An Autonomous Pet Entertainment Device

Robot Name: F.R.E.D.

(Feline Roving Entertainment Device)

EEL 5666C Intelligent Machines Design Lab
Dr. Arryo and Dr. Schwartz

Teaching Assistants: Mike Pridgen and Thomas Vermeer

Brian Flatley

Table of Contents

OPENING

MAIN BODY

CLOSING

Abstract

FRED, or Feline Roving Entertainment Device, is an autonomous cat toy designed with the complexities of a cat's behaviors in mind. It simulates the variety of prey a cat would encounter in the wild. It uses rangefinders to avoid obstacles and pyroelectric motion sensors as a means of sensing nearby motion of a cat or other animal.

Two near the middle of FRED's body provide movement and his electronics are housed in two frames: A wooden balsa base layer and a thin plastic cover that mounts on top of the wooden one. He uses 9 V of double A batteries for power and a PVR board for control.

All Intelligent Machines Design Lab (IMDL) robots needed obstacle avoidance and an interfaced special sensor. The initial purpose behind using motion/heat sensors is while the CMU cam would have probably given more reliability, motion sensing grants more real world application. Rather than picking up a color, it picked up a heat source and thus be versatile. It is also intended to not be limited by its environment: It should be sturdy enough to hold up against a cat as well as any object it should meet in its environment.

FRED is a useful companion when a human is not able to play with the cat, and is designed to coexist with the cat in its environment, acting as stimuli in a relatively unstimulating existence of the common housecat.

Executive Summary

FRED, the Feline Roving Entertainment Device, is an autonomous robot designed to engage and play with a cat. He is driven by two motors allowing for spot turning, and a furniture glide completes FRED's three point balance system. FRED is equipped with an antenna with a ribbon affixed at the end as a means of enticing the cat, a plastic shield to protect his vital electronics and keep the cat from harmful or possibly toxic parts contained inside, and an LCD and power switch mounted to the top to aid troubleshooting and enable power to the circuit board respectively.

For obstacle avoidance, FRED uses three long range infrared proximity sensors mounted on the front. They return values which are proportional to distance from the sensor. When values are high enough FRED turns away from it. If the left and right IR are both high, FRED turns a set direction, ensuring he will not get trapped in a corner. The center IR works in conjunction with the motion sensors during location mode to determine when he is close enough to enter flee mode.

The special sensors FRED uses are 4 pyroelectric motion sensors. These sensors take a picture of the room and use this picture to determine if an object has moved. If motion is detected, the switch turns high. Using 4 of these a broad spectrum of behavior can arise from a set of on-off switches.

Fred has three behavior modes: Random, Locate, and Flee. In random mode, FRED wanders around and avoids obstacles. After a set amount of time has passed or a great deal of motion is picked up on the motion sensors, FRED enters locate mode.

In locate mode, FRED uses its pyroelectric sensors to search for motion. Based on which of the sensors return motion, FRED turns toward the motion and approaches the source. When it gets close enough as deemed by the central motion sensor it enters flee mode.

In flee mode, FRED runs around similarly to random mode, but in flee mode FRED turns away whenever moving heat is detected on the motion sensors.

Introduction

The design intent for FRED's creation was making a machine that could not only track a small heat source like a cat, but avoid it and other obstacles in its path. This robot, F.R.E.D. (Feline Roving Entertainment Device) has many features to be subsequently analyzed though this report.

Choice of Subject

Initially the dilemma was what common pet should FRED be designed to engage. Between dogs and cats, cats were chosen primarily because they pose the least threat to the hardware being utilized. Dogs' prey tends to be larger on average than cats, and therefore dogs can naturally deal more damage to expensive components. In addition, cats have a tendency to play with their food. They bat at it and pounce until their prey is subdued. Once the prey has stopped resisting, then the feline will eat. Since FRED will not give up unless its batteries die, it will never arrive at the phase where the cat would start trying to eat it.

Although keeping FRED and the animal safe was a top priority, it was not the only reason cats were chosen as the intended audience. It is a commonly held assumption that dogs are higher maintenance than cats. Dogs are in nature pack animals, and thus require much more social attention than felines, the solitary hunters. For a would-be owner, unbiased to either and looking for the least amount of effort for upkeep, would choose a cat as a pet. Cats don't need to be walked regularly and can be left to their own devices and still be perfectly content. However with no mandatory physical activity like dog walking, cats are prone to becoming overweight. It is estimated that 40% of cats in the United States are overweight [1]. With this insight, cats stand to gain the most from a toy that they can interact and play with that can in fact interact and play back.

Environmental Enrichment Devices

Cats are complex creatures, and so it cannot be assumed that any one method of entertainment will apply to every cat. As such, FRED contains several physical features and behaviors that will appeal to different cats, and are listed below.

FRED will be relatively small in size, roughly equal to or smaller than the size of an average housecat. Thus he will be less intimidating: being seen as either prey or as a rival cat it play with. His random motion and attempts to locate the cat will appeal to the cats stalking hunter instincts. In this sense, FRED will act like a mouse or rodent. Additionally, FRED will have a ribbon mounted on an antenna on his back which simulates flying prey such as an insect or bird.

To protect the cat as well as protect FRED from any harm the cat may do, FRED will be encased in a large semicircular bump shield that, when activated will stop the machine's motion. The purpose of this is so the robot cannot cause physical harm to the animal. The shield will have the added effect of keeping the cat away from the parts that could be hazardous if ingested such as the lead solder, wire, pointy pieces and electricity.

FRED will also be designed in such a way to have a long battery life. As taking batteries

in and out is a hassle and he isn't a simple 'turn on and accomplish task' robot, FRED will need time for the cat to locate it as well as it locate the cat. Cats, as mentioned earlier, are stalking hunters, and so a long operational window would be most ideal.

Integrated System

Movement and Obstacle Avoidance

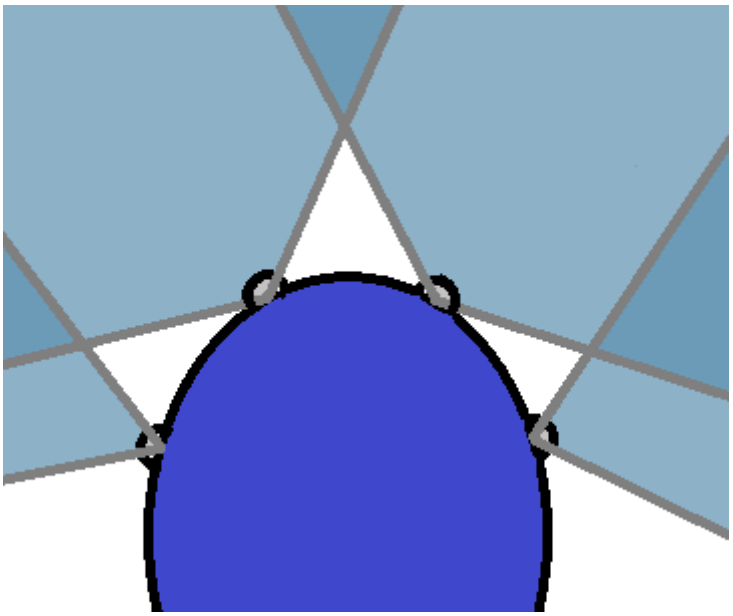
Once FRED starts up, he begins to drive around using IR for obstacle avoidance. In the event that one of the left and right front IR rangefinders in FRED's front are tripped from a distance, FRED will turn away from the object while continuing moving. If the object is very close, indicating FRED is not turning fast enough to avoid, FRED will stop, turn, and then continue moving. In the event that any of the three total IR sensors are not tripped by the obstacle, the bump sensor attached to FRED's front will cause FRED to turn away to the object that it bumped into (or bumped into it) and flee it.

