Project 1 – Sensing and Movement

Group 1
Jeremy Branecky
Camilo Reyes
Stephen Mckinney

Overview

- Team Organization & Task Allocation Proposal
- Success of Organization & Task Allocation Proposals
- Robot Design
- Robot Code
- Conclusion & Changes for Next Project

Team Organization

- Proposal
  - Rotating Leader
    - Rotating Tasks
    - Group Responsible for Every Phase
    - General Exposure to the Project
    - Even workload

Task Allocation

- Proposal
  - Tasks divided according to who wanted to do what
    - All members had much software background and little hardware background
    - Everyone needs to help
    - Members assigned tasks to oversee
    - Hard to estimate how much work was involved

Task Allocation

- Proposal
  - Most important tasks to complete
    - Hardware design – Team
    - Hardware implementation – Stephen/Team
    - Software design – Team
    - Software implementation – Camilo/Team
    - Robot testing – Team
    - Paperwork organization – Jeremy
    - Group presentation – Camilo

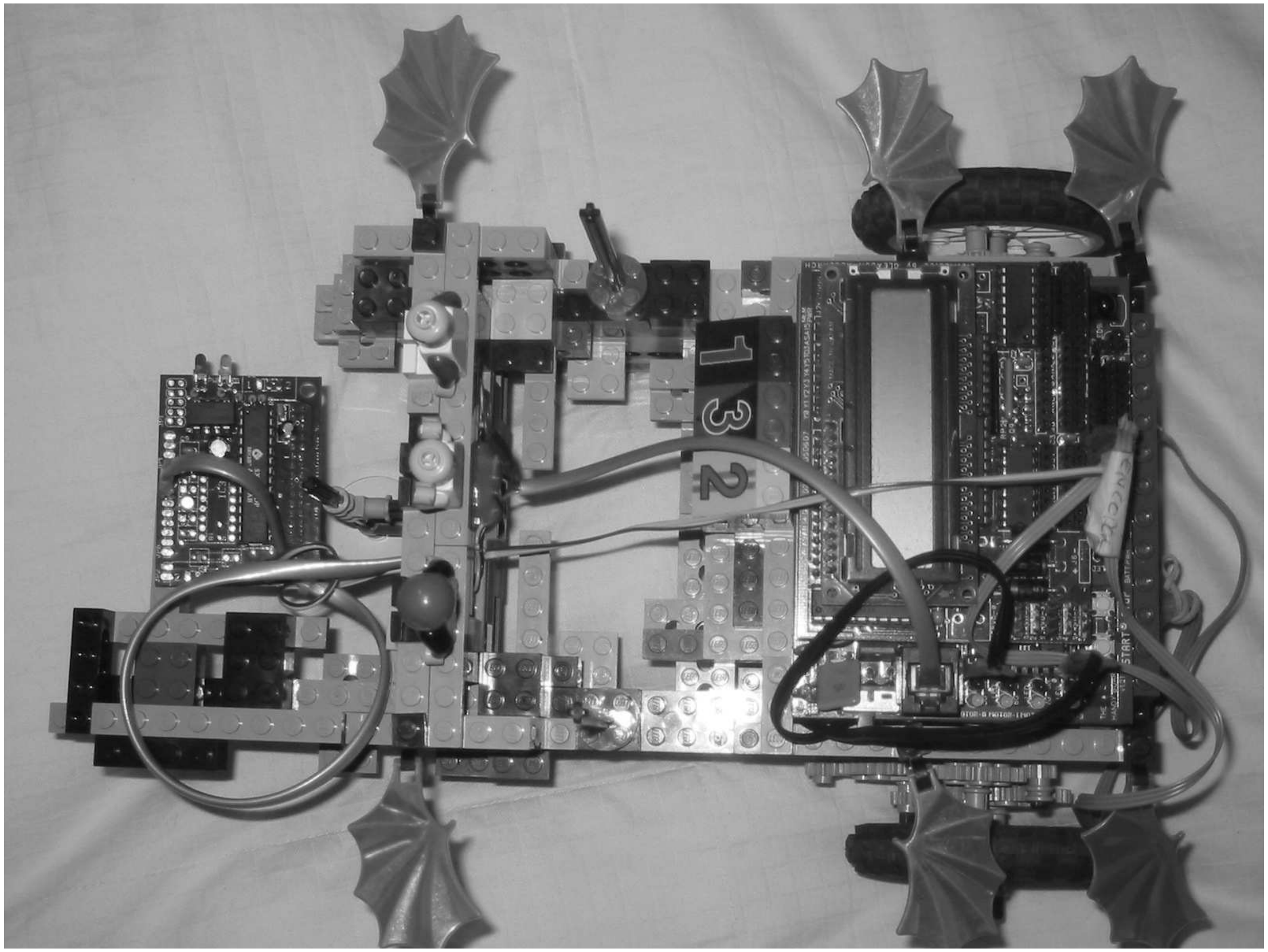
Team Organization

- Success of Proposals
  - Rotating Leader
    - Worked well most of the time
      - Created needed ideas as robot progressed
      - Created even workload
    - Hard to meet with everyone at the same time
      - Too crowded around robot or computer

