

# Richardo Fourcand

## AIRFA

r4cand@eel.ufl.edu

(352) 335-8797

AIRFA is an autonomous agent that can detect a laser beam being fired at it, track a person and fire back. AIRFA uses two M68HC11 processors communicating via the serial lines.

Airfasp.wpd

# Table of Contents

Abstract.....	Page 3
Executive Summary.....	Page 4
Introduction.....	Page 6
Integrated System.....	Page 7
Mobile Platform.....	Page 8
Actuation.....	Page 9
Sensor.....	Page 10
Behavior.....	Page 13
Conclusion .....	Page 14
Documentation.....	Page 15
Appendix A.....	Page A-16
Integrated System.....	Page A-17
Mobile Platform.....	Page A-18
Actuation.....	Page A-20
Sensor.....	Page A-22
Behavior.....	Page A-26
Appendix B.....	Page B-27
Code for lower microcontroller .....	Page B-28
Code for Upper microcontroller.....	Page B-33
Code for laser module.....	Page B-39

## **Abstract**

AIRFA is the name of my robot. AIRFA stands for Autonomous IR Firing Agent. The purpose of my robot is to avoid various obstacles while a servo rotates to track a person's movement. My robot AIRFA is equipped with several sensors and actuators. There are 10 IR detectors and 14 IR emitters for basic obstacle avoidance. There are a set of 4 bump sensors to react to obstacles which cannot be detected by the IR sensors. There are also 3 cadmium sulfide cells to help the robot react to changing lighting conditions. For tracking I used 3 passive infrared (PIR) detectors to detect movement. To fire a laser at the robot I created a laser gun with a 40 kHz frequency. AIRFA detects the laser using a photo detector. AIRFA can fire back a 33.9 kHz IR to a person's 32.7 kHz detector. For actuation I am using three servos. Two of the servos are hacked to move the wheels of the robot while the other is used to turn the disk to rotate the robot's upper sensor array. To program AIRFA I decided to use ICC11. This allows for more compact code and interrupt handling.

## Executive Summary

For the project I decided to create a robot that I could use to play laser tag with. To do this I began with a basic platform starting from my M68HC11 EVBU board and an ME11 expansion board. I decided to use ¼ inch piece of airplane plywood for the main platform. For the secondary platform I used 1/8 inch piece of airplane plywood. The structure is supported by four 5 inch bolts. For the sensors I began with 40 kHz IR generated by the ME11 board and 40 kHz IR detectors. Then I added cadmium sulfide cell and four normally open bump switches. These sensors are tied to the analog port of the first processor and are primarily used for the actuation of the motors. To get the various readings from the analog ports I used IC. This allowed for basic data retrieval and movement for my robot.

To track a person AIRFA uses three PIR motion detectors. The detectors are set so that one PIR detector is directly in front and the other two are 45 degrees from the center position to the left and right direction.

To detect a laser shot I set up two types of sensors, a 40 kHz light detector using a hacked Sharp IR sensor and replacing the IR photo diode with a Sharp BS-120 visible photo diode. The second type of sensor I used was a columnated cadmium sulfide cell. These two sensors can be interchanged with the robot without any difficulty.

To fire back at a person, I generated a 33.9 kHz IR signal using a CMOS 555 timer circuit. This circuit is controlled by the robot's second microprocessor.

AIRFA uses two separate controllers. The first controller controls the movement of the robot. Here is where the data from the obstacle avoidance IR, cadmium sulfide, and the bump sensor. The second processor is used for the control of tracking algorithm, IR gun firing and laser

detection. To communicate the processors use their serial communication lines.

For the laser gun module I use a 3 mW laser diode to fire a 40 kHz laser beam at the robot. The 40 kHz laser beam is generated using a single chip board's E-CLOCK and a clock circuit. To power the diode I use a variable voltage regulator to set up the voltage need. The module also uses four 32.7 kHz IR detector to detect when the robot fires back.

To code the robot I used both IC and ICC11. I wrote the first program using IC since it shows the fastest result also since it allow dynamic manipulation of the various ports and sensors I allowed me to look at the values from the various readings. I used ICC11 when I was writing the final set of code for the robot because of its versatility. It allows for the use of interrupts and one has more of the valid data types available for use.

## **Introduction**

For EEL 4914, I chose to add to the robot which I created in EEL 5934. These addition would make the robot a autonomous IR firing agent. The addition which I made to the robot were the addition of the laser detectors, two IR emitters to fire at a person and the laser gun module. For a discussion on the previous additions to AIRFA see appendix A.

I also made several improvements to the code and other parts of the robot. The code was improve to allow for more complex communication between the two processor on the robot. Also a dynamic was used in the motor driving routines to allow for a smother turning algorithm then was originally used. I also modified the bumper to remove some of the problems which occurred.

# Integrated System

AIRFA is made up of two distinct parts the upper half and the lower half. The lower half of the robot is where the basic actuation for motion takes place. It is here that the results from the various sensors are processed to produce the robot's movement. Since ICC11 does not facilitate multitasking like IC the functions which I implement are turn bases. The robot begins by moving around a room avoiding objects and waiting to detect something. If there are no objects in front the robot will spin its head and fire in the back just to make certain it is not being stalked. This is done by the lower processor sending a serial signal to the upper processor telling it to turn and fire. When the robot gets bumped it moves backward until it is turned again. This action signals the head that it wants to turn to face the same direction that the robot is firing to. When the upper half of the robot detects a heat source using the left PIR a signal via serial communication a code is sent to the lower processor to turn left. The same applies for a signal detected on the right. If the center PIR detects motion then the processor fires a shot and signals to the lower processor to turn away. The processor then using a random number generator the processor decides whether or not to turn in the direction signaled. This random number generator is used to help factor out the multiple of signals caused by noise and multiple sources of body heat that may be present. If when a hit is detected the lower processor is signaled to begin spinning while the upper processor waits for a signal to continue.

