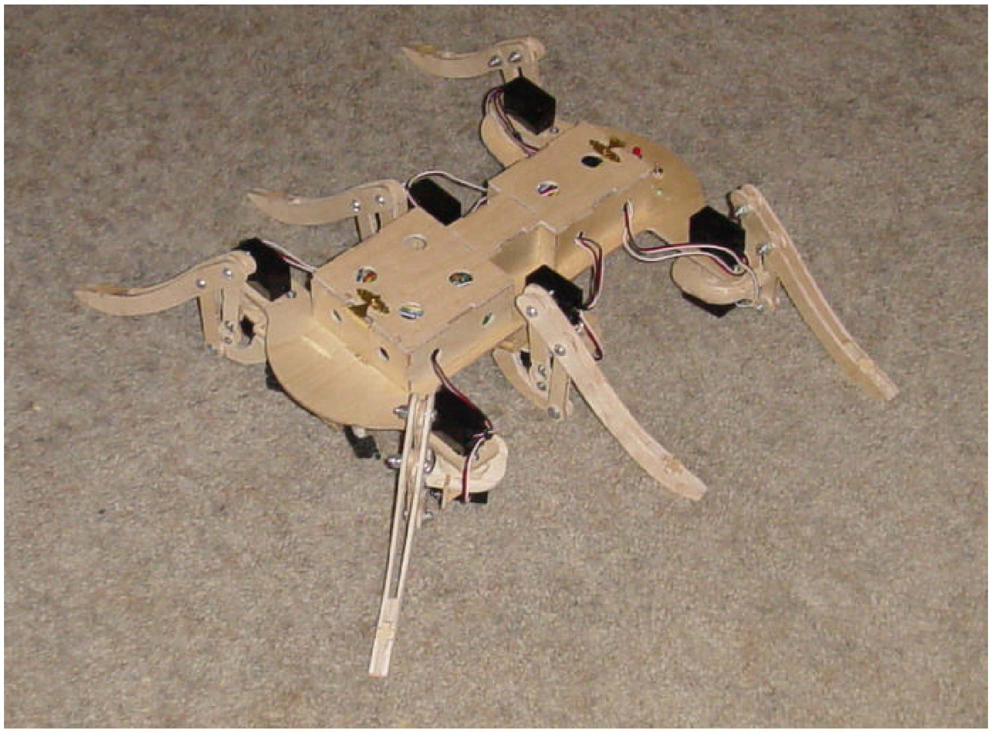


SEIS PIERNAS

THE SIX LEGGED STEP CLIMBING SPIDER (SPIDY)

Amit David Jayakaran

DECEMBER 7, 2002

University of Florida
Department of Electrical and Computer Engineering
EEL 5666
Intelligent Machines Design Laboratory

Table of Contents

Abstract

The following paper is a description of the work done to create a step climbing spider, Spidy. Spidy has six legs to achieve this. It wanders around looking for obstacles and then checks to see if the obstacle is a step or not. In case it is not a step it turns around walks off. If it finds a step it tries to climb it. It makes an attempt to climb the step and in case it can’t it gives up and walks away.

Executive Summary

The goal of my project was to make a step climbing spider. Spidy physically has six legs with every leg having 2 degrees of freedom. It walks around in a random fashion around an area and looks for obstacles. When it locates an obstacle it checks to see if it is a stair. If it sees a stair it attempts to climb it. If it doesn't see a stair it turns and walks away. The robot has 12 servos, two on each leg that gives each of them two degrees of freedom. The robot uses IR to detect obstacles and then checks to see if it can climb the obstacle, i.e. it checks to see if it is a stair. It does this using IR, too. Once it has decided that it can climb it aligns itself and tries to climb the step. It uses an accelerometer to judge if he has made it up the step. In case he hasn't, he backs up, and then attempts twice more before giving up.

The brain of the robot is the Atmel Mega323 microcontroller mounted on the MegaAVR Development Board by Progressive Recourses. An expansion board was built on Protel to attach the servos and the sensors.

The sensors consist of IRs to detect obstacles and also to judge f there is a step. In addition there is an accelerometer which checks the inclination of the robot with the horizontal to check if the robot has fallen over or not, indicating that it made the step or not.

Introduction

Two previous attempts have been made at the step climbing robot but both have not been successful. I felt this was a bigger challenge than a wheeled robot where implementation of motion is the easiest. Originally I planned to use an expired patent [1] by my advisor, Dr. Carl Crane III.. The work that he had put in was only theoretical and never practically implemented due to several physical implementations. After several modifications I decided to go with a fresh design based on the Robobug. I decided to go with six long legs making the body as wide as possible. Being a mechanical engineer I had to first deal with trying to first understand the microcontroller. I needed the servos to all be directly controlled by the microcontroller in order to avoid using any extra hardware. This basically meant I needed robust software to be able to give the PWMs to the servos and at the same time be able to perform other operations without any degradation in performance.

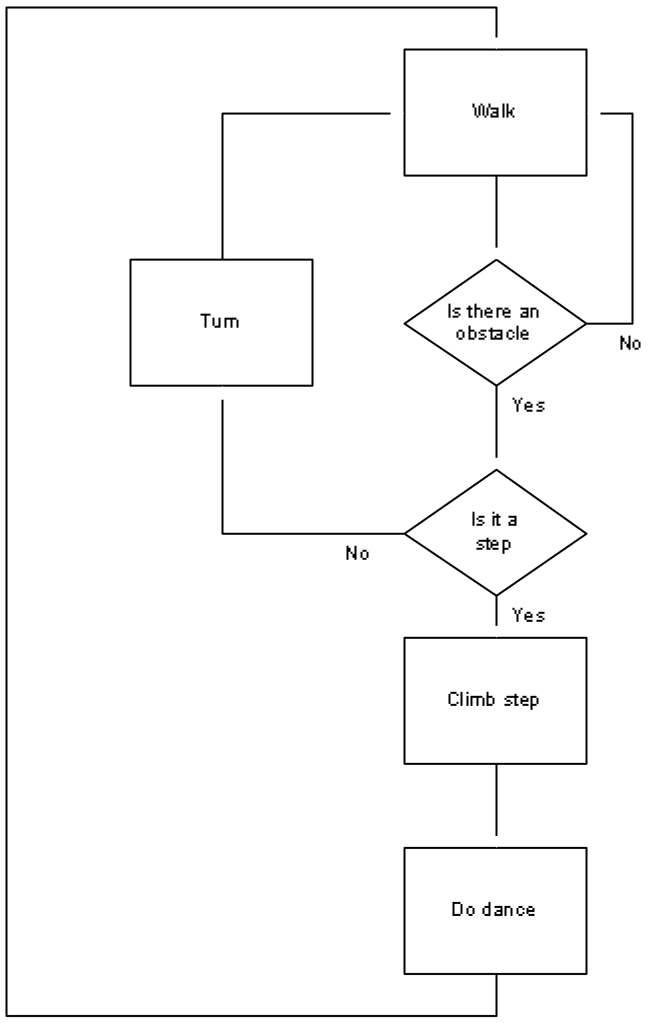
Although a lot of time was spent in developing the platform it was nowhere as close as the time spent in developing drivers and components to control individual components of the system and moreover the integration of all the components.

Integrated System

Spidy’s overall operations are simply to walk around and look for a step and when it finds one it tries to climb it. If it climbs it does a little dance to rejoice and then walks away.

The motion is basically random in nature using obstacle avoidance to get where it wants. It has two IRs that check for obstacles. When it finds an obstacle it lifts itself up and checks to see if it can look over the obstacle using an IR on its belly. If it can see over the obstacle it means that the obstacle is a step. Spidy then attempts to align itself with the step. Once it has done that it walks towards the step and then lifts itself up. It then lifts one leg at a time and tries to get itself over. This is followed a series of complicated steps that finally get it on top. Once on top it takes a couple of steps and then does a little dance and continues on its way.

The whole setup runs off the ATMega323 from ATMEL mounted on a board built by Progressive Resources. Besides this to enable easy connection of the servos and other peripherals I made an expansion board for the servos and the IRs. There are two sources of power. One is a 9V battery that powers the board, IRs and the accelerometer. The other battery pack is a set of 5 1.2V Ni-Cd that can source upto 3600mAh of current.

