Lab Worksheet

**Lab Number (circle this week’s lab)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**Name**: Chimzim Ogbondah

**Lab Section**: 1

**Lab Partner Name**: **Ruiyu Sun**

This lab worksheet is the final deliverable for a lab. You will usually have three deliverables for a lab:

1. **Prelab assignment BEFORE LAB**: Posted with the lab manual, typically involves a system sketch, submitted in Canvas before the start of your lab section, may be worked on and used by lab partners in class on Tuesday during lab planning
2. **Demonstrations IN LAB**: Demonstrated/discussed with a TA in lab and recorded using a demo evaluation sheet to be printed and signed in lab (functional demo of a lab milestone, debug demo using debugging tools to explain something about the internal workings of your system, Q&A demo showing ability to formulate and respond to questions)
3. **Postlab assignment AFTER LAB**: Submitted in Canvas before the start of your next lab section, may be reviewed by lab partners in class on Tuesday during lab retrospective, consists of three items (prelab planning boards, lab notes, and lab retrospective)

Deliverable #1 has its own Canvas assignment submission. (10 points)

Deliverable #2 has an evaluation sheet that is printed in lab, used as a checklist, and submitted to your TA. The TA will enter points in Canvas based on the demo evaluation sheet. (40 points)

Deliverable #3 has its own Canvas assignment submission. (30 points)

This worksheet will help you develop the items needed for deliverable #3.

1. **PRELAB PLANNING BOARDS**
2. Question Board: What are the three priority questions from your lab planning work?
   1. How to effectively work with structured pairing?
   2. What are best abilities were with structured pairing?
   3. How to use the different functions that align with the iRobot sensors
3. Task Board: What are several tasks you identified in your planning (for you and lab partner)?
   1. Designate when we would switch off between driver and navigator
   2. What the roles of navigator and driver meant
   3. How to effectively turn the iRobot
   4. Manipulating the iRobot based on given conditions
4. **LAB NOTES**

During lab, keep notes about the following so that you can submit information with this deliverable.

1. Results related to the three priority questions (might be answers, might be more questions, write brief summaries, don’t include code files)
   1. As we went through the lab me and my lab partner discussed how we would go about using structured pairing. I started on the keyboard programming and thinking of logic when we got to the final step of the lab, based on what the iRobot was doing my lab partner would then explain to me why she thought something was going wrong and then together we would look to see what was going on in the code. From there I would alter it to try and fix the problem at hand. It was kind of an unsaid rule, she let me program first but still helped with logic, then we switched both doing the same thing. I think we were both very good at navigating. We looked inside the command.c file to see that we needed to use the sensor data “direction” and “angle” to have the right data. We also thought about how the set\_wheels(0,0) function worked to get it to turn left and right messing with logistics on how to actually turn (negative one way positive the other way) and then how left and right versus back and forward moved when we passed a parameter through the functions we made.
2. Any additions, refinements, or corrections to the prelab system sketch based on what you learned (include an updated sketch, or briefly describe at least one update you made)
   1. I didn’t make an corrections to my prelab sketch
3. Description of your debug demo (what did you demo and why, what did you find, a paragraph is fine, may want to include a screenshot)
   1. We did our demo over debugging by stepping through our final step. So initially we had a bigger if state saying if either side was bummed then we should move the iRobot back then inside that if statement we had another one that based off of which sensor was touched instructed the iRobot to move the other way. At first, we didn’t understand why it kept hitting a sensor and then continuing to move back and then proceeded forward. After debugging we saw that it was never able to enter the second condition because once it was touch it moved back and would keep repeating the processes because it wasn’t touched back to back or held down for a long enough time.
4. **LAB RETROSPECTIVE**

Take 10-15 minutes and answer these questions as you think about your lab experience. You don’t need to describe everything, try to pick something notable.

1. What did we set out to do?
   1. We set out to try and understand how the sensor data communicated and then manipulate it using turnLeft(), turnRight(), move\_forward() and move\_backwards() to help us complete the lab by having the iRobot react to running into an object or “wall”
2. What actually happened?
   1. We first watched how the move forward function worked observing its behavior, then based on that we did the move back function by changing the velocities of the set\_Wheel() function to be negative. From there we worked on turnLeft() and turnRight(). We used the same approach by messing with the setWheels() function. At first, we had just on wheel going and the other one set to zero (200,0) but upon seeing the results we changed the wheel set to zero to be negative with the same velocity. Once this was done, we began changing the move\_forward function so upon hitting something it exited the function. Then inside our main method we worked logic to get it to move backwards and turn the opposite way of the sensor it touched while still keeping track of the distance gone… then moving forward until the end distance was met.
3. Why did it happen?
   1. This happened because of our attack using structured pairing and through the collaboration of both our ideas for how to start the problem and then coming back through to fix any errors to our solution.
4. What are we going to do next time (to improve)?
   1. Try to walk through the logic of our code before going to test, make sure me truly understand what is meant to happen. This could have saved some time, and then working through getting a better 90-degree turn. Essentially understanding how to meet the deliverables for the lab to the mark.