The laser module is the source where the laser signal is generated from. When the robot fires its IR guns the detectors detect the hit and register a beep so that a person knows that they have been hit.

This integration of behaviors all the robot to act like a laser firing agent.



## **Mobile Platform**

The platform of the robot did not change much from the original design. The only addition was a tower about 2 ½ inches tall and 1 inch wide used to mount the IR gum on each side. The laser detector is mounted on the top of the platform.

The platform for the laser module is a box made of 1/8 inch airplane plywood. It has a hinge so that when opened the one can change the battery or modify any circuits internal to the module. To keep the opening closed velcro was used to seal the box shut. The 32 kHz detector are mounted on a cross made up of the same type of plywood mentioned earlier. The laser diode is mounted onto 4 in brass tube which is then connected to a wooden handle.

This is the complete design of the robot module and platform.

## Actuation

The hardware for the actuation of the robot did not change so for a discussion of what was done please refer to appendix A. The software however did change.

The original code used a linear increase to speed up and slowdown the motor. Such that  $\text{motor\_speed} = \text{motor\_speed} + 1$ ; and some time delay. The problem with this is it gave the robot a jerk movement at times and if the  $\text{motor\_speed}$  went from -100 to 100 it would take 201 cycles complete. This length of time would waste the CPU. So I change the  $\text{motor\_speed}$  equation to  $\text{motor\_speed} = \text{motor\_speed} + (\text{target\_speed} - \text{motor\_speed})/\text{ACC}$ . This equation is now on the order of  $\log(n)$  when the other was on the order of  $(n)$ . This mean that there are fewer cycles needed to change to the target speed.

With this equation change the actuation code for the robot simplified since with the linear equation you need to use twice the number if lines since the equation will vary depending on whether or not the motor speed is increasing or decreasing. While the dynamic self compensates.

## Sensor

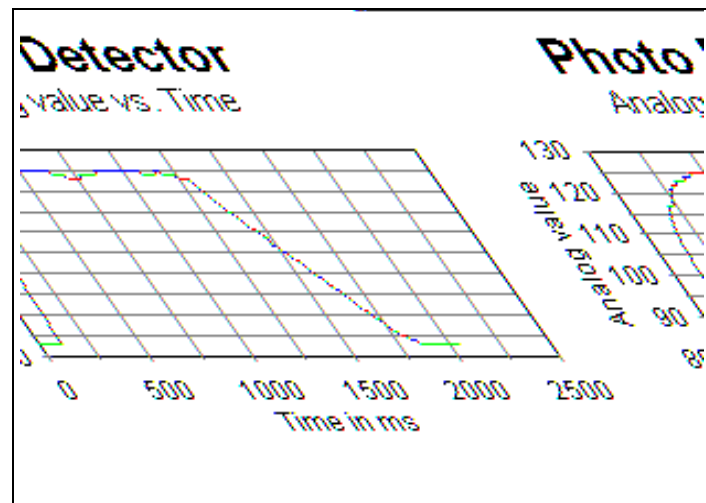
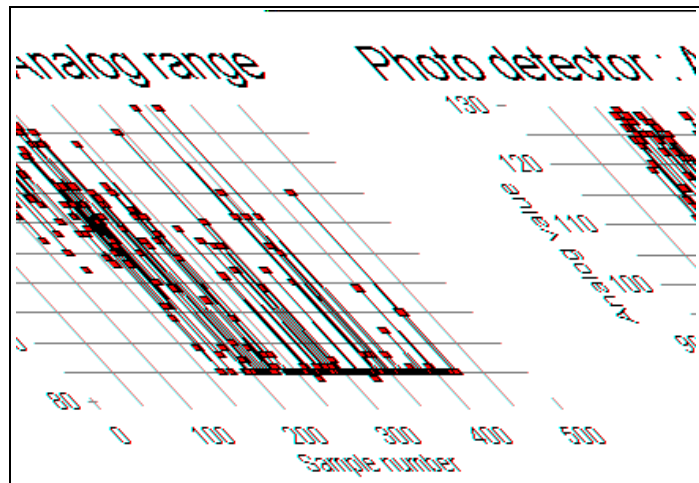
There are five main sensor types which I used for AIRFA. For obstacle avoidance, I used IR and bump sensors. I also use three cadmium sulfide cells for light detection. And lastly I used an ultrasonic motion detector. These above mention sensor were used in the original robot so see appendix A for a complete discussion. For the robots, augmentation I used the PIR, 32 kHz IR emitter, 32 kHz detector, visible photo detector and cadmium sulfide cells.

The PIR motion detector was mounted and test for the original design but it was not implemented. The PIR detector was chosen to track a person instead of the ultrasonic motion detector because it can work while the robot is moving while the ultrasonic motion detector cannot. This is because the ultrasonic motion detector detects changes in the phase shift of the signal and since the robot is usually moving there is usually motion detected. The PIR on the other hand detects changes of the passive IR spectrum caused by the movement of a heat source, such as a person. The sensor I used has a digital output so it declares that there either is or is not a heat source or person near. The only draw back is that other objects which emit IR or have a temperature greater than that of its neighbor will also set it off. It also has a setup delay of roughly .5 sec. Even with these draw backs it does a good job detecting and tracking a person.