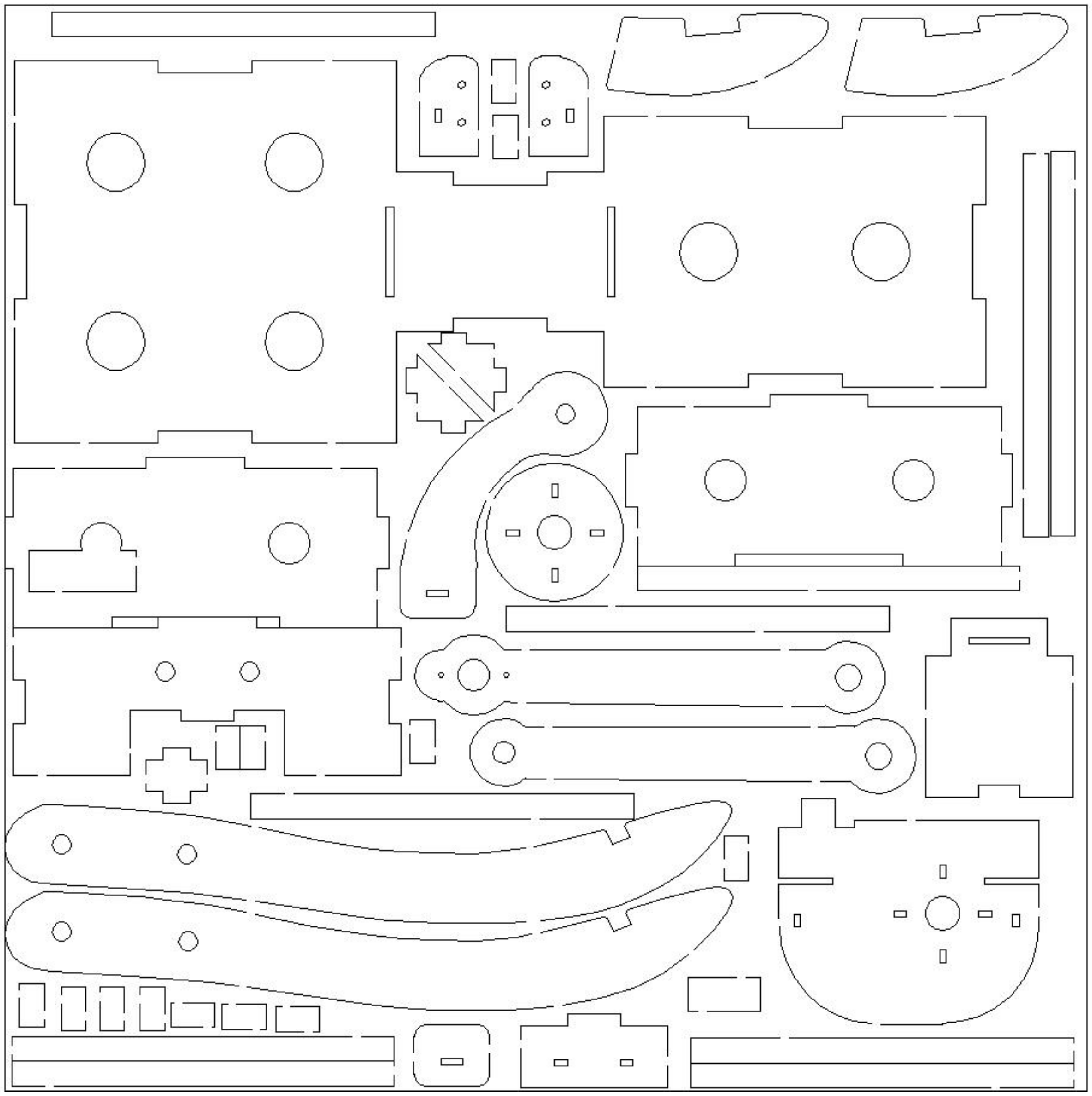

Walk
Turn
Is there an obstacle
No
Yes
Is it a step
No
Yes
Climb step
Do dance

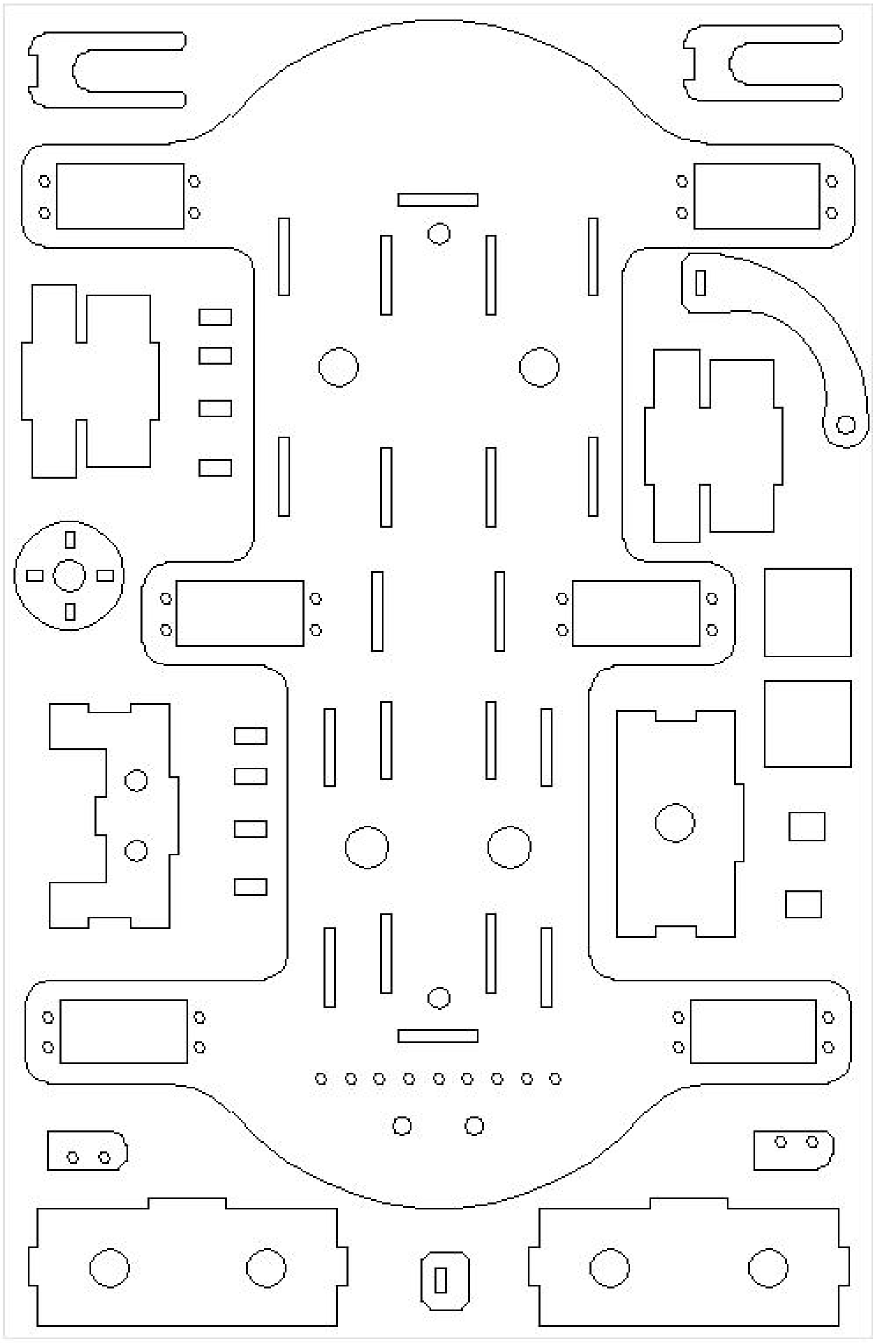
Mobile Platform

Body

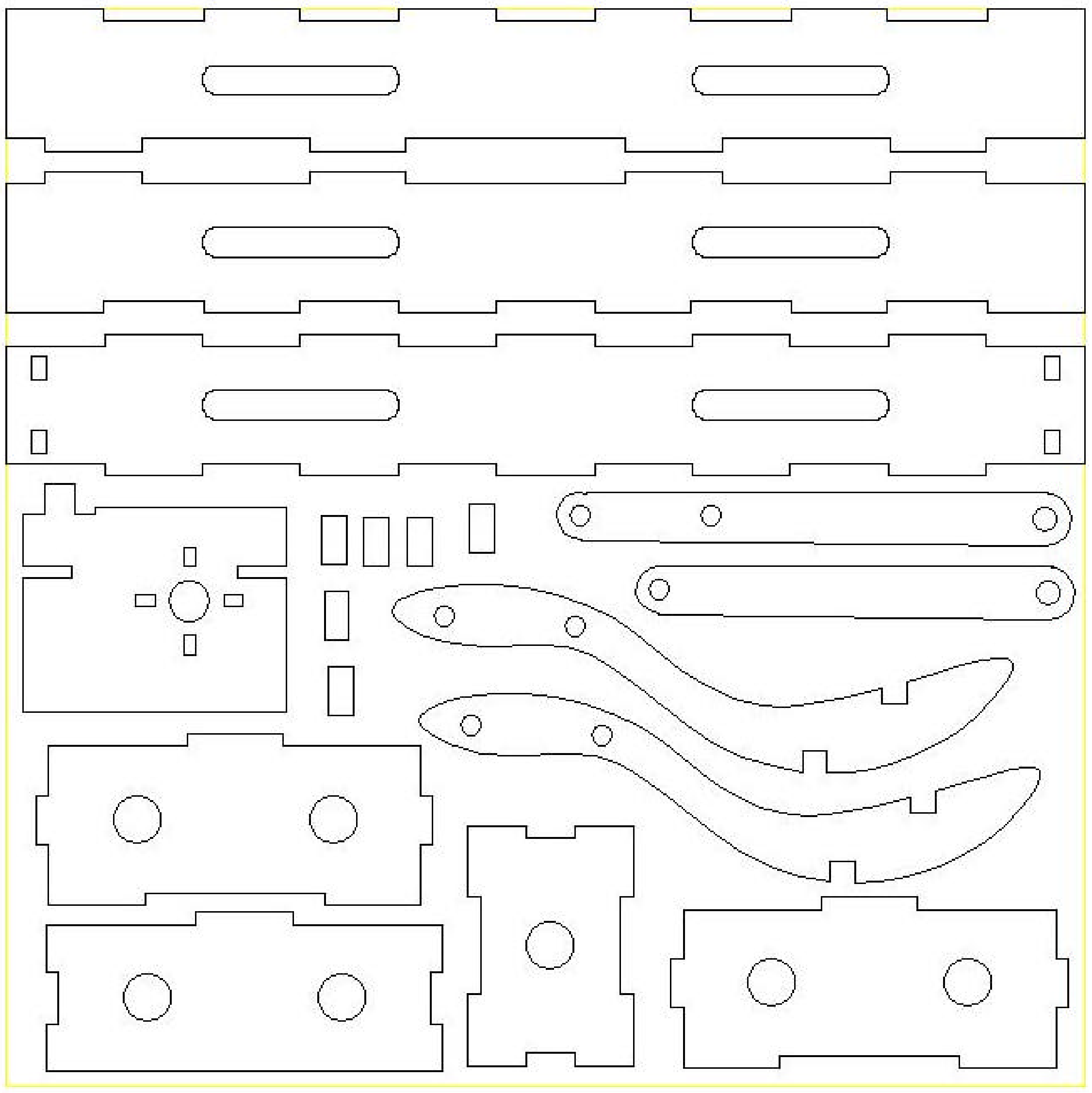
The body is made of the supplied wood, which is 1/8" thick Aircraft Ply. Due to the extra load of the battery packs and the size of Spidy itself I had to use reinforcements to strengthen it against bending and possibly breaking.

Unlike previous walkers I didn't line up the servos on both sides of Spidy. This was done to give the robot more flexibility in motion and also for easier packing up.