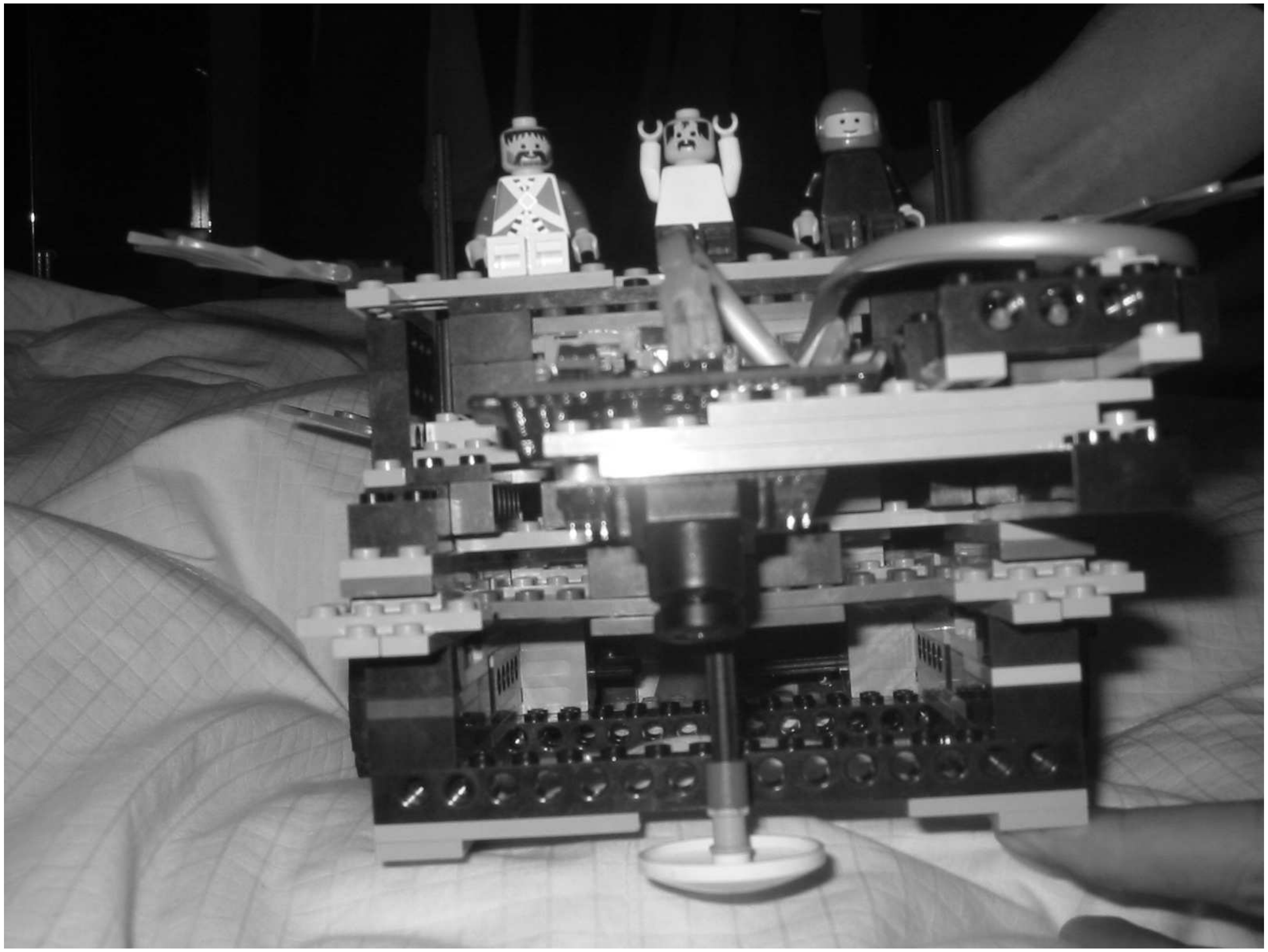
Robot Design

- Basic two-motor design
  - Base design taken from Martin’s book
    - 2 3” wheels in back
    - Ski- like device in front
      - Low friction
    - Encoders
      - Increased accuracy in driving straight and turning
    - Camera in front
      - Less shade

