University of Florida
Department of Electrical and Computer Engineering
EEL5666
Intelligent Machines Design Laboratory

Morris
(Autonomous Pet Feeder)

Name: Joseph Stanley
Date: 8/8/02
TAs: TaeHoon Choi
Uriel Rodriguez
Instructor: A.A. Arroyo
Eric M. Schwartz

Table of Contents

Abstract	**3**
Executive Summary	**4**
Introduction	**5**
Integrated System	**6**
Mobile Platform	**8**
Actuation	**10**
Sensors	**12**
Behaviors	**18**
Experimental Layout and Results	**20**
Conclusion	**22**
Documentation	**23**
Vendors	**23**
Appendix A.	**24**

Abstract

Morris is the prototype for an autonomous servic e robot that will feed a pet at a scheduled time of the day. Once activated, Morris will move away from his starting location, locate a bowl, align himself properly, dispense the food, and return to wandering around. While performing these tasks, Morris will avoid obstacles in his quest to dispense food.

Executive Summary

Morris is the prototype for a totally autonomous service agent that will assist in the daily activity in feeding a household pet. Using a variety of sensor systems and unique platform design, Morris will navigate an area looking for a specialized homing beacon that signifies a target food bowl. Morris will then align himself with the food bowl, dispense food, and then back away and continue wandering.

The finalized design will eventually contain a timer system that will be easily adjusted to automatically activate Morris at a specified time of day. During the off times, Morris will be docked in a recharging station allowing him to stay fully charged, and awaiting the next feeding time. These systems are not active as of yet, but will be integrated eventually when time permits.

Introduction

Every day, billions of people around the world have to wake up early or go out of their way just to feed their pets. Well, maybe not everyone who owns a pet has to go out of his or her way, but there are countless people that have a strict schedule as to when their pet should be fed. But more importantly, there are also many people who don't have the ability to bend down every day to fill up the food bowl due to a disability. This is the true purpose of Morris, to assist those who have a difficult time feeding their pets every day. Once activated, Morris will track down a specially designed pet food bowl, all the while navigating household obstacles (Furniture, walls, pets, humans, etc.). After aligning himself appropriately with the bowl, food is dispensed by Morris through the bottom of his chassis, falling into the bowl. In this paper I will discuss the design of Morris, go over all the physical systems and behaviors currently in use by Morris, and then discuss my thoughts about the final outcomes on Morris.

Integrated System

Figure 1 is the overall system diagram. The heart of Morris is the MegaAVR-Dev board, made by Progressive Resources LLC. Figure 2 is a picture of the MegaAVR-Dev. This single board computer controls every sensor system on Morris; being the proximity IR sensors, contact switches, three standard servos, and the IR detectors. The board also contains the brains for Morris, the Atmel ATmega323 micro-controller, a very powerful micro-controller with many optional features.

