EEL 5666

IMDL

Fall 1999

Final Report

"Sparky"
The Robotic Dog

By:
Mike Gerhard

Table of Contents

Abstract

Have you ever wanted a pet dog, but thought that they are too much trouble? If you answered "yes" than Sparky is the perfect companion for you. Sparky is a four-legged autonomous mobile agent. Sparky, however, is different from the typical quadruped. Sparky was designed to be both simple (in design, construction and programming) and cheap. At a cost of ~$100 Sparky is not only one of the cheapest robots in the lab, but one of the most practical. The reason that you don't see robots everyday as either toys or assistants isn't because we don't have the ability to make them. It is because we don't have the ability to make the cost-effective. Sparky is the exception. Sparky was design to be a toy (or experimenter platform). A necessity in designing a robot for this target market is a low cost.

The design is elegant in its simplicity. The design has several pertinent consequences. As a result of the simplistic design Sparky is both easy to program and able to turn on his own axis. Sparky can walk forward, backward, turn left and turn right. This is accomplished using only four 42oz/in servos, hacked to spin 360 degrees. He currently has 6 total sensors: 4 IR emitter/detector pairs for leg synchronization and 2 IR LED and hacked (analog) sharp cans.

Executive Summary

What is innovation? I like to say that it is doing a familiar task in a new way. This is exactly what Sparky represents. Sparky is a four-legged autonomous mobile agent designed by me (Mike Gerhard). What makes Sparky innovative is the way he walks. Nearly all of the quadrupeds (four-legged walkers) that I have seen before Sparky have either been extremely complicated devices that use 8 or more servos or they have been very simplistic creatures with limited ability to move. Sparky can do almost everything that his eight and twelve servo peers can do, with only four (low-torque) servos. This makes him more efficient, cheaper and much easier to program than the competition.

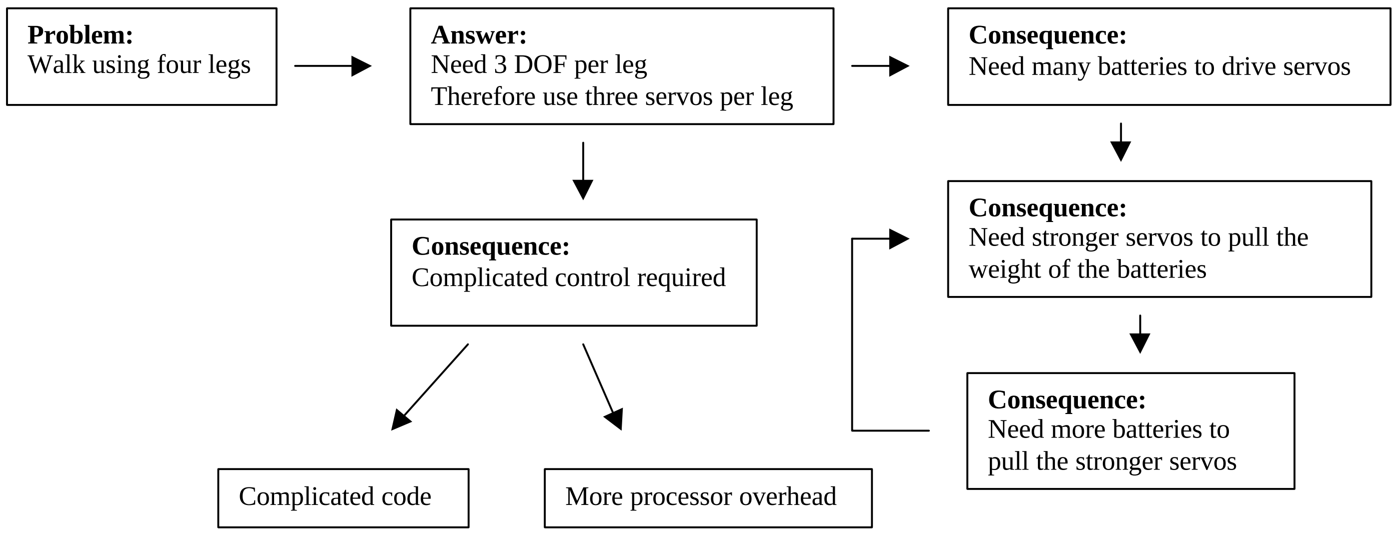
Introduction

I have built a robotic dog. The purpose is to be a toy. What makes this project different from previous attempts is the philosophy behind it. This project was born of a philosophy of simplicity. This philosophy is driven by two reasons: 1) simple=cheap=marketable 2) simple=cheap=good for experimenting. I have built my robotic dog with only four servos (one for each leg). With this simplistic design I have made a platform that is cheap enough for a wide range of experimenters (or possibly customers) and has comparable abilities to its eight and twelve-servo brethren.