Feline Detection History

While still in the design phase, the original idea was to have an infrared thermometer. A thermometer located elsewhere on the robot would supply a reference temperature from which the infrared thermometer would be able to distinguish between room temperature and a heat source. When a significant spike in this heat is demonstrated (~8+ degrees above reference temperature), FRED would approach the source and taunt it.

The next unique cat-sensing concept and largest influence on the final system design came from a freak accident with my sensors. When my IR rangefinders got lost in the mail with the obstacle avoidance deadline fast approaching, I toyed with the idea of using my motion sensors to detect the obstacles. The idea behind this was to a moving robot, even a stationary object like a wall would be considered moving, and hence would trip the motion sensor. With a motion sensor on the front corners of FRED, he could effectively steer away from walls and hence, avoid obstacles.

However, problems started when this theory was put to the test. For some reason FRED would turn until it faced a wall and then run straight into it. A colleague clarified this issue for by saying that the sensor actually detects moving *heat*. I then realized that FRED was seeking the walls because the cold walls provided the least likely interference for the motion detectors. This comment also led me to realize its potential as a heat source identifier. The result of this postulation was the arc of motion sensors across the frontal hemisphere of FRED. With the PVC shields attached, FRED has about a 200 degree frontal viewing arc from which any motion detected triggers turning and movement.



Fred's Motion Sensor Field of Vision

FRED's motion sensors seemed very sporadic at first but through further testing it seemed he could pick up on non-moving obstacles, recreating the large issue of having a moving object take motion readings.

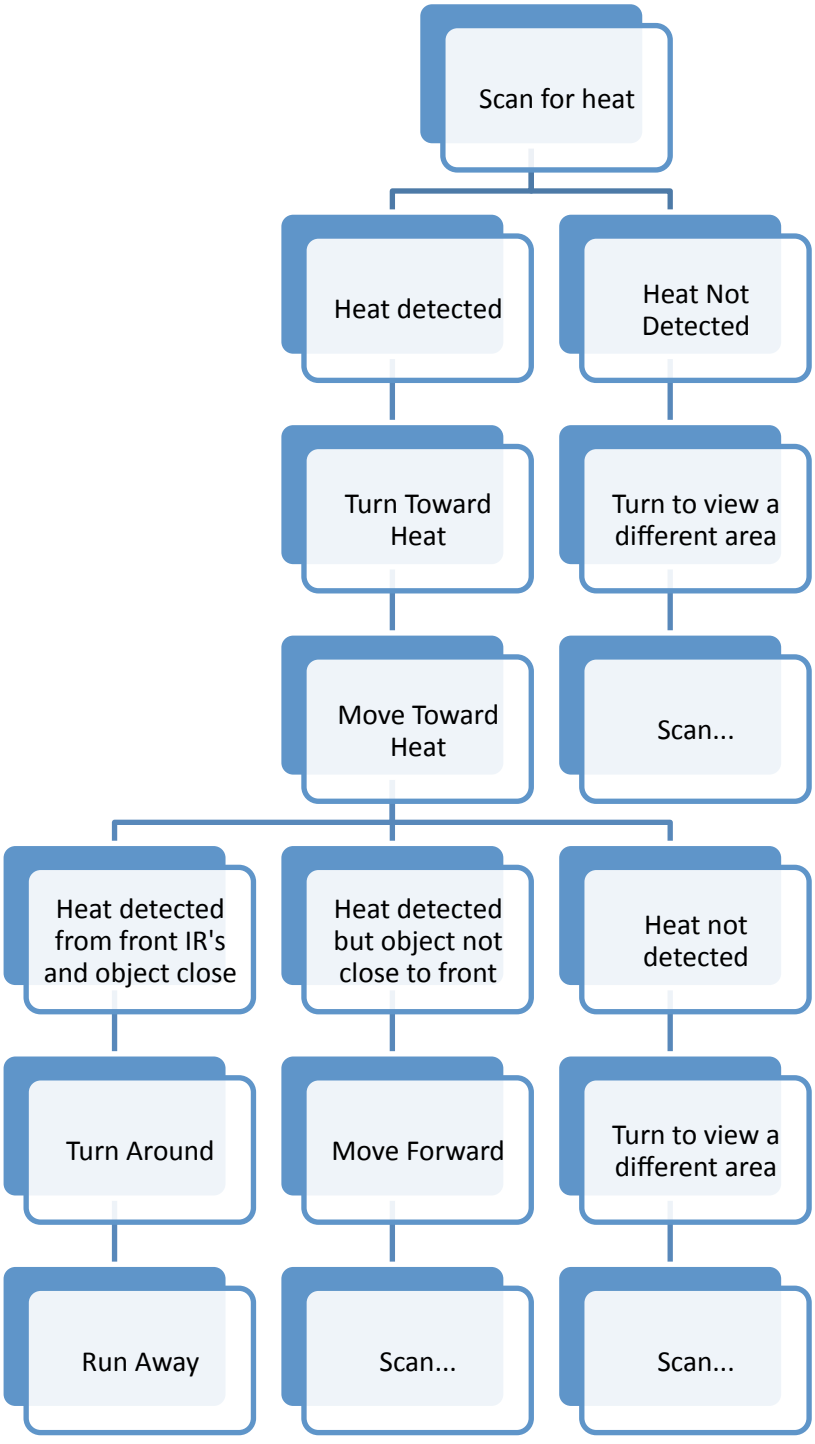
Feline Detection Conclusion

With the final discovery that FRED picked up on all motion, the idea behind his code is to stop any time a motion measurement is to be made. Once locate mode is entered, FRED waits 2 seconds, then based on which section of the motion viewing area the object is seen in FRED will do some combination of turning and forward motion with the end objective to be directly in front of and facing the source of movement. The reason for the 2 second delay is that the motion sensors activate for a duration of two seconds before turning off again. Once FRED is centered with the heat source, the center IR rangefinder is used to detect whether the cat is close enough for FRED to start fleeing from it.

During the second week of March I brought FRED in contact with his primary test subject, Ashley: An adult female housecat. At this point FRED was composed of a wooden frame, all the circuitry, the wheels, and a motion sensor mounted at each of the two front corners. I then outfitted him with a fairly simple program to look in the direction heat was seen and put him in a fairly cool dim room with Ashley nearby. The greatest trouble with the motion sensors is they yield many false positives and negatives and thus their readings need to be filtered. However despite having only 2 of the eventual 4 sensors mounted, it was still capable of keeping directed at Ashley give or take roughly 60 degrees to the left or right. Eventually these sensors will enable FRED to find a heat source to play with, approach that heat source, and then flee from it.

FRED's behavior will occur as follows. This behavior tree will occur periodically as FRED attempts to locate a source of movement:

FRED Locate Behavior Flow Chart



Motion Scanning Operation

For motion scanning FRED has four pyroelectric motion detectors mounted on its front. This means that when FRED turns, all visible heat sources will trip whichever motion detector whose line of sight it is in. Having four such sensors enables for a centering of the heat source in the robot's field of vision. A source that trips the middle 2 and not the outer 2 is directly in front of FRED and is ready to be approached. A source on the outside being tripped without the inner one means the object is nearing the edges of FRED's range of view, and thus FRED needs to turn to be facing that direction.

Mobile Platform

The mobile platform is composed of several layers. Its outermost layer is a thin plastic casing that is flexible enough to bend easily when pressure is applied to it. In this way, the bump sensor can be attached to the second layer without any visible hint of its existence. Mounted outside and to the plastic casing is the LCD screen on top, four pyroelectric motion sensors along the front, three infrared rangefinders around the front, and an antenna and power switch mounted on the top. The antenna will be an elastic plastic tube with a ribbon attached to the end to act as a cat stimulus. Underneath this protective plastic case is a rectangular balsa wood box on which the electronics, wheels, and battery pack are all mounted.

