**ECEN 345**

**Mobile Robotics I**

**Prof. A. Gilmore (Instructor)**

**Kashfia Nabile (TA)**

**Laboratory Assignment # 5**

*Homing on a Light Source using BBC*

*Written by*

***Abdul Rahman Al kharusi***

***David Perez***

**Department of Electrical and**

**Computer Engineering (ECE)**

**University of Nebraska**

**Lincoln**

**Due**: 03/31/2024

**Contents**

[**Introduction** 1](#_Toc162189171)

[**Background** 1](#_Toc162189172)

[**Procedure** 2](#_Toc162189173)

[**Part I** 2](#_Toc162189174)

[**Part II** 2](#_Toc162189175)

[**Source Code Discussion** 3](#_Toc162189176)

[**Results** 4](#_Toc162189177)

[**Conclusion** 6](#_Toc162189178)

# **Introduction**

The main purpose of this lab was to create a behavior-based controlled robot that could cruise around, avoid obstacles using infrared sensors, and home in on a light source using photoresistors. The specific objectives included adopting a formalized behavior-based control (BBC) program structure to implement modular behaviors emulating cooperative multitasking and priority arbitration with the order of increasing priority defined as: *CRUISE, LIGHT\_FOLLOW*, and *IR\_AVOID*. Both Abdul and David worked together in completing the lab. Abdul focused on implementing the IR\_SENSE(), IR\_AVOID(), and bonus functionality for obstacle avoidance in part 1. He also coded the PHOTO\_SENSE() function for part 2. David focused on the *LIGHT\_FOLLOW* behavior, including both the debugging and coding to make the robot home in on a light source using differential photoresistor values, as well as the bonus for that part. We successfully completed all required and bonus tasks to create a robot that could cruise, avoid obstacles, and follow light source. Approximately 4 hours were invested working on this lab assignment.

# **Background**

In this lab, the code was structured using a formalized behavior-based control (BBC) architecture in C that implements cooperative multitasking among different behavioral modules with fixed priorities. This modular BBC structure, presented in the bbc skeleton program and shown in **Figure 1**, separates sensing, motor control, and each behavior into distinct modules.

A diagram of behavior control

Description automatically generated

**Figure 1**: BBC Structure

The sensing modules, such as IR\_SENSE() and PHOTO\_SENSE(), interface with the physical sensors to read data. The motor control module handles actuation of the robot's motors based on the active behavioral module's output. Finally, the core behavioral modules like *IR\_AVOID, CRUISE*, and *LIGHT\_FOLLOW* encapsulate the logic for each capability like obstacle avoidance, random wandering, and light tracking respectively. Arbitration between the behaviors is achieved by calling them in a fixed priority order within the main program loop.

# **Procedure**

## **Part I**

The provided BBC skeleton program was modified to include the *IR\_AVOID* and CRUISE behaviors implemented as independent modular functions, with *IR\_AVOID* having the highest priority in the arbitration order. The time interval within the *IR\_SENSE* function for reading the IR sensor values was adjusted, and the effects of different intervals on the robot's behavior were noted. The *IR\_AVOID* behavior was executed ballistically, meaning the final motor command in the sequence of avoid actions was sent directly to the motor action structure in bbc.c.

The robot was tested by allowing it to run freely in the lab space to ensure it could cruise around the room normally, but properly triggered the *IR\_AVOID* behavior when the sensors detected an obstacle. The implementation logic for the *IR\_AVOID* and *CRUISE* behaviors was built upon the previous work from Lab 3, with improvements made based on experimentation. As a bonus, an *ESCAPE* behavior within the *IR\_AVOID* function was created. This allowed the robot to trigger the *ESCAPE* behavior and maneuver out of scenarios where the IR sensors may have failed, enabling it to resume wandering and searching for the light source.

