PUFF: The Autonomous Agent
For EEL 5666
Intelligent Machine Design Lab

Steve McKinnon
Instructor: Keith L. Doty
TA: Scott Jantz
FALL 97

Table of Contents

Abstract

This document is a basic summary of a project completed for EEL 5666: Intelligent Machine Design Lab (IMDL). The project required for this course was the design, testing, and implementation of an autonomous agent. This agent must utilize four basic sensors to perform a variety of behaviors. These requirements are left simple and vague, to promote creativity and allow undergraduates like myself to participate in this unique class.

Executive Summary

For the completion of this project an autonomous agent must be designed and implemented. The agent must have four sensors, and one of these sensors must be of the student's design. Also, the agent must perform at least four different functions.

The project discussed in this document contains one basic robot platform with basic sensor suite. Using this sensor suite the robot will be able to "sense" its environment. However, two algorithms will be used to simulate two different agents. One agent will be a predator who will seek out a prey. On the other hand, there will be a prey who will avoid the predator agent. After both algorithms have been designed and implemented, they will hopefully interact with each other in a similar relationship found in nature.

Introduction

This document contains the information obtained through the design and implementation of the autonomous agent to be developed for IMDL. The scope of this paper is confined to the basic robot platform and the algorithms to be used. The paper discusses the integrated system and the individual sensors used to perceive the agent's environment. This document also discusses the variety of behaviors that were performed with the basic sensor suite. Finally, the paper closes with a summary of work accomplished and discusses the limitations of the work performed. Future work on this project is also mentioned in the conclusion.

Integrated System

The goal of this project was to design an autonomous agent with a variety of functions and behaviors. Specifically, this agent could simulate predator and prey behaviors depending on the arbitrator process imbed in the machine's code. Mekatronix's Talrik Junior was the platform for this robot's design. A M68HC11 microprocessor is the heart of the TJ. The TJ is also has two types of sensors, and the modified design in this project adds two additional sensors. Using all of these sensors, the microprocessor is able to sense some characteristics about its environment and behavior accordingly. The figure below shows the sensors as inputs to the microprocessor which uses this to perform the associated behaviors.

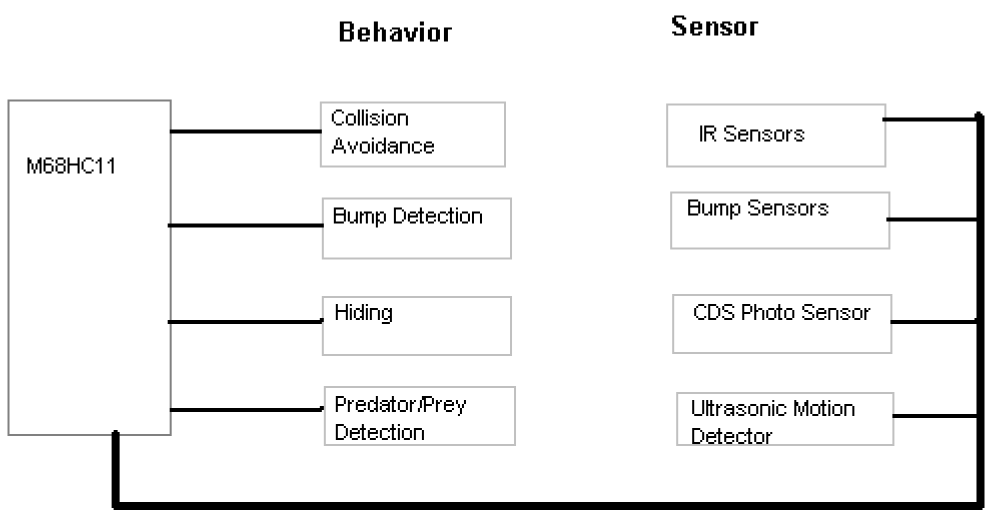


Figure 1: Fuctional Block Diagram of system

Mobile Platform

The platform used is Mekatronix Talrik Jr shown below in Figure 2. This robot has a very compact design making it agile and very maneuverable. In fact, the wheels are placed in a manner allowing the agent to spin on its center. The TJ has a single chip computer board (Novasoft MSCC11), and a battery supply located beneath its topside. For this project the TJ platform was modified with two additional sensors. The modified TJ platform is shown in Figure 3 below.

