Team 4
Project #1
David Rothell
Jeff Williams
Kevin Royall

Team 4
David Rothell
Jeff Williams
Kevin Royall

Robot Design

The main design scheme of our robot centered on the use of tracks. We felt that tracks would give us better turning and traction, as compared to wheels. Each track would have its own motor, that way we could drive the tracks in different directions so that the robot would turn on a central pivot point. Each track consisted of 4 "wheels" with the bottom having a bigger base than the top. (See Figure 1.) Again, we felt that this would help in traction and turning.

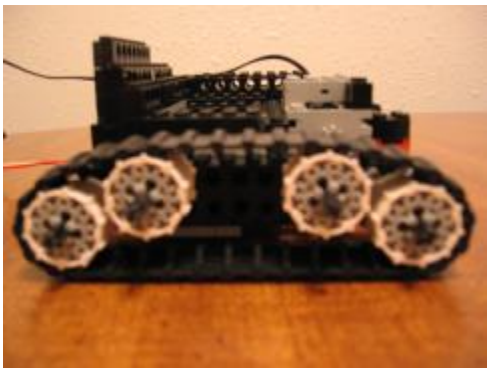

Figure 1

Once we had an overall layout for the track, we then work on attaching the motors to the tracks. The motors had to be geared down to help turn the tracks. In the end, we ended up using a 9:1 gear reduction. (See Figure 2.) Using a 9:1 reduction with two motors gave us all the power that we needed. The robot was able to move at a steady pace and made excellent turns.

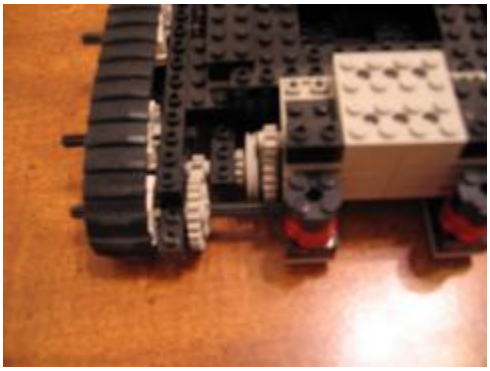

Figure 2.

The two motors were mounted side by side with the attached tracks. Additional LEGOs were used to create a floor for the motors and to add support. This gave our robot a nice wide stance to help with stability and alignment. I small hole was left in the floor so that the camera could be mounted in the center of the robot. (See figure 3 and 4.) This added two benefits to the design. One, the robot did not have to move forward or backwards after recognizing a color, since it was directly over the color. Two, the camera could later on be covered by the handy board to conceal it and give it a stable lighting environment. It turned out that we needed an LED light to give the CMU cam enough light to work with. After, the addition of an LED we got consistent results with the CMU cam.

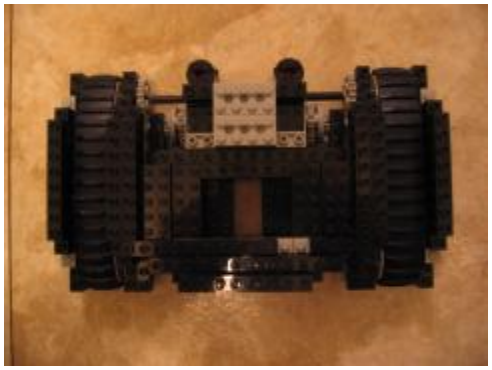

Figure 3

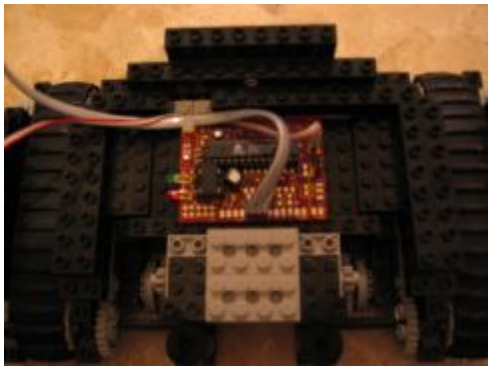

Figure 4

Once the camera was mounted, the handy board could be placed on top to finish off the robot and conceal the camera from view. (See figure 5 and 6) Our team felt that the finished robot met all of our original goals and functioned quite well.