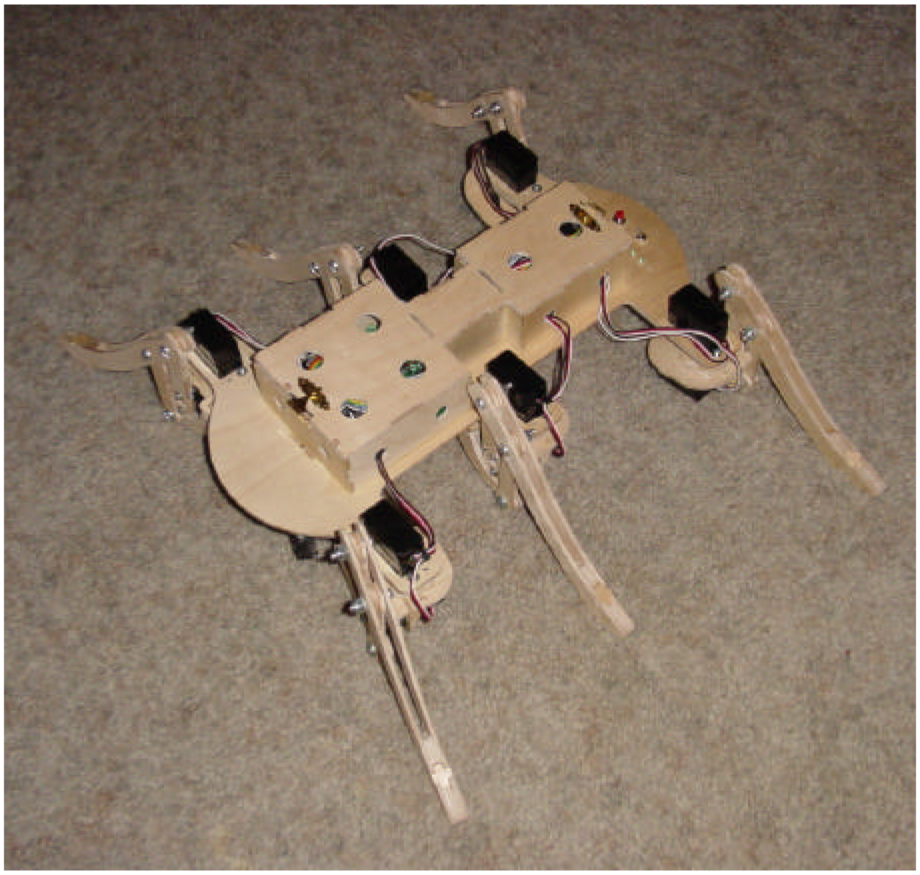

AutoCAD diagrams for body

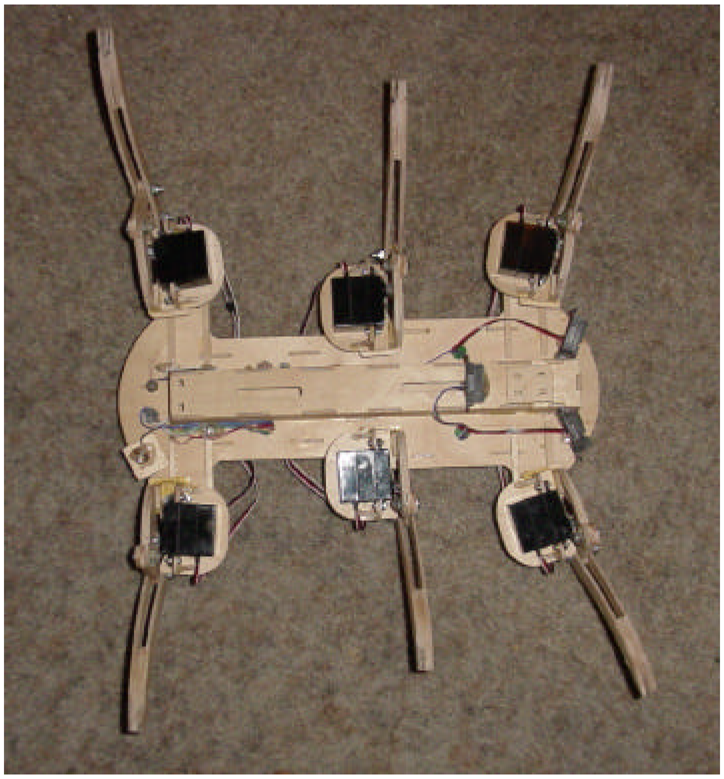


AutoCAD diagrams for body



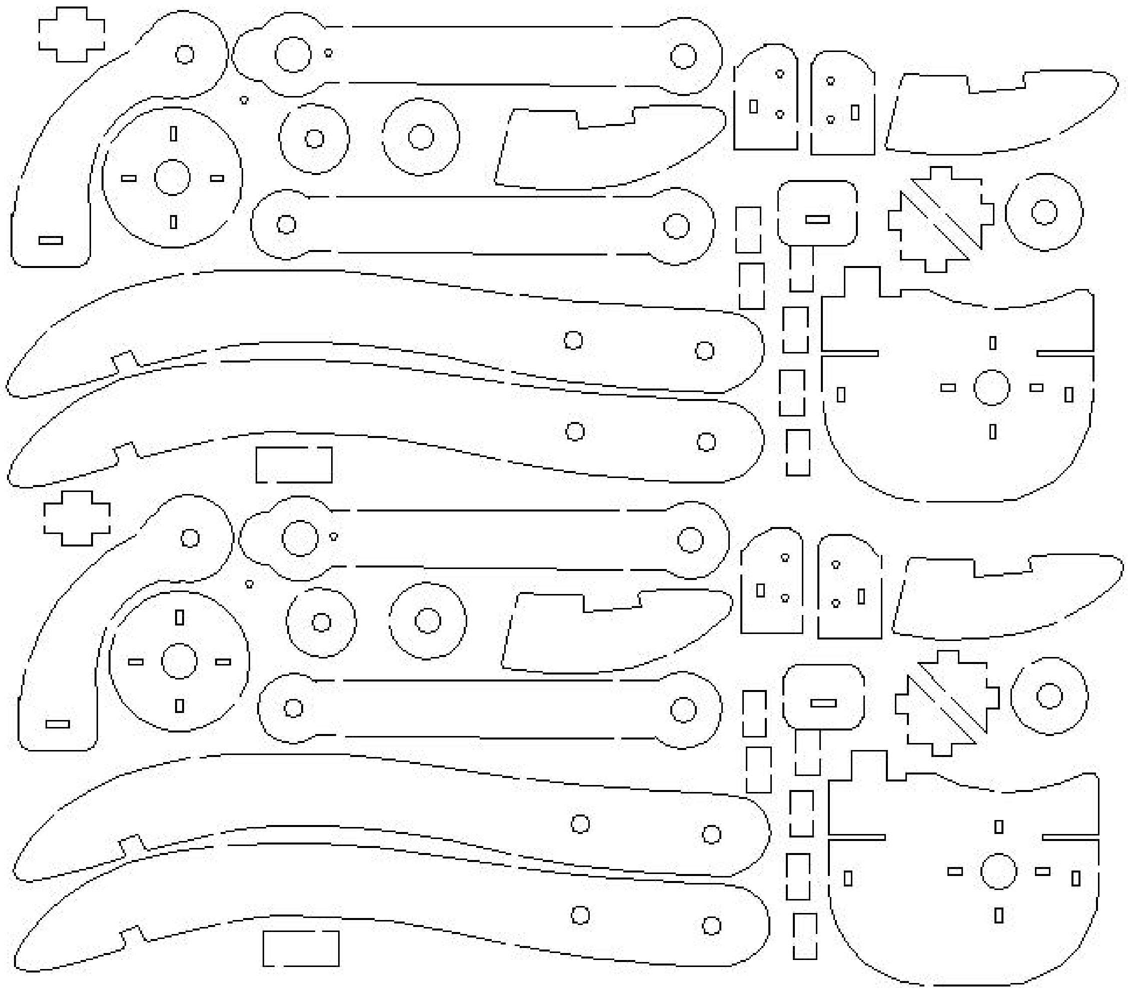
AutoCAD diagrams for body


The robot

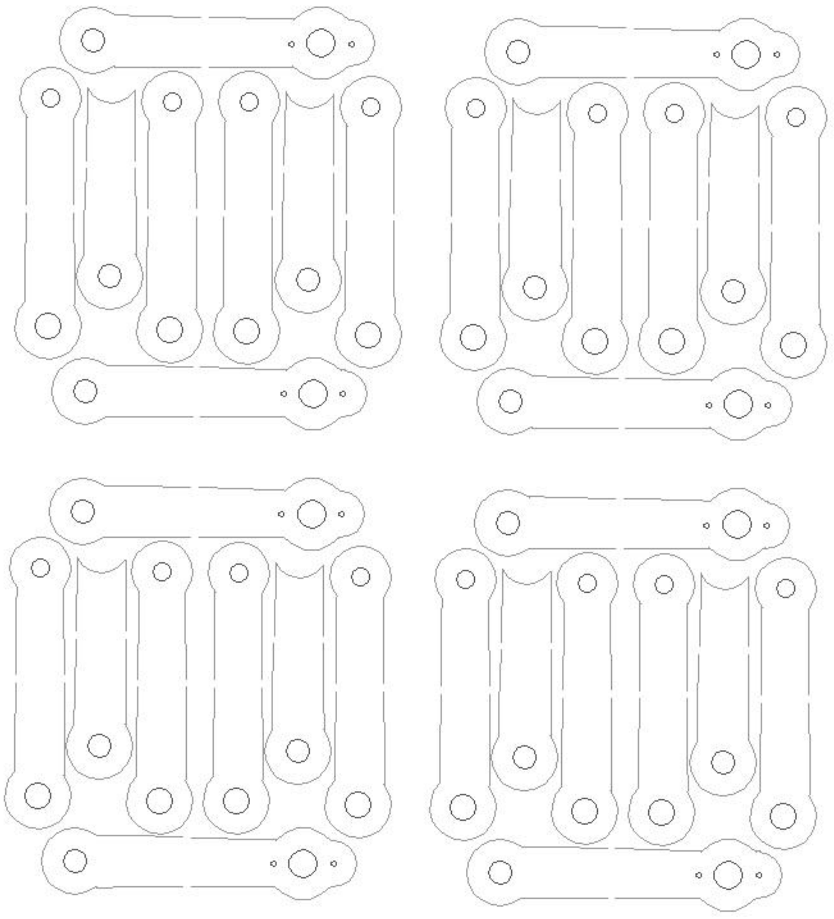

The robot from underneath

Legs

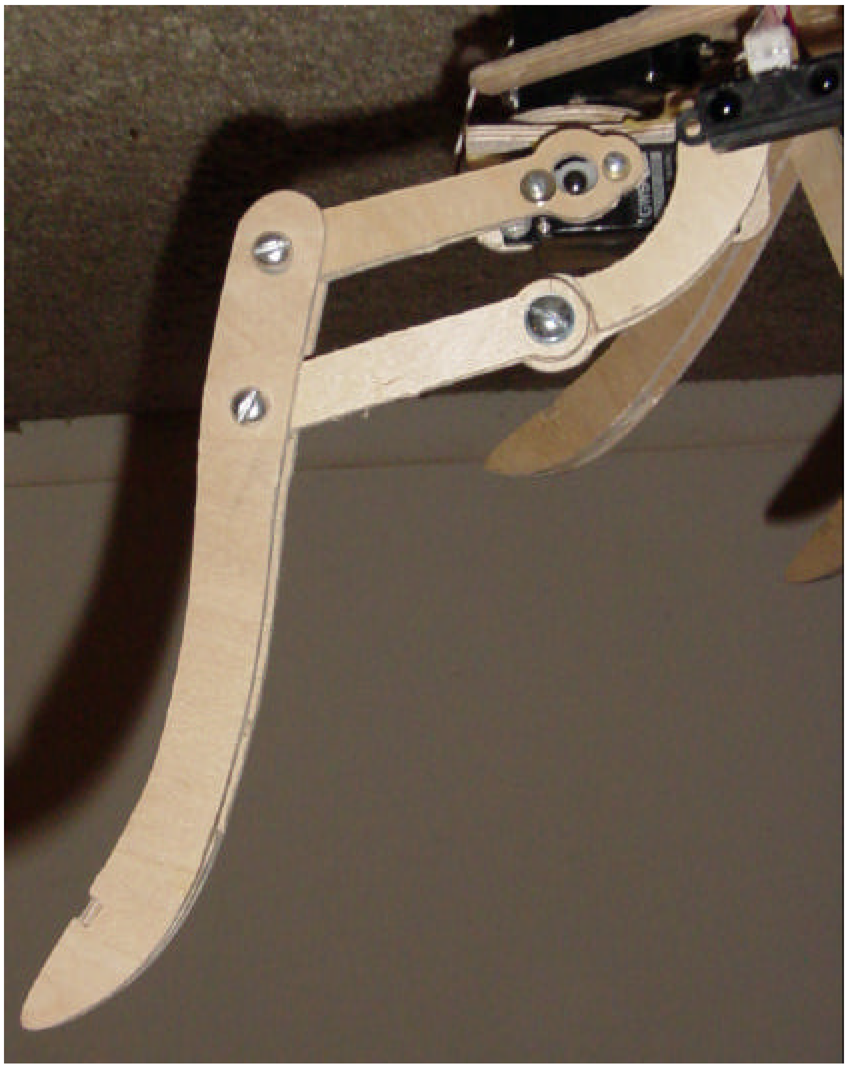
The legs were made long and slender to give Spidy the ability to stand really high. With my design I was able to achieve about 8 inches. The legs were build based on a four bar mechanism like the previous walkers. However Spidy had more leverage to enable it to lift itself high enough. This design was a real test for the servos that were probably built to handle just about the same torque as that was required. This meant that batteries drained fast and performance dropped very soon.



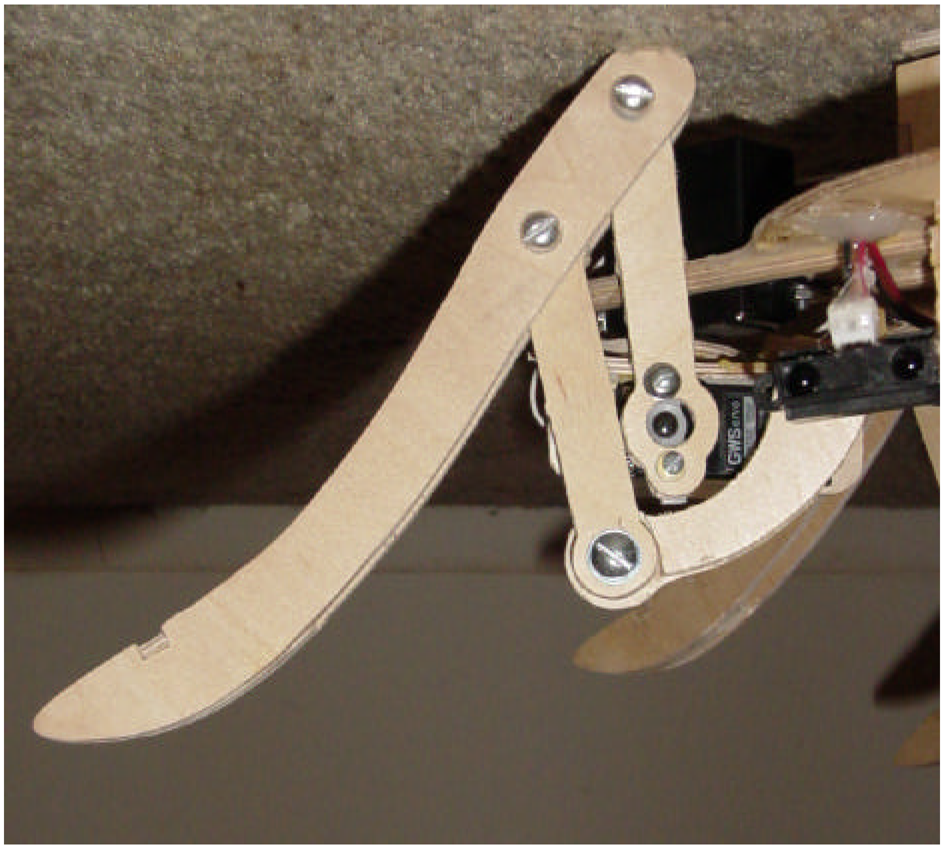
Drawings of legs



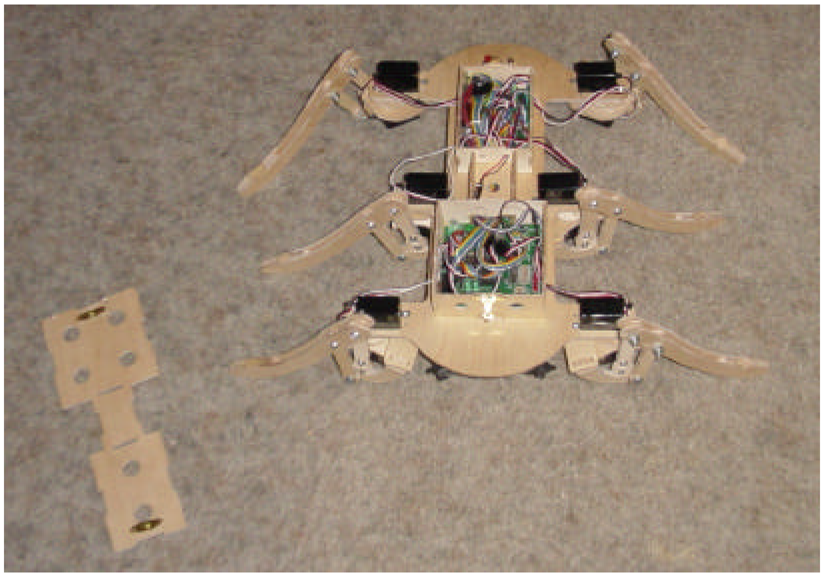
Drawings levers used

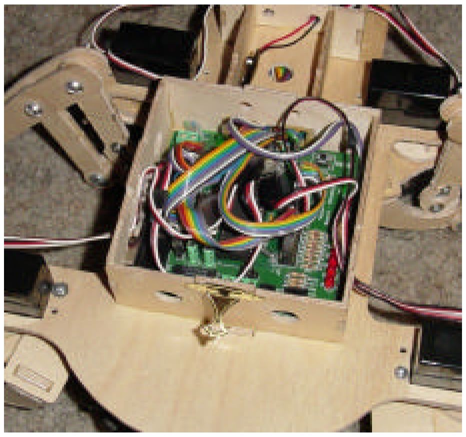


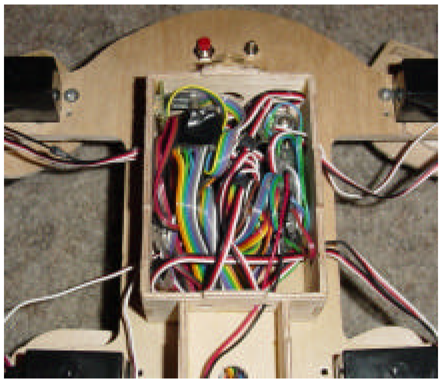
The Leg extended



The leg retracted



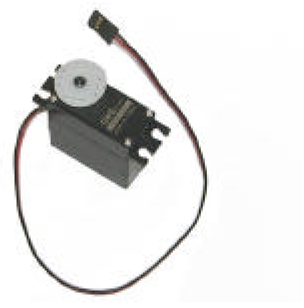




Wiring

Actuation

Actuation of each leg was done with two servos giving each of them two degrees of freedom. Geometrically all the legs were similar varying a bit because they had to be placed on different sides of the body. The specs of the servo are:

High-torque Ball-bearing Servo Motor from GWS		
Dimensions (mm):	1.56 x 0.79 x 1.56 in. 39.5 x 20.0 x 39.6 mm	
Weight:	46 Grams / 1.62 oz.	
Ball Bearings:	Yes, 2BB	
Metal Gears:	No	
Torque (4.8V):	69 oz.in.	
Transit Time (4.8V):	0.21 sec./60°	
Torque (6.0V):	86 oz.in.	
Transit Time (6.0V):	0.17 sec./60°	

The servos that I chose were high torque but gave just about the power required by the walker and were in no way economical in power consumption. The twelve servos drew more current than a 25 A power supply at 5V could provide. Also I've had smoke come out of my battery pack once. The springs that I used to attach the batteries have turned back due to the high power consumption of the servos.

