COOPERATIVE ROBOTICS: IMDL PROJECT

ASHISH JAIN

September 2002

Abstract

This project is derived from the idea of RoboSoccer Competition. The main idea behind the project is to involve a team of robots to do a task like playing soccer. The idea uses the concept of a single camera, computer vision algorithms, a centralised computer, a communication link for communicating between the robots, each having its own capabilities of local object detection, to perform the task. Since main criteria is cooperative robotics there is a not very strong emphasis on computer vision. It is used mainly to provide location and orientation information of robots and the object(e.g. ball). There are lot of factors which comeinto this task, one is the centralised approach, second is the master slave criteria and the third is the communication time from the central computer to each of the robots.

Contents

List of Figures

Chapter 1

Introduction

The idea of cooperative robotics has been old but there is never been a platform for testing ideas. RoboSoccer was one such event where they actually got people involved in performing a task done through teams. The task of playing soccer between robots involves lot of issues that needs to be addressed, the biggest being the team work.

The problem with cooperative robotics is to define task for each individual robots and to give it its space and time. A problem like RoboSoccer addresses the issue of who is going to be the defender, who is going to be in the midfield, who is going to be the forward(job specification). There are issues like who is going to pass the ball to whom, will it be defender to forward or defender to midfield who in turn will pass it to forward(fulfilling the team effort).

Then there are other issues like the position of the ball, where the ball is heading for, how to intercept it and how to pass the ball to your player. These problems fall into the domain of cooperative behaviour.

1.1 Problem Statement

The team consist of atleast three robots and a goalkeeper,(but I am planning to have no goalkeepers) because then I need to have eight robots making two teams. The task that I have in mind is to usethree robots to seek a ball pass to each other and head for the goal.

The robots will be moving in a region which will be bounded by a wall to give them the idea of the borderline. There will be a goal created for them so that every time they are told to start, they will go seek the ball and finish

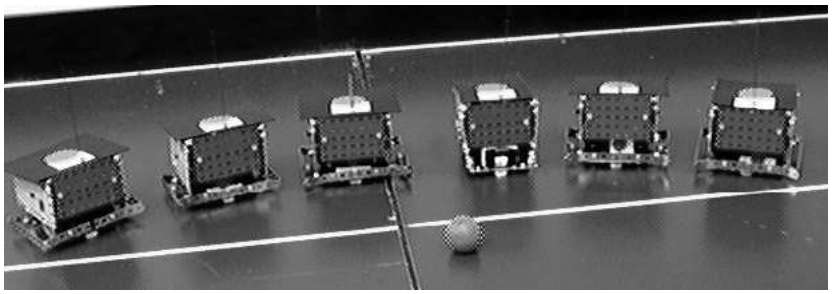


Figure 1.1: Team of Robots

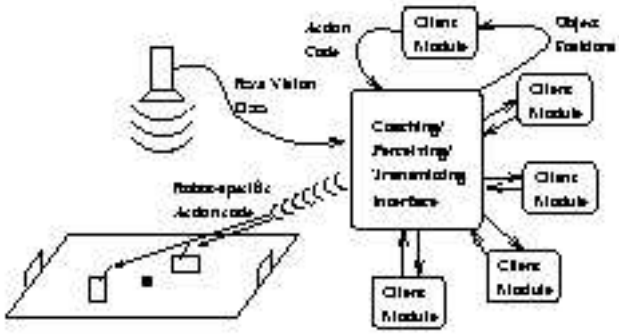


Figure 1.2: Control System

Figure 1.3: The Field

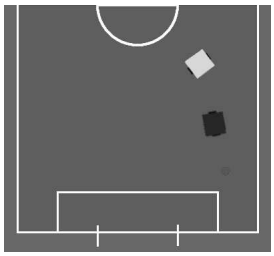


Figure 1.4: defender-annoy

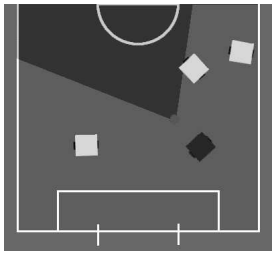


Figure 1.5: defender-clear

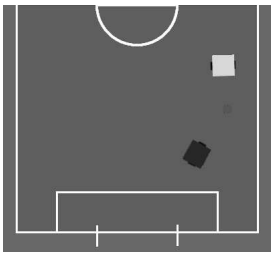


Figure 1.6: defender-annoy

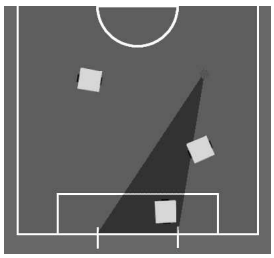


Figure 1.7: position-block

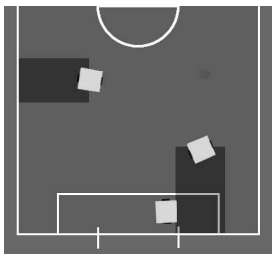


Figure 1.8: position-pass

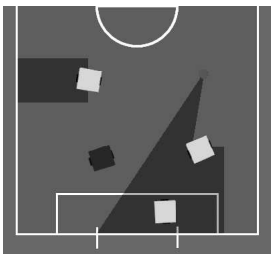


Figure 1.9: position-final

the task of scoring the goal.

Due to certain advantages in detection from computer vision point of view, the field as it generally is will be green, the ball in this case is orange(which is the only one I found under my drawer) and the robots will be blue in color with pink stribes at the top.

Chapter 2

Integrated System

The entire system is build upon the idea of using a camera, a central processing unit and a wireless transmission system that communicates between the computer and the robots. The camera is focussed on the entire field on which robots are placed. The camera captures the image at a rate of 30 frames/sec at 640x480 color resolution, and sends it over to the computer. The computer then processes this information and detects for the ball, and each individual robot. Once the robots are detected they are send information over the wireless to move towards the direction of the ball or to pass it to the other robots.

Since the computer is aware of the position of the robot at every possible instant of time, it guides the robot towards the ball. Once the robot reaches the ball the robot can turn and shoot the ball towards the other robot or towards the goal.

