

# Robotics: Assignment I

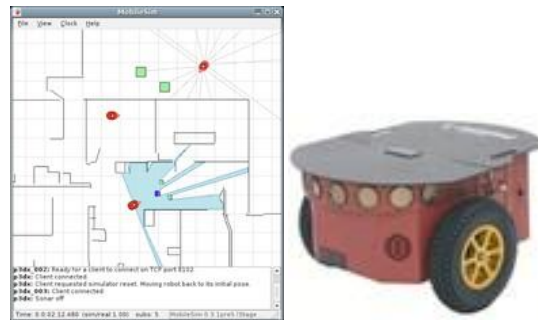
## Pioneer 3-DX Simulation

Due: 2019/10/07 13:00 pm

Pioneer 3-DX is a mobile, differential-drive robot platform from Adept MobileRobots. MobileRobots provides the Pioneer SDK, a collection of libraries and applications, which help the users to control the robot platform and develop programs on it. In this assignment, you will learn how to use ARIA library to control the Pioneer robot and examine your program with the robot simulator, MobileSim.

ARIA is the core C++ library of the Pioneer SDK which provides an interface and framework for controlling and receiving data from the robot platforms.

MobileSim provides a simulation environment in which you can test your program on a simulated robot to make sure it works properly.



**Environment:** Ubuntu 16.04

### Part A: Getting Started

- 1) Install the ARIA library and MobileSim.  
ARIA: <https://reurl.cc/W4dl2x>  
MobileSim: <https://reurl.cc/Qpdzr5>
- 2) Start MobileSim and load the map named "Columbia" in the MobileSim directory. Try out following functions:
  - [View]->[Position Data for all models]: show the robot position on the screen.
  - [View]->[Show Trails]: show the trail of the robot.
  - [File]->[Export]: record the process as images.
- 3) Run the ARIA demo program (/usr/local/Aria/examples/demo) and connect to MobileSim. Try out different operation modes (teleop, wander...).

### Part B: Building a C++ Program That Uses ARIA

- 1) Please refer to "/usr/local/Aria/README.txt" to find how to build a C++ Program on Linux.
- 2) We provide an example code ("example.cpp") which can connect to a simulated robot. Try to build this program and test it with MobileSim.

*Note: If everything is working correctly, the simulated robot should keep moving forward until it bumps into a wall.*

### Part C: Keyboard Control

- 1) The ARIA demo program provides the “teleop” mode in which the user can navigate the robot with the keyboard. Try to implement similar functionality with ARIA library and name your source code “part\_c.cpp”. You are free to design the behavior of the robot when each key is hit (for example, hitting the left arrow will do a rotational motion when pushed and return it to zero when released); however, you must control the robot by setting the velocity directly. Do not use ArAction.

*Note: You can try these functions: setVel(), setRotVel() (see the documentation on the Aria website)*

### Part D: Collision Avoidance using Sonar Sensor

- 1) When developing a robot, safety is always the first priority. What happened in Part B should be avoided. With the onboard sonar arrays, the Pioneer 3-DX is able to detect surrounding environment and respond to it. Try to get the readings and corresponding orientations from the sonar sensors.
- 2) Modify your keyboard control program and try to make the robot slow down when it is close to the obstacles to avoid a collision.

### Part E: Robot Pose and Odometry

- 1) Robot pose includes robot’s position  $(x, y)$  and orientation  $\theta$ . With Aria, you can get the current robot pose by using the getX(), getY(), and getTh() functions, which are estimated by odometry. In MobileSim, you can get the true pose. *Is the odometric pose the same as the true pose? Why?*
- 2) Devise a program that can move the robot from  $(x, y, \theta) = (0, 0, 0)$  to a specific target pose, e.g.  $(7, 5, \frac{\pi}{2})$ . Let’s use triangle.map located in “./MobileRobots/Aria/maps/” and assume the target is reachable. The program must take input from standard input with following format:

`>>x y theta`

(For example, if input is `>>7 5 1.57`, the robot should move to pose (7, 5, 1.57) accordingly.)

Name your source code “part\_e.cpp”.

### Bonus Round: Obstacle Avoidance

Use the “obstacle.map” that is included in the zip file you got from CEIBA. We will test obstacle avoidance using this map by selecting a target pose  $(x, y, \theta)$  across the obstacle. Your program should make use of the sonar information (as in Part D) to plan a local path around the obstacle to reach its target.

*Note: You cannot pre-program a path around the obstacle. You must use the sensor input to receive credit for your program.*

Name your source code “bonus.cpp”.

## Evaluation Criteria

- Completeness: Part C and Part E run smoothly.
- Report: In English. Brief and concise.
- Tidiness: Proper syntax and indentation in your code.
- Bonus: Extra points depending on the performance of your algorithm.

## Submission

- 1) Source code: Only part\_c.cpp, part\_e.cpp and bonus.cpp (if you decide to give it a try) are needed. Do not send the entire project to us. Please make sure your code can be successfully built and run before submission.
- 2) Report: Please describe your implementation of Part C, D and E and why you designed it that way. In part E, please elaborate on what is the difference between odometric and true pose in MobileSim. If you submit the bonus round, please briefly explain what kind of algorithm you used for navigation.
- 3) Save your report in PDF format, zip it together with your .cpp files and upload to CEIBA. Name your zip file as [Student ID]\_HW1.zip

## References

[1] [ARIA Developer's Reference Manual at /usr/local/Aria/Aria-Reference.html](/usr/local/Aria/Aria-Reference.html)