p

University of Florida
Department of Electrical and Computer Engineering
EEL5666C
Intelligent Machines Design Laboratory

Final Report (08/01/05)

Bi-Mode

Joel Padilla
TAs: William Dubel
Steven Pickles
Instructor: A. Arroyo
E. Schwartz

Table Of Content

***Disclaimer: All credits will be given in this section, and not during the paper**

Abstract

Bi-Mode is an autonomous multi behavior robot. The robot will have two modes of operation: 1) run away (emulating a prey) and 2) follow (emulating a prey). This will be accomplished through the use of 4 different sensors working together to have a very specific behavior that will be controllable and interactive. The core of robot will be the ATmega128 micro-controller embedded in the popular Mavric-IIB board.

Most importantly, I want to create a robot that operates in any environment. The reason for this was the fact that I noticed that most robots that were created, or are been created, have to operate in a special environment created solely for the robot. Such environments include operating on top of a completely white floor with black lines around it (boundary, edge, etc), or have a background full of different of specific colors to color detection (and those colors were preset).

Frankly, Bi-Mode is a robot suffering from an identity crisis. Sometimes it is in a good mood, and other times he is just downright vicious – you just never know. Bi-Mode models nature's predator-prey behavior in today's synthetic ecosystem. It's a matter of "survival of the fittest." There are NO RULES, and there is NO COMPROMISE.

Executive Summary

Bi-Mode is autonomous robot built for the Summer 2005 EEL5666C IMDL class. The purpose of the robot was an attempt to emulate nature's animalistic inhabitants. When it comes to the phrase "survival of the fittest," Bi-Mode takes it literally. The robot exhibits the two primary behaviors of an animal, prey or predator. In essence, Bi-Mode is like two robots in one.

The popular Mavric-IIB is the core of the robot. It contains the popular Atmega128 micro controller, which is more than you possibly need for any robot. WinAVR and the PonyProg program were used to program the C/C++ code and to down the C program to the board.

Bi-Mode uses a pair of continuous servos for mobility. To sense the world around the robot, 4 different sensors were used. IR sensors are used to detect nearby objects, Bump sensors are used to detect collision with objects. Photo-reflectors are used to make sure the robot does not fall from an edge if there is a lower level as the robot is operating. A single Pyro-electric sensor is used to detect human in front of the robot.

As a prey, Bi-Mode will run away from any chasing object that is directly behind. At the same time, it will avoid obstacle collision with objects in the front. The movement will be similar to a prey. For instance, if a predator is chasing Bi-Mode, the robot will align itself to have the predator right behind him to provide maximum distance. If another predator or object is present in front and to the right, Bi-Mode will travel to the left in an attempt to avoid both predators or avoid getting corned by the chasing predator.

As a predator, Bi-Mode will only target preys with a detectable heat signature. With the use of the front IR sensors and the pyro-sensor, the robot will be able to track down its prey (humans and some animals). The tracking will be very accurate, as the pyro-sensor can detect its target about 3 meters away with a decent degree of certainty.

An LCD will be used to transmit behavioral feedback for anyone to see. For instance, a message will appear prompting the mode the robot is in: prey or predator.

The cool part of the robot is the fact that you can change the mode of behavior at will. All you need is to have the control remote that I integrated into the Bi-Mode's behavior routines and press the right button to cause Bi-Mode to have a sudden identity crisis.

Ultimately, the purpose of this report is to document all the steps taken into the creation of Bi-Mode with the aim to have it reproduced, if desired, by anyone that reads this report, even if it is by a beginner such as myself.

Introduction

If you or your pet ever wanted to run for your life, or just running after something, Bi-Mode is the solution. Bi-Mode has an interchangeable behavior between a prey and a predator, which you can control by pressing a button on a RF control. Immediately, you can watch Bi-Mode have an identity crisis right in front of your eyes. The question now is, are what kind of running will you be making?

As far as I can tell, Bi-Mode is the first controllable multi mode autonomous robot. It is like having two robots for the price of one. Talk about getting your money's worth.

Integrated System

Bi-Mode's brain is the Atmega128 micro-controller on a Mavric-IIB development board. Features of the board are 128K-program flash, 4K static RAM, 4K EEPROM, dual level shifted UARTs, RS485, I2C, up to 53 I/O pins, 16MHz clock (adjustable), and much more.

The robot will be equipped with an LCD display for feedback information and 2 continuous servos for locomotion. There will also be about 5 IR sensors (by Sharp: GP2D12) for object detection, 2 photoreflectors for "end of the world" sensing, 6 bump switches for collision detection, and a RF control remote. RF Remote Control

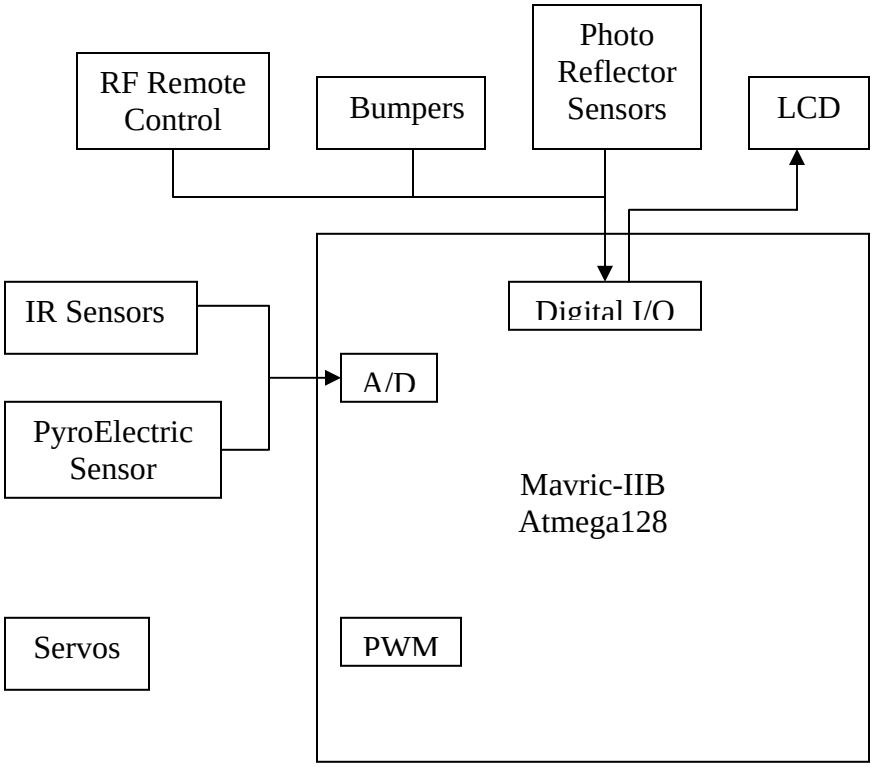


Figure 1: Hardware Organization

Figure 2: Mavric-IIB board

Figure 1 shows the general interface of the hardware found in Bi-Mode. What is not shown is a power board that supplies power to every peripheral. Figure 2 shows the actual board that will be controlling the robot's hardware and behaviors.

Mobile Platform

The platform will be specifically designed for the peripherals that will be used in the robot. The platform will be designed in AutoCAD and cut out in the T-Tech machine. The design will be circular in shape, about 1 foot in diameter and 1/8th inch in thickness. The choice for the shape was the fact that a circle provides best mobility in any environment, especially in those with a large amount of obstacles to avoid. Right underneath the circle is 4”x 4”x 2” box that houses all the hardware, from servos and LCD, to the boards and the sensors. All put in a nice tight package, which hides all the mess from the wires and hardware. To better illustrate, see the following pictures:

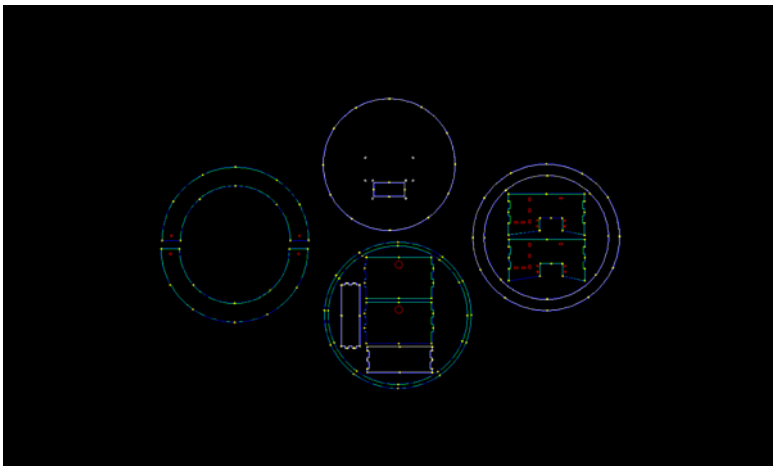


Figure 3: AutoCAD Platform Design

Figure 3 shows the AutoCAD platform design before it is cut out in the T-Tech Machine. Figure 4 shows the assembled platform obtained from the AutoCAD design.

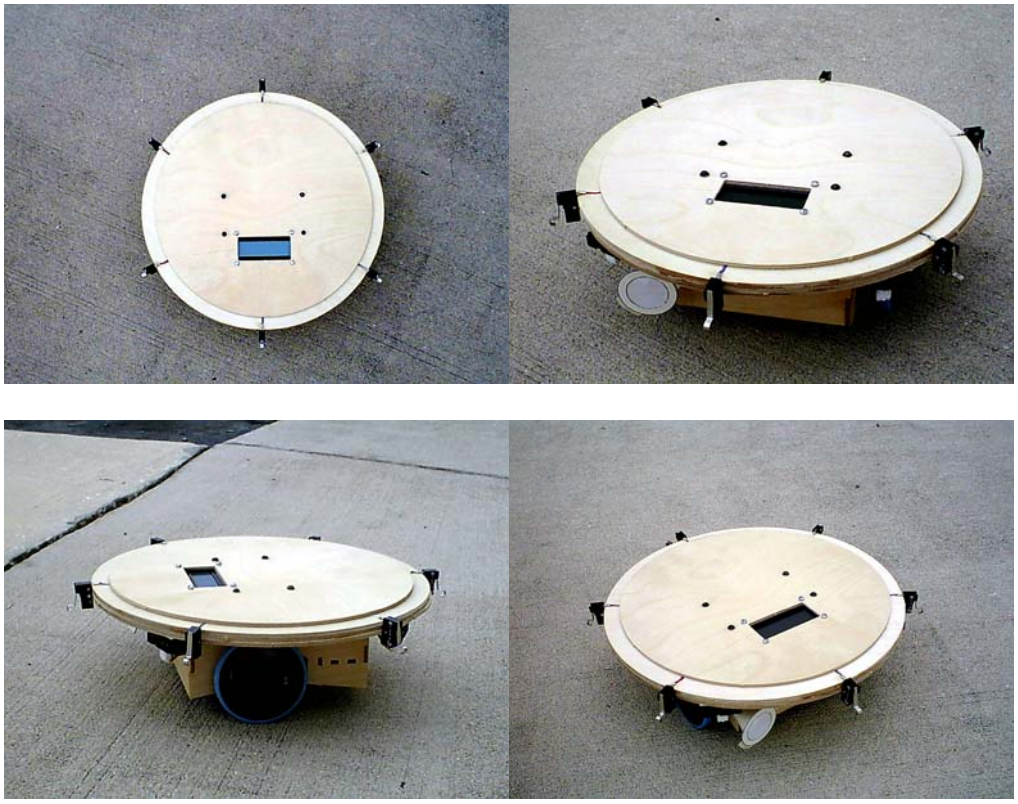


Figure 4: Bi-Mode's Assembled Platform

Figure 4 shows the platform already assembled. The LCD can be seen on the top of the robot, along with all six bump sensors. From a side view, you can notice some of the mounted IR sensors, the pyro sensor, and the location of the servos. For more on the sensors and their respective location, refer to the "sensor" section of this report.

