Roo

The Autonomous Hopping Robot

by

Aaron Suedmeyer

University of Florida Department of CISE Gainesville, Florida
April 26, 1998
Final Report

Prepared for:

Prof. A. Antonio Arroyo EEL 5666
Department of Electrical and Computer Engineering University of Florida

Table of Contents

Abstract

Roo traverses the world by hopping. He spins around searching for me(I have an IR source). Then he centers on me and hops toward me. Roo goes through a unwinding and locking cycle. Afterwards he returns to the search cycle.

Executive Summary

A robot was created that would travel about its environment by hopping. The robot was built in two distinct parts the body and the legs. The robot uses infrared detectors for IR searching and following. The robot also uses a single chip 68HC11 board purchased from Novasoft . The robot also use three servos hacked as servomotors and one as an actual servo. Two of the larger servomotors were used for the force to hop the other motor was for spinning the robot toward IR. The servo was for locking the robot down to generate the force to jump. The robot was programmed to exhibit IR searching with a specific search pattern and a shutoff mode when there is no IR present, IR following, Self calibration for its IR detectors, as well as Hopping.

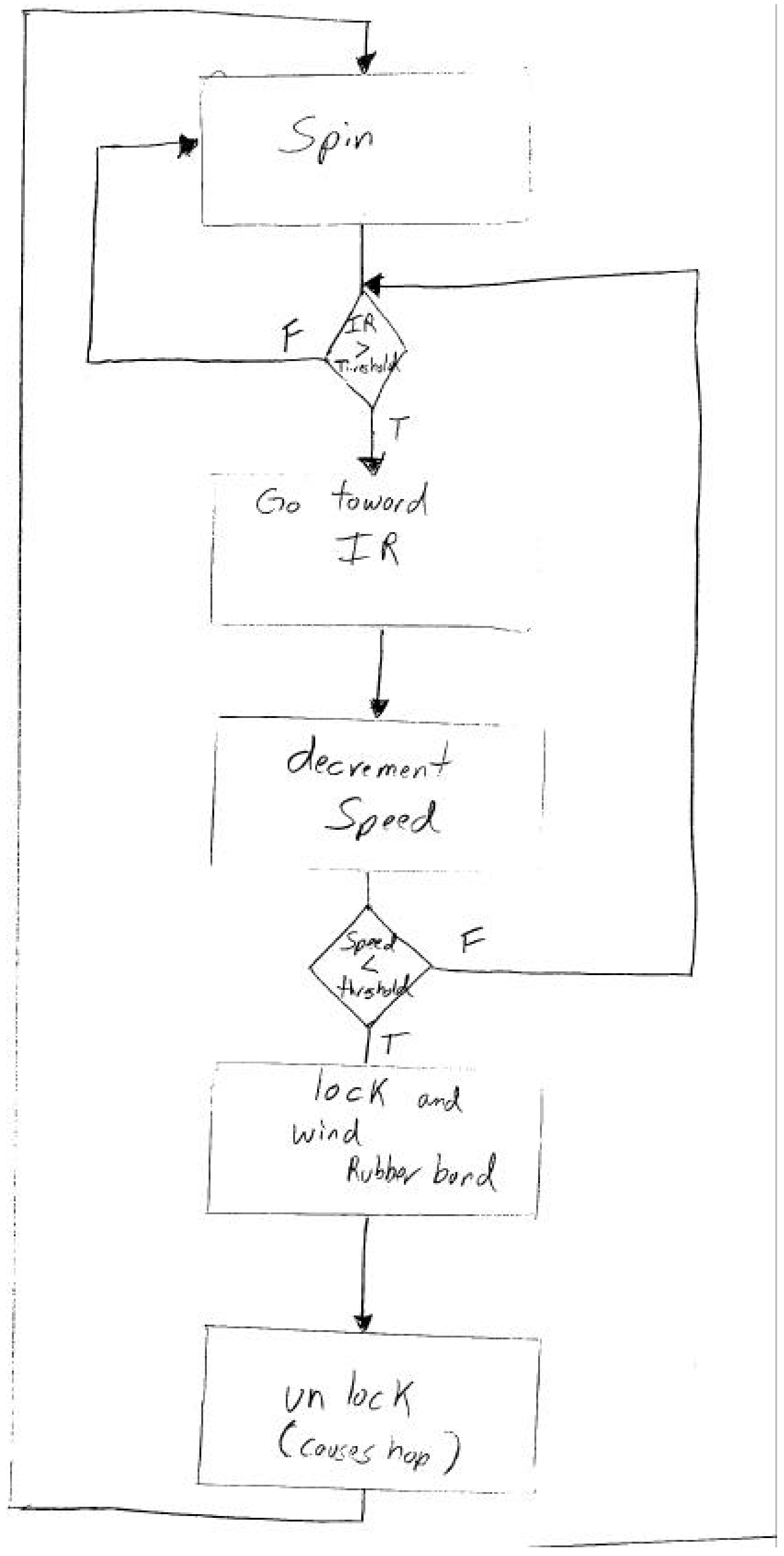
Introduction

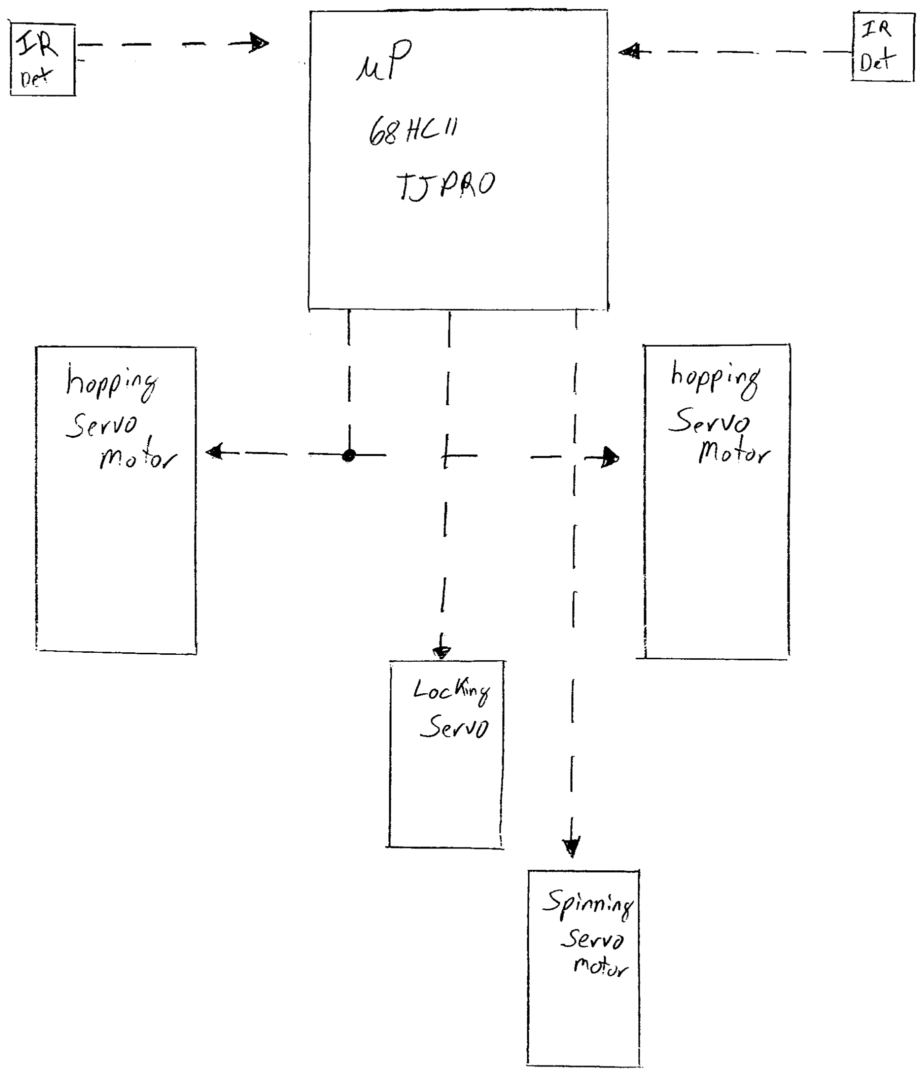
Roo was intended to test the feasibility of hopping as a means of locomotion. Hopping would have benefits including traversal of uneven ground, stairs, outdoors, as well as indoors. We must try to conquer all means of motion in search for the best for a specific task. The next mars rover may hop. Hopping may lead use to fast mean by which robots may move. The ability to climb stairs would be an grand feat in itself.

