino.cc/Learning/Pins

http://www.theengineeringprojects.com/2015/07/interfacing-arduino-nrf24l01.html

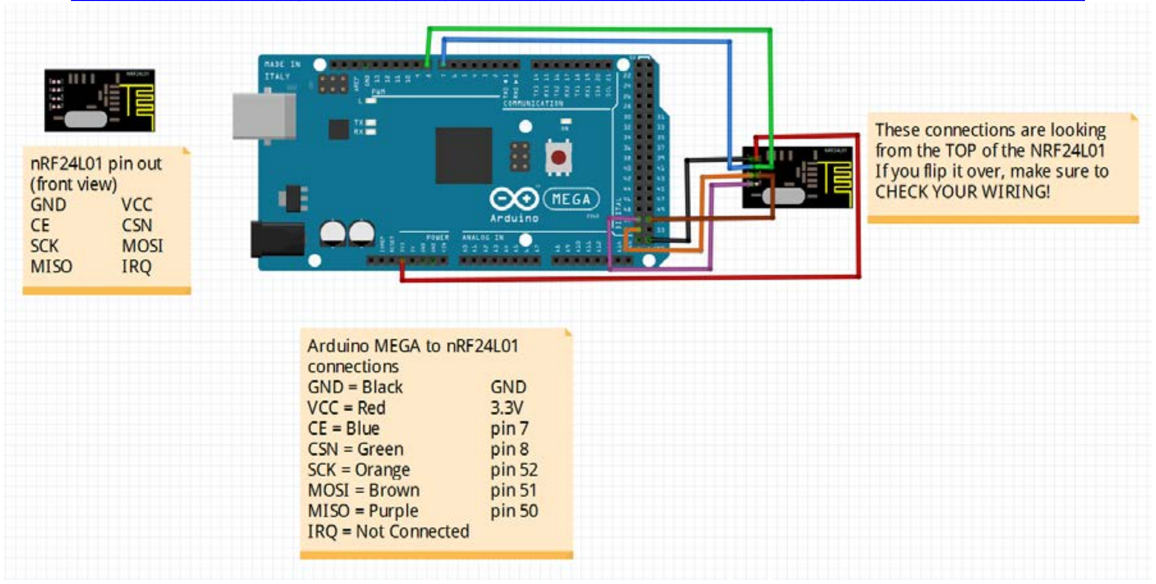

nRF24L01 pin out (front view)
GND VCC
CE CSN
SCK MOSI
MISO IRQ
These connections are looking from the TOP of the NRF24L01 If you flip it over, make sure to CHECK YOUR WIRING!
Arduino MEGA to nRF24L01 connections
GND = Black GND
VCC = Red 3.3V
CE = Blue pin 7
CSN = Green pin 8
SCK = Orange pin 52
MOSI = Brown pin 51
MISO = Purple pin 50
IRQ = Not Connected

