University of Florida
Department of Electrical and Computer Engineering

EEL 5666

Intelligent Machines Design Laboratory

"Centipede"

Final Report

4/21/99

Designer
James S. Rutledge

Table of Contents

Abstract

The goal of this paper is to describe the design project that created a robot that has the ability to creep over objects with a diverse suspension system. The robotics system will climb small obstacles without being stopped and objects it cannot overtake will be bypassed. Having a snaking body that might even convince the dog to run for cover. This machine has been constructed to live in the world with the rest of the pest of the day. This pest will be better than most though, the only thing it will need to eat are electrons, just recharge the batteries and it will continue to be an annoying pest.

Executive Summary

The Centipede received its name because it is designed into three sections and each is steered by the actuation of the two outer bi-wheeled sections. The Centipede is an autonomous agent prepared for “the end of the world” problems. Which means the robot is able to detect when the system accedes its capabilities in the real world. For instance if it begins to climb something that is to steep it realizes it and backs off and continues its path elsewhere. It is equipped with bump sensors and so to detect collisions with objects that could not be climbed and once again continue its path elsewhere. The Centipede will also have a wall-following capability. All pests hug the wall the only difference is the Centipede follows it at variable distances that the owner specifies through an input on the robot.

Platform

The platform of the Centi is divided into sections. These sections are connected with a spring loaded hinging system. This hinge is designed to remain in an open state. If the hinge is bent in either direction it tends to return to the original state. This has been done to create a platform suspension that is flexible yet rigid. This suspension is designed to allow the robot to climb things in its path.

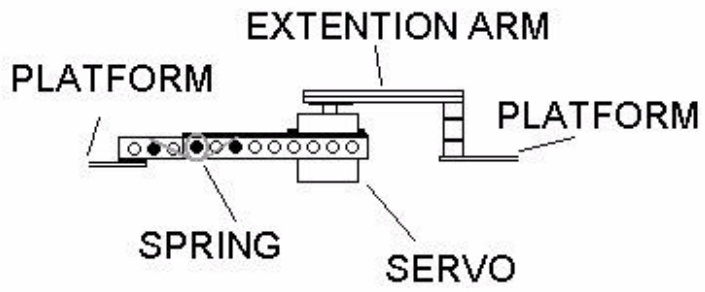

Figure 1

The Platform also contains four-inch wheels that are wider than the 3.75-inch base. This system is then actuated by six servos that have been partially hacked as motors. Another two servos are then used for direction control, mounted in the spring-loaded hinge.

This Platform allows the system to climb most anything that the wheels can get friction on including walls, boxes, …etc.

Sensors

The Centi will have a sensor array consisting of two mercury tilt sensors, two bump switches, IR emitters and detectors, and a simple 1K variable resistor. The

sensor array will be connected to the TJ-pro board using a Motorola 68HC11 chip. This board will communicate through the serial port to a single chip board that will be acting as the motor controller.

IR Sensor System

The IR sensors are Sharp GPIU58Y (hacked). Hacked means that the digital circuitry has been bypassed so to give an analog value out so that the intensity of the IR can be obtained. The sensors are tuned to accept 40 kHz IR light that will be emitted by LEDs. The more intense the IR light the higher the voltage.

The Sensors are placed perpendicular to the robot so as to follow a wall. They have been mounted on a raised platform giving a view above the 4in wheels.

The voltage from the sensors will be brought into the A/D port on the TJ-pro board and will be used to follow walls. Note these sensors are not for obstacle avoidance but for wall tracking only.

Variable Wall Following System

The variable wall following system will be implemented by simply taking a 10k pot in series with a 10k resistor creating a voltage divider with another 10k resistor. This divider will give a minimum voltage of 2.5V and a maximum of 3.4 volts to put into the A/D of the TJ-Pro board. This will vary the voltage coming into the A/D allowing change, on the fly, the number that will be compared to the intensity of the IR receivers. This allows the system to also be more adjustable to ambient IR without needing to be reprogrammed.