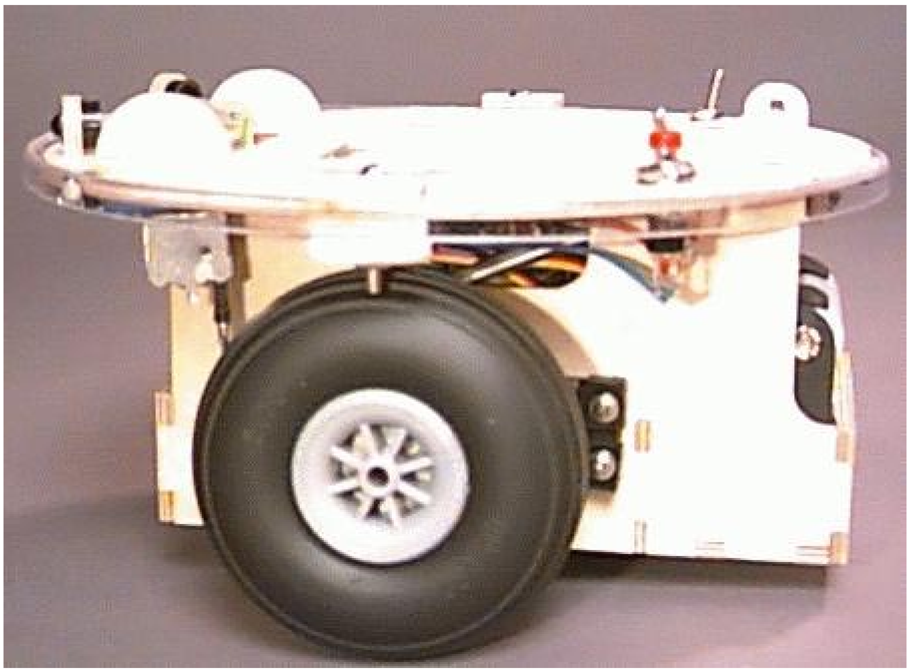


Figure 2b

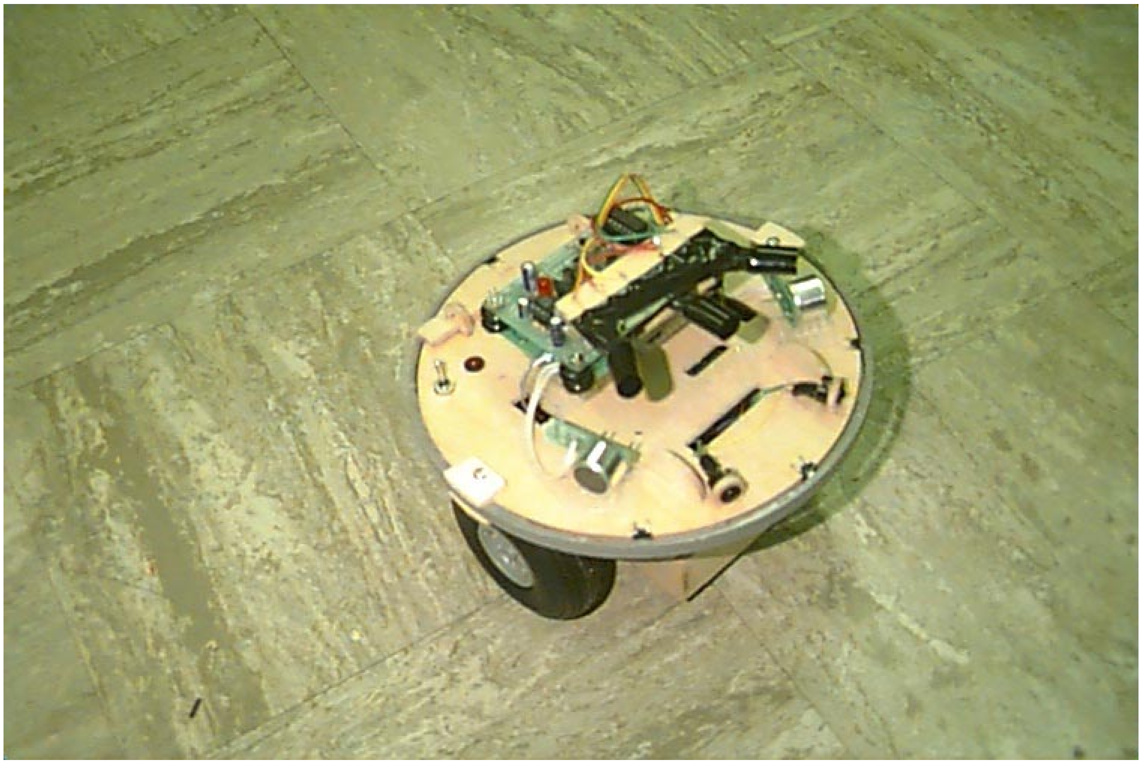


Figure 3: Modified TJ Platform

Actuation

The robot is driven by two independent servo motors.   These servos act essentially as DC motors with feedback circuitry. A 50Hz signal is used to drive the servos in a given direction. Depending on the duty cycle of this signal the servos will drive the wheels at a given speed. The servos are the only type of actuation in the system. The software for the motor drivers can be found in the TJ application package from Novasoft.

