Robot Code: Group 2 – PROJECT #1

Prateek Duggal
Jonathan Siegel
Kumaresh Rajan

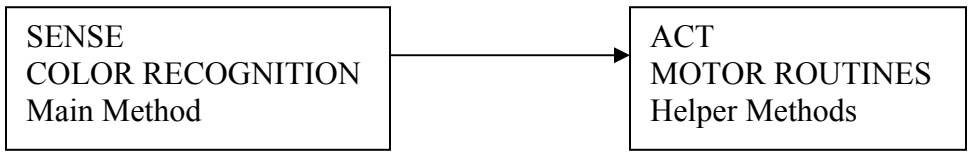
Introduction:

After testing the responsiveness and accuracy of the robot using Interactive C it became clear that simple algorithms were needed to operate the robot. Extra complexity in the code proved to be more of a hindrance than a benefit because of the unpredictability of the equipment in the robotics kit. Also, debugging the robot would have proven to be much more difficult with a complex design because it would be harder to tell if the robot was behaving incorrectly due to bad programming logic or if the problems were caused by inaccurate sensor readings. In keeping with this philosophy of simplicity it was decided that each module would only contain the bare essentials. For example, in doing the color recognition extra cmucam library functions like setwin() and variables such as track_x were not included in the code because these extra features caused the robot to behave unexpectedly.

Data Structures:

No data structures were used in the source code.

Functions:



main()
The main method in the code was responsible for sensing the environment (i.e. color recognition) and calling the appropriate helper function. The entire color recognition module was place in the main method and "if conditionals" were used to select the appropriate actions. Color was sensed using the trackRaw() method and if a high enough track_confidence was found then a helper method would be called to change the position of the robot, else the default go straight action would be preformed. The main method would terminate after the yellow color was found by setting the yellowfound variable to equal one.

forwardclicks()
Routine that is called after the robot has found a color. The routine moves the center of the robot over the colored square in order to align the robot with the maze.

reset_encode()
This method when called will reset both the left and right slot sensors to zero.

calc_turn()
Method returns the sum of axle rotations for the left and right wheel.

corr_err()
Note this code was taken from group 1 from the spring 2003 term of intelligent robotics. The code was modified to suit the needs of our robot.
Code calculated which motor of the robot rotated the fastest using the encoder values from the slot sensors. If the difference of the motor outputs for the robot is greater that the TURN_TOLERANCE (10 axle rotations) then the code would try to realign the orientation of the robot.
The code was never implemented in the final version of the robot because of the difference of power levels in the robot's motors (i.e. because the left motor was always stronger then the right motor the code would always indicate left drift even if no drift occurred)

forward()
Code tried to make the robot move in a straight line. The motors were allowed to pulse for .37 seconds and then went into idle for 1 second. The pulsing allowed the viewing window of the camera to examine every inch of the floor, and idling the motors gave enough time for the camera to sense the colored square before the motors began to move again. The "proper" power levels for the motors were found after intensive testing of the robot. In theory setting the power levels of each motor to the same value would allow the robot the to move in a straight line, but this was not the case for our robot due to the huge power difference between the left and right motors. In order to compensate for the power difference the left motor power level was made to be significantly less than the power level of the right motor.

reverse()
When routine is called the robot will perform a right turn for 180 degrees. Both slot sensors are reset and the motors are allowed to turn until the sum of the motor rotations equal TURN_TIME_REV (266 axle rotations). corr_err() was supposed to called to compensate for drift, but given the inaccurate readings of the motors the method was commented out.

left()
When routine is called the robot will perform a left turn for 90 degrees. Both slot sensors are reset and the motors are allowed to turn until the sum of the motor rotations equal TURN_TIME_L (120 axle rotations). corr_err() was supposed to called to compensate for drift, but given the inaccurate readings of the motors the method was commented out.

right()

When routine is called the robot will perform a right turn for 90 degrees. Both slot sensors are reset and the motors are allowed to turn until the sum of the motor rotations equal TURN_TIME_R (134 axle rotations). corr_err() was supposed to called to compensate for drift, but given the inaccurate readings of the motors the method was commented out.