This project was to cover the actual action of hopping with a few behaviors to demonstrate its abilities. The mechanics of the hopping system will be discussed in detail along with the hardware and soft ware needed to accomplish the all might hop.

Integrated System

The integrated system consists of the TJ Pro board, two IR detectors, two 50 oz/in hopping servomotors, one locking servo, and one spinning servomotor. The IR detectors are the only input to the system. They do a self calibration and have a threshold value by which the 6811 decides which way to spin, using the spinning motor. After an appropriate position has been set then the system goes into hopping mode. The locking servo locks and the hopping servomotors wind the rubber band. Then the locking servo unlocks causing the hop. It then returns to searching mode.


Spin
IR > Threshold
F
T
Go toword IR
decrement Speed
Speed < threshold
F
T
lock and wind Rubber band
un lock (causes hop)


IR Det
μP
68HC11
TJPRO
IR Det
hopping Servo motor
hopping Servo Motor
Locking Servo
Spinning Servo motor

Mobile Platform

My object is to have a hopping robot and the platform is the key element. I built a total of three platforms during development. The first platform was a test platform for the development of the hopping mechanics.

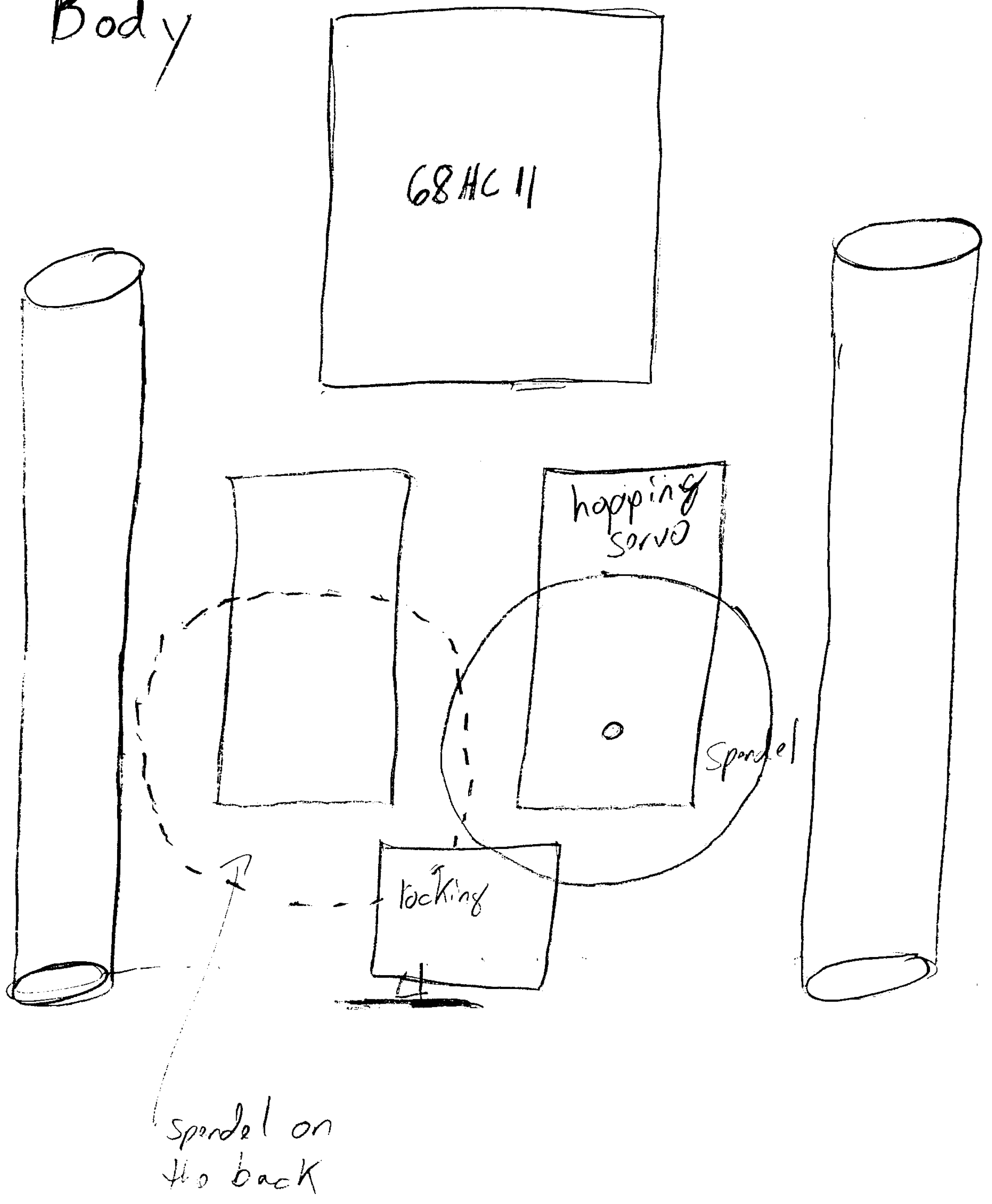
First, I tried to use a lever to stretch the rubber band. The lever was 4 inches long so gave me a total stretch of 8 inches the problem was that the servos were not strong enough. Next, I created a spindle system. Doing this I had to hack the expensive servos so it took a lot of time to build up to this. This spindle system is much better. There are no calculation. In the lever system, the lever arm length was a major factor. The spindle just wound until it could not wind any more. The body had to be rebuilt because the cable when through a loop the made a 90 degree turn therefore it lost a lot of power. So, the final body was two servomotors at the bottom of the body that wound up rubber bands connected to the top of the legs.

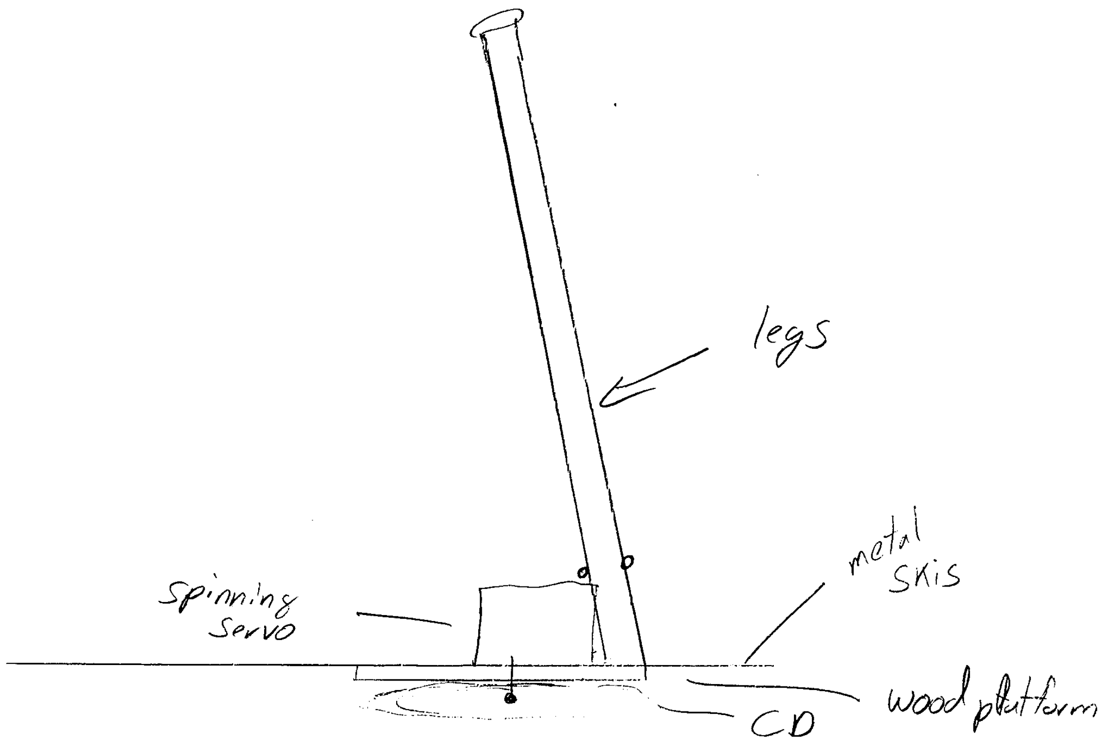
The body also has a locking servo at the bottom of the body. When the body is in the lowered position, there are two bars that the locking servo can catch.

