Robot Challenge Journal

Challenge No.: 3

Team Name: Group 4 Members' Names: Tim Loverin, Chris Barret, Mohammed Juma, Brandon Soto

Date	Seq. #	Name(s)	Hypothesis/Behavior	Description/Results
5/12/15	1	Chris, Tim, Brandon		We began discussing and diagramming the behaviors for Challenge 3. **(FIGURE 1)**
	2	Chris, Tim, Brandon		We discussed the idea of an arbiter (controller) that would control task activation. These tasks will control motor functions rather than the arbiter.
	3	Chris, Tim, Brandon		After getting an initial diagram sketch, we looked into getting the bricks to communicate with one another through Bluetooth. Unfortunately, it doesn't seem this will work.
	4	Chris, Tim, Brandon		To prepare for Thursday's meeting, we need to find out how to pass values from one brick to the next. We've decided on using USB to have the bricks communicate. Also, we need to think of some ideas for robot design.
5/14/15	1	all	Mohammed found a NXT program that allows two bricks to communicate. By making a few modifications to it, our EV3 bricks may be able to communicate with each other.	**result:** Failure. The provided functions are not supported for EV3. After running into a dead end with bluetooth communication, we've decided to complete this challenge using LeJOS. LeJOS should allow for easier Bluetooth communication and will give us extra credit.
	2	all		We started designing the layout for the robot. We've placed one brick on top of the other to allow for easy access.

	3	all		Based upon our diagram (**FIGURE 1**), we'll code the skeleton for the controller.
	4	all		We showed our diagram to Fowler. He was concerned with our controller and it not seeming to prioritize tasks. He also commented that it was too much of a black box for his liking.
5/19/15	1	Tim		Redesigned robot structure, added a 3rd light sensor for navigation, removed back two motors since steering with 4 proved to be too shaky and hard on the servo motors. Leveled bottom two light sensor now have full range of color spectrum on sensors. Added a push bar for touch sensors.
5/21/15	1	Tim, Brandon, Chris		After having complications with LeJos, we've decided to work with NXT kits. The plan is to work on both types of kits. Chris and Mohammed will work on EV3 with LeJos. Brandon and Tim will work on NXT with RobotC. Note: The only NXT parts we're using are the sensors, the parts that connect the bricks together, and the parts connecting the sonar.
	2	Tim, Brandon	Online sources said that EV3 motors are compatible with NXT bricks. We've made some tests to see if this is true.	results: Success. The sensors and motors seem to be working properly.
	3	Tim, Brandon	Now that we're assured that the NXT sensors are working, we need to focus on getting the two bricks to communicate. We've found some guides online that we'll be using. (LINK)	result: Success. We were able to send values from one brick to another. Now we should try sending sensor values. We also need to decide on which sensors will be attached to our slave and master bricks.
	4	Mohammed, Chris	Testing socket interaction by sending a message from one ev3 brick to a second ev3 brick via wifi	Successfully was able to pass a simple message from ev3 brick to the other using sockets through Wifi. The message passed was the date according to the first brick. The solution to the wifi problem was solved by creating our own wifi hotspot using a cell phone because the schools wifi will not allow us to create access points.

	5	Mohammed, Chris	Trying to get motors and sensors working through LeJOS.	
5/26/15	1	All		We've decided to abandon LeJos and stick with RobotC and the NXT bricks. With that taken care of, we need to decide on the code that we'll reuse.
	2	all	Fowler would like us to show the EV3 bricks communicating with one another for extra credit. We will show one brick connected to a touch sensor that can stop the motor connected to another brick.	result: Success.
	3	Brandon, Tim	We're reusing Tim's wander code and will now be testing the code with the NXT bricks to ensure that everything works. We must do this before going forward.	result: Success. The robot wandered as expected.
			Need to test the sonar sensor to ensure that it works properly.	result: Success. The sensor worked as it should.
	4	All	Need to test the two light sensors. Specifically, we need to test ambient and reflected light to see the returned values.	result: Success. The sensors worked as expected.
	5	All		We're creating a skeleton for the sender and receiver code. We've started with the wander behavior and will work our way up through the hierarchy.
	6	All		Started working on a gradient following algorithm. (**FIGURE 2**)
	7	All	Still need to calibrate the sonar sensor to determine its minimum and maximum distance for objects.	Successfully determined the minimum and maximum distances for other objects.
	8	All		To do: ● state diagrams ● feeding (modified wander) ● gradient following