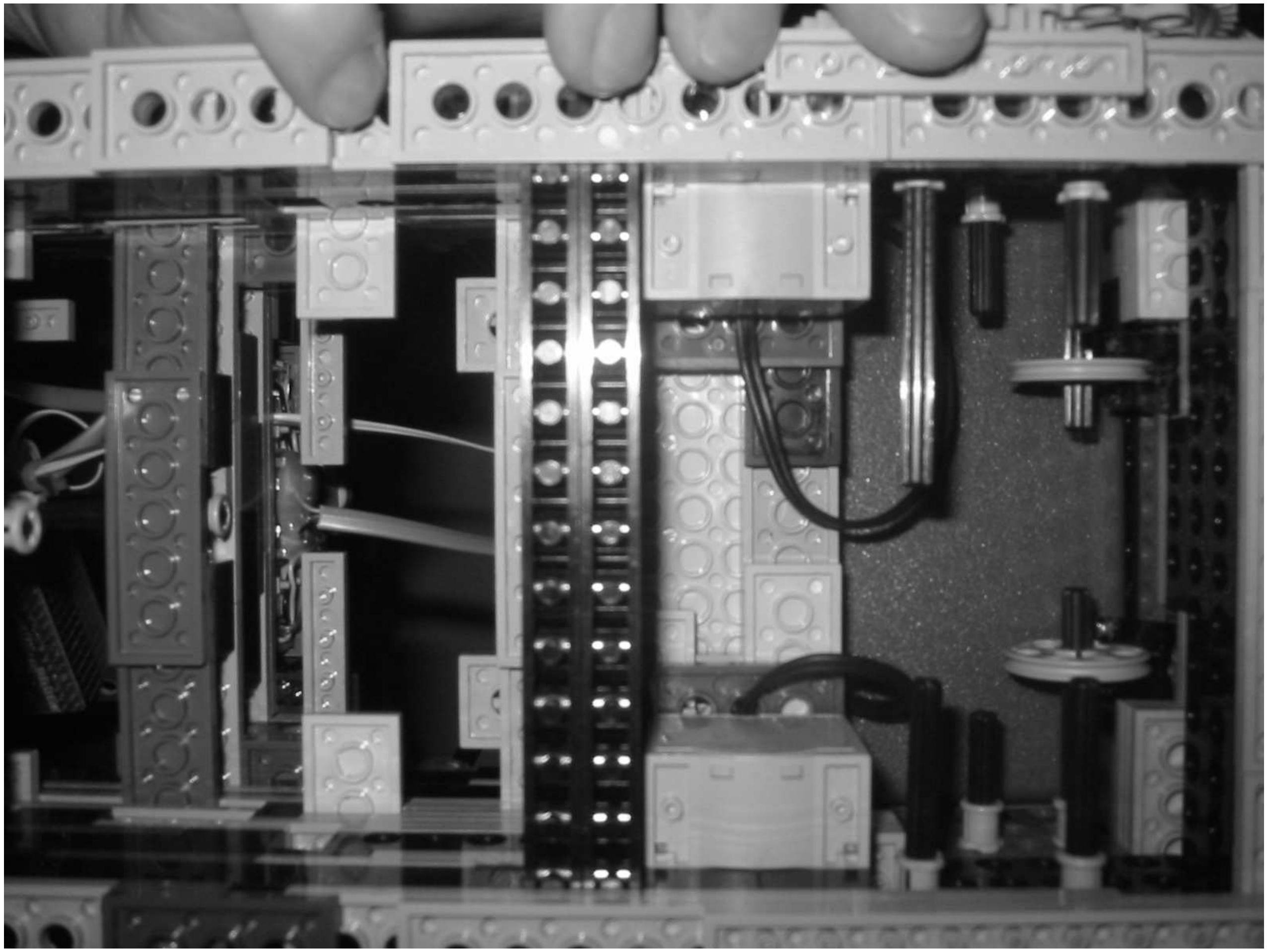
Robot Design – Back wheels



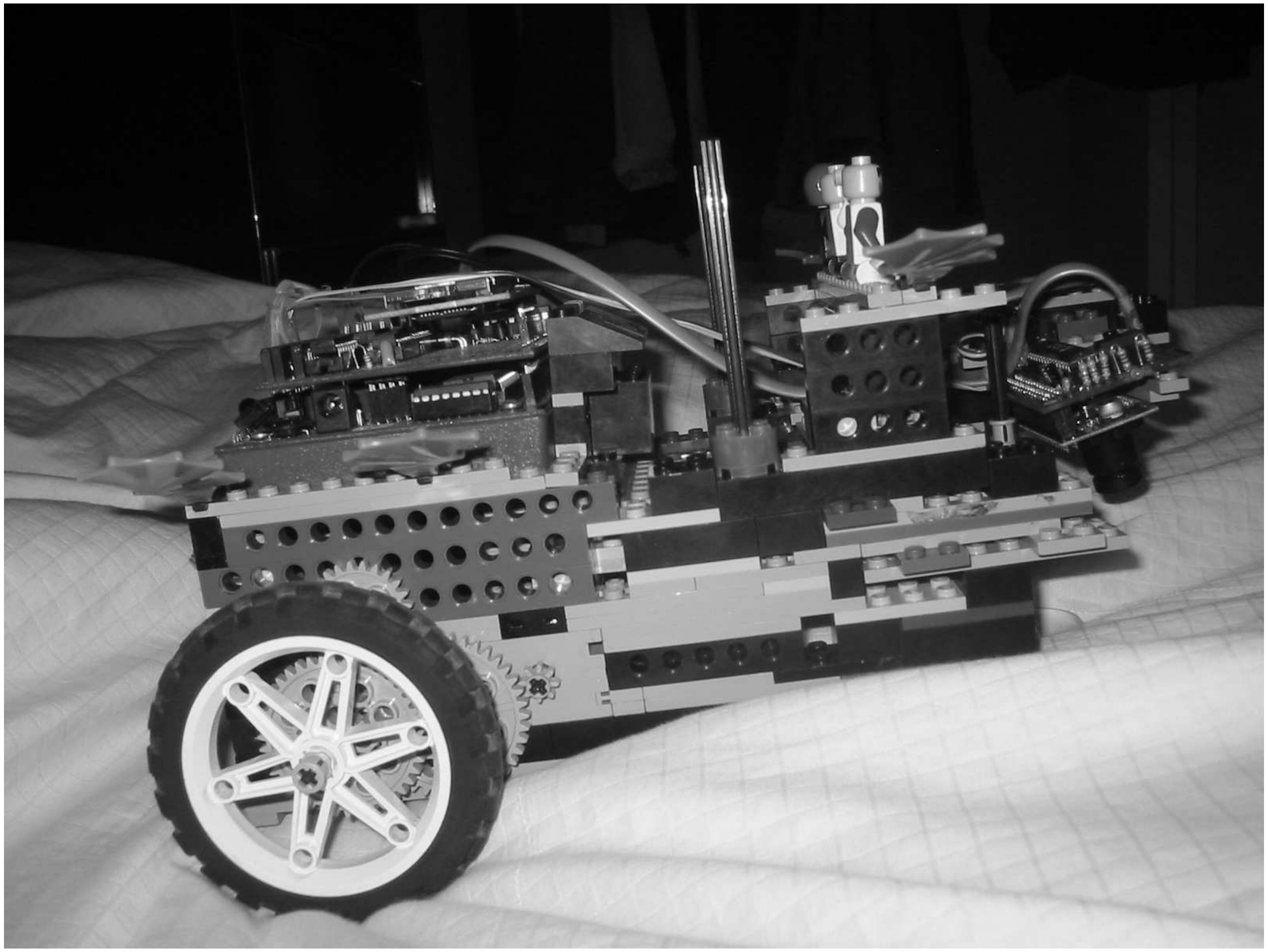
Robot Design – Ski-like front