Figure 14: Connect nRF24L01 wireless transceiver to Arduino MEGA 2560 on robot

8. Figure 15 shows the diagram to connect the transceiver to the Arduino Uno microcontroller attached to your laptop.

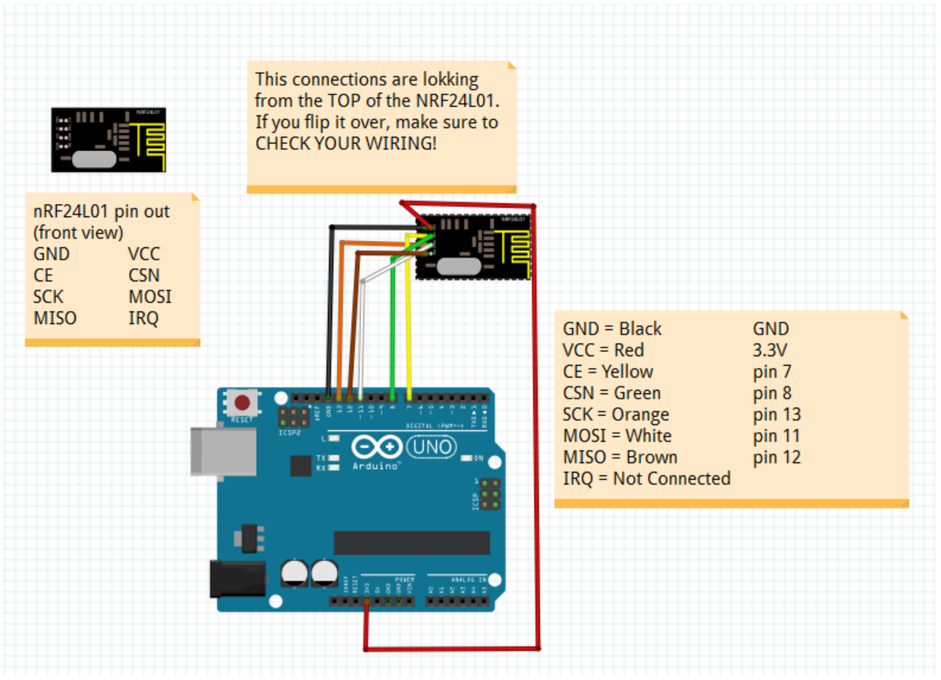


Figure 15: Connect nRF24L01 wireless transceiver to Arduino UNO on laptop

9. Note that your code must have the following for the transmitter and the receiver:

```
#include <SPI.h>      //include serial peripheral interface library
#include <RF24.h>     //include wireless transceiver library
#include <nRF24L01.h> //include wireless transceiver library
#define CE_PIN   7
#define CSN_PIN 8
RF24 radio(CE_PIN, CSN_PIN);
```

Part II – Test Connections

1. Install the *RF24* library available in the Arduino IDE by using the following directions: https://www.arduino.cc/en/Guide/Libraries#toc2
2. Download *“RobotReceiver.ino”* from the project zip file. Verify and upload this program to the microcontroller mounted on the robot. We may eventually need to modify this code to change radio.setChannel() to reduce interference. This means selecting any value between 1 and 125 that does not match another teams. You may also want to change pipes by selecting a new 5 Byte number.
3. Keep the robot connected, open up the serial monitor for the COM port where the receiver robot microcontroller is connected to your laptop. You can do this by clicking the magnifying glass in the upper right corner of the Arduino IDE for the *RobotReceiver.ino* program.
4. Download *“RobotTransmitter.ino”* from the project zip file. Open up a second instance of the Arduino IDE and set it to the microcontroller connected to the laptop COM port. Verify and upload this program to the microcontroller connected to your laptop. We may eventually need to modify this code to change radio.setChannel() to reduce interference. This means selecting any number between 1 and 125 that does not match another teams. You may also want to change pipes by selecting a new 5 Byte number.
5. Open up the serial monitor for the COM port where the transmitter microcontroller is connected to your laptop. You can do this by clicking the magnifying glass in the corner of the Arduino IDE for the the *RobotTransmitter.ino* program.
6. What you should observe is that as you type “1” and click ***Send*** in the transmitter serial monitor, the “1” should print on the receiver serial monitor. This shows that the wireless communication is working.
7. Also as you type “1” and click ***Send*** in the transmitter serial monitor, the LED on pin 13 of the robot microcontroller should turn **ON**. If you type “0” and click ***Send*** the LED on pin 13 of the robot microcontroller should then turn **OFF**.
8. Read all the code and comments for the transmitter and receiver to understand how it works in order to modify it in the next part. You may want to also download and read *“Robot-BiWireless ReadMe.txt”* and *“Robot-BiWireless”* code and comments to understand how that option works.
9. *Question: Can you figure out how to make the robot a transmitter and the laptop microcontroller a receiver? Can you explain and describe how you did this in your lab report method section? What information did the robot send? What information did the laptop receive?*

Part III – Keyboard Drive

1. Rename the *“RobotReceiver.ino”* program *to “YourRobotName-KeyDrive.ino”*
2. Modify your code to use the number pad to drive the robot remotely. Since there are 9 keys, you can program at most 9 different functions. It is your choice which ones to use. View some serial communication examples on the Arduino IDE by clicking File→Examples→04.Communication.
3. Alternately, you can modify the code to use ASCII characters to drive the robot instead of integers. If you would like to try that option, view the programs at File→Examples→04.Communication.
4. There is also more serial communication help at the following links:

   https://www.safaribooksonline.com/library/view/arduino-cookbook/9781449399368/ch04.html

   http://www.instructables.com/id/Controlling-Servo-motor-using-Keyboard-input/

https://forum.arduino.cc/index.php?topic=415263.0

5. *Questions: what method did you use to create bidirectional communication between your robot and computer? Can you describe it in detail in the methods section of the final report?*