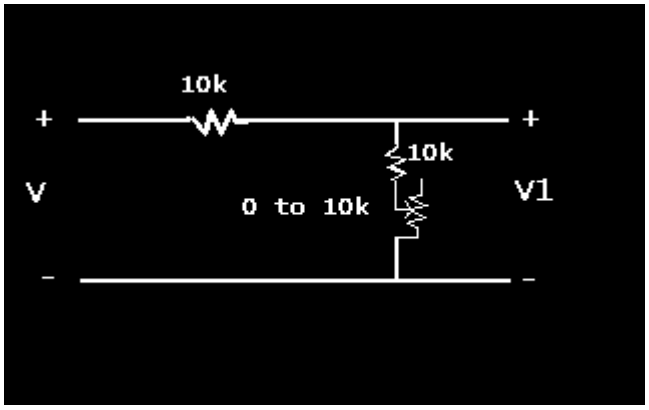

Figure 2

Bump Sensor System

The Centi's bump sensor array consists of two sensors one on the front and one on the back of the robot. They will be connected to the TJ-pro's analog inputs as well. If the sensors are touched it causes a short in the line resulting in a 255 on the analog register, which can then be interpreted by the processor to be an obstacle, and the Centi will perform evasive maneuvers.

Tilt Sensor System

The tilt sensor has been designed so to keep the Centi from climbing an obstacle that may be to steep for its platform configuration. This has been defined as approximately 45° relative to the center of rotation of the hinge. The final sensor can be seen in Figure 3 below. The θ= 145° and the σ=55° these angle were determined by noticing the configuration of the sensor. If the switch became 1°past parallel with the earth's surface it would connect. I wanted to be able to detect a 45° incline with respect to the surface of the earth so it was angled by the following formula θ =180°- 45°+10° = 145°. The 10° are for considering the changes in accelerations in the robot. The accelerations would cause movement in the free-floating mercury, which would give a false reading from the sensor.

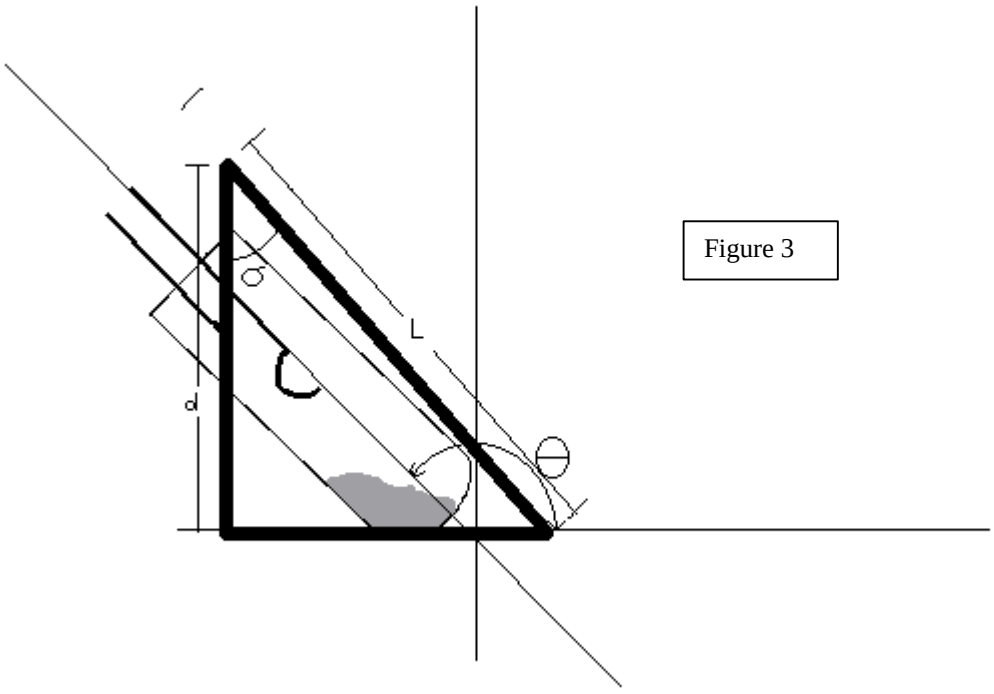

Figure 3

NOTE: L=1.7cm, d=1.2cm

Experiments

IR Testing

The IR were tested with the use of a cotton white sheet of paper and a black sheet of paper. These pieces of paper where placed a different distances from the receiver and then a value was taken.