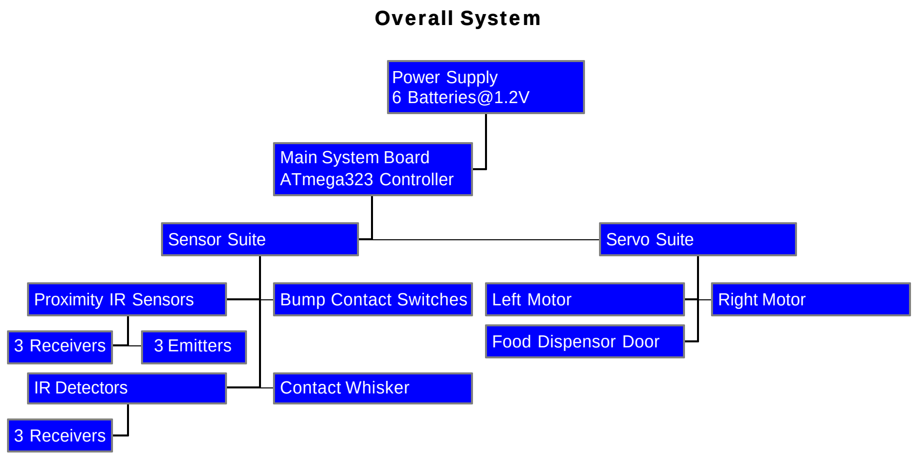

Figure 1

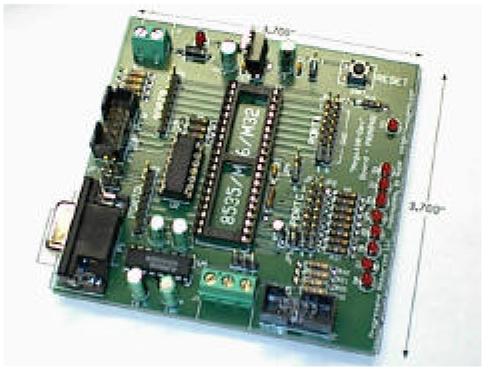

Figure 2

Two of the three servos are used for the movement of Morris, and will be hacked to provide full 360° movement. Thus Morris will be able to move forward, move backward, turn left and right, and also be able to run in place. Three IR detector/emitter pairs will be used for obstacle avoidance. Bump switches will be placed strategically around the chassis to assist the IR sensors in obstacle avoidance when the IR sensors fail. The final servo is used to control the door mechanism of the food dispenser. Finally, a set of 3 IR detectors is used to track down the food bowl that has an IR beacon attached to it.

Mobile Platform

The mobile platform used for Morris is a unique design that I have come up with. The platform is constructed out of 1/8” aircraft plywood, cut on the IMDL T-tech machine and designed in AutoCAD 2002. The platform itself resembles the shape of a “ T “ if viewed from top-down. The estimated length (From the top of the “ T “ to the bottom) is around 9”. The width of the “ T “ is around 12”. The storage compartment for the dispenser is located in the front of the robot (Being the top of the “ T “), centered on the top of the robot. Holes are made in the chassis on both the top and bottom, with the dispenser door located inside the chassis surrounded by a funnel. The height of Morris (Not counting the storage bin) was mainly determined by the height of the funnel inside, being around 7”. The bottom of the chassis has a smaller hole cut out matching the radius of the spout on the funnel. The electronics are housed inside of the robot on the back with switches and access holes located on the top for easy access. The back of the chassis is sealed off, with one removable panel on the side for reprogramming access to the serial port. But the front is open and houses the IR sensors and servos. Figure 3 is an early rough wire-frame of the skeleton of Morris (Notice the two large holes in the front, this was made for the original design of two tanks, one for water and the other for food).

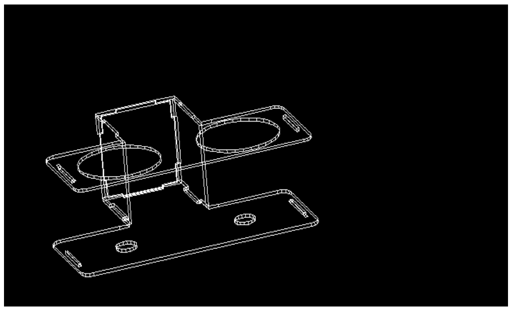

Figure 3

The platform was designed to “run over” the target bowl and dispense food that way. Thus, the platform had to be raised at least 2” off the ground in order to effectively pass over the food bowl.

Although the platform is unique, it should be obvious that I am NOT a mechanical designer by any means. The wheel mountings gave me the most problems out of everything, being very unstable and causing the wheels to cave inward. However, the platform does perform as it should, but too much weight and the wheels might cave in entirely.

Actuation

Morris requires two hacked servos for movement, and a standard servo to control the dispenser door.

Mobility Servos

Morris uses two standard Futaba S3003 servos for movement. These servos were chosen because they can be hacked to rotate 360° without the use of an H-bridge or motor driver circuit. Given 5V, these servos can generate about 40oz-in in torque. Attached to these servos are 2" lightweight aircraft wheels made out of foam. Although an odd choice for wheels, they give Morris a very flexible chassis and the ability to climb over some wide, small objects.