Part IV – PlayStation Control (Teleoperation)

1. Install the *PS2X* library from the project zip folder by following the directions at the following link: https://www.arduino.cc/en/Guide/Libraries#toc2

2. The pin out for the PlayStation Controller Bluetooth dongle are shown in Figure 16. Wire the PlayStation dongle to the microcontroller on your robot by using the connections in Table 3. The video at the following link may be helpful to understand the wiring and code https://youtu.be/xlupRVF_6W8

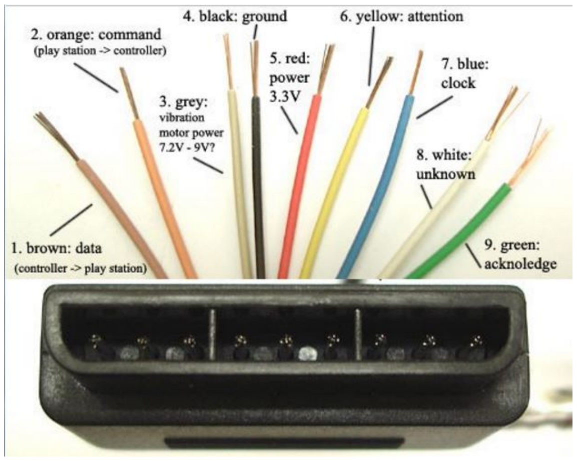


Figure 16: PlayStation Controller Bluetooth Dongle

Table 3: PlayStation Controller and Arduino MEGA 2560

1	Data	Empty row on breadboard (i.e. 45J) Pin 2 to row on breadboard (i.e. 45F) 10 kΩ resistor from 5V to row on breadboard (i.e. 45H)
2	Command	Pin 3
3	Vibration	5V or not connected if vibration not needed
4	Ground	GND
5	Power	3.3V
6	Attention/Select	Pin 4
7	Clock	Pin 5
8	Unknown	Not Connected
9	Acknowledge	Not Connected

3. Download *"RobotPlayStationDrive.ino"* to your computer from the project zip folder. Verify and upload the code to the microcontroller on the robot.

4. If the hardware and software are working correctly, you should see the buttons pressed on the serial monitor. If you press the select and start buttons on the controller, you should see the LED on pin 13 on the microcontroller turn on and off. If this part is not working correctly, debug and troubleshoot to resolve the problem. Check the following website for help troubleshooting:

http://www.billporter.info/2011/03/27/arduino-playstation-2-controller-library-troubleshooting-guide/

5. Rename the *"RobotPlayStationDrive.ino" to "YourRobotName-PlayDrive.ino"* .
6. Modify the code so that you can use the PlayStation controller to drive the robot using the motion functions you created.

PART C – LOCALIZATION AND MAPPING

Part I - Localization

1. The user will provide the robot with an a priori map as a matrix to input into your code. The map will be a 4 x 4 array with topological map or occupancy grid encoding.
2. The localization algorithm should process this map to identify key features such as the gateways or distinctive places.
3. The first test of your algorithm will involve using teleoperation to drive your robot through the world until it localizes. Where the robot believes it is in the world should be communicated to the user by using blinks on LEDS or display on the laptop serial monitor using wireless communication.
4. Once the algorithm is working, place the robot in the world. The robot should then use some motion algorithm such as random wander, follow center, or wall following to explore the world and identify gateways. It should keep track of the gateways passed and the order in which they were passed and use it to localize itself. The robot should then send its location to the user by using LEDs or display on the serial monitor.
5. Lastly, after determining its location using registration, the robot should use wavefront propagation to plan a path from its current location to the goal location (or home). The robot's goal will be specified at run time, this information can be transmitted wirelessly to the robot. After localization, you can use prior code to plan a path for the robot.

Part II - Mapping

1. Place the robot in the artificial world and use teleoperation to create a complete occupancy grip or topological map. Use wireless communication, to display the robot's progress and the complete map to the user.