In FRED's early development, the frame was mostly held together with masking tape. This allowed for easy access to its core at any given time and combined with the interlocking balsa panels meant it was almost as sturdy as if it were glued. Later on the masking tape was replaced largely by hot glue. There is a hinged flap on the top of the plastic dome so the sensors can more easily be connected and disconnected when the need arises. The top is connected by a hinge for easy access to electronic components.

The purpose for the outer layer is not only to provide a safer interaction with the cat, but to protect the delicate inner components from the feline. The inner box comes from the inability to mount much of the hardware on such a flexible frame. It provides a solid foundation to build onto. A problem that became most apparent once the IR rangefinders were hot glued to the plastic casing is FRED became tedious to reprogram. The actual prongs on the PVR board the programmer needs to connect to are buried deep within FRED's armor: While it keeps him safe it means he had to be somewhat disassembled to access those prongs.

Actuation

Drive System

FRED is driven by a motor on either side and will rest on the ground through its two motor-driven wheels and a plastic ball castor located at the front of the robot. The two wheel drive will result in excellent turning capability, allowing FRED to turn in place. The castor is to provide balance and ensure his frame does not drag along the floor. Because of the two wheel design, FRED has360 degree range of movement and can turn in place.

The castor on the front is a plastic furniture glide, used to protect flooring from dragging furniture around while moving it, so it was an easy solution to expanding FRED's range of motion.

Antenna

Originally there was to be a servo that would "pluck" the antenna rather than just wave it back and forth. With all of the work that went into FRED's creation, there was not enough time or space to put this function into effect without decreasing some of the aesthetic qualities he has.

Motor Movement Functions

FRED has several functions that send signals to the motors and interface with said motors to generate his motion. These functions are labeled in the Appendix as forward, backward, left, right, slightLeft, slightRight, slightLeftBack, slightRightBack, and stop. There is a very large idiosyncrasy with FRED in that any function that references backwards actually moves FRED forward and vice versa. This strange phenomenon arises from the fact that originally FRED was designed with its back side as the original front of the robot. During the sensor testing phase, however, I found that the sensor wiring and LCD screen favored FRED moving the other direction. I started testing this way, moving backward instead of forward, with the intent that eventually I would mount his sensors in my original orientation but FRED worked well enough in this setup that there was no reason to revert to the original design.

All the movement functions have a similar pattern in code, with much of the inspiration coming from Mike and Thomas' servo control sample code for the PVR board[3]. Each of the functions generally start by powering on and off various leads, sending the motor driver the direction a given wheel should spin. Then, as a means of fool proofing against a careless programming error that could break or overload the motors or driver, the input value is limited to being between 0 and 100. The internal timers are then referenced, taking the value input and multiplying it by a constant to put that value in a range such that 100 will generate the maximum pulse width modulus (pwm) the motor can receive and 0 will generate the minimum.

The forward and backward functions cause both motors to run in the same direction as each other at the percent of max speed specified by the user as the input of the function. The right and left functions do the same as the forward and backward functions but instead of the motors turning the same direction, they turn in the opposite direction as each other. These two are the only functions not affected by FRED's idiosyncrasy, since turning either direction is the same regardless of direction being faced.

The ‘slight’ movement functions cause one of the wheels to operate at 30 % of the other wheel, causing a slight turn. For instance slightRight causes the right wheel to move 30% of the speed of the left wheel. slightRight and slightLeft move FRED in the direction of the forward function and slightRightBack and slightLeftBack move FRED in the direction of the backward function.

Lastly, the stop function sets the wheels to no direction, or coasting, and delays a tenth of a second. The stop function was designed to provide the wheels a transition between movement functions to reduce spikes caused from a complete reversal of the motor.

The reason for so many different functions for movement is to allow limited intelligence with regards to relative motion. For instance seeing a wall is 5 feet away should cause a different turning reaction than seeing a wall 2 feet away. Instead, at 5 feet FRED starts turning away from the obstacle with a slight movement function. If the object get within 2 feet and FRED hadn’t avoided it altogether, the much more dire pure directional function (like right or left) would be used to have FRED turn on point and continue moving unobstructed.

Lessons Learned

Because of the complexity of the motor driver, determining which pins determined which directions was primarily trial and error. Instead of making the function then determining what switches to activate, it was more along the lines of throw the switches then label its behavior. Originally I had a couple other movement functions, such as attempts to make acceleration leading into and out of movement, but it ended up being far more glitch than useful, and considering FRED actually moves very smoothly and continuously, it didn’t end up being a large issue.

Originally I had intended to have a label for whenever FRED was moving forward, backward, or turning. These labels were later deemed unnecessary and now remain as vestigial code inside the function.

Sensors

Obstacle Avoidance

In its basest of functions, FRED drives around randomly, avoiding obstacles. Using two front mounted IR transmitters/receivers it determines whether it is approaching an obstacle (or an obstacle is approaching it) to within certain proximity, at which point it will turn away from the perceived object and continue driving

In the case where the IR fails to detect an object in its movement range, a collision with that object will result in the robot moving in the opposite direction, altering its trajectory, and continuing on with its task.

Heat Sensing

FRED uses 4 motion sensors mounted in an arc on the front of the robot. These sensors are used to detect heat sources in the local proximity, and when a heat source moves, these sensors read high. They spend about 20 seconds to get acquainted with the IR emissions of the room, then after that point, any movement from a heat radiating body will be noticed.

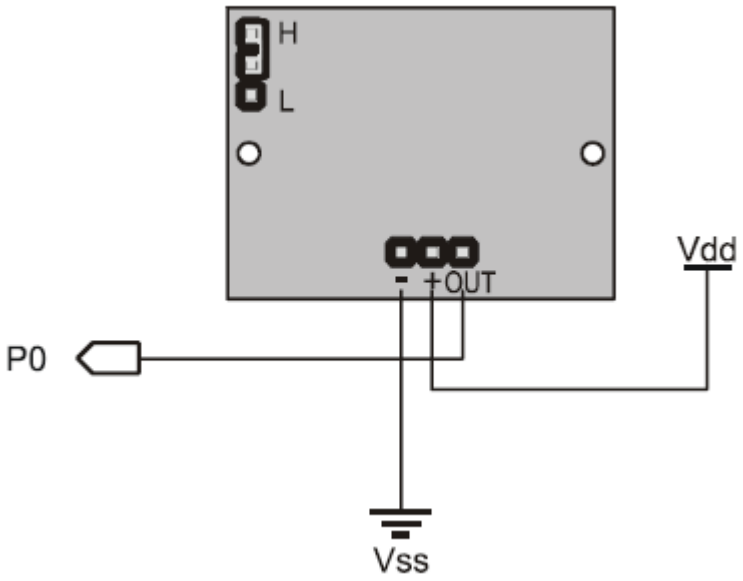
Using the motion sensors in this way was actually a learning process in itself. I knew that I wanted to use these sensors somewhere in the design but at first I didn't know what for. The main breakthrough actually began as a mistake from FedEx. During the week of the obstacle avoidance demo I learned that they lost my package of sensors in the mail, including my IR rangefinders, which meant that I would certainly end up with an incomplete for that week. Determined to avoid obstacles however I could, I turned to the only sensors I had picked up at that point: Two motion sensors.

Now ordinarily one might think that using a motion sensor attached to a moving object would be a futile effort. In fact, mounting the sensors to a moving platform was the only way one could get obstacle avoidance from a motion sensor. Using masking tape to limit the visible area of the motion sensor, primarily the ability to look down or up, would solved the largest problem of motion everywhere. The idea was as FRED would approach a wall, the motion sensors would view it as the wall moving past FRED, activating a motion sensor that would tell the robot to turn away from the wall and continue.