Distance	Cotton White Sheet	Black Sheet
1”	NA	92
2”	125	88 (ambient)
5”	110	87 (ambient)
10”	95	87 (ambient)
15”	87 (ambient)	87 (ambient)

Table 1

The data shows that at one inch the black sheet of paper could barely be seen, as the distance was increased the only IR light seen was ambient as produced by the IMDL’s lab lights. There is a distinct drop in the IR light seen by the white sheet as the sheet of paper distance increased the value decreased dramatically.

Variable Wall Follow Testing

The variable wall follow test was done simply by connecting the circuit to PE2 on the pro board and using IC to take the analog voltages into the processor. The values obtained are located in table two below. Using the variable voltage divider as a remote control for the robot also tested the system.

	Resistor Value in ohms	**PE2 Value**
R2	**10k**	**91**
R2	**20k**	**127**

Table 2

Tilt Testing

A continuity meter and a protractor were all that was needed to test the tilt sensors. The continuity meter was placed in line with the sensor. The sensor was then placed at different angles to see if a sound was heard.

Angle (degrees)	Beep or No Beep
30	No Beep
40	No Beep
50	Beep

Table 3

This data in table three proves that it is a good design for my purposes.

Conclusion

From the platform to the sensor integration the Centi has been carefully thought out for a successful all terrain robot. The sensor systems that are placed on the Centi are simple but have proven to be very effective for their purposes. This system has been integrated to allow the Centi to roam freely without worry of self-destruction.

Vendor Information

IR System

IR emitters:	LEDs Mekatronics 316 NW 17th St., Suite A Gainesville, FL 32608 http://www.mekatronics.com $0.75 each
IR detectors:	Sharp GPIU58Y Mekatronics $3.00 each

Bump System

Switches	Microswitches Mekatronics $0.75 each

Tilt System

Tilt Switch	Glass Mercury Switch Aamir Qaiyumi $1.00 each

Variable Wall Following System

Potentiometer	1kohm Variable Resistor Electronics Plus 2026 SW 34 St. Gainesville, Fl 32608 (352) 371-3223 $2.00 each

Platform

Spring Loaded Hinge	Torsion Spring 180° Deflection Stock Drive Products Sterling Instrument (516) 328-3300
Servos	43 oz\in Diamond 4000 Mekatronics $11.00