2.1 Microcontroller on Robot

The microcontroller I am using is ATMEL ATMEGA8L. It is a 28 Pin DIP. This microcontroller comes with a very good simulator AVRSTUDIO and a C compiler avrgcc which could be found on the ATMEL website. The microcontroller has 3 PWM, 4IO ports, both internal and facility for external clock, and you can design your own ISP for programming the chip. There is a very good free software you get for both Linux and Windows which lets you do ISP for all ATMEL chips (PONYPROG).

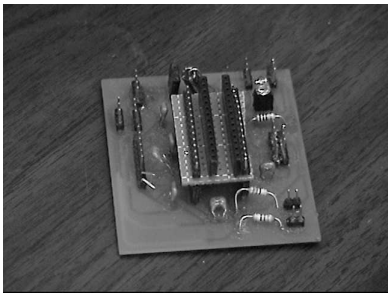


Figure 2.1: Experimental Design

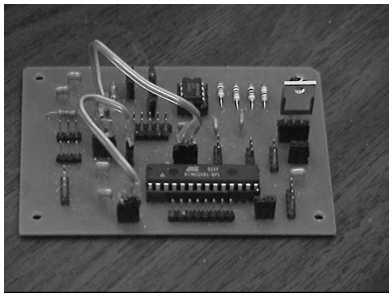


Figure 2.2: First Stage

2.2 Board Design

The board I designed involved using the PortD for IO, pins OC1A,OC1B,OC2 for generating the PWM for the motors and using A/D converter on the chip to get signal from the IR sharpcans. I also incorporated within the board ISP for programming the Microcontroller chip. I initially planned to use OC2 for generating a signal of 40Khz so that I could run my IRs. The reason was that SharpCan IR detector detects IR signal only at 40Khz. Since OC2 is incapable of generating signal of frequency as low as 40Khz I had to go for the 555 Timer. The 555 Timer is very easy, once you understand the equation for generating the particular frequency. I was looking into a possibility of using IRs for wall detection, and so I added it to my board as an option.

The boards (Fig 2.1 & 2.2) were the first stage board. I used first stage board(Fig2.2) for for running the servos. But that idea didn't work because of problems I faced with servos. So I had to design a motordriver board and for that I needed another microcontroller board. The motordriver board had 4 optoisolators and one 2993B Motor Driver chip which allows at max two motors to be controlled.

The second stage board turned out to be much smaller in size because I separated the 555 Timer. I could do that because all 555 needed was power supply and it didn't need any control from the microcontroller.

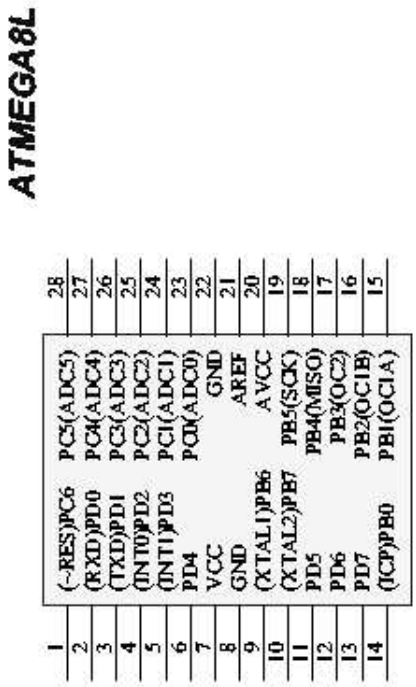


Figure 2.3: Microcontroller

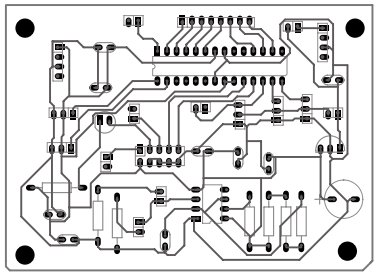


Figure 2.4: First Stage PCB

2.2.1 555 Timer

I had to use the 555 Timer because I could easily obtain 40Khz needed for IRLed. The reason for having 40Khz frequency was to be able to use the hacked Sharp Cans for distance detection.

Chapter 3

Mobile Platform

Designing the Mobile Platform was a difficult problem. The issues that I had were that I had no idea how big my Microcontroller board would be and how would I like to place. Another of my problem was to include the kick mechanism. I was planning to have a solenoid based system, which although being simple from mechanical design was hard to implement on the board. I experimented with it and it drew so much of current that within minutes I drained my entire battery. I then switched to the motor based mechanism. This meant creating a paddle for hitting the ball. Uriel and Rolando had already designed a paddle as part of the IEEE competition. On their permission I altered their design to fit into my robot. This meant incorporating it into the electronics too. Since Atmel already had 3 PWM I thought I could use it. The OC2 suffers from lack of Input Capture and so frequency cannot be brought down to 50 Hz needed for running the paddle. I had to again go for the 555 Timer to do the required.