2. Once the initial mapping algorithm is working, place the robot in the artificial world and use motion behaviors to create an occupancy grid or topological map. Use wireless communication, to display the robot's progress and the complete map to the user. You can use LEDs to indicate robot state.

3. Lastly, after creating the map, the robot should localize itself in the map and plan a path to return the robot to its' home or start position. This information will be transmitted wirelessly to the robot. After localization, you can use prior code to plan a path for the robot.

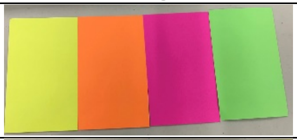
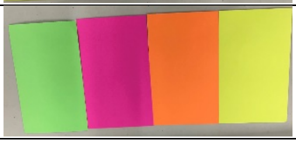
Extra Credit – Path Planning, Mapping and Feedback with Computer Vision

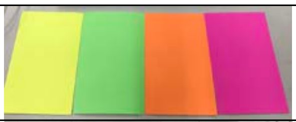
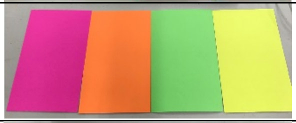
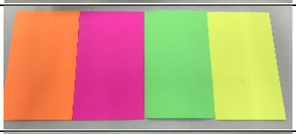
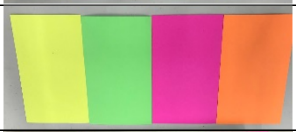
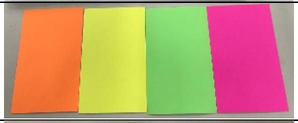
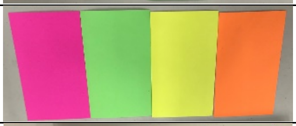
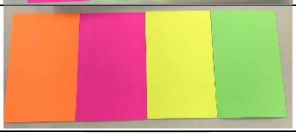
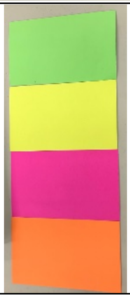
This is the first year we are using the Pixy camera and we may need to work out some bugs so completing navigation tasks with computer vision will be extra credit. No, I don't know how much yet! ☹ In fact, the points may be based upon an evaluation of the level of difficulty of what you implement.

Part EI – Read Color Signatures

1. Review the Pixy Camera documentation available at the links in the reference section. Download the PixyMon software.
2. You will use color codes for navigation by using the green, pink, orange and yellow index cards. A color code is 2 or more colors which we will use to represent cells or distinctive places in an artificial world or world map. You can use the x center, y center, width, height, and angle of the object to distinguish between distinctive places in the artificial world.
3. You can make distinct color code signatures by changing the order and orientation. Table 4 provides some examples of signatures (see more details at http://cmucam.org/projects/cmucam5/wiki/Using_Color_Codes
4. Mount the Pixy camera to your robot by using Velcro as shown in Figure 17. Make sure you only power from the USB or the battery pack, it won't work if you use both!
5. Connect the Pixy camera to the Arduino Mega 2560 microcontroller by using Figure 18. Pay attention because the connector only works if connected in the correct direction.
6. Use PixyMon to create color code signatures and to view the detected signatures.
7. Download the *Pixy.h* library from the CMUcam website and install the zip file into the Arduino software. Use the following example program, File->Example->Pixy->hello_world to print out color code signatures to the serial monitor. This will help to understand the data that you can use for navigation.