## **Part II**

The main goal of the second part was to make the robot follow the light source. A *PHOTO\_SENSE* function was created to read the left and right photoresistor values at some time interval. The code for the *IR\_SENSE* function was used as a starting point, and the timer service interval was adjusted to control how frequently the light sensor data was polled.

After implementing *PHOTO\_SENSE*, the *LIGHT\_FOLLOW* behavior was created to make the robot home in on a light source when the average light intensity from the two photoresistor sensors exceeded a minimum threshold value. Logic based on the difference in light intensity between the left and right sensors was implemented to guide the robot's motion towards the brighter direction using differential light detection.

An important aspect was internally calibrating the photoresistor sensors so the robot moved towards the sensor detecting greater light intensity, not just a higher reading value. The sensors were tested by measuring their voltage outputs when exposed to the same light source at an equal distance to check for imbalances.

With the addition of *LIGHT\_FOLLOW*, the final arbitration order of the behaviors became: *CRUISE*, *LIGHT\_FOLLOW*, and *IR\_AVOID* from lowest to highest priority. As a **BONUS**, a *LIGHT\_OBSERVE* behavior with higher priority than *LIGHT\_FOLLOW* was implemented. This behavior is activated when the *LIGHT\_FOLLOW* behavior gets the robot within 4-5 inches of the light source. It stops the robot for a short amount of time, back it up, and then turn away from the light.

# **Source Code Discussion**

The first part of this lab was to complete the *IR\_avoid* and *CRUISE* behavior to the bbc skeleton program. The *IR\_avoid* was ballistic to ensure it had the highest priority. The *CRUISE* behavior on the other hand was a servo behavior to ensure that it had the lowest priority. The *IR\_avoid()*  function would use the infrared sensors to detect if an object is to the left, if so, stop the robot, back up, turn right, and set the future motor functions to move forward. The same sequence would occur if the right sensor was triggered and the robot would turn left instead.

The *IR\_sense()* function is the function that populated the sensor data that was used by the *IR\_avoid()* function. This function would populate the global sensor data in intervals of 125ms. That is, if 125ms hasn’t elapsed then the robot would not sample the sensors. This is done in part to prevent oversampling of the sensors which would consume substantial resources and could cause the program to crash.

Additionally, an *ESCAPE* behavior was embedded into the *IR\_AVOID* behavior. This behavior triggers when both infrared sensors detect an object. This allows the robot to get out of corners or when facing directly against the wall. The condition under which this occurs is when both ir sensors detect an object thereby initiating the robot to stop, backup, and turn around. If none of the aforementioned IR sensor conditions occurred the robot would perform the *CRUISE* behavior, which means the robot moves forward in whatever direction it is in. This behavior had the lowest priority and would run if there were no objects detected, or not light to follow (as mentioned below).

The *Photo\_sense()* function addresses the objective of adding a light sensing behavior to the robot as requested by the second part step 1 of the handout. The function has a sampling period of 125ms in which a timer is initialized to inhibit the sensor to sample during the active timer period. The sampling is done by measuring the voltage of a photo-resistor using an ADC GPIO pin for both the left and right sensors. This sensor data is then utilized by the *Light\_follow()* function. The function first checks if the average light intensity of the sensors is greater than some threshold. The following equations provide the scaling and difference to get the motor to turn the appropriate direction given light intensities. These equations inhibit the robot to turn left when the left reading is greater than the right, and turn right when the right reading is greater. Additionally, the acceleration is set to 400 for each motor for smoother movement.

|  |  |  |
| --- | --- | --- |
|  |  | (1) |
|  |  | (2) | |

A *LIGHT\_OBSERVE()* function was also implemented as part of this lab. This function would first get the average reading of the photo-resistors sensors. If that average was greater than some threshold value (2.50) it would perform the behavior. The behavior first stops the robots, sets a timer for 3 seconds, backs up the robot, turns the robot left by 90 degrees, then updates the motors velocity and acceleration for future motor actions.