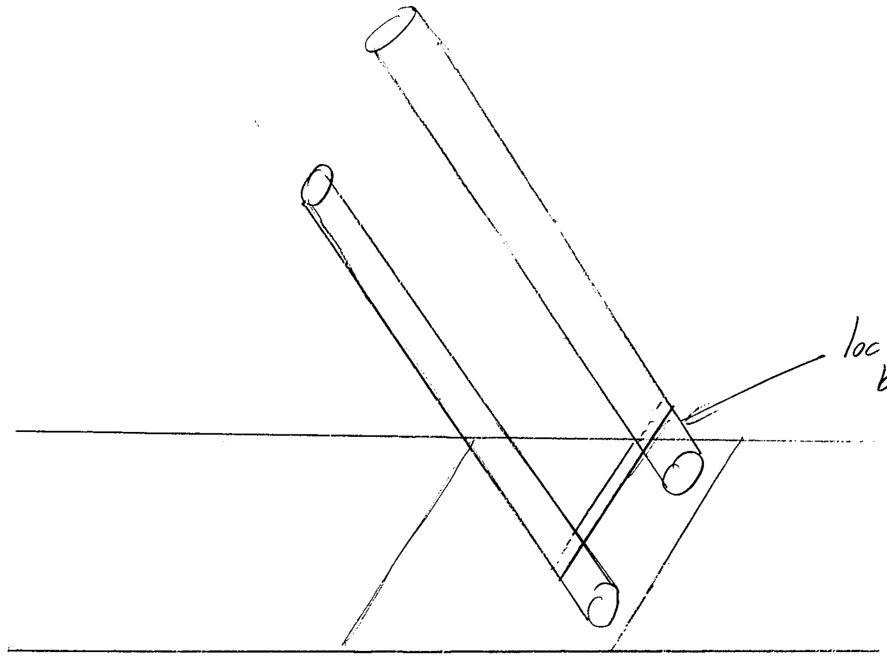
The legs went through several evolutions. The first was a 6 inch diameter circle platform with two legs go up at a 15 degree offset form vertical. The platform had a servo attached so that the tip was just lower than the bottom of the platform where a CD was glued on to the servo mount. This allowed the platform to spin.

The final platform is the circular platform with two skis about 8 inches in front and 4 inches in back. The legs are now at a 10 degree offset of vertical. The original platform was not strong enough and fell over to much.

Body




legs
metal SKis
spinning servo
wood platform
CD


locking bars

Sensors

A sensor is needed that can detect changes in the angle to the direction of the acceleration of gravity. Analog Devices makes a device that will detect acceleration. Using the acceleration of gravity as a constant, this device can relay its angle with respect to gravity.

The advantages of using an accelerometer are:

Signals are filterable.
The device is inherently analog.
It has an adjustable range.
The accelerometer has adjustable sensitivity.
It has adjustable center value.
The unit has only one active axis.
It is small and light.