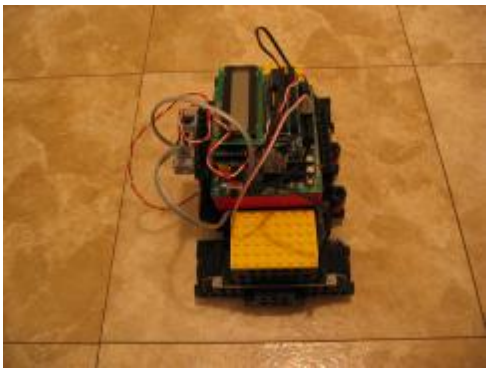

Figure 5

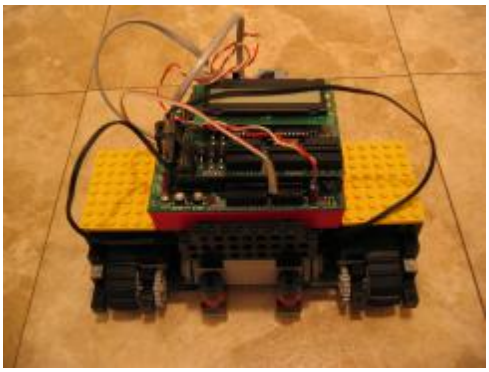

Figure 6

```
#use "cmucamlib.ic"
#define GREEN 0
#define PINK 1
#define BLUE 2
#define ORANGE 3
#define YELLOW 4
#define FLOOR 5
#define BLACK 6

int RIGHT_FORWARD = 100; //#defines for the motors
int RIGHT_BACK = -100;  //we had to use different values
int LEFT_FORWARD = 68;  //to get the motors to drive straight
int LEFT_BACK = -68;

int colors[7][3] =
{
    {60, 60, 0},  //green
      {220, 40, 60}, //pink
      {40, 40, 220}, //blue
      {200, 40, 0},  //orange
      {100, 80, 0},  //yellow
      {140, 100, 20}, //floor
      {120, 20, 120}//black
};

void forward(){
    motor(0,LEFT_FORWARD);
    motor(1,RIGHT_FORWARD);
}

void back(){
    motor(0,RIGHT_BACK);
    motor(1,LEFT_BACK);
}

void right(){
    motor(0,LEFT_FORWARD);
    motor(1,RIGHT_BACK);
}

void left(){
    motor(0,LEFT_BACK);
    motor(1,RIGHT_FORWARD);
}


void main(){
    int conf = 0;
    int gconf = 0;
    int color = 5;
    int i;
    int exit = 0;
    bit_set(0x1000,0x40);     // turn on the LED
    init_camera();            // initialize the camera in YUV mode
    clamp_camera_yuv();       // clamp camera white balance in YUV mode
    while(!stop_button() && exit == 0){
        gconf = 0;

        while(trackRaw(colors[FLOOR][0],(colors[FLOOR][0]+20),
                       colors[FLOOR][1],(colors[FLOOR][1]+20),
                       colors[FLOOR][2],(colors[FLOOR][2]+20)) > 85){
            //do nothing
        }
```

```
        // we found something other than floor
        // so we trackRaw all the colors and see
        // which one it is most likely
        for (i = 0; i < BLACK+1; i++)
          {
            if (i != FLOOR)
              {
                conf = trackRaw(colors[i][0],(colors[i][0]+20),
                                colors[i][1],(colors[i][1]+20),
                                colors[i][2],(colors[i][2]+20));
                if (conf > gconf)
                  {
                    gconf = conf;
                    color = i;
                }
            }
        }

        if (color == BLACK)
          {
            //do nothing
        }
        if (color == GREEN)
          {
            forward();
        }
        if (color == PINK)
          {
            left();
            sleep(6.0);
            forward();
        }
        if (color == BLUE)
          {
            right();
            sleep(3.1);
            forward();
        }
        if (color == ORANGE)
          {
            left();
            sleep(3.0);
            forward();
        }
        if (color == YELLOW)
          {
            beep();
            beep();
            ao();
            exit = 1;
        }
        printf("Found %d with conf of %d\n", color, gconf);
    }
}
```