The problem with this plan was illustrated in several tests of this theory. It would turn away well enough from people, but the obstacle avoidance was failing to avoid the wall more often than not. Upon discussing this problem with some of my fellow lab students, one pointed out that they only sensed moving *heat*. Although their usefulness at obstacle avoidance failed unless the walls breathed, it meant that using only motion detectors, I could detect a heat source, moving or not, in the vicinity of the robot. FRED would merely have to turn in a circle and even a stationary heat source would activate the sensor(s). Preliminary trials using the motion detectors to track a heat source was capable of keeping a cat within a 60 degree arc in front of FRED.

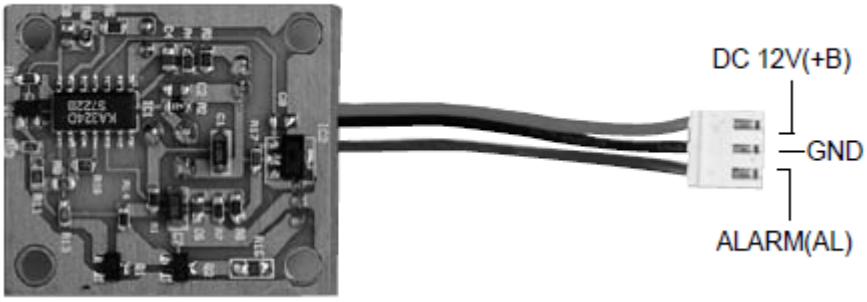
Heat Sensors

- 2 Passive Infra-Red (PIR) Sensor from Parallax Inc
    - Part # 555-28027
    - **Parallax Inc.**
      599 Menlo Drive
      Rocklin, CA 95765
      USA
    - **(888)-512-1024**



The parallax PIR sensors were used as the front two motion sensors. Their purpose was to make sure the motion source remained directly in front of FRED while he was attempting to locate it.

- 2 PIR motion sensors from Sparkfun
    - SEN-08630
    - SparkFun Electronics
      6175 Longbow Drive
      Suite 200
      Boulder, CO 80301
    - **1-303-284-0979**



Specifications

ITEM		Specification	Unit	Condition
Sensor Type		Dual Element		
Housing		TO 5		
Element Size		2×1	mm	
Spacing		1	mm	
Responsivity	Min Typ	3.2 4.0	xv/w	7...14mm, 1Hz,100℃ (One element cover)
Match	Max	<10	%	7...14mm, 1Hz,100℃ (Both element expose)
Noise	Typ Max	20 50	μVp-p V	25℃, 0.4...10Hz
Effect Voltage	Min Max	0.2 1.5		Re=47XO
Window Material		Silicon, coated		
Spectral Range	Transmission Blocking	T>30 average T<0.1	%	7...14mm <5mm
Operating Voltage		12	V	
Operating Temperature		-10~40	℃	
Storage Temperature		-40~80	℃	

The sparkfun PIR sensors were used as the right and left motion sensor. Their goal was to ensure that if, during locate mode, a movement source was detected on the right or left, that FRED could turn to face that direction. Also because neither sensor can see directly in front of

the robot, if motion is detected from these sensors it means that FRED is still not centered enough.

Both sensors work very similarly, as they have the same general hardware. Both work by taking several seconds to take a snapshot of the room. If change is detected, the sensor will transmit a high value to the board. The differences is the Sparkfun PIRs required a pull-up resistor to be installed onto the back to be useful for detection, although they also have a lower startup time of a couple seconds vs. the 5 or so seconds required for the Parallax sensors to get up to speed. Also, the Sparkfun PIRs had a flat back but the wires connected from the side of the sensors, while Parallax PIR's had a very contoured back side with the port for connecting the signal wires to them on the back, which was more convenient for my purposes.

	Parallax PIR	Sparkfun PIR
Boot up time	5 seconds	2 seconds
Duration of high value	2 seconds	2 seconds

The algorithm that utilized both sets of PIRs was the scanHeat2 and shy functions. scanHeat2 is used for locate mode while shy is used for flee mode.

scanHeat2

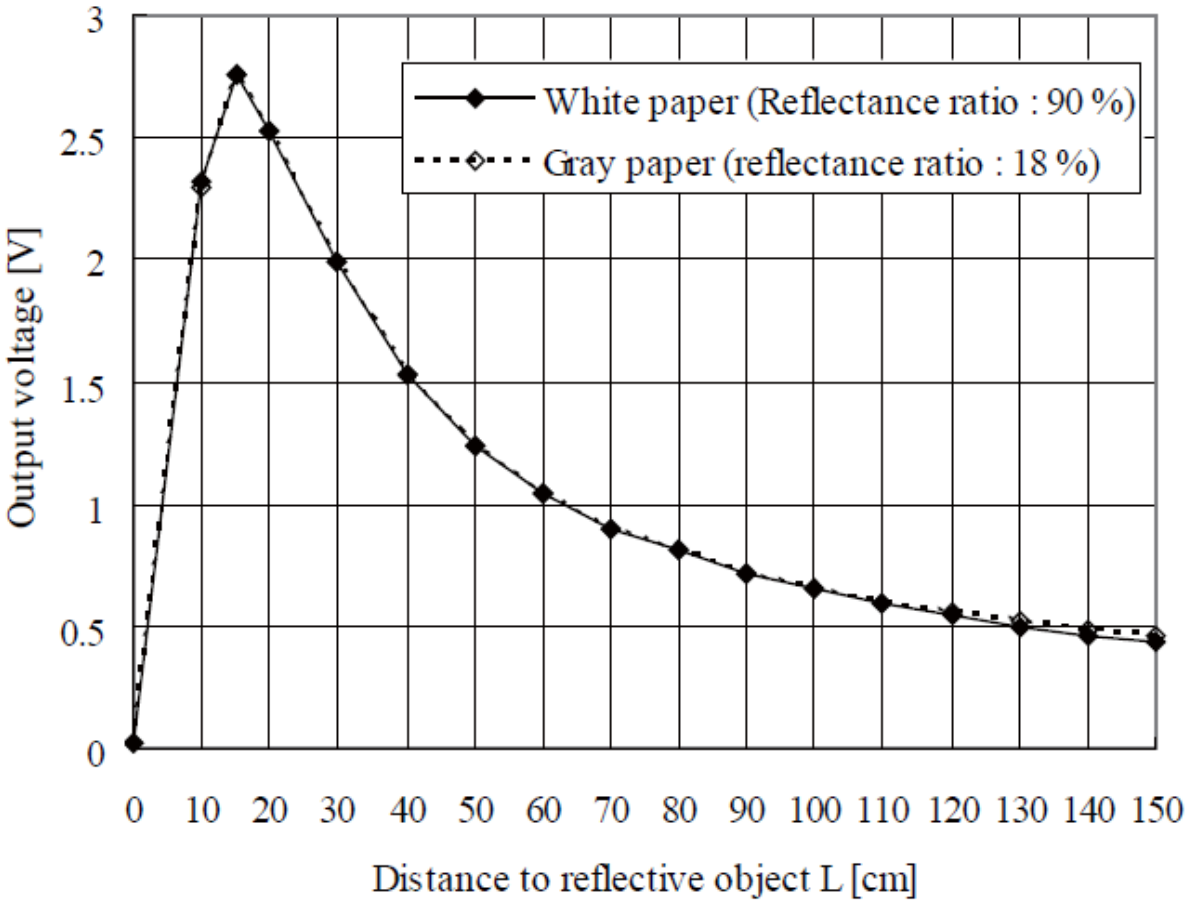
The second conception of my heat sensing program, scanHeat2 utilizes pauses between scans and a vast series of conditions relating the motion sensors that have been tripped to patterns of movement FRED should be executing. The change that was made in moving from the scanHeat2 function to the scanHeat function is the understanding that after being tripped, a PIR will stay on a full two seconds before turning off. This was probably the largest limiting factor in my design, and if I continued with this project I would probably try hacking the sensors to yield brief values and hence more of them could be taken in the same amount of time. Such a process is possible, as mentioned by a hobbyist who described some of the intricacies of the Parallax motion sensor [2], albeit somewhat difficult for someone with limited experience with circuits. Originally FRED would take values until one of the sensors was read high for over four consecutive readings and then act on the data. The problem was that even though they may seem high, the PIRs were really just affected by the delay and that many of these scans were sampling the stagnant data.