Using an ADXL05 from *Analog Devices* and an evaluation board from the same, the construction of the tilt sensor is simple. The evaluation board is used to make the addition of the other resisters and cap's easier. The calculations for the value outputted by the accelerometer are made easier by the diagram and chart in material given by *Analog*. In this case, the full-scale is equal to +-1g . To make the assembly easier the optional scale factors are avoided therefore the value for R1 = 30.0 K Ohms and the value for R3 = 301 K Ohms. The accelerometer is a cheap and easy way to measure tilt. It is also more accurate in most case to the alternatives. The features of adjustable sensitivity and range combined with its filtering capability make this sensor a useful tool for none-wheeled autonomous machines.

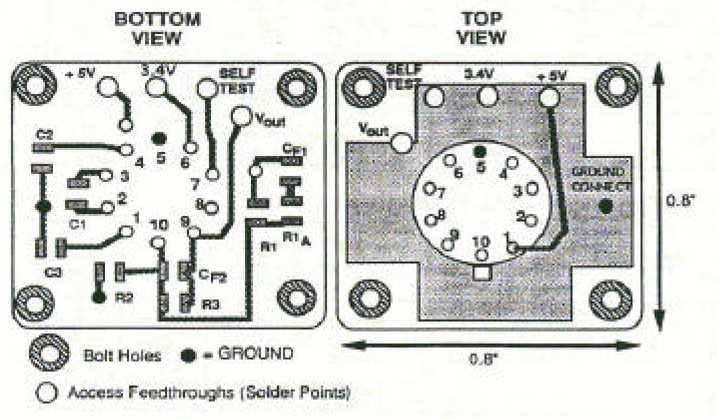
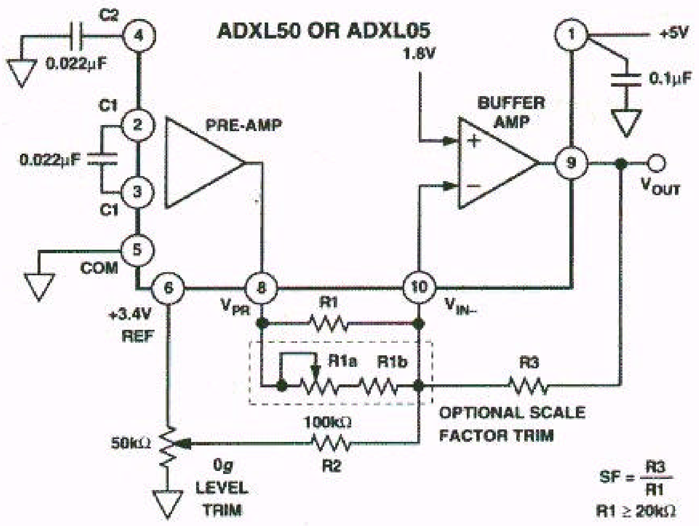


FIGURE 1. ADXL EVALUATION BOARD LAYOUT



ADXL05 0*g* TRIM ONLY, RECOMMENDED COMPONENT VALUES FOR VARIOUS OUTPUT SCALE FACTORS

FULL SCALE	mV PER *g*	R1 kΩ	R3 kΩ
±1*g*	2000	30.1	301
±2*g*	1000	40.2	200
±4*g*	500	40.2	100Bœ
±5*g*	400	49.9	100

ADXL05 WITH 0*g* AND SF TRIMS

FULL SCALE	mV PER *g*	R1a kΩ	R1b kΩ	R3 kΩ
±1*g*	2000	10	24.9	301
±2*g*	1000	10	35.7	200
±4*g*	500	10	35.7	100
±5*g*	400	10	45.3	100

ADXL50 0*g* TRIM ONLY, RECOMMENDED COMPONENT VALUES FOR VARIOUS OUTPUT SCALE FACTORS

FULL SCALE	mV PER *g*	R1 kΩ	R3 kΩ
±10*g*	200	23.7	249
±20*g*	100	26.1	137
±40*g*	50	39.2	105
±50*g*	40	49.9	105

