# Memo

**To:** Dr.Berry

**From:**  Carson Stone | Peter Garnache

**Date:**  02/03/2020

**Subject:**  Mobile Robotics Lab 4

|  |
| --- |
|  |

## Introduction

The purpose of this lab is to develop the code needed to add a light homing behavior to the wall following code from lab 3. The following functions were developed to allow for this behavior.

* goToLight()
* atGoal()
* returnHome()
* reset()
* fearState()
* loveState()
* explorerState()
* aggressiveState()

## Method

**PART A**

We kept many of our functions from prior versions of our code because we were asked to do many similar tasks. The following are the new functions that we wrote in order to complete the objectives of part A.

When the fear function was called, the robot is supposed to increase the speed of the left stepper motor based on how much light is seen on the left compared to the right. We accomplished this by setting the turn portion of our heading equal to the negative of the difference between the left and right photoresistors divided by 400 to reduce the heading value to less than one.

When the love function was called, the robot is supposed to decrease the speed of the left stepper motor based on how much light is seen on the left compared to the right. We accomplished this by setting the turn portion of our heading equal to the difference between the left and right photoresistors divided by 400 to reduce the heading value to less than one.

When the explorer function was called, the robot is supposed to increase the speed of the right stepper motor based on how much light is seen on the right compared to the left. We accomplished this by setting the turn portion of our heading equal to the difference between the left and right photoresistors divided by 400 to reduce the heading value to less than one.

When the aggressive function was called, the robot is supposed to decrease the speed of the right stepper motor based on how much light is seen on the left compared to the right. We accomplished this by setting the turn portion of our heading equal to the negative of the difference between the left and right photoresistors divided by 400 to reduce the heading value to less than one.

**PART B**

We kept many of our functions from prior versions of our code because we were asked to do many similar tasks. The following are the new functions that we wrote and functions that we updated in order to complete the objectives of part B.

Because the updateSensors() function was the call for our interrupts, and we wanted to implement finite time movements (like spin 90 degrees) we had to use an if statement to stop the interrupt from triggering the updateState function to keep the logic from updating when we were either moving through a finite time movement function, or when we were acting after we had detected the light.

The goToLight() function drove the robot from when it initially found the light, until the robot had reached its destination and was ready to dock. Each logic loop, we took note of which of the sensors read the higher light value, then turned away from that side until the other sensor read a higher light value. We then moved forward and re-implemented this movement command. This allowed us to effectively track the light and kept our traveled path as simple as possible. Our traveled path was only a vector of the angles that we turned, with the length of the values in the vector being the distance that we travelled. We stopped the movement when the front IR sensor detected an object within its threshold. We then called the atGoal() function.

The atGoal() function stops the robot, then spins it 180 degrees, then pauses it again. This simulates the docking response.

The returnHome() function replays the movements taken when following the light. The robot turns the same angle that it did on the way there, then moves forward. It continues this behavior until it reaches the end of the recorded path. Then it turns 180 degrees and resumes the last state.

reset() was another function that we created that resets our position tracking variables, as well as our stepper variables.

We created the readPhRight() and readPhLeft() functions to read the photoresistors and update their global variables.

runToStop() has been updated to stop the interrupt from interfering with a movement command.

## Results

## **https://lh5.googleusercontent.com/_aFMnkX9EdI7E82QX5bKUQ_0ZBvoooZeOhlMevN733Pqf5d5ceU4PC4PAzO0VlQbR8Dunbg7LQYf-X-ltanJV2UNgwU96ttyfB3IOGBq0e2rVpqW57kr_gk6Se4xTdjFmRLHkR6z**

1. How reliable was the photoresistor at detecting the light in different environments, various distances and angles of incidence (head on, slightly left, slight right).

From distances less than 30 inches the photoresistors proved to be fairly accurate at detecting how far the light source was from the robot. The photoresistors could also detect whether the light source was shining directly on the photoresistor. The photoresistor was not accurate enough to detect the actual angle the robot is with respect to the light source, but it could tell if the left photoresistor or right photoresistor had a greater angle of incidence with the light source

2. How significant was the difference in photoresistor voltages for the left and right sides. How did you use this difference to extract directional information to move the robot toward the beacon?

The difference between left and right photoresistor was too small to tell what angle the robot was at relative to the light source, however the difference between the photoresistor values was more than adequate to reliably use as a boolean to turn right or left.

3. How did you integrate the light sensors into the obstacle avoidance behavior?

The light sensor was used to detect whether the robot should exit wall following or random wander and go into light following. Obstacle avoidance is triggered from the IR sensors detecting that an obstacle is too close. This can happen from the light following state or anh other state.

