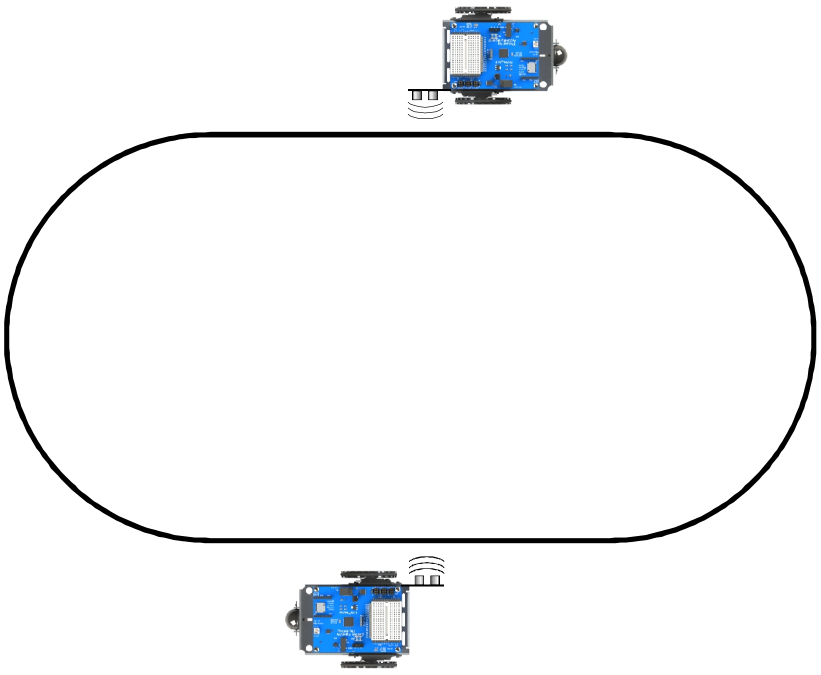
5 February 2014

Individual Pursuit Race!

Introduction

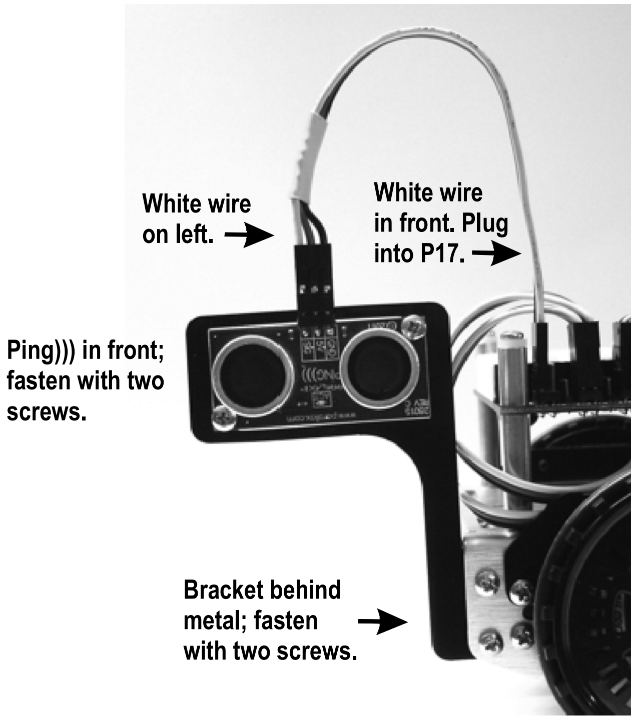
A bicycle "individual pursuit" race is held in a velodrome. Two cyclists start simultaneously at opposite sides of the track. The objective is to catch up to the other cyclist before he catches up with you. Here's a photo of an individual pursuit race in progress:

We are going to have pursuit races between pairs of robots! This will be a tournament with one team being the winner. The "track" is an oval barrier that reflects ultrasonic pings. You will have your Ping))) units facing the port side (left side) of your robots to detect the distance to this barrier. You will program your robots to keep a constant distance from the barrier as you circle it counter-clockwise. Here's a diagram:



Construction

Assemble the Ping))) unit and bracket as shown. Let Phil check everything before powering up.