The next sensor which was added was a 32.7 kHz IR detector. It behaves similar to the 40 kHz detector used except that it works with a lower bandwidth. This detector is mounted onto the laser gun module. It is used to detect the IR signal generated by the CMOS 555 timer circuit. The timer circuit is power off the 8 volt battery to increase the output to the IR LED's. By powering off the voltage the frequency becomes a function of the voltage as well as from the equation  $T =$

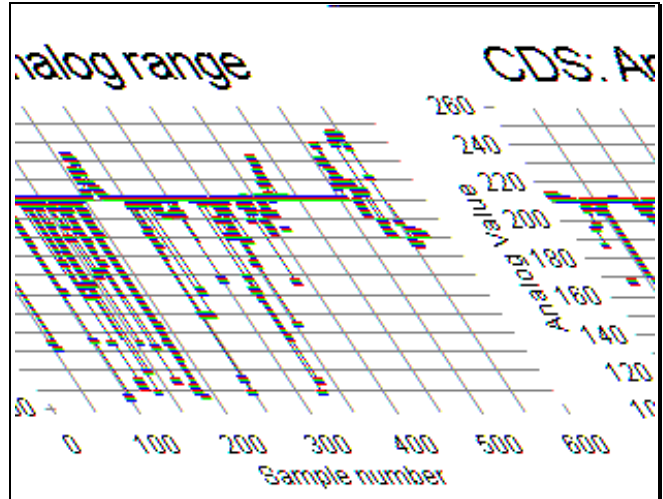
$\ln 2(R_a + 2R_b) \cdot C$ . The frequency range of the 32.7 kHz IR detector and the fact that the frequency will shift at most 2 kHz still makes the detector a feasible choice for the robot and person IR battle.

Then next set of sensors were the 40 kHz visible light detector and the CDS cell. Each of the detector were used to detect the 40 kHz laser light signal. The 40 kHz visible light detector was made up of a hacked 40 kHz IR detector. The IR photo diode in the can was removed and replaced with a BS-120 visible light diode. Now the IR can detect the 40 kHz visible light and ignore any other sources. At the head of the can a lense was added to help focus the light so that it can be hit by angled shot thus making it easier to hit. Figure 1 show the range of the analog values. Figure 2 shows how the

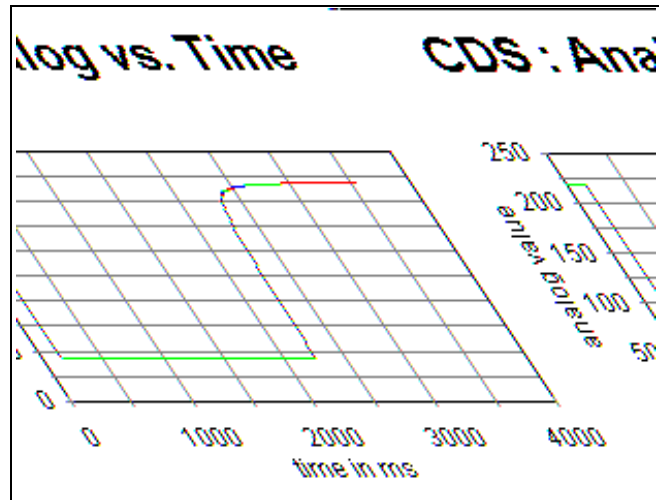


detector reacts over time. The monitor was set to display the analog values and the laser was fired at the detector from 4 feet away. A graph was made to see how the signal react over time.

Then next laser detector I used involved a columnated CDS. The CDS cell has a 1 K resistor in series with it and the voltage change can be check. The CDS cell does not require the laser to modulated at 40 kHz. Figure 3 shows the analog range of values from the CDS cell. Figure 4 show how it reacts over time.



Since both sensors can detect the 40 kHz light well, there may seem to be no reason to chose one over the other well that is wrong. When testing the two detector I found that the lights in the room generated a 40 kHz signal which would interfere with the detector. This



interference with the 40 kHz light detector but not CDS cell. The CDS cell discharges much quicker so that if the second microcontroller does not sample the laser while it is being fired then it may not be detected. The photo diode on the other hand will store a detectable value for about .5 seconds after the laser was fired giving the microcontroller time to react. To offset this time for the CDS require I increase the sampling rate of the CDS detector which would increase the complexity of the code.

So the differences in each of the detectors must be factored in to decide which one to use.



## **Behavior**

There are a series of behaviors which AIRFA is able to complete. The first behavior is obstacle avoidance through the use of 40 kHz IR. When the signal reached a certain value then the controller knows that there is an object up ahead so that means it is time to turn. The next behavior deals with those objects which are not avoided by the IR. When the bumpers hit an object the controller reacts by shifting the robot into reverse and continuing from there the second controller then turns the robots head so it is facing the direction that it is moving. The next on turns the robot to track a person using the PIR. The next set of behaviors deal with the laser light detector. If the second controller read a signal that is close to the hit value but not the hit value it will fire and turn to avoid the detector reaching the hit value. If it detect a hit the robot spins around and then continues on its way. These behaviors are the basis of the robot abilities to play laser tag with a person.

## **Conclusion**

I completed the work I began in EEL 5934 to my robot AIRFA. The robot is now a valid partner for playing laser tag with the robot I found it more fun to use the 40 kHz detector since it stored the value long enough that it would register hits even if it occurs some time after the laser was already fired. The CDS cell was difficult to play with since it is columnated the only shots which could be detected had to go straight down the pipe. It did make the robot more challenging. So no matter which device was used the robot was enjoyable to play with. AIRFA tended to fire frequently enough to keep me on my toes but not so often as to make the game frustrating. In all the robot perform extremely well.