Integrated Systems

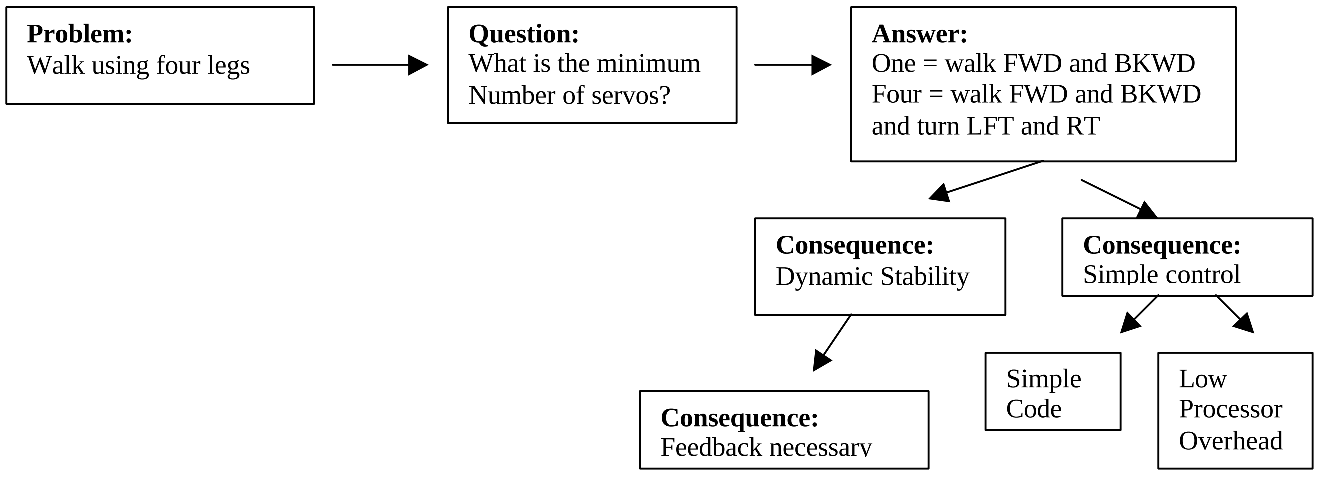
Sparky is a simple quadruped that performs obstacle avoidance and can be easily programmed to execute any sequence of movements (e.g. walk forward then turn left then back up). The key to this platform is the mechanics of the legs. My leg design has overcome several major difficulties that plague quadrupeds (e.g. expensive servos, complex movements). Most of the problems that I have seen associated with the majority of quadruped designs currently under development can be summed up in one word complexity. An excellent example is the "Bob" or "Thing" platform. I am told that this was the most successful quadruped design in the MIL lab. This design has been used by several successors and I have seen similar designs at other research institutions. The problems associated with this type of design can be seen best in a flow chart:

<u>Previous Approach to Quadrupeds:</u>


Problem: Walk using four legs
Answer: Need 3 DOF per leg Therefore use three servos per leg
Consequence: Need many batteries to drive servos
Consequence: Complicated control required
Consequence: Need stronger servos to pull the weight of the batteries
Consequence: Need more batteries to pull the stronger servos
Complicated code
More processor overhead

It's obvious that this thought process will get ugly fast. My approach is quite different.