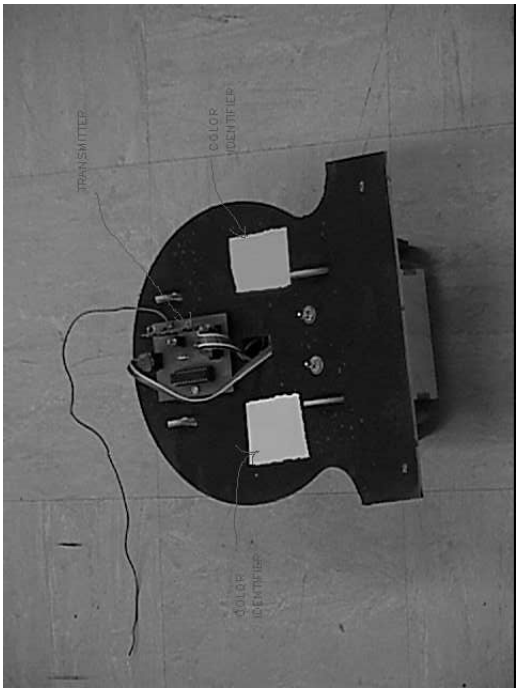


Figure 3.1: Robot TopView

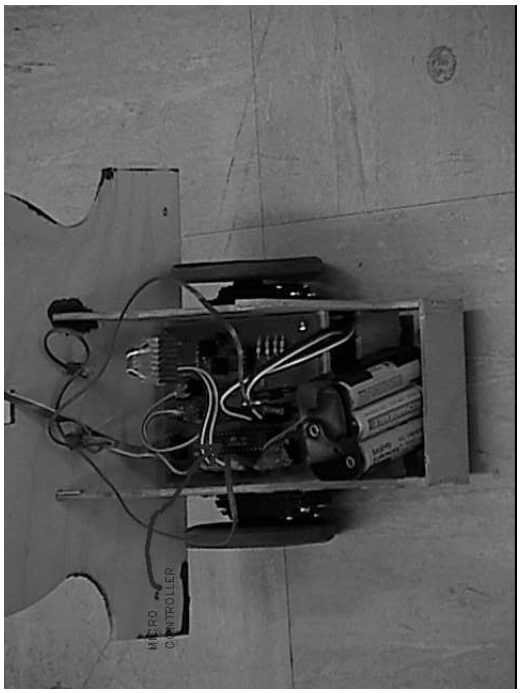


Figure 3.2: Robot Behind

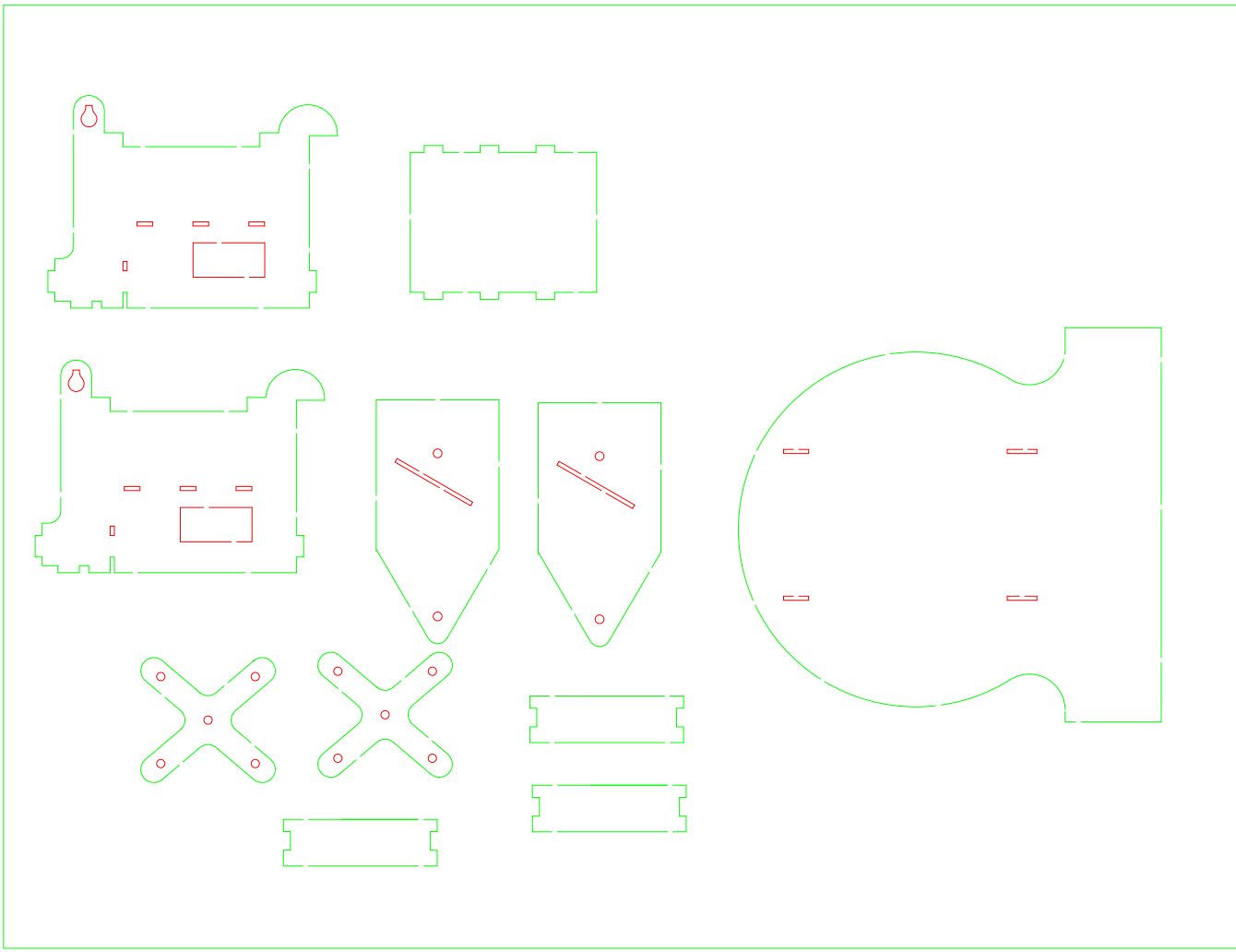


Figure 3.3: Platform

Chapter 4

Sensor

The sensors I have used are the central camera and IR(optional). The sensor basically involves the vision code which I have written for doing the stuff.

4.1 Vision

I have written vision code that uses a Hi-8 Sharp camera, a Meteor framegrabber and driver code. The part of getting the frame to process the image is changing the already available code Mike had written for his vision stuff. This code helps me in getting the frame and then I process the frame to get the information I need.

The image processing gives me the color of stripes which are placed over the robot and the ball. A frame generally contains RGB values of a pixel(basic unit of an image on computer). The code converts RGB into HSV space a much more useful and intensity independent space and I use H values to detect the color. The Hue(H) is a linear function on color, i.e. for every colour there is a single value of Hue. Then I take the mean value of all the pixels that contain the Hue of the stripe I have placed over the robot. This gives me roughly the location of both the stripes. These values then gives me two points on the robot and the ball gives me the third point. Using these two points on robot I identify the direction of motion and using the point on the ball I identify the angle robot makes from the ball. The points also helps me in knowing the distance between the robot and the ball.