# Documentation

Nova-Soft web site

6.270 LEGO Robot

Fourcand, Richardo; AIRFA 1996

Jones, Joseph L. and Flynn, Anita M.; Mobile Robots Inspiration to implementation; A K Peters;  
1993

# Appendix A

## Integrated System

AIRFA is made up of two distinct parts the upper half and the lower half. The lower half of the robot is where the basic actuation for motion takes place. It is here that the results from the various sensors are processed to produce the robot's movement. Since ICC11 does not facilitate multitasking like IC the functions which I implement are turn bases. The first state of the robot begins with a 60 degree rotation until a movement is detected by the ultrasonic motion detector. When motion is detected a signal is sent to the lower micro controller via the serial data lines. After this communication the two processors' applications diverge.

The lower processor goes through four basic functions. The first is *Avoid*. When an object enters the range of the IR detectors it turns left or right to avoid the object. After the turning function, the bump function is called. It causes the robot to change direction from forward to backward or vice versa depending on whether or not a hit has been detected. The final lower level sensor is the cadmium sulfide cells. These cells change the overall speed of the robot whether it is dark or bright so that it can go into a tracking mode. With the completion of this last function the list is looped again.

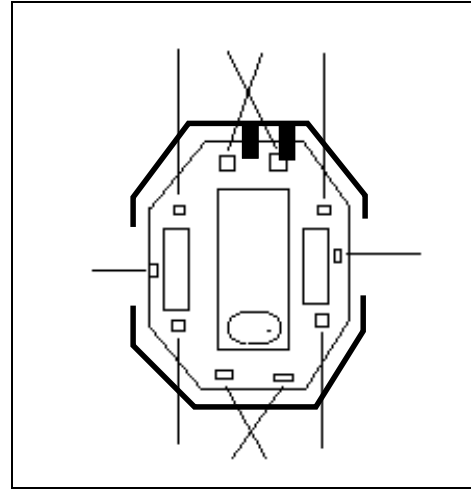
Simultaneously, the second processor is used to track someone. The top disk turns 360 degrees to keep a target on a person. This is done without any knowledge to the lower processor. This works because the three PIR motion detectors react to changes of heat and detect the movement of people. When the middle one gets a signal the other two on the left and right cause the servo to turn so that there is a lock on to the object being tracked.

This structure allows for the integration of four of the sensors to work in unison while the final one works in the background. Each sensor is discussed further later in the paper.

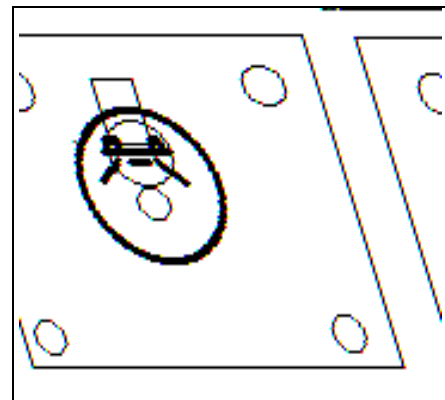


## Mobile Platform

The design of the mobile platform is seen in Figure 1. The 10 boxes on the outer sides denote the positioning of the IR sensor on AIRFA. The dark boxes in the front denote the positioning of the cadmium sulfide cells. The borders in the front and back of the platform are the bumpers which are on the robot. The large rectangle in the middle is the EVBU board and the small rectangle in at the end is the positioning of the



battery pack. AIRFA is designed with an octagonal shape to facilitate the ease at which the sensors are placed. Also Scott noted that a rounded shape keeps the robot from becoming stuck on any edges and make maneuvering around objects somewhat easier. The placement of the wheel at the center of the robot keeps the turning rotation of the robot simple and easier to manage. Figure 2 is a picture of the robots upper platform. The four circles on the side is where the anchor screws are placed. The object in the center are the servo and sensors which go on to the top. The two light inner-circles are the gears used to give the servo more than 180 degrees of movement.



In designing the lower platform, I used ¼" piece of plywood. This material was used because it is light and sturdy. Also the wood is easy to modify and change if any situations should arise in the construction of the robot. The platform is 9" long and 10¼" wide. When I was

cutting the platform I found it helpful to draw various lines on the board to get a feel for the overall design instead of using a piece of paper or a CAD program. The only problem which occurred was that I initially had the wheel lopsided so that I had to cut an extra  $\frac{1}{2}$ " on one side to make it even. For the upper platform I used on  $\frac{1}{8}$  in of the same type of wood since this platform supports less weight. For the rotating disk I am using a metal disk to mount the sensors. The platform design seems easy to manage and improve if a situation should arise. The only problems that came about was that the cutting and shaping is a bit lop sided at place.

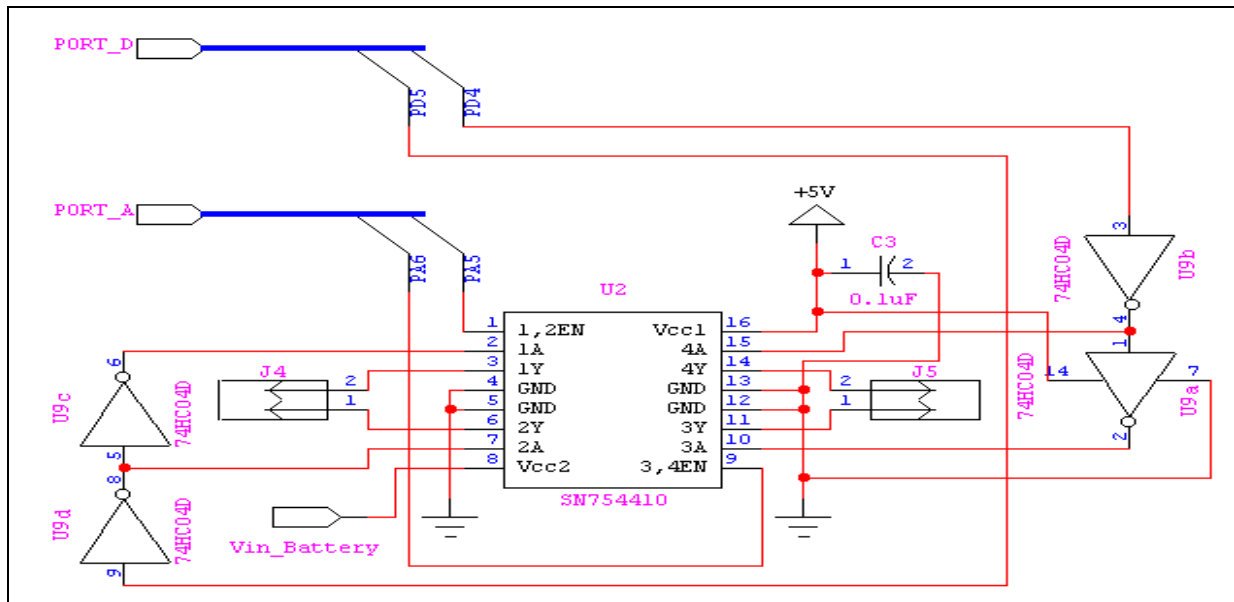
A useful building tool since it was easy for me to add holes to support the various design changes that the robot went through as the semester rolled on.