Sensors

IR Emitters and Detectors:

The TJ also detect infared light, or IR. Two infared IR emitters are used to project 40kHz IR light. Moreover, there are two IR detectors which can detect this projected light. There are one of these detectors mounted on the right side of the robot and another sensor mounted on the left side. In this arrangement, the TJ can detect which direction the light is coming from. Furthermore, the IR detectors (Sharp GP1U58) are modified to give an analog voltage corresponding to the amount of IR received. The nominal value for these IR detectors is 88 (decimal) on the A/D port and the saturation value is about 130. The left sensor and right sensors are connected to the A/D converter on pins PE6 and PE7 respectively. Software to modulated the IR emitters correctly can be found in the TJ applications package from Novasoft.

Using the analog output from the IR detectors the direction and intensity of incoming light can be measured. If the TJ approaches an object at close range the IR from the emitters is reflected off the object and back to the IR detectors. As a result, an object can be detected and avoided through arbitration of the motors. One of the major drawback to this sensor is its occasional problems sensing its environment. For example, if the object is of the dark in color less light will be reflected. It is almost impossible to determine range without complex algorithms. Moreover, the TJ has problems detecting some shapes such as skinny or round due to lack of reflected light. It is important to leave the TJ in a non-complex environment for reliable interaction.

Bump Detectors:

The bump detectors are the simplest sensor located on the TJ. There are three forward facing bump switches and one on the rear. Each sensor is simple a switch that is either closed or open. The forward looking bump switches are all interconnected with a pull down resistor and then connect to the PC5 of the microprocessor. Therefore, if the switch becomes closed at any time the signal becomes high on the input pin.

A bumper is placed around the outside of the TJ over these bumper switches, so that collisions can be detected from other areas besides the switches themselves. It is important to have a properly placed bumper, so that if a collision does occur the bumper returns to its original position after the TJ has separated itself from the object. This sensor has the most priority over the other sensors in this project. If I collision is detected by the switches the TJ immediately backs up before returning to its original function.

Ultrasonic Motion Detector:

This sensor is a modified version of a electronics kit provided from Jameco Electronics. The part number for this motion detector kit is 125090 and can be purchased by calling the vendor at 1-800-831-

4242. The kit itself contains a matched pair of ceramic transducers which convert movement to electrical energy and vice versa. Through some modification, the basic functionality inherent to these transducers will be used in a way for one robot to detect the direction and range of another robot. Furthermore, this will allow them the robots to interact in a specific algorithm chosen.

JAMECO'S ULTRASONIC MOVEMENT DETECTOR:

The kit provided by Jameco provides two ultrasonic transducers advertised to detect motion from 4-7m away. A crystal provides an operation frequency of 40kHz to the two transducers. The kit also provides a variety of additional characteristics. First, an LED provides visual evidence that movement has been sensed. Second, there is a PCB-mounted switch which can be used to switch between an automatic reset about 0.3 seconds after the detector has been triggered or to stay latched on. Finally, the unit provides pads which allow for additional circuits such as buzzers or lights.

The transmitter circuit consists of a crystal-locked cmos gate oscillator feeding a 40kHz square wave to a cmos driver. This in turn drives the transmitter in anti-phase to get the maximum output. The transmitter circuit is shown below in Figure 4.