Robot Design – Encoders



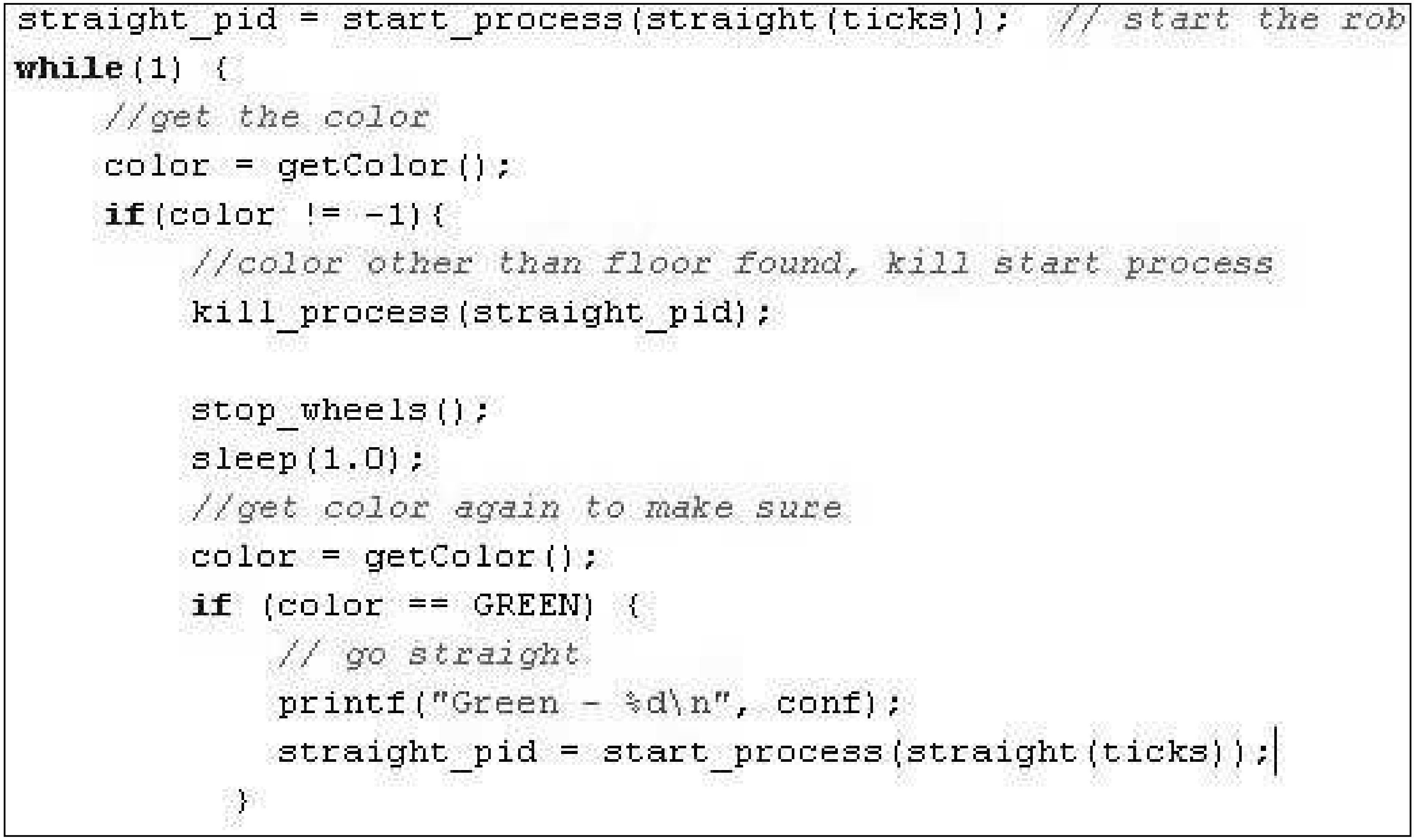
Robot Design – Camera



Robot Code

- Multi-process approach
- Checking for color
  - Perform appropriate action upon seeing color
- Driving straight
  - Use encoders
- Turning
  - Use encoders
- General Flow

