Final Project

Navigation Competencies

Read this entire project before starting to work on the final project. Please make sure to ask questions if anything is unclear.

Reading and Review:

Ch. 4 of the text and Weeks 1 through 7 lectures

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 robot. To achieve the navigation tasks, the student team will use wireless communication and a graphical user interface to communicate data between the robot and team laptop.

Objectives:

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

- Use Bluetooth communication to send and receive data to the robot from the computer
- User serial communication to transmit data between Arduino and a GUI (Human-Robot Interface)
- Use sensor feedback with an IMU, encoders and/or motor steps to estimate the robot's pose in the world
- 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 given a robot's known pose
- (optional) Implement simultaneous localization and mapping (SLAM) to locate a beacon in an unknown world and then escape from the world when a wall suddenly disappears.

Equipment:

- Base Robot
- Masking Tape
- Ruler
- Various Wires
- Various LEDs
- HC05 Bluetooth Module
- IMU (Inertial Measurement Unit) - MPU6050 Module 3 Axis Analog Gyro + 3 Axis Accelerometer Module

References:

- 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
- Processing Resources
  https://processing.org/reference/
- HC05 Bluetooth Module tutorial
  https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/
- MPU6050 IMU tutorial
  https://maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor

****

THEORY

****

Navigation in mobile robotics deals with several questions that you will answer in completion of the final project. These questions are:

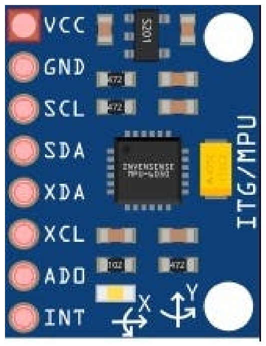
- Where am I going? (Path Planning)
- How do I get there? (Path Following)

- Where am I? (Localization)
- Where have I been? (Mapping)

Peripherals

Inertial Measurement Unit

The IMU takes inertial measurements with six degrees of freedom. It can measure acceleration, inertia to determine the robot's spatial position and velocity. It has a 3-axis accelerometer and a 3-axis gyroscope. It can sense the changes in velocity in the x, y, and z axis. It can also measure the angular momentum or rotation around the x, y, and z axis. Figure 1 shows the pin out for the IMU.



VCC - 3.3V DC power supply
GND - Ground
SCL - Serial Clock
SDA - Serial Data
AD0 - I2C Address bit. Allows you to change the internal I2C address of the MPU-6050 module. It can be used if the module is conflicting with another I2C device, of if you wish to use two MPU-6050s on the same I2C bus.
INT - Interrupt output
XDA - Auxiliary Serial Data (Used when another sensor is connected to this module)
XCL - Auxiliary Serial Clock (Used when another sensor is connected to this module)

Figure 1: MOU-6050 IMU Module pinout

To use the IMU, it will be necessary to download the MPU-6050 library. Figure 2 shows how to wire the IMU to the Arduino.

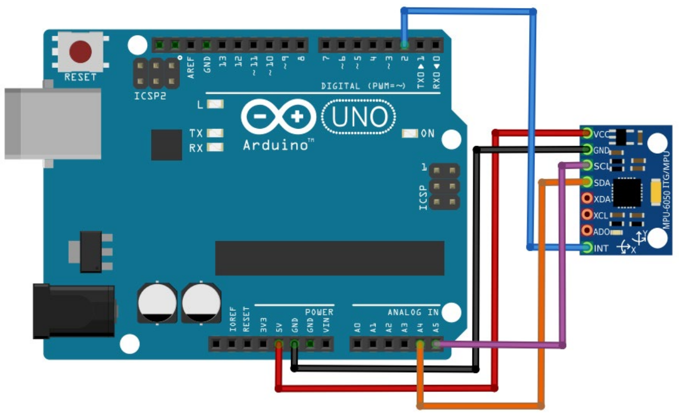


Figure 2: IMU connection to Arduino Uno

Bluetooth module for wireless communication

To establish wireless communication between the robot and your laptop. Wireless communication will be used to send and receive data between the robot and laptop and can also be used to program the robot. It will also be used to command the robot to drive remotely (teleoperation), send path plans to the robot for path following (autonomously), send robot locations to the laptop (localization), send robot worlds to and from the laptop (mapping), send robot updates to the laptop (SLAM).