## Actuation

AIRFA's motor system was simple since it required very little hard wiring. To drive the 3" wheels of the robot I used two servos like the one in Figure 3. The servo has a torque of 42oz-in, operates at 5



volts, and has a speed of 0.22s per 60 degrees. The servo's had to be hacked to enable the use of them in the robot. This was done by removing a stopper tab on one of the gears and by removing the motor control board inside the box. I removed the yellow control wire since it was no longer necessary. To control the motor I used the ME11 board's built in motor controller. There is a schematic of it in Figure 4.



I also used an unhacked servo in the design of AIRFA. I geared down the servo 38 to 16 so that there will be more than 180 degree rotation of the servo. To gear it down I brought a set of gears used in remote control racing cars. They seem to work nicely.

In designing the diving system for the robot I found that getting the wheel straight was the most difficult task. Also for the servo hack, it was easier to use a soldering iron to smooth out the rough edges. For the servo the difficulty arose in putting the two gears together. The gears need to mesh closely to limit the amount of slipping that can occur.

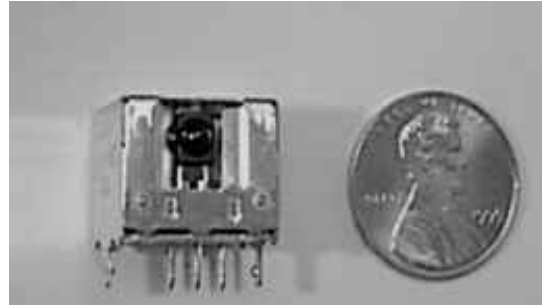
The code for the various actuation appears in the appendix.



## Sensor

There are five main sensor types which I used for AIRFA. For obstacle avoidance, I used IR and bump sensors. I also use three cadmium sulfide cells for light detection. And lastly I used an ultrasonic motion detector and a PIR motion detector to track a person.

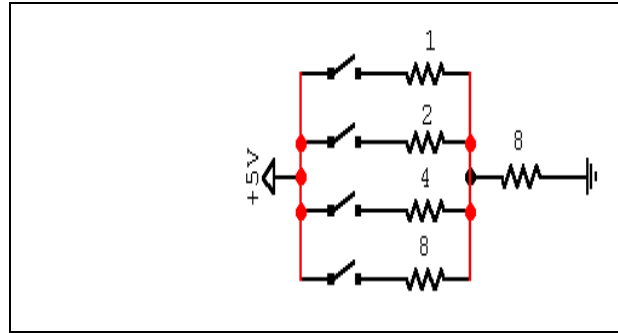
The first sensor I used was the IR. The IR configuration for the detectors is seen in Figure 1 and a picture of the IR detector is seen in Figure 5. The IR detectors receive information from the 14 IR emitters which let out a 40 kHz IR beam. The IR emitters are tied in parallel. A parallel combination of 5 emitters,



in the front and back, that draws on 2.5 mA of current and run on a 1 volt voltage. There is also a parallel combination of 4 emitters for the side which draw about 2 mA. I found that the difficult part came in hacking the sensor since there was a small space to work with. However, I did create a small system to test the IR detectors. This was done by wiring an IR LED to the 40 kHz J6 and the sharps were wired to power and ground. The LED was then waved in front of the device and I watched the oscilloscope to detect changes in the voltage. Only one of the ten detectors failed the test due to a badly cut trace.

After placing the IR detectors I began to write the obstacle avoidance code which is to be used. I found for the various positions of the sensors require different threshold voltages. For example a back sensor may have a threshold of 118 before it needs to turn to avoid an object while another may have a value of 131 before it needs to turn. Other than the various thresholds of the detectors IR seemed to be easier to use.

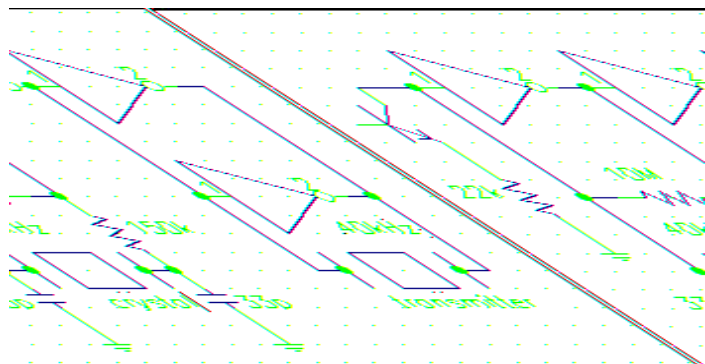
To help with obstacle avoidance I use a series of 4 bump sensors. The bump sensors are tied to a resistor network as seen in Figure 6, so that only one analog port is used to detect any bumps. When a bump is detected the voltage rises from 0 to larger depending in which of the switches is closed.