				● fear response
	9	Chris	creating and testing the fear behavior for the robot	Still getting the hang of the slight syntactic difference between the EV3 and NXT but with only minor issues with a backwards < the fear behavior should work in our set of behaviors.
	10	Tim	added linkedlist.h file	result: Success
	11	Tim	filtered sonar values with a queue to remove extreme values.	result: Success
	12	Tim	Created file for secondary brick which sends a single integer value to the main brick containing the information in both touch sensors via bluetooth.	result: Success
	13	Tim	tested object detection	result: performed as desired.
	14	Tim	tested sending brick to brick touch sensor data messaging.	result: successfully sent data.
	15	Tim	wrote and tested death code	Robot shuts down when it dies.
5/28/2015	1	Chris	some code of overly complicated for no reason in the fear behavior	simplified code and tested successfully. Uploaded code to google docs
	2	all	We wrote code on our own. Specifically, we wrote code for updating the robot’s state, fits ear response,avoiding objects, following gradients, and feeding. We now need to integrate all of this code into one file.	Started integration. Ran into to a few difficulties with calibrating the light threshold, but we eventually got it by having the robot wait a bit.
	3	all		**TO DO (in no particular order):** ● test gradient follow / feeding ● ***fully*** integrate code ● fix bumper design - it can be unresponsive at times ● create finite state diagrams

				● finalize behavior hierarchy (see here) ● finalize implementation of update_state function - make sure we’re decrementing energy at the proper time/rate
5/30/2015	1	Chris	Fixing up fear to fit the way in which the integration code is set up	did not test
5/31/2015	1	Chris	implementing the same turnaround function from the touch sensor functions.	did not test
6/1/2015	1	Tim	Fixed issue of state getting stuck	Success
	2	Tim	Integrated calibrate wander and object avoidance by adding return statements	Success **Calibrate code leapfrogs last part of calibrate code
	3	Tim	Integrated object Detection have 4 combined states working	Success
6/2/2015	1	all		Chris made changes to the calibrate and fear functions, so we added the code he had to the main file.
	2	all	After adding Chris’s changes, we need to test the fear function to ensure it behaves properly.	result: Partial success. The fear response worked the first time, but we ran into an issue related to the touch bumpers not detecting a bump. Need to replaced battery and try again. 2nd result: Seemingly successful. The robot responded to the light at the proper times and habituated to it.
	3	all		We also need to edit the implementation of our feeding function. At the moment, the robot does not turn back onto the patch while it is feeding.
	4	all	The feeding implementation has been edited. Now we need to test the robot with all behaviors enabled.	result: Failure. The robot does not even start calibration because of one of the changes we made. Need to find the change that is causing this.
	5	all	The robot may be stuck in a limbo state because of the way we organized our main function. Changed the location of our	result: Failure. Robot still stuck in limbo state.

			update_hunger call and calibrate call.	
	6	all	Adding a delay to the calibrate function may allow it to execute.	result: Failure. The robot calibrated itself but got stuck in the wander state. The robot did not respond to light flashes or bumping the bumpers. 2nd result: Failure. The robot calibrated and changed states successfully for a while, but got stuck in the object detection (sonar) state. Need to investigate sonar thread. Also, after looking at the assignment again, we've found we found that our timing decrementing and incrementing the energy level was wrong. Need to edit this.
	7	all	The sonar thread is low priority right now. The gradient following needs to be focused on and tested.	result: Failure. Robot does not turn around when it experiences the end of a patch as it feeds. The robot is having problems staying on the patch. This may be due to the robot not being calibrated properly this time through.
	8	all		**TO DO:** ● finish implementing and testing gradient follow / feeding ● create finite state diagrams ● fix object detection (sonar) function ● fix bumper design - it can be unresponsive at times
	9	Chris	found improper syntax in the feed and follow functions	did not test for reasons that I cannot get it to run properly. Will test when we meet back up later today.
	10	Chris	testing for #9	the motors do stop when appropriate. but they need to turn around once to do it so would only start feeding then,
	11	Chris	found that the turn was bouncing between the two different branches of the if statement continuously and missing return`	Turn works now. Only turns left though and never enters the feeding state nor do we wander and feed. Had one time where I transitioned to state 1 from state 4 and I have no idea why that happened. The turn radius needs to be looked at as the corners were too big.
6/3/2015	1	Tim	fixed bumper design and stability	Bumper is now responsive and more stable.
	2	Tim, Brandon	Need to test what we have to double check what we need to work on.	The robot gets stuck in the gradient following state. Tim and I believe that adding a timer may help with this issue. For instance, suppose that the robot doesn't use one of the gradients to enter the patch, a timer would help the robot realize that it's already in the patch.