Actuation

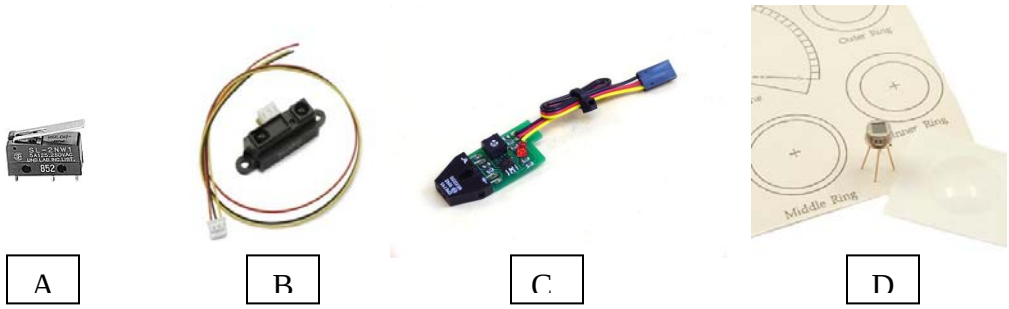
The robot will operate on a pair of STD TS-53 standard continuous rotation servos. To control the servos, a pulse width modulated signal is generated on the Mavric-IIB board and applied to the servos. The features: 42 oz.-in. of torque, 1.5 oz. Weight, 0.22sec/60 degree of transit time, 360 degree range of motion, a Futuba spline, and an

operation voltage of 4.8-6.0 volts. To provide motion, I added a pair of servo tires (Diameter: 2.63” Width: 0.35” Weight: 0.42 oz.

Figure 5: Servo and Tires/Wheels

Figure 5 displays the actual servos and tires used on Bi-Mode. There is no particular reason for the selection of this hardware. However, I recognize the fact that this is not the best choice. A pair a high speed motors would have been more adequate to the robot as it can emulate much better the speed of a prey or predator.

Sensors



A B C D

Figure 6: Sensors: A) Bump B) IfraRed C) PhotoReflector D) PyroElectric

Bump sensors

- SA STSP switch: For object collision detection
- Will be integrated to the system using the digital ports

IR sensors

- Sharp GP2D12 and Sharp GP2Y0A02YK: For object detection
- Will be integrated to the system using the analog ports

PhotoReflector sensors

- SLD-01: For “end of the world” behavior: to avoid falls from a platform
- Will be integrated to the system using the digital ports

PyroElectric sensor

- Eltec 442-3: To detect and track a heat signature/movement
- Will be integrated to the system using the analog ports

Bump Sensors

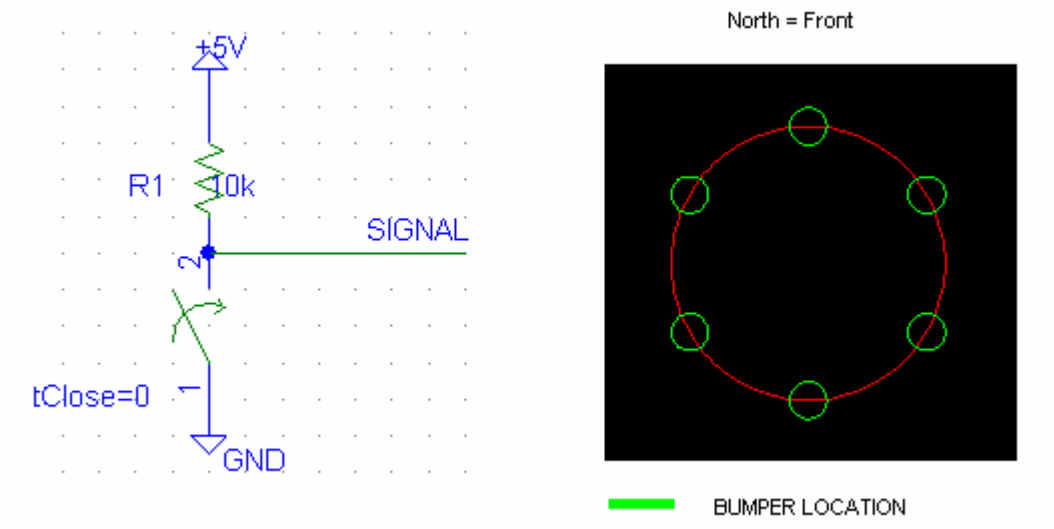


Figure 7: Bump schematics and location on the robot

The bump switches are wired as depicted in the schematics of Figure 7. The circuit will be a pull-up switch, which yields an active low signal. The signal will go a

digital I/O port in the Mavric-IIB board. A total of six bumpers will be positioned in a hex format around the robot, which allows for complete coverage of the perimeter.

To detect a collision, the software will detect if the signal is high (no collision) or low (a collision occurred because the switch is closed).

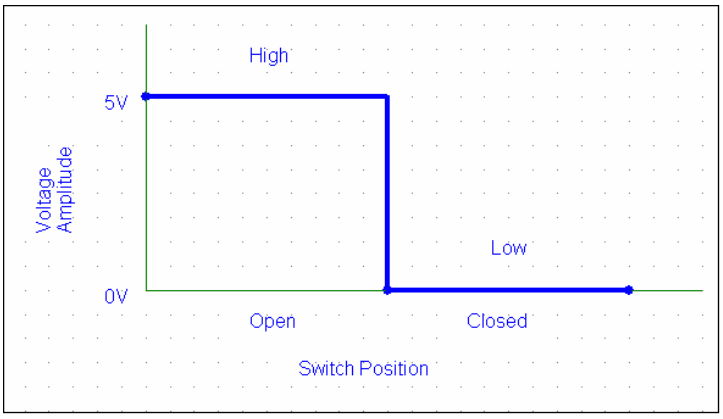


Figure 8: Switch Operational Data

IR Sensors

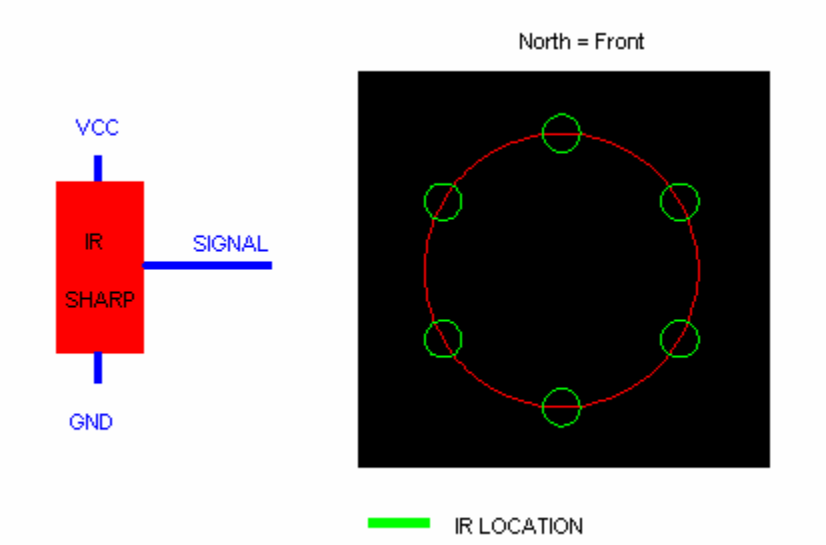


Figure 9: IR wiring schematics and IR location on platform

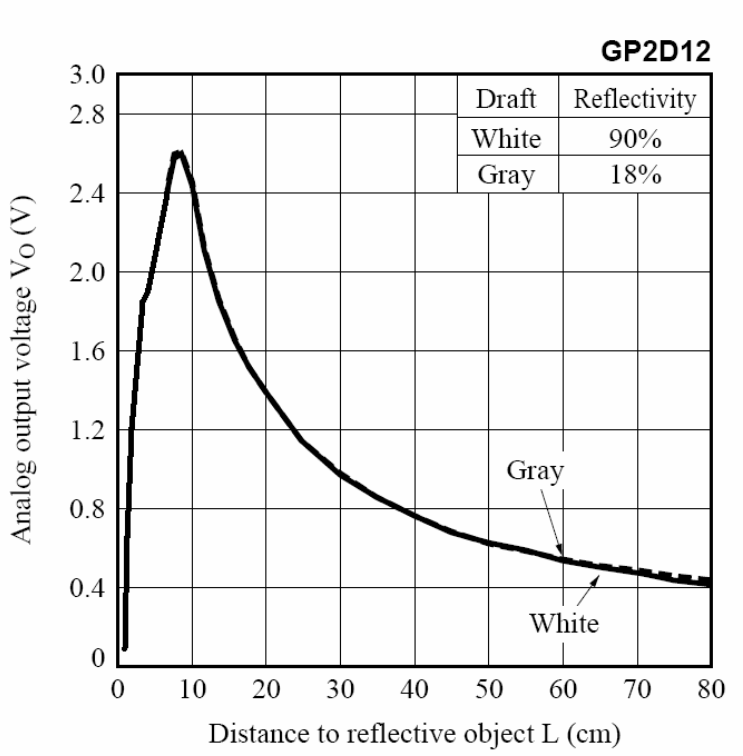


Figure 10: IR Theoretical Operation of Voltage vs. Distance

A total of five IR sensors will be located as depicted above. All of the IR will be GP2D12 with the exception of the very bottom sensor, a GP2Y0A02YK. The wiring is basic, as shown in the schematics of Figure 9. The signal will go to an analog I/O port in the Mavric-IIB board. The output voltage follows the general characteristics of the chart above in Figure 10. The easy way to integrate the distance is to be in the environment of operation and measure a desired distance of operation by taking the voltage at the output. The following formula gives the ADC value corresponding to the distance (voltage) desired: (Vsignal*1000*1024)/(Vreference*1000) = ADC. Vsignal comes from the IR and is put in an A/D port. The Vref is the 5V reference of the A/D I will be using. This is the technique I will be using to adjust to the environments lighting, rather than creating a calibration routine in the robot. You can also read the following graphs of data for the

GP2d12 that were take in different environments, and thus obtain the desired values.

Notice how Figure 11 and 12 are very similar to Figure 10.

