University of Florida
Department of Electrical and Computer Engineering

EEL 5666
Intelligent Machines Design Laboratory

Fall 1996

"Fizgig"

Elizabeth Thiel

Prof. Keith L. Doty

December 15, 1996

Table of Contents

Abstract

The goal of this project was to design an autonomous prey robot capable of avoiding collisions, detecting collisions, detecting motion, and seeking out dark areas to hide. The robot also had to be small enough to be able to get captured by a larger predator robot. This paper details the design of the robot platform, the sensors used, and the behaviors created using the sensors and programming. It discusses each sensor type and the circuits needed to connect them to the robot.
The behaviors are integrated into one working system using programming code in C language (shown in the appendix).

Executive Summary

This project entailed the design and construction of an autonomous, mobile robot, capable of reacting to its environment.

The first stage of the project involved developing a collision avoidance behavior. The prey robot Fizgig uses IR emitters and detectors to “sense” objects and react accordingly. The sensors detect oncoming objects and the collision avoidance program tells the robot which way to turn to get away from them.

The second stage involved adding a bumper system. This bumper assists in object detection. If the robot is incapable of reacting quickly enough through IR, the bumper created by a series of switches is triggered, and the robot backs up and turns away.

The third stage was concerned with motion detection. Fizgig uses a pyrosensor to detect humans and robots. Since this is a prey robot, when it senses something it jerks back and tries to run away.

The last stage of behavior development involved getting the robot to hide. When it was running away, it would look for a dark area using collimated Cadmium Sulfide photocells. When it senses darkness, it thinks it is hidden from view and it stops for a few seconds. Then, it thinks the coast is clear and it proceeds to move around as usual.

Finally, the last stage involved integrating all the behaviors so it could move around and react accordingly.

Introduction

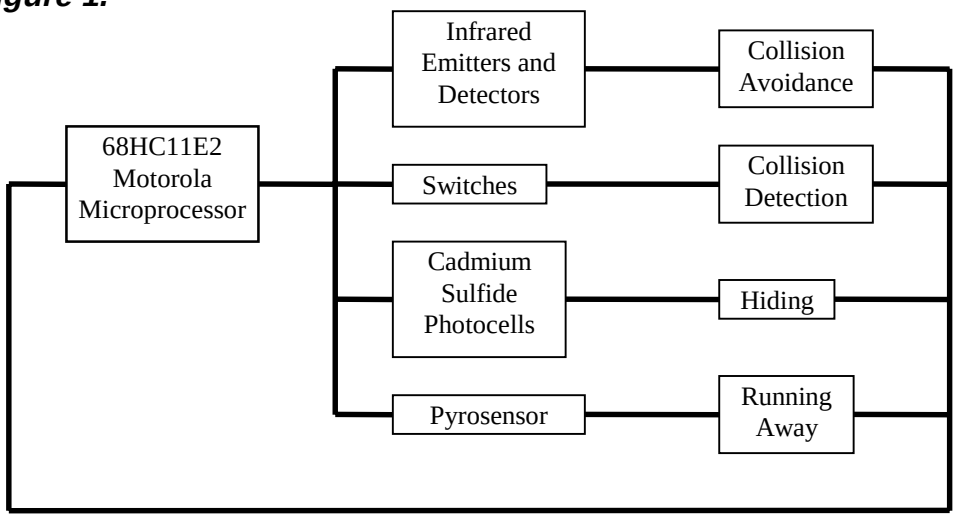
The goal of this project was to design an autonomous mobile prey robot. The robot must be capable of avoiding obstacles, detecting the predator robot, and hiding from the predator after it has been detected.

The robot “Fizgig” uses a single-chip board to control all of its sensors and behaviors. The size of the board allows the design of the robot to be small, making it an ideal prey. The predator is much larger with a claw mechanism that is the size of the prey. This allows it to trap the prey. The prey is a little faster than the predator, giving it one advantage to get away from the predator.

Integrated System

The sensors and behaviors of the robot are all controlled by the microprocessor. The following diagram shows the different types of sensors, the behaviors that use them, and their integration into a system controlled by the microprocessor.

Figure 1.

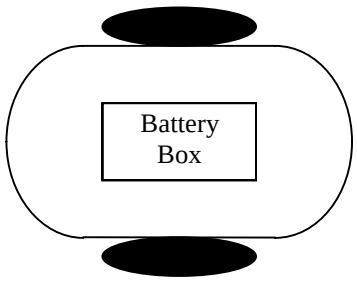


The diagram shows that there is a different type of sensor for each behavior the robot exhibits. All the sensors are controlled by the microprocessor and the behaviors are determined by the programs written in C.