The servos were hacked using the hack from the Mekatronix web site (www.mekatronix.com). The physical stop inside the output gear was cut away, and the mechanical stop tab located on the output gear was cut away. After this, the servos were calibrated using a simple program. Placing 0x15 on the output compare pins, the potentiometers were then turned until the servos no longer moved. Anything higher than 0x15 causes forward rotation, anything lower causes reverse rotation. Each servo was calibrated using a TCNT value of 0xFF, a frequency of 32kHz, and using standard PWM.

Dispenser Door Servo

One, standard Futaba S3003 servo is used for the opening mechanism of Morris's dispenser door. This servo allows for precise angle control, allowing the door to open slightly.

The door itself is made out of a square piece of aircraft plywood, a small hinge, a tiny eyehook, and a tight spring. The door is mounted to the robot using the hinge. The eyehook is screwed into the bottom of the door, and the spring is attached to the eyehook. The other end of the spring is attached to one of the servo horn's holes. The servo itself is attached to the top of the chassis, placed far enough back until the spring is completely tight. The tightness of the spring determines the maximum load you can carry, being able to support more force.

The most difficult part in dealing with the servos is making sure they are hacked appropriately and are properly calibrated. This will save you a huge headache later on if you get this done early and as clean as possible. Also mounting the servos appropriately were a huge challenge for me, due to my lack of mechanical know how.

Sensors

Morris uses a variety of sensor systems in order to perform his behaviors to the highest efficiency. The sensors systems that are used by Morris are IR proximity sensors for obstacle avoidance; bump contact switches for object detection, a IR beacon/detection system for finding a bowl, and contact whisker for bowl positioning and backup avoidance.

IR Proximity Sensors

Three Sharp GP2D12 Proximity IR sensors are used for object avoidance. These are mounted on the chassis in an appropriate manner as to prevent Morris from colliding with any objects; one centered and the others facing 45 degree angles on both sides.

The GP2D12 IR sensors are a two-part device. One part of the device is an IR LED that operates at 40Khz. When the IR light hits and object, the beam is bounced back and the second part of the device, an IR detector, picks up the bounced signal. This detector is set to filter out any noise by listening only for the 40Khz signal. Depending on the distance from the sensors, the value that is sent as an analog output will vary. Using a threshold value of around 70-85, Morris can avoid obstacles at a distance of about 4". Anything closer than 2" yields inaccurate values that cannot be used to determine a correct distance. At 2" away a value of about 125-130 is produced, and at about ½" away, the value is 90, even though it should be greater than 130.

Bump Switches

When an object fails IR detection, a backup system must be employed. This backup system is in the form of bump contact switches. These switches are located on the edges of the chassis in strategic locations, particularly on the sides of Morris, the under-chassis, and the front beside the IR proximity detectors. These switches are wired directly through internal pull-ups to PORT B on the MegaAVR-Dev board. When a switch causes a pin to go high, an appropriate turning behavior is produced.

IR Beacon System

The core feature of Morris it the ability for him to find the target bowl to dispense food in. After some thinking, I figured the best approach to this would be to use a simple IR beacon / detector system. After deciding on the course of action, I then sat down to figure out how the system would be mounted on both the bowl and on Morris, and also how many detectors/emitters I should use to be the most effective. Figure 5 is a sketch of the final realization I decided on which would hopefully yield the most effective solution to the problem at hand.

