TABLE OF CONTENTS:

ACKNOWLEDGEMENTS:

This project was based solely on a previously designed mechanical platform. Actual copy was made form the robot “Bob” created by Steve Stancliff, whose platform was based on the robot “THING”, created by Willad MacDonald.

The servo code for actuation was written by Drew Bagnell. This code was used to control the 12 servos, and was in turn controlled by an EVBU in combination with its daughter board the ME11. I also utilized code written by Billy Eno to allow the two processors to communicate.

The project would not have been possible without the help of all of the IMDL TA’s. Their patients and knowledge was appreciated.

ABSTRACT:

The purpose of this project was to build a walking platform with behaviors similar to those of a dog. Included in this task was: building and testing the hardware necessary, and writing the code necessary to incorporate walking and other behaviors. Four legs and three degrees of freedom per leg was chosen for a more stable platform with a wide range of possible movements. Four legs were chosen as a compromise between hardware complexity, when more legs are used, and software complexity, when less legs are utilized. The robot, at the time of this paper, has the ability to walk forward, backward and to turn to the left, when an object is detected by one of its two IR detector/emitter pairs. Other behaviors and sensors were not implemented on the robot due to time and other constraints.

EXECUTIVE SUMMARY:

The basic design used for ASTRO was copied from a second generation robot named “BOB”, created by Steve Stancliff, which in turn was somewhat based on the robot named “THING”, created by Willad MacDonald. ASTRO was meant to have the behaviors of a canine pet, or a dog. At the time of writing this paper, ASTRO has the ability to walk (forward, backward, and to the left) over flat and semi rough/smooth surfaces with minimal object avoidance. Other proposed behaviors that were not implemented were: responding to voice commands, via a microphone sensor (the circuit has been built but not implemented), responding to body heat and assuming a wandering behavior when none of the above are being detected.

Most of the work during the semester was spent of the mechanical design. This was not the intention, since the design was theoretically being copied from a previously designed platform. The lack of Cad files provided with the previous design proved to be quite a set back. The initial body design was much larger than it needed to be and was eventually scaled down in specific areas such as the shoulder boxes and the arms (that connect the shoulder boxes to the legs). This new design proved to be much more stable and sturdy, but could be made even smaller to obtain more stability and strength.

Two processors were used, an EVBU along with its daughter board the ME11, in conjunction with the MSCC11, or single chip board. The single chip board was used for controlling the servos, with code written by Drew Bagnell (and others). The EVBU was used to control the IR emitter/detector pairs, and all behaviors, including walking and object avoidance. The code written so far for the walking behavior has been trivial. The servos are represented as an array and the task of walking just includes assigning the

correct value to each servo so that it actuates the position desired. The object avoidance is also trivial, a value is checked for, if found a obstacle must be avoided if not, the behavior does not change. Interrupts were not needed at this time since other behaviors were not implemented.

INTRODUCTION:

HISTORY OF LEGGED LOCOMOTION:

Walking robots can, potentially more successfully, traverse over rugged terrain than either wheeled or tracked robots. However they also face more challenges, some of which can stem from the number of degrees of freedom the agent is trying to walk with, along with the number of legs chosen and the amount of stability obtained with the number of legs. This issue of stability leads to the issue of more directions of motions to coordinate. Such control algorithms can become complicated just to get the agent to take a step, before any consideration of additional behaviors becomes an issue. Some facts listed were obtained from "Mobile Robots", written by Joseph L. Jones and Anita M. Flynn.

OBJECTIVES OF PROJECT:

The objective of this project was to build a quadruped, walking, autonomous agent exhibiting dog-like behaviors. The scope of this project included, building (a previously designed mechanical platform), building the hardware necessary for control, and sensor detection, and writing code to obtain actuation and implementation of behaviors. In this paper I will talk about the Integrated System used, the mobile platform, actuation, sensors used and built, and the behaviors obtained.