APENDIX A

```
/*********************************************************************
**                                                                  **
**        Title: Centi1.C                                           **
**        Programmer: James S. Rutledge                             **
**        Description: Program is designed to give the Centi life   **
**        This code was installed on the Master processor           **
**                                                                  **
*********************************************************************/

#define DELAY    300  /* Delay between moves (in general) */

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

#define OUTLATCH *(unsigned char*)0x7000

/* currpos holds the current position of the servos */
char cp[9];

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

int min, max, center, rightt, leftt, rightmax, leftmax, centert, centertf, centertr, vardis, vardismax, vardismin;

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]=center;}
 doit(10*DELAY);
}


void positive(int servo)
{
 int pulsew;
 if ((servo==1)||(servo==3)||(servo==5)||(servo==6)){pulsew=max;}
 if ((servo==2)||(servo==4)){pulsew=min;}
 cp[servo]=pulsew;
 doit(DELAY);
}

void negative(int servo)
{
 int pulsew;
 if ((servo==1)||(servo==3)||(servo==5)||(servo==6)){pulsew=min;}
 if ((servo==2)||(servo==4)){pulsew=max;}
 cp[servo]=pulsew;
 doit(DELAY);
}

void right(int servo)
{
 int pulsew;
 if ((servo==0)||(servo==7)) {pulsew=rightt;}
 cp[servo]=pulsew;
 doit(DELAY);
}

void left(int servo)
{
 int pulsew;
 if ((servo==0)||(servo==7)) {pulsew=leftt;}
 cp[servo]=pulsew;
 doit(DELAY);
}

void rightm(int servo)
{
 int pulsew;
 if ((servo==0)||(servo==7)) {pulsew=rightmax;}
 cp[servo]=pulsew;
 doit(DELAY);
}

void leftm(int servo)
```

```
{
 int pulsew;
 if ((servo==0)||(servo==7)) {pulsew=leftmax;}
 cp[servo]=pulsew;
 doit(DELAY);
}

/*void strait(int servo)
{
 int pulsew;
 if ((servo==0)||(servo==7)) {pulsew=centert;}
 cp[servo]=pulsew;
 doit(DELAY);
}*/

void straitf()
{
 int pulsew;
 pulsew=centertf;
 cp[0]=pulsew;
 doit(DELAY);
}

void straitr()
{
 int pulsew;
 pulsew=centert;
 cp[7]=pulsew;
 doit(DELAY);
}

void forward()
{
 positive(1);
 positive(2);
 positive(3);
 positive(4);
 positive(5);
 positive(6);
}

void reverse()
{
 negative(1);
 negative(2);
 negative(3);
 negative(4);
 negative(5);
 negative(6);
 waitabit(300000);
}


void right_turn()
{
 right(0);
```

```
 right(7);
 forward();
}

void left_turn()
{
 left(0);
 left(7);
 forward();
}

void right_turnm()
{
 rightm(0);
 rightm(7);
 forward();
}

void left_turnm()
{
 leftm(0);
 leftm(7);
 forward();
}
void retreat_right()
{
 reverse();
 straitf();
 straitr();
 reverse();
 reverse();
 right_turnm();
 right_turnm();
}

void retreat_left()
{
 reverse();
 straitf();
 straitr();
 reverse();
 reverse();
 left_turnm();
 left_turnm();
}

void evasive_right()
{
 straitf();
 straitr();
 reverse();
 reverse();
 reverse();
 right_turnm();
}
```

```
void evasive_left()
{
 straitf();
 straitr();
 reverse();
 reverse();
 left_turnm();
}

void squigle()
{
 zeroservos();
 straitf();
 straitr();
 leftm(0);
 rightm(7);
 waitabit(90000);
 zeroservos();
 straitf();
 straitr();
 leftm(7);
 rightm(0);
 waitabit(90000);
}

void descision()
{
 if((analog(4)>95)&&(analog(1)>95)){evasive_right();}
 if(analog(4)>analog(1)){retreat_right();}
 else {retreat_left();}
}


int main(void)
{
 int x;
 rightmax=80;
 leftmax=25;
 leftt=35;
 rightt=68;          /* For turning left right or straight */
 centert=62;
 centertf=47;
 centertr=62;
 min=2;
 max=125;
 center=64;

 init_serial();
 init_analog();

 zeroservos();
 forward();
 straitr();
 straitf();
 OUTLATCH=0xff;
```

```
 while(1)
 {
/***************Variable Wall Following inputs************/

            vardis=analog(6);
            vardismax=vardis+10;
            vardismin=vardis-7;


/****************Wall Following Routine*******************/

while(((analog(4)<vardismax)&&(analog(4)>vardismin))||((analog(1)<vardismax)&&(analog(1)>vardismi
n)))
 {
            {
             straitf();
             straitr();
             forward();
            }

            while(analog(0)>23){squigle();}

            if(analog(7)>100){descision();}


            if(analog(7)>100)
            {descision();}
            if(analog(2)>100)
            {descision();}
            if (analog(3)>100)
            {descision();}

            if((analog(4)>vardismax)||(analog(4)>128))
            {
             right_turn();
            }

            if((analog(1)>vardismax)||(analog(1)>128))
            {
             left_turn();
            }
 }
 right_turn();

/****************Tilt Sensor System***************/

            if(analog(2)>100)
            {descision();}
            if (analog(3)>100)
            {descision();}

/*******************Bump Sensor System*********************/

            if(analog(7)>100)
            {descision();}
            doit(DELAY);
```

```
/******************Squigle Motion**********************/

            while(analog(0)>23){squigle();}
 }
}
```

Appendix B

```
/*This code was installed on the Slave processor for servo and motor control*/
/* This is the servo controller written initially by Billy Eno */
/* Servoc99.c last and greatest for fall 98 */

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



void
main(void)
{
   int actualpw,sn,pw;
   init_serial();
   init_servos();
   while(1)
    {
     sn=get_char();
     if (sn==0)
       {
        sn=get_char();
        pw=get_char();
        if(sn!=99)  /*servo number 99 means do sidewinder move*/
         {
          actualpw=1500+(27*pw);
          servo(sn,actualpw);
         }


       }
    }
}
```