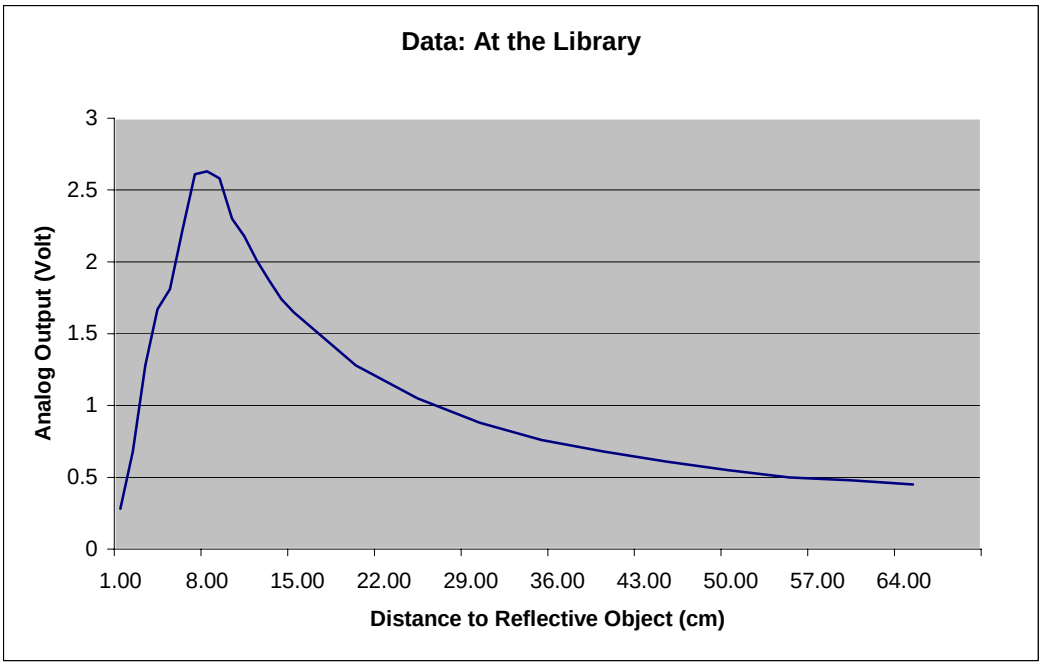


Figure 11: IR Data taken at Marston Library

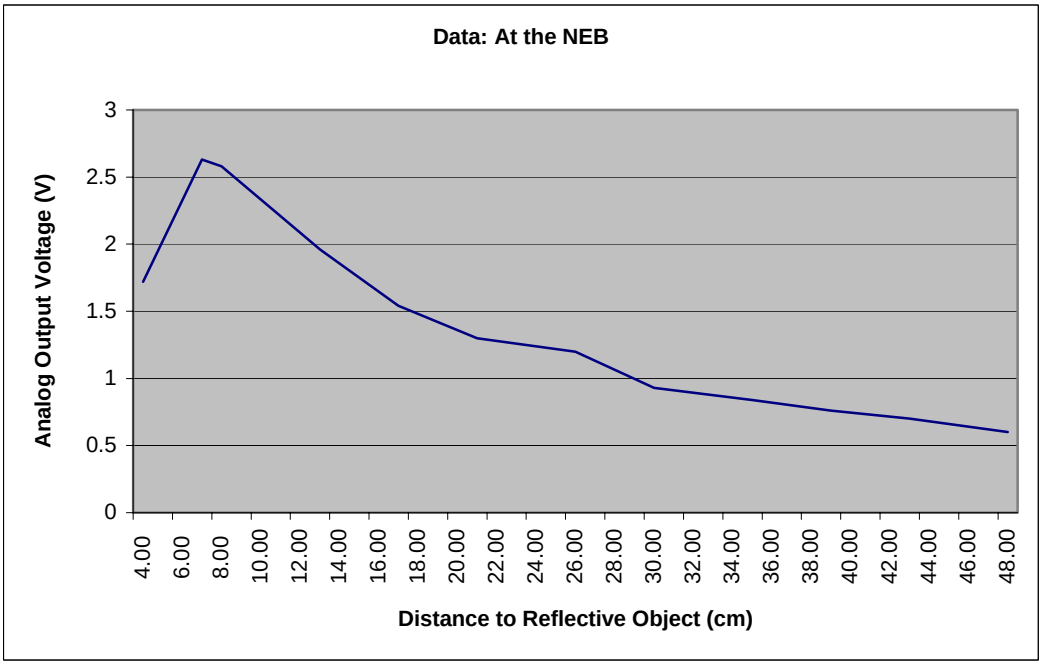


Figure 12: IR Data taken at NEB

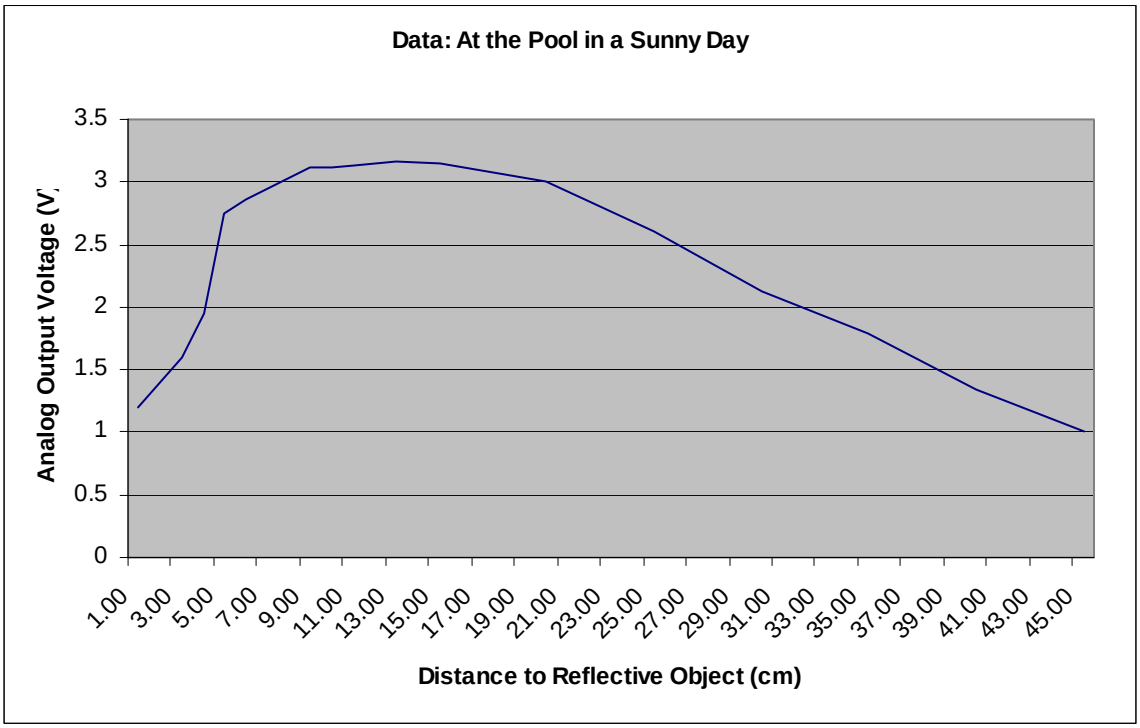
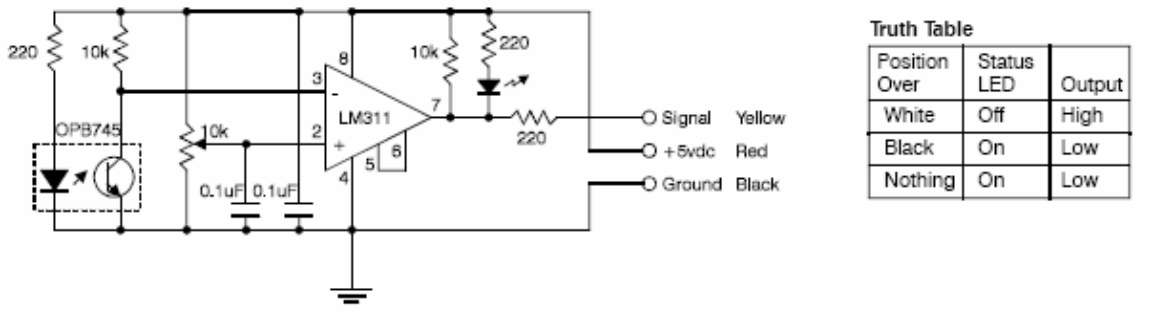


Figure 13: IR Data taken at a Swimming Pool

PhotoReflector



Truth Table

Position Over	Status LED	Output
White	Off	High
Black	On	Low
Nothing	On	Low

Figure 14: Photo's Circuitry and Operational Truth Table

Two photo-reflectors will be mounted in the front of the robot. The choice to use two photo-reflectors is because sometimes when the robot roams around, black spot or lines can trigger the sensor. Thus, having two will guarantee functionality, especially in the floor of NEB building (where the "Media Demo" will take place). I selected this

sensor because it is easy to debug, integrate, and flexibility. You can use this sensor to either detect an edge when the robot is on an elevated platform (table, second floor with stairs, etc.), or you can follow lines (with a pair or more) if you want to. However, under default settings, the sensor has to be very closed to the ground. Experimentation shows that the most ideal height is 0.75cm because it can detect the surface that is on regardless of questionable shades and decent luminescence of the environment. The only warning for this sensor is if the sensor is too close to the ground, the sensors malfunction by giving an erroneous output (does not detects the light surface – thinks it in on a dark spot or no surface). The signal will go a digital I/O port in the Mavric-IIB board. The signal will be a logic signal, which makes is ideal to integrate in our system, especially since it has an external LED that shows the status of the sensor (see Figure 11: Truth Table shows its general operation on a surface). It operates in a similar manner to a switch, but the wiring is simple (just ground and power the sensor, and take the signal to the board – Figure 11 shows the schematics that makes the sensor's circuitry and the external wires that come out, which are the ones I am using to interface the sensor).