<u>My Approach to Quadrupeds:</u>

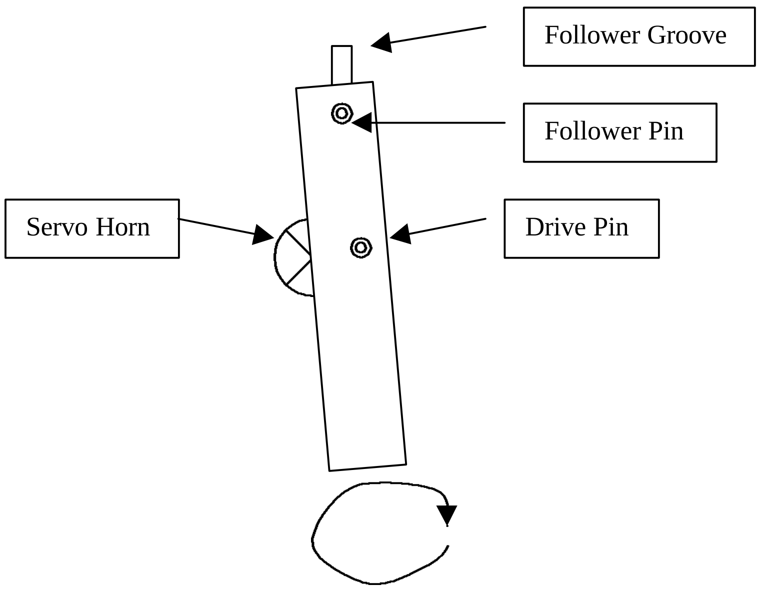

Problem: Walk using four legs
Question: What is the minimum Number of servos?
Answer: One = walk FWD and BKWD Four = walk FWD and BKWD and turn LFT and RT
Consequence: Dynamic Stability
Consequence: Simple control
Consequence: Feedback necessary
Simple Code
Low Processor Overhead

Mobile Platform

The platform is a four-legged walker of my own design. I used one servo per leg. The servos are standard 42oz/in servos by diamond. The motherboard is a TJpro board by Mekatronix that features a 68HC11 with 32k SRAM. The frame and legs were carved from 1/8" plywood using a T-tech milling machine (intended to cut and drill PCB's). The plans were drawn in AutoCAD version 14. I had intended on using a voice recognition module by VoiceDirect for communication between Sparky, and I, but the module was damaged during testing and there wasn't time to get another. Currently, the only sensors that he (Sparky) has are 4 opto-couplers and 2 IR LED and can pairs. The four opto-couplers are used as feedback from the legs to synchronize them and the two IR LED and can pairs are used for obstacle avoidance.

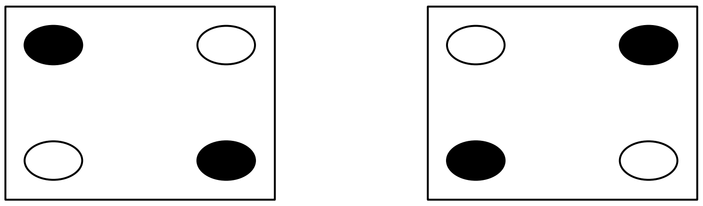
Actuation

The servos are standard 42oz/in servos by diamond hacked to spin 360 degrees. My design is very different from the typical quadruped. Instead of assigning one servo per DOF (Degree of Freedom) I used a cam and follower system to translate the circular motion of the servo to the end of each leg. Here's how I did it:



With this simple leg design I am able to make Sparky walk forward and backward, turn left and right. However, since I am not able to center his center-of-gravity over three legs before I move the fourth, a method of dynamic stability is required. For this, I chose to use a 1-2 Gait and balance Sparky on two legs for half of each step cycle.

<u>The Gait:</u>



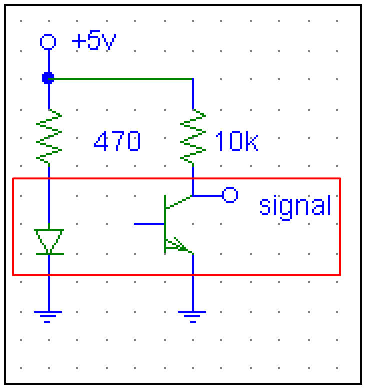
The gait (method of walking) involves moving two pairs of legs as shown. To move forward or backward all of the legs spin in the same direction. To turn each leg must spin in the opposite direction of their partner.

Sensors

I have used the typical IR sensors for basic obstacle avoidance. This consists of two hacked sharp can receivers and two IR LED's. The "special sensor" for this project is my unusual use of opto-couplers. I have four opto-couplers (usually used with shaft encoders) mounted on the robot above each leg. Instead of using the typical shaft encoders to interrupt the beam I am using thin pieces of black plastic attached to each leg. The effect achieved by using the plastic pieces instead of a shaft encoder is twofold: 1) the opto-couplers can be mounted further from the motor and a uniform spacing on all sides of the motors is not required as would be inherent with the disk shape of an encoder. 2) An interrupt only occurs once per cycle and therefor no counting is required; an interrupt means that the leg is at the top of its cycle. There isn't much to say about the experimental results. The sensors read ~.2V (10/256) when there is no interruption of the beam. When there is an interruption in the beam the reading jumps to ~4.85V (255/256).

Sensor Construction and Wiring

The opto-coupler sensor is very simple to construct. The circuit is comprised of the opto-coupler itself (Radio Shack #276-142) and two resistors (10k and 470). The circuit is shown below with the Radio Shack part outlined in red.