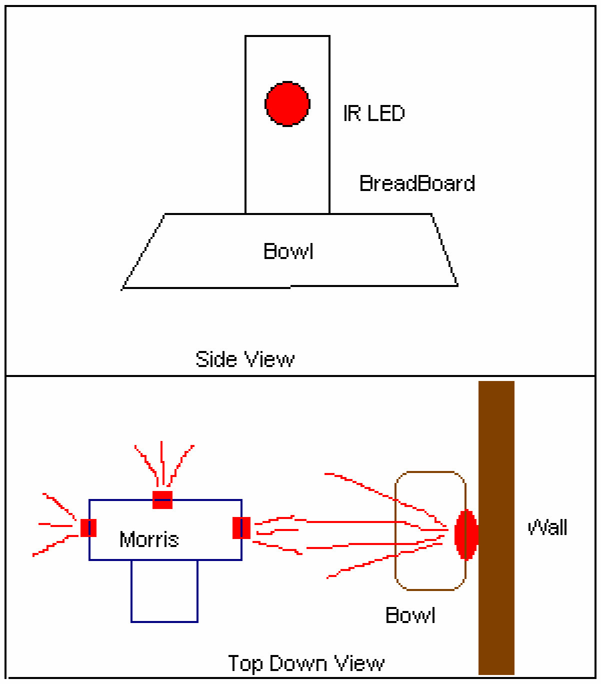

Figure 5

As can be seen from figure 5, one IR LED emitter is mounted to the bowl via a breadboard, which contains all the modulation circuitry required for the IR LED (described later). This breadboard also serves as a means to elevate the IR LED slightly so it can be detector by Morris, which is more elevated than the bowl alone. The only limitation of this approach is the fact that the bowl must be placed near a wall, with the LED facing perpendicular to the wall. By placing the bowl in this manner, Morris will never travel “behind” the LED and yield the best results for acquiring the bowl.

In order to get the LED to modulate at the appropriate frequency, a 555-timer control circuit is needed. Using a standard 555 timer, the circuit in Figure 6 is used to generate the appropriate frequency of about 56.5kHz.

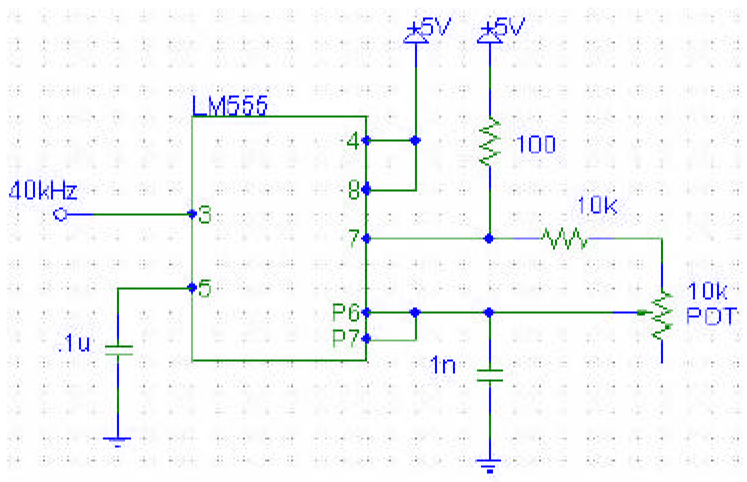

Figure 2

The circuit is fairly simple. The timer is running in the astable mode of operation and has a duty cycle of approximately 48.4%. The pot is used to change the frequency from 56.7kHz down to about 29.9kHz. This allows IR detectors at different frequencies to be used if there are interference problems. This circuit was borrowed from Michael Hattermann’s robot, STEVE.

To power the IR LED circuitry, I am using a simple 5V AC/DC adapter. This will keep the beacon running at all times, and hopefully an outlet will be nearby. In the case where one is not, 4 AA batteries will do the trick, or a 9V battery would do fine as well.

Now that the bowl beacon has been setup, an appropriate detector system is needed on Morris himself. As can be seen above from figure 1, I decided on using 3 IR-detectors mounted on the front, left side, and right side of Morris. The detectors I am using are LiteOn IR receivers, modulated at 56.5kHz. These detectors initially give out a digital signal, but Michael Hattermann discovered a simple hack in order to get an analog signal out of them. Figure 7 illustrates the hack, which was borrowed from Michael's report.