The main problem for the bump switches was in implementing the bumper structure which I would use on the robot. After several attempts I chose a design using a series of tubes to wrap around the front and serve as bump sensors.

The next sensor I built was for cadmium sulfide cells. This circuit design is just a simple resistor divider network. For the network I used 100K resistors. Depending on the light source the data would fluctuate for various amounts. After taking several readings I declared that anything below 20 was absolute dark and anything above 200 was absolute light. From there the coding and use of the sensor became very easy.

The ultrasonic motion detector has two main parts a transmitter and a receiver. The transmitter circuit is shown in Figure 7.



This is a crystal lock transmitter that causes the transmitter to generate a constant 40 kHz signal. The basic circuit on the printed circuit board was without the npn transistor seen in the figure.



As a basic motion detector the circuit works fine. It is able to reset its self in .3 seconds after being tripped so that there is no need for external inputs. The npn transistor which I added allows me to give the circuit an extra ability. That ability is to work as a sonar system to detect objects while AIRFA is moving. Since when the robot is moving the motion detector is useless since this detector measures the phase shift of the wave, I tied the npn to stop the transmitter from transmitting at 40 kHz. This is done by putting 5 volts to the base of the transistor. Then when I want to measure time of flight I put 0 volts to the base and note the time it takes for a signal to arrive. After watching the characteristics of the motor I found that when the npn is on the receiver output read a value of about 103. And when the detector detects motion the value can range from 41 to 193.

The final sensor I used are three PIR motion detectors. The output a digital signal when they receive a signal. This makes it on of the easier sensors to set up since there was no major rewiring. I did however caluminate the sensors so that there range would be limited. I did this by adding a piece of tubing inside to cut of the peripheral vision. This sensor serve to track a person while the robot is moving since this sensor is not dependent on motion but on heat.

The five sensors that I used help to create the behavior which will govern the operation of the robot AIRFA.

## **Behavior**

There are a series of five behaviors which AIRFA is able to complete. The first behavior is obstacle avoidance through the use of IR. When the signal reached a certain value then the controller knows that there is an object up ahead so that means it is time to turn. The next behavior deals with those objects which are not avoided by the IR. When the bumpers hit an object the controller reacts by shifting the robot into reverse and continuing from there. The next behavior is caused by the cadmium sulfide cells. When a decrease in light is detected the motor slows down and when an increase in light is detected the motor speeds up. The next two behaviors deal with the various motion detectors. The first behavior enables the robot when there is an initial movement signal received from the ultrasonic motion detector. The next one turns the servo to track a person using the PIR. These five behaviors are the basis of the robot abilities which can be expanded and fine tuned at any time.

# Appendix B

## Code for lower microcontroller

```
#include <mil.h>
#include <hc11.h>
#include <motor.h>
#include <analog.h>
#include <serial.h>
// GLOBAL VARIABLES
int a7, flag;
int motor_left, motor_right, motor_speed_left, motor_speed_right;
int light_store;
#define ACC .08

void msleep(int t)
{
    int loop;
    t = t * 100;
    for (loop = 0; loop < t ; loop++)
        asm("NOP");
}
//*****
void init_cont0 (void)
// This funtion initalizes all necessary registers
{
    init_motors();
    init_analog();
    init_serial();
    // set direction of PA7 to output
    a7 = 0;
    SET_BIT(PACTL,0x80);
    CLEAR_BIT(PORTA,0x80);
    // set PD3 and PD2 to output
    SET_BIT(DDRD,0x0c);
    // turn on IR
    SET_BIT(ADDR7,0xff);
    // set inital light value
    light_store = analog(5);
    motor_speed_left = 100;
    motor_speed_right = 100;
}
//*****
void StopMotor()
{
    while ((int) (ACC*motor_left) || (int) (ACC*motor_right))
    {
        motor_left = motor_left - ACC*(motor_left);
        motor_right = motor_right - ACC*(motor_right);
        motor(1-a7,(motor_right - 2*a7*motor_right));
        motor(a7-0,(motor_left - 2*a7*motor_left));
        msleep(10);
    }
    motor_right = 0;
    motor_left = 0;
    motor_speed_right = motor_right;
    motor_speed_left = motor_left;
    motor(1-a7,(motor_right - 2*a7*motor_right));
    motor(a7-0,(motor_left - 2*a7*motor_left));
}
}
```

```

/*****/
void StartMotor()
/*start motor to a %equal to max */
{
    int counter;

    counter = 0;
    while ((int)(ACC*(100 - motor_left)) || (int)(ACC*(100 - motor_right)))
    {
        motor_left = motor_left + (ACC*(100 - motor_left));
        motor_right = motor_right + (ACC*(100 - motor_right));
        motor(1-a7,(motor_right - 2*a7*motor_right));
        motor(a7-0,(motor_left - 2*a7*motor_left));
        msleep(10);
        motor_speed_right = motor_right;
        motor_speed_left = motor_left;
    }
    motor_right = 100;
    motor_left = 100;
    motor_speed_right = motor_right;
    motor_speed_left = motor_left;
    motor(1-a7,(motor_right - 2*a7*motor_right));
    motor(a7-0,(motor_left - 2*a7*motor_left));
}
/*****/
void TurnRight()
{
    motor_right = motor_right + (ACC*(-100 - motor_right));
    motor(1-a7,(motor_right - 2*a7*motor_right));
    msleep(10);
}
/*****/
void TurnLeft()
{
    motor_left = motor_left + (ACC*(-100 - motor_left));
    motor(a7-0,(motor_left - 2*a7*motor_left));
    msleep(10);
}
/*****/
void Turn()
/*choose direction to turn when object encounered */
{
    if (analog(1) <= analog(2))
        while ((int) (ACC*(-100 - motor_left)))
            TurnLeft();
    else
        while ((int) (ACC*(-100 - motor_right)))
            TurnRight();
}
/*****/
void RestoreMotor()
{
    while ((int)(ACC*(motor_speed_right - motor_right)) ||
           ((int) (ACC*(motor_speed_left - motor_left))))
    {
        motor_right += ACC*(motor_speed_right - motor_right);
        motor_left += ACC*(motor_speed_left - motor_left);
        motor(1-a7,(motor_right - 2*a7* motor_right));
    }
}

