LAB 4

PROJECT EXPLORATION MISSION

There is an error in the design of the simulation GUI compared to operation of the actual CyBot for the 180-degree scan by the sensors.

- **Simulated CyBot (GUI): 0 degrees is on the left, and sensors scan clockwise to 180 degrees on the right.**
- **Physical CyBot: 0 degrees is on the right, and sensors scan counter-clockwise to 180 degrees on the left.**

INTRODUCTION

In this lab, you will conduct a simple mission with the CyBot mobile robot. You have been training with the CyBot platform for several weeks, and it's time to use what you know to accomplish a mission in which the CyBot moves through a test field, detects and avoids objects, and drives to the smallest object. These represent the minimum capabilities you will implement in your final lab project. In Lab 2, you developed code to move the robot. In Lab 3, you developed code to send messages between the CyBot and the PC. In this lab, you are given a precompiled library to find objects. The code in this library scans the area in front of the CyBot, rotating the sonar and infrared sensors 180 degrees using the servo motor. You will learn how to write code that interfaces with the sensors and motor using microcontroller peripherals in later labs. For the project, you will write and use your own code – based on code that you will be writing in later labs. Until then, the FindObjects library will let you start experimenting with the more capabilities of the CyBot in the test field. Play around with the bot and have fun!

You will develop an embedded application that can autonomously: 1) identify the smallest width (tall) object in the test field, and then 2) navigate to that object while avoiding obstacles. To help guide you through this mission, we have decomposed the mission into three parts.

Note: In Lab 3, you programmed a GPIO port. In this lab, you will not directly program the GPIO registers. You will return to GPIO and I/O module programming in Lab 5. The GPIO programming from Lab 3 is a foundation for the rest of the semester and project. Take this week to review GPIO material and reinforce your understanding as needed.

REFERENCE FILES

The following reference files will be used in this lab:

- cyBot_FindObjects.h, header file for pre-compiled library to scan for, find and return objects
- libcybotFindObjects.lib: pre-compiled library for finding objects (note: must change extension of file from .txt to .lib after copying, do not open the file, download/copy only)
- cyBot_uart.h, header file for pre-compiled CyBot-PC UART communication library
- libcybotUART.lib: pre-compiled library for CyBot-PC UART communication (note: must change extension of file from .txt to .lib after copying, do not open the file, download/copy only)
- lcd.c, program file containing various LCD functions
- lcd.h, header file for lcd.c
- timer.c, program file containing various wait commands
- timer.h, header file for timer.c
- open_interface.c, API functions for basic Open Interface functions
- open_interface.h, header file for open_interface.c
- Cybot baseboard and LCD schematics: Cybot-Baseboard-LCD-Schematic.pdf

The code files are available to download.

PRELAB

See the prelab assignment in Canvas and submit it prior to the start of lab.

STRUCTURED PAIRING

You are expected to continue to use structured pairing in this lab and in future labs. It was introduced in Lab 2.

MISSION GROUND RULES

1. Allowed sensors
   a. `cyBot_FindObjects()`, which uses the sonar and infrared sensors on the CyBot
   b. iRobot sensors accessed through Open Interface, including
      i. distance
      ii. angle
      iii. bumpLeft, bumpRight
      iv. cliffFrontLeft, cliffFrontRight

2. Objects in the test field
   a. Short objects: detected by the bump sensors
   b. Holes: detected by the cliff sensors
   c. Tall objects: detected by `cyBot_FindObjects()`. The CyBot should never collide with or come within 5 cm of a tall object.