White wire on left.
White wire in front. Plug into P17.
Ping))) in front; fasten with two screws.
Bracket behind metal; fasten with two screws.

Programming

You will pretty much be on your own with the programming details. But here's a skeleton program to get you started, and it's okay to ask for help.

```
#include "simpletools.h"      // Include simpletools header
#include "ping.h"             // Include ping header
#include "abdrive.h"          // Include abdrive header.

int distance;                 // Declare distance variable.

int ping_mm(int pin);

int main() {                  // Main function
  drive_setRampStep(12);      // Max step 12 ticks/s every 20 ms
  while(1) {                  // Repeat indefinitely
    distance = ping_mm(17);   // Get mm distance from Ping)))
    Your stuff goes here.     // This code will determine the winner!
    pause(?);                 // How many milliseconds to pause?
  }
}

int ping_mm(int pin) {        // Return Ping))) distance in mm
  return ping(pin) * 10 / 58;
}
```

Use the **drive_ramp(left_speed, right_speed)** function to set the left and right wheel speeds. These will vary, depending upon how far you are from the wall.

Race Rules

Each pair of racers will start on opposite sides of the course, as determined by the clips that hold the two wall segments together. Teams may place their bots as close to, or as far from, the wall as they wish. Each team will power up its bot and hold its reset button down until the starter says, “Go!” The race will continue until one of the following occurs:

1. A robot commits a disqualifying infraction before both bots have made one circuit around the course. In this case, the race will be restarted after both teams have a chance to reprogram their bots.
2. One robot catches up to the other, either touching its opponent or passing it. In this case the faster robot wins the race and proceeds to the next heat or is declared the overall winner in the last heat.
3. A robot commits a disqualifying infraction after both bots have made at least one circuit around the course. In this case, the infracting robot loses, and the winner proceeds to the next heat or is declared the overall winner in the last heat.

These are the disqualifying infractions:

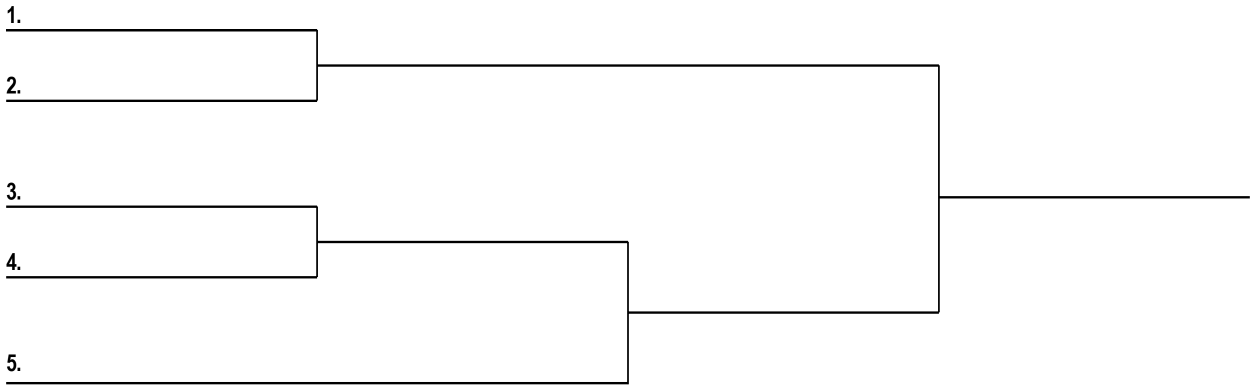
1. A robot touches the barrier.
2. A robot wanders away from the course.
3. A robot spins two revolutions or more without forward progress.

Initial team pairings will be drawn at random. Each team will get as many chances as it needs to test its program on the course and to modify it before committing to its next race.

Strategy

Obviously, the closer to the wall you can stay, the less distance you have to travel; but this also increases your chances of hitting the wall and being disqualified. Also, the faster you go, the better chance you have of catching up to your opponent. This also carries the risk of losing “sight” of the wall entirely and wandering off the course.

Race Heats



Give your robots names! I will make nameplates for them for the race next week.