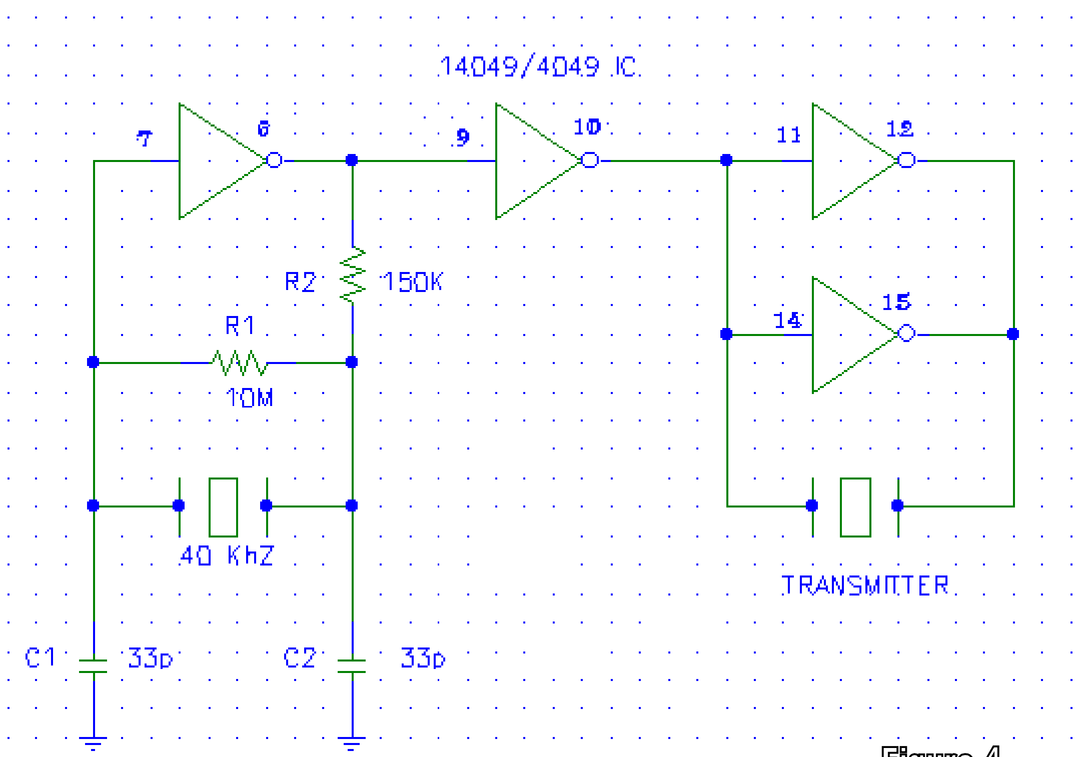


Figure 4

The receiver circuit (Figure 5) takes the electrical output of the transducer and amplifies the signal over a transistor and operational amplifier. Then, the output is converted to DC in a peak detector and taken to the non-inverting input of another operation amplifier. The feedback circuit of this operation amplifier can be adjusted by a potentiometer. This second op amp quickly adjusts to a steady high output if there is no change in the incoming signal.

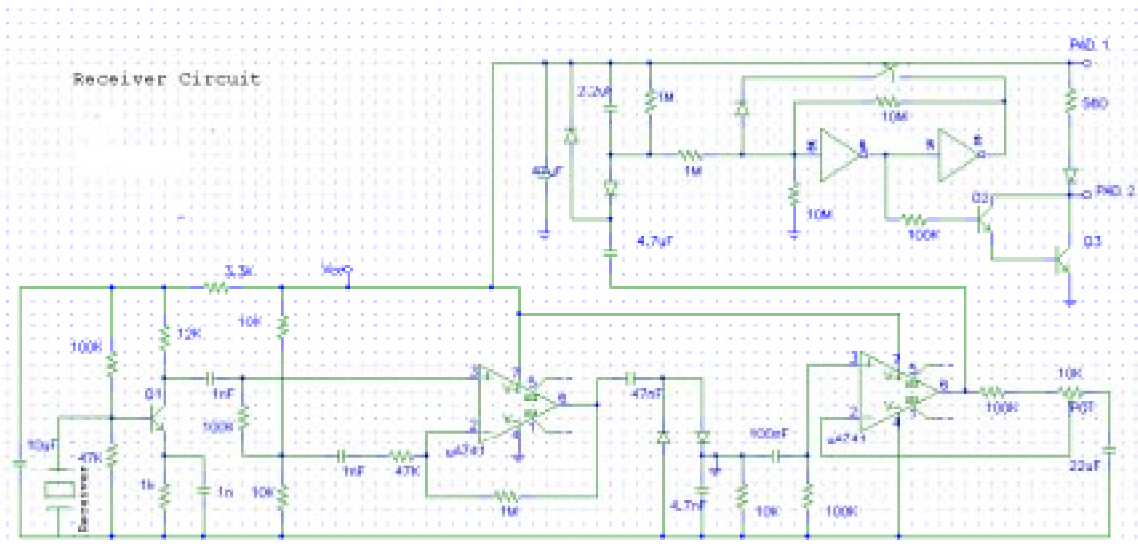


Figure 5: Receiver Circuit

Sound waves reflected by objects arrive at the receiver in different phases. Moreover, if the sound waves are in phase they create a larger signal whereas if the sound waves return out of phase they cancel each other out. Due to this characteristic if an object moves towards or away from the receiver it cause the receiver signal to alternate cycles from high to low. This is the change that triggers the device. They steady high output of the second op amp is pulled down causing the cmos gates to switch.

KIT MODIFICATION AND INDENTED ALGORITHM:

The transmitter and receiver circuits will be separated for use with the autonomous agents. Since, circuitry is common to both the transmitter and receiver some hardware modifications will be made. The predator robot will carry two receiver circuits and will listen for the prey. The receiver pair will be located on different sides of the robot, so that the direction of the signal can be quickly perceived by the agent. The algorithm to sense direction is similar to the IR obstacle avoidance, except that the robot’s arbitrator will turn the motors in the direction of the strongest signal, rather than avoiding this. The predator will "listen" for the signal that the prey will transmit.

PERFORMANCE AND OBSERVATION OF SENSOR CHARACTERISTICS:

An experiment was used to determine some of the characteristics of this ultrasonic motion detector. The Jameco kit provides a LED which lights up every time the device is triggered by movement, and provided an easy method of testing the limitations of the sensor.