INTEGRATED SYSTEM:

The high level control is conducted by the 68HC11, along with its daughter board the

ME11. The lower level control is conducted by the MSCC11 single chip board. This board uses code written by Drew Bagnell and controls all of the twelve servos used in actuating the movements.

The 68HC11 controls the IR detectors and emitters. It uses a memory mapped output latch for the IR emitter, and the analog port (port E) for the IR detection. A value is read from the port and some action is performed or not performed depending on that value.

The positions of each of the servos, depending on IR detection and the sequence of the walking algorithm, is also determined by the code that resides on the 68HC11. This is then sent, through serial interface, to the MSCC11 which translates the position into a pulse width, which is sent to the servos and actuation is obtained.

MOBILE PLATFORM & ACTUATION:

As stated previously ASTRO's mobile platform was directly copied from a previously designed autonomous agent. Some specifications for the platform were that it should be light, using model-aircraft plywood supplied by IMDL lab, and that there should be three degrees of freedom per leg for a wide range of possible movements. At the time of the initial construction size was not taken much into consideration, but as walking was implemented is was found that a much smaller body could have been used and would have proved to be much hardier and stable.

A total of three servos per leg is used to implement three degrees of freedom on each of the four legs. Each servo provides one degree of freedom, i.e. movement in one direction. Three degrees of freedom was chosen for the ability to have a wider range of motion, including upward/downward, forward/backward, and outward/inward motions.

The platform was constructed out of model-aircraft plywood, supplied by the IMDL lab. The components were cut out using the T-Tech routing machine. The servos used were ordered from Hobby Shack and were CIRRUS CS-70 Standard Pro Servo with 42oz/in capabilities. In the end I realized that the four lift servos should have been stronger. These four are the ones that endure the most stress at any time. The compromise was made between having a heavier platform that would allow less slip on the floor but with less or not enough strength in the lift servo, or having a lighter platform and having enough strength in the lift servo but not providing enough weight to prevent slippage on the floor. Slippage occurred even when rubber stops were used on the bottom of the legs due to the lack of weight used because of the lift servo capabilities.

The upper and lower plates are attached to one another, on the top by the shoulder box servo and on the bottom by a simple screw system with a flat nylon washer and a cylindrical nylon spacer to allow for fairly frictionless movement.

Cad files were not obtained from the previous platform in order to download the correct shapes to the T-Tech routing machine. Instead measurements were taken off the actual robot "BOB" and then scaled to the appropriate proportions (considering servos size etc). A recreation of cad files was then made using these measurements. Several things that did not go smoothly were the shoulder boxes and the arms connecting the boxes to the legs. The shoulder boxes (made with four sides), each have a tongue and groove edge so that the boxes fits together. When offsetting the original drawing, for the T-Tech bit, these tabs DO NOT offset correctly. Additionally the shoulder boxes and arms were made before the servos arrived, and were made too large so they had to be remade. The body as a whole could have been made smaller, but due to time constraints this was not a priority. The algorithm for walking is displayed in Diagram 1. It will be used to describe taking a step with the right side:

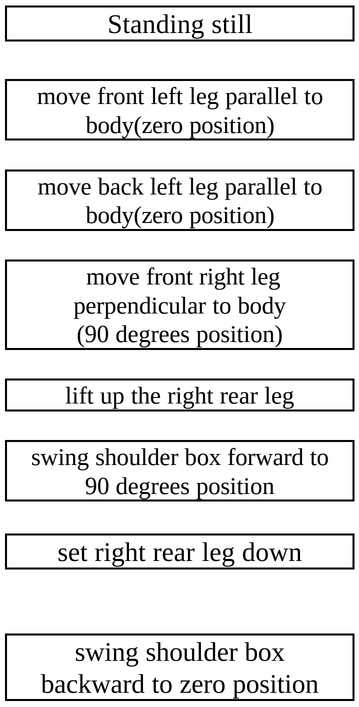
Standing still
move front left leg parallel to body(zero position)
move back left leg parallel to body(zero position)
move front right leg perpendicular to body (90 degrees position)
lift up the right rear leg
swing shoulder box forward to 90 degrees position
set right rear leg down
swing shoulder box backward to zero position

DIAGRAM I

This algorithm can be generalized for stepping with the left side as well. The front legs don't really have a stepping algorithm. Moving from the perpendicular position to the parallel position (while setting up the tripod for rear leg movements) is the only stepping motion they perform.