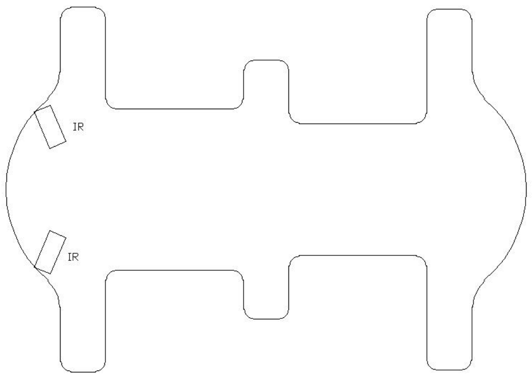
One servo on the leg was used for moving the leg forward and backwards while the other was used for moving it up and down.

Sensors

IR Sensor Suite

There are two IRs in front of the robot that will be used to detect an obstacle in front of the robot. The robot will basically walk in a random fashion until it finds an obstacle. Then once it finds one it will align itself with it so that the front is facing the obstacle. It will then try to stand up as much as possible and see if it can see over the obstacle with its third IR. If this happens it will assume that there is a step present there. And the step climbing routine will start. If not it will back off and wander away in a random direction again.

The IR used is the Sharp GP2D12 Distance Measuring Sensor.



Front sensor layout

The Accelerometer – The special sensor

The accelerometer is a device to detect the acceleration in a given direction. Newer and more sensitive accelerometers can also be used for tilt sensors and that is what I am using my accelerometer for. The accelerometer will detect the orientation of the robot platform with the horizontal. This basically helps put an additional feed-back loop into my system for letting the robot know if it missed a step and force it to back off and try again..

The accelerometer has a sensitivity of ±2g. It is the ADXL202AE mounted on an evaluation board ADXL202EB. The readings are taken digitally through the input capture available on the board. The accelerometer is very cheap and so the values that I get out of it are also not very stable. I had to take about 1000 samples to get an accurate reading that I could use!

Lynette Miller, a student of the MIL, helped me design the foot print of the accelerometer based on the ADXL202EB from Analog Devices

Behaviors

The robot walks around in a random fashion looking for walls. When it finds one it stands up and tries to look over it. If it can it means that it has found a step. In case it can’t it turns away and walks off. In case it sees over the obstacle it goes in a step climbing routine. The routine is very simple in that it turns right and left depending on what the IR’s tell it the distance each one of them are from the step. Once it is aligned it moves towards the step and pushes itself towards it and then lifts himself up. It can goes through a number of steps to maintain stability and at the same time achieve the task at hand. It does a pair of legs at a time. When it finally clears the step it does a dance.

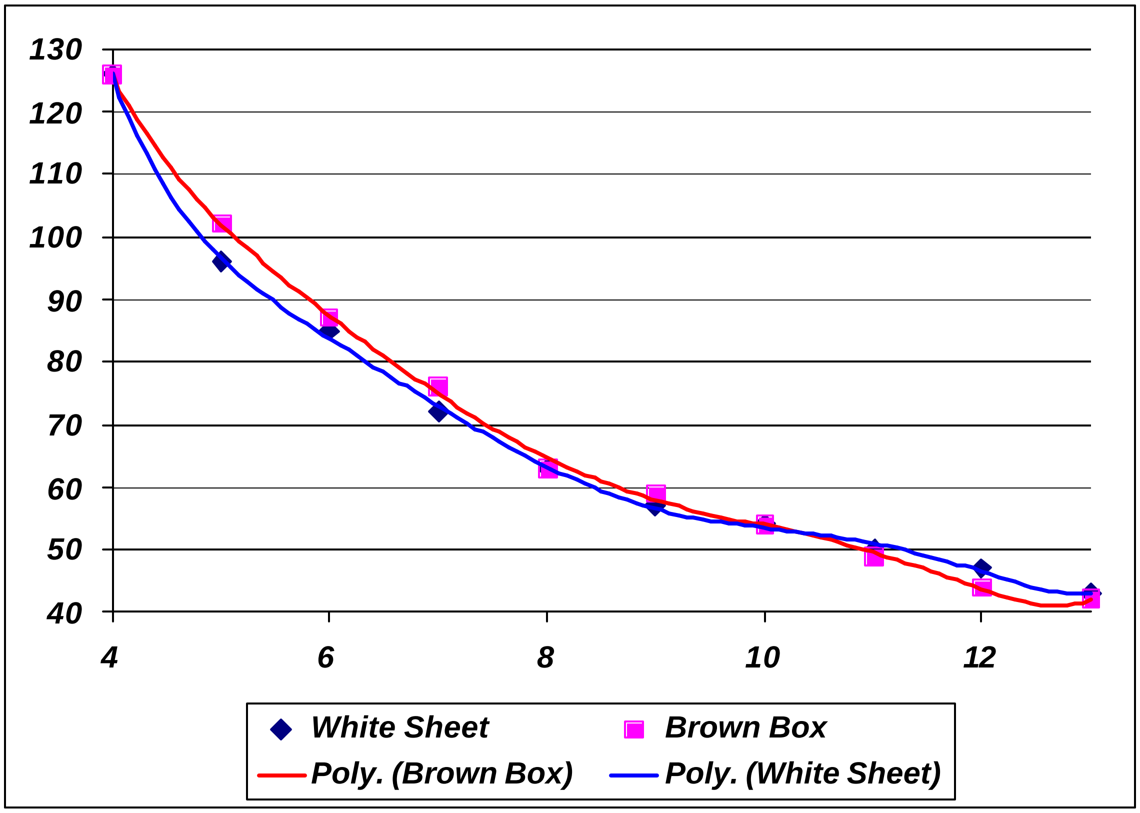
In case the robot is not able to align itself to the step it may sometimes miss a step. In case it does the accelerometer detects this and backs the robot up a couple of steps and then retries the step. It will try this twice before giving up.

Experimental Layout and Results

IR sensors

To test the IR sensors I wrote a small code to display values on hyper terminal on my computer (Appendix B). The IR sensor gave almost consistent results between experiments and almost matched the specifications of the manufacturer.

Distance	**Value**	
	White sheet	**Brown box**
4	126	126
5	96	102
6	85	87
7	72	76
8	63	63
9	57	59
10	54	54
11	50	49
12	47	44
13	43	42
∞	10	12

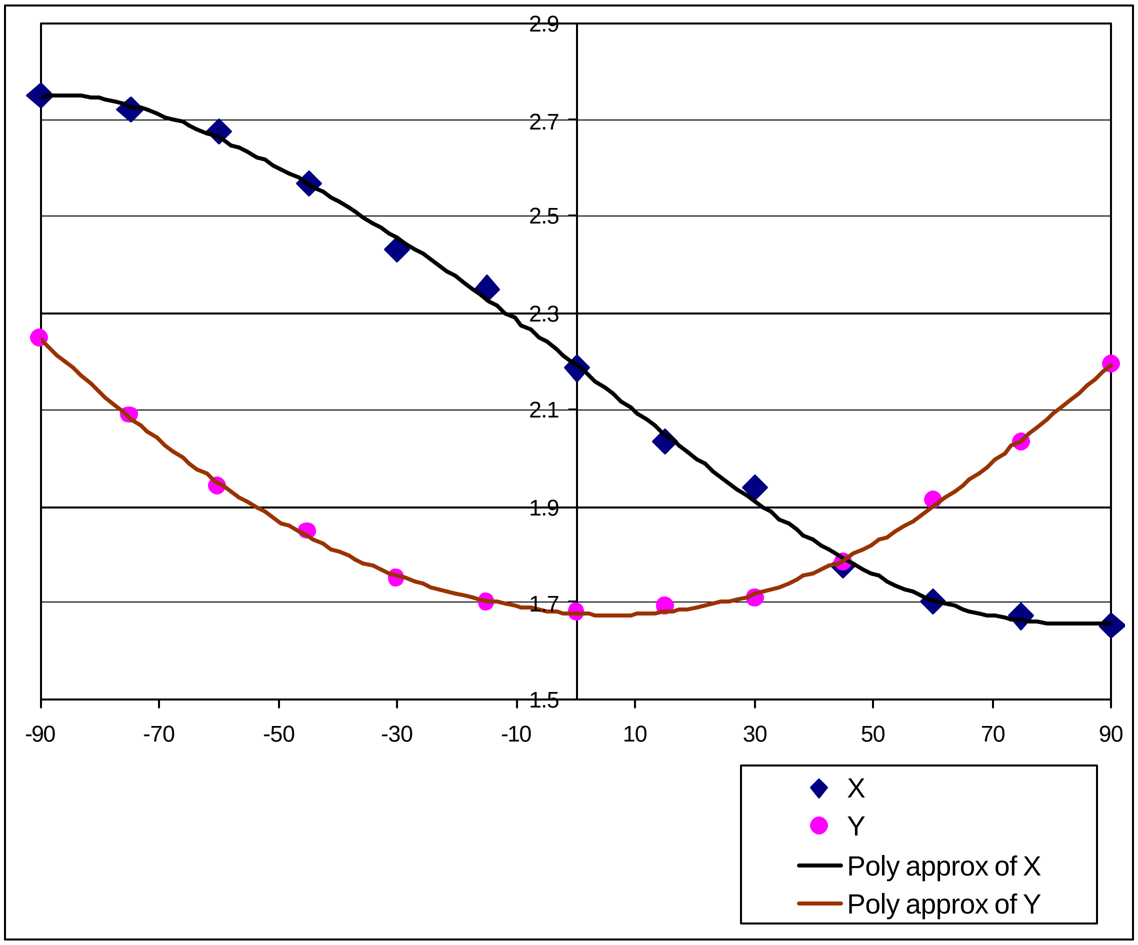

130
120
110
100
90
80
70
60
50
40
4
6
8
10
12
White Sheet
Brown Box
Poly. (Brown Box)
Poly. (White Sheet)

IR Readings

Accelerometer

The accelerometer has two outputs and the outputs are in the form of a PWM with a duty cycle of about 50% for 0g in each direction. Below is a table of values that were obtained from experiments that I conducted keeping the accelerometer at different angles. The values are lengths of the duty cycles in ms. (Code in Appendix B)

Angle	**Length** in ms	
	X	**Y**
-90	2.75	2.245
-75	2.719	2.088
-60	2.676	1.94
-45	2.57	1.85
-30	2.434	1.75
-15	2.35	1.7
0	2.186	1.68
15	2.035	1.695
30	1.939	1.709
45	1.775	1.782
60	1.703	1.913
75	1.671	2.033
90	1.656	2.193



Accelerometer Readings

Conclusion

Overall I am very satisfied with the output of the course, Spidy stood up to almost all that I expected it to. The flexibility in the system was the biggest draw back but I managed to overcome most of it through software.

When I started this course, practically everything was new to me. I had to figure a lot of stuff all by myself including trivial stuff like common ground. I enjoyed working with the microcontrollers because of the challenge they present when trying to debug.

Writing a lot of modular code really helped figuring out loose connections and damaged peripherals and also in calibration.

Most of all I really enjoyed watching and working along other people in their coding because that really gave me a very wide perspective of the capabilities of microcontroller and sensors available out there.

If I were to do Spidy again, I'd design him with higher power servos with metal gear heads and make him longer and more rigid. It was a real challenge to try and deal with hardware inadequacies through software.