IR sensors have a limitation when searching for a reflected signal. For example, a stronger IR signal can be measured off a light surface than a dark surface. However, these ultrasonic transducers do not have this same limitation. Sound waves are not effected by the color of a reflecting surface. However, they are effected by the shape. A concave object will provide a stronger reflected signal than a flat object. Moreover, a flat object will reflect a stronger signal than a convex object.

The receiver circuit's sensitivity can be adjusted by the potentiometer. Varying the sensitivity of the device, a flat and concave object were used to determine the greatest distance that could be detected. The following data was obtained,

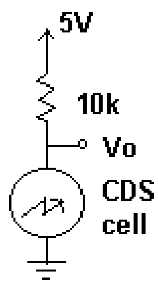
RESISTANCE(OHMS)	DISTANCE OF FLAT OBJECT(FT)	DISTANCE OF OF CONCAVE OBJECT(FT)
100	4	6
2.5K	4.5	6.5
5K	5	7.25
7.5K	6	8
10K	7	9.5

This data shows one of the limitations of the sensor. For if used for collision avoidance, a robot would not easily detect some objects due to shape. Moreover, if this sensor was used to range object error should be expected. The receiver circuit is very sensitive, and would result in greater range errors with greater distances.

This motion detector sensor is the most elaborate sensor found in this project. Modifications must be made from the current design however. Through implementation of this sensor with the robot a problem has been observed. The A/D converter samples at a given rate which is much slower than oscillating signal given from the receiver circuitry. The output waveform would probably be readily usable if a low pass filter was used on the current output waveform. Despite this problem, the TJ can detect motion consistently. Range and directions problems will hopefully be rendered at a later date.

CDS Photo Resistor:

The CDS photo resistor sensor is very simple. The sensor itself is a simple voltage divider as shown in Figure 6. The value of the resistor can be adjusted to give a variety of sensitivities to different types of light. For example, a 1k and 100k resistor force the circuit to be sensitive to bright and dark light respectively.



The voltage over the CDS cell is taken to the A/D converter on the microprocessor. There are three of these sensors placed on the robot, and they all vary in their direction.

Behaviors

Collision Avoidance:

Collision Avoidance is accomplished with the IR system. The IR detectors are connected to the analog port of the microprocessor. When a tolerance is exceeded on one of the detectors the system's arbitrator "senses" an object and sends the proper speed and direction outputs to the motors. For example, if the left IR detector give a value above a certain tolerance the arbitrator tells the robot to turn right, or away from the object.

One of the biggest problems with this behavior is called the "Braitenburg Trap." This situation occurs when both the right and left detectors sense a value above threshold. The system's arbitrator wants to go both direction, so when implemented the robot just seems to shake back and forth. To solve this problem the arbitrator looks to see when this situation happens and then tells the motors to back up for a set amount of time (see source code in ARBIT_SR).

Bump Detection:

This behavior depends on the bump switches located around the outside of the TJ. If they are triggered the arbitrator simply backs up for a set amount of time. In fact, this behavior has the highest priority in the system.

Hiding/Searching:

This behavior can either be seen as hiding or searching depending on if the algorithm being used is that of the prey or predator respectively. In either case, the system uses the CDS cells to search for the darkest part of the room. Once all three CDS sensors register a dark reading, then the robot knows it has reached its destination. In the prey's case, it now hides until a predator robot comes. In contrast, when the predator robot reaches its destination it will poke around searching for the prey who should have left by then.

Predator/Prey Detection:

The ultrasonic motion receivers are used for this behavior. The prey uses this sensor when it is hiding. Once it detects a prey in its vicinity it begins to run away. Similarly, the predator uses this sensor to detect the prey. However, with a better output signal from the motion detector the predator should be able to chase the prey with this same signal.

Similar to the IR system, there are two receivers placed on opposite sides of the robot. This should give the arbitrator some sense of direction. However, due to the fact the receiver circuit is too sensitive or the A/D converter does not sample fast enough this sensor can only be used to detect motion.

Conclusion

The goals of this project have been achieved for the most part. The final robot platform has a great sensor suite which has many possible functions. However, the M68HC11E2 only has 2K EEPROM and 256 bytes of RAM, so the memory capacity is limited. With a bigger memory, the entire sensor suite could be utilized to perform elaborate and more “intelligent” behaviors. Moreover, modification still needs to be done on the ultrasonic sensor. This sensor has some advantages over using IR for collision avoidance or object chasing. If the output of this sensor was passed through a low pass filter the signal may actually be used for a more useful function. Eventually, both the prey and predator algorithms will be completed, and both designs will be implemented together to hopefully simulate some behaviors found in nature.