There are several types of wireless communication including RF transceiver, Wi-Fi, Xbee, as well as Bluetooth. In this lab we will use Bluetooth communication with the DSD TECH HC-05 Bluetooth Serial Pass-through Module Wireless Serial Communication with Button for Arduino. It works within 100 meters of the robot at a 2.4 GHz frequency and a maximum data rate of 1Mb/s.

The Bluetooth module operates at 3.3V and has a 3.3V internal regulator so it can be connected to the 5V power line. Figure 3 shows the pinout for the HC-05 Bluetooth Module.

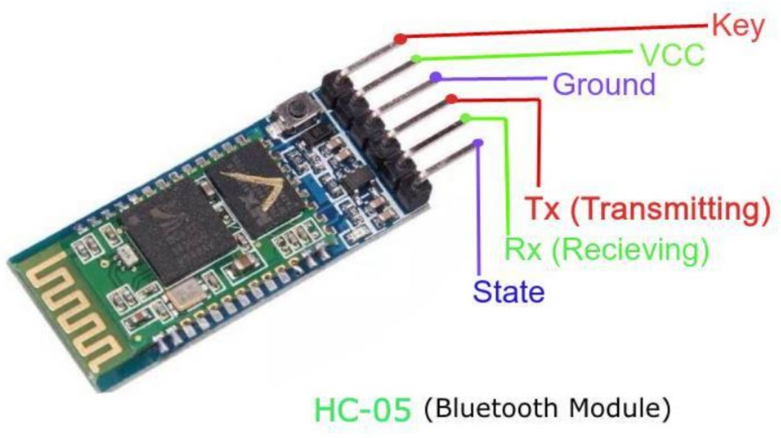


Figure 3: HC-05 Bluetooth Module pinout

The Enable pin sets the module to Data mode if LOW (default) or AT command mode if HIGH. VCC is connected to the +5V power supply. Ground is connected to the ground on the robot. TX (transmitter) is the pin that transmits the received data serially. The RX (receiver) sends data serially over Bluetooth. The state pin is used to check if the Bluetooth module is working properly. The red LED indicates the connection status and slows down to 2 seconds when connected. To send and receive data to the Bluetooth module, connect your computer to a Bluetooth device, find the HC-05 module and use the pin number is '1234'.

To program the Arduino over a Bluetooth serial connection, to change the default serial baud rate on the HC-05 Bluetooth module to 115200 bp/s. This requires putting the module in AT data command mode. In this mode the onboard LED will change to a 2s blink.

To program the Arduino over a Bluetooth serial connection, it is necessary to trigger a reboot on the microcontroller. The Arduino uses the DTR (Data Terminal Relay) signal to reset it by pulling the reset line to ground temporarily. To make a temporary connection, it is necessary to insert a capacitor in the circuit. This will block a continuous signal and allow for a quick transition. This can be done with the state pin on the HC-05. Build the circuit as shown in Figure 4 to program your robot using the Bluetooth module.

Run the Arduino IDE or Visual Studio in administrator mode. There are two COM ports connected to the Bluetooth module. The first is for receiving data and the second is for sending data. Select the second COM port to program the Arduino. Verify and upload the code to the COM port as you typically do. See more information about this process at the following link.

