Group 10

Amit Maole
Brent Eskridge
Klo Utley
Tony Lopez

PROJECT 3: Hardware Final Report

Design:

The design criterion for Project 3 was threefold: build it sturdy (the robot from Project 2 fell apart too easily), drive straight, and make accurate 90^o turns. In order to accomplish the first criteria, we started with the motors and build a box-type chassis with multiple beams vertically mounted to brace the sides. The Handy Board was seated inside and secured with black Lego pins. To accomplish driving straight and accurately turning, we went with a 4-wheel design, each side powered by its own motor. Attached to each motor itself is a 24-tooth gear. This gear is flanked by two 40-tooth gears, each of which is mounted on the same axle as a wheel. To help with the accuracy problem, a 24-tooth gear is meshed to the front 40-tooth gear on each side. Attached to this gear is a pulley wheel and encoder.

Since using a claw-type device to grab the target cubes might possibly add both durability and implementation issues to the robot, we decided instead to add a cradle consisting of two beams extending 3" from the front of the robot and spaced 4½" apart. Also mounted to each of these beams were a large post touch sensor and an IR sensor. The large post sensors are to detect a collision with the blue robot (the path planner should prevent the robot from running into a wall). The IR sensors are used to detect the presence of the destinations outlined in black electrical tape.

The CMUcam was affixed to the top of a servo and the servo was mounted in the vertical position on a ledge built into the chassis behind the target cradle. The servo was calibrated before being mounted to allow the CMUcam to be rotated a full 180^o. This was to allow the robot to track the target cubes and only turn when the cube was 90^o to the left or right of the camera.

Testing/Anomalies:

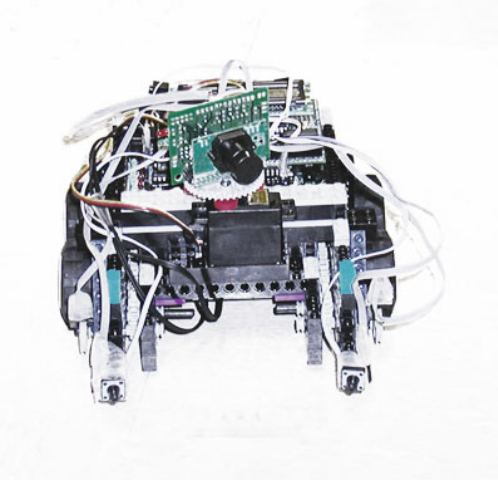
The first testing task was to make the robot drive in a straight line and consistently stop at a specified distance. This was achieved by used the encoders on both sides of the robot. If the left encoder read more events than the right encoder, the power was boosted on the right side and decreased on the left. The same was true in reverse for the right encoder reading more events than the left one. It was determined experimentally that there were approximately 116 encoder events per foot traveled. We factored in a tolerance of 6 encoder events, leaving the robot to drive until it detected the number of feet to travel * 110 events. This feature worked flawlessly. After traveling several feet, the error of distance was less than ½".

The next testing task and ultimately the most difficult was the turning algorithm. Regardless of how we approached the problem, the robot failed to return consistent results. Repeated trials of the same settings also failed to be consistent. In the first trial, it might turn 90^o dead on, but on the next (without changing any settings) it would turn 100^o or 75^o. We tried

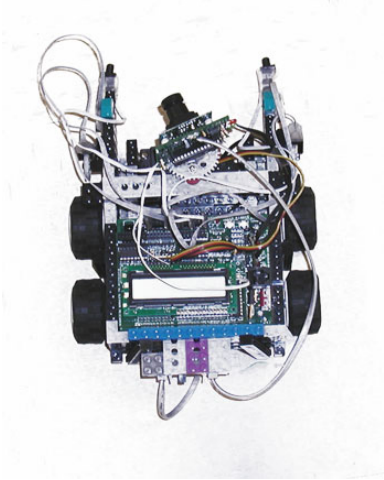
pulsing the motors and then later tried ramping the motors to reduce slipping and prevent jerking. We tried to use the encoders to provide precise measurements of how far we had traveled, but that also didn't work. We tried different treads on the white hubs. This inconsistency turned out to be the Achilles heel of our robot.

The last testing task was to verify that the path planning algorithm parsed the coordinate data correctly and choose appropriate paths to targets and to destinations. This part of the project worked great.

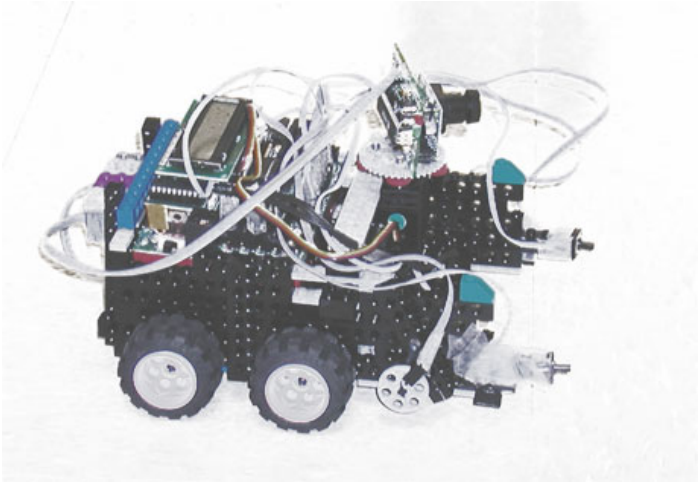
Since we spent so much time trying to troubleshoot the turning problem, we didn't get to a few of the tasks we originally wanted to. Our initial design called for using the CMUcam to verify that there was a target cube at the specified location and to assist the robot in making small adjustments to its planned path that might be the result of a shifted cube or one whose coordinates were not given.



(Figure 1)



(Figure 2)



(Figure 3)