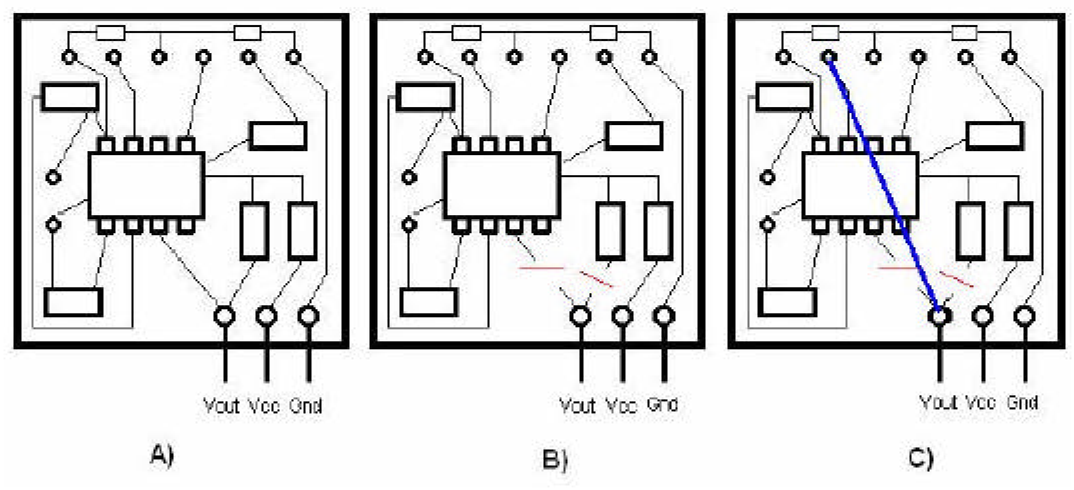

Figure 7

Figure 3 is the inside of the IR can. In order to hack the IR module, first cut the traces to the output pin, which is shown in step B. Then, solder a piece of wire connecting the output pin to the appropriate pin shown in step C. After this hack is complete, the voltage on the output pin is about 1.5V when no IR is detected to about 2.5V when IR is less than an inch from the detector.

When no IR is detected from the emitter, the detectors output an ambient voltage of around 1.52V, and when the IR is at its max, around 2.30V. The maximum effective range I have achieved is around 12 feet, maybe even further. However, anything beyond

5-6 feet and its hard to accurate find the source because of the 30 degree cone emitted from the LED.

Which leads me to my most difficult and still prominent problem I have faced during the course of the robot project. I have come to realize that using IR for a beacon type system is NOT very accurate at all. Due to the large cone emitted from the IR LED, it is very difficult for Morris to position himself correctly with the target food bowl. Apparently, the strongest part of the 30-degree cone is not in its center, but the EDGES of the cone, making it even more difficult for me to locate the source of the LED. With a little more time ingenuity, it is possible to maybe triangulate the source by finding the 2 edges of the cone and triangulating the center spot.

Behaviors

Obstacle Avoidance

Morris implements obstacle avoidance using the IR sensors and bump switches. When an IR sensor outputs a larger value than the threshold value, Morris will turn to avoid that obstacle. The bump switches and contact whisker are used when the IR sensors fail to detect an object in the robots path. This will cause Morris to stop, backup, and turn away from the object before progressing forward.

Bowl Hunting

This is the primary behavior of Morris, and the one that most time is spent in. Morris will wander a room waiting for his IR detectors to pick up a signal. Once a signa l has been located, Morris will attempt to position himself with the bowl. If he approaches the bowl at a bad angle, he will turn around and wander around until a better angle is found to align him. This was implemented due to the inaccuracy of using IR as an accurate positioning device.

Food Dispensing

Once the bowl has be found, Morris will align himself with the bowls signal and begin to travel over the bowl. When the dispenser is located above the bowl, a contact switch on the bottom of the chassis is used to let Morris know he is right above the bowl and to open the dispenser door. The dispenser door will then open, and stay open until enough food has been dispensed, and will then shut again.

Endless Wandering

Once dispensing is complete, Morris will begin to wander aimlessly, ignoring the IR signal coming from the bowl. This behavior will be used in the future to implement the return to a base station to recharge and wait for the next feeding.

Experimental Layout and Results

Experiment 1: Proximity IR Threshold

This experiment was necessary to find the threshold value for the IR proximity detectors. The code irtest.c in appendix A was used to display the values taken from the IR detectors, among other things. The readings from the IR detectors were read when my hand was placed at several positions in front of the platform. It was determined that the threshold value for the IR detectors should be 85, when an object is about 4” in front of the robot.