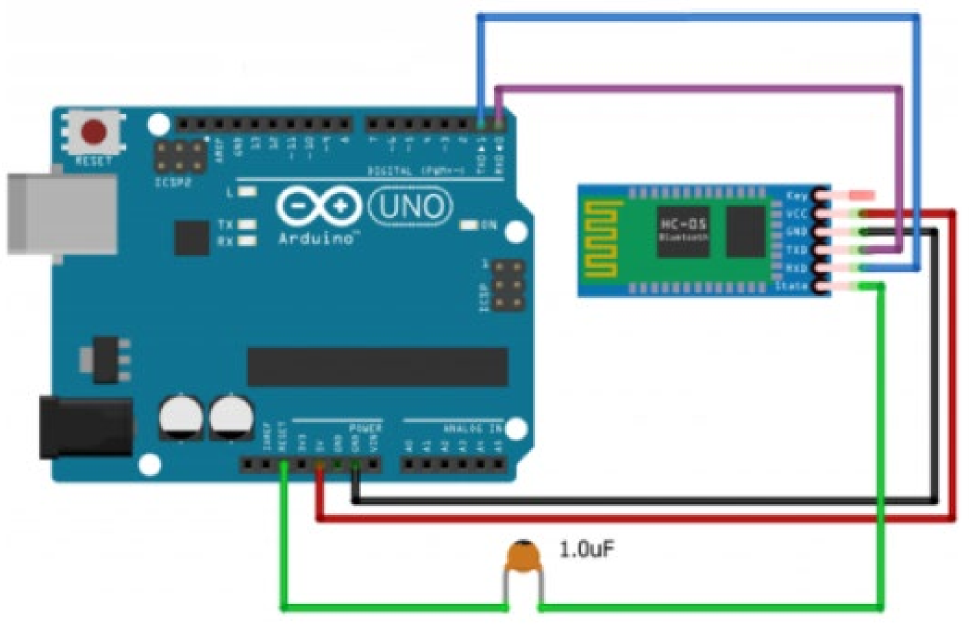


Figure 4: HC-5 Bluetooth Module for programming robot

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 5 is an example of topological navigation using the command "SRLT".

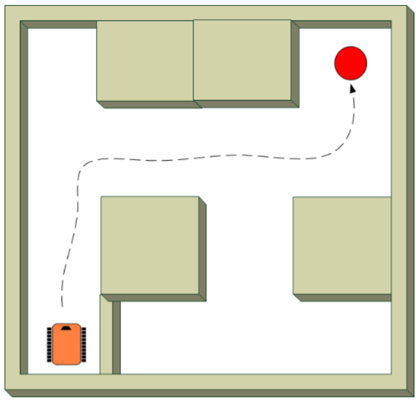
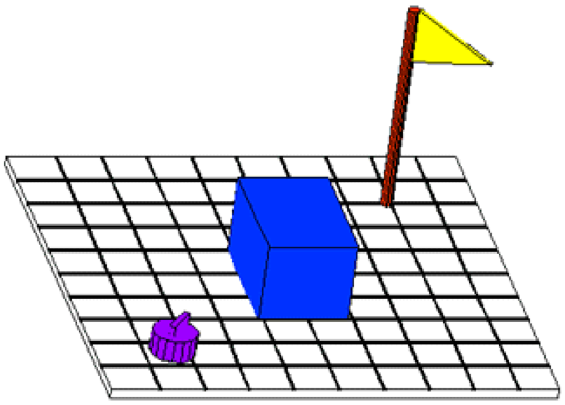


Figure 5: Topological Navigation ("SRLT")

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 6 is an example of the world representation.



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

a. Real world b. Configuration Space (8 x 8 matrix)

Figure 6: World Representation

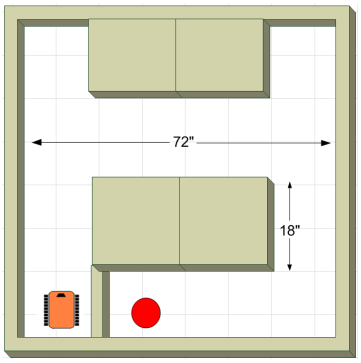
The wavefront is created by starting at the destination and creating eight connected neighbors back to the start point. The numbers represent the number of steps to the goal point. The robot would then follow the numbers in the reverse order to arrive at the goal point (see Figure 6). The goal is for 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.