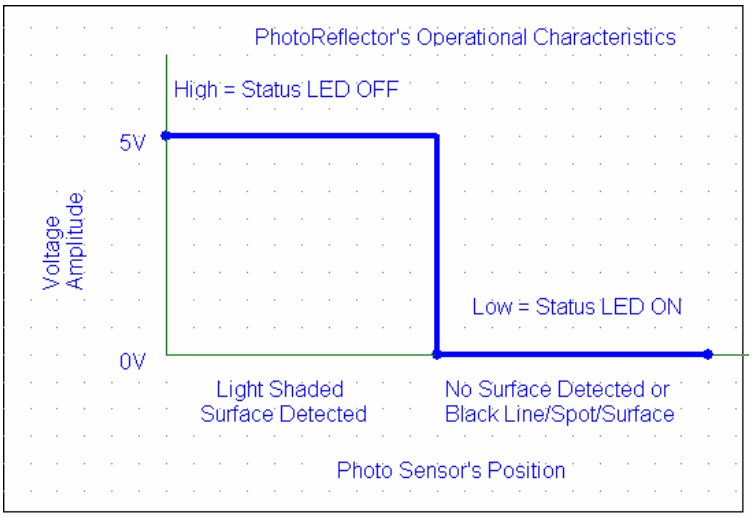


Figure 15: Photo-Reflector's Operational Characteristic Data

PyroElectric Sensor

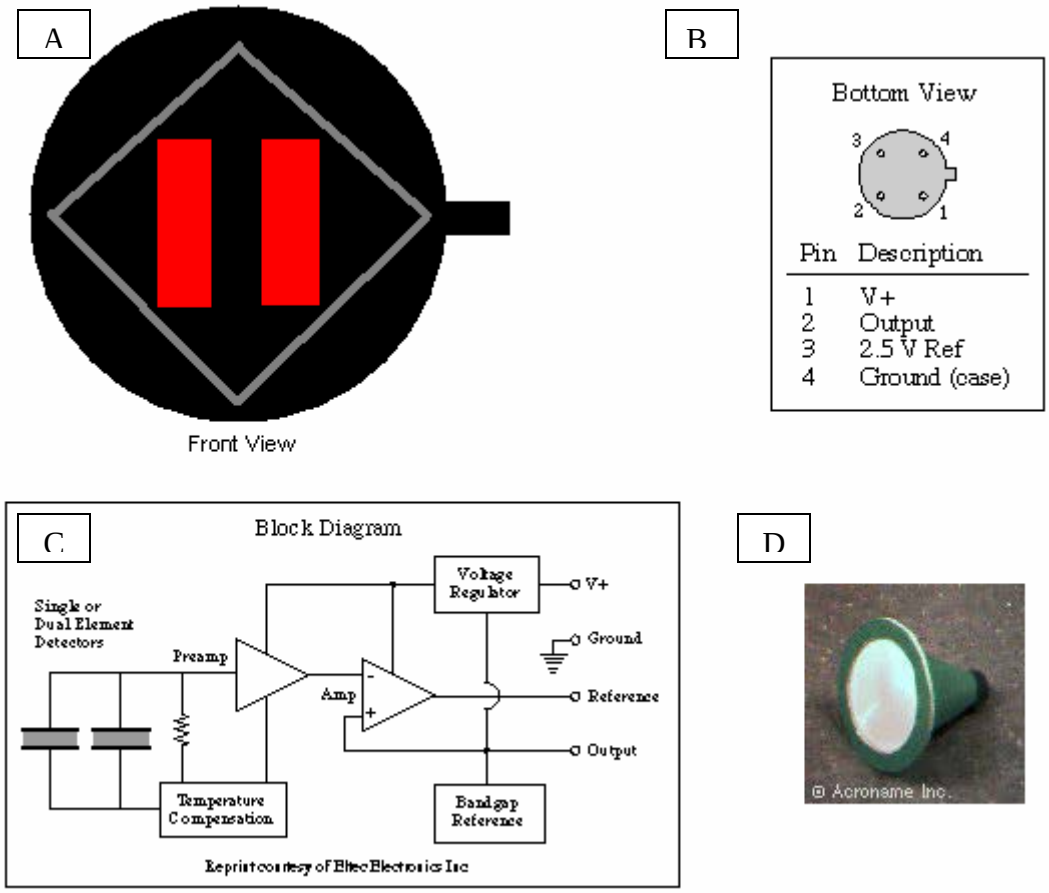


Figure 16: A) Front View w/o Cone B) Back View = Pin Out
C) Pyro's Circuitry D) Cone w/ Filtering Lens

The interface is simple, as seen on the schematics. From the front view picture, motion is detected when there is horizontal movement by an object with a heat signature. The sensor comes equipped with a cone to place a Fresnel lens on it, providing a 30º field of view (FOV). The lens filters several wavelengths and only allows wavelengths of 8-14 micrometers to pass into the sensor. The reason for this is that humans have an energy/heat radiation of about 10 micrometers. Thus allowing great human motion detection. The sensor has a steady state voltage of about 2.5V, meaning that the sensor will settle down to this voltage after a little while, regardless of the presence of a human.

When there is motion in the right direction (when facing the front of the sensor), the output goes up. When there is motion in the left direction, the output goes down. The degree of the change of the voltage seems to depend on the speed of motion and relative proximity to the detector. For instance, if a human runs across the sensor, the voltage will go practically high or low, depending on the direction of motion. Normal waking pace produces about 1.3V and 3.6V, depending on the direction of motion. In my opinion, the sensor is not a great piece of equipment. It seems simple, but due to its inaccuracy and logistic design, it is very cumbersome to track a person. However, it is flawless when it comes to the detection of the presence of a person when the individual comes in the FOV of the sensor, even when the person is 3 meters away. From Figure 15A, you can see two parallel rectangles, this are the actual sensors in the sensor. The way they work is if the right sensor senses the proximity of a person, the sensor “heats up” due to the thermal radiation emitted by the individual and thus increases the output voltage above the steady state value of about 2.5V. But when the left sensor detects the “heat source” moving near by, the sensor decrease the output voltage below the steady state value. The inaccuracy for human tracking comes when the “heat source” is somewhat in the middle or goes across the middle. This is the hard part to code because of the erratic behavior of the sensor under these circumstances. For instance, sometimes the sensor changes voltage (lower to higher, or higher to lower) depending on the direction of motion, or it just plainly gives a higher, or just lower voltage at the output. This behavior is hard to predict, and thus provides problems when it comes to tracking a person.

RF Remote Control

Bi-Mode's function is to provide the two behavior sides of the animal kingdom, prey and predator. To accomplish this task, the prey and predator were both separated into two different software algorithms. Each algorithm can be selected at the push of a single button from a RF remote control. The implications of this idea or concept, is the fact that you can literately create different behavior algorithms (or completely different robots) and put all your code in the robot and you just select which robot you want to be operational. My control remote is a 4 channel remote, in other words, I have 4 buttons. Thus, due to the way I have my code, I have the ability to have 5 different robots (a default algorithm that runs from startup, and 4 different algorithms/robots when I press the desired button/robot). In practice, you can have as many selectable robots as you have buttons in your control remote, plus the one default algorithm (if you want it to start with a robot at startup, this is optional). As far as I am concern, Bi-Mode is the first robot in the class history to have such capabilities. To deselect the current operational robot, all you have to do is press the same button you pressed to select it in the first place.

The way the remote works is by the push of one of its buttons (momentary relay), a 315Mhz signal is send out, as long as the button stays pressed. The way the receiver works is when the signal from the remote is received, the output goes high; otherwise the output is always low. The power supply to the receiver is a 5V input, while the remote has a 12V battery. The receiver has an antenna, and together with the remote signal strength, they are capable of working in a range of 200m. But such distance will never be required since you are usually using the remote nearby. Figure 17, below, shows the actual receiver and a similar remote that is in used with Bi-Mode.

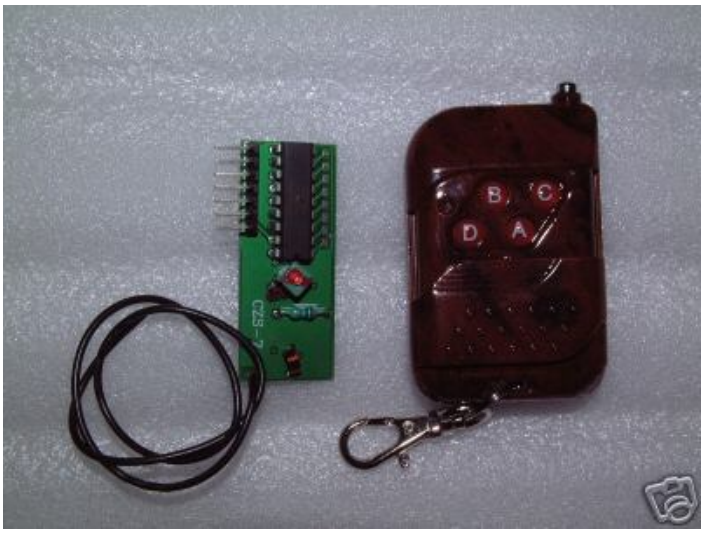


Figure 17: RF Remote Control used in Bi-Mode

Power Supply

Since the Mavric-IIB board can not supply sufficient power and current to all the sensors, actuators, LCD, and other hardware that were used to construct Bi-Mode, I created a power board for such purpose. The power board consists of a 7805-voltage regulator; it brings voltages from up to 30-volts down to a regulated 5-volt supply. My power supply to the power board is an 8 pack of AA batteries, rated at 1.2 volts each. My power board contains 5 different 7805 to have sufficient terminals and to minimize any power failures and hardware damage during operation of the robot. Below, in Figure 18, is the general schematic using a single 7805 chip.

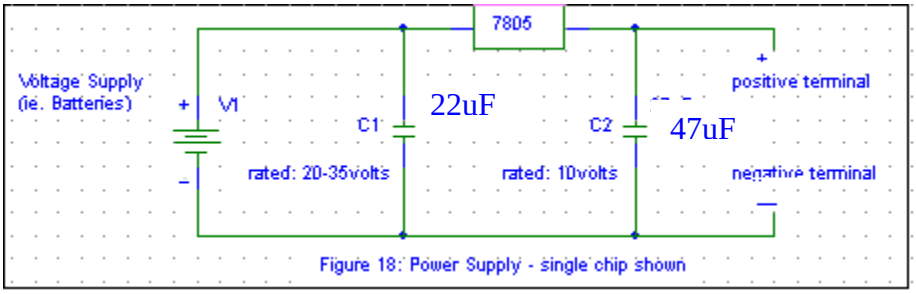


Figure 18: Power Supply - single chip shown

Behaviors

Mode 1: Prey: Run away from chasing predator

This is the default mode at startup. With the use of the IR sensors, the robot will run away from approaching predators, at any angle of approach. The robot will turn in the opposite direction and start running. While being chased, the robot will avoid obstacles along the way. Also, the robot has abilities to detect traps, like corners or dead ends, and the robot will also avoid these paths. Along the way, if the predator that is chasing it gets too closed, the robot will try to shake him off by turning left and right (much like a real prey, as seen on TV). At the same time, if the robot comes to an edge (i.e. Cliff, stairs, second level floor, etc.), the robot will go back and seek an alternative route. Also, at the same time it will check for its bump sensors. If any bump is activated, meaning the predator got too close to touch it, the robot will move in the opposite direction to have the maximum distance in between.

Mode 2: Predator: Heat signature tracking

This mode is selectable through the RF remote and a momentary switch (as a fail-safe). The robot will check for a heat signature present, through the pyro sensor, to track (to prey upon). If there is no reading on the pyro, Bi-Mode will depend on the IR sensors to locate an object. Once located, the robot will turn and face the object to check if it is a heat signature to track. Else, Bi-Mode will keep roaming around until it finds food/ a prey. If the front bump sensor ever hits an object that it was tracking, means that Bi-Mode was able to catch it's prey and eat it. This mode also shares the "Edge of the World" behavior. In other words, the robot will constantly check for an edge so it will not fall off while in operation.