Appendix

Source code in assembly:

```
*Steve McKinnon
*IMDL
*12/1/97
*shell v1.4 collison avoidance WITH BUMP DETECTION (FINAL)
**********************************************************************
EEPROM          EQU     $B600
RAM             EQU     $0000
RAMEND          EQU     $01FF
REGS            EQU     $1000
OPTION          EQU     $1039
ADCTL           EQU     $1030
ADR1            EQU     $1031
ADR2            EQU     $1032
ADR3            EQU     $1033
ADR4            EQU     $1034
TMSK1           EQU     $1022
TMSK2           EQU     $1024
TFLG2           EQU     $1025
PACTL           EQU     $1026
PORTC           EQU     $1003
*       Constants
NSAMP           EQU     100
BIT0            EQU     %00000001
BIT1            EQU     %00000010
BIT6            EQU     %01000000       ;LEFT SENSOR
BIT7            EQU     %10000000       ;RIGHT SENSOR
INV6            EQU     %10111111
STSIZE          EQU     20

****** DATA ********
        ORG     $00
PID     RMB     1       ;PROCESS ID
SP1     RMB     2       ;STACK POINTER FOR IR READ
SP2     RMB     2       ;STACK POINTER FOR SONIC READ
SP3     RMB     2       ;STACK POINTER FOR PHOTO READ
SP4     RMB     2       ;STACK POINTER FOR ARBITRATOR
SP5     RMB     2       ;STACK POINTER FOR MOTOR ROUTINE
STACK1          RMB     STSIZE
STACK2          RMB     STSIZE
STACK3          RMB     STSIZE
STACK4          RMB     STSIZE
STACK5          RMB     STSIZE
BUMP    RMB     1
IR_LEFT         RMB     1
IR_RIGHT RMB 1
SONAR_LEFT RMB          1
SONAR_RIGHT RMB         1
CDS_LEFT RMB            1
CDS_CENTER RMB          1
CDS_RIGHT RMB           1
MOTOR_LEFT RMB          2
```

```
MOTOR_RIGHT RMB     2
ACTUAL_LEFT RMB 2
ACTUAL_RIGHT RMB 2
_SERVOS_BUSY:
        RMB     2
_SIGNAL_STATE:
        RMB     2
_CURRENT_WIDTH:
        RMB     2
_WIDTH:
        RMB     4
_current_mode
        rmb   1
****** INTERRUPT VECTORS *******
        ORG     $FFFE
        FDB     MAIN
        ORG     $FFE4
        fdb     _IRREAD
        ORG     $FFE6
        FDB     _SERVO_HAND
        ORG     $FFF0
        FDB     RTI_ISR
***********************************************************************8
        ORG     $F800
MAIN    LDX     #REGS
*Turn on A/D System
                LDAA    #%10000000              ; Bit 7 (ADPU) of OPTION
                STAA    OPTION-REGS,X
* Wait 100 ms for Pump Charge to Stabilize
        LDAA    #40                             ; 2 E cycles
WAIT1   DECA                                    ; 2 E cycles
        BNE     WAIT1                           ; 3 E cycles
*
        BCLR    PACTL,X BIT1    ;SET INTERRUPT RATE FOR 8.19MSEC...
        BSET    PACTL,X BIT0
        BSET    TMSK2,X BIT6    ;ENABLE RTI INTERRUPT
*
        LDX     #STACK1+STSIZE-1        ;INITIALIZE STACK POINTERS FOR PROCESSES
        STX     SP1
        LDX     #STACK2+STSIZE-1
        STX     SP2
        LDX     #STACK3+STSIZE-1
        STX     SP3
        LDX     #STACK4+STSIZE-1
        STX     SP4
        LDX     #STACK5+STSIZE-1
        STX     SP5
*                                       ;INITIALIZE ALL STACKS
        LDS     SP5             ;ASSUME PREVIOUS PID WAS 5
        LDX     #MOTOR_SR
        PSHX
        LDY     #0
        PSHY                    ;IY
        PSHY                    ;IX
        PSHY                    ;A AND B
        LDAA    #BIT6           ;CCR WITH X=1 AND I=0
```

```
        PSHA                    ;CCR
        STS     SP5
*
        LDS     SP4             ;ASSUME PREVIOUS PID WAS 4
        LDX     #ARBIT_SR
        PSHX
        LDY     #0
        PSHY                    ;IY
        PSHY                    ;IX
        PSHY                    ;A AND B
        LDAA    #BIT6           ;CCR WITH X=1 AND I=0
        PSHA                    ;CCR
        STS     SP4