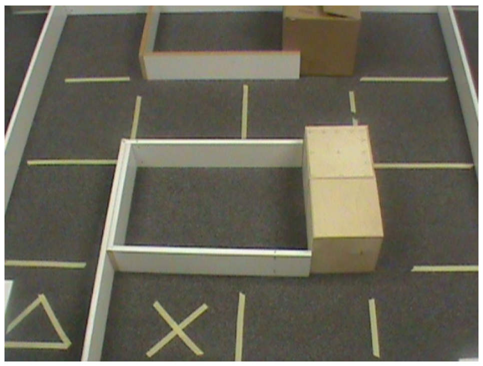
The test arena for the project demonstration will be 6 ft x 6 ft with 18" x 18" obstacles. This artificial world will be a 4 x 4 grid where the robot's start point is denoted by a 'Δ' and the goal point is marked by an "X". Figure 8 shows a sample test arena.

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 7: World Representation Wavefront





a. Artificial world b. Real world

Figure 8: 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 8 is represented by an 8 x 8 matrix of integers shown in Figure 9. Using the coding in Table 1, the maze shown in Figure 10 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.

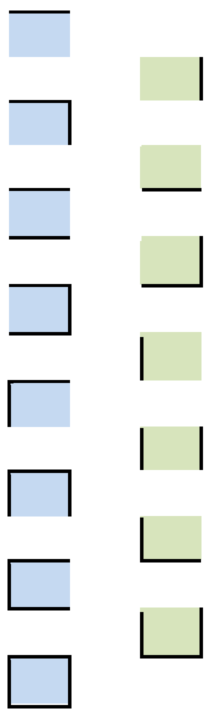
9	3	15	15	15	15	9	3
8	2	15	15	15	15	8	2
8	0	1	1	1	1	0	2
8	0	4	4	4	4	0	2
8	2	15	15	15	15	8	2

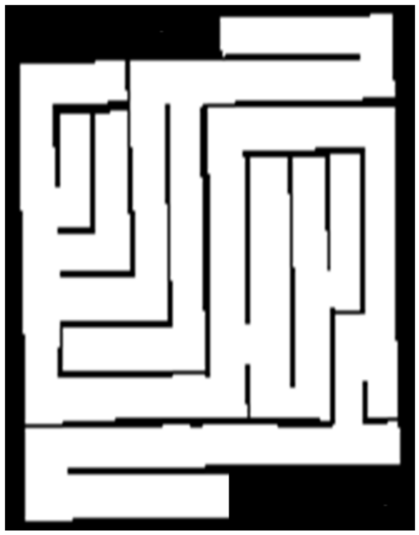
8	2	15	15	15	15	8	2
8	2	9	1	1	1	0	2
12	6	12	4	4	4	4	6

Figure 9: 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 10: 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 indicates 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 11.

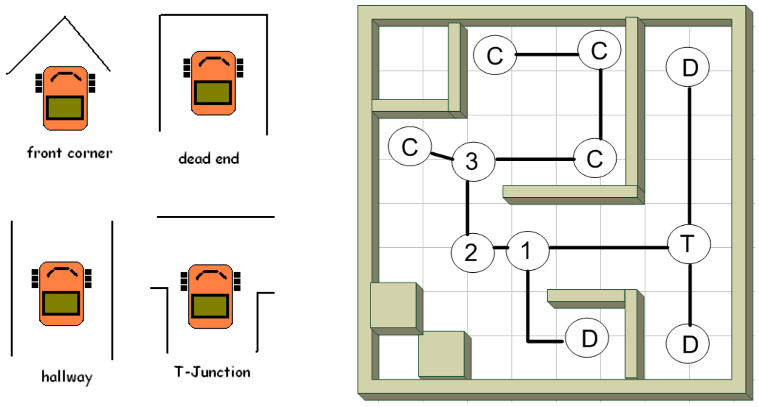


Figure 11: Feature Extraction (Map Coding)

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 12 provides an example of this localization process with the proposed robot locations after each step marked on the map.