SENSORS:

“Sensing is not perceiving. Sensors are merely transducers that convert some physical phenomena into electrical signals that the microprocessor can read.” As stated in “Mobile Robots”, by Jones, J. and Flynn, A. The sensors originally intended to be implemented on

ASTRO include the standard IR emitter/detectors for object avoidance, a microphone for voice command response, a pyroelectric sensor for motion detection, and bump switches for additional object detection (to back up the IR).

Unfortunately most of the effort was concentrated on the mechanical design so the only sensors built were, two standard IR emitter/detector pairs, and a microphone circuit to detect frequency in the range of 1kHz which would be used for detecting a whistle. The later sensor was never implemented on the actual robot but was built and tested. The microphone had a sensitivity of 1Vp-p output for up to about five feet away. The circuit for this microphone is included in Fig 1. The circuit contains a two stage amplifier, to amplify the output of the microphone. Next the signal is passed through a low-pass filter and then through a high pass filter and then sent through a comparator, in order to convert the signal to digital for the processor to read. The frequency used for the low-pass filter was 5kHz, and the frequency used for the high-pass filter was 1kHz. The test results are given in Table 1 below.

Distance (ft)	Output Voltage (Vp-p)
0.5	2.5
1	2
2.5	1.5
5	1

TABLE 1

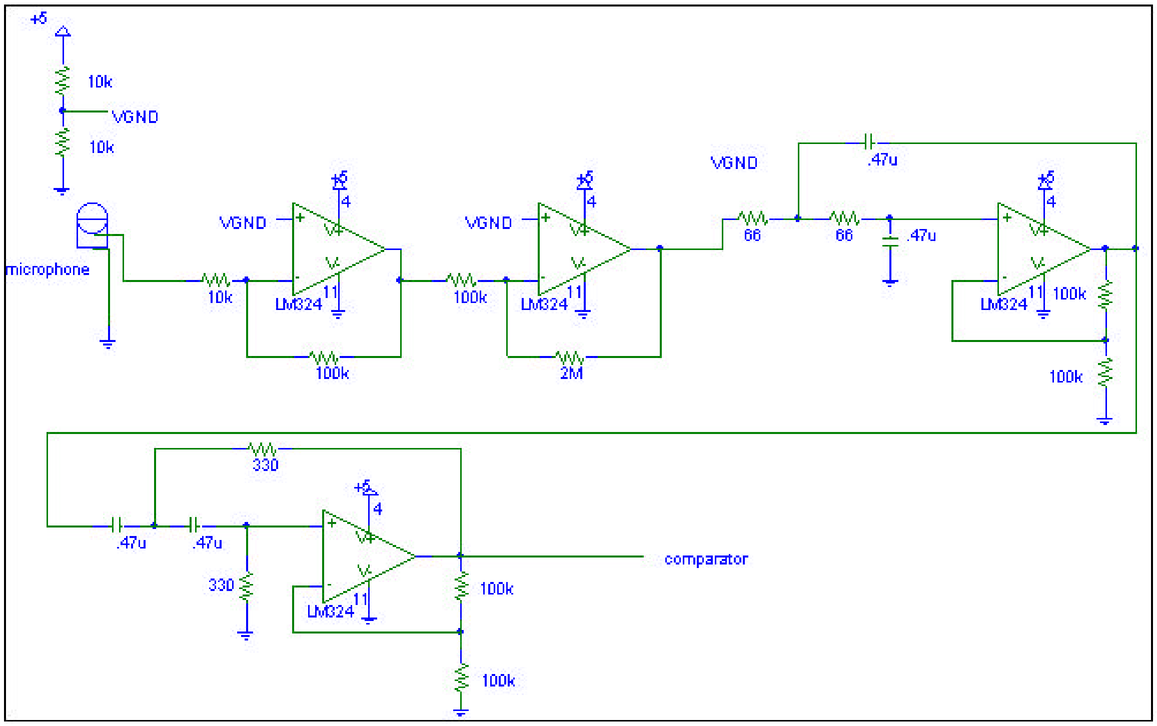
+5
10k
VGND
10k
microphone
VGND
10k
LM324
100k
VGND
100k
LM324
2M
VGND
66
66
.47u
.47u
LM324
100k
100k
330
.47u
.47u
330
LM324
100k
100k
comparator