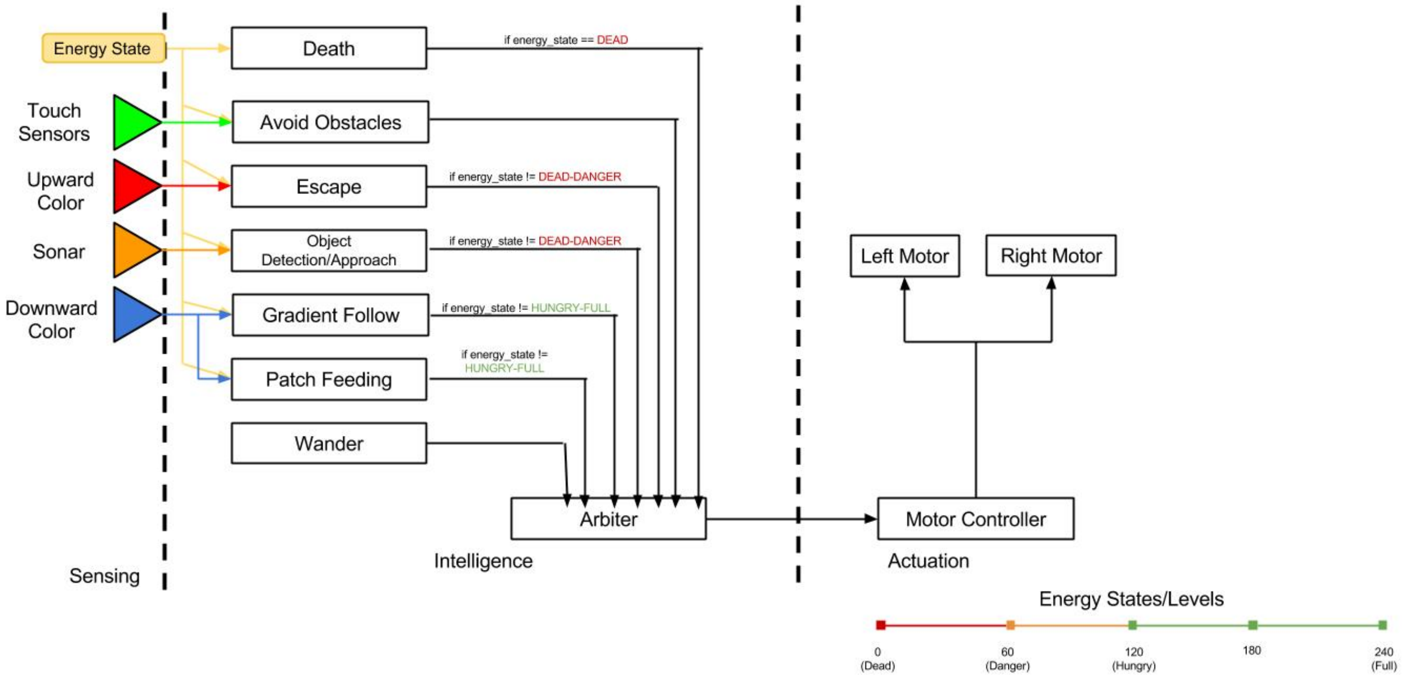
				Another issue: the robot seems to get stuck quite a bit.
	3	Tim, Brandon	added a spacer to the wheels distancing them from the body to reduce part friction. Also cleaned out hair from servo motors.	The robot performed a little better, but it still got stuck at times. Instead of having the robot perform a turn with one motor set to 0, we might want to set that motor to the inverse of the other.
	4	Tim, Brandon	Removed unused back light sensor to reduce drag. Also changed the other motor's speed while turning to -15.	result: robot turned too tightly, drag was noticeably less. The robot responds to black way too late. This is causing the robot to get stuck in a loop while it is following a gradient. The seems to respond to black quickly, but, because of the light sensor's position, it may never find the patch again.
	5	Tim, Brandon	moved feeding above following so we can call feed in follow	
	6	Tim	added alive() in touch sensor brick to keep it from shutting down mid program	
	7	Tim, Brandon, Chris	Adjusting the motor speeds for the gradient following turns. These have been changed to 40 and -25.	result: Robot is still getting stuck in loop. Need to move sensor further forward to be closer to the wheels.
	8	Tim, Brandon, Chris	Changed the location of the light sensor to be closer to the wheels.	result: Success. Testing on tail of patch
	9	all	Adding an addendum to the if statement that leads from following to feeding	still not entering feeding
	10	all	Due to bad design choices early in the design process we have been struggling to get everything working as requested. Mostly because we did the opposite of what Prof. Fowler ascribed at the beginning of this project.	

	11	Tim	Reset seen gradient after following state.	result success.
	12	Tim	added wander part to feeding and adjusted the timer to properly increment	result success
	13	Tim	added conditional to break from feeding when energy level is full.	result feeding works as needed.
	14	Tim	added sound to death	success
	15	Tim	added flag to prevent issue of randomly going into wrong state.	success
	16	Tim	added sound when breaking from feeding	success
	17	all	TODO:	fix object detection (sonar) function. finish implementing and testing gradient follow.
6/3/2015	1	Mohammed ,Tim	commented out gradient check to allow for feeding	Robot now goes from finding the gradient to feeding properly.
6/4/15	1	all		Showed our robot to Fowler. He suggested following a script. Also, our robot got stuck in the sonar function and freeze the robot.
	2	all	We've used a our own linkedlist in the sonar thread. This may be causing our problems. We've replaced the linkedlist with a for loop and are using a rolling average. This should prevent the robot from freezing.	result: Success. The robot no longer freezes from the sonar function and properly transitions into another state.
	3	all	Here's our script for presenting: 1) wander 2) Fear	result: robot does not respond to touch bumpers while escaping. Mohammed suggesting editing the way our sender brick communicates with our receiver brick. At the moment, the receiver brick is constantly polling the sender