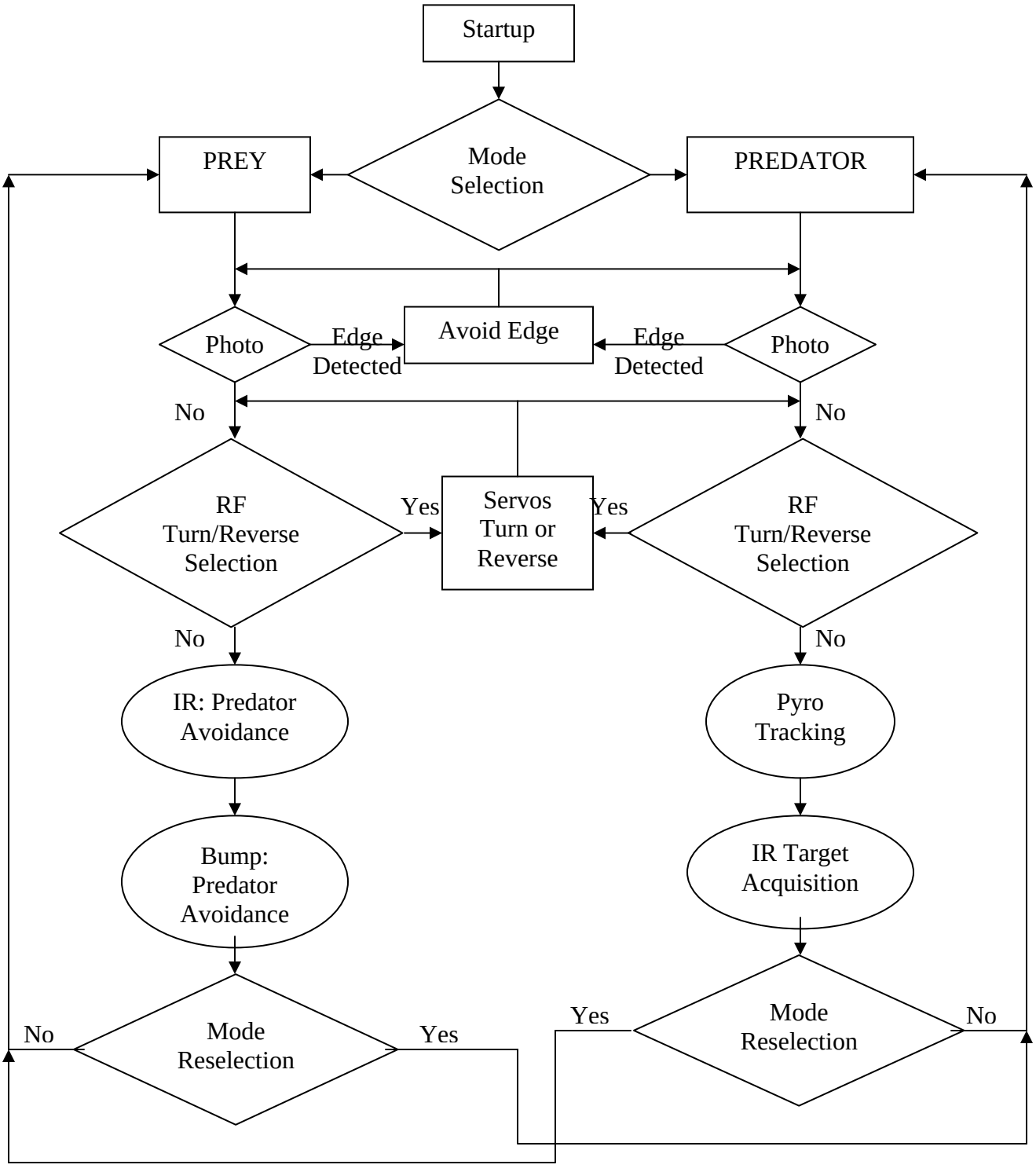


Figure 19: Basic Flowchart of Bi-Mode's Operation

Experimental Layout and Results

IR:

Scope: Testing Functionality

Specifications: None

Objectives: Obtain Data and/or Info on IR

Materials: Digital Multi-meter and Ruler

Data: Data reading were taken in different locations.

Library Dist (cm)	Library Voltage (V)	NEB Dist (cm)	NEB Volt (V)	Pool Dist (cm)	Pool Volt (V)
65	0.45	48.125	0.6	45	1.01
60	0.48	43.75	0.7	40	1.35
55	0.5	39.375	0.76	35	1.79
50	0.55	35	0.84	30	2.12
45	0.61	30.625	0.93	25	2.6
40	0.68	26.25	1.2	20	3
35	0.76	21.875	1.3	15	3.15
30	0.88	17.5	1.54	13	3.16
25	1.05	13.125	1.96	11	3.14
20	1.28	8.75	2.58	10	3.12
15	1.65	7.875	2.63	9	3.11
14	1.74	4.375	1.72	6	2.86
13	1.87			5	2.75
12	2.01			4	1.95
11	2.18			3	1.6
10	2.3			1	1.2

9	2.58
8	2.63
7	2.61
6	2.22
5	1.81
4	1.67
3	1.28
2	0.68
1	0.28

Figure 20: IR Data taken at Different Locations
See Figures 11 to 13 for respective graphs

Bumps:

Scope: Testing Functionality

Specifications: After wiring the bumps as in Figure 7's schematics

Objectives: Obtain Data and/or Info on Bumpers

Materials: Digital Multi-meter

Data: The voltage changed almost immediately after pressing the switch. It was noticed that the voltage changed from one logic level to another in about half a second.

Pyro:

Scope: Testing Functionality

Specifications: The Cone with the Lens was used

Objectives: Obtain Data and/or Info on Pyro

Materials: Digital Multi-meter

Data: After setting it up, I walked in front of the pyro, while at the same time taking continuous readings with the multi-meter. If you position the pyro as in Figure 16A, the voltage change from the steady state voltage of 2.5V to about 1.6V on average as I was moving on the left of the pyro. Also, the voltage level change to about 3.2V on average when there was movement on the right side of the pyro. Almost immediately, when the movement ceased but I still remained on the field-of-view of the pyro, the voltage level tried to go back to the steady state level.

Photo:

Scope: Testing Functionality

Specifications: None

Objectives: Obtain Data and/or Info on Bumpers

Materials: Digital Multi-meter and Ruler

Data: The voltage changed almost immediately in the presence of a shade change or if there was ground or not. Usually, the photo sensor was pretty good at not detecting obscure shades, as dark gray. You literally needed a solid black color to have a voltage change. It was noticed that the voltage changed from one logic level to another in about half a second. If there was a black color underneath or if there was no ground to detect, the voltage level will be low. However, if it was positioned over a light background, the voltage level was high. The photo sensor was positioned about 0.75cm above the ground, after experimental trial, for optimal

detection. The reason was that if it was above 1cm or below 0.25cm, the voltage level was low.

Conclusion

Brief Summary: Over all, the project was a success, all aspects of the project were accomplished and even more was done than were in my goals. The project was finished with a little time to spare, good to take care of last minute complications. I learned so much while in this class, basically everything that I did in this class was totally new to me.

Hopefully this report was easy to learn from and follow, as it was aimed to help any beginner interested in robotics.

Limitation of My Work: The servos are too slow. To illustrate, they run at about 1 foot per second or slower when at max forward speed. For a prey-predator robot, this is too slow. In fact, servos are too slow for any robot that moves around. I highly recommend that anyone creating a robot that moves, that this individual buy fast motors (which you can slow down in your software). Also, since I am using the photo sensors for “Edge of the World” and shade detection in any environment of operation (something that I aimed from the beginning because I noticed that most robots that were

created needed a special environment to operate – i.e. on top of a white floor.), I should have bought 2 more photo sensors to have a much better algorithm. This would also have enabled me to create a very good line following algorithm.

Technical caveats for students to follow:

Create a power board such as mine. The Mavric boards or any other board you buy will not be able to handle all the power demand required (especially from your actuators). My voltage regulator on my board burned out, and thus I had to create the power board (with 5 different voltage regulators- just to avoid similar incidents).

Future Work: I will get much fast motors (for any robot that moves around). Also, I will try to add two more photo sensors and rewrite my photo sensor algorithm to add additional conditions and features (like line following). I will also try to make it flashier. By that I mean, adding LED or anything else to present a more visual appearance. Also, for future robots or Bi-Mode, I will add a camera (either the CMUCAM2 (features: video, tracking, servo control, color processing) or the 10X wireless camera (features: live high resolution video feedback)). This should be a great audience pleaser at the very least.

Documentation/References/Credits/Links/Acknowledgement

Here is were all documents, references, credit to other people, and links are given.

Code Credits:

Analog to Digital Conversion Code
www.bdmicro.com/code/
My code took an excerption of their sample codes. Here you can also find code specific for the Mavric boards for almost any sensor (Sonar, IR, Servo, Speech, Compass, I2C)

Jeff Panos (creator of GIMP during Fall '04 semester)
www.mil.ufl.edu
Adaptation from Jeff: Servo excerption
Follow the link to previous semester reports. I took the timer/initialization exception from his servo code.

Lynette Miller (creator of ELIMEN during Summer '03 semester)
www.mil.ufl.edu
Adaptation from Lynette: LCD
Follow the link to previous semester reports. She gave credit to *Max Billingsley* as the primary author of the LCD code, for which I also adapted.

Picture Credits:

Figure 2: Marvric-IIB board picture
www.bdmicro.com
This is the vendor of the board

Figure 5: Servo and Tires' pictures
Figure 6A and 6D: Photo and Bump Sensors
Figure 11: Photo Sensor's schematics and Truth Table
www.lynxmotion.com

Figure 6B and 6D: IR and Pyro sensors
Figure 10: IR Theoretical Operational Data
Figure 16B, 16C, and 16D: Pyro's back view and pin diagram, schematics, lens
www.acroname.com/robotics/parts/c_Sensors.html

Figure 17: RF Transmitter and Receiver's picture
http://stores.ebay.com/ColdfusionX-Electronics
The vendor of my RF was ColdfusionX-Electronics. If you cannot find this link, just search for "RF channel remote" on ebay.com, which should give you a lot of hits for this part.

Document References:

Mavric-IIB Manual
http://bdmicro.com/images/mavric-iib.pdf

Atmega128 micro-controller Datasheet
http://atmel.com/dyn/resources/prod_documents/doc2467.pdf

Servo and Tires/Wheels Specifications
http://www.lynxmotion.com/Product.aspx?productID=360&CategoryID=38

Photo Reflector Specifications and Datasheet
http://www.lynxmotion.com/Product.aspx?productID=58&CategoryID=8
http://www.lynxmotion.com/images/data/sld-v1.pdf

Sharp IR Specification and Datasheet
http://www.acroname.com/robotics/parts/R48-IR12.html
http://www.acroname.com/robotics/parts/SharpGP2D12-15.pdf

PyroElectric Specs
http://www.acroname.com/robotics/parts/R1-442-3.html

16x2 Hitachi HD44780 LCD Specification and Datasheet
http://www.sparkfun.com/datasheets/LCD/GDM1602K.pdf
http://www.sparkfun.com/datasheets/LCD/HD44780.pdf

Software Reference:

PonyProg2000 Serial/Parallel Programmer
http://www.lancos.com/prog.html
This is the program that I used to transfer my program to the Mavric-IIB
I downloaded the Beta version, recommended by TAs.

Programmer Notepad and WinAVR and Library files
http://savannah.nongnu.org/projects/avr-libc/
http://www.pnotepad.org/index.html
http://sourceforge.net/projects/pnotepad/
http://winavr.sourceforge.net/
WinAVR contains the Programmer Notepad (programmer and compiler) with a library for the Atmega128. Need to run wish84.exe to make each MakeFile

My Personal WebPages:

http://plaza.ufl.edu/jpadilla/
http://bellsouthpwp.net/m/n/mnemonix/
http://home.bellsouth.net/p/PWP-mnemonix

Hardware:

Mavric-IIB (assemble and tested, with screw terminals) $139
www.bdmicro.com

Parallel Port Dongle Programmer for STK Port $16
http://www.sparkfun.com/shop/index.php?shop=1&cart=246338&cat=4&

TS-53 (42 oz. in.) Standard Continuous Rotation Servo $15each
http://www.lynxmotion.com/Product.aspx?productID=360&CategoryID=38

Servo Tire and Wheel - 2.63" x 0.35" (pair) $7.50
http://www.lynxmotion.com/Product.aspx?productID=361&CategoryID=75

Photo Sensor $15each
http://www.lynxmotion.com/Product.aspx?productID=58&CategoryID=8

IR GP2D12 $12each
http://www.acroname.com/robotics/parts/R48-IR12.html

Pyro Sensor $64
http://www.acroname.com/robotics/parts/R3-PYRO1.html

Bump Sensors $4each
Store: Electronic Plus (Gainesville, Florida)

7805 Voltage Regulators $1each
Store: Electronic Plus (Gainesville, Florida)

RF Remote (receiver and transmitter) $15
http://stores.ebay.com/ColdfusionX-Electronics

Miscellaneous Items $150
Like: paint, solder, wires, markers, Velcro, tools, batteries

Note: Above prices do not include taxes, shipment, extra items or parts, etc.