*
        LDS     SP3             ;ASSUME PREVIOUS PID WAS 3
        LDX     #PHOTO_SR
        PSHX
        LDY     #0
        PSHY                    ;IY
        PSHY                    ;IX
        PSHY                    ;A AND B
        LDAA    #BIT6           ;CCR WITH X=1 AND I=0
        PSHA                    ;CCR
        STS     SP3
*
        LDS     SP2             ;ASSUME PREVIOUS PID WAS 2
        LDX     #SONIC_SR
        PSHX
        LDY     #0
        PSHY                    ;IY
        PSHY                    ;IX
        PSHY                    ;A AND B
        LDAA    #BIT6           ;CCR WITH X=1 AND I=0
        PSHA                    ;CCR
        STS     SP2
*
        LDAA    #1
        STAA    PID
        LDS     SP1             ;INITIALIZE CURRENT PROCESS AS 1
*
        LDY     #3000
        STY     ACTUAL_LEFT ;INITALIZE CURRENT MOTOR WIDTH
        STY     ACTUAL_RIGHT
        JSR     _INIT_SERVOS ;INTIALIZE SERVOS
        JSR     _INIT_IR        ;INTIALIZE IR EMITTERS
        JSR     _IR_ON          ;TURN ON IR
        CLI                     ;TURN ON INTERRUPTS
        JMP     IR_SR           ;GO TO PROCESS 1
****************************************
************************************
RTI_ISR LDX     #REGS
        BRCLR TFLG2,X BIT6 ENDRTI  ;IGNORE ILLEGAL INTERRUPT
        BCLR    TFLG2,X INV6       ;CLEAR FLAG
        LDAA    PID                ;FIND CURRENT PID
        CMPA    #1
        BEQ     SAVEP1
```

```
        CMPA  #2
        BEQ   SAVEP2
        CMPA  #3
        BEQ   SAVEP3
        CMPA  #4
        BEQ   SAVEP4
        CMPA  #5
        BEQ   SAVEP5
        BRA   ENDRTI                    ;IGNORE ILLEGAL PID
SAVEP1        STS    SP1                ;SET NEXT PROCESS
        LDAA  #2
        STAA  PID
        LDS   SP2
        BRA   ENDRTI
SAVEP2        STS    SP2
        LDAA  #3
        STAA  PID
        LDS   SP3
        BRA   ENDRTI
SAVEP3        STS    SP3
        LDAA  #4
        STAA  PID
        LDS   SP4
        BRA   ENDRTI
SAVEP4        STS    SP4
        LDAA  #5
        STAA  PID
        LDS   SP5
        BRA   ENDRTI
SAVEP5        STS    SP5
        LDAA  #1
        STAA  PID
        LDS   SP1
        BRA   ENDRTI
ENDRTI        RTI
************************************
****************************************
IR_SR   BRA   IR_SR
        LDX   #REGS
LOOP1   LDAA  #%00010110    ;SAMPL
        STAA  ADCTL-REGS,X
        LDAA  #6             ; 2 E cycles
WAITR   DECA                 ; 2 E cycles
        BNE   WAITR          ; 3 E cycles
        LDAA  ADR3
        STAA  IR_LEFT
        LDAA  #%00010111    ;SAMPR
        STAA  ADCTL-REGS,X
        LDAA  #6             ; 2 E cycles
WAITL   DECA                 ; 2 E cycles
        BNE   WAITL          ; 3 E cycles
        LDAA  ADR4
        STAA  IR_RIGHT
        BRA   LOOP1
****************************************
SONIC_SR LDX #REGS
```

```
LOOP2  SEI
       LDAA  #%00010001      ;SAMPL
       STAA  ADCTL-REGS,X
       LDAA  #6              ; 2 E cycles
WAITR  DECA                  ; 2 E cycles
       BNE   WAITR           ; 3 E cycles
       LDAA  ADR1
       STAA  SONAR_LEFT
       LDAA  #%00010101      ;SAMPR
       STAA  ADCTL-REGS,X
       LDAA  #6              ; 2 E cycles
WAITL  DECA                  ; 2 E cycles
       BNE   WAITL           ; 3 E cycles
       LDAA  ADR2
       STAA  SONAR_RIGHT
       CLI
       BRA   LOOP2
***************************************
PHOTO_SR LDX        #REGS
LOOP3  BRA   LOOP3
***************************************
ARBIT_SR LDX #REGS
       JSR   LOOP4
       BRA   ARBIT_SR
LOOP4  LDAA  PORTC
       ANDA  #%00100000      ;CHECK PC5
       BNE   GOBACK                ;BUMP DETECTION IF NOT ZERO
       LDAA  CDS_RIGHT
       CMPA  #253
       BLT   STOPQU                ;IF CDSRIGHT DARK CHECK LEFT