To solve this problem, a two second delay was added between the movements and scanning so FRED's own movement would not give false positives on the PIR sensors. Then because taking four scans at 2 seconds per scan seemed time intensive and overkill, the filtering was removed and only one scan was taken every 2 seconds and subsequently acted upon.

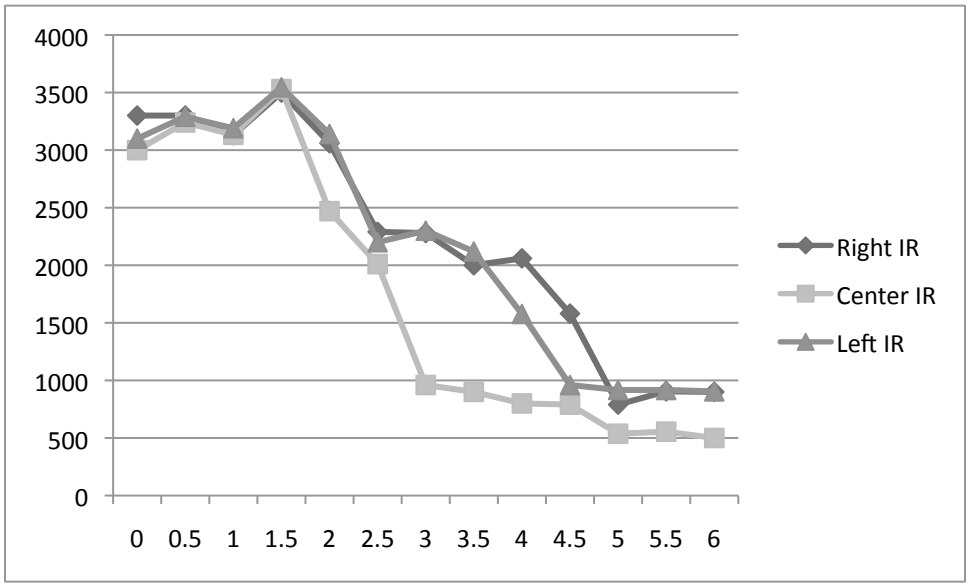
Shy

The shy function plays more of a supporting role than the direct role played by the scanHeat2: It advises more than anything else. FRED will continue moving forward regardless but if motion is detected from his IRs he will slowly turn away from the sensor which returned the reading. This makes shy the closest function to the original design concept of having free-running motion sensors which gave values while moving. As the original idea had some problems, the motion sensors used in this way merely add an element of random movement loosely based on perceived motion.

- 3 Long Range Infrared Rangefinders from Sharp
  - Part # GP2Y0A02YK0F
  - Sharp Electronics Corporation
    Sharp Plaza
    Mahwah, NJ
    07495-1163
  - **(201) 529-8200**



Specification Sheet Predicted Performance Data



Experimentally Determined Rangefinder Values (in feet)

Table: Recorded Distance Values for Rangefinders

Distance	Left IR	Center IR	Right IR
0	3100	3000	3300
0.5	3290	3240	3300
1	3190	3133	3130
1.5	3545	3530	3500
2	3140	2470	3060
2.5	2200	2010	2290
3	2300	960	2280
3.5	2120	900	2000
4	1577	800	2060
4.5	960	790	1580
5	917	537	790
5.5	917	555	905
6	905	500	900

All the obstacle avoidance was done using these rangefinders. Certain threshold values were set in my obstacle avoidance function called 'avoid'. When a sensor reached a threshold value, FRED would turn in the direction opposite of the sensor, ensuring the wall or object was avoided. During location mode, the central IR transforms into a means of determining if a cat is close enough to enter flee mode. The threshold for this is set using the front bump sensor during the boot up phase.

Lessons Learned

Motion sensors have been very nice in their battery efficiency. Throughout testing there was only one time FRED's batteries died. Most of the battery consumption came from the motors though certainly. However, motion sensors seem very glitchy: They read false positives sometimes, so having the four sensors taking readings instead of just two was a definite improvement. Also to reliably pick up on motion, the movement being sensed should be a heat source or have a source of heat. However, the sensors can still pick up on other movement so it is important to take FRED's own movement into account when reading the motion sensor values.

The rangefinders are very effective at locating walls, but because FRED uses the long range kind, the values saturate at about half a foot. To compensate, FRED starts avoiding obstacles long before they get near to him, using the 'slight' functions to make slow meandering corrections. This also means, however, that FRED cannot be tested in a confined space, because he would see obstacles all the time and get stuck in a perpetual spin. This is fine with his purpose though, because he is meant to wander around a house seeking cats to play with. The ability to see an obstacle long before he reaches it allows FRED to have a very smooth motion when driving around and the high speed of his motors makes him able to travel vast distances in a short period of time if he needs to.

Behaviors

If left to its own devices, FRED will wander around avoiding walls and obstacles. After locating a cat, FRED will slowly approach. Once it is close enough, the robot will flee the cat, using the pyroelectric sensors on the sides to turn away from approaching heat sources. After a set amount of time or if no cat is detected near the robot, it will resort back to its original roving behavior.

This ideal behavior pattern is actually a combination of a few useful behaviors, which will subsequently be shown.

Boot Sequence

While not directly involved in the behavior this has some key functionality that relates to the behaviors later. Once the power switch is flipped, the boot up sequence starts. The user is permitted to calibrate the center IR so changes in lighting won't negatively affect his programmed behavior. Once this calibration is done, there is a brief delay and the main behaviors start.

Wall Avoiding

With the table of recorded IR readings, FRED is programmed so that IR readings in its highest range will cause a very tight turn away from its closest barrier. If both barriers are roughly the same distance away, it solves the dilemma by choosing left. This turn is executed by putting one wheel in forward motion and the other in reverse motion. This motion causes FRED to turn in place. If the IR range is high but not yet above the maximum threshold, a slow turn away from the near wall is executed. The slow turn is performed by making the wheel closest to the barrier to spin full speed forward and the wheel furthest at half speed. If no such barrier is nearby, both wheels will spin forward at full speed causing forward motion with a slight tilt to the left due to castor placement.

Heat Sensing

The 4 Pyroelectric Infrared sensors at the front of the robot create a wall of sensors allowing a heat source to be centered. The 4 sensors will be referred to as LL, LR, RL, and RR from now on, referring to each of the sensors from the leftmost to the rightmost. The same labeling system is used in the actual program. If LL is high and the others are low, it means that a heat source is detected to the left of the robot's vision so FRED will turn to the left to try to put the object in view. In doing this, the motion sensors LR and RL will slowly turn on. Once both LR and RL are on, the object is seen as being centered.

