SIMPLE MISSION: OBJECT DETECTION REFINEMENT, AND AVOIDANCE

Introduction

You will extent your simple mission embedded application to a) support toggling between autonomous, and manual mode, b) refine its ability to detect and localize objects, and c) avoid short objects with `bump` sensors as you identify and navigate to the smallest width object in the test-field.

Part 1: Object Detection/Localization Refinement

Previously you calibrated/converted Raw IR values gather with the CyBot Scan function into actual estimates of distance. Here you will use this capability, along with PING sensor measurements, to refine your ability to detect and localize objects.

PING sensor characteristics: Here are some strengths and weaknesses of the PING sensor

- Strengths:
  - Does not need to be calibrated
  - Measurements have very little noise
- Weaknesses:
  - "Cone" of detection is very wide. Compared to the IR sensor its angular resolution is coarser. In other words, if objects are too close to one another they look like one large object.

IR sensor characteristics: Here are some strengths and weaknesses of the IR sensor

- Strengths:
  - Has a narrow "Cone" of detection. This means it can distinguish between objects even if they are close to one another. This is referred to as having a fine-grained angular resolution.
- Weaknesses:
  - Needs recalibration if conditions change: For example, a different CyBot is used, the material of the objects being detected changes.
  - Noise: A number of factors contribute to the measurements made with the IR sensor being nosier than with the PING sensor. Averaging multiple measurements at a given scan angle is a good way to reduce the noise of IR sensor distance measurements.

The following approach is recommended for combining the strengths of these two sensor types to improve your ability of detecting and localizing object in the test-field.

1. Make a scan from 0-180 (or 360, see bonus) degrees. Take multiple measurements at each angle scanned
2. At each angle you measure average the IR distance values
3. Use the IR distance values to detect the angle of the starting and ending edge of each object. This is sometime called Object Detection, or Identifying Distinct Objects
4. Make a second scan that points the PING sensor at the mid-angle (i.e. halfway between the angle of the starting edge, and ending edge) of each detected object, then measure the distance of that object using the PING sensor

Checkpoint: Demonstrate your refined object detection and localization that points the sensor to the smallest width object. Display your results in PuTTy, and graphically.

Part 2: Extend your Simple Mission embedded application to autonomously navigate to the smallest width object.

Using your refined object detection capability from Part 1, extend your Simple Mission embedded application to **autonomously** identify and **autonomously** navigate to within 5cm of the smallest object in the test-field without hitting it.

Checkpoint: Demonstrate your Simple Mission Embedded Application autonomously identifying and autonomously navigating to within 5cm of the smallest width object in the test-field without hitting it.

Part 3: Integrate your UART functions

For the remainder of the semester, you will no longer be allowed to use the pre-compiled UART functions we provided you. Previously, you developed your own code to initialize and use the UART device to send/receive information to/from your CyBot. Integrate into your extended Simple Mission embedded application the UART code you developed. Make sure that your code for the CyBot receiving UART data is **non-blocking** (i.e., it does not wait forever for a byte to arrive). Also, it is recommended that you do **not use interrupts**, as this may initially make debugging the integration with your Simple Mission application more difficult.

Checkpoint: Demonstrate that your extended Simple Mission works using your own version of the UART functions for Initializing the UART, and for sending and receiving data.

Part 4: Update your Simple Mission to switching between autonomous and manual mode, and avoid short objects using `bump` sensors.

Often autonomous embedded systems may find themselves in a situation they are not able to deal with. For such situations, it is useful to allow a human to take over control at least briefly to get back to a situation where the embedded system can handle things autonomously.

Update your extended "Simple Mission" embedded application so that you can switch back-and-forth, **as you deem is needed**, between autonomous mode, and manual mode, to get **within 5 cm** of the smallest width object without hitting it. **Additionally, the test-field will be populated with short objects that can only be detected by the CyBoT `bump` sensors.** You must use the CyBot `bump` sensors to detect, and then move to avoid short objects encountered during navigation.

Autonomous Mode:
In Autonomous Mode, the CyBOT will automatically scan and move. Use the 'h' key to tell the Cybot to scan and autonomously move based on that scan in the following manner: 1) Send 'h' to have the Cybot scan, and return to your PC what movement it plans to execute, 2) Send 'h' again to have the Cybot execute its planned movement, perform another scan, and return its next planned movement, and 3) Keep hitting 'h' until the Cybot has completed its goal. Transmit the data of each scan to your PC to monitor the Cybot's progress.

Manual Mode:
In Manual Mode, the user drives the CyBOT using the 'w','a','s','d' keys of the keyboard, and uses the 'm' key to have the CyBOT make an 180 degree scan. The user needs to be able to interpret the sensor data (using the text sensor data, and a graphical view) to manually navigate the Cybot. During manual driving mode only sensor data can be used for navigating the CyBOT (i.e., the user is not allowed to look at the CyBOT).

Toggle:
The user should be able to freely switch back and forth between Autonomous and Manual Mode at any point during the mission. For example, the 't' key could be used to "toggle" between these modes.

Checkpoint: Demonstrate that your Simple Mission lab allows switching between Autonomous and Manual Mode as the CyBot navigates/is navigated to within 5cm of the smallest width object without hitting it. **Your lab mentor** will have you switch between modes **at least once** to check that you can manually interpret the sensor data to navigate, and that the CyBOT can autonomously navigate to some degree. **Your lab mentor** will ensure that your CyBot encounters a short object at least once.

Bonus:

Bonus 1: 2 bonus points for demonstrating your Simple Mission embedded application can be effectively used in autonomous mode-only, manual mode-only, and toggling between the two modes. Where effectively for 1) autonomous mode-only means your CyBot can get to within 5cm of the smallest object within 1 minute, 2) manual mode-only means you can manually get to within 5cm of the smallest object with-in 2 minutes, and 3) toggling between the two modes means you can get to within 5cm the smallest object with-in 2 minutes **as the TA indicates** the times at which you should toggle modes. A **short object must be encountered** during each scenario (autonomous-only, manual-only, and toggling between the two).

Bonus 2: 1 bonus point if your Embedded Application can **autonomously** identify, and point the sensor to the smallest width object when objects are placed 360 degrees around the Cybot. In other words, instead of performing a 180 degree scan, you need to be able to scan 360 degrees. You do not need to navigate to the identified smallest width object, but it would be cool if you could!