FIG 1

BEHAVIORS:

The behaviors that ASTRO currently exhibits are walking and collision avoidance. The walking behaviors that he exhibits are walking forward over semi rough/smooth surfaces. Walking backwards when an object is detected in both of the IR detectors. If only one IR detector "sees" an obstacle the behavior is to turn left. I tried to implement both turning right and turning left, but due to mechanical limitations turning right was not successfully accomplished. The object avoidance was achieved through hacking the standard IR detectors and writing a small amount of code to implement the signal that the detector is receiving.

Other behaviors were intended to be implemented but several time constraints were an obstacle in accomplishing additional behaviors. This area would be the concentration of further work done in the future.

CONCLUSION:

In conclusion ASTRO was a success. He is not perfect, by any means, but with all factors considered (other classes, work, and learning new material in the process) he is a great accomplishment. ASTRO is a third generation quadruped walking robot with minimal object avoidance. A walking behavior was the minimum goal set for the semester and it was accomplished.

Limitations of the work done would be mostly attributed to the mechanical design. The fact of not having initial CAD files to start with was a draw back and led to many other unforeseen difficulties that could have been avoided had I had the CAD files to start from.

The success of getting the robot to walk was a big accomplishment in itself. I thought the problem of balance was going to be more a hinderance than it turned out to be. The main issue once I started working on the walking algorithms was the strength provided by the lift servos. The walking patterns were learned from watching a downloaded video of ASTRO's ancestor THING. This video can be obtained by visiting the following site: http//piglet.cs.umass.edu:4321/thing/thing.html.

Work planned for the future: make the body a more compact, tight structure, increase the strength of the lift servos, add sensors and behaviors to the platform, and possibly recreate the structure out of metal, like the ancestor THING. If I were to start the project over tomorrow, and know what I do now, I would first research the body structure more thoroughly and design the body right the first time so that more time could be dedicated to other areas.

REFERENCES:

Arroyo, Dr. A. "The 6.270 Robot Builder's Guide", Fred G. Martin, 1992.

Jones, Joseph L. “Mobile Robots: Inspiration to Implementation. “ A. K. Peters, Ltd., 1993.

Maes, Pattie, “Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back”, The MIT press, 1991.

MacDonald, W., “Design and Implementation of a Multilegged Walking Robot”, University of Massachusetts - Amherst Laboratory for Perceptual Robotics, 1994.

Reddish, A., “DOGBOT”, University of Florida Intelligent Machine Design Laboratory, 1997.

Stancliff, S., “BOB”, University of Florida Intelligent Machine Design Laboratory, Spring 1998.

Van Anda, J., “QUADRO”, University of Florida Intelligent Machine Design Laboratory, 1997.