Approach

LR and RL will be at slight angles though, since otherwise the heat source would fall out of focus when FRED approached it. Once LR and RL are both activated, FRED will start moving forward. In the case where the cat is seen by two of the left or two of the right PIR sensors, FRED will turn and move forward slightly as it accomplished a little bit of both the other motion sensor requirements and because if FRED only moved forward when the both the front PIRs saw it may take forever to get close enough to enter flee mode. At angles, moving forward would cause the object to seemingly drift right from the vantage point of sensor LR and to the left from sensor RL throughout movement, leaving those sensors tripped. The central mounted Rangefinder is really the means of determining distance to the heat source. Once the desired distance is reached, the approach behavior ends. If at any point LR or RL lose sight of the heat source, FRED will look left and right. If only one of those sensors is tripped, he will start into the heat sensing behavior to try to acquire those sensors again.

Flee

Upon locating the heat source and getting close enough to it, FRED turns sharply around and enters flee mode represented by flee = 1 in the code. While in flee mode he will continue obstacle avoiding, though if a moving heat source is detected from his PIR, he shifts his driving direction away from the perceived motion. This continues for a set amount of time, or until no motion is detected for a shorter set amount of time.

Lessons Learned

Because all the timers were made in terms of times cycled through the code, the same set time could vary based on functions added or removed.

Experimental Layout and Results

IR Long Rangefinder

To measure the range of the IR Rangefinder, I set FRED to relay the value of the port that was tied to the sensor, having it feed updated values roughly every 200 ms. Then using masking tape and a ruler, I marked the floor every foot until 6 feet had been sectioned off. I then put an object at the 6 ft marker and recorded the sensor value in a spreadsheet. I then moved the marker to 6 inches closer (halfway between 2 tape markers) and recorded the value. I continued this process until I made it within 6 inches of the sensor.

Next I took a bicycle light and pointed it directly into the receiver and repeated the testing for key values to attempt to see a trend. Most of the new values were higher, which would be understandable since more light was being recorded for the same distance, though the change was a fairly constant change for all the recorded values.

Lastly, I used another test to determine the relationship of light to IR readings. I placed the object at the 2 ft marker, a marker that was sufficiently within the range of dynamic readings, being far away from both the maximum and minimum recorded readings, and moved the bicycle light around, observing how the reading changed. While this method did not give me any concrete values like the other two tests, it did give me a stronger understanding of the cause/effect relationship of light on the sensor. Closer light sources and brighter light sources raised the reading by a maximum of about 200 higher, though as the light source was pulled farther away and less light reached the sensor, the reading slowly fell back to the readings from the first test.

Comparing the graph of the 3 rangefinders with the approximate value on the rangefinder data sheet, it can be seen that they both follow the same trend. There are, however, a few more bumps in the experimental value which is understandable, considering the datasheet is ideal conditions. Also the ADCA value is what the experimental rangefinder data is measured in instead of voltage. Since ADCA value is not necessarily linear, the values were probably pretty close. All three of the experimental rangefinders yielded almost the same values, however the center IR exhibited poor distance sight. This works fine, however, since its main use is not avoiding from afar but determining distance from cat and turning in two tight of spaces.

Pyroelectric Infrared Sensors

After proving useless at helping the robot avoid walls, the PIR sensors gained usefulness in other areas. As it is very difficult to obtain dynamic values from a true/false sensor, I do not have the

same detailed sensor data. However, there are some key values and properties that were observed. At 3.3V, the active range of the sensor as measured in my apartment was between 6 and 6.5 feet. The uncertainty range is because in different environments the actual range is sure to differ. This range was recorded by me starting out of the sensor's range but right in front of it, then stepping forward slowly, all the while flailing my arms. Once FRED reported that he saw me moving, I measured the distance from where my arms had previously been flailing to the motion sensor.

Conclusion

Ultimately, FRED was largely successful. While there were some pieces of my initial plan that I never got to implement such as a servo controlled antenna or scanning while moving, I was able to accomplish several things I did not expect to when starting the project. The biggest constraint with my robot was that since I am not an electrical engineer, many of the tools needed to work on FRED such as wire crimpers, header pins, wire and solder were things I could only use at lab, which meant work on FRED was limited to 3 times a week. So time and lack of tools played a big role. FRED locates pretty well although I would like to improve his reliability and perhaps fine tune the functions a little more. I would also like to hack the PIR sensor so it didn't stay active so long after a reading. That would speed FRED up tremendously.

If I started the project over I would probably have ended the semester with an extra month for tuning because a lot of my time was spent constructing parts that I ended up unable to use and making up for careless data sheet reading errors by constructing complicated circuits. I would also raise the bump sensor so it wouldn't drag on carpet and make a cable that ran from the programmer prongs of the board to outside the wooden skeleton so I wouldn't have to dismantle FRED every time I wanted to make a programming change.

Documentation

[1] OBESITY IN CATS... and What To Do About An Overweight Cat
http://www.thepetcenter.com/article.aspx?id=3401

[2] Parallax Motion Detector from Radio Shack
http://www.neufeld.newton.ks.us/electronics/?p=208

[3] Mike and Thomas' PVR Board Documentation

[4] IMDL website and class notes from Dr. Arroyo
http://www.mil.ufl.edu/imdl/handouts.htm