Documentation

[1] Crane, III; Carl D., Haukoos; Dana D. “Hybrid robotic vehicle” United States Patent 4,977,971, May 17th, 1989

[2] Data Sheets on the ATMega323

[3] David Novick, Keith L. Doty. *ROBOBUG Assembly Manual*; Mekatronix 1999

Acknowledgments

- The TA's Uriel Rodriguez and Jason Plew for their invaluable guidance
- Dr A. A. Arroyo and Dr. Eric M. Schwartz for their guidance
- Chris Preston for helping me debug my robot
- Lynette Miller for helping me with the footprints of the accelerometer

Sources of parts

- Mega 323 Board from PRLLC.com
- Wood from IMDL
- Servos from Mark III robots
- IRs from Mark III robots
- Accelerometer from Analog Devices
- Cabling, jumpers, headers, etc from the IMDL lab
- Batteries from www.batterystation.com

Appendices

Appendix A

Main code for program

Robot.c

```
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>

typedef unsigned short u16;
typedef unsigned char  u08;

#include "ADC.h"
#include "InputCap.h"
#include "Delay.h"
#include "motors.h"
#include "constant.h"
#include "roboops.h"
#include "roboops2.h"



int main(void)
{
        //Variables used in Main
        u08 hieght;
        //Initialize the servos and analog ports

        init_motors(14);
        ADC_init();
        walk_init();

        //Delay before starting program
        delay(0x0FFF);

        sei(); //Set global interrupt enable

        //command robot to stand
        stand();
        delay(0x1FFF);

        while(!tilt_forward());
        stand_high(90);
        delay(0x0FFF);

        stand();

        for(;;)
        {


                //Take a step forward
                walk_six(0x01FF,0x00FF);

                //detect hieght of obstacle
                hieght=obstacle();

                //Check if ti needs to climb
                if (!(hieght==0))
                {
                        stand();
                        align();
                        climb_step(hieght);
                }

        }
}
```

ADC.h

```
void ADC_init(void)
{
	DDRA=0;
	outp((1<<ADEN) | (1<<ADPS2) | (ADPS1), ADCSR); //Initialize to use 8bit
resolution for all channels
}


u08 ADC_getreading(u08 channel)
 {
	u08 temp_valueH;


	outp((1<<REFS0)|(1<<REFS1)|(1<<ADLAR), ADMUX);  //use 4.95V as
reference voltage

	ADMUX=ADMUX & 0xF8;
	ADMUX=ADMUX | channel;


	sbi(ADCSR, ADSC);

	loop_until_bit_is_set(ADCSR, ADIF);			//wait till conversion is
complete

	temp_valueH = inp(ADCH);

	sbi(ADCSR, ADIF);

	ADMUX=0;

	return temp_valueH;

}


u08 Left_IR(void)
{
	return (ADC_getreading(0));
}


u08 Right_IR(void)
```

```
{
	return (ADC_getreading(1));
}


u08 Top_IR(void)
{
	return (ADC_getreading(2));
}
```

constants.h

```
#define Leg1U 87+35
#define Leg1L 60+30+10

#define Leg2U 93-33
#define Leg2L 134-25-10

#define Leg3U 95
#define Leg3L 92+20+10

#define Leg4U 99
#define Leg4L 88-20-10

#define Leg5U 93-20  //because of breaking
#define Leg5L 85-10-10

#define Leg6U 104+20
#define Leg6L 82+10+10

#define LIFT_UP 25
#define MOVE_FW -25
#define MOVE_FW_SLOW -15
#define LIFT_UP_CLIMB 50
#define MOVE_FW_CLIMB -25

u08 climb_trials=0;
```

delay.h

```
void delay(u16 delay_time)
{
	do
	{
		u08 i=0;
		do
		{
			asm volatile("nop\n\t"
			  "nop\n\t"
			  "nop\n\t"
			  "nop\n\t"
			  ::);
		} while(--i);
	} while(--delay_time);
}
```

inputcap.h

```
unsigned volatile short Cap_value1,Cap_value,Cap_Average;
unsigned volatile char Cap_cnt;

void InputCap_start(void)
{
        sbi(TIMSK,5); //Enables the T1C0 overflow interrupt
        outp(0,TCNT1H); //start value of timer variable
        outp(0,TCNT1L);
        outp(0,TCCR1A); //No compare/capture/PWM
        outp(65,TCCR1B); //prescale
        Cap_value=0;
        Cap_cnt=10;
        Cap_Average=0;
}

void InputCap_stop(void)
{
        cbi(TIMSK,5);
        //outp(0,TIMSK); //Enables the T1C0 overflow interrupt
        outp(0,TCNT1H); //start value of timer variable
        outp(0,TCNT1L);
        outp(0,TCCR1A); //No compare/capture/PWM
        outp(0,TCCR1B); //prescale
        //Cap_value=0;
}

SIGNAL (SIG_INPUT_CAPTURE1)
{

        u08 lo_val, hi_val;

        lo_val = inp(ICR1L); /* read low byte first */
        hi_val = inp(ICR1H);

        if (TCCR1B==65)
        {
                Cap_value1=lo_val+hi_val*256;
                outp(0,TCCR1A); //No compare/capture/PWM
                outp(1,TCCR1B); //prescale
        }
        else
        {
                Cap_value=(lo_val+hi_val*256)-Cap_value1;
                outp(0,TCCR1A); //No compare/capture/PWM
                outp(65,TCCR1B); //prescale
                outp(0,TCNT1H); //start value of timer variable
                outp(0,TCNT1L);

                Cap_Average=Cap_Average+Cap_value;


                Cap_cnt=Cap_cnt-1;
        }



}

u16 get_angle(void)
{
        //Code to ensure that the input capture doesn't interfere with the motor PWMS.
This code
        //will delay the input capture angle finding procedure till it feels it has sufficient
time
        //to perform it
        u16 avg=0;
        u08 cnt;

        for (cnt=0;cnt<10;cnt++)
        {

                while (TCNT2>20)
                {}
                InputCap_start();
                while(Cap_cnt>0)
                {}
                InputCap_stop();
                avg=avg+Cap_Average/16;
                Cap_cnt=10;
                Cap_Average=0;
        }

        return(avg/16);

}


u08 tilt_forward(void)
```

```
{
u08 cnt=0,cntd=0;
u16 val;
        for (cnt=0;cnt<20;cnt++)
        {
                val=get_angle();
                if (val<797)
                        cntd++;
        }

if(cntd>15)
        return (1);
else
        return (0);

}


u08 tilt_back(void)
{
u08 cnt=0,cntd=0;
u16 val;
        for (cnt=0;cnt<10;cnt++)
        {
                val=get_angle();
                if (val>875)
                        cntd++;
        }

if(cntd>7)
        return (1);
else
        return (0);

}
```

motors.h

```
u08 motor_value[17],motor_ini_B,motor_ini_C,motor_max;

void init_motors(u08 num)
{
        u08 cnt;

        //Init timer 2
        outp((1<<OCIE2),TIMSK); // Enable interrupt of timer 2
        outp(0,TCNT2);//Initial value of timer 2
        outp(7,TCCR2);//Prescale of 1024
        outp(0x5E,OCR2);//Decimal value 94 - equal to 16.04ms - 94*1024/6M
        motor_max=num;

        for (cnt=0;cnt<=num;cnt++)
        {
                motor_value[cnt]=0;
        }
        if (num>8)
        {
                motor_ini_B=0xff;
                num=num-8;
                motor_ini_C=(1<<num)-1;
        }
        else
        {
                motor_ini_B=(1<<num)-1;
                motor_ini_C=0;
        }

        DDRB=DDRB | motor_ini_B;
        DDRC=DDRC | motor_ini_C;

        //Initialise the motors
        PORTB=PORTB & (~motor_ini_B);
        PORTC=PORTC & (~motor_ini_C);

}


SIGNAL (SIG_OUTPUT_COMPARE2)
{
        u08 cnt;
```

```
	cli();
	PORTB=PORTB | (motor_ini_B);
	PORTC=PORTC | (motor_ini_C);
	outp(0,TCNT0); //start value of timer variable
	outp(3,TCCR0); //prescale 64

	while(TCNT0<=45)//Orig was 84
	{}

	outp(0,TCNT0); //start value of timer variable

	while (TCNT0<=180) //orig 131
	{
		for (cnt=1;cnt<=motor_max;cnt++)
		{
			if (TCNT0>motor_value[cnt])
			{
				if(cnt<=8)
				{
					PORTB=PORTB & (~(1<<(cnt-1)));
				}
				else

				{
					PORTC=PORTC & (~(1<<(cnt-9)));
				}
			}
		}
	}

	PORTB=PORTB & (~motor_ini_B);
	PORTC=PORTC & (~motor_ini_C);


	outp(0,TCCR0); //stop it
//		outp(0,TCNT0);//reload initial value into timer

	outp(0,TCNT2);//Reinitializr value of timer 2
	sei();
}

void motor(u08 num, u08 angle)
{
	motor_value[num]=angle;//*180/180;
}
```

roboops.h

```
u08 legpos[7][3];

#define TURN_LEFT -1
#define TURN_RIGHT 1
#define TURN_FAST 1
#define TURN_SLOW 0
#define TURN_FORWARD 1
#define TURN_BACK -1

void stand(void)
{
	//leg1
	motor(1,Leg1U);
	motor(2,Leg1L);

	//leg 2
	motor(3,Leg2U);
	motor(4,Leg2L);

	//leg 3
	motor(5,Leg3U);
	motor(6,Leg3L);

	//leg 4
	motor(7,Leg4U);
	motor(8,Leg4L);

	//leg 5
	motor(9,Leg5U);
	motor(10,Leg5L);

	//leg 6
	motor(11,Leg6U);
	motor(12,Leg6L);

}

void walk_init(void)
{
	legpos[1][1]=Leg1U;
	legpos[1][2]=Leg1L;

	legpos[2][1]=Leg2U;
	legpos[2][2]=Leg2L;

	legpos[3][1]=Leg3U;
	legpos[3][2]=Leg3L;

	legpos[4][1]=Leg4U;
	legpos[4][2]=Leg4L;

	legpos[5][1]=Leg5U;
	legpos[5][2]=Leg5L;

	legpos[6][1]=Leg6U;
	legpos[6][2]=Leg6L;
}

void walk_six(u16 longdelay, u16 shortdelay)
{

	u08 x=6;

	motor(1,Leg1U+MOVE_FW);
	motor(3,Leg2U-MOVE_FW);
	motor(5,Leg3U+MOVE_FW);
	motor(7,Leg4U-MOVE_FW);
	motor(9,Leg5U+MOVE_FW);
	motor(11,Leg6U-MOVE_FW);
	delay(longdelay*3);


		motor(x*2,legpos[x][2]+LIFT_UP);
		delay(shortdelay);
		motor(x*2-1,legpos[x][1]+MOVE_FW);
		delay(longdelay);
		motor(x*2,legpos[x][2]);

		delay(shortdelay);
		motor(x*2-2,legpos[x-1][2]-LIFT_UP);
		delay(shortdelay);
		motor(x*2-3,legpos[x-1][1]-MOVE_FW);
		delay(longdelay);
		motor(x*2-2,legpos[x-1][2]);

	for (x=4;x>0;x=x-2)
	{
		delay(shortdelay);
		motor(x*2,legpos[x][2]-LIFT_UP);
		delay(shortdelay);
		motor(x*2-1,legpos[x][1]+MOVE_FW);
		delay(longdelay);
		motor(x*2,legpos[x][2]);

		delay(shortdelay);
		motor(x*2-2,legpos[x-1][2]+LIFT_UP);
		delay(shortdelay);
		motor(x*2-3,legpos[x-1][1]-MOVE_FW);
		delay(longdelay);
		motor(x*2-2,legpos[x-1][2]);
	}

}

void walk_six_slow(u16 longdelay, u16 shortdelay)
{

	u08 x=6;

	motor(1,Leg1U+MOVE_FW_SLOW);
	motor(3,Leg2U-MOVE_FW_SLOW);
	motor(5,Leg3U+MOVE_FW_SLOW);
	motor(7,Leg4U-MOVE_FW_SLOW);
	motor(9,Leg5U+MOVE_FW_SLOW);
	motor(11,Leg6U-MOVE_FW_SLOW);
	delay(longdelay*3);


		motor(x*2,legpos[x][2]+LIFT_UP);
		delay(shortdelay);
		motor(x*2-1,legpos[x][1]+MOVE_FW_SLOW);
		delay(longdelay);
		motor(x*2,legpos[x][2]);

		delay(shortdelay);
		motor(x*2-2,legpos[x-1][2]-LIFT_UP);
		delay(shortdelay);
		motor(x*2-3,legpos[x-1][1]-MOVE_FW_SLOW);
		delay(longdelay);
		motor(x*2-2,legpos[x-1][2]);

	for (x=4;x>0;x=x-2)
	{
		delay(shortdelay);
		motor(x*2,legpos[x][2]-LIFT_UP);
```

```
                delay(shortdelay);
                motor(x*2-1,legpos[x][1]+MOVE_FW_SLOW);
                delay(longdelay);
                motor(x*2,legpos[x][2]);

                delay(shortdelay);
                motor(x*2-2,legpos[x-1][2]+LIFT_UP);
                delay(shortdelay);
                motor(x*2-3,legpos[x-1][1]-MOVE_FW_SLOW);
                delay(longdelay);
                motor(x*2-2,legpos[x-1][2]);
        }

}


void walk_six_back(u16 longdelay, u16 shortdelay)
{

        u08 x;

        motor(1,Leg1U-MOVE_FW);
        motor(3,Leg2U+MOVE_FW);
        motor(5,Leg3U-MOVE_FW);
        motor(7,Leg4U+MOVE_FW);
        motor(9,Leg5U-MOVE_FW);
        motor(11,Leg6U+MOVE_FW);
        delay(longdelay*3);

        for (x=2;x<5;x=x+2)
        {
                delay(shortdelay);
                motor(x*2,legpos[x][2]-LIFT_UP);
                delay(shortdelay);
                motor(x*2-1,legpos[x][1]-MOVE_FW);
                delay(longdelay);
                motor(x*2,legpos[x][2]);

                delay(shortdelay);
                motor(x*2-2,legpos[x-1][2]+LIFT_UP);
                delay(shortdelay);
                motor(x*2-3,legpos[x-1][1]+MOVE_FW);
                delay(longdelay);
                motor(x*2-2,legpos[x-1][2]);
        }
```

```
	x=6;

		motor(x*2,legpos[x][2]+LIFT_UP);
		delay(shortdelay);
		motor(x*2-1,legpos[x][1]-MOVE_FW);
		delay(longdelay);
		motor(x*2,legpos[x][2]);

		delay(shortdelay);
		motor(x*2-2,legpos[x-1][2]-LIFT_UP);
		delay(shortdelay);
		motor(x*2-3,legpos[x-1][1]+MOVE_FW);
		delay(longdelay);
		motor(x*2-2,legpos[x-1][2]);



}


void turn(u16 longdelay, u16 shortdelay,u08 direction, u08 speed,u08 forward)
{
	u08 speed_left=1,speed_right=1;

	if (speed==TURN_SLOW)
	{
	if (direction==TURN_RIGHT)
	{
		speed_left=1;
		speed_right=0;
	}
	else
	{
		speed_left=0;
		speed_right=1;
	}
	}


	//leg1
	motor(1,Leg1U);
	motor(2,Leg1L);

	//leg 3
	motor(5,Leg3U);
	motor(6,Leg3L);
```

```
//leg 5
motor(9,Leg5U);
motor(10,Leg5L);


u08 x=6;

motor(1,Leg1U+MOVE_FW*speed_left*direction*forward);
motor(3,Leg2U+MOVE_FW*speed_right*direction*forward);
motor(5,Leg3U+MOVE_FW*speed_left*direction*forward);
motor(7,Leg4U+MOVE_FW*speed_right*direction*forward);
motor(9,Leg5U+MOVE_FW*speed_left*direction*forward);
motor(11,Leg6U+MOVE_FW*speed_right*direction*forward);

delay(longdelay*3);

        //delay(shortdelay);

        motor(x*2,legpos[x][2]+LIFT_UP*speed_right);
        delay(shortdelay);
        motor(x*2-1,legpos[x][1]-MOVE_FW*speed_right*direction*forward);
        delay(longdelay);
        motor(x*2,legpos[x][2]);

        delay(shortdelay);
        motor(x*2-2,legpos[x-1][2]-LIFT_UP*speed_left);
        delay(shortdelay);
        motor(x*2-3,legpos[x-1][1]-MOVE_FW*speed_left*direction*forward);
        delay(longdelay);
        motor(x*2-2,legpos[x-1][2]);

for (x=4;x>0;x=x-2)
{
        delay(shortdelay);
        motor(x*2,legpos[x][2]-LIFT_UP*speed_right);
        delay(shortdelay);
        motor(x*2-1,legpos[x][1]-MOVE_FW*speed_right*direction*forward);
        delay(longdelay);
        motor(x*2,legpos[x][2]);

        delay(shortdelay);
        motor(x*2-2,legpos[x-1][2]+LIFT_UP*speed_left);
        delay(shortdelay);
        motor(x*2-3,legpos[x-1][1]-MOVE_FW*speed_left*direction*forward);
        delay(longdelay);
```

```
                motor(x*2-2,legpos[x-1][2]);

        }




}

void turn_right_fast_fw(u16 longdelay, u16 shortdelay)
{
        turn(longdelay,shortdelay,TURN_RIGHT,TURN_FAST,TURN_FORWARD);
}



void turn_right_fw(u16 longdelay, u16 shortdelay)
{
        turn(longdelay,shortdelay,TURN_RIGHT,TURN_SLOW,TURN_FORWARD);
}


void turn_left_fast_fw(u16 longdelay, u16 shortdelay)
{
        turn(longdelay,shortdelay,TURN_LEFT,TURN_FAST,TURN_FORWARD);
}


void turn_left_fw(u16 longdelay, u16 shortdelay)
{
        turn(longdelay,shortdelay,TURN_LEFT,TURN_SLOW,TURN_FORWARD);
}

void turn_right_fast_back(u16 longdelay, u16 shortdelay)
{
        turn(longdelay,shortdelay,TURN_RIGHT,TURN_FAST,TURN_BACK);
}



void turn_right_back(u16 longdelay, u16 shortdelay)
{
        turn(longdelay,shortdelay,TURN_RIGHT,TURN_SLOW,TURN_BACK);
}
```

```
void turn_left_fast_back(u16 longdelay, u16 shortdelay)
{
	turn(longdelay,shortdelay,TURN_LEFT,TURN_FAST,TURN_BACK);
}


void turn_left_back(u16 longdelay, u16 shortdelay)
{
	turn(longdelay,shortdelay,TURN_LEFT,TURN_SLOW,TURN_BACK);
}


void stand_high(u08 hieght)
{
	//leg1
	motor(1,Leg1U);
	motor(2,Leg1L-hieght);

	//leg 2
	motor(3,Leg2U);
	motor(4,Leg2L+hieght);

	//leg 3
	motor(5,Leg3U);
	motor(6,Leg3L-hieght);

	//leg 4
	motor(7,Leg4U);
	motor(8,Leg4L+hieght);

	//leg 5
	motor(9,Leg5U);
	motor(10,Leg5L+hieght+10);

	//leg 6
	motor(11,Leg6U);
	motor(12,Leg6L-hieght-10);

}

u08 find_hieght(void)
{
		stand();
		delay(0xFFF);

	u08 Top_read_ini,Top_read;
```

```
    Top_read_ini=Top_IR();

        stand_high(90);
        delay(0x1FFF);
        Top_read=Top_IR();
        if (Top_read<Top_read_ini-10)
            retur n(60);
        else
            return(0);

    return (0);

}
```