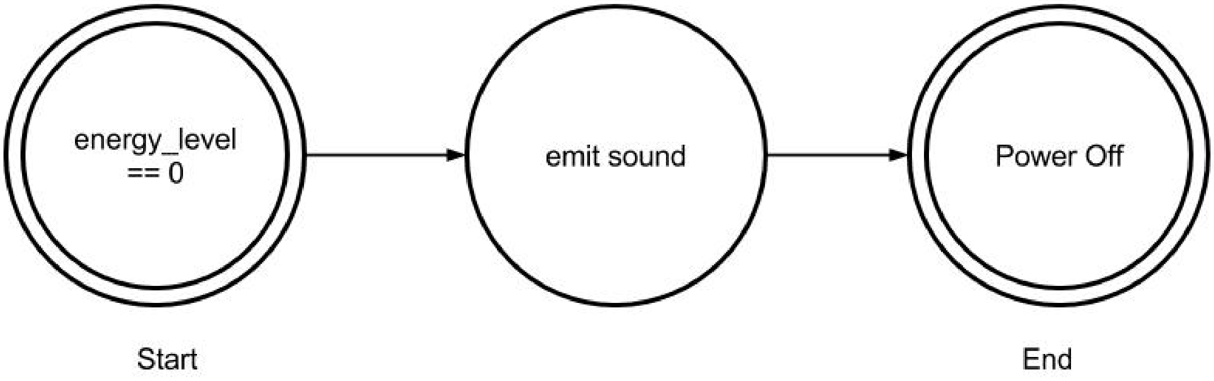
			3) Bumper during fear 4) sonar during fear 5) bumper alone 6) fear habituation 7) bumpers during sonar detection 8) sonar detection 9) following 10) entering feeding (sound) 11) fear during feeding 12) sonar during feeding 13) exit feeding (sound) 14) death	brick. Instead, the sender should notify the receiver when it is touched. It got caught in the gradient following state and would not transition at all to feeding.
	4	all	Testing all of our script behaviors.	result: Robot still doesn't respond the bumpers while escaping. Robot still doesn't transition into the feeding state. This may be a problem with our gradient following function. Because of this, we haven't been able to test 8-10 from our script.
	5	all	Our "fed" variable is set to 0 when we need it to be set to 1. We've set it to 1 in the gradient following function. This should allow the robot to transition into feeding.	result: Success. We were able to successfully test 8-10 from our script. While feeding and not being in DEAD-DANGER state, the robot responded to fear and the sonar.