TOTAL COST (after going through all my receipts) $631

SPECIAL THANKS

I would like to especially thank both of my TAs, **William Dubel** and **Steven Pickles** for their time, guidance, input, knowledge, and help throughout the process of making, Bi-Mode. Also to my teachers for their inputs and ideas. Not to forget all my classmates for their help and input as well. As a disclaimer, I would like to give credit (just in case) to anyone else that in some form or another had some sort of input in the making of Bi-Mode

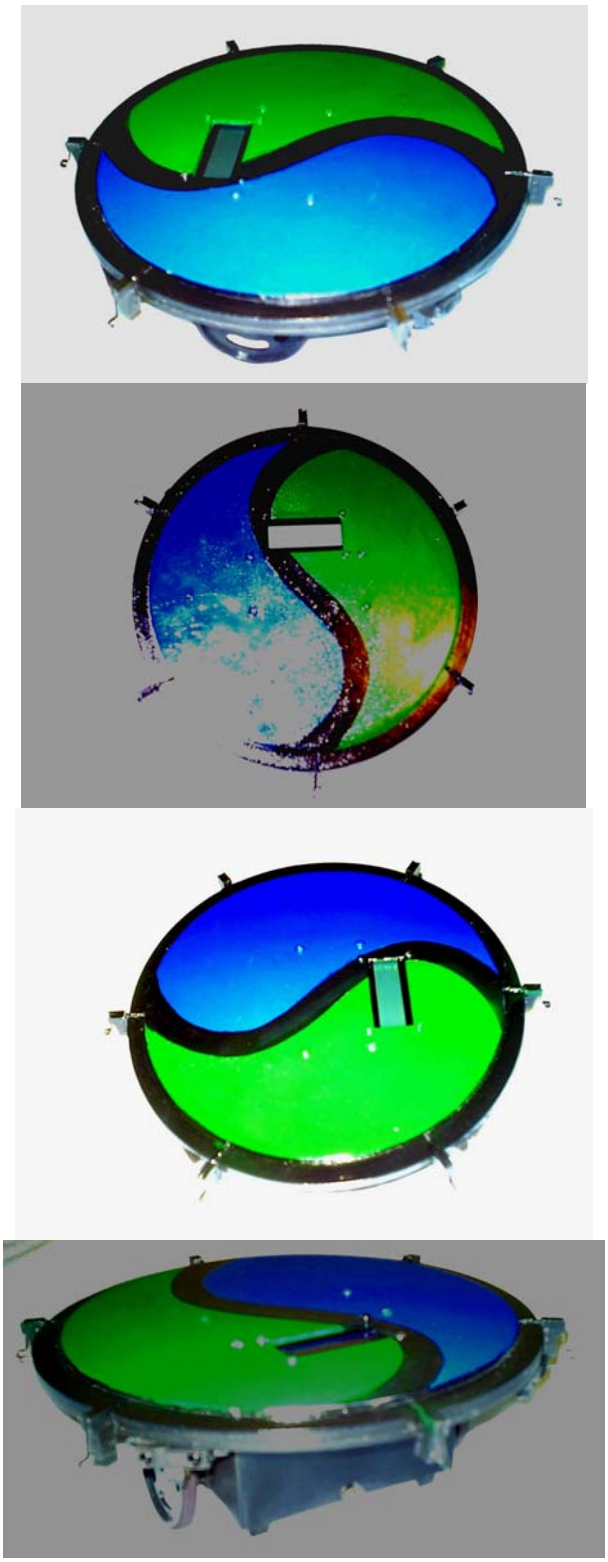


Figure 21: Bi-Mode

Appendix

This section contains the code for Bi-Mode

SEE FOLLOWING SECTION (for credits) ON PAGE 27:
Documentation/References/Credits/Links/Acknowledgement

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
LCD.c**************************************************************************

/*************************************************
* Desciption: Code to initialize and use the 16x2 Hitachi interface LCD display in 4bit mode
*
* my configuration(Bi-Mode)
* LCD_PORT0 - DB4
* LCD_PORT1 - DB5
* LCD_PORT2 - DB6
* LCD_PORT3 - DB7
* LCD_PORT4 - RS = 0x10
* LCD_PORT5 - r/w = 0x20
* LCD_PORT6 - EN = 0x40
*
* LCD_PORT:        7       6       5       4       3       2       1       0
* LCD_Wire:  -     EN      RW      RS      DB7     DB6     DB5     DB4
*
* LCD obtained from junun.org
*
*R/W: you can ground it if you want to save a pin on the port. in the code it is not used = GND
*
* RS: Register Select
* 0 - Command Register
* 1 - Data Register
*************************************************/
//the includes
#include "lcd.h"
#include <inttypes.h>
#include <avr/io.h>

void LCD_setDDR(void)
{
LCD_DDR = 0xff;
}

void LCD_init(void)
```

```
{
//initialize the DDR
LCD_setDDR();
LCD_delay();
//initilize Command Register
LCD_sendCommand(0x33); //enable 4-bit mode
LCD_sendCommand(0x32);
LCD_sendCommand(0x2c); //enable 2-line mode
LCD_sendCommand(0x0f); //display, cursor, blink
LCD_sendCommand(0x01); //clear home
}

void LCD_delay(void)
{
uint16_t time1;
//for(time1 = 0; time1 < 2000; time1++);
for(time1 = 0; time1 < 65000; time1++);
        for(time1 = 0; time1 < 65000; time1++);
}

void LCD_delayLong(void)
{
uint16_t i;
uint16_t k;
uint16_t var1 = 0;
for (i = 0; i < 30000; i++)
{
        for (k = 0; k < 30000; k++)
{
var1 = 0;
}
}
}

void LCD_delayShort(void)
{
uint16_t i;
uint16_t k;
uint16_t var1 = 0;
for (i = 0; i < 15000; i++)
{
        for (k = 0; k < 15000; k++)
{
var1 = 0;
}
}
```

```
}

void LCD_sendCommand(uint8_t val)
{
uint8_t temp = val;                                    // first do the 4bit in DB and then do
the (-,E,RW,RS) section
temp &= 0x0f;                                          // &= is equivalent to logic AND
val >>= 4;                          // >>= is equivalent to logic right shift value by 4 spaces
LCD_PORT = val;
LCD_delay();
LCD_PORT |= ENABLE;                                          // |= is equivalent to logic OR
LCD_PORT &= ~ENABLE;           // ~ is equivalent to logic NOT (aka. complement)
LCD_delay();
LCD_PORT = temp;                                       // the other half of the bits
LCD_delay();
LCD_PORT |= ENABLE;
LCD_PORT &= ~ENABLE;
LCD_delay();
}

//&& =logic AND
// || = or, as in checking if either or condition is true

void LCD_sendString(char *s)
{
while (*s) LCD_sendByte(*s++);      // ++ is equivalent to increment
}

void LCD_sendByte(uint8_t val)
{
uint8_t temp = val;                                    // saving it first
val >>= 4;                                                   // taking what is in DBs
val |= RS;                                                   // set data mode/
LCD_PORT = val;
LCD_delay();
LCD_PORT |= ENABLE;
LCD_PORT &= ~ENABLE;
temp &= 0x0f;
temp |= RS;                                            // set data mode
LCD_PORT = temp;
LCD_delay();
LCD_PORT |= ENABLE;
LCD_PORT &= ~ENABLE;
LCD_delay();
}
```

```
void LCD_clearScreen(void)
{
LCD_delay();
LCD_sendCommand(0x01);
LCD_delay();
}

void LCD_home(void)//brings blinker 1space to the left = on top of the the previous char
{
LCD_sendCommand(0x10);
}


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
LCD.h**************************************************************************

/***********************************************
* Title: LCD.h
* Desciption: Code to initialize and use the 16x2 Hitachi interface LCD display in 4bit
mode
*
* LCD_PORT0 - DB4
* LCD_PORT1 - DB5
* LCD_PORT2 - DB6
* LCD_PORT3 - DB7
* LCD_PORT4 - RS = 0x10
* LCD_PORT5 - r/w = 0x20
* LCD_PORT6 - EN = 0x40
*
* LCD_PORT:        7      6      5      4      3      2      1      0
* LCD_Wire:  -     EN     RW     RS     DB7    DB6    DB5    DB4
*
* LCD obtained from junun.org
*
*R/W: you can ground it if you want to save a pin on the port. in the code it is not used =
GND
*
* RS: Register Select
* 0 - Command Register
* 1 - Data Register
***********************************************/

/*.................Includes.....................*/
#include <inttypes.h>
#include <avr/io.h>
/*.................end of Includes.................*/
```

```
/*.................Constants.....................*/
#define LCD_PORT PORTA
#define LCD_DDR DDRA
#define RS 0x10 // RS Signal "0001 0000"
#define RW 0x20 // RW Signal
#define ENABLE 0x40 // ENABLE Signal
/*.................end of Constants.................*/
//Method Signatures
void LCD_setDDR(void);
void LCD_init(void);
void LCD_delay(void);
void LCD_delayLong(void);
void LCD_sendString(char *s);
void LCD_sendByte(uint8_t val);
void LCD_sendCommand(uint8_t val);
void LCD_home(void);
void LCD_clearScreen(void);
int main(void);


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
servo.c**************************************************************************

#include <stdio.h>
#include <avr/io.h>


/***************************** Move Forward ******************************/
void move_forward(void)
{
//OCR1A=Right Servo -when looking from behind the robot
//OCR1B=Left Servo
OCR1A=1000;
// right servo -PortB5 -> forward
OCR1B=2620;
// left _PortB6-> forward
}
/*************************************************************************/
/************************** Slowly Move Forward **************************/

void slowly_move_forward(void)
{
OCR1A=1440;
OCR1B=2180;
}
/********************** Back Up ********************************/
```

```
void back_up(void)
{
OCR1A=1670;
OCR1B=1800;
}
/******************** Left Turn ***********************************/
void turn_left(void)
{
OCR1A=1200; // right -> forward
OCR1B=1500; // left -> backward
}
/*****************************************************************/
/********************** Slowly Turn Left ***************************/
void slowly_turn_left(void)
{
OCR1A=1440;
OCR1B=1850;
}
/******************** Right Turn ***********************************/
void turn_right(void)
{
OCR1A=1800; //
OCR1B=2400; //
}
/******************** Slow Right Turn ******************************/
void slowly_turn_right(void)
{
OCR1A=1550;
OCR1B=2180;
}
/*****************************************************************/
/********************** Stop Servos ********************/
void servo_stop(void)
{
OCR1A=0;
OCR1B=0;
}
/*************** Servo / Timer 1 A and B Initialization ************************/
//void timer1_ab_init(void)
void servo_init(void)
{
DDRB=0xE0;
PORTB=0x00;
TCCR1A=0xA8; // Gets timer channels A,B,C ready for use
TCCR1B=0x12;
TCCR1C=0x02;
```