4. How reliable was the photoresistor at detecting different objects at various shapes, sizes and distances. Compare and contrast sensor data.

The photoresistor could accurately detect the distance the robot was from the light source as long as the robot was less than 30 inches from the light source.

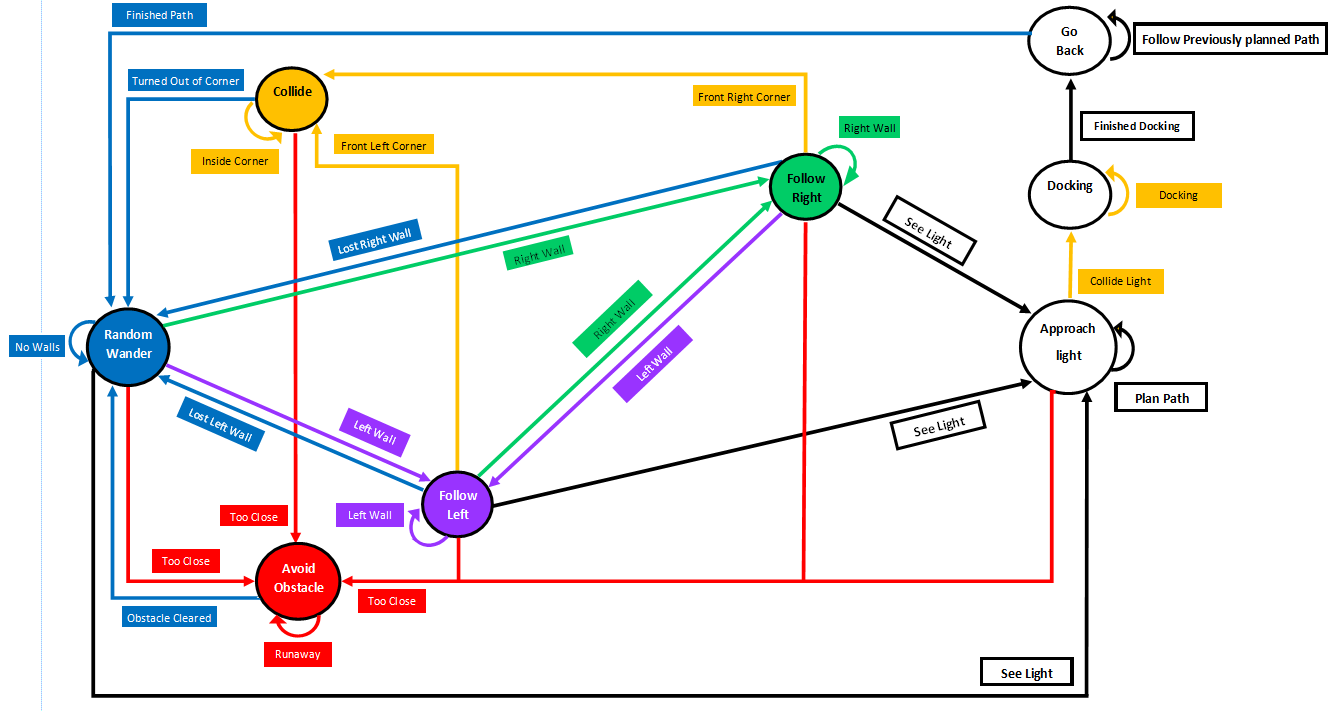
5. How significant was the difference in photoresistor voltages for the left and right sides. How did you use this difference to extract directional information to move the robot toward the beacon?

The difference between left and right photoresistor was too small to tell what angle the robot was at relative to the light source, however the difference between the photoresistor values was more than adequate to reliably use as a boolean to turn right or left. There was about a dropped 5 or 10 from the light hitting the photoresistors at an angle of 45 degrees. The follow light state detects whether the light is stronger on the left or the right side and turn towards the light then moves forward.

6. How significant was the difference in sensor data based upon distance from the source? How did you use this difference to extract distance information to move the robot toward the beacon?

The photoresistors values dropped about 10 each time we moved the light source 6 inches away from the robot. We did not use the intensity of the photoresistors to determine the robot’s distance from the light source, but instead just used the photoresistors to determine what direction to spin to look directly at the light before moving forward. We instead used the front IR sensor to detect the distance the robot was from the light source and used it to stop moving towards the light and move back where the robot was before it detected the light source.

7. What does the hybrid control architecture for your design look like? What was on the planning layer? Middle layer? Reactive layer?