       BRA   SKIP            ;ELSE SKIP
STOPQU       LDAA  CDS_LEFT
       CMPA  #253
       BLT   MSTSTOP               ;IF CDSLEFT DARK TOO THEN STOP
       BRA   SKIP            ;ELSE SKIP
MSTSTOP      JSR   STOP
       RTS
SKIP   LDAA  IR_LEFT               ;COLLISION AVOIDANCE
       LDAB  IR_RIGHT
       CMPB  #98
       BLT   RLOW
       CMPA  #98
       BLT   GOLEFT
       LDAA  IR_LEFT
       LDAB  IR_RIGHT
       CMPB  #250
       BGT   RHIGH
       CMPA  #250
       BGT   GOLEFT
*
       LDAA  IR_LEFT
       CMPA  IR_RIGHT
       BLT   GOLEFT
       BRA   GORIGHT
*
RHIGH  CMPA  #250
```

```
        BGT     GOBACK
        BRA     GORIGHT
*
RLOW    CMPA    #98
        BLT     FORWARD
        BRA     GORIGHT
GOBACK          LDY     #3500
        STY     MOTOR_LEFT
        LDY     #2400
        STY     MOTOR_RIGHT
        JSR     DELAY
        RTS
GOLEFT          LDY     #3500           ;WORKS
        STY     MOTOR_LEFT
        LDY     #3500
        STY     MOTOR_RIGHT
        RTS
LILLEFT         LDAA    CDS_LEFT
        SUBA    CDS_RIGHT
        CMPA    #150
*       BPL     STR
        LDY     #3000           ;WORKS
        STY     MOTOR_LEFT
        LDY     #3500
        STY     MOTOR_RIGHT
        RTS
GORIGHT         LDY     #2300
        STY     MOTOR_LEFT
        LDY     #2400
        STY     MOTOR_RIGHT
        RTS
LILRIGHT LDAA           CDS_RIGHT
        SUBA    CDS_LEFT
        CMPA    #100
*       BLT     STR
        LDY     #2300
        STY     MOTOR_LEFT
        LDY     #3000
        STY     MOTOR_RIGHT
        RTS
FORWARD         LDAA    CDS_CENTER      ;NO COLLSION, SO FIND DARKNESS
        CMPA    #250
        BLT     STR             ;GOSTRAIGHT IF DARK AHEAD
        LDAA    CDS_LEFT
        SUBA    CDS_RIGHT
        BLT     LILRIGHT        ;LEAN RIGHT
        BRA     LILLEFT                 ;LEAN LEFT
STR     LDY     #2300           ;STRAIGHT MOTOR VALUES
        STY     MOTOR_LEFT
        LDY     #3500
        STY     MOTOR_RIGHT
        RTS
STOP    LDY     #3000           ;STOP ROUTINE
        STY     MOTOR_LEFT
        STY     MOTOR_RIGHT
WAIT4   LDAA    SONAR_LEFT      ;WAIT FOR PREDATOR
```

```
        CMPA  #180
        BHI   DONE4         ;DETECT WITH LEFT...RUN!
        LDAA  SONAR_RIGHT
        CMPA  #180
        BHI   DONE4         ;DETECT WITH RIGHT...RUN!
        BRA   WAIT4
DONE4   LDY   #3500         ;BACK UP OUT OF HIDING AND GO...
        STY   MOTOR_LEFT
        LDY   #2800
        STY   MOTOR_RIGHT
        JSR   DELAY         ;WAIT TO FINISH BACKING UP...
        RTS
*************************************
MOTOR_SR
        LDD   ACTUAL_LEFT
        CPD   MOTOR_LEFT
        BLT   INCL
        BEQ   FIXLEFT
        LDD   ACTUAL_LEFT
        SUBD  #50
        STD   ACTUAL_LEFT
        BRA   FIXLEFT
INCL    LDD   ACTUAL_LEFT
        ADDD  #50
        STD   ACTUAL_LEFT
FIXLEFT       LDD   #0
        LDX   ACTUAL_LEFT
        PSHX
        JSR   _SERVO              ;LEFT
        PULX
*
        LDD   ACTUAL_RIGHT
        CPD   MOTOR_RIGHT
        BLT   INCR
        BEQ   FIXR          ;DO NOTHING
        LDD   ACTUAL_RIGHT
        SUBD  #50
        STD   ACTUAL_RIGHT
        BRA   FIXR
INCR    LDD   ACTUAL_RIGHT
        ADDD  #50
        STD   ACTUAL_RIGHT
FIXR    LDD   #1
        LDX   ACTUAL_RIGHT
        PSHX
        JSR   _SERVO              ;RIGHT
        PULX
        BRA   MOTOR_SR
*************************************
DELAY   LDX   #5            ;APPROX = 2 SEC
OUTER   LDY   #10000        ;4 CYCLES
INNER   CMPA  $00           ;3
        DEY                 ;4
        BNE   INNER         ;3
        DEX                 ;3
        BNE   OUTER         ;3
```

RTS
**Source Code for IR modulation and to drive the servos can be found in the TJ applications software from Novasoft.