Introduction to Anita Borg, the Barbie-Jeep Robot

Comp 380

Susan Fox

February 17, 2014

1 Overview of Borg

Borg was the class project in a previous version of this class. It was modified by Ken Moffett and then programmed by the students in the class. The modifications included:

- Installing a linear actuator, a motor that extends and contracts a straight bar. This is attached to the bar that the original car used to control its steering.
- Installing fuses and wiring to connect the sensors and controller boards to each other and to the car's own motor.
- Installing an on/off switch to turn the system on and off, and to permit emergency shutdown as needed.
- Installing front-facing sonar sensors (mounted in model airplane jet engines).
- Installing an inexpensive web-cam on the front of the robot. This should really be replaced at some point.
- Installing an Arduino Diecimila board to control the motors and sonar sensors. This board can be programmed in a variant of C, with the resulting code compiled into byte code and downloaded to the board.
- Installing a motor controller board. It takes its commands from the Arduino. I believe it is a Trossen Sabertooth board, but I've forgotten the details. There are a few issues with communications between the Arduino and the motor board that could make a nice project.
- Installing a clear plastic, hinged tray to set a laptop on. The laptop serves as the robot's brain; camera and GPS can plug directly into the laptop, as does the Arduino board.
- Building a wooden dolley to roll the robot around. Now that its wheels are robot-controlled, it can no longer roll freely.
- Building a cardboard cover for shading the laptop when outdoors in the sun.

In addition, we also purchased a compass, which was not installed (could be a nice project).

The software to use the robot is installed on the larger Dell laptop in the robot lab. This laptop, like the big desktop machine, has a `macalester` username, with the password `r0b0tics` (zeros instead of o's).

I have had troubles getting the laptop to stay on the Macalester network without constantly asking for the password. Let me know if you have troubles.

All of the software written for this robot is several years old, and could stand to be updated. A possible project for some team!

2 The Arduino board

You don't have to modify the programming for the Arduino board; it really is a task for someone who likes digging into the guts of computers. The program to use in writing Arduino programs is called `arduino`, and the newest version we have is in the macalester user's home directory, in a folder called `arduino-0016`. Test programs for our board are stored in a folder called `sketchbook`, and the final version of the code is in a folder called `microcontroller`.

The program for the Arduino expects text commands to control the motors, and it returns text containing the sensor values. This makes it easy to communicate with from a Python program.

3 Using the Camera

The robot's camera connects using USB. We wrote a program in OpenCV 1 to capture images from the robot's camera. I am going to try to get it running this week, but in the meantime avoid using the parts of the code that require the camera. As for the motor/sensor controller, the camera program expects to communicate through text interactions, with either a human user or a Python program.

4 Borg's Brain

In a folder called `Brain` on the desktop of the `macalester` account, there are two Python files: `Brain.py` and `GoBrain.py`. The `Brain.py` contains some classes to operate in separate threads and perform underlying tasks (communicating with the camera, or the Arduino, or logging what is going on). It also contains a set of functions, sadly weakly commented, for controlling the robot.

The `GoBrain.py` file contains some sample programs that were written for the robot. The `manual` program allows you to tele-operate the robot. Most of the other options require the camera to be working.

5 Tasks to try out

1. Run the basic program to communicate with the Arduino board. It is called `robot_control` and it is in the `microcontroller` folder. Make sure that the robot is turned on, and the USB cable from the Arduino is in place. This program once running, will print the sonar sensor values, and will take input in a very specific form to control the motors:

`S 000` S says to change the steering angle. There must be one space, and then three digits, ranging from 000 to 255. Zero means full to the left, 255 means full to the right, and 128 is straight forward. Then hit return.

`F 000` F says to set the motors going forward. There must be one space and then three digits, ranging from 000 to 255. Zero means stop, 255 is full speed ahead. Be sure to hit return.

`R 000` R says to set the motors going in reverse. There must be one space and then three digits, ranging from 000 to 255. Zero means stop, 255 is full speed in reverse. Be sure to hit return.

The program is balky; I am not quite sure why we start getting bad data. But it seems to happen when the Arduino gets out of sync with the motor controller board, and then stops sending data by USB.

2. Try running the `GoBrain.py` file, and select the manual option. Try tele-operating the robot. How does it do?

   The options for the manual program are:

   `f` Set the motors going forward at one tenth speed (0.1)
   `b` Set the motors going backward at two tenths speed(-0.2)
   `r` Turn the steering 0.3 more to the right (1.0 is full right)
   `l` Turn the steering 0.3 more to the left (-1.0) is full left)
   `c` Stands for "center," reset the steering to 0.0, center
   `s` Stop the motors
   `q` Quit the manual control program

3. Write a program using the `Brain.py` file to control the robot. Your program should have the robot move in a circle for a fixed amount of time. Note that it takes two people to safely load and unload the robot from the dolley. When placing the robot on the dolley, make sure that the robot is not resting on the linear actuator, but rather on its front bumper.

4. Write a second program to use the `Brain.py` file. Experiment with the robot's sonar sensors; see if you can make the robot roll forward until it detects an obstacle with the sonar sensors. Write the program carefully, and then report whether or not it worked.