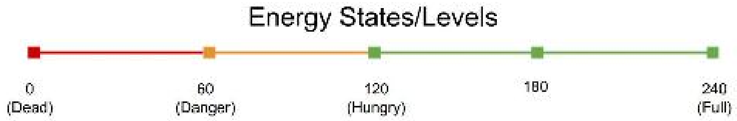
Composition Architecture

Overall Behavior Diagram

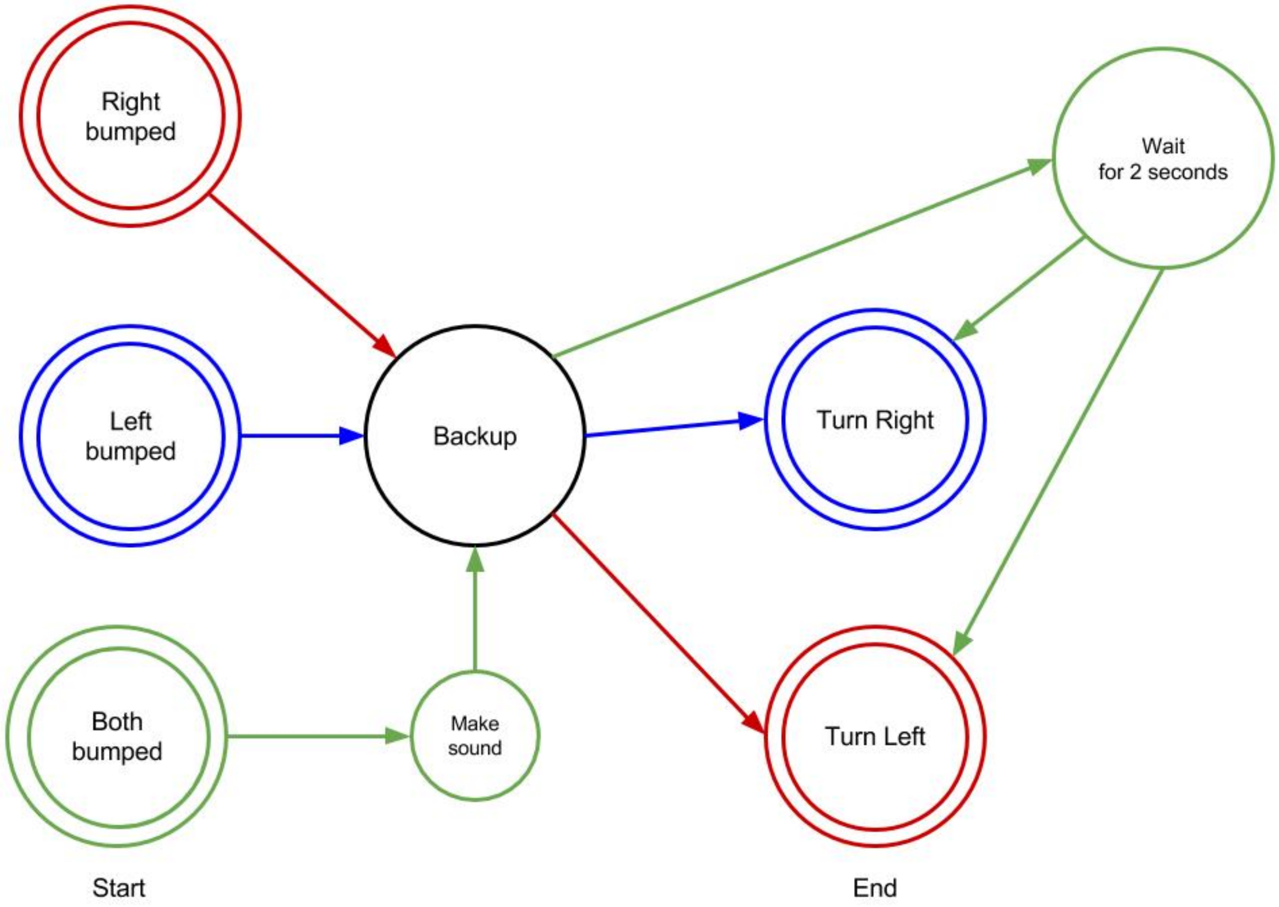


Death Diagram