Appendices

Program Code:

```
#include <avr/io.h>
#include "PVR.h"

int F = 0;                      //Forward running if 1
int B = 0;                      //Backward running if 1
int R = 0;                      //Right Turning if 1
int L = 0;                      //Left Turning if 1
int msl = 0;        //Motion Sensor Filter Left
int msr = 0;        //Motion Sensor Filter Right
int msbl = 0;       //Motion Sensor Filter Back Left
int msbr = 0;       //Motion Sensor Filter Back Right
int noheat = 0;     //No heat detected variable
int fleecnt = 0;//Counts duration of fleeing
int v = 35;                     //Average running velocity
int IRT = 0;        //Variable for use with IRTest
int left1 = 0;      //For use with vision filter
int left2 = 0;      //For use with vision filter
int leftave = 0;//For use with vision filter
int mid1 = 0;       //For use with vision filter
int mid2 = 0;       //For use with vision filter
int midave = 0;     //For use with vision filter
int right1 = 0;     //For use with vision filter
int right2 = 0;     //For use with vision filter
int rightave = 0;//For use with vision filter
int stopScan = 0;//For use with stopSense function
int boot = 0;       //For Initial boot process
int centerIR = 3000;        //Variable to store middle IR threshold
int bumpRedundancy = 0; //Fix problem where bump sensor would activate repeatedly
int bumpReset = 0;          //Works with bumpRedundancy

//Behaviors//
int locate = 0;     //True when FRED is searching for a heat source
int flee = 0;       //True when FRED is fleeing a heat source

//------------INFO--------------//
//ADCA0 is Motion Sensor LL                 //
//ADCA1 is Motion Sensor RR                 //
//ADCA2 is Motion Sensor LR                 //
//ADCA3 is Motion Sensor RL                 //
//ADCA4 is Long Range IR R                  //
//ADCA5 is Long Range IR L                  //
//ADCA6 is Long Range IR M                  //
//ADCA7 is Unused                                   //
//-------------------------------//
```

```
void forward(int value);              //FRED toward the face that wheels are closest to
void backward(int value);             //Moves toward the face that wheels are farthest from
void right(int value);                        //Turns Right
void slightRight(int value);          //Left wheel full speed, right wheel half speed
void left(int value);                         //Turns Left
void slightLeft(int value);           //Left wheel half speed, right wheel full speed
void stop(void);                              //Stops the motor and turns off directional switches
void senseMotion(void);                       //Controls msr and msl, reads motion sensors
void shy(int time);                                   //FRED runs from heat sources
void scanHeat(void);                          //FRED turns to find heat sources
void scanHeat2(void);                         //scanHeat, but better
void avoid(void);                             //Obstacle Avoidance
void slightLeftBack(int value);       //Same as slightLeft but backward
void slightRightBack(int value);//Same as slightRight but backward
void reportValue(void);                       //Display selected value on line 1 of LCD and Motion Sensors on line 2
void IRTest(void);                                    //Scroll Front IR values on bottom of LCD screen
void visionFilter(void);              //Takes average of IR range data and saves them as ___ave
void behaviorCaller(void);            //Displays on Line 1 of the LCD what behavior is active
void verifyHeat(void);                        //Determines if flee would be initiated if a heat source is truly in front
of fred.
void stopSense(void);                         //Alternates FRED between moving and sensing to avoid false
positives.

void main(void){
        xmegaInit();                                                    //setup XMega
        delayInit();                                                    //setup delay functions
        ServoCInit();                                                   //setup PORTC Servos
        ServoDInit();                                                   //setup PORTD Servos
        ADCAInit();                                                             //setup PORTA analong
readings
        lcdInit();                                                      //setup LCD on PORTK
        lcdGoto(1,0);                                                   //move LCD cursor to the second
line (Line 1) of LCD
        PORTF_DIR = 0xFF;                                               //Specifies Port F as output in
direction register.  0x00 is input
        PORTQ_DIR |= 0x01;                                              //set Q0 (LED) as output
        PORTC_DIR = 0x00;

        //Initial Boot
                lcdGoto(0,0);
                lcdString("Initialize FRED ");                  //      Display "!!Motion!!" when motion is
detected
                delay_ms(2000);
                lcdGoto(0,0);
                lcdString("Set Main Bumper ");
                while(boot == 0){
                        if (PORTC_IN == 0b00000001){
                                centerIR = ADCA6();
                                boot = 1;
                        }
                        else{
                                delay_ms(500);
                        }
                }
                lcdGoto(0,0);
                lcdString("   Thank You    ");
```

```
		lcdGoto(1,0);
		lcdString("Starting up FRED");
		delay_ms(2000);
		lcdGoto(0,0);
		lcdString("               ");
		lcdGoto(1,0);
		lcdString("               ");

	while(1){
		behaviorCaller();                              //Displays current mode on line 1 of LCD
		visionFilter();                                        //Create ____ave IR values
		stopSense();                                           //Counts how long til scan begins.
		senseMotion();                                         //Activate msr and msl
		if(locate==0){
			avoid();                                       //Drive around avoiding obstacles
		}
		if (locate==1){
			scanHeat2();
		}
		if (flee == 1){
			fleecnt ++;
			shy(100);
			if (fleecnt==20){
				flee = 0;
				locate = 0;
			}
		}
		if (bumpRedundancy>0){
			bumpReset++;
			if(bumpReset > 9){
				bumpRedundancy = 0;
				bumpReset = 0;
			}
		}
		reportValue();
	}
}

//Functions//
void avoid(void){
//		if (PORTC_IN == 0b00000001 && bumpRedundancy == 0){		//Bump Sensor
//			forward(v);
//			delay_ms(1200);
//			right(v);
//			delay_ms(1000);
//			bumpRedundancy = stopScan;
//		}
//		else{

			if (leftave>3000 && rightave>3000){
				left(v);
				delay_ms(1000);
			}
			else if (midave>centerIR && locate == 0){
				left(v);
				delay_ms(1000);
```

```
                        }
                        else if (rightave>3000){
                                right(v);
                                delay_ms(600);
                        }
                        else if (leftave>3000){
                                left(v);
                                delay_ms(600);
                        }
                        else if (leftave>2500){
                                slightLeftBack(v);
                                delay_ms(500);
                        }
                        else if (rightave>2500){
                                slightRightBack(v);
                                delay_ms(500);
                        }
                        else{
                                backward(v);
                        }
//                      }
//          }
}

void reportValue(void){
        int MS = 0000;
        lcdGoto(1,0);
        if (ADCA0() < 2500){
                lcdGoto(1,0);
                lcdString("X");
        }
        else{
                lcdGoto(1,0);
                lcdString("-");

        }
        delay_ms(1);
        if (ADCA1() < 2500){
                lcdGoto(1,1);
                lcdString("X");
        }
        else{
                lcdGoto(1,1);
                lcdString("-");
        }
        delay_ms(1);
        if (ADCA2() > 3000){
                lcdGoto(1,2);
                lcdString("X");
        }
        else{
                lcdGoto(1,2);
                lcdString("-");
        }
        delay_ms(1);
        if (ADCA3() > 3000){
```

```
                lcdGoto(1,3);
                lcdString("X");
        }
        else{
                lcdGoto(1,3);
                lcdString("-");
        }
        delay_ms(1);
}

void scanHeat(void){

        if (msl == 0 && msr == 0){
                        right(v);
                        delay_ms(150);
                        stop();
                        noheat ++;
                        if (noheat == 5){
                                locate = 0;
                                flee = 0;
                                noheat = 0;
                        }
        }
        if (msl == 0 && msr > 4){
                right(v);
                delay_ms(100);
                stop();
                noheat = 0;
        }
        if (msl > 4 && msr == 0){
                left(v);
                delay_ms(100);
                stop();
                noheat = 0;
        }
        if (msl > 4 && msr > 4){
                backward(v);
                delay_ms(100);
                if (midave>centerIR){
                        locate=1;

                }
                noheat = 0;
        }
}

void scanHeat2(void){
        msl=0;
        msr=0;
        stop();
        delay_ms(2300);
        senseMotion();
        if (msl == 0 && msr == 0 && msbl == 0 && msbr == 0){
                        noheat ++;