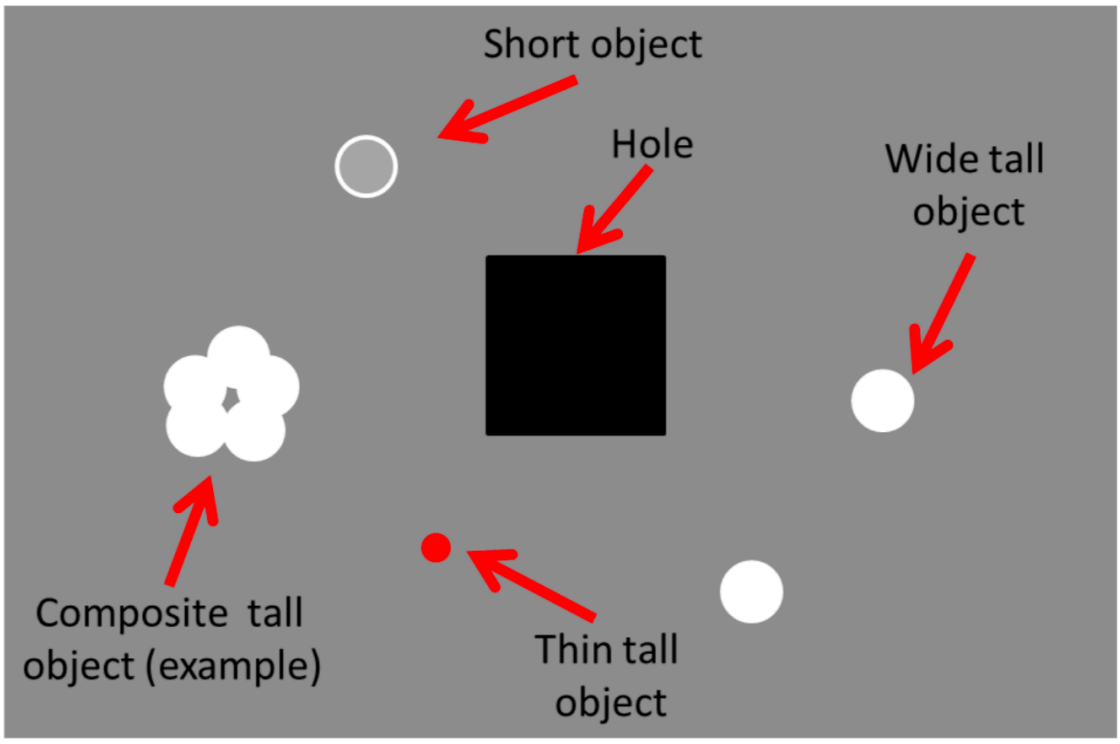

Figure 1: Example test field

For all three parts below, the test field will be populated with 3 to 5 tall objects, one of which will have a smaller width than the others. Additionally, some of the tall objects could be a composite object created by clustering a number of tall objects close together, as shown in Figure 1.

An actual test field used with a real CyBot in the lab project in past semesters is shown in these videos. You may have watched these earlier in the semester. These links are also in Canvas at the bottom of the Lab Resources page.

- https://www.youtube.com/watch?v=ulidN0rs1NA (early generation CyBot)
- https://www.youtube.com/watch?v=OU5m58bhZ6Y (later generation CyBot)

Notice the sensors scanning the region in front of the robot.

Reminder: Copy your CCS project from a previous lab and modify the main program for this lab. As before, the main program will use a while loop that continuously calls functions to implement control, status and data processing tasks in software. Copy/download the new FindObjects library files (header file and lib file). Write new functions as needed to implement functionality for this lab.

For the simulated CyBot: 0 degrees is on the left, and sensors scan clockwise to 180 degrees on the right.

PART 1: DETECT THE SMALLEST WIDTH OBJECT

Assumption for Part 1:

- Objects are located in front of the CyBot so that a 180-degree scan sees all objects. Note: Looking forward from the front of the CyBot, ~~the 0-degree position is to the right, and 180 is to the left~~ **the 0-degree position is to the left, and 180 is to the right**.

Write code to perform a 180-degree scan and display on PuTTY the angle, distance and width information for each object found. Angles are in degrees, and distances are in centimeters. Use the function **cyBot_FindObjects()** from the library provided for you.

Here is a sample display of information in PuTTY:

```
Object#    Angle          Distance   Width
1          85             60         10
2          120            35         5
3          150            70         7
```

Tip: In order to format your data in PuTTY, you can send '\r' to return, '\t' for tab, and '\n' for newline.

Tip: To save data from PuTTY into a file, you can set up "Session > Logging," and under filename, click "Browse" to specify where you want the output saved. Data displayed in PuTTY will then also be written to the file.

In addition to displaying the object information in PuTTY, after the scan is complete, your CyBot should "point" to the object that has the smallest width (turn the CyBot so that the front of the robot where the sensor is located points toward the smallest width object).

Try to write your embedded application so that the CyBot performs its tasks autonomously (i.e., a user is not controlling the robot).

Confirm that the data collected matches the state of the test field.

CHECKPOINT:

The object information is displayed in PuTTY, and the CyBot points to the object with the smallest width.

PART 2: DRIVE TO THE SMALLEST WIDTH OBJECT

Start with the assumption from Part 1. Extend your code from Part 1 so that the CyBot drives autonomously to within 10 cm of the object with the smallest width.

Next, remove the assumption, and let objects be placed at any angle around the CyBot. Update your code so that the CyBot moves autonomously to within 10 cm of the object with the smallest width anywhere around it based on scans from its initial location.

CHECKPOINT:

The CyBot moves autonomously to within 10 cm of the object with the smallest width.

PART 3: DRIVE TO THE SMALLEST WIDTH OBJECT WHILE AVOIDING OTHER OBJECTS

In this final part of the lab, the test field will also be populated with short objects that can only be detected by the bump sensors.

You can let your robot detect both small and tall objects using the bump sensors to help navigate around objects that the CyBot encounters along the way.

Extend your code from Part 2 so that the CyBot drives autonomously to within 10 cm of the object with the smallest width, while navigating around objects it encounters.

OPTIONAL: Don't let the CyBot make contact with tall objects. In other words, don't use the bump sensors to detect tall objects. Detect tall objects before making contact using information from **cyBot_FindObjects()**. Stay more than 5 cm from tall objects.

CHECKPOINT:

Object information is displayed in PuTTY, and the CyBot moves autonomously to within 10 cm of the object with the smallest width, while navigating around objects it encounters along the way.

DEMONSTRATIONS:

1. **Functional demo of a lab milestone –** Specific milestone to demonstrate in Lab 4: Checkpoint for Part 3
2. **Debug demo using debugging tools to explain something about the internal workings of your system –** The TA will announce any specific debugging requirements at the start of lab; otherwise you will create your own debug demo based on your needs and interests in the lab.
3. **Q&A demo showing the ability to formulate and respond to questions –** This can be done in concert with the other demos.