Behaviors

Sparky has 2 modes. Each mode consists of four functions. Mode1 is the demonstration mode. Mode2 is the obstacle avoidance mode. In each mode, Sparky is capable of his four basic functions: 1) Walk Forward 2) Walk Backward 3) Turn Left 4) Turn Right. In obstacle avoidance mode Sparky demonstrates two behaviors: 1)-avoiding obstacles 2) scared. The obstacle avoidance behavior consists of 4 cases: 1) obstacle in front and to the right so turn left 2) obstacle in front and to the left so turn right 3) obstacle directly ahead so back up 4) no obstacles so go forward. The scared behavior is provoked by using the "reprimand device" (i.e. a 40kHz IR generator). When the "reprimand device" is activated Sparky moves away from it. This is linked to the obstacle behavior through the obvious method.

Conclusion

I was able to accomplish all that I expected to with the robot. I set a list of goals for this project:

Goals:
1. use four legs
2. walk forward and backward, turn right and left
3. use inexpensive low-torque servos
4. respond to voice commands

I completed three out of the four. I probably would have been able to do all four if it weren't for an unfortunate accident involving my voice module late in the semester. I would still consider this project to be a success because I listed the voice response as a secondary concern in my proposal. Sparky is not without flaws, however, he is very slow and takes very small steps. These are problems that I expected and they are inherent to the design. The speed issue could be resolved by using fast gearhead motors instead of servos. The size of the step could be increased through a few slight mechanical changes. There are, however, difficulties involved in doing this. An increase in the step size would require an increase in the torque of the servos/motors. Also, since the design is dynamically stable and balances on two legs during half of the walking motion, there are balance concerns (bigger step = more airtime).

Future Work

I don't plan on doing any further development with this platform, but I will list my suggestions in case I change my mind or someone decides to follow in my footsteps. My first suggestion is to clean-up the CAD design a bit. I would make the two top pieces removable, like TJ. I would also add two supports across the middle, which would make a compartment for the batteries, motherboard, etc. My next suggestion would be to decrease the total size. I think that it can be built at half the current size (I scaled it up by a factor of two several weeks into the course because of balance concerns). Next, I would suggest removable legs. The design was intended for this as a "back-up" (in case it didn't walk), but it would be nice to see it as a "feature". Finally, I would suggest an array of cool sensors: camera, laser, etc.