Robot Code – Multi-process Approach & Checking for color



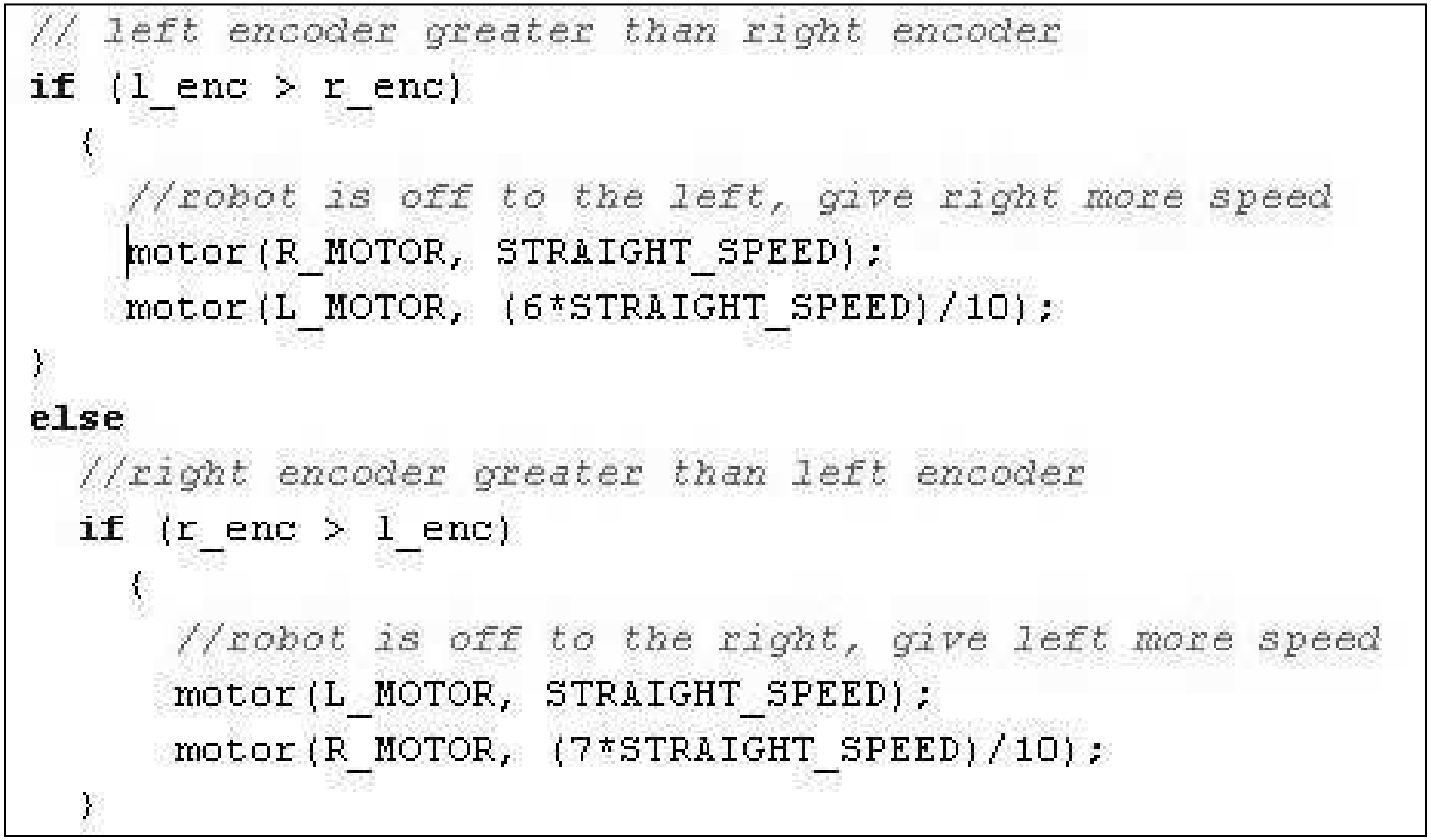
```
straight_pid = start_process(straight(ticks));  // start the rob
while(1) {
    //get the color
    color = getColor();
    if(color != -1){
        //color other than floor found, kill start process
        kill_process(straight_pid);

        stop_wheels();
        sleep(1.0);
        //get color again to make sure
        color = getColor();
        if (color == GREEN) {
            // go straight
            printf("Green - %d\n", conf);
            straight_pid = start_process(straight(ticks));
          }
```

Robot Code – Driving Straight

- Encoders
  - Placed on smallest gear to achieve best accuracy
  - Compared as robot drove
    - Changes made when needed

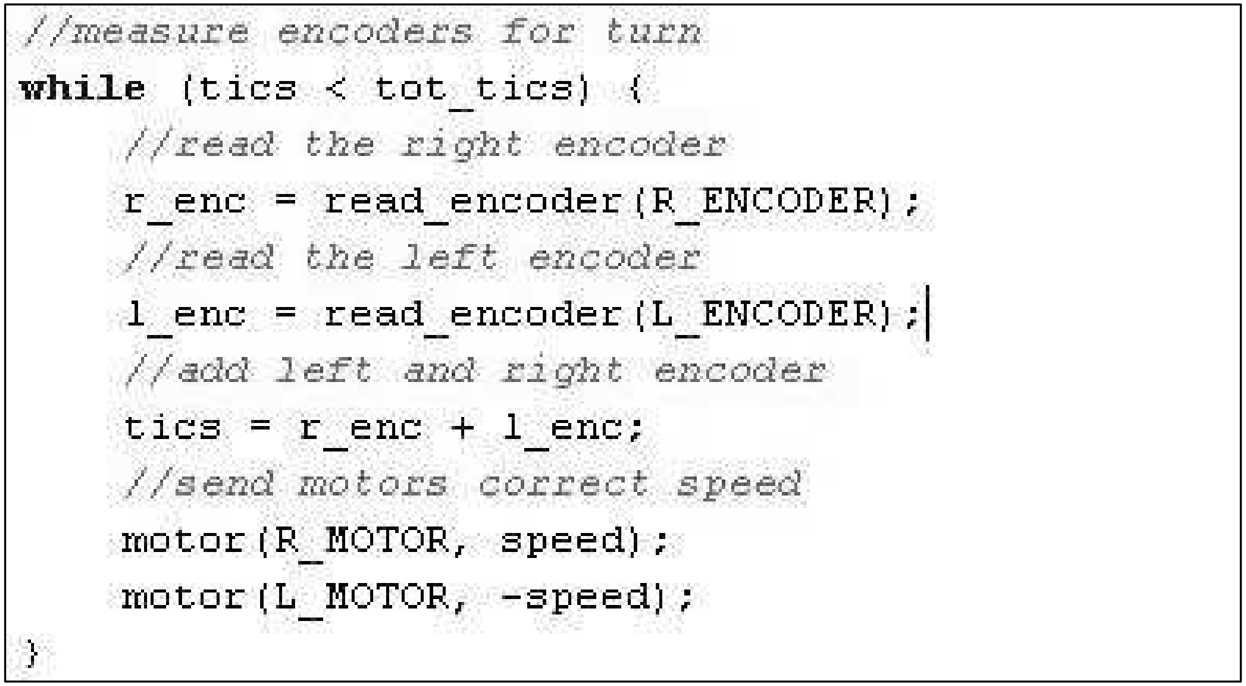
Robot Code – Driving Straight



```
// left encoder greater than right encoder
if (l_enc > r_enc)
  {
    //robot is off to the left, give right more speed
    motor(R_MOTOR, STRAIGHT_SPEED);
    motor(L_MOTOR, (6*STRAIGHT_SPEED)/10);
}
else
  //right encoder greater than left encoder
  if (r_enc > l_enc)
    {
      //robot is off to the right, give left more speed
      motor(L_MOTOR, STRAIGHT_SPEED);
      motor(R_MOTOR, (7*STRAIGHT_SPEED)/10);
  }
```

