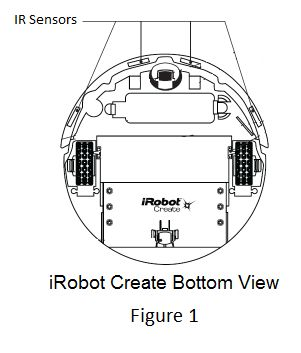
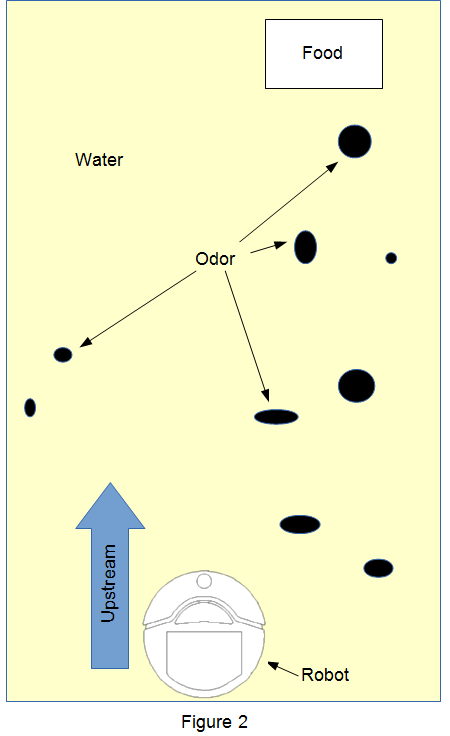
**Lobster Casting Lab**

This lab simulates a lobster's odor gated rheotaxis behavior on the iRobot Create. This behavior includes moving toward the mean odor flow when food odor is detected, casting randomly to the left and right when food odor is lost, and stopping when food is found.

**Translating lobster casting to a skid steering ground robot**

A Create moving on the ground is not a perfect analogy for a lobster standing in a turbulent stream of water searching for food, so we must consider the approximations we make:

1. Sensors. A lobster uses its sense of smell to detect approaching food and determine its direction. For the Creates, we will use the array of four downward facing IR sensors at the front of each robot (Figure 1). These sensors are actually cliff-sensors, meant to detect when the robot is about to move off an edge (such as stairs). However, they are sensitive enough that they can be used to read the infrared reflectivity of the ground. In this way, the robot can distinguish different surfaces/materials by their IR reading. Since they are spaced along the front of the robot, they can also determine direction.
2. “Food” and “Odor.” Since we are using IR sensors to distinguish different surfaces/materials, we only need three distinct materials to represent the “water” the robot is in, the “odor” of food, and the “food” itself. For this lab, the water is represented by brown butcher paper. The odor is marked by black marks on the butcher paper. And the food is marked by a piece of aluminum foil. These have distinct enough IR reflectivity to be distinguished by the Create's IR sensors. Figure 2 shows an example of this setup.
3. Flowing water. A lobster searching for food stands relatively still facing upstream, allowing water to flow past it and thus detect food odor and move to intercept the food. For the Creates, the best approximation is to constantly move “upstream” on the butcher paper, and move left and right when appropriate.

**Robot Lobster Behavior**

The lobster\_casting program implements 3 behaviors with 3 perceptual schemas and 1 motor schema:

Behaviors

* + *find\_food (hungry, T\_max)*
    - Data: *T\_since\_last\_detected*
    - Methods: *Behavior::move\_upstream, Behavior::cast, PSchema::detect\_food, PSchema::detect\_odor*
  + *move\_upstream ()*
    - Data: *direction*
    - Methods: *PSchema::detect\_mean\_flow, MSchema::locomote*
  + *cast (T\_since\_last\_detected)*
    - Data: *direction*
    - Methods: *PSchema::random, MSchema::locomote*

Perceptual Schemas

* + *detect\_food*
    - Returns: *yes/no*
  + *detect\_odor*
    - Returns: *yes/no*
  + *detect\_mean\_flow*
    - Returns: *direction (LEFT, RIGHT or FORWARD)*
  + *random*
    - Returns: *direction (LEFT or RIGHT)*

Motor Schemas

* + *locomote (direction)*

When the program begins, the robot will spin in place to calibrate its IR sensors, and will then pause to allow the user to reposition the robot if necessary. This calibration is necessary because IR readings are not consistent across different Creates or different IR sensors on the same Create. **The calibration needs to be done on a blank area of butcher paper.**

After calibration, *Behavior::find\_food* is activated. In a 10-millisecond timed loop, it does the following: check if food is detected with *PSchema:: detect\_food* (if so, the robot stops and the program terminates), check if odor is detected with *PSchema::detect\_odor* (if so, *T\_since\_last\_detected* is set to 0, otherwise it is incremented), and call either *Behavior::move\_upstream* or *Behavior::cast* depending on the value of *T\_since\_last\_detected*. If *T\_since\_last\_detected* exceeds *T\_max*, the robot stops searching for food and the program terminates.