Mobile Platform

The design of the platform is a very simple design. It is a 5" diameter circle with an inch cut off on opposite sides to make the wheels fit closer to the body. This also makes the robot more compact and circular. The back has a caster to give it balance. There is a small box built on the top of the platform that holds the battery pack, and the microprocessor sits on the lid of the box. Figure 2 shows the platform layout with the battery box. There is also a small second platform which holds all the infrared emitters.

Figure 2.



Robot
Platform

(top view)

Actuation

The robot uses a single-chip board by Novasoft Mekatronics© designed for the Motorola MC68HC11E2 chip. The only actuation is the two wheels of the robot. The wheels' speed and direction are controlled by two Aristo-craft Tracker

D3-410 servo motors. They were hacked in a method discovered by Scott Jantz and Ivan Zapata. This hack removes a piece which generally moves the setting of the potentiometer. The potentiometer is then manually set at the middle resistance where the motor stops turning. This can be done using the program MOTION.C written by Ivan Zapata. Also, one of the motors was connected in reverse. This was just a coincidence, but proved to be extremely helpful in programming. Both motors could be given the same commands to go forward or reverse. Usually, they have to be given opposing commands in order for the robot to go forward or reverse.

Sensors

Fizgig uses four types of sensors: infrared emitters and detectors, switches, Cadmium Sulfide photocells, and a pyrosensor (heat sensor).

Fizgig uses two Sharp 32.75kHz IR detectors on the bottom of the platform for collision avoidance. There are eight emitters arranged in a circle on the second platform above the microprocessor. This creates a beacon which will allow the predator to track the prey.

There are three switches located on the front edge of the platform that serve as a bumper. They are connected in series and the robot is programmed to backup and turn when either one of them is closed.

Fizgig uses a motion detector purchased at Lowe's. I hacked a Motion Activated Twin Floodlight Kit Model MS35 by Regent Lighting Corporation. I removed the AC to DC transformer and associated circuitry so that it could be connected to the board. I determined where the power, ground, and output

were, and connected them to the board. The output was connected to an analog port.

Finally, the robot also has two Cadmium Sulfide photocells located on the right and left front of the platform. They are connected using a simple voltage divider circuit with a 15kΩ resistor each.

Behaviors

Fizgig exhibits four distinct behaviors: collision avoidance, collision detection, predator detection, and hiding.

Collision Avoidance

The collision avoidance behavior uses the IR emitters and detectors to avoid objects. When the robot detects an object or wall, it is programmed to react accordingly. The program GIMPY3.C (found in the appendix) is for collision avoidance. The program also allows the emitters to be modulated for 32.75kHz instead of the usual 40kHz. This program was written by Ivan Zapata and proved useful for my project. The predator robot uses 40kHz IR, but tracks 32.75kHz IR. This allows the predator's collision avoidance not to be confused with the prey, and to keep the tracking system completely independent.

Collision Detection (Bumper)

Included in the program GIMPY3.C is the bumper programming. The program simply tells the robot to backup and turn whenever either one of the three switches is closed.

Predator Detection (Run Away)

Experimentation allowed me to determine the approximate threshold of the pyrosensor and write the program RUN.C (in the appendix). When there was no motion, the sensor gave an analog reading of about 110. The program causes the robot to jerk back whenever it detects anything above 106. Then, it turns "nervously" trying to get away.

Hide

The program HIDE.C compares the analog returns of the two Cadmium Sulfide photocells and tells Fizgig to go in the direction of the lowest reading. The robot is trying to seek out a dark area in the room in order to hide. When the two readings are below a certain threshold the robot thinks it is in a hiding spot and stops. It waits for a few seconds and then proceeds to move around again.

FIZGIG.C is the program in the appendix which integrates all of the behaviors.

The robot moves around avoiding objects and using the bumper when necessary. If it detects any motion, human or robotic, it jerks back and turns looking for a place to hide. When it finds the dark spot, it stops for a few seconds. Then, it comes out of hide mode and continues using collision avoidance and the bumper until it senses motion again.

Experimental Layout and Results

When I was trying out the HIDE.C program I had to change the thresholds several times in order to get it to work properly. Sometimes it would react to

anything, including walls. It was difficult to get it right because the robot is moving, therefore, everything around it is also in motion.

The program FIZGIG.C worked really well. The behaviors were integrated and the robot reacts accordingly. I can easily adjust the sensitivity of the robot in the program if necessary.

Conclusion

The goal of this project was to create an autonomous prey robot capable of detecting a predator robot, hiding and avoiding obstacles. The project worked really well without requiring complicated circuitry.

Creating predator/prey environments can allow us to study the effects of chemicals and other harsh changes on the environment. This is only a simple predator/prey project, but we can design more complicated systems as needed.

Documentation

[1] Fred Martin, *The 6.270 Robot Builder's Guide*, MIT Media Lab, Cambridge, MA, 1992.

[2] Joseph Jones & Anita Flynn, *Mobile Robots: Inspiration to Implementation*, A.K. Peters Publishers, Wellesley, MA, 1993.