Experiment 2: Servo Calibration Test

This experiment was necessary to find the values to stop and move the servo in a specified direction. It was found that using a stop value of 0x15 and calibrating the servos to that value, forward rotation is given by using any value greater than 0x15, and reverse rotation can be found by using values lower than 0x15. I use 0x10 for reverse movement, and 0x30 for forward movement.

Experiment 3: Beacon Proximity Value

This experiment proved to be the most difficult to perform due to the characteristics of the IR LED. Using the code found in irtest.c in appendix A, the IR emitter was placed at various places around and away from the robot and values were examined. A maximum effectives range was found at around 12’, but the values were small at that range, around 82-83 (The ambient value is about 79-80). When an object is closer than 3’, a value greater than 90 is given when the detector enters the emitter’s range, but the value reduces as the detector approaches the emitter’s center, and then rises again as it approaches the opposite edge. Thus, the voltage output is higher when the detector is on the edges of the LED’s output cone.

Conclusion

In regards to my original plans, I did not accomplish all that I originally intended to do. The timer and recharge behaviors were intended to be a key function to Morris's overall behavior, but due to the problems found with the IR detections and personal time constraints, I did not have enough time to realize these two systems into Morris. However, I am very pleased with what I have accomplished, being my first hands on project that I have ever done in the field of electronics, micro controllers, robotics, and mechanical design.

I was very pleased with how the door worked out for the food dispenser, and the entire dispenser unit turned out to be a huge success from the start. This is probably the only area that exceeded my expectations. On the other end of the spectrum, the IR detector system and the platform itself are the two areas that need to be heavily improved in order to be 100% effective.

Even though the class is over, I plan on continuing my work on Morris and adding all the features that I originally wanted on him. I will perhaps use a sonar system as opposed to an IR system for beacon detecting, and I will give Morris a huge facelift in terms of platform design.

Documentation

Thanks to
IMDL class: Instruction from Dr. Arroyo, TaeHoon Choi, Uriel Rodriguez, and Dr. Schwartz.

Vendors

Cables and Connectors
2315 Berlin Turnpike
Newington, Ct. 06111
(860) 665-9904
Fax (860) 665-9993
www.cablesandconnectors.com

Mark III Robot Store
No Address Provided
www.junun.org/MarkIII/Store.jsp

Progressive Resources LLC
4105 Vincennes Road
Indianapolis, IN 46268
(317) 471-1577
FAX 471-1580
www.prllc.com

Radio Shack
3315 SW Archer Road
Gainesville, FL 32608
352-375-2426
www.radioshack.com

ServoCity
620 Industrial Blvd.
Winfield, KS. 67156
1-877-221-7071
www.servocity.com