Team 4
David Rothell
Jeff Williams
Kevin Royall

Robot Code Documentation

Code Documentation

Our code was an attempt at KISS code, so we used a single process. We could not depend on the scheduler of the Handy Board OS to return control to the needed functions in a timely manner and we did not want to write userland locking. Either way would not have been very effective or easier to implement as the single process.

Constant Definitions

In our one process, we had several constants, dictating what each color our camera was expecting to see – YELLOW, GREEN, PINK, BLUE, ORANGE, FLOOR, and BLACK. Corresponding to each of these defines we had an array row with 3 color data points for RGB mins (RGB maxes were min + 20). The color lookup table was generated using a brute force algorithm. We also had 4 constants to determine the strength of the two motors to move forward and backwards. The motor strengths were modified until they gave satisfactory results. One motor was found to be faster, so it was ran at 68% of the other motor's speed.

Functions

We used 4 secondary functions to control the motors and the main function to control the flow of the program. The 4 functions forward, back, left and right each made two Handy Board OS calls to the motor, depending on the desired effect. The main function gears up

the Handy Board: turns on the LED underneath, initiates the cam and calibrates it and jumps into a while loop that only exits if the stop button is pushed or yellow is detected by the camera. Then while in the while loop it jumps into another one that polls the camera with a cmucamlib call to trackRaw. Using the previously mentioned color lookup table, we run through each of the 6 colors, and if we get a confidence level back from trackRaw, greater than zero and are not currently on the FLOOR, then we exit out of the while loop and run through a series of IF statements that depending on which color for which we were last looking, we either moved forward, backwards, performed a u-turn, turned left or right, or come to stop. We had originally planned to use a switch statement for this portion, but interactive c did not implement it. After going through the color tests, then we proceed back into the color checking code.

Team 4
David Rothell
Jeff Williams
Kevin Royall

Team Organization Evaluation and Plans

Meetings

As a whole our team worked really well together, we met frequently and were productive at the meetings. Although each of us had busy schedules and were under a lot of pressure to meet deadlines, we worked effectively to resolve outside problems to come together to efficiently design and implement our robot. This as a whole we did effectively. At meetings we put forth ideas to one another, wrote code, and designed the final product. We do not feel the need to increase the amount of effort put forth were we given a chance to do it again.

Individual Work

Another positive aspect of our group was that we were very good at implementing individual work outside the group. Small, individual orientated tasks we delegated to each other, tasks that would be better suited without group distractions. These individual tasks included creating the basic structure of the robot, testing of brute force methods to discover the colors (we tried this in a group environment, and ended up sitting around staring at the Handy Board), brainstorming, and small code revisions. We felt this was an effective way of managing our time, and so we will likely choose to complete future projects using this allotment of work.

Robot Difficulties

The real determining factor in our failure to complete the course was overestimating the capabilities of the Handy Board and the effectiveness of the sensors. By the time we discovered that the cmu cam was ineffective at quickly determining the colors, we had already designed a slow-and-steady machine that could ill-afford to adjust itself to the playing field and stop and attempt to read the colors inside the boxes. It was not until we saw the test course the Thursday before it was due that we understood that the black electrical tape was surrounding the tile rather than the two inch color square. Knowing this we could have implemented a better system to handle the colors. If we had more experience with the cmu cam we would have modified our robot accordingly to adjust for the slow detection time.