Homework 2 - due 9/17/2003

15-491: CMRoboBits

1 Introduction

You will learn how to subscribe to sensor messages, read data from the gsensor and foot pads, set the LEDs in the robot's face, and issue a simple motion command. It is also important to note the location of the header files that we use because they are valuable reference sources for additional information.

2 Background

You will need to access sensor data as a part of this lab. To do so, you subscribe to updates from the "SensorData" event processor. This is just like we went through in class for FeatureSet in the SpinDog behavior.

This subscription will provide a SensorData object with up to date information from the robots buttons, foot pads, and gsensor. The declaration of the SensorData class can be found in dogs/agent/headers/Sensors.h and /dogs/agent/shared_code/Sensors.cc. The header file is more interesting for the purposes of this assignment. Specifically, the fields of the SensorDataFrame structure and the SensorData::getFrame() method.

Recall that sensor frames arrive at 125hz while the camera operates at approximately 25hz. This means that there are multiple sensors frames available for each camera frame. In this lab, we will only worry about the most recent (unless you want to get more complicated - the assignment can be done using only the most recent frame).

To access foot pad data (assuming you have a pointer to a SensorData object and that you have included the file "../headers/Sensors.h" at the top of your source file so the relevant structures are available):

- bool pad_val = sensor_data_object_ptr->getFrame(0)->paw[foot_number]

The getFrame method takes an integer telling it which sensor frame you are interested in (because there will be several possible ones for the current vision frame). A value of 0 means use the most recent. A value of -1 means use the frame before the most recent frame. And so forth.

The paw field of the SensorDataFrame structure that is returned by getFrame is an array of 4 boolean values. Offsets 0 and 1 are for the left and right front legs respectively. Offsets 2 and 3 are for the left and right rear legs.

You will also need data from the gsensor to complete this lab. This information is also found in the SensorDataFrame structure that is returned by getFrame. The accelerations are contained in a vector3d structure named accel. The vector3d class is an instantiation of the template found in "dogs/agent/headers/gvector.h". It has lots of useful utility methods available. However, you won't need any of them in this lab; you'll just want to access the individual fields of the vector. They are named x, y, and z. As a concrete example, the find the value of the acceleration along the robots z-axis in gravities, you would use the following:

- double z = sensor_data_object_ptr->getFrame(0)->accel.z;

You do the same for x and y. Recall that x is from the axis from the front of the robot to the back. Positive x is in front. Y is from left to right. Positive y is to the left. Z is up and down. Positive Z is up.

Finally, you will need to fill in a MotionCommand structure to send motions to the robot. This structure is defined in "dogs/agent/Motion/MotionInterface.h". The relevant fields are motion_cmd, which you must set to MOTION_WALK_TROT, and vx, which you should set to a possitive value to move forward. You will also need to set the led field (which is a bitmask) by ORing together LED constants. Three additional constants, MAX_DA, MAX_DX, and MAX_DY may be of use. They are the maximum velocities the robot can actually achieve when rotating and translating.

3 Procedure

- Go to your dogs directory and run the "cvs update" command to retrieve an updated walk and source code additions. Be sure to do a "stickit -a" in order to move the new walk to your memory stick (after you compile). Be aware that this will also overwrite run.cfg, so you may need to edit that file again. It is located in /memstick/config.
- Create a new behavior called "FootDog" in dogs/agent/Behaviors using the files SpinDog.h and SpinDog.cc as a template. This new behavior should act in the following way:
  - Set LED_LOWER_LEFT when the left front paw pad is depressed
  - Set LED_UPPER_LEFT when the left rear paw pad is depressed
  - Set LED_LOWER_RIGHT ... ... right front ... ...
  - Set LED_UPPER_RIGHT ... ... right read ... ...
  - Walk forward in a straight line at 1/2 max velocity.
  - The robot should stop walking when it's lifted off the ground.
  - Set LED_MIDDLE_LEFT when the robot is off the ground and tilted to the left.

  - Set LED_MIDDLE_RIGHT when the robot is off the ground and tiled to the right. Neither of the middle LEDs should be set while the robot is on the ground.

- Remember to include "../headers/Sensors.h" in your behavior. You may also need to add "../Motion/MotionInterface.h" if it is not already present in order to use the LEDs.
- You MUST add your .cc file to dogs/agent/Main/Makefile in order for your code to be compiled. Find the section with behavior sources in it and made a new entry using that same format.

4 Questions

Answer the following using *no more than 3 sentences* for each answer.

- How did you detect that the robot was laying on its side?
- In class we used the example of maintaining an equilibrium distance from an object as an example of a place where hysteresis using two separate thresholds would be useful. This was actually a poorly chosen example. Explain why.

5 Grading

- Setting 4 LEDs for footpads - 4 pts
- Stopping the robot when it is lifted - 2 pts
- Setting 2 LEDs when the robot is tilted - 2 pts
- Questions - 2 pts