Appendix A

MAIN PROGRAM

```
#include <mega323.h>
#include <stdio.h>

#define ADC_VREF_TYPE 0x20

int ir_left,ir_right,ir_center,i,x,n;                    // Global Variables
int beacon_left,beacon_right,beacon_center;
int doorswitch;
int beacon_found = 0;
int task_complete = 0;

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSR|=0x40;
// Wait for the AD conversion to complete
while ((ADCSR & 0x10)==0);
ADCSR|=0x10;
return ADCH;
}

void init()
{
// Input/Output Ports initialization
// Port A initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T State6=T State7=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func0=In Func1=In Func2=In Func3=Out Func4=In Func5=In Func6=In Func7=In
// State0=T State1=T State2=T State3=0 State4=T State5=T State6=T State7=T
PORTB=0x00;
DDRB=0x08;

// Port C initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T State6=T State7=T
```

```
PORTC=0xFF;  // Disregard above comment, all pins are OUTPUTS
DDRC=0xFF;

// Port D initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=Out Func6=In Func7=Out
// State0=T State1=T State2=T State3=T State4=T State5=0 State6=T State7=0
PORTD=0x00;
DDRD=0xA0;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 31.250 kHz
// Mode: Phase correct PWM top=FFh
// OC0 output: Non-Inverted PWM
TCCR0=0x64;
TCNT0=0xFF;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 31.250 kHz
// Mode: Ph. correct PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x81;
TCCR1B=0x04;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0xFF;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 31.250 kHz
// Mode: Phase correct PWM top=FFh
// OC2 output: Non-Inverted PWM
TCCR2=0x66;
ASSR=0x00;
TCNT2=0xFF;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
GICR=0x00;
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x08;
UBRRL=0x26;
UBRRH=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 4000.000 kHz
// ADC Voltage Reference: AREF pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE;
ADCSR=0x81;
}

void delay(int delay_loops)  //Delay for a specified amount of time
{
        int n,i;
        for(n=0;n<delay_loops;n++)
        {
                for(i=0;i<60000;i++);
        }
}
```

```
void read_doorswitch()    // read the analog value of the doorswitch
{
        doorswitch = read_adc(4);
}

void read_beacon()        // read the analog value of the beacon detectors
{
        beacon_center = read_adc(7);
        beacon_left = read_adc(5);
        beacon_right = read_adc(6);
}

void read_ir()            // read the analog value of the proximity detectors
{
        ir_center = read_adc(0);
        ir_right = read_adc(1);
        ir_left = read_adc(2);
}


// MOVEMENT CODE CONTROL FOR SERVOS
void stop()
{
        OCR2 = 0x00;
        OCR0 = 0x00;
}

void go()
{
        OCR2 = 0x30;
        OCR0 = 0x10;
}

void back()
{
        OCR2 = 0x10;
        OCR0 = 0x30;
}

void right()
{
        OCR2 = 0x30;
        OCR0 = 0x30;
}

void left()
```

```
{
        OCR2 = 0x10;
        OCR0 = 0x10;
}

void open_food_door()           //Opens the dispensor door
{
        OCR1AL = 0x10;
        delay(3);
        OCR1AL = 0x00;
        delay(15);
        OCR1AL = 0x30;
        delay(5);
        OCR1AL = 0x00;
}

void beacon_track()
{
        while(beacon_found == 1)
        {
                PORTC.0 = 0;
                go();
                read_doorswitch();
                if(doorswitch > 100)
                {
                        stop();
                        delay(9);
                        open_food_door();
                        task_complete = 1;
                        beacon_found = 0;
                        PORTC.1 = 0;
                        back();
                        delay(19);
                        left();
                        delay(19);
                        go();
                        return;
                }
                read_ir();
             if(ir_left > 70 && ir_left < 200)
                {
                        right();
                        delay(19);
                        go();
                        beacon_found = 0;
                        return;
                }
            if(ir_right > 70 && ir_right <200)
            {
                        left();
                        delay(19);
                        go();
                        beacon_found = 0;
                        return;
                }
                if(ir_center > 90 && ir_center < 200)
                {
                        back();
                        delay(20);
                        right();
                        delay(25);
                        go();
                        beacon_found = 0;
                        return;
                }
        }
}

void found_left()
{
        delay(6);
        left();
        delay(20);
        go();
        beacon_found = 1;
        beacon_track();
}

void found_right()
{
        delay(6);
        right();
        delay(20);
        go();
        beacon_found = 1;
        beacon_track();
}


void search()
{
        x = 0;
        PORTC.0 = 1;
        if(task_complete==0)
        {
                read_beacon();
                if(beacon_left>85)
                        found_left();

                if(beacon_right>85)
                        found_right();
        }
        read_ir();
      if(ir_left > 70 && ir_left < 200)
        {
                right();
                delay(20);
                go();
        }
      if(ir_right > 70 && ir_right <200)
      {
                left();
                delay(20);
                go();
        }
        if(ir_center > 70 && ir_center < 200)
        {
                back();
                delay(20);
                right();
                delay(25);
                go();
        }
        else
                go();

}

void main(void)
{

        init();
        while (1)
        {
                search();

        }
}
```

IR/BEACON/DOOR TEST PROGRAM

```
void main()
{
init();
while(1)
{
read_ir();
printf("Center: %d\n",ir_center);
printf("Left: %d\n",ir_left);
printf("Right: %d\n",ir_right);
delay(15);
read_beacon();
printf("Be Center: %d\n",beacon_center);
printf("Be Left: %d\n",beacon_left);
printf("Be Right: %d\n",beacon_right);
delay(15);
read_doorswitch();
printf("Door: %d\n",doorswitch);
delay(15);
}
}
```