Table 4: Color Codes and Image Signatures

Image	Signature	Alpha
	1234	0°
	1234	180°

	1423	0°
	3214	180°
	1432	180°
	1432	0°
	2143	0°
	2143	180°
	2314	0°
	2314	90°

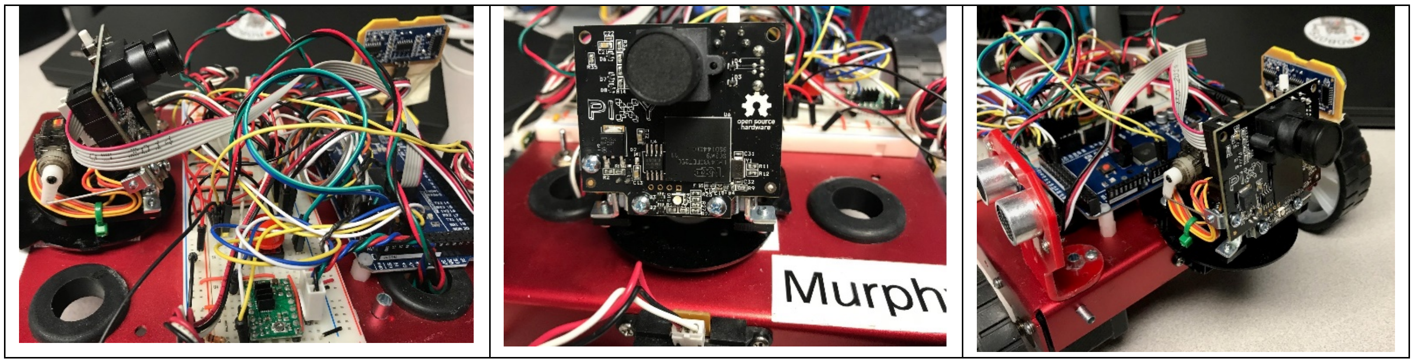


Figure 17: Mounting Pixy Camera to the robot

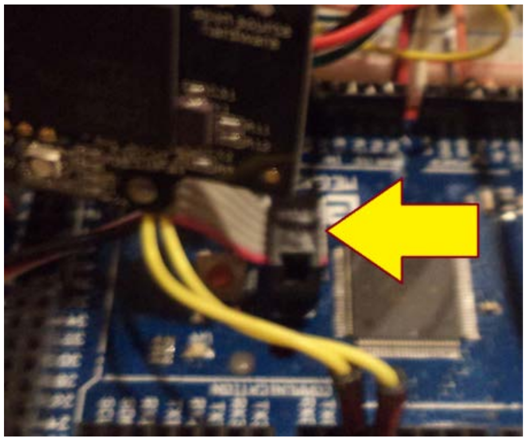


Figure 18: Connecting Pixy camera to the microcontroller (BE CAREFUL! Direction Matters!!)

Part E2 – Set up your world

1. Use painter's tape to place color codes on the walls or floor in the cells of the world. All distinctive places should have the same color code, for example if you have 4 dead ends they would all have the same signature as shown in Figure 19.

2. Determine a method to encode the world into a 4 x 4 matrix that you can use for the path planning and localization tasks.

Part E3 – Navigation using computer vision

Write a program so the robot can use topological path planning with computer vision to move from a start position to goal position. The path will have to be supplied in terms of the color codes. These color codes could be on the walls or on the floor of the artificial world.

Part E4 – Localization using computer vision

Write a program so the robot can use computer vision to localize with the color codes and then move to a given goal position.

Part E5 – Mapping using computer vision

Write a program so the robot can create a topological map by using computer vision to build a world map with the color code. Then the robot uses the created map to move from a start position to goal position.

Figure 20 shows an example of raw data from the Pixymon using color code signatures that the robot can use for path planning, localization and mapping.