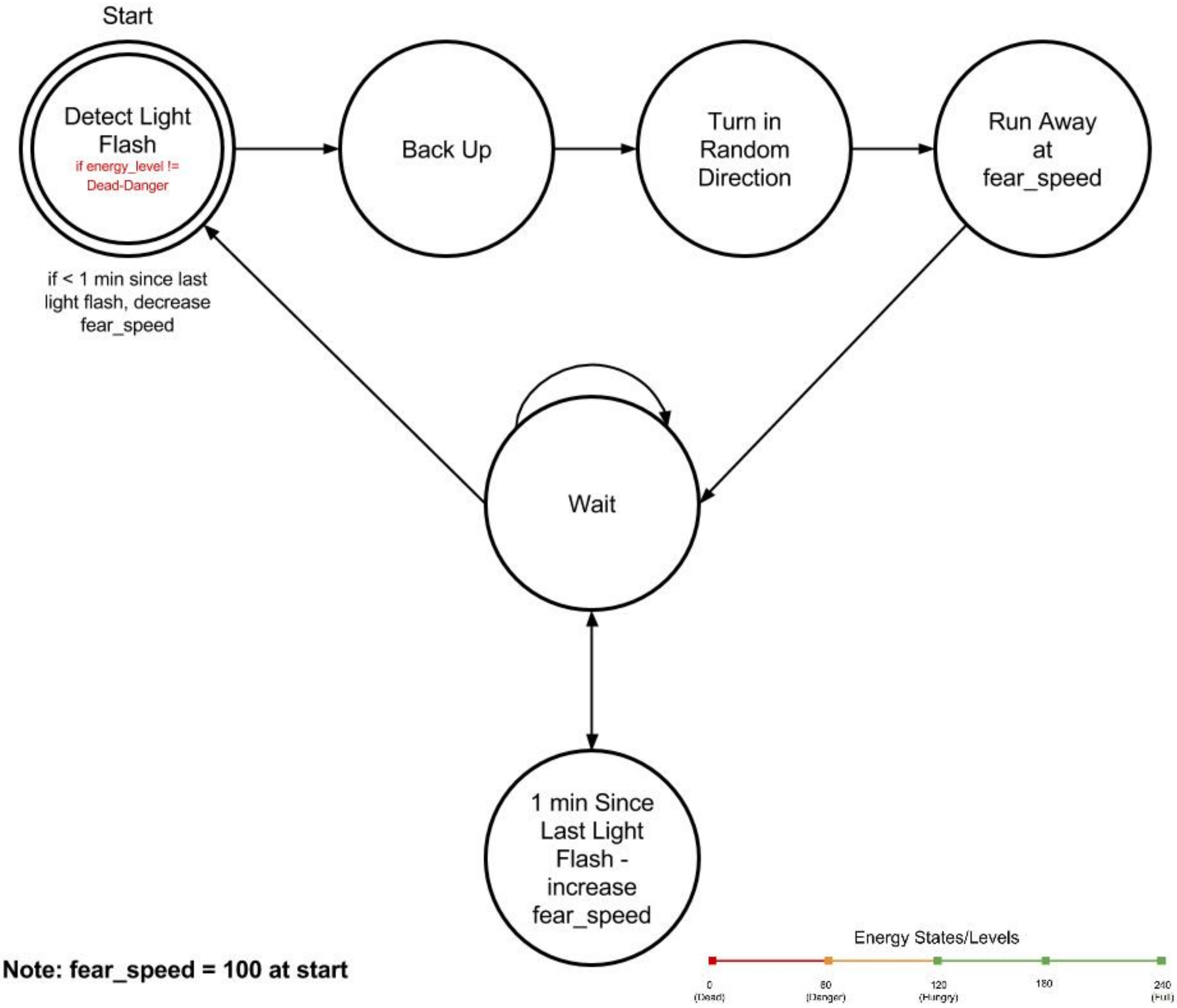




Avoid Obstacles (Touch Sensors) Diagram

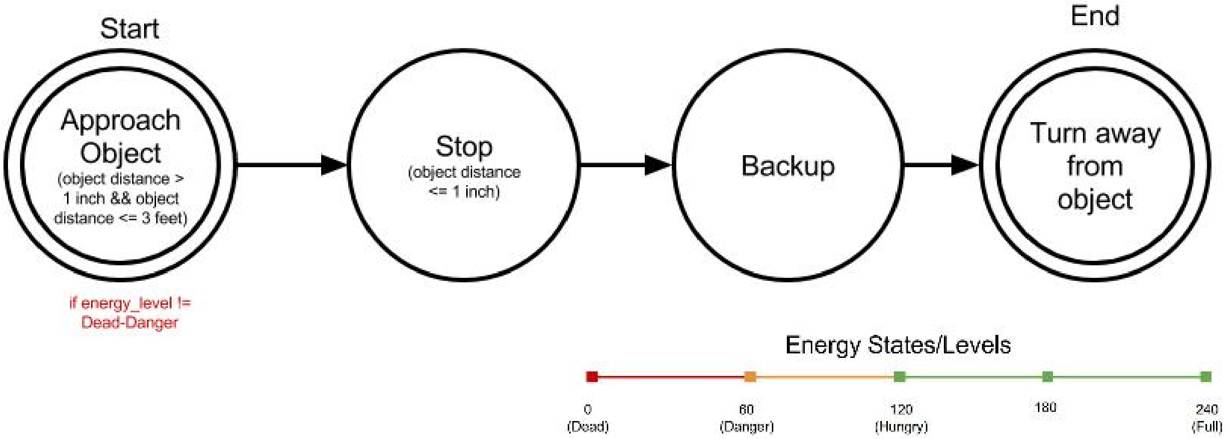


Escape Diagram