```



```

        motor(a7-0,(motor_left - 2*a7*motor_left));
        msleep(10);
    }
}
/*****
void Rotate()
{
    int loop;

    motor_speed_left = 100;
    motor_speed_right = -100;
    RestoreMotor();
    for (loop = 0; loop <= 7; loop++)
    {
        msleep(100);
        msleep(100);
        msleep(100);
    }
    motor_speed_left = 100;
    motor_speed_right = 100;
    RestoreMotor();
}
/*****
void Avoid()
{
    flag = 0;
    /* set s1 s0 to 00 */
    CLEAR_BIT(PORTD,0x0c);
    while (analog(0) > 100)
        if (!flag)
        {
            Turn();
            flag = 1;
        }
    /* set s1 s0 to 01 */
    SET_BIT(PORTD,0x04);
    while (analog(0) > 100)
        if (!flag)
        {
            Turn();
            flag = 1;
        }
    /* set s1 s0 to 11 */
    SET_BIT(PORTD,0x0c);
    while (((analog(0) > 100) && (a7 == 0)) ||
           ((analog(0) > 100) && (a7 == 1)))
        if (!flag)
        {
            Turn();
            flag = 1;
        }
    /* set s1 s0 to 10 */
    CLEAR_BIT(PORTD,0x04);
    while (analog(0) > 100)
        if (!flag)
        {

```

```

        Turn();
        flag = 1;
    }
    if (flag)
    {
        flag = 0;
        RestoreMotor();
    }
}
//*****
/* subroutine to act when bumped */
void Bump (void)
{
    if (analog(3) > 20)
    {
        StopMotor();
        if (a7 == 1)
        {
            a7 = 0;
            CLEAR_BIT(PORTA,0x80);
        }
        else
        {
            a7 = 1;
            SET_BIT(PORTA,0x80);
        }
        StartMotor();
        Turn();
        msleep(1000);
        RestoreMotor();
        putchar('F');
    }
}
//*****/
void Light (void)
{
    // uses light to slow down or speed up motors
    float diff;

    diff = .5 * (analog(5) - light_store);
    motor_speed_right = motor_speed_right + (int) diff;
    light_store = analog(5);
    if ((motor_speed_right > 100) || (light_store > 200))
        motor_speed_right = 100;
    if ((light_store < 20) || (motor_speed_right < 50))
        motor_speed_right = 30;
    motor_speed_left = motor_speed_right;

    RestoreMotor();
}
//*****/
void CheckChar() {
    // check for and characters from CONT0
    char c;
    long int loop;

```

## Code for Upper microcontroller

```
#include <mil.h>
#include <hc11.h>
#include <analog.h>
#include <serial.h>
#pragma interrupt_handler servo0
#define PERIOD 40000
#define TOC4_ISR servo0

unsigned width;
int last_hit;
int pir[2][3];
int servo_position;
/*****/
void msleep(int t)
{
    int loop;
    t = t * 100;
    for (loop = 0; loop < t ; loop++)
        asm("NOP");
}
/*****/
void servo0()
{
    if(width == 0)
    {
        CLEAR_BIT(TCTL1, 0x04);
    } else if (TCTL1 & 0x04) {
        TOC4 += width;          /* Keep up for width */
        CLEAR_BIT(TCTL1,0x04); /* Set to turn off */
    } else {
        TOC4 += (PERIOD - width);
        SET_BIT(TCTL1,0x04);    /* Set to low signal */
    }
    CLEAR_FLAG(TFLG1,0x10);    /* Turn off OC4 interrupt */
}
/*****/
void init_cont1 (void)
{
    init_analog();
    init_serial();
    // set direction of PA7 to output
    SET_BIT(PACTL,0x80);
    SET_BIT(PORTA,0x80);
    SET_BIT(TCTL1,0x08);      //italize the OC4
    CLEAR_BIT(TCTL1,0x04);
    SET_BIT(TMSK1,0x10);
    SET_BIT(DDRD,0x04);
    CLEAR_BIT(PORTD,0x04);
    last_hit = analog(4);
    pir[0][0] = 0;
    pir[0][1] = 0;
    pir[0][2] = 0;
    pir[1][0] = 0;
    pir[1][1] = 0;
    pir[1][2] = 0;
    width = 0;
}
```

```

        servo_position = 3;
        INTR_ON();
    }
    /*****/
void Fire()
// fire the ir gun
{
    int loop;

    SET_BIT(PORTD,0x04);
    for (loop = 0; loop < 3; loop ++){
        msleep(100);
        CLEAR_BIT(PORTD,0x04);
    }
    /*****/
void sonar (void)
{
    unsigned short int time1, time2;
    INTR_OFF();
    SET_BIT(PORTA,0x80);
    msleep(10);
    CLEAR_BIT(PORTA,0x80);
    time1 = TCNT;
    while ((analog(0) > 120) && (analog(0) < 70)) {}
    time2 = TCNT;
    time1 = time2 - time1;
    printf("diff = %d\n",time1);
    INTR_ON();
}
    /*****/
void ServoDeg (int degree)
{
    switch (degree)
    {
        case 1: width = 1100;
            break;
        case 2: width = 1800;
            break;
        case 3: width = 2250;
            break;
        case 4: width = 3200;
            break;
        case 5: width = 3900;
            break;
        // case 6: width = 4500;
        // break;
        default: width = 1100;
    } /* endswitch */
}
    /*****/
void DetectMotion ()
//detects the first sign of motion
{
    int data_sent;
    int temp, flag;

    CLEAR_BIT(PORTA,0x80);
    data_sent = 0;
}