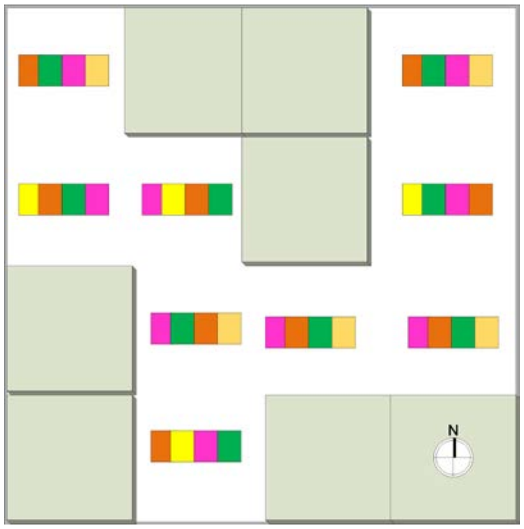


Figure 19: **Artificial World with Color Codes**

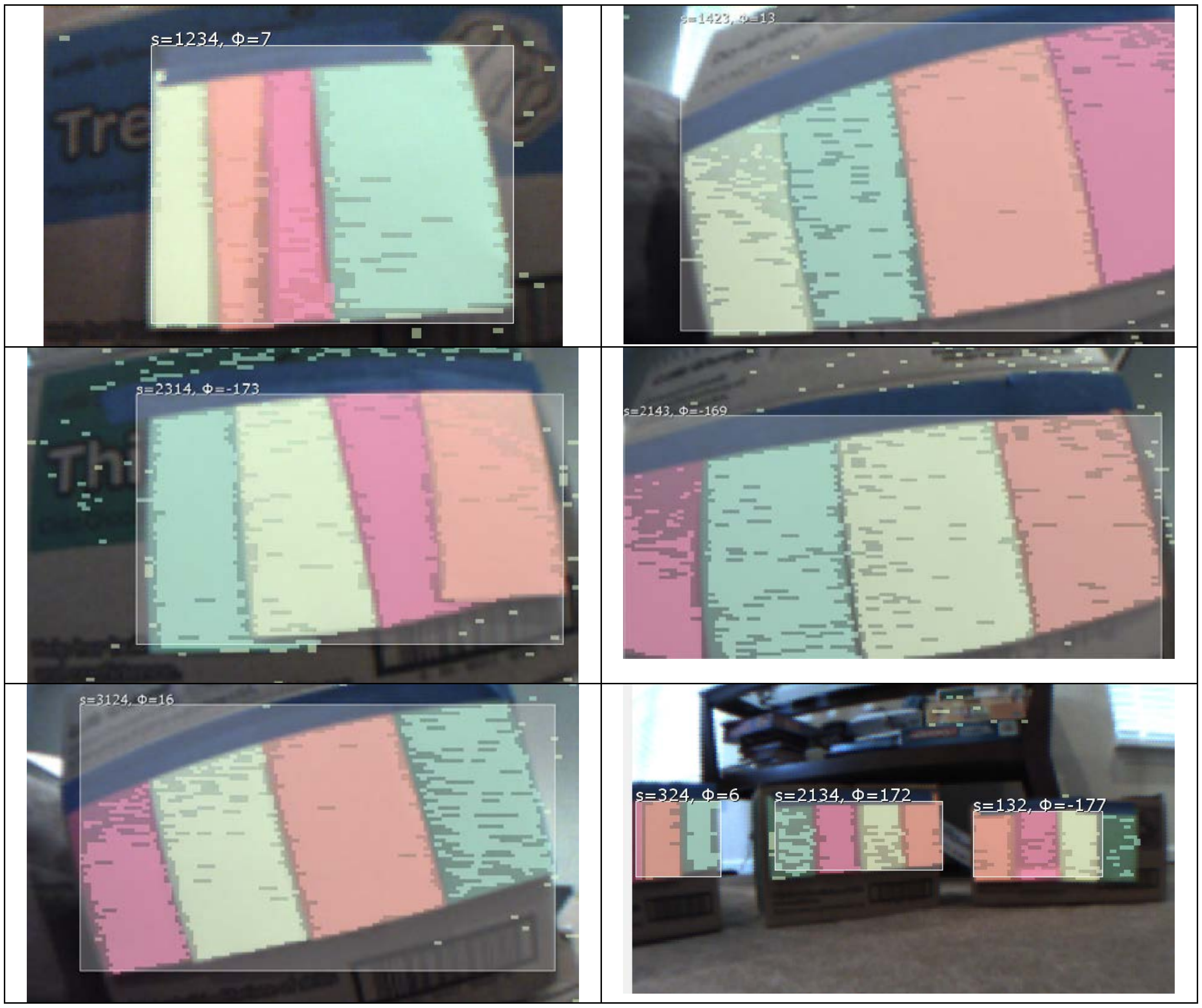


Figure 20: **Color Codes Raw Data**

SUBMISSION REQUIREMENTS

Final Project Code:

The final project code should be properly commented and modular with each new behavior representing a new function call. Recall that properly commented code has a header with the solution name, team members' names, description of the functionality and key functions, revision dates. In addition, all of the key variables and functions in the code are commented. The design of the architecture should be evident from the program layout. You should also include the design of the architecture in the appendix of the report and reference it in the text. You must submit your properly commented by midnight on **Sunday of Finals week**.