Using the criteria of the distance and the angle between the robot, I give

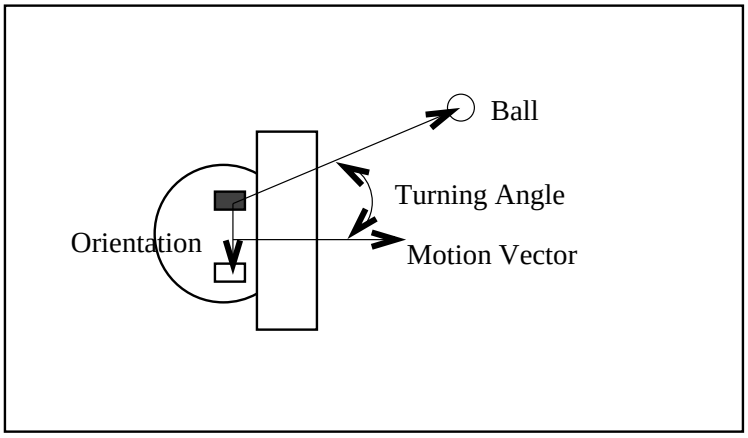


Figure 4.1: Ball Direction

the robot the desired motion to reach to the ball.

Chapter 5

Actuation

The actuators that I am using are the hacked GWS servo that can be found on MarkIII Robot as high torque servo. The hacked servos require a PWM at 20ms and one needs to calibrate the servos for the stop, forward and backward duty cycles.

There were some problems I found with the application I had in mind. The servos would drift a bit when I tried to stop them after running for some time using the PWM. The servos would drift adding noise to the system and would interfere with the receiver's signal. Once they would start drifting they would not respond to transmitter at all and would continue to do so. The way I found the solution for it was not letting the servos to stop throughout the entire game. This made vision algorithm way more complicated and difficult to work.

Figure 5.1: Servo

There is another solution to this problem and I have already designed the board for this. The solution is using the full hacked servo (DC Motors) and use motor driver chips to run the motors. I have finally been able to implement the motor driver to run motors. The motor drivers certainly give a better digital control and PWM required for it is not that difficult.

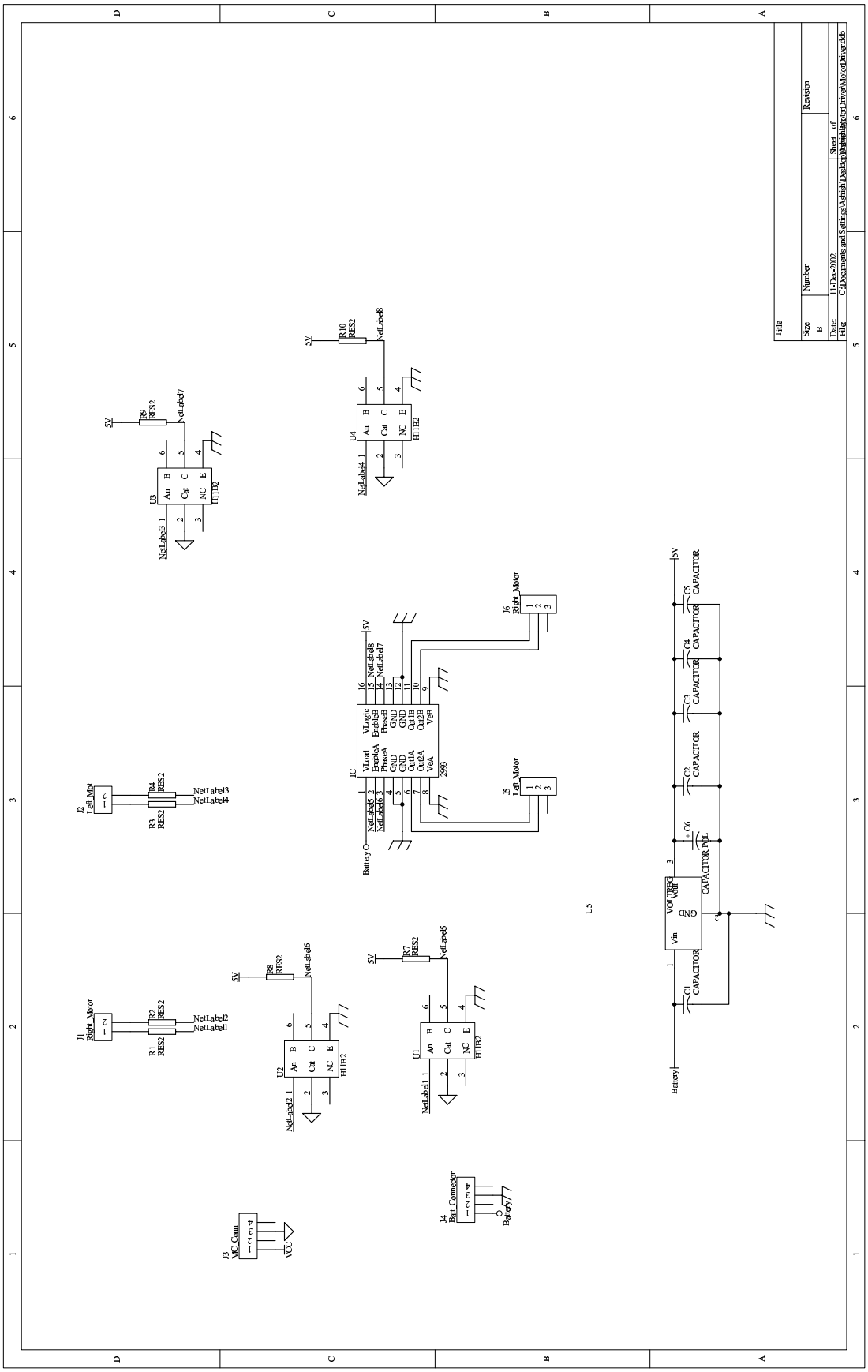


Figure 5.2: MotorDriver

Chapter 6

Behaviors

The idea of building multiple robots was to get them involved in a cooperative task. A task they could accompolish as a team. The robots can be assigned various jobs while still being part of the team. One of the robot can be forward, while two could defend or if I manage to build more robots I could actually define more tasks to them as a team.

One of the rather interesting lesson to learn was to see the robot oscillating trying to get the ball. I had not designed the robot with proportional control in mind. But when I went ahead and tried to give it the guidelines towards getting the ball, I actually ended up making a proportional controller which was the cause of my oscillations.

Chapter 7

Conclusion

The work involved was really an extremely rewarding experience. Realising the circuits was one part of the deal, while getting everything working was a totally different deal. I managed to get the robot move according to the wishes of the computer. I learned a lot about designing circuit and how design gets even more complicated when you have two things to design mechanical as well as electronics. The wireless communication which was extremely low power could detect real far distances, which I found to have more applications for the hobbyists.