The state diagram of the robot behavior is shown above. The planning layer occurs while the robot is in the approach light state. While in this state, the robot is remembering all of the turns it takes to get to the light and all of the movements it makes. The middle layer is found in the drive() and spin() functions. These functions take finite time input behaviors and translate them into movement commands for the robot. They also suppress the output from the state diagram and other reactive states during their execution. Our entire wall following program constitutes the reactive layer, because all of these movements are direct responses to sensor input.

8. What was your general strategy for planning the path back to the wall from the beacon?

While the robot is approaching the light source, it is constantly turning towards the photoresistor that has the higher value. Once the robot determines that it needs to turn towards the light, it spins in 5 degree increments until the photoresistor that was higher is no longer higher. After this, the robot moves forward and checks again if it needs to turn towards the light again. This process of checking if the robot needs to turn towards the light before moving happens

9. How did the architecture respond to differences in robot start position or beacon location?

One problem that the architecture did not like was when the light source was pointed at an angle towards a wall the robot was following. This would result in the photoresistor closer to the wall falling into the path of the light source first which would cause the robot to think that the light source is on the side of the wall it is following. This would result in the robot turning towards the wall and running into it. This problem was avoided whenever the light source was perpendicular to the robot so that the photoresistor that was closer to the light source was always triggered first, or if both of the light sensors are activated at the same time.

10. How did the robot’s hybrid controller respond to dynamic changes in the environment (i.e. other robots and people) and compare this to purely deliberative control.

The robot behaved differently from the different walls and obstacles in the environment. The robot would stay in random wonder until it detected a wall to follow or an obstacle to avoid. The robot would also leave the reactive layer and enter follow light as soon as a light source was detected. When the robot entered follow light and what that behavior looked like changed based off of how the light source was positioned relative to the robot.

11. Were there any challenges in implementing the homing routine?

There were no challenges in implementing the homing route. Our first Idea on how to implement the homing route worked just as planned. Our code did not work perfectly the first time, but we realized that that was the fault of a single missing semicolon.

12. What could you do to improve the robot homing?

We could have created a more optimized path to the light source that used more than boolean logic to turn left or right before moving forward to the light source. We also could have calculated an optimized direct path back once the robot has reached the light source.

13. How did docking the robot modify the control architecture or algorithm?

Once the robot entered follow light, we stopped using the interrupt with reactive control and moved more towards deliberate control by having interrupt not cause anything to happen while in the move towards light state. The biggest challenge for this lab was having the states switch between continuous time with an interrupt and discreet distances with the moveToStop stop function being used with the move to light state.