Final Project Demonstrations:

Due to the complexity of this project and the fact that it requires an integration of several concepts, there will be **five** demonstrations required on a graduated grading scale. You must have all parts of the final project demonstrated by **the last day of class**. The demonstration due dates are given in Table 5.

Final Project Report

Please use the following checklist to insure that your final project report meets the minimum guidelines. You should also view the sample final project reports available on the Moodle course site. You must submit your final project report by midnight on **Sunday of Finals week**.

Project Report Guidelines

1. The document should have default Word settings with respect to font and margins
2. All pages should be numbered
3. All headings must be numbered, left-justified, bolded, and capitalized at the beginning of the section.
4. All figures must have a number and caption underneath (i.e. GUI screenshots)
5. All tables must have a number and title above it (i.e. results error analysis)
6. The report should order should be:
   - Cover page
   - Abstract
   - Table of Contents
   - I. Objective
   - II. Theory
   - III. Methods
   - IV. Results
   - V. Conclusions and Recommendations
   - References
   - Appendix/Supplementary Materials

7. The ***cover page*** should include the school, course number, course title, team member names, robot name, project title, and date submitted.
8. The ***abstract*** should be a brief statement of the experiment purpose, verification and relevant results. The abstract should be on a separate page after the cover page and before the Table of Contents.
9. The ***table of contents*** should be on a separate page after the abstract and should have page numbers.
10. The ***objective*** should state the purpose of the project and associated tasks in your own words and should be detailed for the reader to understand what the robot must accomplish.
11. The ***theory*** should state relevant theory that will be used to achieve the objective. You must cite relevant theory from the text and other sources and there must be citations and references.
12. The ***methods*** section should summarize the algorithms implemented to achieve the purpose and the procedures used to test the robot algorithms. The method must include a control architecture, flowchart, pseudocode, state diagrams for implementing each part of the objective.
13. The ***results*** section should summarize the results of the tests and verify that the robot was able to achieve the purpose and meet the project objectives. The results should also address the results of any parts that were unsuccessful.
14. The ***conclusions and recommendations*** should address whether the purpose was achieved, possible sources of error, recommendations to improve the robot algorithm and answer any relevant questions related to the project.
15. The ***references*** should include a list of all of the works cited with proper APA or MLA format. Use the Purdue OWL website for more help on proper formatting: https://owl.english.purdue.edu/owl/
16. The ***appendix*** should include all relevant graphics or content that is too dense for the body of the report including flowcharts, control architectures, state diagrams, or pseudocode.
17. Remember this is only a guide for the minimum requirements of your report. You are required to answer any questions or provide any details that you feel aid the reader in understanding the objective, theory, procedure, implementation and results of your project.
18. You should review the sample reports on Moodle for help with the proper format for the document.

Questions to Answer in the Final Report:

1. Were there any issues with the wireless communication? How could you resolve them? If at all.
2. What does the state machine, subsumption architecture, flowchart, or pseudocode look like for the path planning, localization, and mapping? (It should be in the appendix of the report).
3. How would you implement SLAM on the CEENBot given what you have learned about navigation competencies after completing the final project? If you research solutions, make sure you cite and list references in APA or MLA format.
4. What was the strategy for implementing the wavefront algorithm?
5. Were there any points during the navigation when the robot got stuck? If so, how did you extract the robot from that situation?
6. How long did it take for the robot to move from the start position to the goal?
7. What type of algorithm did you use to selection the most optimal or efficient path?
8. How did you represent the robot's start and goal position at run time?
9. Do you have any recommendations for improving that robot's navigation or wavefront algorithm?

10. How did you use the serial monitor and bi-directional wireless communication to represent the map?
11. What type of map did you create and why?
12. What was key in the integration of the localization, mapping, and path planning?

Grading

The final project due dates and grade points are shown in Table 5.

Table 5: Project Grade Point Distribution

Task	Points	Due by
Demo 1 – Topological Path Following	10	Second Class of Week 8
Demo 2 – Metric Path Planning	10	First Class of Week 9
Demo 3 – Wireless Communication	10	Second Class of Week 9
Demo 4 – Mapping and Path Planning	10	First Class of Week 10
Demo 5 –Localization and Path Planning	10	Second Class of Week 10
Code	20	Sunday of Finals Week
Report	30	Sunday of Finals Week