For debugging purposes the *info\_display()* function was utilized to display text denoting the current state that the robot was in. Additionally, the robot would blink the red LED when the robot was in the photo-resistor sampling state.

# **Results**

When it came to determining the sampling interval for the *IR\_sense()* function in Part 1 of the lab there were various intervals that were experimented with. As one would expect when the sampling interval was increased the robot would allocate more of its time simply sampling the sensor readings. This in turn prevents the robot from spending that time exploring. When an interval of 2 seconds was used the robot would have a quite slow reaction time to sensor inputs and would run into objects. The most appropriate interval was found to be around 125ms since this enabled the robot to process other sensors while still being responsive enough to not run into obstacles.

Step 4 in Part 1 of the lab handout was to enable the robot to cruise around the room while still avoiding objects. The implementation is discussed in detail in the Source Code Discussion section so only a brief overview will be mentioned here.The *IR\_avoid()* and *Cruise()* functions were carried over from the previous lab. The *IR\_avoid* behavior was ballistic to ensure that it would have the highest priority and always avoid objects. The *CRUISE* behavior, having the lowest priority, would only run when no other sensor conditions were met allowing the robot to roam when there was no object detected.

The *ESCAPE* behavior for the bonus of Part 1 was implemented directly into the *IR\_avoid* function. The escape behavior would occur when both sensors detect an object. The robot would then stop, back up, turn 180 degrees, then set future motor actions to move the robot forward. This functionality would ensure that if a robot were facing a corner or directly at a wall it would move in the opposite direction of that object.

The *Photo\_sense()* function outlined in the Code Description section provides the robot a means of determining the light intensity in front of the robot. The sampling interval that was chosen was 125ms and this interval enabled the robot enough time to sense while also allowing time for the execution of the other behaviors. The homing behavior used these intensities to direct the robot towards the light using the photo-resistors as differential detectors. That is, the robot would get the sensor readings, check if the average is greater than some threshold, if so, the difference between the left and right sensor readings would determine the velocity of each robot's motors. This equation for the voltage values is given by **Equation 1** and **Equation 2** above.

The photo-resistor readings were off by about .15V of the 0-5V range. There wasn’t any direct rescaling of the sensor discrepancy in the source code to account for the difference in sensor readings. Although this was the case, this discrepancy had no noticeable effect on the robot's behaviors, and the robot still performed as expected. As stated at the end of the Source Code Discussion section, the robot would display its current running behavior using the LCD display. This was done by whenever a behavior is run, the robot would update the LCD display with that behavior's name.

A diagram of light source

Description automatically generated

**Figure 2**: Behavior Diagram

The behavior Diagram shown in **Figure 2** illustrates the relationship between the robots behaviors and robot resources. The arbiter determines which behavior determines what movement is sent to the motor controller. This is done via a priority structure where the

Highest priority behavior is at the top and lowest is at the bottom (*IR\_avoid*  and *EXPLOR* respectively. The *LIGHT\_FOLLOW* and the *LIGHT\_OBSERVE* behaviors should both be servo behaviors since we won't continuously monitor sensor activity when performing the behavior. When it came to the actual implementation of the *LIGHT\_OBSERVE* behavior this was much more challenging to achieve and parts of this function became ballistic although it shouldn’t to ensure the *IR\_avoid* behavior would still have the highest priority.

# **Conclusion**

The focus of this was to implement behaviors in a behavior based control architecture. The behaviors that were implemented as part of this lab were the *IR\_avoid*, ESCAPE, *CRUISE*, *LIGHT\_FOLLOW,* and the *LIGHT\_OBSERVE* behavior. This control architecture was priority based where the *IR\_avoid* behavior had the highest priority to ensure the robot would not run into objects. The other behaviors allowed the robot to cruise around its environment but also follow a light if the light intensity surpassed a threshold of 2.50.

All in all, the combination of these behaviors allowed the robot to home in on a light source while simultaneously avoiding collisions with objects.