APPENDICES:

APPENDIX A: Walking Algorithm

APPENDIX A:

```
/* This code implements the walking algorithm described in the Actuation
section of this paper.  It uses a single array to represent the 12
servos.  The values assigned to each servos represent a pulse width,
this number is directly related to the position of the servo in degrees.
*/


#define DELAY 5500
#define IRLATCH *(unsigned char*)(0x7000)

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

/* currpos holds the current position of the servos */
char cp[12];

void servoit(int sn, int pw);
void doservos(void);
void waitabit(int waittime);
doit(int waittime);
void zeroservos(void);

void backwards(void);
void turn( int m );
void setup( void );


int main(void)
{
 int i, j, lval, rval;
 init_serial();
 zeroservos();
 init_analog();
 IRLATCH = 0xFF;

 /* first set the new starting positions */
 setup();
 j = 0;
 while(1)
 {

  lval = analog(6);
  rval = analog(7);

  if( lval > 100 && rval > 100 )
     {
       setup();
       backwards();
       j++;
       if( j == 2 )
        { turn( 8 ); }
     }

  else if( lval > 100 )
     {
       j = 0;
       turn( 7 );
     }

  else if( rval > 100 )
```

```
    {
      j = 0;
      turn( 15 );
    }

 else
 {
 /* begin right side movement */
 /* first shift the weight on to the left side */

 cp[1] = 58;  /* zero posn for shoulder box */
 doit(DELAY);

 cp[4] = 55; /* lift up to move to zero posn */
 cp[5] = 64;  /* zero posn for shoulder box */
 doit(DELAY);
 cp[4] = 80;  /* return to start posn */
 doit(DELAY);

 cp[7] = 120;  /* zero posn - 20 */
 doit(DELAY);

 /* back right leg movement */
 cp[10] = 45; /* bring leg up (was 40 in test7) */
 doit(DELAY);
 cp[8] = 0;  /* swing shoulder box forward */
 doit(DELAY);

 cp[10] = 100; /* set back right leg more down for movement */
 doit(DELAY);

 cp[8] = 55;  /* swing shoulder box back */
 doit(DELAY);

 /* start movement on the left side */
 /* first shift weight to the right side */

 /* cp[8] is already a zero posn */

 cp[11] = 65; /* lift up to move to zero posn */
 cp[7] = 58;  /* zero position for shoulder box */
 doit(DELAY);
 cp[11] = 20; /* return to start posn */
 doit(DELAY);

 /* front left perpendicular for tripod */
 cp[5] = 0;  /* 90 degrees position */
 doit(DELAY);

 /* back left movement (small movements are good) */
 cp[0] = 75;  /* pick leg up */
 cp[1] = 75;  /* swing left rear shoulder forward */
 doit(DELAY);

 cp[0] = 25; /* more down for movement left rear */
 doit(DELAY);
 cp[1] = 70; /* swing shoulder back for movement */
 doit(DELAY);
 }
}
```

```
}

void backwards()
{
int m = 4;

/* begin backwards movement */

 while(m>0)
 {
  setup();
  /* begin right side movement */
  /* first shift the weight on to the left side (bring legs down)*/
  cp[5] = 64;  /* zero posn for shoulder box */
  doit(DELAY);

  cp[0] = 75;  /* lift up to move shoulder box */
  cp[1] = 58;  /* zero posn for shoulder box */
  doit(DELAY);
  cp[0] = 40;  /* return to start posn */
  doit(DELAY);

  /* back leg used in tripod position/ front used to propell */
  cp[8] = 10;  /* zero posn + 10 */
  doit(DELAY);

  /* front right leg movement */
  cp[11] = 75; /* bring leg up */
  doit(DELAY);
  cp[7] = 90;  /*(was 45 for 8) swing shoulder box forward */
  doit(DELAY);

  cp[11] = 20; /* set back right leg more down for movement */
  doit(DELAY);
  cp[7] = 80;  /*(was 55 for 8) swing shoulder box back */
  doit(DELAY);

  /* start movement on the left side */
  /* first shift weight to the right side */
  cp[7] = 58;  /* zero position for shoulder box */
  doit(DELAY);

  cp[10] = 45; /* up for shoulder movement */
  cp[8] = 64;  /* zero position for shoulder box */
  doit(DELAY);
  cp[10] = 90; /* back down to start posn */
  doit(DELAY);

  cp[1] = 100;  /* zero position - 20 */
  doit(DELAY);

  /* front left movement (small movements are good) */

  cp[4] = 40;  /*(was 80 for 0) pick leg up */
  doit(DELAY);
  cp[5] = 40;  /* swing left rear shoulder forward */
  doit(DELAY);

  cp[4] = 90; /* more down for movement left rear */
  doit(DELAY);
  cp[5] = 60; /* swing shoulder back for movement */
  doit(DELAY);

  m = m - 1;
 }
}

void turn( int  h )
{
  backwards();

 while( h > 0 )
  {
     setup();
   cp[1] = 58;
   doit(DELAY);

   /* before turning left side balance the right side with tripod */
   cp[11] = 65;  /* up for shoulder movement */
   cp[7] = 64;
   doit(DELAY);
   cp[11] = 45;  /* down to start */
   doit(DELAY);

   cp[8] = 64;
   doit(DELAY);

   cp[4] = 55; /* up for shoulder movement */
   cp[5] = 0;
   doit(DELAY);
   cp[4] = 75;  /* down to start */
   doit(DELAY);

   cp[1] = 120;
   cp[5] = 64;
   doit(DELAY);

   /* now move right side to turn left */
   cp[7] = 120;
   doit(DELAY);
   cp[10] = 45;
   cp[8] = 25;
   doit(DELAY);
   cp[10] = 75;
   doit(DELAY);

  h = h-1;
  }

}

/* CCW is postitive */
/* 64 is centered, 1 is ~45degrees all the way right,
   127 is all the way left   */
void servoit(int sn, int pw)
{
 put_char(0);
 put_char(sn);
 put_char(pw);
}
```

```
void doservos(void)
{
 int i;
 for (i=0;i<12;i++){servoit(i,cp[i]);}
}

void waitabit(int waittime)
{
 int i;
 for (i=1;i<waittime;i++){}
 return;
}
doit(int waittime)
{
 doservos();
 waitabit(waittime);
}
void zeroservos(void)
{
 int i;
 for(i=0;i<=12;i++) {cp[i]=64;}
 doit(2*DELAY);
}
void setup( void )
{
  cp[7] = 100;
  cp[8] = 25;
  cp[5] = 25;
  cp[1] = 100;
  cp[11] = 45;
  cp[10] = 90;
  cp[0] = 40;
  cp[4] = 70;
  cp[9] = 50;
  cp[6] = 58;
  cp[2] = 58;
  cp[3] = 55;
  doit(DELAY);
}
```