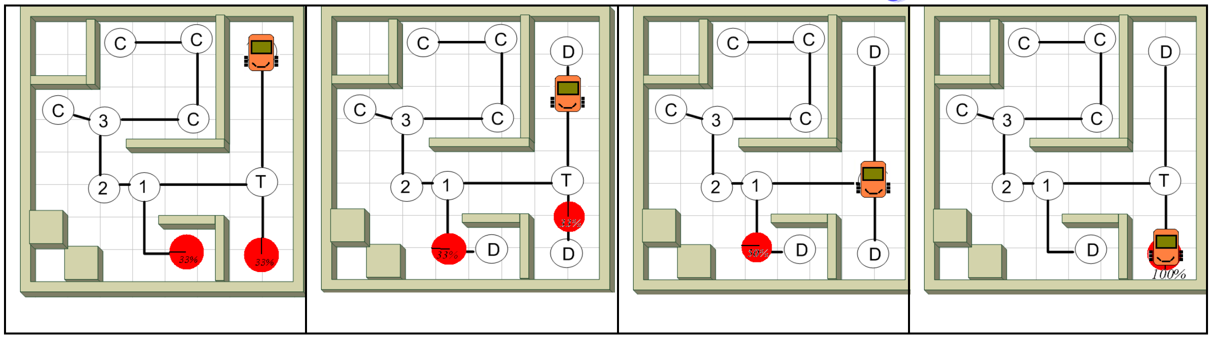


Figure 12: PMDP Localization

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 13 provides an example of using a Generalized Voronoi Graph (GVG) and HIMM to create a world map.

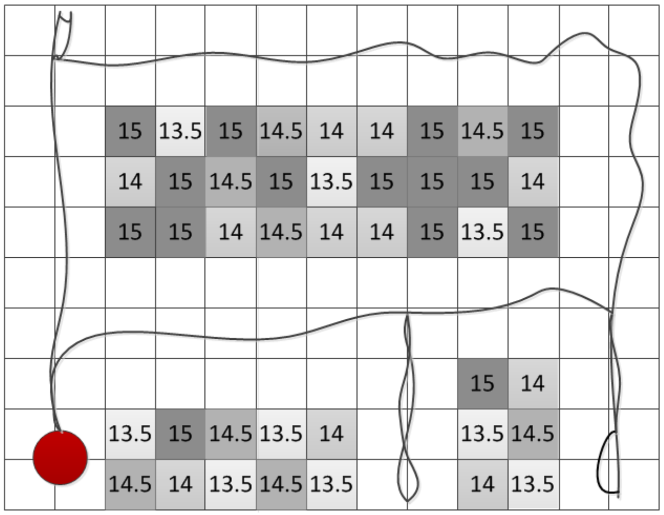


Figure 13: GVG with HIMM

****

PROCEDURE

****

PART A – IMU, WIRELESS COMMUNICATION & GRAPHICAL USER INTERFACE

Part A.1 – Attach IMU

To help with navigation and odometry, attach an IMU to the robot to work in conjunction with the encoder and stepper motor steps.

1. Review the link here for setting up the IMU and downloading the library.
2. Download the I2C library from this link here. Unzip the folder and then Select Sketch → Include Libraries → Add .ZIP Library from the Arduino IDE's menu.
3. Select the “Arduino/I2Cdev” subfolder of the unzipped folder. Click the Open button.
4. Run the example by select File → Examples → MPU6050 → Examples → MPU6050_DMP6.
5. Display the yaw, pitch, roll data on the serial monitor to understand the type of information the IMU provides.
6. Run the example by select File → Examples → MPU6050 → Examples → MPU6050_raw. Understand the data displayed on the serial monitor.

Question to answer in the final report:

How could you use this data along with dead reckoning, motor steps and encoders for robot navigation?

Part A.2 – Attach Bluetooth module to send and receive data to and from robot.

1. Connect VCC to 5V, GND to ground, TX to pin 10, and RX to pin 11. The password for the Bluetooth module is ‘1234’ and the baud rate is ‘57600.
2. Follow the instructions here to confirm that your computer connects to the Bluetooth module.
3. Follow the instructions here and here to confirm you can turn an LED on and off.
4. Next, figure out a way to wirelessly read sensor data from the robot and print it to the serial monitor.

Question to answer in the final report:

How did you implement bidirectional wireless communication on the robot?