[3] Pattie Maes, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT Press, Cambridge, MA, 1990.

Appendix

The program GIMPY3.C is for collision avoidance and the bumper system.

```
/*
  GIMPY.C
  11.28.96
  This program was written for a Talrik Junior platform in order to
  achieve collision avoidance. Ivan Zapata
  Modified for Fizgig by Elizabeth Thiel.
*/

#include <servotj.h>
#include <hc11.h>
#include <mil.h>
#include <irtj.h>
#include <analog.h>
#include <vectors.h>


#define ZEROL  3000   /* Servo signals */
#define ZEROR  3000
#define FORWL  2000
#define FORWR  2000
#define BACKL  3500
#define BACKR  3500
#define RVOFF 6148
#define LVOFF -498
#define SENSORMIN 85
#define SENSORMAX 118
#define LEFT_SERVO 0
#define RIGHT_SERVO 1



void turn(void);


int main(void)
{
  int rval, lval, rv, lv;
  int motion;
  int scared;
  int i;

  scared=0;

  init_servos();
  init_analog();
  init_ir();
```

```
  ir_mode(3);

  DDRC = 0x02;          /* PORTC is input except for bit 2*/


  while(1)
  {

/* servo0 = left, servo1 = right */
/* PE6 is on the right, PE7 is on the left */



    rval = ir_value(6);             /* Get sensor readings */
    lval = ir_value(7);

    if (rval < SENSORMIN) rval = SENSORMIN;
    if (lval < SENSORMIN) lval = SENSORMIN;

    rv = (rval << 5) + LVOFF;    /* Calculate indexes */
    lv = (lval << 5) + LVOFF;

    if (scared==1)
     {


    servo(RIGHT_SERVO, rv);           /* Set actual servo speed */
    servo(LEFT_SERVO, lv);


    if(PORTC & 0x20)
    {
      servo(LEFT_SERVO, BACKL);
      servo(RIGHT_SERVO, BACKR);
      for(i = 0; i < 25000; i++);
      turn();
    }

    motion = analog(2);             /* Get sensor readings */

    if ((motion < 108) && (scared==0))
      {
       scared=1;
       servo(RIGHT_SERVO, 4000);
       servo(LEFT_SERVO, 4000);
       for (i=0;i<25000;i++);
       servo(RIGHT_SERVO,2000);
       servo(LEFT_SERVO,4000);
       for (i=0;i<25000;i++);
```

```
      servo(RIGHT_SERVO, 2000);
      servo(LEFT_SERVO, 2000);
     }


  for(i = 1; i < 2500; i++);

  }
}



void turn()
{
 int i;
 unsigned rand;

 rand = TCNT;

 if (rand & 0x0001)
 {
   servo(RIGHT_SERVO, FORWR);
   servo(LEFT_SERVO, BACKL);
 }
 else
 {
   servo(RIGHT_SERVO, BACKR);
   servo(LEFT_SERVO, FORWL);
 }
 for (i = 0; i < rand; i++);
}
```

The program RUN.C is for motion detection.

```
/*
  RUN.C
  12.11.96
  This program was written for Fizgig in order to
  detect the predator.
 */

#include <servotj.h>
#include <hc11.h>
#include <mil.h>
#include <irtj.h>
#include <analog.h>
#include <vectors.h>


#define ZEROL  3000   /* Servo signals */
#define ZEROR  3000
#define FORWL  2000
#define FORWR  2000
#define BACKL  4000
#define BACKR  4000
#define RVOFF 6148
#define LVOFF -498
#define SENSORMIN 85
#define SENSORMAX 118
#define LEFT_SERVO 0
#define RIGHT_SERVO 1



void turn(void);


int main(void)
{
  int rval, lval, rv, lv;
  int motion;
  int i;


  init_servos();
  init_analog();

    servo(RIGHT_SERVO,2000);
```

```
    servo(LEFT_SERVO,2000);

  while(1)
  {

/* servo0 = left, servo1 = right */
/* PE6 is on the right, PE7 is on the left */
/* Right  = 3  Left =1 */



    motion = analog(2);             /* Get sensor readings */

    if (motion <106)
      {
       servo(RIGHT_SERVO, 4000);
       servo(LEFT_SERVO, 4000);
       for (i=0;i<25000;i++);
       servo(RIGHT_SERVO,2000);
       servo(LEFT_SERVO,4000);
       for (i=0;i<25000;i++);
       servo(RIGHT_SERVO, 2000);
       servo(LEFT_SERVO, 2000);
      }

    for(i = 1; i < 2500; i++);

  }
}
```