*Behavior::move\_upstream* uses *PSchema:: detect\_mean\_flow* to update the value of *direction*. This value is then passed as a parameter to *MSchema::locomote*. *Behavior::cast* works similarly, but uses *PSchema:: random* to update *direction*. *Behavior::cast* also uses an internal variable called *T\_casting\_timer* (which is derived from *T\_since\_last\_detected*) to handle moving left and right while maintaining an “upstream” movement as well. *T\_casting\_timer* is compared to two parameters (*begin\_casting\_threshold* and *end\_casting\_threshold*) to determine whether to turn sideways, return forward, or move straight during each cycle of the main loop in *Behavior::find\_food*.

**Running the Program**

*What you will need*:

* **1 iRobot Create** (numbered B0 - B9), charged
  + should have a white serial cable and a blue or silver serial-to-USB adapter attached
* **1 netbook** (numbered B0 - B9), charged

*Instructions*

1. **Turn on netbook and login**
   * username: netbook, password: netbook
2. **Plug in Create’s USB cable to netbook, and turn on Create**
   * you should hear a beep and the Create’s power LED should turn green
3. **Open 2 terminal windows**
   * click icon on left panel with “>\_” symbol, right click on it to open a second one
4. **Change directory to playerprojects/lobster\_casting in BOTH TERMINALS**
   * “cd playerprojects/lobster\_casting”
   * run “ls” to ensure the files “lobster\_casting” and “create.cfg” are both in the directory
5. **Place Create robot on blank area of butcher paper**
6. **In one of the terminals, run player with the Create configuration file**
   * “player create.cfg”
   * The power LED may turn off
7. **In other terminal, run dead\_reckoning** program
   * “./lobster\_casting” to run with default parameters
   * The robot will spin in place to calibrate its IR sensors. After this is done, the program will pause to let you place the robot where you want it to start. Press the Enter key to continue.

*Lobster Casting program usage*:

./lobster\_casting [-h] [-m <T-max>] [-v <velocity>] [-b <begin-cast-threshold>]

[-e <end-cast-threshold>] [-o <odor-locomote-threshold>] [-f <food-stop-threshold>]

'-h' or '--help' will display this program usage information

'T-max' is in seconds and defines the maximum about time the robot will cast before it gives up and stops. Valid range is [5, 100]. Default value tis 15.

'velocity' is in centimeters per second. Valid range is [1, 50]. Default value is 10.

'begin-casting-threshold' is in seconds and defines the amount of time after losing the food odor that the robot will wait before casting left or right. Valid range is [1, 10]. Default value is 3.

'end-casting-threshold' is in seconds and defines the amount of time the robot will cast left or right before returning to upstream direction. Valid range is [1, 10]. Default value is 4.

'odor-locomote-threshold' is in milliseconds and defines the continuous amount of time that odor must be detected before the robot will locomote toward it. Valid range is [100, 2000]. Default value is 500.

'food-stop-threshold' is in milliseconds and defines the continuous amount of time that food must be detected before the robot will stop (to 'eat' it). Valid range is [100, 2000]. Default value is 500.

*Understanding the program in action*

**IR Calibration**

* Before starting the lobster\_casting program, you will need to place the Create robot on a blank area of butcher paper. It will use this area to calibrate its IR sensors by spinning in place in one direction then the other while constantly recoding IR measurements from each of its four IR sensors. After this is done, the program computes a linear normalization factor for each of the sensors to bring their runtime values to the correct, known IR reflectivity value for the butcher paper. IR values for other materials scale accordingly. The reason the robot must spin in place to compute these normalization factors is that illumination affects the IR reflectivity of a material, so a full 360 degree measurement is needed to account for this variability.

**LED Lights**

* There are three LEDs on top of each Create. These are used to indicate internal state to the user. The different lighting patterns and their corresponding states are:
  + No lights: in odor stream, continuing upstream
  + 1 green LED: odor detected to the left or right
  + 2 green LEDs: food detected (robot should stop)
  + 1 red LED: no odor or food detected

*Troubleshooting*:

* If dead\_reckoning program does not run:
  + Close player and try again:
    - In dead\_reckoning terminal, press Ctrl-C to stop program
    - In player terminal, press Ctrl-C to stop program
    - Run “ps -a” to ensure no player instances are still running. If there are, run “kill -9 $(pgrep player)”
    - start again from step 5 in instructions above
  + Make sure the Create is connected correctly
    - With player running, check that the LEDs on the Create’s serial cable are constantly blinking. If not:
    - Close player as above
    - Turn off Create and check serial cable connection
    - Turn Create back on and start again from step 5 in instructions above
  + Restart the netbook, or try with a different one
    - If problem persists, the problem is likely a low battery on the Create
* If Create stops responding (e.g. runs out of control):
  + Quickly stop player program
    - Ctrl-C in player terminal
  + Failing that, pick up the Create and turn it off