                        right(v);
                        delay_ms(1000);
```

```
                stop();
                if (noheat == 2){
                        locate = 0;
                        flee = 0;
                        noheat = 0;
                }
}
else if (msl > 0 && msbl == 0 && msr == 0 && msbr == 0){
        left(v);
        delay_ms(200);
        stop();
        backward(v);
        delay_ms(200);
        noheat = 0;
                if (midave>centerIR){
                        locate=0;
                        verifyHeat();
                        flee=1;
                }
}
else if ((msl == 0 && msbl == 0) && (msr > 0 && msbr == 0)){
        right(v);
        delay_ms(200);
        stop();
        backward(v);
        delay_ms(200);
        noheat = 0;
                if (midave>centerIR){
                        locate=0;
                        verifyHeat();
                        flee=1;
                }
}
else if(msl > 0 && msbl > 0 && msr == 0 && msbr == 0){
        left(v);
        delay_ms(250);
        stop();
        noheat = 0;
}
else if(msl == 0 && msbl == 0 && msr > 0 && msbr > 0){
        right(v);
        delay_ms(200);
        stop();
        noheat = 0;
}
else if (msl == 0 && msbl > 0 && msr == 0 && msbr == 0){
        left(v);
        delay_ms(300);
        stop();
        noheat = 0;
}
else if ((msl == 0 && msbl == 0 && msr == 0 &&  msbr > 0)){
        right(v);
        delay_ms(300);
        stop();
        noheat = 0;
```

```
		}
		else if (msl > 0 && msr > 0){
			if (midave>centerIR){
				locate=0;
				verifyHeat();
				flee=1;

			}
			else{
				backward(v);
				delay_ms(500);
				if (midave>centerIR){
					locate=0;
					verifyHeat();
					flee=1;
				}
			}
			noheat = 0;
		}
}

void shy(int time){
		if (msl == 0 && msr == 0){
			backward(v);
			delay_ms(100);
			stop();
			noheat ++;
			if (noheat == 10){
				locate = 0;
				flee = 0;
				noheat = 0;
			}
		}
		else if (msl == 0 && msr > 4){
			slightLeftBack(v);
			delay_ms(100);
			stop();
			backward(v);
			delay_ms(300);
			stop();
			noheat = 0;
		}
		else if (msl > 4 && msr == 0){
			slightRightBack(v);
			delay_ms(100);
			stop();
			noheat = 0;
		}
		else if (msbr == 0 && msbl > 4){
			slightRightBack(v);
			delay_ms(200);
			stop();
			noheat = 0;
		}
		else if (msbr > 4 && msbl == 0){
			slightLeftBack(v);
```

```
                delay_ms(200);
                stop();
                noheat = 0;
        }
        else if (msl > 4 && msr > 4){
                backward(v);
                delay_ms(100);
                if (midave>centerIR){
                        locate=1;

                }
                noheat = 0;
        }
}

void forward(int value){                        //Ranged from 0 to 100%
                B = R = L = 0;
                F = 1;
                PORTF_OUT = 0b00000110;                 //must include '_out' to specify output
                if (value > 100){
                        value = 100;
                }
                else if (value <0){
                        value = 0;
                }
                TCD1_CCA = 100*value;           //Generate PWM for left wheel
                TCD1_CCB = 100*value;           //Generate PWM for right wheel.  Varies from 0 to 10k
(possibly 100k)
}

void backward(int value){                       //Ranged from 0 to 100%
                F = R = L = 0;
                B = 1;
                PORTF_OUT = 0b00001001;                 //must include '_out' to specify output
                if (value > 100){
                        value = 100;
                }
                else if (value <0){
                        value = 0;
                }
                TCD1_CCA = 100*value;           //Generate PWM for left wheel
                TCD1_CCB = 100*value;           //Generate PWM for right wheel.  Varies from 0 to 10k
}

void slightLeftBack(int value){                 //Ranged from 0 to 100%
                F = R = L = 0;
                B = 1;
                PORTF_OUT = 0b00001001;                 //must include '_out' to specify output
                if (value > 100){
                        value = 100;
                }
                else if (value <0){
                        value = 0;
                }
                TCD1_CCA = 100*value;           //Generate PWM for left wheel
                TCD1_CCB = 30*value;            //Generate PWM for right wheel.  Varies from 0 to 10k
```

```
}

void slightRightBack(int value){     //Ranged from 0 to 100%
                F = R = L = 0;
                B = 1;
                PORTF_OUT = 0b00001001;                 //must include '_out' to specify output
                if (value > 100){
                        value = 100;
                }
                else if (value <0){
                        value = 0;
                }
                TCD1_CCA = 30*value;            //Generate PWM for left wheel
                TCD1_CCB = 100*value;           //Generate PWM for right wheel.  Varies from 0 to 10k
}

void left(int value){
                F = B = R = 0;
                L = 1;
                PORTF_OUT = 0b00000101;                 //must include '_out' to specify output
                if (value > 100){
                        value = 100;
                }
                else if (value <0){
                        value = 0;
                }
                TCD1_CCA = 100*value;           //Generate PWM for left wheel
                TCD1_CCB = 100*value;           //Generate PWM for right wheel.  Varies from 0 to 10k
}

void slightLeft(int value){
                B = R = L = 0;
                F = 1;
                PORTF_OUT = 0b00000110;                 //must include '_out' to specify output
                if (value > 100){
                        value = 100;
                }
                else if (value <0){
                        value = 0;
                }
                TCD1_CCA = 40*value;            //Generate PWM for left wheel
                TCD1_CCB = 100*value;           //Generate PWM for right wheel.  Varies from 0 to 10k
(possibly 100k)
}

void right(int value){
                F = B = L = 0;
                R = 1;
                PORTF_OUT = 0b00001010;                 //must include '_out' to specify output
                if (value > 100){
                        value = 100;
                }
                else if (value <0){
                        value = 0;
                }
                TCD1_CCA = 100*value;           //Generate PWM for left wheel
```

```
		TCD1_CCB = 100*value;		//Generate PWM for right wheel.  Varies from 0 to 10k
}

void slightRight(int value){
		B = R = L = 0;
		F = 1;
		PORTF_OUT = 0b00000110;			//must include '_out' to specify output
		if (value > 100){
			value = 100;
		}
		else if (value <0){
			value = 0;
		}
		TCD1_CCA = 100*value;		//Generate PWM for left wheel
		TCD1_CCB = 40*value;		//Generate PWM for right wheel.  Varies from 0 to 10k
(possibly 100k)
}

void stop(void){
		PORTF_OUT = 0b00000000;			//must include '_out' to specify output
		F = B = L = R = 0;
		delay_ms(100);
}

void behaviorCaller(void){
	if (locate == 1){
		lcdGoto(0,0);
		lcdString("Locate");
	}
	else if (flee == 1){
		lcdGoto(0,0);
		lcdString("Flee  ");
	}
	else{
		lcdGoto(0,0);
		lcdString("Random");
	}
}

void stopSense(void){
	stopScan++;
	if (locate == 0 && stopScan==45){			//Transition from random to locate
		stopScan = 1;
		locate=1;
	}
	else if (locate == 1 && stopScan==25){		//Transition from locate to random
		stopScan = 1;
		locate=0;
	}
	else if (flee == 1 && stopScan==60){		//Transition from locate to flee
		stopScan=1;
		locate=0;
		flee=0;
	}
}
```

```
void verifyHeat(void){
	stop();
				lcdGoto(0,0);
				lcdString("Flee  ");
				stop();
				left(v);
				delay_ms(2000);
				backward(v);
				delay_ms(500);
				flee=1;
				locate=0;
}

void IRTest(void){
	if (IRT==1){
		lcdGoto(1,0);
		lcdString("              ");
		lcdGoto(1,0);
		lcdString("Left  ");
		lcdInt(leftave);
		delay_ms(10);
	}
	if (IRT==60){
		lcdGoto(1,0);
		lcdString("              ");
		lcdGoto(1,0);
		lcdString("Mid   ");
		lcdInt(midave);
		delay_ms(10);
	}
	if (IRT==120){
		lcdGoto(1,0);
		lcdString("              ");
		lcdGoto(1,0);
		lcdString("Right ");
		lcdInt(rightave);
		delay_ms(10);
	}
	if (IRT==180){
		IRT = 0;
		delay_ms(10);
	}
	else{

		delay_ms(10);
	}
	IRT++;
}

void visionFilter(void){
	left1 = left2;
	left2 = ADCA4();
	leftave = (left1+left2)/2;
	delay_ms(10);

	right1 = right2;
```

```
		right2 = ADCA5();
		rightave = (right1+right2)/2;
		delay_ms(10);

		mid1 = mid2;
		mid2 = ADCA6();
		midave = (mid1+mid2)/2;
		delay_ms(10);

}

void senseMotion(void){
//Left Motion Sensor Controller

			if (ADCA2()>3000){			//  3100 is low value for ADC
				msl++;				//	left motion sensor filter
			}
			delay_ms(1);
			if (ADCA2()<3000){			//	280 is low value for ADC
				msl = 0;
			}
			delay_ms(1);

//Right motion Sensor
			if (ADCA3()>3000){			//  3100 is low value for ADC
				msr++;				//	left motion sensor filter
			}
			delay_ms(1);
			if (ADCA3()<3000){			//	280 is low value for ADC
				msr = 0;
			}
			delay_ms(1);

//Back Left Motion Sensor
			if (ADCA0()<2500){		//  3700 is high value for ADC
				msbl++;		//	left motion sensor filter
			}
			if (ADCA0()>2500){		//	280 is low value for ADC
				msbl = 0;
			}
//Back Right Motion Sensor
			if (ADCA1()<2500){		//  3700 is high value for ADC
				msbr++;		//	left motion sensor filter
			}
			if (ADCA1()>2500){		//	280 is low value for ADC
				msbr = 0;
			}
//		}
		if(flee==0){
			if (msl > 12 || msr > 12){
				locate=1;
			}
		}

}
```