```
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x4A;          //4AAA=19114
ICR1L=0xAA;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
//OCR1CH=0x00;
//OCR1CL=0x00;
}
/*********************************************************************/
/*************** Servo Delay ***********************************/
void servo_delay(uint16_t ms)
//every 10k = 1sec of servo delay -run or off////* UPTO 6sec delay *//////
{
for(int k = 0; k < ms; k++)
        {
                for(int h = 0; h< 20000; h++);
//
        }
}
//turn_right();servo_delay(11500);turn_left();servo_delay(10000); =>  160 degree turn
//turn_right();servo_delay(07000);turn_left();servo_delay(05000); ==> 45  degree turn
//turn_right();servo_delay(05100);turn_left();servo_delay(03000); ==> 30  degree turn
//turn_right();servo_delay(06500);servo_delay(06500);turn_left();servo_delay(05500);ser
vo_delay(05500); ==> 90 degree trun // do 2x to get 180 degree rotation
//turn_right();servo_delay(13000);turn_left();servo_delay(12000); ==> 180 degree turn
//
/*************************************************************/


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
bump.c*************************************************************************

/***********************************************
* Title: Bump.c
* Programmer: Anne Harmeson
* Date: 3/30/2004
* Version: 1
*
* Description:
* Code to initialize and use bump switches on Port C.
***********************************************/
#include "bump.h"
```

```
//#include "lcd.c"        //commented out for main b/c you do code 2x (main and here)
//#include "servo.c"


//XXXXXXXXXXXXXXXXXXXXXXX/////////

void bump_init(void)
{
bump_setDDR();
//BUMP_PORT = 0x00; //disable pullup resistor
}
void bump_setDDR(void)
{
BUMP_DDR = 0x00; //set as inputs
}

void bumpDB(void) //debounce?
{
uint16_t time2;
for(time2 = 0; time2 < 2000; time2++);
}


void bump_check4Collision(void)
{
if (PINC == 0x3E)
{
LCD_clearScreen();
LCD_sendString("Front = N");
servo_stop();                                    //immediately stop upon impact = avoid
backward current from reseting the board
servo_delay(05000);  //stop for 2.5sec
back_up();
servo_delay(05000);   //back up for 1sec
servo_stop();
servo_delay(05000);
turn_left();
servo_delay(12000);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
else if (PINC == 0x3D)
{
LCD_clearScreen();
```

```
LCD_sendString("NW");
servo_stop();
servo_delay(05000);
turn_right();
servo_delay(05100);   //30 degree right turn
servo_delay(05100);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
else if (PINC == 0x1F)          //it might just be an "ELSE"
{
LCD_clearScreen();
LCD_sendString("NE");
servo_stop();
servo_delay(05000);   //stop for 2.5sec
turn_left();
servo_delay(03000);   //for 0.5sec
servo_delay(03000);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
else if (PINC == 0x37)
{
LCD_clearScreen();
LCD_sendString("back = S");
servo_stop();
servo_delay(05000);
turn_right();
servo_delay(05100);   //trying to shake up off
servo_stop();
servo_delay(05000);
turn_left();
servo_delay(03000);
LCD_clearScreen();
}
else if (PINC == 0x3B)
{
LCD_clearScreen();
LCD_sendString("SW");
turn_right();
servo_delay(05100);   //30degree right turn
servo_stop();
```

```
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
else if (PINC == 0x2F)
{
LCD_clearScreen();
LCD_sendString("SE");
turn_left();
servo_delay(03000);   //for 0.5sec
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}


}
//if no collision, then it just terminates and program keeps going into something else if
you have more written after this behavior


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ADC.c**************************************************************************

#include <avr/io.h>
#include <stdio.h>
/*
* adc_init() - initialize A/D converter
*
* Initialize A/D converter to free running, start conversion, use
* internal 5.0V reference, pre-scale ADC clock to 125 kHz (assuming
* 16 MHz MCU clock)
*/
void adc_init(void)
{
/* configure ADC port (PORTF) as input */
DDRF = 0x00;
//PORTF = 0x00;
ADMUX = _BV(REFS0);
ADCSR = _BV(ADEN)|_BV(ADSC)|_BV(ADFR) |
_BV(ADPS2)|_BV(ADPS1)|_BV(ADPS0);
}
//aden=a/d enable//adsc=a/d start conversion in single conversion//adfr=free running conv
mode=>constant sample/update//adps=division factors=>128 B/C XTAL freq and int clk
to adc
```

```
//admux == refs1 refs0 adlar mux4 mux3 mux2 mux1 mux0   ////  refs1 refs0 ==
00=>aref, int Vref turned off; 01=>avcc w/external cap at aref; 10=>reserved; 11=>int
2.56V ref w/ external cap of aref

/*
* adc_chsel() - A/D Channel Select
*
* Select the specified A/D channel for the next conversion
*/
void adc_chsel(uint8_t channel)
{
/* select channel */
ADMUX = (ADMUX & 0xe0) | (channel & 0x07);//ADMUX = (ADMUX & 0xe0) |
(channel & 0x07);
}


/*
* adc_wait() - A/D Wait for conversion
*
* Wait for conversion complete.
*/
void adc_wait(void)
{
/* wait for last conversion to complete */
while ((ADCSR & _BV(ADIF)) == 0)
;
}

/*
* adc_start() - A/D start conversion
*
* Start an A/D conversion on the selected channel
*/
void adc_start(void)
{
/* clear conversion, start another conversion */
ADCSR |= _BV(ADIF);
}

/*
* adc_read() - A/D Converter - read channel
*
* Read the currently selected A/D Converter channel.
*/
uint16_t adc_read(void)
```

```
{
return ADC;
}

/*
* adc_readn() - A/D Converter, read multiple times and average
*
* Read the specified A/D channel 'n' times and return the average of
* the samples
*/
uint16_t adc_readn(uint8_t channel, uint8_t n)
{

uint16_t t;
uint8_t i;
adc_chsel(channel);
adc_start();
adc_wait();
adc_start();
/* sample selected channel n times, take the average */
t = 0;
for (i=0; i<n; i++)
{
        adc_wait();
        t += adc_read();
        adc_start();
}
/* return the average of n samples */
return (t/n);
}


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
prey.c**************************************************************************

//PREY == MAIN      LCD SERVO IR BUMP == Collision Avoidance

#include "LCD.c"
#include "servo.c"
#include "ADC.c"
#include "bump.c"
#include "photo.c"
#include <stdio.h>
#include <inttypes.h>
#include <avr/io.h>
```

```
void random_turn(void)
{

}

void ir_check4Traps(void)
{

}

void prey(void)
{
LCD_sendString("....... 123451235123451234512345123456");            //16char on 1st line +
25spaces to go to 2nd line + 16char on 2nd line = space to write on LCD

                        //different than stated b/c i cover part of the LCD with my platform

//ADC values: [(Vin=ie. 2V=2000)*1024]/(Vref=5V=5000) = 410            //256<400
thus use 16bit denomination or something that works
//62=0.3V 102=0.5V 112=0.55V 123=0.6V 143=0.7V 163=0.8V 184=0.9V 205=1V
266=1.3V 307=1.5V 328=1.6V 348=1.7V 369=1.8V 410(bin)=2V 430=2.1V 451=2.2V
471=2.3V 512=2.5V 553=2.7V 573=2.8V 594=2.9V 614=3V 655=3.2V 717=3.5V
819=4V 922=4.5V 1024=5V

bump_check4Collision();                    //checking BUMPERS

uint16_t threshold = 123;           //about 1feet 6in
uint16_t threshold2 =264;           //about 9in
uint16_t threshold3 =348;           //about 6in

uint16_t IR_NE;                                 //initializing size of IR values
uint16_t IR_NW;
uint16_t IR_SE;
uint16_t IR_SW;
uint16_t IR_South;
uint16_t IR_North;

IR_NW = adc_readn(0,10);            //test for IR_left
IR_NE = adc_readn(1, 10);           //sample channel 0 for a total of 10 times  //Test for
IR-right. If IR_Right >=threshold, avoid obstacle

IR_SW = adc_readn(2,10);
IR_SE = adc_readn(3,10);

IR_North = adc_readn(4,10);
IR_South = adc_readn(5,10);
```

```
if((IR_NE>=threshold2) && (IR_NW>=threshold2))  //checking for traps
{
LCD_clearScreen();
LCD_sendString("Front Obstacle NW NE");
servo_stop();                                   //immediately stop = avoid backward current
from reseting the board
servo_delay(05000);                             //stop for .5sec
back_up();
servo_delay(05000);                             //back up for .5sec
servo_stop();
servo_delay(05000);
turn_left();
servo_delay(12000);                             //180degree left turn
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
else if((IR_SE>=threshold2) && (IR_SW>=threshold2))     //checking for traps
{
LCD_clearScreen();
LCD_sendString("South Obstacle SW SE");
move_forward();
servo_delay(10000);
LCD_clearScreen();
}

bump_check4Collision();                         //checking BUMPERS
RF_check4Remote();                              //checking RF Remote

//ok, then single IR it is
if (IR_North>=threshold2)               //threshold2 because want to use IR NE and NW to
curve away first, if possible
{
LCD_clearScreen();
LCD_sendString("IR North Obstacle");
servo_stop();                                   //immediately stop upon impact = avoid
backward current from reseting the board
servo_delay(05000);                             //stop for 0.5sec
back_up();
servo_delay(05000);                             //back up for 1sec
servo_stop();
servo_delay(05000);
turn_left();
servo_delay(12000);                             //180degree left turn
```

```
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
else if (IR_NE>=threshold)
{
LCD_clearScreen();
LCD_sendString("IR Right Obstacle");
servo_stop();
servo_delay(05000);                    //stop for 2.5sec
turn_left();
servo_delay(03000);                    //30degree
servo_delay(03000);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
else if (IR_NW>=threshold)
{
LCD_clearScreen();
LCD_sendString("IR Left Obstacle");
servo_stop();
servo_delay(05000);                    //stop for 2.5sec
turn_right();
servo_delay(05100);                    //for 0.5sec
servo_delay(05100);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}

bump_check4Collision();                //checking BUMPERS
RF_check4Remote();                     //checking RF Remote

if (IR_SE>=threshold)
{
LCD_clearScreen();
LCD_sendString("IR SE Obstacle");
turn_left();
servo_delay(05000);                            //45degree left turn
servo_stop();
servo_delay(05000);
move_forward();
```

```
LCD_clearScreen();
}
else if (IR_SW>=threshold)
{
LCD_clearScreen();
LCD_sendString("IR SW Obstacle");
turn_right();
servo_delay(07000);                                    //for 0.5sec
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
else if (IR_South>=threshold2)         //wide beam IR (GP2Y0A21) on the back
{
LCD_clearScreen();
LCD_sendString("IR South Obstacle");
servo_stop();
turn_right();
servo_delay(05100);                              //trying to shake you off since you are too
close
servo_stop();
move_forward();
servo_delay(05000);
servo_stop();
turn_left();
servo_delay(03000);
servo_stop();
move_forward();
LCD_clearScreen();
}
else                                                          //no bump or IR, move
forward => refresh while(1) afterwards
{
move_forward();
LCD_sendString("RUN 4Your Life");//16char on 1st line + 26spaces to go to 2nd line +
16char on 2nd line = space to write on LCD
servo_delay(03000);                              //just so you can see the display
LCD_clearScreen();
}


}//end of prey();


//turn_right();servo_delay(11500);turn_left();servo_delay(10000); =>  160 degree turn
```