roboops2.h

```
void align(void)
{
        u08 Left_read,Right_read,Top_read,correction,tolerance;
        tolerance=5;
        correction=5;

                        stand();
                        delay(0x1FF);

                        Top_read=Top_IR();

                        while (Top_read>70)
                        {
                                walk_six_back(0x01FF,0x00FF);
                                delay(0x3FF);
                                Top_read=Top_IR();
                        }

                        stand();
                        delay(0x1FF);
                        Left_read=Left_IR();
                        Right_read=Right_IR();

                        while (Left_read+correction-tolerance>Right_read)
                        {
                                turn_left_fast_fw(0x01FF,0x00FF);
                                Left_read=Left_IR();
                                Right_read=Right_IR();
                        }

                        stand();
                        delay(0x1FF);
                        Top_read=Top_IR();

                        while (Top_read>90)
                        {
                                walk_six_back(0x01FF,0x00FF);
                                delay(0x3FF);
                                Top_read=Top_IR();
                        }

                        stand();
```

```
delay(0x1FF);

Left_read=Left_IR();
Right_read=Right_IR();

while (Left_read+correction+tolerance<Right_read)
{
        turn_right_fast_fw(0x01FF,0x00FF);
        delay(0x1FF);
        Left_read=Left_IR();
        Right_read=Right_IR();

}

stand();
delay(0x1FF);

Top_read=Top_IR();

while (Top_read>70)
{
        walk_six_back(0x01FF,0x00FF);
        delay(0x3FF);
        Top_read=Top_IR();
}

stand();
delay(0x1FF);

Left_read=Left_IR();
Right_read=Right_IR();

while (Left_read+correction-tolerance>Right_read)
{
        turn_left_fw(0x01FF,0x00FF);
        delay(0x1FF);
        Left_read=Left_IR();
        Right_read=Right_IR();
}

stand();
delay(0x1FF);

Top_read=Top_IR();

while (Top_read>90)
```

```
                        {
                                walk_six_back(0x01FF,0x00FF);
                                delay(0x3FF);
                                Top_read=Top_IR();
                        }

                        stand();
                        delay(0x1FF);

                        Left_read=Left_IR();
                        Right_read=Right_IR();

                        while (Left_read+correction+tolerance<Right_read)
                        {
                                turn_right_fw(0x01FF,0x00FF);
                                delay(0x1FF);
                                Left_read=Left_IR();
                                Right_read=Right_IR();

                        }

                        stand();
                        delay(0x1FF);

                        Top_read=Top_IR();

                        while (Top_read>90)
                        {
                                walk_six_back(0x01FF,0x00FF);
                                delay(0x3FF);
                                Top_read=Top_IR();
                        }

}



u08 obstacle(void)
{

        u08 Left_read,Right_read,hieght;

        Left_read=Left_IR();
        Right_read=Right_IR();

        //Top_read=Top_IR();
        if (Left_read>50 || Right_read>50)
        {
                hieght=find_hieght();

                if (hieght==0)
                {
                        stand();
                        if (Left_read>40)
                        {

                                turn_left_fast_back(0x01FF,0x00FF);
                                delay(0x01FF);
                                turn_left_fast_back(0x01FF,0x00FF);
                                delay(0x01FF);
                                turn_left_fast_back(0x01FF,0x00FF);
                                delay(0x01FF);
                                turn_left_fast_back(0x01FF,0x00FF);
                                delay(0x01FF);
                                turn_left_fast_back(0x01FF,0x00FF);
                                delay(0x01FF);

                        }
                        else //if (Right_read>40)
                        {
                                turn_right_fast_back(0x01FF,0x00FF);
                                delay(0x01FF);
                                turn_right_fast_back(0x01FF,0x00FF);
                                delay(0x01FF);
                                turn_right_fast_back(0x01FF,0x00FF);
                                delay(0x01FF);
                                turn_right_fast_back(0x01FF,0x00FF);
                                delay(0x01FF);
                                turn_right_fast_back(0x01FF,0x00FF);
                                delay(0x01FF);
                        }
                        return(0);
                }
                else
                        return(hieght);

        }

        return(0);
}
```

```
void up_step(u08 hieght,u08 legs)
{
        u16 shortdelay=0x03FF,longdelay=0x06FF;
        u08 legs34=1,legs56=1;
        u08 x=6;

        motor(1,Leg1U+MOVE_FW);
        motor(3,Leg2U-MOVE_FW);
        motor(5,Leg3U+MOVE_FW);
        motor(7,Leg4U-MOVE_FW);
        motor(9,Leg5U+MOVE_FW);
        motor(11,Leg6U-MOVE_FW);

        if (legs==34)
        {
                legs34=0;
                legs56=1;
        }

        if (legs==56)
        {
                legs34=0;
                legs56=0;
        }

        //Leg 2 & 1
        x=2;
                delay(shortdelay);
                motor(x*2,legpos[x][2]-LIFT_UP_CLIMB);
                delay(shortdelay);
                motor(x*2-1,legpos[x][1]+MOVE_FW);
                delay(longdelay);
                motor(x*2,legpos[x][2]-40);//+LIFT_UP_CLIMB);
                delay(longdelay);

                delay(shortdelay);
                motor(x*2-2,legpos[x-1][2]+LIFT_UP_CLIMB);
                delay(shortdelay);
                motor(x*2-3,legpos[x-1][1]-MOVE_FW);
                delay(longdelay);
                motor(x*2-2,legpos[x-1][2]+40);//-LIFT_UP_CLIMB);
                delay(longdelay);

        //Leg 4 & 3
        x=4;
```

```
                delay(shortdelay);
                motor(x*2,legpos[x][2]-LIFT_UP_CLIMB);
                delay(shortdelay);
                motor(x*2-1,legpos[x][1]+MOVE_FW);
                delay(longdelay);
                motor(x*2,legpos[x][2]+hieght*legs34-20);
                delay(longdelay);

                delay(shortdelay);
                motor(x*2-2,legpos[x-1][2]+LIFT_UP_CLIMB);
                delay(shortdelay);
                motor(x*2-3,legpos[x-1][1]-MOVE_FW);
                delay(longdelay);
                motor(x*2-2,legpos[x-1][2]-hieght*legs34+20);
                delay(longdelay);

        //Leg 6 & 5
        x=6;

                motor(x*2,legpos[x][2]+LIFT_UP_CLIMB);
                delay(shortdelay);
                motor(x*2-1,legpos[x][1]+MOVE_FW);
                delay(longdelay);
                motor(x*2,legpos[x][2]-hieght*legs56+20);
                delay(longdelay);

                delay(shortdelay);
                motor(x*2-2,legpos[x-1][2]-LIFT_UP_CLIMB);
                delay(shortdelay);
                motor(x*2-3,legpos[x-1][1]-MOVE_FW);
                delay(longdelay);
                motor(x*2-2,legpos[x-1][2]+hieght*legs56-20);
                delay(longdelay);



        delay(longdelay*3);

}


void up_step12(u08 hieght,u08 legs)
{
        u16 shortdelay=0x03FF,longdelay=0x06FF;
        u08 legs34=1,legs56=1;
        u08 x=6;
```

```
motor(1,Leg1U+MOVE_FW);
motor(3,Leg2U-MOVE_FW);
motor(5,Leg3U+MOVE_FW);
motor(7,Leg4U-MOVE_FW);
motor(9,Leg5U+MOVE_FW);
motor(11,Leg6U-MOVE_FW);

delay(longdelay*3);

if (legs==34)
{
        legs34=0;
        legs56=1;
}

if (legs==56)
{
        legs34=0;
        legs56=0;
}

//Leg 4 & 3
x=4;
        delay(shortdelay);
        motor(x*2,legpos[x][2]-LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-1,legpos[x][1]);//+MOVE_FW_CLIMB);
        delay(longdelay);
        motor(x*2,legpos[x][2]+hieght*legs34);
        delay(longdelay);

        delay(shortdelay);
        motor(x*2-2,legpos[x-1][2]+LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-3,legpos[x-1][1]);//-MOVE_FW_CLIMB);
        delay(longdelay);
        motor(x*2-2,legpos[x-1][2]- hieght*legs34);

        delay(longdelay*4);

//Lean back
x=6;
        motor(x*2,legpos[x][2]-hieght*legs56+50);
        motor(x*2-2,legpos[x-1][2]+hieght*legs56-50);
        delay(longdelay);
```

```
//Leg 2 & 1
x=2;
        delay(shortdelay);
        motor(x*2,legpos[x][2]-LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-1,legpos[x][1]+MOVE_FW);
        delay(longdelay);
        motor(x*2,legpos[x][2]-20);//+LIFT_UP_CLIMB);
        delay(longdelay);

        delay(shortdelay);
        motor(x*2-2,legpos[x-1][2]+LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-3,legpos[x-1][1]-MOVE_FW);
        delay(longdelay);
        motor(x*2-2,legpos[x-1][2]+20);//-LIFT_UP_CLIMB);
        delay(longdelay);

//Lean cancel
x=6;
        motor(x*2,legpos[x][2]-hieght*legs56);
        motor(x*2-2,legpos[x-1][2]+hieght*legs56);
        delay(longdelay);


//Leg 4 & 3
x=4;
        delay(shortdelay);
        motor(x*2,legpos[x][2]-LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-1,legpos[x][1]-MOVE_FW);
        delay(longdelay);
        motor(x*2,legpos[x][2]+hieght*legs34);
        delay(longdelay);

        delay(shortdelay);
        motor(x*2-2,legpos[x-1][2]+LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-3,legpos[x-1][1]+MOVE_FW);
        delay(longdelay);
        motor(x*2-2,legpos[x-1][2]- hieght*legs34);


//Leg 6 & 5
x=6;
```

```
		motor(x*2,legpos[x][2]+LIFT_UP_CLIMB);
		delay(shortdelay);
		motor(x*2-1,legpos[x][1]+MOVE_FW_CLIMB);
		delay(longdelay);
		motor(x*2,legpos[x][2]-hieght*legs56+20);