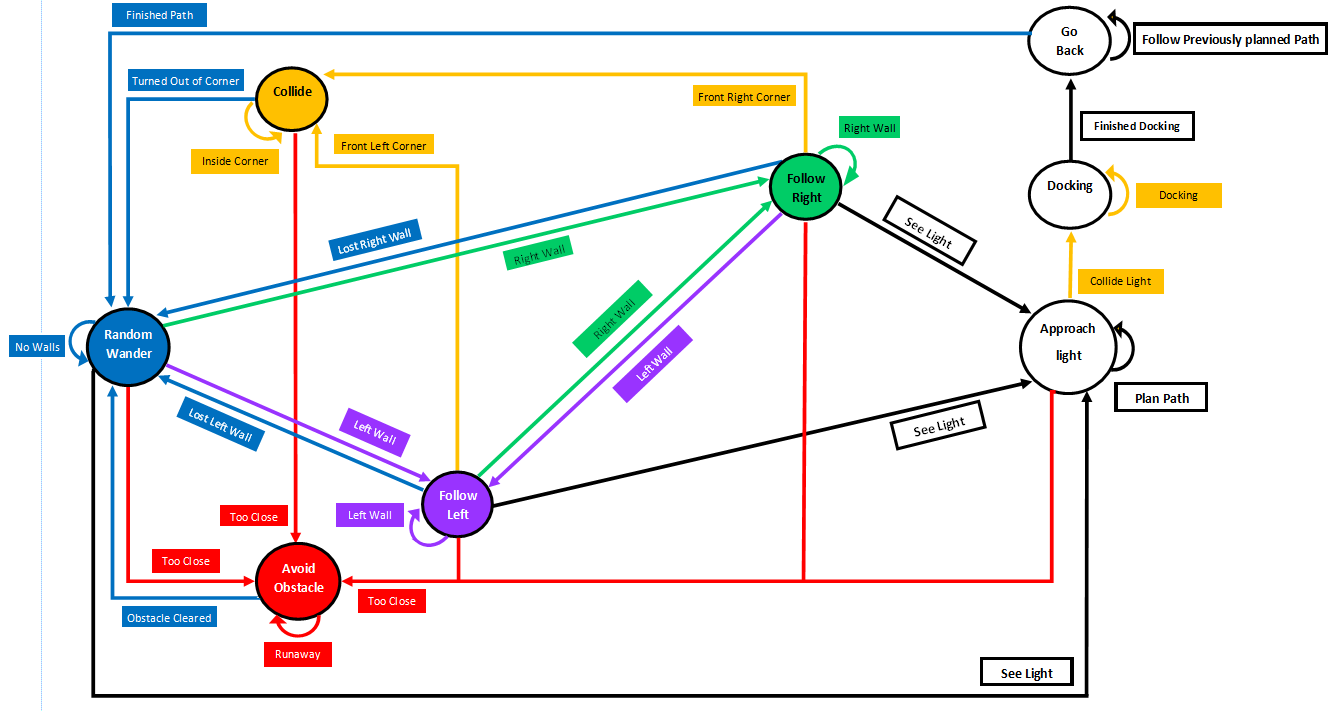
## Conclusion

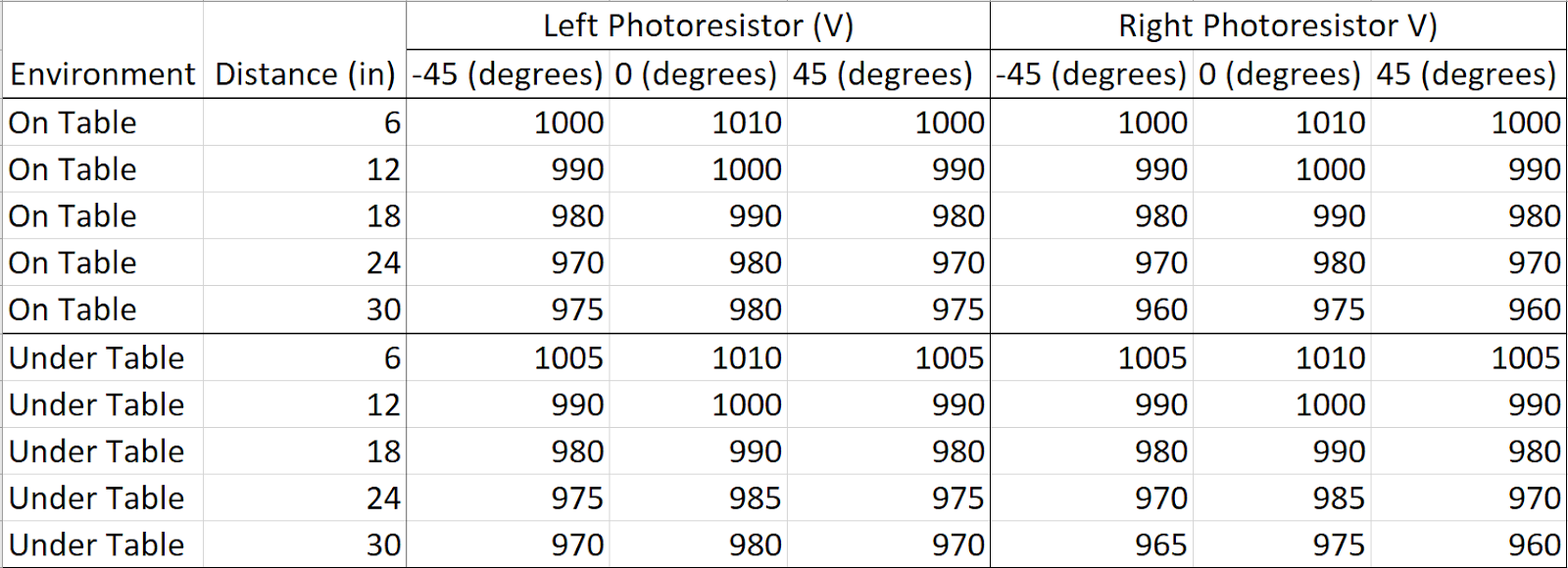
This lab, we learned to implement a hybrid control architecture that utilized both reactive and deliberative control to move our robot. We used reactive control to navigate around the environment and track the light to the position of the light source. We used deliberative control to navigate back to the position where the robot lost the light using internal position tracking and path storage.

One of the interesting things that we were forced to implement was adding finite time movements to our prior continuous program. This was accomplished with a global variable that stopped the loop from continuing to execute logic after updating the sensor values. This allowed movement that spanned across multiple logic cycles, which made the light following behavior much easier than expected.

We successfully completed the target of the lab and created several different behaviors for our robot that will help us be successful in future labs and tasks

## Appendix:



****