CS 4970/5973
Intro Robotics

Amandeep Gill
Joshua Shuller

Team 2 / Project 2

Robot Code Documentation

Code Overview:

The robot code for this project is basically behavior based, basically. The robot hardware had two rear drive wheels, one front steer wheel, and a rotating platform holding a range sensor and a light sensor, which we called the radar because it rotates. See the hardware document for full hardware details. The software made the robot drive towards the brightest light, and avoid obstacles while doing so. If one of the touch sensors hit something, the robot would simply back up, forget everything, and try again.

Code Detail:

The colors in this document have no meaning whatsoever. It's just that different colors emphasize that something is directly from the code and same colors are used to stress a relation. The code was broken down into the following functions:

- `void steerTo(int degrees)` - causes the steering servo to turn to the specified degrees in the range of –90 to +90; where negative degrees is left, zero degrees is center, and positive degrees is right.
- `void radarTo(int degrees)` – causes the radar servo to turn to the specified degrees in the range of –90 to +90; where negative degrees is left, zero degrees is center, and positive degrees is right.
- `int getRadarRange()` – returns the analog value from the ET range sensor on the radar. This function maybe should not exist due to it's simplicity, but it was separated out just in case something more complicated was ever needed for getting the radar light. The function was not removed from the code because it is time-sensitive. The handy board is so slow that the radar would have been too fast to read the range and light sensors without using a function to do so and then the radar function would begin getting erroneous readings.
- `int getRadarLight()` – returns the analog value from the light sensor on the radar. Same story as with the above function `getRadarRange()`.
- `void radar(void)` – the function defining the radar process. Makes the radar rotate and puts the sensor values read into the input slices. The radar process has 4 distinct modes:
    - `RADAR_OFF` – radar does not move.

- `RADAR_SCAN_NORMAL` – radar scans back and forth, going from left-to-right, and then from right-to-left back to it's original position. This mode is normally used while in normal operation.
- `RADAR_SCAN_LEFT_TO_RIGHT` – radar goes to the far left, and then scans from left-to-right, and then quickly goes back to the far left again without scanning. This is used while making sharp turns to the right.
- `RADAR_SCAN_RIGHT_TO_LEFT` – radar goes to the far right, and then scans from right-to-lelft, and then quickly goes back to the far right again without scanning. This is used while making sharp turns to the left.

To ensure that the radar is at the specified position when reading the sensors, the radar process actually polls the sensors several times to delay the software, and to get the maximum value for that slice. The number of times that the radar sensors are actually polled while waiting for the servo to move to the correct angle is defined by `RADAR_SCAN_DELAY_BY_POLLING_COUNT`. For the light, the minimum reading is stored for that slice, and for the range, the maximum reading is stored for that slice.

- `void doFullRadarScan()` – does not return until at least one full scan from either left-to-right or right-to-left of the radar has been completed.
- `int isObstacleAt(int slice)` – returns if there is an obstacle present at or near the specified slice. In specific, if the range values for any slice within ½ of `MIN_PASS_SLICE_COUNT` slices away from the specified slice is above the range value of `RADAR_RANGE_AVOID_LEVEL` is found, this function returns true (`1`); otherwise, this function returns false (`0`).
- `int findObstacleFreeSlice(int targetSlice)` – starts at the target slice, and finds the closest slice to the target slice that is free of obstacles as returned from the `isObstacleAt()` function.
- `void steerToSlice(int slice)` – causes the steering servo to steer towards the specified radar input slice.
- `void main(void)` – defines the main functionality of this robot. It initializes everything, and then goes into the main program loop. This loop continues until the stop button is pressed. Inside the loop, it just seeks the brightest light. If the fixed ET sensor on the bottom front of the robot detects something, the robot stops, makes a full scan to determine if there is something close enough to worry about, and to see which direction to go if there is something there. If the fixed ET sensor on the bottom reads nothing, however, then the robot continues as normal, just going towards the brightest light, avoiding obstacles on the sides.