```
//turn_right();servo_delay(07000);turn_left();servo_delay(05000); ==> 45  degree turn
//turn_right();servo_delay(05100);turn_left();servo_delay(03000); ==> 30  degree turn
//turn_right();servo_delay(06500);servo_delay(06500);turn_left();servo_delay(05500);ser
vo_delay(05500); ==> 90 degree trun // do 2x to get 180 degree rotation
//turn_right();servo_delay(13000);turn_left();servo_delay(12000); ==> 180 degree turn



++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
predator.c**********************************************************************


//#include "LCD.c"
//#include "servo.c"
//#include "ADC.c"

void pyro_check4Target(void)
{
//ADC values: [(Vin=ie. 2V=2000)*1024]/(Vref=5V=5000) = 410            //256<400
thus use 16bit denomination or something that works
//62=0.3V 72=0.35V 82=0.4V 92=0.45V 102=0.5V 112=0.55V 123=0.6V 143=0.7V
163=0.8V 184=0.9V 205=1V 266=1.3V 307=1.5V 328=1.6V 348=1.7V 369=1.8V
410(bin)=2V 430=2.1V 451=2.2V 471=2.3V 512=2.5V 553=2.7V 573=2.8V 594=2.9V
614=3V 655=3.2V 717=3.5V 819=4V 922=4.5V 1003=4.9V 1024=5V

uint16_t threshold_high = 614;  //= 594;
uint16_t threshold_low = 370;          //= 369;
uint16_t pyro;
pyro=adc_readn(7, 50);                          //sample channel 0 for a total of 10 times
///analog(0-4=IRs)
if(threshold_high<=pyro && 999>=pyro)
{
LCD_clearScreen();
LCD_sendString("HIGH <<=="); //to the right of the pyro when facing it
servo_stop();
servo_delay(05000);
turn_left();//when facing pyro, if you on right = high.  THUS: robot should go left to
center itself to the target
servo_delay(03000);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
pyro=adc_readn(7, 100);
if (pyro<=threshold_low && 150<=pyro)//else if (IR_right>=threshold)
```

```
{
LCD_clearScreen();
LCD_sendString("LOW ==>>"); //to the left of the pyro when facing it// ==>>shows
where robot should go to center itself
servo_stop();
servo_delay(05000);//stop for 2sec
turn_right();    //when facing pyro, you on left = low. THUS: robot should go right to
center itself to the target
servo_delay(05100);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
}//pyro_check4Target()



void predator(void)
{
//ADC values: [(Vin=ie. 2V=2000)*1024]/(Vref=5V=5000) = 410            //256<400
thus use 16bit denomination or something that works
//62=0.3V 72=0.35V 82=0.4V 92=0.45V 102=0.5V 112=0.55V 123=0.6V 143=0.7V
163=0.8V 184=0.9V 205=1V 266=1.3V 307=1.5V 328=1.6V 348=1.7V 369=1.8V
410(bin)=2V 430=2.1V 451=2.2V 471=2.3V 512=2.5V 553=2.7V 573=2.8V 594=2.9V
614=3V 655=3.2V 717=3.5V 819=4V 922=4.5V 1024=5V

//LCD_clearScreen();
LCD_sendString(",,,,   12345123451234512345123451234");//16char on 1st line + 25spaces to
go to 2nd line + 16char on 2nd line = space to write on LCD

//different than stated b/c i cover part of the LCD with my platform

uint16_t threshold = 102;              //2feet
uint16_t threshold2 = 123;             //1feet 6in
uint16_t threshold3 =200;              //about 1ft
uint16_t threshold4 =264;              //about 9in
uint16_t IR_right;
uint16_t IR_left;
uint16_t IR_North;
uint16_t IR_SE;
uint16_t IR_SW;
uint16_t IR_South;

uint16_t threshold_high;// = 594;
uint16_t threshold_low;// = 369;
```

```
uint16_t pyro;


pyro_check4Target();
RF_check4Remote();                              //checking RF Remote

IR_right=adc_readn(1, 10);              //sample channel 0 for a total of 10 times //IR_right
= channel 0
IR_left=adc_readn(0,10)       ;                 //for IR_left
IR_North=adc_readn(4,10);

IR_SW = adc_readn(2,10);
IR_SE = adc_readn(3,10);
IR_South = adc_readn(5,10);

if (((IR_right>=threshold3) && (IR_left>=threshold3)) || (IR_North>=threshold4))
{
LCD_clearScreen();
LCD_sendString("front object");
servo_stop();
servo_delay(05000);
turn_left();
servo_delay(03000);
pyro=adc_readn(7, 50);
servo_stop();
servo_delay(05000);
turn_right();
servo_delay(07000);
pyro=adc_readn(7, 50);
        if ((threshold_high>=pyro) || (pyro>=threshold_low)) //if neutral
        {
        LCD_clearScreen();
        LCD_sendString("nothing there");
        servo_stop();                                   //immediately stop upon impact =
avoid backward current from reseting the board
        servo_delay(05000);                             //stop for 0.5sec
        back_up();
        servo_delay(05000);                             //back up for 1sec
        servo_stop();
        servo_delay(05000);
        turn_left();
        servo_delay(12000);                             //180degree left turn
        servo_stop();
        servo_delay(05000);
        move_forward();
        LCD_clearScreen();
```

```
        }
        else if ((threshold_high<=pyro) || (pyro<=threshold_low)) //"human" present
        {
        move_forward();
        servo_delay(05000);
                if (PINC == 0x3E)                       //main course = taste like
chicken
                {
                LCD_clearScreen();
                LCD_sendString("tasty");
                servo_stop();                           //immediately stop upon
impact = avoid backward current from reseting the board
                servo_delay(05000);                     //stop for 2.5sec
                back_up();
                servo_delay(15000);                     //back up for 1sec
                servo_stop();
                servo_delay(05000);
                turn_left();
                servo_delay(12000);
                servo_stop();
                servo_delay(05000);
                move_forward();
                LCD_clearScreen();
                }
        }
move_forward();
LCD_clearScreen();
}

//bump_check4Collision();               //checking BUMPERS
pyro_check4Target();
RF_check4Remote();                      //checking RF Remote

if (IR_right>=threshold)
{
LCD_clearScreen();
LCD_sendString("right");
servo_stop();
servo_delay(05000);
turn_right();
servo_delay(05100);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
```

```
if (IR_left>=threshold)
{
LCD_clearScreen();
LCD_sendString("left");
servo_stop();
servo_delay(05000);
turn_left();
servo_delay(03000);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}

bump_check4Collision();                 //checking BUMPERS
pyro_check4Target();
RF_check4Remote();                      //checking RF Remote

if (IR_SE>=threshold2)
{
LCD_clearScreen();
LCD_sendString("se obstacle");
turn_right();
servo_delay(06500);                     //90+45degree turn : to face possible target
servo_delay(06500);
servo_delay(05100);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
else if (IR_SW>=threshold2)
{
LCD_clearScreen();
LCD_sendString("sw obstacle");
turn_left();
servo_delay(05500);
servo_delay(05500);                     //145degree turn
servo_delay(05000);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}
else if (IR_South>=threshold2)      //need to add the 5th very long range IR on the back
```

```
{
LCD_clearScreen();
LCD_sendString("south obstacle");
servo_stop();
servo_delay(05000);
turn_left();
servo_delay(12000);                    //keep going forward
servo_stop();
servo_delay(05000);
turn_left();
servo_delay(03000);
LCD_clearScreen();
}

else
{
move_forward();
LCD_sendString("I Will Find U");
servo_delay(03000);                    //just so you can see the display
LCD_clearScreen();
}

bump_check4Collision();                //checking BUMPERS
pyro_check4Target();

if (PINC == 0x3E)                      //you were the main course
{
LCD_clearScreen();
LCD_sendString("Delicious");
servo_stop();                          //immediately stop upon impact = avoid
backward current from reseting the board
servo_delay(10000);                    //stop for 2.5sec
back_up();
servo_delay(15000);                    //back up for 1sec
servo_stop();
servo_delay(05000);
turn_left();
servo_delay(12000);
servo_stop();
servo_delay(05000);
move_forward();
LCD_clearScreen();
}


}//predator();
```

```
//turn_right();servo_delay(11500);turn_left();servo_delay(10000); =>  160 degree turn
//turn_right();servo_delay(07000);turn_left();servo_delay(05000); ==> 45  degree turn
//turn_right();servo_delay(05100);turn_left();servo_delay(03000); ==> 30  degree turn
//turn_right();servo_delay(06500);servo_delay(06500);turn_left();servo_delay(05500);ser
vo_delay(05500); ==> 90 degree trun // do 2x to get 180 degree rotation
//turn_right();servo_delay(13000);turn_left();servo_delay(12000); ==> 180 degree turn


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
main.c**************************************************************************
//main program of Bi-Mode


#include "main.c"               //prey.c == where I wrote the prey algorithm
#include "pyro.c"               //predator.c == where I wrote the predator algorithm
//prey.c and predator.c already contain the LCD, ADC, Bump, etc code == no need to add
them again (else u get error=redefenition error)
#include <stdio.h>
#include <inttypes.h>
#include <avr/io.h>



int main(void)                  //BI-Mode's main program
{

DDRE= 0x00;



LCD_init();
servo_init();
bump_init();
adc_init();
photo_init();
adc_init();

LCD_clearScreen();
LCD_sendString("BiMode ON");
LCD_delayShort();
LCD_clearScreen();
LCD_sendString("Mode Selection123451243512345124351245");
LCD_sendString("In Progress...");
turn_left();
```

```
servo_delay(12000);                    //2x 360degree turn = 2 full turns ==> just playing
around with the design
servo_delay(12000);
servo_delay(12000);
servo_delay(10000);
servo_stop();
LCD_clearScreen();

while(1)
{
uint8_t RT;
RT = 0;                          //RT = return to default
uint8_t WS;
WS = PINE;
WS &= 0x01;            //button4 = mode change


if (WS == 1) //if RF signal is received == PREDATOR   //only executes if RF mode
change button is pressed
{
LCD_clearScreen();
LCD_sendString("identity       12345123451243512345123 crisis");
turn_left();
servo_delay(12000);                    //2x 360degree turn = 2 full turns ==> just playing
around with the design
servo_delay(12000);
servo_delay(12000);
servo_delay(10000);
servo_stop();
LCD_delayShort();                      //going to keep this small delay to allow steady-state
settling of the pyro
LCD_clearScreen();
        while(RT == 0)
        {
        LCD_clearScreen();
        LCD_sendString("PREDATOR ");
        edge_avoid();
        RF_check4Remote();
        predator();
        WS = PINE;
        WS &= 0x01;
                if (WS == 1)   //PINE == 0x03 ||if button is pressed again => get out and
go into the default
                {
                RT = 1;
                LCD_sendString("Identity       12345123451243512345123 Crisis");
```

```
                turn_left();
                servo_delay(12000);                    //2x 360degree turn = 2 full turns
==> just playing around with the design
                servo_delay(12000);
                servo_delay(12000);
                servo_delay(10000);
                //LCD_delayShort();//REMEMBER TO TAKE OUT
                LCD_clearScreen();
                WS = 0;
                }
        }//while(RT=0)
}//no RF signal == default == PREY

RF_check4Remote();          //checking RF Remote b4 prey()
//else
//{
LCD_clearScreen();
LCD_sendString("PREY ");//16char on 1st line + 26spaces to go to 2nd line + 16char on
2nd line = space to write on LCD
edge_avoid();
prey();                     //edge_avoid: already in the prey routine
//}

}//while(1)
}//main
```