		delay(longdelay);

		delay(shortdelay);
		motor(x*2-2,legpos[x-1][2]-LIFT_UP_CLIMB);
		delay(shortdelay);
		motor(x*2-3,legpos[x-1][1]-MOVE_FW_CLIMB);
		delay(longdelay);
		motor(x*2-2,legpos[x-1][2]+hieght*legs56-20);
		delay(longdelay);

		//Leg 4 & 3
	x=4;
		delay(shortdelay);
		motor(x*2,legpos[x][2]-LIFT_UP_CLIMB);
		delay(shortdelay);
		motor(x*2-1,legpos[x][1]+MOVE_FW);
		delay(longdelay);
		motor(x*2,legpos[x][2]+hieght*legs34);
		delay(longdelay);

		delay(shortdelay);
		motor(x*2-2,legpos[x-1][2]+LIFT_UP_CLIMB);
		delay(shortdelay);
		motor(x*2-3,legpos[x-1][1]-MOVE_FW);
		delay(longdelay);
		motor(x*2-2,legpos[x-1][2]- hieght*legs34);

	delay(longdelay*4);

	motor(1,Leg1U+MOVE_FW);
	motor(3,Leg2U-MOVE_FW);
	motor(5,Leg3U+MOVE_FW);
	motor(7,Leg4U-MOVE_FW);
	motor(9,Leg5U+MOVE_FW);
	motor(11,Leg6U-MOVE_FW);

	delay(longdelay*3);

}
```

```
void up_step34(u08 hieght,u08 legs)
{
        u16 shortdelay=0x03FF,longdelay=0x06FF;
        u08 legs34=1,legs56=1;
        u08 x=6;

        motor(1,Leg1U+MOVE_FW);
        motor(3,Leg2U-MOVE_FW);
        motor(5,Leg3U+MOVE_FW);
        motor(7,Leg4U-MOVE_FW);
        motor(9,Leg5U+MOVE_FW);
        motor(11,Leg6U-MOVE_FW);

        delay(longdelay*3);

        if (legs==34)
        {
                legs34=0;
                legs56=1;
        }

        if (legs==56)
        {
                legs34=0;
                legs56=0;
        }

        //Leg 6 & 5
        x=6;

                motor(x*2,legpos[x][2]+LIFT_UP_CLIMB);
                delay(shortdelay);
                motor(x*2-1,legpos[x][1]+MOVE_FW);
                delay(longdelay);
                motor(x*2,legpos[x][2]-hieght*legs56+20);
                delay(longdelay);

                delay(shortdelay);
                motor(x*2-2,legpos[x-1][2]-LIFT_UP_CLIMB);
                delay(shortdelay);
                motor(x*2-3,legpos[x-1][1]-MOVE_FW);
                delay(longdelay);
                motor(x*2-2,legpos[x-1][2]+hieght*legs56-20);
                delay(longdelay);
//Leg 4 & 3
x=4;
        delay(shortdelay);
        motor(x*2,legpos[x][2]-LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-1,legpos[x][1]+MOVE_FW);
        delay(longdelay);
        motor(x*2,legpos[x][2]-20);
        delay(longdelay);

        delay(shortdelay);
        motor(x*2-2,legpos[x-1][2]+LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-3,legpos[x-1][1]-MOVE_FW);
        delay(longdelay);
        motor(x*2-2,legpos[x-1][2]+20);

//Lean back
x=6;
        motor(x*2,legpos[x][2]-hieght*legs56+50);
        motor(x*2-2,legpos[x-1][2]+hieght*legs56-50);
x=4;
        motor(x*2,legpos[x][2]);
        motor(x*2-2,legpos[x-1][2]);
        delay(longdelay);



//Leg 2 & 1
x=2;
        delay(shortdelay);
        motor(x*2,legpos[x][2]-LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-1,legpos[x][1]+MOVE_FW);
        delay(longdelay);
        motor(x*2,legpos[x][2]);//+LIFT_UP_CLIMB);
        delay(longdelay);

        delay(shortdelay);
        motor(x*2-2,legpos[x-1][2]+LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-3,legpos[x-1][1]-MOVE_FW);
        delay(longdelay);
        motor(x*2-2,legpos[x-1][2]);//-LIFT_UP_CLIMB);
        delay(longdelay*4);
        //Lean cancel
        x=6;
                motor(x*2,legpos[x][2]-hieght);
                motor(x*2-2,legpos[x-1][2]+hieght);
        x=4;
                motor(x*2,legpos[x][2]-20);
                motor(x*2-2,legpos[x-1][2]+20);
                delay(longdelay);

        motor(1,Leg1U+MOVE_FW);
        motor(3,Leg2U-MOVE_FW);
        motor(5,Leg3U+MOVE_FW);
        motor(7,Leg4U-MOVE_FW);
        motor(9,Leg5U+MOVE_FW);
        motor(11,Leg6U-MOVE_FW);

        delay(longdelay*3);

}


void up_step56(u08 hieght,u08 legs)
{
        u16 shortdelay=0x03FF,longdelay=0x06FF;
        u08 legs34=1,legs56=1;
        u08 x=6;

        motor(1,Leg1U+MOVE_FW);
        motor(3,Leg2U-MOVE_FW);
        motor(5,Leg3U+MOVE_FW);
        motor(7,Leg4U-MOVE_FW);
        motor(9,Leg5U+MOVE_FW);
        motor(11,Leg6U-MOVE_FW);

        delay(longdelay*3);

        if (legs==34)
        {
                legs34=0;
                legs56=1;
        }

        if (legs==56)
        {
                legs34=0;
        legs56=0;
}

//Lean back
x=6;
        motor(x*2,legpos[x][2]-hieght+50);
        motor(x*2-2,legpos[x-1][2]+hieght-50);
x=4;
        motor(x*2,legpos[x][2]);
        motor(x*2-2,legpos[x-1][2]);
        delay(longdelay);

x=4;
        delay(shortdelay);
        motor(x*2,legpos[x][2]-LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-1,legpos[x][1]+MOVE_FW+80);
        delay(longdelay);
        motor(x*2,legpos[x][2]-20);
        delay(longdelay);

        delay(shortdelay);
        motor(x*2-2,legpos[x-1][2]+LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-3,legpos[x-1][1]-MOVE_FW-80);
        delay(longdelay);
        motor(x*2-2,legpos[x-1][2]+20);


//Leg 2 & 1
x=2;
        delay(shortdelay);
        motor(x*2,legpos[x][2]-LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-1,legpos[x][1]+MOVE_FW);
        delay(longdelay);
        motor(x*2,legpos[x][2]-20);//+LIFT_UP_CLIMB);
        delay(longdelay);

        delay(shortdelay);
        motor(x*2-2,legpos[x-1][2]+LIFT_UP_CLIMB);
        delay(shortdelay);
        motor(x*2-3,legpos[x-1][1]-MOVE_FW);
        delay(longdelay);
        motor(x*2-2,legpos[x-1][2]+20);//-LIFT_UP_CLIMB);
        delay(longdelay*2);
```

```
//Lean cancel
x=6;
        motor(x*2,legpos[x][2]-hieght-20);
        motor(x*2-2,legpos[x-1][2]+hieght+20);
x=4;
        motor(x*2,legpos[x][2]+40);
        motor(x*2-2,legpos[x-1][2]-40);
x=2;
        motor(x*2,legpos[x][2]+40);//+LIFT_UP_CLIMB);
        motor(x*2-2,legpos[x-1][2]-40);//-LIFT_UP_CLIMB);
        delay(longdelay*3);

x=2;
        motor(x*2,legpos[x][2]+10);//+LIFT_UP_CLIMB);
        motor(x*2-2,legpos[x-1][2]-10);//-LIFT_UP_CLIMB);
x=4;
        motor(x*2,legpos[x][2]+50);
        motor(x*2-2,legpos[x-1][2]-50);
x=6;
        motor(x*2,legpos[x][2]-hieght);
        motor(x*2-2,legpos[x-1][2]+hieght+30);

        delay(longdelay*3);

//Leg 6 & 5
x=6;

        motor(x*2,legpos[x][2]+LIFT_UP_CLIMB+20);
        delay(longdelay);
        motor(x*2-1,legpos[x][1]+MOVE_FW_CLIMB-40);
        delay(longdelay*2);
        motor(x*2,legpos[x][2]-80);
        delay(longdelay*2);

        delay(shortdelay);
        motor(x*2-2,legpos[x-1][2]-LIFT_UP_CLIMB);
        delay(longdelay);
        motor(x*2-3,legpos[x-1][1]-MOVE_FW_CLIMB+40);
        delay(longdelay*2);
        motor(x*2-2,legpos[x-1][2]+80);
        delay(longdelay);


x=2;
```

```
			motor(x*2,legpos[x][2]+30);//+LIFT_UP_CLIMB);
			motor(x*2-2,legpos[x-1][2]-30);//-LIFT_UP_CLIMB);
		x=4;
			motor(x*2,legpos[x][2]-20);
			motor(x*2-2,legpos[x-1][2]+20);
			delay(longdelay);

		delay(longdelay);

		motor(1,Leg1U+MOVE_FW);
		motor(3,Leg2U-MOVE_FW);
		motor(5,Leg3U+MOVE_FW);
		motor(7,Leg4U-MOVE_FW);
		delay(longdelay*2);

		motor(9,Leg5U);
		motor(11,Leg6U);

		delay(longdelay*2);

	}

	void dance(void)
	{
		u08 x=0;

		delay(0x1fff);
		stand();
		delay(0xfff);
		stand_high(90);
		delay(0xfff);
		x=6;
		motor(x*2,legpos[x][2]+20);
		motor(x*2-2,legpos[x-1][2]-20);
		delay(0x4ff);
		stand();
		delay(0x4ff);
		stand_high(90);
		delay(0x4ff);
		turn_left_fast_fw(0x0FF,0x008F);
		delay(0x4ff);
		stand_high(90);
		delay(0x4ff);
		turn_right_fast_fw(0x0FF,0x008F);
		delay(0x4ff);
		x=2;
	motor(x*2,legpos[x][2]+20);
	motor(x*2-2,legpos[x-1][2]-20);
	delay(0x4ff);
	stand();
	delay(0x4ff);
	stand_high(90);
	delay(0x1fff);

}

void climb_step(u08 hieght)
{
	u08 left_read,right_read,x;

	left_read=Left_IR();
	right_read=Right_IR();

	//Recheck for alignment!!!
	if(left_read>right_read+10 || left_read<right_read-10)
		align();

	left_read=Left_IR();
	right_read=Right_IR();
/*
	while ((left_read+right_read)<160)
	{
		walk_six_slow(0x01FF,0x00FF);
	}*/

	walk_six(0x01FF,0x00FF);
	walk_six(0x01FF,0x00FF);
	walk_six_slow(0x01FF,0x00FF);
	walk_six_slow(0x01FF,0x00FF);
	walk_six(0x01FF,0x00FF);


	stand();
	motor(1,Leg1U+MOVE_FW_CLIMB);
	motor(3,Leg2U-MOVE_FW_CLIMB);
	motor(5,Leg3U+MOVE_FW_CLIMB);
	motor(7,Leg4U-MOVE_FW_CLIMB);
	motor(9,Leg5U+MOVE_FW_CLIMB);
	motor(11,Leg6U-MOVE_FW_CLIMB);

	delay(0x3ff);
	stand_high(90);
	motor(1,Leg1U+MOVE_FW_CLIMB);
	motor(3,Leg2U-MOVE_FW_CLIMB);
	motor(5,Leg3U+MOVE_FW_CLIMB);
	motor(7,Leg4U-MOVE_FW_CLIMB);
	motor(9,Leg5U+MOVE_FW_CLIMB);
	motor(11,Leg6U-MOVE_FW_CLIMB);
	delay(0x1fff);

		stand();
	motor(1,Leg1U+MOVE_FW_CLIMB);
	motor(3,Leg2U-MOVE_FW_CLIMB);
	motor(5,Leg3U+MOVE_FW_CLIMB);
	motor(7,Leg4U-MOVE_FW_CLIMB);
	motor(9,Leg5U+MOVE_FW_CLIMB);
	motor(11,Leg6U-MOVE_FW_CLIMB);

	delay(0x3ff);
	stand_high(90);
	motor(1,Leg1U+MOVE_FW_CLIMB);
	motor(3,Leg2U-MOVE_FW_CLIMB);
	motor(5,Leg3U+MOVE_FW_CLIMB);
	motor(7,Leg4U-MOVE_FW_CLIMB);
	motor(9,Leg5U+MOVE_FW_CLIMB);
	motor(11,Leg6U-MOVE_FW_CLIMB);
	delay(0x1fff);

//Actual climbing

	up_step12(80,12);
	delay(0xfff);
	if (tilt_forward())
	{
		climb_trials++;
		if (climb_trials<3)//Three trials
		{
			walk_six_back(0x01FF,0x00FF);
			delay(0x3FF);
			walk_six_back(0x01FF,0x00FF);
			delay(0x3FF);
			walk_six_back(0x01FF,0x00FF);
			delay(0x3FF);
			stand();
			delay(0x1ff);
			align();
			climb_step(hieght);
		}
		else
		{
		//turn right
			climb_trials=0;
			walk_six_back(0x01FF,0x00FF);
			delay(0x3FF);
			walk_six_back(0x01FF,0x00FF);
			delay(0x3FF);

			turn_right_fast_back(0x01FF,0x00FF);
			delay(0x01FF);
			turn_right_fast_back(0x01FF,0x00FF);
			delay(0x01FF);
			turn_right_fast_back(0x01FF,0x00FF);
			delay(0x01FF);
			turn_right_fast_back(0x01FF,0x00FF);
			delay(0x01FF);
			turn_right_fast_back(0x01FF,0x00FF);
			delay(0x01FF);
		}

		return;
	}

	up_step12(80,12);
	delay(0xfff);
	up_step34(80,12);
	up_step34(80,12);
	up_step34(80,12);
	delay(0xfff);
	up_step56(80,12);
	up_step56(80,12);
	walk_six(0x01FF,0x00FF);
	walk_six(0x01FF,0x00FF);
	walk_six(0x01FF,0x00FF);
	dance();

}
```