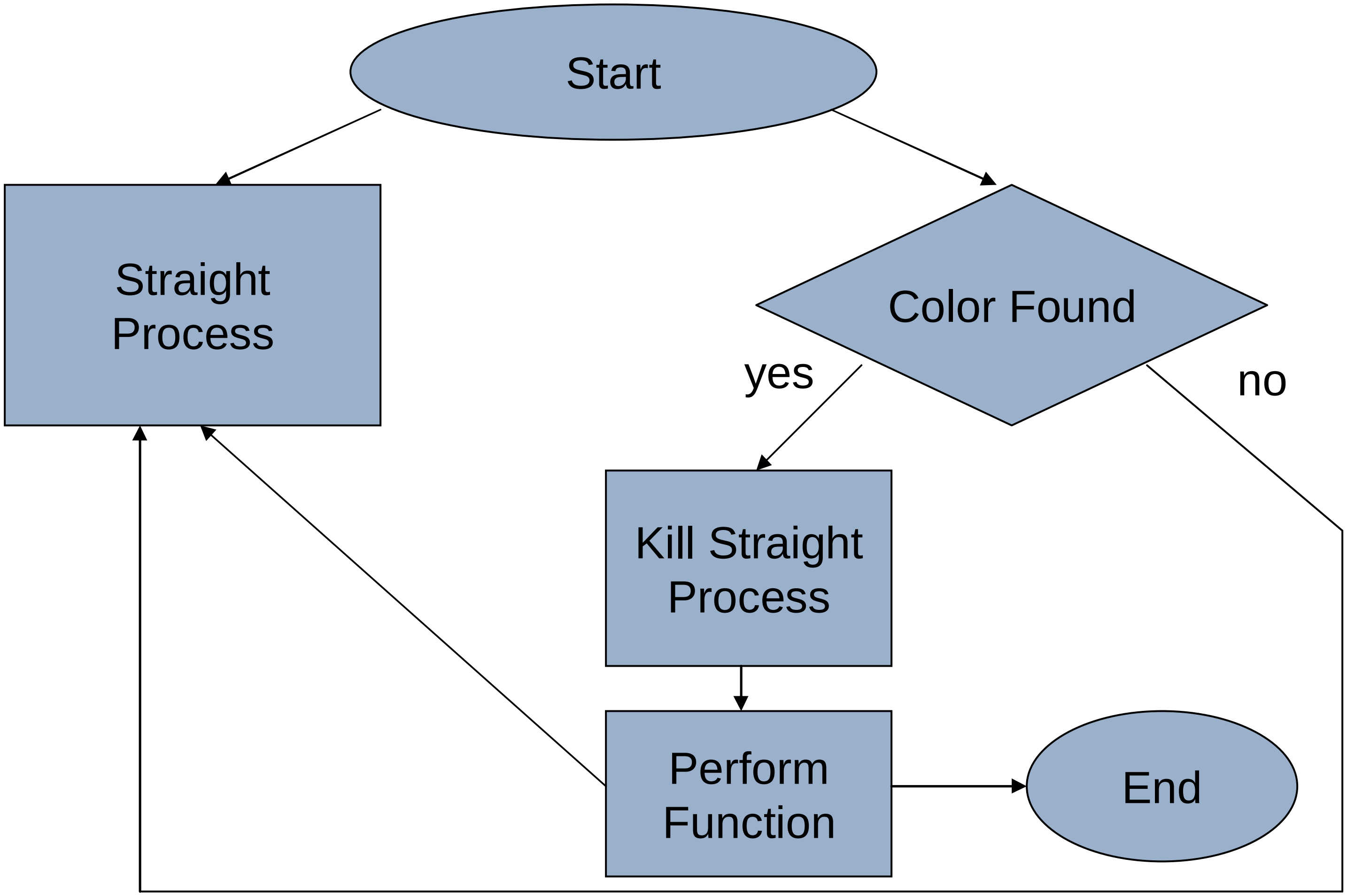
Robot Code – Turning

- Measure number of encoder ticks needed
- Compare number as robot turns
    - Send one wheel forward and one backward



```
//measure encoders for turn
while (tics < tot_tics) {
    //read the right encoder
    r_enc = read_encoder(R_ENCODER);
    //read the left encoder
    l_enc = read_encoder(L_ENCODER);
    //add left and right encoder
    tics = r_enc + l_enc;
    //send motors correct speed
    motor(R_MOTOR, speed);
    motor(L_MOTOR, -speed);
}
```

Robot Code – General Flowchart



Conclusions & Changes for Next Project

- Conclusions
  - Rotating leader works well
  - Workload was divided evenly
  - Project 1 was a success
- Changes for next project
  - Don’t wait until the last 20 hours
  - Work on a faster design
  - More individual work