Part A.3 – Attach Bluetooth module to remotely program the robot [OPTIONAL]

1. See details on this process here and here.
2. Rewire Bluetooth module by connecting GND to GND, RX to pin 11 and TX to pin 10 (Do not connect VCC yet!)
3. Hold down button on Bluetooth module and connect VCC to 5V, then release the button. The blink rate on the LED should slow down to 2s,

Change the baud rate on the Bluetooth to 115200 bps.

4. See details on this process here.
5. Rewire Bluetooth module by connecting GND to GND, RX to pin 11 and TX to pin 10 (Do not connect VCC yet!)
6. Hold down button on Bluetooth module and connect VCC to 5V, then release the button. The blink rate on the LED should slow down to 2s,
7. Upload the *HCO5.ino* code in the resources folder to the microcontroller.
8. Change serial monitor to both NL & CR to send AT commands to Bluetooth module
9. Type 'AT' and click enter. Confirm you receive 'OK' to make sure the module is communicating.
10. Type 'AT+UART=115200,0,0' and click enter. Confirm you receive 'OK' to confirm the baud rate has changed. Do the same for all the following commands:
    - AT+ROLE=0
    - AT+POLAR=1,0
    - AT+INIT
11. To change name use 'AT+NAME=<PARAM>'.
12. To change password use 'AT+PSWD=<PARAM>'.
13. See all commands at the following link.

Upload a program to the Arduino using the Bluetooth module

1. Remove all connections and wire the Bluetooth module to your robot as shown in Figure 2. (RX to TX pin 1, TX to RX pin 0, VCC to 5V, GND to GND, STATE to 1μF capacitor to RESET pin on Arduino)
2. Open the Blink code at File → Examples → Basics → Blink.ino
3. Select the second COM port for the Bluetooth module and upload to the Arduino.
4. After this is working correctly move on.

Part A.4 – Graphical User Interface

1. Review the following links to learn how to use Processing, MATLAB, JAVA, or Python to create a GUI (Human-Robot Interface) for your robot. It will use the Bluetooth module and serial communication to send and receive data with the robot and laptop. The team can use any language they are comfortable with for this part.

- http://www.sojamo.de/libraries/controlP5/#examples
- https://www.youtube.com/watch?v=NTXkmpfTklQ
- https://www.youtube.com/watch?v=5WjEQSMiqMQ&t=506s
- https://www.youtube.com/watch?v=Rgq8oy3to-E&t=10s
- https://www.mathworks.com/help/supportpkg/arduinoio/index.html?s_tid=CRUX_lftnav
- https://www.mathworks.com/help/matlab/creating_guis/about-the-simple-guide-gui-example.html
- https://www.mathworks.com/help/supportpkg/arduinoio/getting-started-with-matlab-support-package-for-arduino-hardware.html?s_tid=CRUX_lftnav
- Building Apps with MATLAB & APP DEsigner

https://www.youtube.com/watch?v=nQb0qBiDurU&ab_channel=MATLAB

- Create and Run a Simple App Using App Designer

https://www.mathworks.com/help/matlab/creating_guis/create-a-simple-app-or-gui-using-app-designer.html

2. Design a user interface that at a minimum includes all of the following features

- User can remotely drive the robot through the interface
- User can enter topological paths to send to robot
- User can send start and goal positions to the robot
- GUI displays the robot world with planned path
- GUI shows robot status such as task progress, current localization, map created, beacon found, exit found, escape status

Question to answer in the final report:

- Describe how you made design choices for the robot graphical user interface.
- Create a mockup of the GUI you will design to send data and display sensor data for path planning, path following, localization and mapping information, and SLAM if you do that part. Include it in the appendix of the final report and discuss it in the results section.

PART B – PATH FOLLOWING & PATH PLANNING

Part B.1 - Topological Path Following

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 10 provides descriptions of the possible world landmarks.
3. Include a table similar to Table 3 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. You can send the path to the robot using the GUI and indicate robot status using LEDs, the serial monitor and on the GUI.