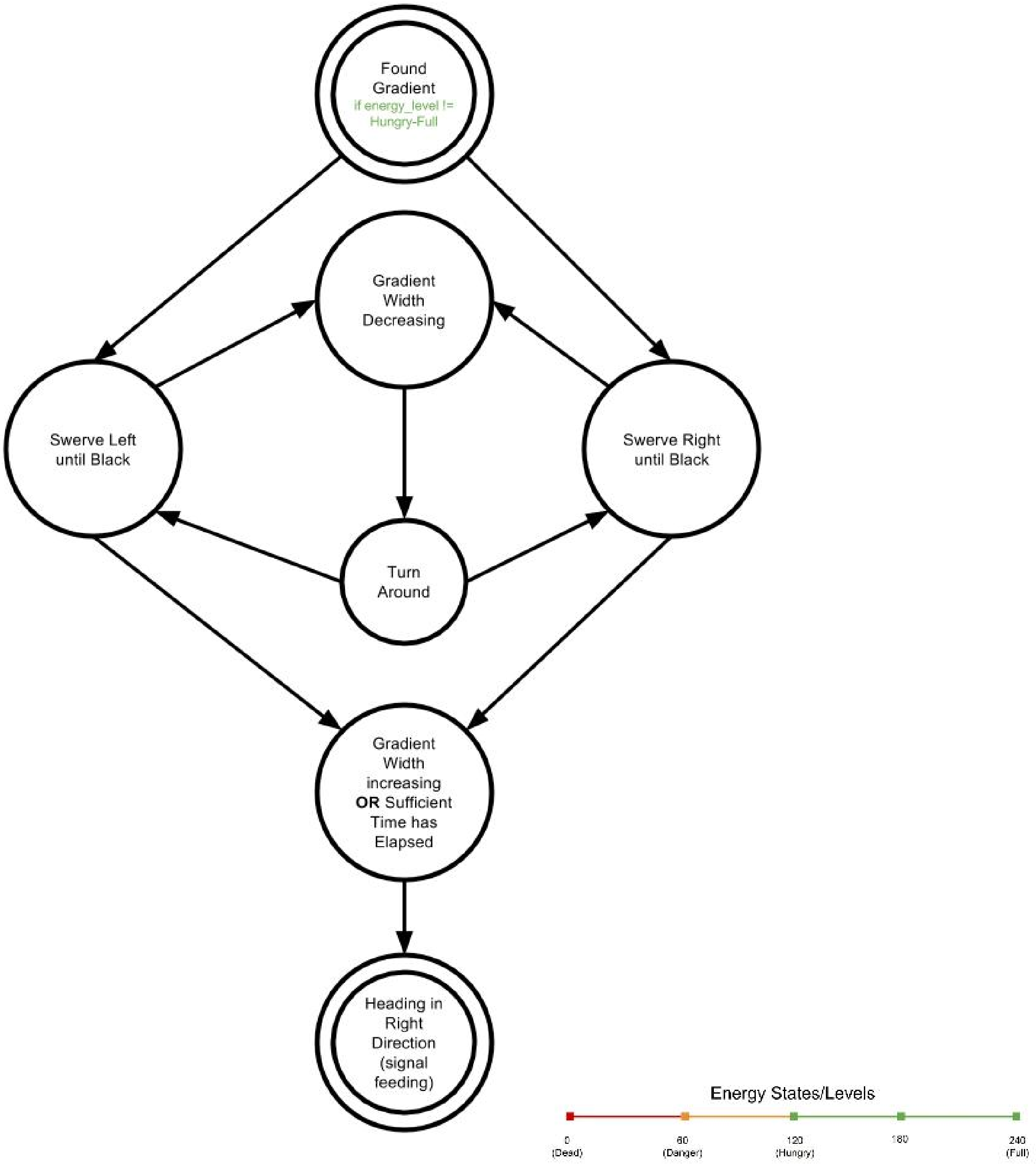


Note: fear_speed = 100 at start

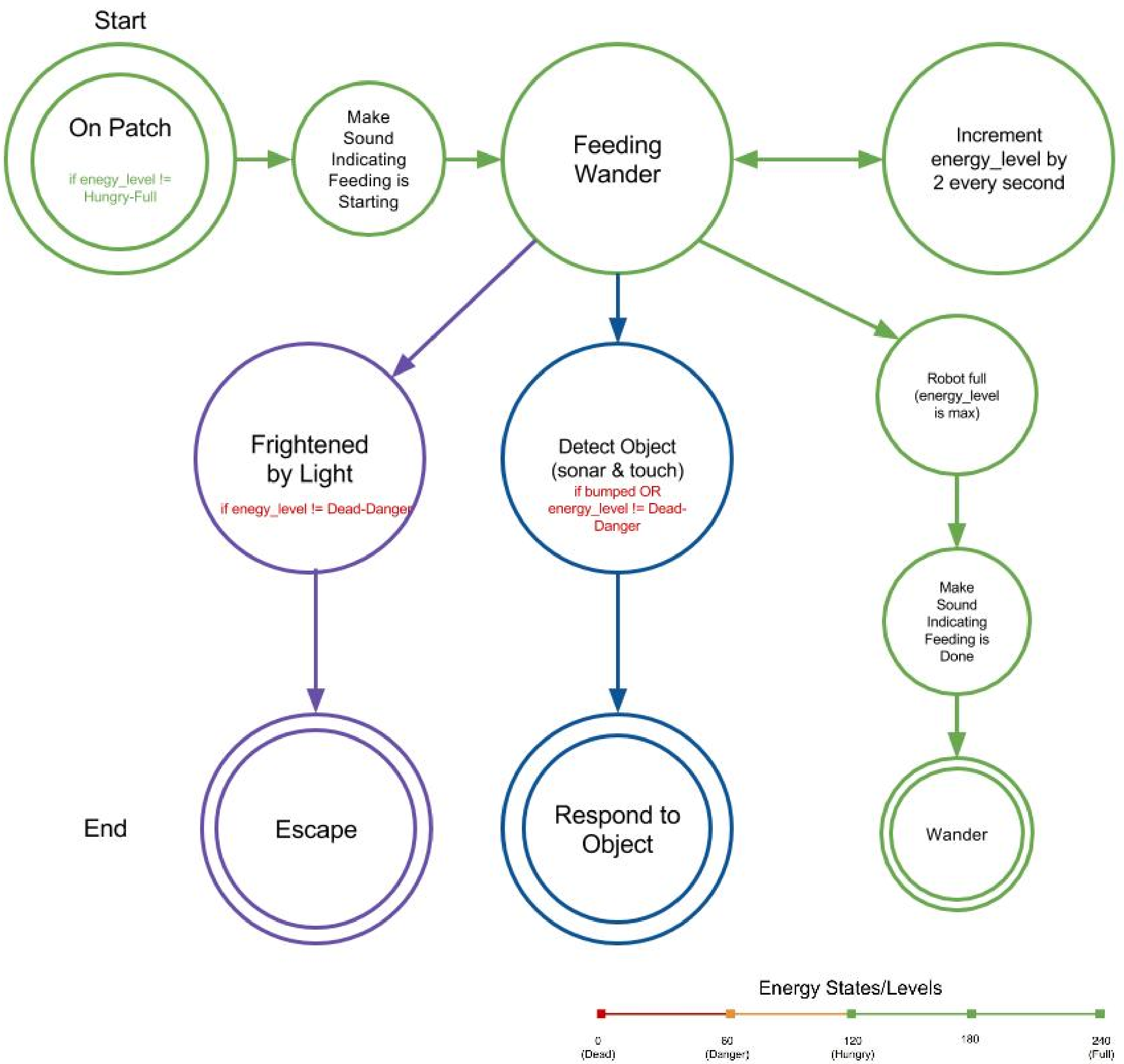
Object Detection & Approach (Sonar) Diagram



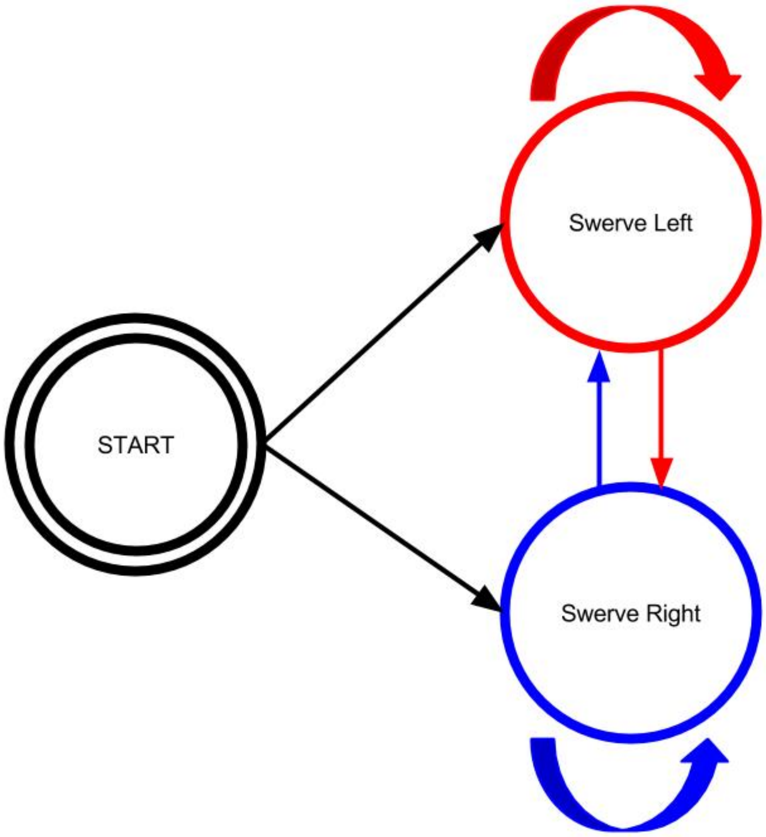
Gradient Following Diagram


Found Gradient
if energy_level != Hungry-Full
Gradient Width Decreasing
Swerve Left until Black
Swerve Right until Black
Turn Around
Gradient Width increasing **OR** Sufficient Time has Elapsed
Heading in Right Direction (signal feeding)
Energy States/Levels
0 (Dead)
60 (Danger)
120 (Hungry)
180
240 (Full)

Patch Feeding Diagram


Start
On Patch
if enegy_level != Hungry-Full
Make Sound Indicating Feeding is Starting
Feeding Wander
Increment energy_level by 2 every second
Frightened by Light
if enegy_level != Dead-Danger
Detect Object (sonar & touch)
if bumped OR energy_level != Dead-Danger
Robot full (energy_level is max)
Make Sound Indicating Feeding is Done
End
Escape
Respond to Object
Wander
Energy States/Levels
0 (Dead)
60 (Danger)
120 (Hungry)
180
240 (Full)

Wander Diagram



Description of Brick Task Distribution and Communication

For task distribution, we tried to keep as many tasks as we could on a single brick. We connected the touch sensors to one brick (the sender) and connected all other sensors and motors to the other brick (the receiver). The sender brick is *only* responsible for detecting and sending touch sensor values. The receiver brick is responsible for getting those touch sensor values, and to use these values and all data from its own sensors to mediate robot behavior through the arbiter. That is, all behavior/state logic is contained within the receiver brick.

Related to brick communication, we used two NXT bricks paired through Bluetooth. In our code, the touch values that the sender brick send are the numbers 0 through 3. 0 means that no bump has been detected; 1 means that the right touch sensor has been bumped; 2 means that the left touch sensor has been bumped; 3 means that both sensors have been bumped. The sender shoots these values off every 100 milliseconds. Our receiver brick then acknowledges these values every time it goes through the update_state function.