7.1 Future Work

The work is useful from the point of doing team tasks. The idea could be used for controlling multiple lawn mowers for mowing the grass, it could be used as multiple waiters in a restaurant serving people and defining for each robot its own table and how to get to it. The multiple robots idea could be extended even far. Right now I have looked into from the possibility of one central camera guiding the robots. The other method that can be used is that each robot have its own camera and processing unit. I have looked into the idea of CMU cam to do the multiple cameras, but there is another technology which could be extremely useful from the point of processing. We can use these robots for doing more vision based stuff. Each robot could be placed with a camera with wireless(as cheap as $50) and the images could be transferred to a central computer over wireless where they are processed and then computer can send signals to robot to do the task(related to Shalom's Work). We could

use multiple vision on one camera using video quad processors which puts four images into one image. This is one possibilty I am going to seek in for next semester.

Appendix

Here is some code that I wrote for the microcontroller and some circuit diagrams. Also included is the vision code.

```
#ifndef SERVO_ATMEL_H
#define SERVO_ATMEL_H

#define SERVO_FREQ 10000
#define ZERO 0
#define ONE 1
#define SERVO_STOP 1000
#define SERVO_RIGHT 1
#define SERVO_LEFT 2000
#define FULL_RIGHT 100
#define FULL_LEFT -100
#define GAIN    10
#define Factor 100


extern void servo_motor_init(void);
extern void motor_speed(int mot_num, int SPEED);
extern void port_init(void);
extern void servo_motor_stop(void);
extern void servo_motor_restart(void);
extern void motor_turn(void);


#endif
```

Header File

```
/* Created By: Ashish Jain
 *
 *      These functions are used to initialize the PWM generator for use with a
 *      fully hacked servo or DC motor. They are build on code from Rolando Panez
 */

#include <io.h>
#include <math.h>
#include "Mega8Servo.h"

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

main()
{

u08 port_value;
u08 timerH,timerL,timerH_prev;
u08 count;
count=0;

servo_motor_init();
port_init();
//motor_speed(0,99);
 //motor_speed(1,99);

while(1)
{
port_value=inw(PIND);

if(port_value==0xA0)
{
outp(0<<PB4,PORTB);
outp(0<<PB5,PORTB);
 motor_speed(0,2);
 motor_speed(1,2);
}
else if(port_value==0x09)
{
 outp(ONE<<PB4,PORTB);
 motor_speed(0,40);
 motor_speed(1,40);
}
else if(port_value==0xE0)
{
 outp(ONE<<PB5,PORTB);
```

```
 motor_speed(1,2);
}
else if(port_value==0xC0)
{
 motor_turn();

}
else if(port_value==0xD0)
{
 outp(0<<PB4,PORTB);
 outp(0<<PB5,PORTB);
 motor_speed(0,50);
 motor_speed(1,50);

}
else
{
motor_speed(0,99);
 motor_speed(1,99);
}
}




}

void motor_turn(void)
{
int i,j,k,l;
outp(1<<PB4,PORTB);
 for(i=0;i<95;i++)
  {
   motor_speed(0,0);
   motor_speed(1,0);
   for(j=0;j<255;j++)
     for(k=0;k<255;k++)
      for(l=0;l<3;l++)
      {
      }
  }
}

void port_init(void)
{
unsigned char input=0,output=0xFF;

outw(DDRD, input);
```

```
}
void servo_motor_init(void)
{
	outw(ICR1L, SERVO_FREQ);		/* Set TOP to match 20ms period */
	outw(OCR1AL, SERVO_STOP);		/* Set PWM to 10% duty cycle on cha
A */
	outw(OCR1BL, SERVO_STOP);		/* Set PWM to 10% duty cycle on cha
B */
	outp((ONE<<COM1A1)|(ONE<<COM1B1), TCCR1A);		/* Set output
compare OC1A/OC1B clear on compare match, */
	outp((ONE<<WGM13)|(ONE<<CS11), TCCR1B);		/* store TOP in ICR1, a
prescaler Clk(I/O)/8. */
	outp((ONE<<PB1)|(ONE<<PB2)|(ONE<<PB4)|(ONE<<PB5), DDRB);

}


void motor_speed(int mot_num, int SPEED){
	u16 OCRTMP;
	s16 TMP_SPEED;
	u16 SERVO_SPEED;

	if(SPEED > FULL_RIGHT)
		SPEED = 100;
	else if(SPEED < FULL_LEFT)
		SPEED = -100;

	if(mot_num == 0){
		OCRTMP = inw(OCR1AL);
	/*	if(SPEED == 0)  // If Speed = 0 then stop motor
			SERVO_SPEED = SERVO_STOP;
		else if( SPEED >= FULL_RIGHT) // If speed = FULL_RIGHT then full spe
the right
			SERVO_SPEED = SERVO_RIGHT;
		else if( SPEED <= FULL_LEFT) // If speed = FULL_LEFT then full speed
left
			SERVO_SPEED = SERVO_LEFT;
		else	// If speed is less than 0 calculate percentage*/
			SERVO_SPEED =Factor*SPEED;
		//outw(OCR1AL, SERVO_SPEED);
		outw(OCR1AL, SERVO_SPEED);
	}
	else if(mot_num == 1){
		OCRTMP = inw(OCR1BL);
		/*if(SPEED == 0)		// If Speed = 0 then stop motor//
			SERVO_SPEED = SERVO_STOP;
```

Code for Microcontroller(Continued..)

```
            else if( SPEED >= FULL_RIGHT) // If speed = FULL_RIGHT then full spe
the right//
                SERVO_SPEED = SERVO_RIGHT;
            else if( SPEED <= FULL_LEFT) // If speed = FULL_LEFT then full speed
left//
                SERVO_SPEED = SERVO_LEFT;
            else     // If speed is less than 0 calculate percentage//*/
                SERVO_SPEED = Factor*SPEED;
            outw(OCR1BL, SERVO_SPEED);
            }

        return;
}


/*
if(port_value==0xA0)
  {
  while(1)
   {
     timerH_prev=timerH;
     timerH=inw(TCNT1H);
     timerL=inw(TCNT1L);
     if(timerL>0)
     count=count+1;
     if(count<=22)
      {
        motor_speed(1,-100);
        motor_speed(0,-100);
      }
     else if(count<=100&&count>22)
     {
       motor_speed(1,-100);
       motor_speed(0,100);
     }
     else
      break;
     if(count==254)
    count=0;
    for(i=0;i<5000;i++);
   }
  count=0;

  }
  else if(port_value==0xE0)
  {
  while(1)
```

```
    timerH_prev=timerH;
    timerH=inw(TCNT1H);
    timerL=inw(TCNT1L);
    if(timerL>0)
    count=count+1;
    if(count<=10)
     {
       motor_speed(1,-100);
       motor_speed(0,-100);
     }
    else if(count<=100&&count>10)
    {
      motor_speed(1,-100);
      motor_speed(0,100);
    }
    else
     break;
    if(count==254)
   count=0;
   for(i=0;i<5000;i++);
  }
 count=0;

}
else if(port_value==0x0E)
{
  motor_speed(0,100);
  motor_speed(1,-100);

}
 else if(port_value==0x09)
{
while(1)
  {
    timerH_prev=timerH;
    timerH=inw(TCNT1H);
    timerL=inw(TCNT1L);
    if(timerL>0)
    count=count+1;
    if(count<=10)
     {
       motor_speed(1,100);
       motor_speed(0,100);
     }
    else if(count<=100&&count>10)
    {
      motor_speed(1,-100);
      motor_speed(0,100);
```

```
    }
    else
     break;
    if(count==254)
   count=0;
   for(i=0;i<5000;i++);
  }
 count=0;

}
else if(port_value==0x09)
{
while(1)
  {
    timerH_prev=timerH;
    timerH=inw(TCNT1H);
    timerL=inw(TCNT1L);
    if(timerL>0)
    count=count+1;
    if(count<=10)
     {
       motor_speed(1,100);
       motor_speed(0,100);
     }
    else if(count<=100&&count>10)
    {
      motor_speed(1,-100);
      motor_speed(0,100);
    }
    else
     break;
    if(count==254)
   count=0;
   for(i=0;i<5000;i++);
  }
 count=0;

}
else if(port_value==0x0B)
{
while(1)
  {
    timerH_prev=timerH;
    timerH=inw(TCNT1H);
    timerL=inw(TCNT1L);
    if(timerL>0)
    count=count+1;
    if(count<=22)
```

```
        {
          motor_speed(1,50);
          motor_speed(0,50);
        }
       else if(count<=100&&count>22)
        {
         motor_speed(1,-100);
         motor_speed(0,100);
        }
       else
        break;
       if(count==254)
      count=0;
      for(i=0;i<5000;i++);
    }
   count=0;
  }
 else
  {
  motor_speed(0,100);
  motor_speed(1,-100);
  }

*/
```

Code for Microcontroller

```
convert(unsigned char r,unsigned char g,unsigned char b,double *h,double *s,doubl
{
 double rd,gr,bl;

 rd=(double)r/255;
 gr=(double)g/255;
 bl=(double)b/255;
 RGBtoHSV(rd,gr,bl,h,s,v);
}



char *rectangle(long x,long y,char *frame,unsigned char r,unsigned char g,unsigned
b)
{
 int k,l;
 if(x<=480&&x>=478)
 {
   x=x-2;
 }
 else if(x>=0&&x<=5)
 {
  x=x+2;
 }
 if(y<=640&&y>=638)
 {
   y=y-2;
 }
 else if(y>=0&&y<=5)
 {
  y=y+2;
 }

     for(k=x-2;k<x+2;k++)
       for(l=y-2;l<y+2;l++)
        {

        frame[4*(640*k+l)]=b;
        frame[4*(640*k+l)+1]=g;
        frame[4*(640*k+l)+2]=r;
        }

     return(frame);
}

int dist_l(int xp,int yp,int i)
```

```
 double xl1,yl1,dis,val;
 double xdis,ydis;
 if(i<4)
 {
  xl1=(double)(x[i]-x[i-1]);
  yl1=(double)(y[i]-y[i-1]);
 }
 else
 {
  xl1=(double)(x[i-1]-x[0]);
  yl1=(double)(y[i-1]-y[0]);
 }
 xdis=(double)(xp-x[i-1]);
 ydis=(double)(yp-y[i-1]);
 dis=fabs(xdis*yl1-ydis*xl1);
 val=sqrt(xl1*xl1+yl1*yl1);
 if(val!=0)
 dis=dis/val;
 else
  dis=0;

 return((int)dis);

}

void get_out(void)
{
 int i;
 unsigned char send;
     val=0xC0;
     send=~val;
     outb(send,base);
     printf("left");
     usleep(2000);
    for(i=0;i<5;i++)
    {
      val=0xC0;
      send=~val;
      outb(send,base);
      usleep(2000);
    }


}
char *change_frame(char *frame)
{
int i,j;
unsigned char r,g,b;
```

```
double h,s,v;
long sumi,sumj;
long sumyi,sumyj;
long sumwi,sumwj;
int count=0,county=0,countw=0;
int xp,yp,xn,yn;
unsigned char send,val;
int dis1,dis2,dis3,dis4;
long disball;
double angle;
double cc,ii,ee;
double dpvect,cpvect;
double mdla,mdlb,anglebt;
sumi=0;
sumj=0;
sumyi=0;
sumyj=0;
sumwi=0;
sumwj=0;
 for(i=0;i<480;i++)
  for(j=0;j<640;j++)
  {
   r=frame[4*(640*i+j)+2];
   g=frame[4*(640*i+j)+1];
   b=frame[4*(640*i+j)+0];
   convert(r,g,b,&h,&s,&v);
  // if(abs(r-g)<10&&abs(g-b)<10&&abs(r-b)<10&&r>180)
  // if(v>0.7&&b>170&&g>190&&r>180)
   if(v>0.7&&s<12&&g>190&&b>170&&r>170)
    {
    frame[4*(640*i+j)+2]=0;
    frame[4*(640*i+j)+1]=255;
    frame[4*(640*i+j)+0]=0;
      sumyi=sumyi+i;
      sumyj=sumyj+j;
      county++;
    }
  else if((abs(h-310)<45)&&(fabs(v-0.75)<0.2))
   {
    frame[4*(640*i+j)+2]=0;
    frame[4*(640*i+j)+1]=0;
    frame[4*(640*i+j)+0]=255;
      sumi=sumi+i;
      sumj=sumj+j;
      count++;

   }
  /*else if((abs(h-60)<5)&&abs(r-g)<10&&s>0.75&&g>160)
```

```
  {
   frame[4*(640*i+j)+2]=0;
   frame[4*(640*i+j)+1]=255;
   frame[4*(640*i+j)+0]=255;
     sumyi=sumyi+i;
     sumyj=sumyj+j;
     county++;

  }*/
  else if(abs(h-20)<10&&b<50&&g>25&&r>80)
  {
   frame[4*(640*i+j)+2]=255;
   frame[4*(640*i+j)+1]=255;
   frame[4*(640*i+j)+0]=255;
     sumwi=sumwi+i;
     sumwj=sumwj+j;
     countw++;
  }
/*  else
  {
   frame[4*(640*i+j)+2]=0;
   frame[4*(640*i+j)+1]=0;
   frame[4*(640*i+j)+0]=0;
  }
 */
 }
if(count>0)
{
 sumi=sumi/count;
 sumj=sumj/count;
}
else
{
 sumi=x[0];
 sumj=y[0];
}
if(county>0)
{
 sumyi=sumyi/county;
 sumyj=sumyj/county;
}
else
{
 sumyi=x[0];
 sumyj=y[0];
}
if(countw>0)
{
```

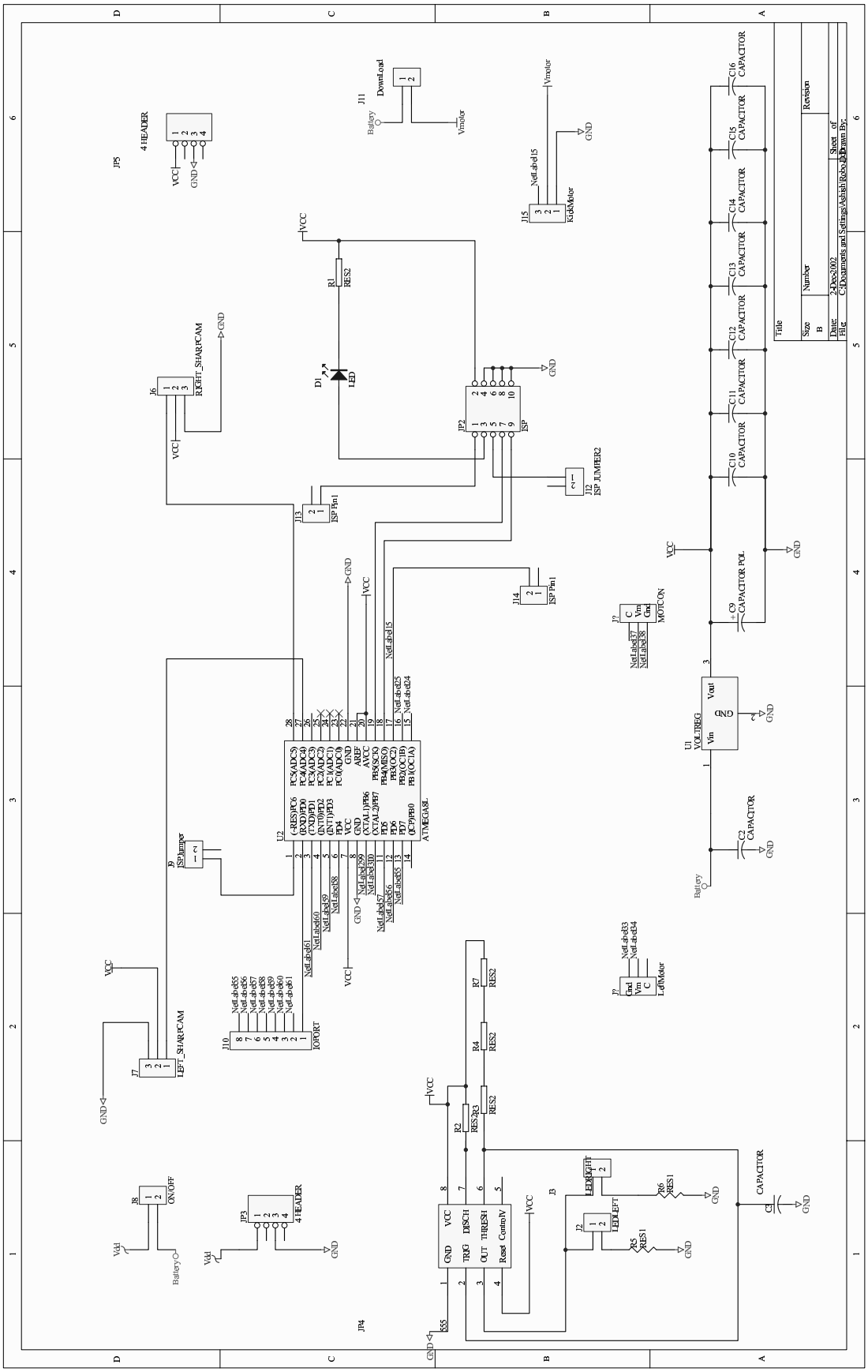
```
 sumwi=sumwi/countw;
 sumwj=sumwj/countw;
}
else
{
 sumwi=240;
 sumwj=320;
}
xp=sumj;
yp=sumi;
xn=sumwj;
yn=sumwi;


angle=180*atan2((double)(sumyi-sumi),(double)(sumyj-sumj))/3.142;
mdla=sqrt((sumyj-sumj)*(sumyj-sumj)+(sumyi-sumi)*(sumyi-sumi));
mdlb=sqrt((sumwj-sumj)*(sumwj-sumj)+(sumwi-sumi)*(sumwi-sumi));
dpvect=(-(sumyj-sumj)*(sumwi-sumi)+(sumyi-sumi)*(sumwj-sumj))/(mdla*mdlb);
cpvect=(-(sumyj-sumj)*(sumwj-sumj)-(sumyi-sumi)*(sumwi-sumi))/(mdla*mdlb);
anglebt=180*atan2(cpvect,dpvect)/3.142;

printf("dotproduct=%d\n",dpvect);
printf("crossproduct=%d\n",cpvect);
printf("distance from ball=%lf\n",mdlb);
dis1=dist_l(sumj,sumi,1);
dis2=dist_l(sumj,sumi,2);
dis3=dist_l(sumj,sumi,3);
dis4=dist_l(sumj,sumi,4);

if(dis1<distance)
 {
  get_out();
 }
else if(dis2<distance)
{
  get_out();
}
else if(dis3<distance)
{
 get_out();
}
else if(dis4<distance)
 {
  get_out();
 }
else
{
  if(anglebt>-20&&anglebt<10)
```

```
    {
     if(mdlb>75)
       {
          val=0xA0;
          send=~val;
          outb(send,base);
          printf("forward");
          usleep(2000);
       }
     else
      {
          val=0x03;
          send=~val;
          outb(send,base);
          printf("stop");
          usleep(2000);


      }
     }
   else if(anglebt<-20&&angle>-160)
    {
          val=0xE0;
          send=~val;
          outb(send,base);
          printf("Turn left");
          usleep(2000);
    }
   else
   {
          val=0x09;
          send=~val;
          outb(send,base);
          printf("Turn left");
          usleep(2000);
   }



/*
   val=0xA0;
   send=~val;
   outb(send,base);
   printf("Moving");
 */
}
```

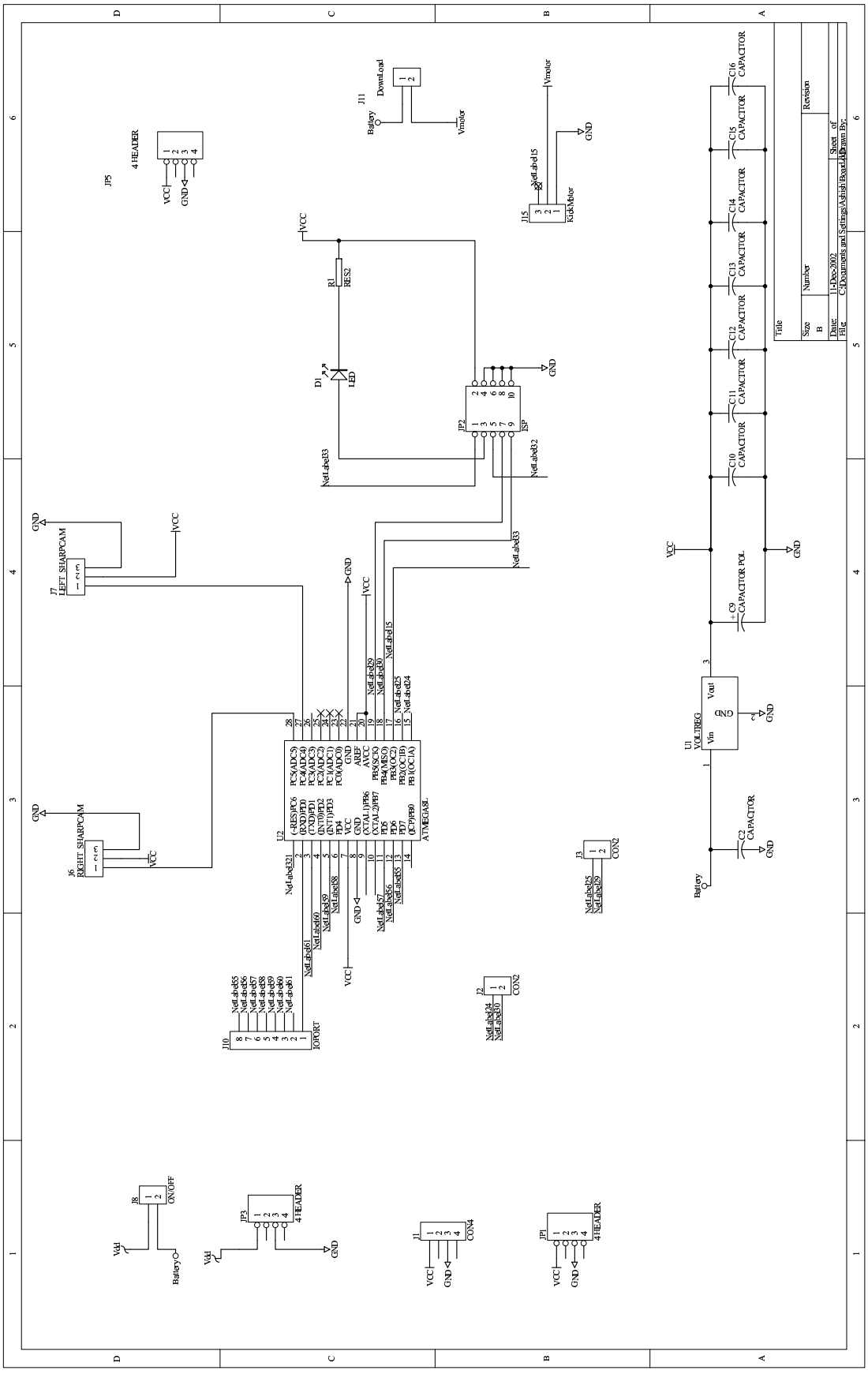
Sample Vision Code

```
 printf("dis1=%d dis2=%d dis3=%d
dis4=%d",dist_l(sumj,sumi,1),dist_l(sumj,sumi,2),dist_l(sumj,sumi,3),dist_l(sumj,sum
 printf(" %d %d\n",sumi,sumj);
 printf("x[0]=%d y[0]=%d\n",x[0],y[0]);
 frame=rectangle(sumi,sumj,frame,255,0,0);
 frame=rectangle(sumyi,sumyj,frame,0,0,255);
 frame=rectangle(sumwi,sumwj,frame,0,0,0);
 printf("w=%d %d \n",sumwi,sumwj);
 printf("angle=%lf anglebt=%lf\n",angle,anglebt);
return(frame);
}


/*****************************************************************************************/
int process_frame(char *frame1, char *frame2)
{
  int i, n = 1;
  int j,k,l;
  int count=0;
  int xval,yval,ct2;
  unsigned char hue;
  double rd,gr,bl;
  int xi,yj;
xval=0;
yval=0;
ct2=0;
frame1=change_frame(frame1);
#ifdef X_DISPLAY_ACTITRACK
  update_xdisplay(frame1, &BoxAll, 0, &BoxAll, n);
#endif

  return(1);
}
```

Sample Vision Code



First Stage Board



Second Stage Board