The program HIDE.C is to run and hide in a dark area.

```
/*
  HIDE.C
  12.10.96
  This program was written for Fizgig to be
  able to hide from the predator Pantera.
 */

#include <servotj.h>
#include <hc11.h>
#include <mil.h>
#include <irtj.h>
#include <analog.h>
#include <vectors.h>


#define ZEROL  3000   /* Servo signals */
#define ZEROR  3000
#define FORWL  2000
#define FORWR  2000
#define BACKL  4000
#define BACKR  4000
#define RVOFF 6148
#define LVOFF -498
#define SENSORMIN 85
#define SENSORMAX 118
#define LEFT_SERVO 0
#define RIGHT_SERVO 1



void turn(void);


int main(void)
{
  int rval, lval, rv, lv;

  int i;


  init_servos();
  init_analog();

    servo(RIGHT_SERVO,2000);
    servo(LEFT_SERVO,2000);
```

```
  while(1)
  {

/* servo0 = left, servo1 = right */
/* PE6 is on the right, PE7 is on the left */
/* Right  = 3  Left =1 */



    rval = analog(1);              /* Get sensor readings */
    lval = analog(3);


    if (rval > lval)
      {
       servo(RIGHT_SERVO, 2000);
       servo(LEFT_SERVO, 2800);
      }
    if (lval > rval)
      {
       servo(RIGHT_SERVO,2800);
       servo(LEFT_SERVO,2000);
      }
    if ((lval < 60)&&(rval < 60))
      {
       servo(RIGHT_SERVO,3000);
       servo(LEFT_SERVO,3000);
      }

    for(i = 1; i < 2500; i++);

  }
}
```

The program FIZGIG.C integrates the behaviors: collision avoidance, bumper, motion detection and hide mode.

```
/*
  FIZGIG.C
  12.12.96
  This program integrates the collision avoidance, bumper,
  run and hide for the prey robot Fizgig.
*/

#include <servotj.h>
#include <hc11.h>
#include <mil.h>
#include <irtj.h>
#include <analog.h>
#include <vectors.h>


#define ZEROL  3000   /* Servo signals */
#define ZEROR  3000
#define FORWL  2000
#define FORWR  2000
#define BACKL  3500
#define BACKR  3500
#define RVOFF 6148
#define LVOFF -498
#define SENSORMIN 85
#define SENSORMAX 118
#define LEFT_SERVO 0
#define RIGHT_SERVO 1



void turn(void);


int main(void)
{
  int rval, lval, rv, lv,reye,leye;
  int motion;
  int scared;
  int i;
  int j;

  scared=0;

  init_servos();
  init_analog();
  init_ir();
```

```
  ir_mode(3);

  DDRC = 0x02;          /* PORTC is input except for bit 2*/


  while(1)
  {

/* servo0 = left, servo1 = right */
/* PE6 is on the right, PE7 is on the left */



    rval = ir_value(6);             /* Get sensor readings */
    lval = ir_value(7);

    if (rval < SENSORMIN) rval = SENSORMIN;
    if (lval < SENSORMIN) lval = SENSORMIN;

    rv = (rval << 5) + LVOFF;    /* Calculate indexes */
    lv = (lval << 5) + LVOFF;

    if (scared==1)
     {
      reye = analog(1);             /* Get sensor readings */
      leye = analog(3);


    if (reye > leye)
      {
       lv = (lv + 500);
       }
    if (leye > reye)
      {
       rv = (rv + 500);
      }
    if ((leye < 60)&&(reye < 60))
      {
       servo(RIGHT_SERVO,3000);
       servo(LEFT_SERVO,3000);
       for (i=0;i<5;i++)
       for (j=0;j<25000;j++);
       scared = 0;

      }
      }


    servo(RIGHT_SERVO, rv);          /* Set actual servo speed */
```

```
    servo(LEFT_SERVO, lv);


    if(PORTC & 0x20)
    {
      servo(LEFT_SERVO, BACKL);
      servo(RIGHT_SERVO, BACKR);
      for(i = 0; i < 25000; i++);
      turn();
    }

    motion = analog(2);             /* Get sensor readings */

    if ((motion < 105) && (scared==0))
      {
       scared=1;
       servo(RIGHT_SERVO, 4000);
       servo(LEFT_SERVO, 4000);
       for (i=0;i<25000;i++);
       servo(RIGHT_SERVO,2000);
       servo(LEFT_SERVO,4000);
       for (i=0;i<25000;i++);
       servo(RIGHT_SERVO, 2000);
       servo(LEFT_SERVO, 2000);
      }


   for(i = 1; i < 2500; i++);

   }
}



void turn()
{
 int i;
 unsigned rand;

 rand = TCNT;

 if (rand & 0x0001)
 {
   servo(RIGHT_SERVO, FORWR);
   servo(LEFT_SERVO, BACKL);
 }
 else
 {
   servo(RIGHT_SERVO, BACKR);
   servo(LEFT_SERVO, FORWL);
```

```
 }
 for (i = 0; i < rand; i++);
}
```