Appendix B

Code to check if IR was working and to get reading for different distances

IR code

```
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>

typedef unsigned char  uint8_t;
#define F_CPU           6000000      /* 6Mhz */
#define UART_BAUD_RATE     19200      /* 19200 baud */
#define UART_BAUD_SELECT (F_CPU/(UART_BAUD_RATE*16l)-1)

typedef unsigned char  u08;
typedef          char  s08;
typedef unsigned short u16;
typedef          short s16;

uint8_t lo_val, hi_val,flag;
uint8_t value1;

SIGNAL (SIG_ADC)
{
        char buffer [10];
        lo_val = inp(ADCL) / 4; /* read low byte first */
        hi_val = inp(ADCH) * 64;

        value1=lo_val + hi_val;

        outp(~value1, PORTC);
        outp(0, ADCSR);
        itoa(value1,buffer,10);
      uart_send(buffer, 5);
      delay(0x0FFF);
}


/* uart globals */
volatile char* uart_data_ptr;
static volatile u08 uart_counter;


SIGNAL(SIG_UART_DATA)
/* signal handler for uart txd ready interrupt */
```

```
{
   uart_data_ptr++;

   if (--uart_counter)
      UDR = *uart_data_ptr; //outb(UDR, *uart_data_ptr);      /* write byte to data buffer
*/
   else
      cbi(UCSRB,UDRIE);              //Have to disable IRQ, or it will repeat
}


SIGNAL(SIG_UART_RECV)
/* signal handler for receive complete interrupt */
{
   register char led;

   led = UDR;       /* read byte for UART data buffer */
   PORTC = ~led;    /* output received byte to PortB (LEDs) */
   if(bit_is_set(UCSRA,UDRE)) {
     UDR = led;
   }
}


void uart_send(u08 *buf, u08 size)
/* send buffer <buf> to uart */
{
   if (!uart_counter) {
      /* write first byte to data buffer */
      uart_data_ptr  = buf;
      uart_counter   = size;
      UDR = *buf;
      sbi(UCSRB,UDRIE);      //Enable the send IRQ
   }
}


void uart_init(void)
/* initialize uart */
{
   UBRRH = 0x00;
   UBRRL = UART_BAUD_SELECT;
   UCSRB = _BV(RXCIE)|_BV(RXEN)|_BV(TXEN);
   UCSRC = _BV(URSEL)|_BV(UCSZ1)|_BV(UCSZ0);
   UCSRA = 0x00;
}
```

```
void delay(u16 delay_time) {
   do {
     u08 i=0;
     do {
     asm volatile("nop\n\t"
        "nop\n\t"
        "nop\n\t"
        "nop\n\t"
        ::);
     } while(--i);
   } while(--delay_time);
}


int main(void)
{
   DDRC  = 0xff;      /* PortC output */
   PORTC = 0x00;      /* switch LEDs on */
   DDRD  = 0x00;      // Port B output for D9 LED
   //PORTB = 0x00;      // Turn on D9

   delay(0x01FF);

   //PORTB = 0xFF;     // Turn off D9


/* Enables ADC and start conversion in free running interrupt mode */
outp(0xFF, DDRC); /* define PORTB as Output (Leds) */

outp(0, ADMUX); /* Select Analog input 0*/


/* Enables ADC and start conversion in free running interrupt mode */
   outp((1<<ADEN)|(1<<ADSC)|(1<<ADFR)|(1<<ADIE), ADCSR);
            /* enable interrupts */


flag=0;

char buffer [10];

   uart_init();
   sei();              /* enable interrupts */
```

```
    for (;;) {              /* loop forever */
            itoa(value1,buffer,10);
        uart_send(buffer, 5);
        delay(0x02FF);
        uart_send("        ", 8);
        delay(0x05FF);
        outp((1<<ADEN)|(1<<ADSC)|(1<<ADFR)|(1<<ADIE), ADCSR);

    }
}
```

Accelerometer

```
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>

typedef unsigned char  uint8_t;
#define F_CPU           6000000      /* 6Mhz */
#define UART_BAUD_RATE     19200      /* 19200 baud */


#define UART_BAUD_SELECT (F_CPU/(UART_BAUD_RATE*16l)-1)


typedef unsigned char  u08;
typedef          char  s08;
typedef unsigned short u16;
typedef          short s16;

uint8_t lo_val, hi_val,flag;
u16 value1,value,temp0,temp1;


SIGNAL (SIG_INPUT_CAPTURE1)
{

if (flag==1)
{
        lo_val = inp(ICR1L); /* read low byte first */
        hi_val = inp(ICR1H);
        value1=lo_val+hi_val*256;
        outp(0,TCCR1A); //No compare/capture/PWM
        outp(1,TCCR1B); //prescale
        PORTC=0x00;
        flag=0;
}
else
{
        lo_val = inp(ICR1L); /* read low byte first */
        hi_val = inp(ICR1H);
        value=lo_val+hi_val*256-value1;
        outp(0,TCCR1A); //No compare/capture/PWM
        outp(65,TCCR1B); //prescale
        outp(0x00,TCNT1H); //start value of timer variable
```

```
            outp(0,TCNT1L);
            PORTC=0xff;
            flag=1;
}

}

/* uart globals */
volatile char* uart_data_ptr;
static volatile u08 uart_counter;


SIGNAL(SIG_UART_DATA)
/* signal handler for uart txd ready interrupt */
{
   uart_data_ptr++;

   if (--uart_counter)
      UDR = *uart_data_ptr; //outb(UDR, *uart_data_ptr);      /* write byte to data buffer
*/
   else
      cbi(UCSRB,UDRIE);              //Have to disable IRQ, or it will repeat
}


SIGNAL(SIG_UART_RECV)
/* signal handler for receive complete interrupt */
{
   register char led;

   led = UDR;       /* read byte for UART data buffer */
   PORTC = ~led;    /* output received byte to PortB (LEDs) */
   if(bit_is_set(UCSRA,UDRE)) {
     UDR = led;
   }
}


void uart_send(u08 *buf, u08 size)
/* send buffer <buf> to uart */
{
   if (!uart_counter) {
      /* write first byte to data buffer */
      uart_data_ptr  = buf;
      uart_counter   = size;
      UDR = *buf;
```

```
        sbi(UCSRB,UDRIE);       //Enable the send IRQ
    }
}


void uart_init(void)
/* initialize uart */
{
    UBRRH = 0x00;
    UBRRL = UART_BAUD_SELECT;
    UCSRB = _BV(RXCIE)|_BV(RXEN)|_BV(TXEN);
    UCSRC = _BV(URSEL)|_BV(UCSZ1)|_BV(UCSZ0);
    UCSRA = 0x00;
}


void delay(u16 delay_time) {
    do {
      u08 i=0;
      do {
      asm volatile("nop\n\t"
          "nop\n\t"
          "nop\n\t"
          "nop\n\t"
          ::);
      } while(--i);
    } while(--delay_time);
}


int main(void)
{
    DDRC  = 0xff;      /* PortC output */
    PORTC = 0x00;      /* switch LEDs on */
    DDRD  = 0x00;      // Port B output for D9 LED

    delay(0x01FF);


/* Enables ADC and start conversion in free running interrupt mode */

flag=0;

outp(32,TIMSK); //Enables the T1C0 overflow interrupt
outp(0x00,TCNT1H); //start value of timer variable
outp(0,TCNT1L);

outp(0,TCCR1A); //No compare/capture/PWM
outp(1,TCCR1B); //prescale
value=0;
char buffer [10];

   uart_init();
   sei();              /* enable interrupts */

   for (;;) {          /* loop forever */
         itoa(value,buffer,10);
      uart_send(buffer, 5);
      delay(0x02FF);
      uart_send("      ", 8);
      delay(0x05FF);
   }
}
```

Input Capture

```
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>

typedef unsigned char  uint8_t;
#define F_CPU            6000000      /* 6Mhz */
#define UART_BAUD_RATE     19200      /* 19200 baud */


#define UART_BAUD_SELECT (F_CPU/(UART_BAUD_RATE*16l)-1)


typedef unsigned char  u08;
typedef          char  s08;
typedef unsigned short u16;
typedef          short s16;

uint8_t lo_val, hi_val,flag;
u16 value1,value,temp0,temp1;


SIGNAL (SIG_INPUT_CAPTURE1)
{

if (flag==1)
{
        lo_val = inp(ICR1L); /* read low byte first */
        hi_val = inp(ICR1H);
        value1=lo_val+hi_val*256;
        outp(0,TCCR1A); //No compare/capture/PWM
        outp(1,TCCR1B); //prescale
        PORTC=0x00;
        flag=0;
}
else
{
        lo_val = inp(ICR1L); /* read low byte first */
        hi_val = inp(ICR1H);
        value=lo_val+hi_val*256-value1;
        outp(0,TCCR1A); //No compare/capture/PWM
        outp(65,TCCR1B); //prescale
        outp(0x00,TCNT1H); //start value of timer variable
        outp(0,TCNT1L);
```

```
            PORTC=0xff;
            flag=1;
}

}

/* uart globals */
volatile char* uart_data_ptr;
static volatile u08 uart_counter;


SIGNAL(SIG_UART_DATA)
/* signal handler for uart txd ready interrupt */
{
   uart_data_ptr++;

   if (--uart_counter)
      UDR = *uart_data_ptr; //outb(UDR, *uart_data_ptr);      /* write byte to data buffer
*/
   else
      cbi(UCSRB,UDRIE);              //Have to disable IRQ, or it will repeat
}


SIGNAL(SIG_UART_RECV)
/* signal handler for receive complete interrupt */
{
   register char led;

   led = UDR;       /* read byte for UART data buffer */
   PORTC = ~led;    /* output received byte to PortB (LEDs) */
   if(bit_is_set(UCSRA,UDRE)) {
     UDR = led;
   }
}


void uart_send(u08 *buf, u08 size)
/* send buffer <buf> to uart */
{
   if (!uart_counter) {
      /* write first byte to data buffer */
      uart_data_ptr  = buf;
      uart_counter   = size;
      UDR = *buf;
      sbi(UCSRB,UDRIE);      //Enable the send IRQ
   }
}


void uart_init(void)
/* initialize uart */
{
   UBRRH = 0x00;
   UBRRL = UART_BAUD_SELECT;
   UCSRB = _BV(RXCIE)|_BV(RXEN)|_BV(TXEN);
   UCSRC = _BV(URSEL)|_BV(UCSZ1)|_BV(UCSZ0);
   UCSRA = 0x00;
}


void delay(u16 delay_time) {
   do {
     u08 i=0;
     do {
     asm volatile("nop\n\t"
         "nop\n\t"
         "nop\n\t"
         "nop\n\t"
         ::);
     } while(--i);
   } while(--delay_time);
}


int main(void)
{
   DDRC  = 0xff;      /* PortC output */
   PORTC = 0x00;      /* switch LEDs on */
   DDRD  = 0x00;      // Port B output for D9 LED

   delay(0x01FF);


/* Enables ADC and start conversion in free running interrupt mode */

flag=0;

outp(32,TIMSK); //Enables the T1C0 overflow interrupt


outp(0x00,TCNT1H); //start value of timer variable
outp(0,TCNT1L);

outp(0,TCCR1A); //No compare/capture/PWM
outp(1,TCCR1B); //prescale
value=0;
char buffer [10];

   uart_init();
   sei();              /* enable interrupts */

   for (;;) {          /* loop forever */
         itoa(value,buffer,10);
      uart_send(buffer, 5);
      delay(0x02FF);
      uart_send("        ", 8);
      delay(0x05FF);
   }
}
```