APPENDIX B:

```
/*//Drew Bagnell
//Servo Controller for single chip board E2
```

```
//This program allows control of 16 servos in C, through the serial
//port. It sends an 'a'
//to indicate it has started. Send the servo number to control in
//hex,followed   by the
//pulse width, and then a comma. To turn the servo on or off, send the
//servo   number in hex,
//and a 0 for off, a 1 for on.
//The chip should return an 'r' for each successfully recieved commands.
//Pulse   widths vary
//full scale of servo from about 1250 to 4500 (depends on the servo). Do
//not   exceed 1000 to
//5000 (since the servo can't handle it, and it screws timing all up.)
*/

#include "xsr16.h"
#include <serial.h>


void
main(void)
{
   int servo_number,instruction_holder,final_pw,i;

   init_serial();

   init_servos();

   putchar('a');

   while(1)
   {

   servo_number = get_char();
   if (servo_number < 58 )
      servo_number -= 48;
   else
      servo_number = servo_number - 'a' + 10;

   final_pw = 0;

   i = 0;
   while( ( (instruction_holder = get_char()) != ',' ) && (i < 5))
      {
              instruction_holder = instruction_holder - 48;
                  final_pw = final_pw * 10;       /*sneaky...
                  final_pw += instruction_holder;                 */
                i++;
   }

   putchar( 'r' );

   if(final_pw == 0 || final_pw == 1)
      servo_power(servo_number,final_pw);
   else
      servo(servo_number,final_pw);
   }
}
```

APPENDIX C:

see floppy submitted with final report.