ADXL50 WITH 0*g* AND SF TRIMS

FULL SCALE	mV PER *g*	R1a kΩ	R1b kΩ	R3 kΩ
±10*g*	200	5	21.5	249
±20*g*	100	5	23.7	137
±40*g*	50	10	34.0	105
±50*g*	40	10	45.3	105

Figure 4. Typical Component Values for Circuit with External 0 g or 0 g and Scale Factor Timing

Behaviors

Roo searches for IR by spinning. When it senses high levels of IR, it will decrease speed and try to face the source. If no IR is present then Roo will spin right for 6 sec then left for 6 sec then hold. In the hold position , shutoff mode is activated and all the servos are turned off so that less power is wasted. Self calibration for its IR detectors is done on startup. The offsets for the IR detectors are calculated from the ambient light. Roo will hop when it has centered on an IR source.

Conclusion

The actual jumping procedure took little power as compared with conventional methods of locomotion. Although Roo was a success, the design had major flaws. The structure was too weak to be ran continuously. There were problems with 68HC11 board withstanding the shock. The platform will not bear jumping repeatedly. Also, the body was could not handle the power delivered by the hopping servomotors. If the system was redesigned in a more efficient manner then robot could achieve much. Roo kicked ass.

List File

```
lib_rw10.c
twoservo.icb
twoservo.c
serialir.icb
serialir.c
roo.c
```

Code

```
int rightIR = 0;
int leftIR = 0;
int rightOffIR = 0;
int leftOffIR = 0;
int threshold = 15;
int tighten = 0;
int down = 0;
int jump = 0;
int spinDir = 0;
int wind  = 1;
int PID;
int done = 1;
int motorSpeed = 0;
int minSpeed = 50;

int main()
{
        servo_on();
      servo_deg1(92.0);
      servo_deg2(0.0);
        start_process(readSensors());
      wait(1000);
      rightOffIR = rightIR;
      leftOffIR = leftIR;
        wait(3000);
      start_process(routine());
      start_process(actionServo());



return 1;
}

void readSensors(){
        while(1){
                leftIR = analog(2)-leftOffIR;/**/
                rightIR = analog(1)-rightOffIR;/**/
                if(analog(0)>10){down=1;}/**/
                defer();
                }
}

void actionServo(){
        while (1){
              if(down && !jump){servo_deg2(90.0);}/**/
            else{servo_deg2(0.0);}/**/

                if(wind) servo_deg1(180.0);
                else if(!wind) servo_deg1(0.0);
      defer();
        }
}

void spinDirection(){
        spinDir=1;
      wait(6000);

      spinDir=0;
      wait(500);

        spinDir=-1;
      wait(6000);

      while(1){
            spinDir=0;
            defer();
            }
}

int search(){
        servo_off();
        PID = start_process(spinDirection());
        done=0;
        motorSpeed = 80;

      while(!done){

                if(rightIR>leftIR+3 )motor(0,motorSpeed);
                else if(leftIR>rightIR+3 ) motor(0,-motorSpeed);
                else if(rightIR>threshold)motorSpeed=motorSpeed-1;
                else motor(0,spinDir*80);
                if(motorSpeed<minSpeed){
                  done=1;
                  motorSpeed=0;
                  motor(0,motorSpeed);}
            defer();
      }
        motorSpeed=0;
      motor(0,motorSpeed);
      kill_process(PID);
        servo_on();
      return 1;
}

void routine(){/**/
      jump=0;/**/
        wind=1;/**/
      wait(2000);/**/
        wind=0;/**/
      wait(2000);/**/
        while (1)
        {
            jump=0;
                wind=1;
                search();
                wait(4000);
                jump=1;
                wait(500);

                jump=0;
                wind=0;
                search();
                wait(4000);
            jump=1;
                wait(500);/**/
        }
}

void wait(int m_second){
       long stop_time;
       stop_time = mseconds() + (long)m_second;
       while(stop_time > mseconds())
             defer();
```

```
}
```