```

```

flag = 0;
while (!flag)
{
    while (!(SCSR & 0x20)) {}
    temp = SCSR;
    data_sent = SCDR;
    if ((analog(0) < 20) || (analog(0) > 100))
        flag = 1;
    if (flag)
        putchar(0xe5);
}

}
/*****/
int DetectHit () {
// detect and repond to a hit
    int hit;
    int diff;

    hit = analog(4);
    diff = last_hit - hit;
    msleep(100);
    last_hit = hit;
    if (hit > 95)
        return 1;
    if (diff > 10 ) {
        putchar('A');
        Fire();
    }
    return 0;
}
/*****/
void PirDetect () {
// detect pir motion

    int loop;

    for (loop = 0; loop < 3; loop++) {
        if (analog(loop +1) > 128)
            pir[1][loop]++;
        else {
            if (pir[0][loop] > 5) {
                pir[1][loop] = 0;
                pir[0][loop] = 0;
            }
            else {
                pir[0][loop] ++;
            }
        }
    }

    if (pir[1][1] > 4) {
        Fire();
        pir[1][1] = 0;
    }

    if (pir[1][0] > 4) {

```

```

        putchar ('D');
        pir[1][0] = 0;
        pir[1][2] = 0;
    }
    if (pir[1][2] > 4) {
        putchar ('C');
        pir[1][0] = 0;
        pir[1][2] = 0;
    }
}
/*****/
void RevServo() {
// reverse the direction of head
    if (servo_position == 1) {
        servo_position = 3;
        ServoDeg(servo_position);
    }
    else {
        servo_position = 1;
        ServoDeg(servo_position);
    }
}
/*****/
void CheckChar() {
// check for and characters from CONT0
    char c;

    if (SCSR & 0x20) {
        c = SCDR;
        if (c == 'F')
            RevServo();
        if (c == 'G') {
            RevServo();
            Fire();
            Fire();
            RevServo();
        }
    }
}
/*****/
void main()
{
    int loop;
    int flag;
    char c;

    init_cont1();
    ServoDeg(servo_position);
    while(1)
    {

        CheckChar();
        PirDetect();
        if (DetectHit()) {
            putchar ('B');
            flag = 1;
            while (flag) {
                if (SCSR & 0x20) {

```

```

        c = SCDR;
        if (c == 'H')
            flag = 0;
    }
}

if (SCSR & 0x20) {
    loop = TCNT;
    c = SCDR;
    if (c == 'A') {
        msleep(100);
        Turn();
        msleep(500);
        RestoreMotor();
    }
    if (c == 'B') {
        Rotate();
        putchar('H');
    }
    if (c == 'D') {
        if (loop > 0xf200) {
            while ((int) (ACC*(-100 - motor_left)))
                TurnLeft();
            msleep(500);
            RestoreMotor();
        }
    }
    if (c == 'C') {
        if (loop > 0xf200) {
            while ((int) (ACC*(-100 - motor_right)))
                TurnRight();
            msleep(500);
            RestoreMotor();
        }
    }
}

}

//*****

void main()
{
    long int loop;

    init_cont0();
    StartMotor();
    while (1)
    {
        Avoid();
        CheckChar();
        Bump();
    }
}

```

```
    loop = TCNT;
    if (loop > 0xffff0) {
        putchar('G');
        msleep(100);
    }
}
```



## Code for laser module

```
#include <mil.h>
#include <hc11.h>
#include <analog.h>
#include <serial.h>

#define MAX_SENSORS 4
int hits;

/*****/
void msleep(int t)
{
    int loop;
    t = t * 100;
    for (loop = 0; loop < t ; loop++)
        asm("NOP");
}
/*****/
void init_cont2 (void)
{
    hits = 0;
    init_analog();
    init_serial();
    SET_BIT(PACTL,0x80);
    CLEAR_BIT(PACTL,0x40);
    CLEAR_BIT(PORTA,0x80);
    CLEAR_BIT(PORTB,0xff);
    INTR_ON();
}
/*****/
void Beep ()
// Beed the pezzo for a time
{
    int loop;

    SET_BIT(PORTA,0x80);
    for (loop = 0; loop < 3; loop ++ )
        msleep(100);
    CLEAR_BIT(PORTA,0x80);
}
/*****/
void RegisterHit ()
// Register a hit from the Robot
{
    int loop;

    hits ++;
    for (loop = 0; loop < hits; loop ++ ) {
        Beep();
        msleep(100);
        msleep(100);
        msleep(100);
    }
}
/*****/
void DetectHit ()
//detects the hit from the robot
{
```

```
int loop;

for (loop = 0; loop < MAX_SENSORS; loop++) {
    if (analog(loop) > 102)
        RegisterHit();
}
}
/*****/
void main()
{
    init_cont2();

    while (hits < 3) {
        DetectHit();
        msleep(10);
    }
    while (1)
        Beep();
}
```

**EEL 4914**

**Senior Design**

**AIRFA**

**By:**

**Richardo Fourcand**

**Spring Term 1997**