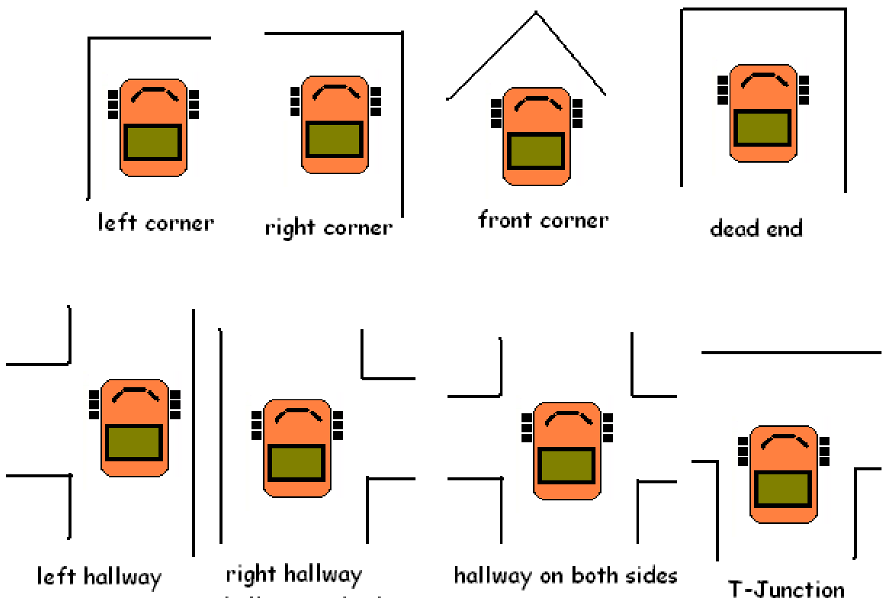


Figure 10: Distinctive Places

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

Table 3: 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 B.2 - Metric Path Planning

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 5 and 6 for examples of occupancy grids and see Figure 10 for example 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. You should also use the GUI to send and receive data.

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.

Question to answer in the final report:

- What method did you use to send path and goal information to the robot?
- What method did you use to display robot status during path planning and following?
- Create a software design plan for how you will implement topological and metric path planning and path following on your robot.

PART C – LOCALIZATION AND MAPPING

Part C.1 - 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 C.2 - 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.

Question to answer in the final report:

- Submit a plan for how you will implement localization and mapping on your robot in the appendix of the report, describe how it worked.
- What method did you use to receive localization data?
- What method did you use to display the world map?

Part D – Simultaneous Localization and Mapping

Required for Graduate Students, Optional Extra Credit for Undergraduate Students

1. The robot will be placed in an unknown world at an unknown location.
2. The robot must use an autonomous coverage algorithm to create a map of the world and display it on the user interface.
3. The robot will then navigate through the world to find a light beacon in an unknown location.
4. Once located, the beacon location must be displayed on the user interface.
5. The robot must then explore the world to find the world exit as identified by a location where a wall has mysteriously disappeared.
6. The robot should display status updates as it localizes, creates the map, find the beacon, and plans the escape route and exits the maze.

Question to answer in the final report:

- Submit a plan for how you will implement SLAM on the robot in the appendix of the report and describe how it worked in the method section of the final report.
- How is this different than what you did to implement localization and mapping separately?

****

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 **three (or four)** software design plans and 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 4.

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.

Question to answer in the final report (results section, or appendix if appropriate):

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 4. There will be rolling demonstrations over the three weeks so as soon as a team has a part ready, they should go ahead and be graded. All demonstrations must be submitted by the last day of week 10.

Table 4: Project Grade Point Distribution

Task	Points	Due by
Software Design Plan I	5	Second Class of week 8
Phase I Demo - Bidirectional Communication & GUI	10	Second class of week 8
Software Design Plan II	5	First class of week 9
Phase II Demo - Path planning and following	10	Second class of week 9
Software Design Plan III	5	First class of week 10
Phase III Demo - Localization and Mapping	10	Second class of week 10
Software Design Plan IV (optional)	(5)	First class of week 10
Phase IV Demo - SLAM (optional)	(10)	Second class of week 10
Code	20	Sunday of Finals Week
Report	35	Sunday of Finals Week