Appendix A -- Program Code

```
/**************************** Includes ********************************/
#include <tjpbase.h>
#include <stdio.h>
#include <HC11.h>
/************************* End of Includes ****************************/

void mike_avoid(void);
void mike_sync(int rrm, int rfm, int lrm, int lfm);
void mike_dance(void);

/**************************** Constants ********************************/

#define AVOID_THRESHOLD 100
#define pwh 4000
#define pwl 2000

#define DANCE_ON  *(unsigned char *)(0x7000) = 0x87
#define DANCE_OFF  *(unsigned char *)(0x7000) = 0x07
#define LEFT_ON  *(unsigned char *)(0x7000) = 0x17
#define RIGHT_ON  *(unsigned char *)(0x7000) = 0x27
#define LEFT_OFF  *(unsigned char *)(0x7000) = 0x07
#define RIGHT_OFF  *(unsigned char *)(0x7000) = 0x07
#define BOTH_ON  *(unsigned char *)(0x7000) = 0x37
#define BOTH_OFF  *(unsigned char *)(0x7000) = 0x07

/************************* End of Constants ****************************/



/****************************** Main **********************************/
void main(void)
{
  unsigned int bl, cv, fb, rb, run_test;
  int rro,rfo,lro,lfo;

  init_analog();
  init_clocktjp();
  init_serial();
  init_motortjp();
  init_servotjp();


  IRE_ON;     /* turn on IR emitters */


#define  rro analog(4)
#define  rfo analog(6)
#define  lro analog(1)
#define  lfo analog(5)
```

```
  START;
  wait (500);
 DANCE_ON;     /* turn on dance indicator light*/
            mike_dance();
 DANCE_OFF;   /* turn off dance indicator light*/

            mike_avoid();


}



void mike_avoid(void)

/*************************** mike_avoid *********************************
 * Description
                                                    *
 *   This is my attempt to do obstacle avoidance                                         *
 ********************************************************************/

{

int irdr,irdl,rrm,rfm,lrm,lfm;

            rrm = pwh; /*rrm = PA7;*/
            rfm = pwh; /*rfm = PA3;*/
            lrm = pwl; /*lrm = PA6;*/
            lfm = pwl; /*lfm = PA4;*/

  IRE_ON;     /* turn on IR emitters */

   while(1)
  {

/*
   The following block will read the IR detectors, and decide whether TJ
   needs to turn to avoid any obstacles
*/
    irdr = RIGHT_IR;
    irdl = LEFT_IR;

    if (irdl > AVOID_THRESHOLD && irdr < AVOID_THRESHOLD ){
            LEFT_ON;                                              /*TURN ON INDICATOR LED*/

                                                                       /*object to the left, so turn
right*/
                        rrm = pwl;
                        rfm = pwl;
                        lrm = pwl;
                        lfm = pwl;
                        mike_sync(pwl,pwl,pwl,pwl);
            LEFT_OFF;                                             /*TURN OFF INDICATOR LED*/
                        }
    if (irdr > AVOID_THRESHOLD && irdl < AVOID_THRESHOLD){
            RIGHT_ON;                                       /*TURN ON INDICATOR LED*/

                                                                    /*object to the right, so
turn left*/
                    rrm = pwh;
                    rfm = pwh;
                    lrm = pwh;
                    lfm = pwh;
                    mike_sync(pwh,pwh,pwh,pwh);
                    RIGHT_OFF;                              /*TURN OFF INDICATOR LED*/
                    }


    if (irdr > AVOID_THRESHOLD && irdl > AVOID_THRESHOLD){
                    BOTH_ON;                                /*TURN ON INDICATOR
LED'S*/

                                                                    /*object straight ahead, so
back up*/
                    rrm = pwl;
                    rfm = pwl;
                    lrm = pwh;
                    lfm = pwh;
                    mike_sync(pwl,pwl,pwh,pwh);
                    RIGHT_OFF;                              /*TURN ON INDICATOR LED*/
                    LEFT_OFF;                               /*TURN ON INDICATOR LED*/
                    }
            else {
                                                                    /*go straight forward*/
                    rrm = pwh;
                    rfm = pwh;
                    lrm = pwl;
                    lfm = pwl;
                    mike_sync(pwh,pwh,pwl,pwl);
                    BOTH_OFF;                               /*TURN OFF INDICATOR
LED'S*/
                    }
    wait(35);
  }

}

void mike_sync(int rrm, int rfm, int lrm, int lfm)

/*************************** mike_sync **********************************
 * Description
                                        *
 *   This is my attempt to synchronize the legs                                         *
 *       INPUTS:                        rrm,rfm,lrm,lfm
                                        *
 *                  right_motor          = PA3
                                        *
 *                  servoz               = PA4
                                        *
 *                      servo1                  = PA5
                                               *
 *                      servo2                  = PA6
                                               *
 *                      left_motor              = PA7
                                               *
 *             motorp(LEFT_MOTOR, rrm);                 *left_motor = PA7
           *
 *     servo(0,lfm);                                            *SERVOZ, PA4
           *
 *     motorp(RIGHT_MOTOR, rfm);                        *right_motor = PA3
           *
 *     servo(2,lrm);                                            *SERVO2, PA6
           *
 ***********************************************************************/
{


/*walk forward  --  two legs at a time*/

/* spin first two */

            servo(0,lfm);                                                   /*spin leg*/
            motorp(LEFT_MOTOR, rrm);                                /*spin leg*/
            wait (400);
            while ( rrm != 0 || lfm != 0 )
            {
            if ( rro > 20  )                                        /*at top*/
                        {
                        servo(0,0);                                                     /*stop it*/
                        rrm = 0;
                        }
            if ( lfo > 20 )                                                 /*at top*/
                        {
                        motorp(LEFT_MOTOR, 0);                              /*stop it*/
                        lfm = 0;
                        }
            }
            servo(0,0);                                                             /*stop it*/
            motorp(LEFT_MOTOR, 0);                                          /*stop it*/
/* spin second two */

            servo(2,lrm);                                                   /*spin leg*/
            motorp(RIGHT_MOTOR, rfm);                               /*spin leg*/
            wait (400);
            while ( lrm != 0 || rfm != 0 )
            {
            if ( lro > 20 )                                                 /*at top*/
                        {
                        servo(2,0);                                                     /*stop it*/
                        lrm = 0;
                        }
            if ( rfo > 20 )                                                 /*at top*/
                        {
                        motorp(RIGHT_MOTOR, 0);                             /*stop it*/
                        rfm = 0;
```

```
				}
		}
		servo(2,0);							/*stop it*/
		motorp(RIGHT_MOTOR, 0);						/*stop it*/

}

void mike_dance(void)
/* do alittle dance make alittle noise */
{
int i;
i = 1;
		for ( i = 1; i < 5; i++ )
				{
				/*turn right*/
				mike_sync(pwl,pwl,pwl,pwl);
				/*turn left*/
				mike_sync(pwh,pwh,pwh,pwh);
				}

}
```

Appendix B -- CAD Drawings: