```
/*****************************************************************************
                                Marco Assignment 2
Author: Jonathan Ambrose & Andrew Fester
Date: 17/04/08 17:44
Version: Final Release
Copyright: JAAF Software
Description: A program


Compile: gcc -pedantic -Wall -lcomedi -lncurses -lm irsw.c -o ir
Run Code: ./jamlight
*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <getopt.h>
#include <ctype.h>
#include <comedi.h>
#include <comedilib.h>
#include <unistd.h>
#include <math.h>
#include <curses.h>
#include <ctype.h>

/***************** # defines**************/

#define EYESOCKET 0x80
#define EYE 3
#define WRITEFILE "w"
#define READFILE "r"
#define LOGFILE "./path.txt"
#define BRIGHTLIGHT 2150

/****Menu #defines***/

#define MENU 0
#define REPLAY 1
#define LINEFOLLOW 2
#define LIGHTFOLLOW 3
#define NORMAL 4
#define ARTIFICIAL 5

/****Replay menu****/

#define REPLAYMENU 0
#define RECORD 1
#define SRECORD 2
#define REPLAYING 3
#define ERASE 4
#define HPLAY 5
#define HREPLAY 6

/***AI States*****/

#define LINE 0
#define RANDOM 1
#define LIGHT 2
#define CONTROLER 3

/*****************************Prototypes**********************************/

void welcome_menu (void);
int init(comedi_t **device,unsigned int *digitaldevice,unsigned int *subdevice);
int wlfollow(unsigned int digitaldevice, comedi_t *device, int read, int aref,
         int *scanrange,int *scantimer);
int lightfollow(unsigned int *digitaldevice, unsigned int *subdevice,
                comedi_t **device, int *read, int *aref,int *range,int *lighttimer);
int bumpers(unsigned int digitaldevice, comedi_t *device, int read,
             int aref, int *hit);
void joy_cali (comedi_t *device,unsigned subdevice, int range, int aref,
           int horizontal,int vertical,int *hori,int *vert,
           int *maxhori, int *maxvert);
void driving(comedi_t *device, unsigned int subdevice, int horizontal,
         int vertical,int range, int aref, int *maxvert, int *maxhori,
         char *filename,int *motrrgt,int *motrlft,int vert, int hori);
/*****************************************************************************/

int main(void)
{
  int i;
  char *filename = "/dev/comedi0";
  comedi_t *device;
  unsigned int subdevice = 0;
  unsigned int digitaldevice = 2;

  int maxhori;
  int maxvert;
  float horidiv=0, vertdiv=0;
  int range = 0;
  int ret;
  int horizontal= 1;
  int vertical= 0;
  int aref = AREF_GROUND;
  /*IR setup*/

  int read = 0x00;
  int stop = 2048;
```

```
101
102     int s;
103     int vert=0, hori=0;
104     int horiarray, vertarray;
105     int motrlft, motrrgt;
106     int array [6][3] =
107       {
108         {4073 , 4073 , 2048}, /*{r-lfw, r-fw    , r-rfw},*/
109         {4073 , 2048 ,   20}, /*{r-l  , r-center, r-rb },*/
110         {  20 ,   20 , 2048}, /*{r-lbw, r-bw    , r-rbw},*/
111         {2048 , 4073 , 4073}, /*{l-lfw, l-fw    , l-rfw},*/
112         {  20 , 2048 , 4073}, /*{l-l  , l-center, l-r  },*/
113         {2048 ,   20 ,   20}, /*{l-lbw, l-bw    , l-rbw},*/
114       };
115     FILE *log;
116     static int  state = MENU;
117     static int  submenu = REPLAYMENU;
118     int error;
119     int move= 1;
120     int temp;
121     int b=0;
122     int t=0;
123     int scanrange=2;
124
125     int hit;
126     int ai = 0;
127     int wander1=0;
128     int wander2=0;
129     int rando;
130     int run;
131     int scantimer=0;
132     int lighttimer=0;
133     int randomtimer=0;
134
135
136     initscr();
137     timeout(10);
138     cbreak();
139     noecho();
140     clear();
141
142     init(&device, &digitaldevice, &subdevice); /* Call initialise device */
143     joy_cali(device,subdevice,range,aref,horizontal,vertical,&hori,&vert,
144          &maxhori,&maxvert); /*Call calibrate joystick function*/
145     printw("%d %d",hori,vert);
146
147     comedi_data_write(device,1,0,1,aref,2048); /* STOP Left Motors */
148     comedi_data_write(device,1,1,1,aref,2048); /* STOP Right Motors */
149
150     /* Print the following menu on the screen*/
151     printw("---------------MENU-------------\n");
152     printw("What do you want to do?\n");
153     printw("R - Replay menu\n");
154     printw("L - Line follow\n");
155     printw("T - Light follow\n");
156     printw("B - Bumper\n");
157     printw("M - Menu\n");
158     /*Printw does not automatically display printw's on the screen so
159       refresh(); is needed.*/
160
161     refresh();
162
163     while(1)
164       {
165         clear(); /*Clear all contents of screen*/
166
167         switch(state)
168       {
169       case MENU:
170         printw("---------------MENU-------------\n");
171         printw("What do you want to do?\n");
172         printw("R - Replay menu\n");
173         printw("L - Line follow\n");
174         printw("T - Light follow\n");
175         printw("N - Controler\n");
176         printw("A - ai\n");
177         printw("M - Menu\n");
178         refresh();
179         state = toupper(getchar());
180          if(state == 'R')state = REPLAY;
181          else if(state == 'L') state = LINEFOLLOW;
182          else if(state == 'T') state = LIGHTFOLLOW;
183          else if(state == 'N') state = NORMAL;
184          else if(state == 'M') state = MENU;
185          else if(state == 'A') state = ARTIFICIAL;
186
187
188          break;
189
190       case REPLAY:
191
192         switch(submenu)
193           {
194           case REPLAYMENU:
195             printw("---------------REPLAY-------------\n");
196             printw("What do you want to do?\n");
197             printw("R - Replay menu\n");
198             printw("\t  1 - Record\n");
199             printw("\t  2 - Stop recording\n");
200             printw("\t  3 - Replay\n");
```

```
            printw("\t  4 - Erase\n");
            printw("L - Line follow\n");
            printw("T - Light follow\n");
            printw("N - Controler\n");
            printw("A - ai\n");
            printw("M - Menu\n");
            refresh();

            submenu = toupper(getchar());
            if(submenu == 'R')state = REPLAY;
            else if(submenu == 'L'){ state = LINEFOLLOW;submenu = 0;}
            else if(submenu == 'T'){ state = LIGHTFOLLOW;submenu = 0;}
            else if(submenu == 'N'){ state = NORMAL;submenu = 0;}
            else if(submenu == 'M'){ state = MENU;submenu = 0;}
            else if(submenu == '0')submenu = REPLAYMENU;
            else if(submenu == '1')submenu = RECORD;
            else if(submenu == 'S')submenu = SRECORD;
            else if(submenu == '3')submenu = REPLAYING;
            else if(submenu == '4')submenu = ERASE;
            else if(submenu == '5')submenu = HPLAY;
            else if(submenu == '6')submenu = HREPLAY;

            break;
          case RECORD:

            log = fopen(LOGFILE , WRITEFILE);
            submenu = 5;




            break;
          case 'S':
            submenu = REPLAYMENU;
            fclose(log);
            break;
          case REPLAYING:

            log = fopen(LOGFILE, READFILE);
            error = 0;
            submenu = HREPLAY;

            break;
          case ERASE:
            log = fopen(LOGFILE, WRITEFILE);
            fclose(log);
            submenu = REPLAYMENU;

            break;
          case HPLAY:
            driving(device,subdevice,horizontal,vertical,range,aref,&maxhori,
                &maxvert,filename, &motrrgt,&motrlft,vert,hori);

            fprintf(log,"%d %d\n",motrrgt , motrlft);
            printw("---------------RECORDING--------------\n");
            printw("What do you want to do?\n");
            printw("R - Replay menu\n");
            printw("\t  1 - Record\n");
            printw("\t  2 - Stop recording\n");
            printw("\t  3 - Replay\n");
            printw("\t  4 - Erase\n");
            printw("L - Line follow\n");
            printw("T - Light follow\n");
            printw("N - Controler\n");
            printw("A - ai\n");
            printw("M - Menu\n");
            refresh();

            submenu = toupper(getch());

            if(submenu == 'R')state = REPLAY;
            else if(submenu == 'L'){ state = LINEFOLLOW;submenu = 0;}
            else if(submenu == 'T'){ state = LIGHTFOLLOW;submenu = 0;}
            else if(submenu == 'N'){ state = NORMAL;submenu = 0;}
            else if(submenu == 'M'){ state = MENU;submenu = 0;}
            else if(submenu == '0')submenu = REPLAYMENU;
            else if(submenu == '1')submenu = RECORD;
            else if(submenu == 'S')submenu = SRECORD;
            else if(submenu == '3')submenu = REPLAYING;
            else if(submenu == '4')submenu = ERASE;
            else if(submenu == '5')submenu = HPLAY;
            else if(submenu == '6')submenu = HREPLAY;
            else submenu = HPLAY;

            break;
          case HREPLAY:
            printw("---------------REPLAYING--------------\n");
            printw("What do you want to do?\n");
            printw("R - Replay menu\n");
            printw("\t  1 - Record\n");
            printw("\t  2 - Stop recording\n");
            printw("\t  3 - Replay\n");
            printw("\t  4 - Erase\n");
            printw("L - Line follow\n");
            printw("T - Light follow\n");
            printw("N - Controler\n");
            printw("A - ai\n");
            printw("M - Menu\n");
            refresh();

            submenu = toupper(getch());
```

```
            if(submenu == 'R')state = REPLAY;
            else if(submenu == 'L'){ state = LINEFOLLOW;submenu = 0;}
            else if(submenu == 'T'){ state = LIGHTFOLLOW;submenu = 0;}
            else if(submenu == 'N'){ state = NORMAL;submenu = 0;}
            else if(submenu == 'M'){ state = MENU;submenu = 0;}
            else if(submenu == '0')submenu = REPLAYMENU;
            else if(submenu == '1')submenu = RECORD;
            else if(submenu == '2')submenu = SRECORD;
            else if(submenu == '3')submenu = REPLAYING;
            else if(submenu == '4')submenu = ERASE;
            else if(submenu == '5')submenu = HPLAY;
            else if(submenu == '6')submenu = HREPLAY;
            else submenu = HREPLAY;


            fscanf(log,"%d %d",&motrrgt , &motrlft);
            t++;
            usleep(100000);
            printw("%d %d",motrrgt,motrlft);
            comedi_data_write(device,1,0,1,aref,motrlft);
            comedi_data_write(device,1,1,1,aref,motrrgt);
            printw("%d",t);
            refresh();


            /* submenu = REPLAYMENU;*/

            break;
          case 'M':
            state = MENU;
            break;
          default:
            printw("---------------REPLAY--------------\n");
            printw("What do you want to do?\n");
            printw("R - Replay menu\n");
            printw("\t  1 - Record\n");
            printw("\t  2 - Stop recording\n");
            printw("\t  3 - Replay\n");
            printw("\t  4 - Erase\n");
            printw("L - Line follow\n");
            printw("T - Light follow\n");
            printw("N - Controler\n");
            printw("A - ai\n");
            printw("M - Menu\n");
            refresh();

            submenu = toupper(getchar());
            if(submenu == 'R')state = REPLAY;
            else if(submenu == 'L'){ state = LINEFOLLOW;submenu = 0;}
            else if(submenu == 'T'){ state = LIGHTFOLLOW;submenu = 0;}
            else if(submenu == 'N'){ state = NORMAL;submenu = 0;}
            else if(submenu == 'M'){ state = MENU;submenu = 0;}
            else if(submenu == '0')submenu = REPLAYMENU;
            else if(submenu == '1')submenu = RECORD;
            else if(submenu == '2')submenu = SRECORD;
            else if(submenu == '3')submenu = REPLAYING;
            else if(submenu == '4')submenu = ERASE;
            else if(submenu == '5')submenu = HPLAY;
            else if(submenu == '6')submenu = HREPLAY;
            break;
          }
        break;
      case LINEFOLLOW:
        printw("---------------WHITE-LINE-FOLLOW--------------\n");
        printw("What do you want to do?\n");
        printw("R - Replay menu\n");
        printw("L - Line follow\n");
        printw("T - Light follow\n");
        printw("N - controler\n");
        printw("A - ai\n");
        printw("M - Menu\n");
        refresh();
        state = toupper(getch());
        if(state == 'R')state = REPLAY;
        else if(state == 'L') state = LINEFOLLOW;
        else if(state == 'T') state = LIGHTFOLLOW;
        else if(state == 'N') state = NORMAL;
        else if(state == 'M') state = MENU;
        else state = LINEFOLLOW;


        printw("%d",t);
        refresh();


        wlfollow(digitaldevice,device, read, aref,&scanrange,&scantimer);
        bumpers(digitaldevice, device, read, aref,&hit);
        refresh();

        break;

      case LIGHTFOLLOW:
        printw("---------------LIGHT-FOLLOW--------------\n");
        printw("What do you want to do?\n");
        printw("R - Replay menu\n");
        printw("L - Line follow\n");
        printw("T - Light follow\n");
        printw("N - controler\n");
        printw("A - ai\n");
        printw("M - Menu\n");
```

```
401
402         state = toupper(getch());
403         if(state == 'R')state = REPLAY;
404         else if(state == 'L') state = LINEFOLLOW;
405         else if(state == 'T') state = LIGHTFOLLOW;
406         else if(state == 'N') state = NORMAL;
407         else if(state == 'M') state = MENU;
408         else state = LIGHTFOLLOW;
409
410         printw("%d",i);
411         refresh();
412         lightfollow(&digitaldevice,&subdevice, &device, &read, &aref,&range,&lighttimer);
413         bumpers(digitaldevice, device, read, aref,&hit );
414
415         break;
416       case NORMAL:
417
418         printw("---------------CONTROLER--------------\n");
419         printw("What do you want to do?\n");
420         printw("R - Replay menu\n");
421         printw("L - Line follow\n");
422         printw("T - Light follow\n");
423         printw("N - controler\n");
424         printw("A - ai\n");
425         printw("M - Menu\n");
426
427         driving(device,subdevice,horizontal,vertical,range,aref,&maxhori,&maxvert,filename, &motrrgt,&motrlft,vert,hori);
428         bumpers(digitaldevice, device, read, aref,&hit );
429
430         state = toupper(getch());
431
432         if(state == 'R')state = REPLAY;
433         else if(state == 'L') state = LINEFOLLOW;
434         else if(state == 'T') state = LIGHTFOLLOW;
435         else if(state == 'N') state = NORMAL;
436         else if(state == 'M') state = MENU;
437         else state = NORMAL;
438
439         break;
440         case'ARTIFICIAL':
441           switch(ai)
442             {
443             case LINE:
444           /*follow the line until marco loses the line*/
445           /*bumpers will hand control back to the user*/
446           wlfollow(digitaldevice,device, read, aref,&scanrange,&scantimer);
447           if(scantimer == 10)
448             {
449               ai=RANDOM;
450               scantimer =0;
451             }
452           bumpers(digitaldevice, device, read, aref,&hit );
453           if (hit == 1)ai=CONTROLER;
454
455           break;
456             case RANDOM:
457           /*moves 5 random movements in order to move to another area*/
458           /*uses array of motor values to randomly move to another area
459             to find the line*/
460           /*bumpers will hand control back to the user*/
461           for(run=0;run<5;run++)
462             {
463
464               rando = (int)(10.0 * rand()/(RAND_MAX + 1.0));
465
466               wander1 = wander1 + rando;
467               if(wander1 >3)wander1 =0;
468
469               rando = (int)(10.0 * rand()/(RAND_MAX + 1.0));
470
471               wander2 = wander2 + rando;
472               if(wander2 > 3)wander2 =0;
473
474               usleep(100000);
475               motrrgt = array[wander1][wander2];
476               comedi_data_write(device,1,0,1,aref,motrrgt);
477               motrlft = array[wander1+3][wander2];
478               comedi_data_write(device,1,1,1,aref,motrlft);
479               bumpers(digitaldevice, device, read, aref,&hit );
480               if (hit == 1)ai=CONTROLER;
481
482             }
483           randomtimer++;
484           ai = LIGHT;
485
486           break;
487             case LIGHT:
488           /*after mvoing away from the line find light and move towards
489             if cannot find move another random path*/
490           /*if the light is not found test for a line again by returning
491             to the LINE state*/
492           /*bumpers will hand control back to the user*/
493
494           lightfollow(&digitaldevice,&subdevice, &device, &read, &aref,&range,&lighttimer);
495           if (lighttimer == 5)/*stop light follow from looping and do another
496                         random move*/
497             {
498               ai=RANDOM;
499               lighttimer =0;
500             }
```

```
          if(randomtimer == 2)/*stop random movementand go back to line */
          {
            ai=LINE;
            randomtimer = 0;
          }
          bumpers(digitaldevice, device, read, aref,&hit );
          if (hit == 1)ai=CONTROLER;

          break;
            case CONTROLER:
          /*user control with access tot ehmenu for other functionality*/
          driving(device,subdevice,horizontal,vertical,range,aref,&maxhori,
              &maxvert,filename, &motrrgt,&motrlft,vert,hori);

          printw("---------------MENU--------------\n");
          printw("What do you want to do?\n");
          printw("R - Replay menu\n");
          printw("L - Line follow\n");
          printw("T - Light follow\n");
          printw("N - Controler\n");
          printw("A - ai\n");
          printw("M - Menu\n");
          refresh();
          state = toupper(getch());
          if(state == 'R')state = REPLAY;
          else if(state == 'L') state = LINEFOLLOW;
          else if(state == 'T') state = LIGHTFOLLOW;
          else if(state == 'N') state = NORMAL;
          else if(state == 'M') state = MENU;

          break;
            default:
          break;
            }

         break;
      default:
        printw("---------------MENU--------------\n");
        printw("What do you want to do?\n");
        printw("R - Replay menu\n");
        printw("L - Line follow\n");
        printw("T - Light follow\n");
        printw("N - Controler\n");
        printw("M - Menu\n");
        refresh();
        state = toupper(getchar());
        if(state == 'R')state = REPLAY;
        else if(state == 'L') state = LINEFOLLOW;
        else if(state == 'T') state = LIGHTFOLLOW;
        else if(state == 'N') state = NORMAL;
        else if(state == 'M') state = MENU;
        break;
      }

        refresh();
      }


    return 0;
  }

  /*********************************FUNCTIONS**********************************/

  /*initialize the rack*/

  int init(comedi_t **device, unsigned int *digitaldevice, unsigned int *subdevice)
  {

    int i;
    int retval;
    int stype;
    char *filename = "/dev/comedi0";

    *device=comedi_open(filename);

    if(!*device)
      {
        comedi_perror(filename);
        exit(0);
      }

    for(i=0;i<8;i++)

      retval=comedi_dio_config(*device,*digitaldevice,i,COMEDI_OUTPUT);

    /* check we have a valid bidirection DIO port subdevice*/


    stype = comedi_get_subdevice_type(*device,*digitaldevice);
    if(stype!=COMEDI_SUBD_DIO)
      {
        printw("%d is not a digital I/O subdevice\n",*subdevice);
        exit(1);
      }

    return 0 ;

  }

  /*****************************************************************************
```

```
* Function name : WHITE LINE FOLLOW (TYPE arg1, TYPE arg2…)
* returns : return value description
* arg1 : description
* arg2 : description
* Created by : Jon Ambrose & Andrew Fester
* Date created : date
* Description : detailed description
* Notes : restrictions, odd modes
*******************************************************************************/
int wlfollow(unsigned int digitaldevice, comedi_t *device, int read, int aref,
             int *scanrange, int *scantimer )
{
  unsigned int irsw1;
  unsigned int irsw2;
  int i;


  /*READ IRSW1 - D2*/
  irsw1=0; /*set irsw1 to 0*/

  comedi_dio_bitfield(device,digitaldevice,read,&irsw1);

  irsw1 >>=8;
  irsw1 =~irsw1;
  irsw1 =irsw1 & 0x04;

  /*READ IRSW2 - D3*/
  irsw2=0;

  comedi_dio_bitfield(device,digitaldevice,read,&irsw2);

  irsw2 >>=8;
  irsw2 =~irsw2;
  irsw2 =irsw2 & 0x08;


  switch(irsw1)/*left sensor*/
    {
    case 0:
      switch(irsw2)/*right sensor*/
    {
    case 0:
      /*no line*/
      comedi_data_write(device,1,0,1,aref,2048);usleep(10000);/*stop ready for line scanning*/
      comedi_data_write(device,1,1,1,aref,2048);usleep(10000);/*stop ready for line scanning*/
      for(i=0;i<*scanrange;i++)
        {
        /*scan one direction for line*/
          comedi_data_write(device,1,0,1,aref,20);usleep(10000);
          comedi_data_write(device,1,1,1,aref,4073);usleep(10000);
          *scantimer++;
        }
      break;
    case 8:
      /*left sensor off line*/
      comedi_data_write(device,1,0,1,aref,4073);usleep(10000);/*correctional movement*/
      comedi_data_write(device,1,1,1,aref,2048);usleep(10000);/*correctional movement*/
      break;
    default:
      break;
    }
      break;
    case 4:
      switch(irsw2)/*right sensor*/
    {
    case 0:
      /*right sensor off line*/
      comedi_data_write(device,1,0,1,aref,2048);usleep(10000);/*correctional movement*/
      comedi_data_write(device,1,1,1,aref,4078);usleep(10000);/*correctional movement*/
      break;
    case 8:
      /*both on line*/
      comedi_data_write(device,1,0,1,aref,4073);usleep(10000);/*carry on forward*/
      comedi_data_write(device,1,1,1,aref,4073);usleep(10000);/*carry on forward*/
      break;
    default:
      break;
    }
      break;
    default:
      break;
    }
  *scanrange=  *scanrange*2;

  switch(irsw1)/*left sensor*/
    {
    case 0:
      switch(irsw2)/*right sensor*/
    {
    case 0:
      /*no line*/
      comedi_data_write(device,1,0,1,aref,2048);usleep(10000);/*stop ready for line scanning*/
      comedi_data_write(device,1,1,1,aref,2048);usleep(10000);/*stop ready for line scanning*/
      for(i=0;i<*scanrange;i++)
        {
          /*scan other direction for line*/
          comedi_data_write(device,1,0,1,aref,4073);usleep(10000);
          comedi_data_write(device,1,1,1,aref,20);usleep(10000);
          *scantimer++;
        }
```

```
        break;
      case 8:
        /*left sensor off line*/
        comedi_data_write(device,1,0,1,aref,4073);usleep(10000);/*correctional movement*/
        comedi_data_write(device,1,1,1,aref,2048);usleep(10000);/*correctional movement*/
        break;
      default:
        break;
      }
        break;
      case 4:
        switch(irsw2)/*right sensor*/
      {
      case 0:
        /*right sensor off line*/
        comedi_data_write(device,1,0,1,aref,2048);usleep(10000);/*correctional movement*/
        comedi_data_write(device,1,1,1,aref,4078);usleep(10000);/*correctional movement*/
        break;
      case 8:
        /*both on line*/
        comedi_data_write(device,1,0,1,aref,4073);usleep(10000);/*carry on forward*/
        comedi_data_write(device,1,1,1,aref,4073);usleep(10000);/*carry on forward*/
        *scanrange =2;
        break;
      default:
        break;
      }
        break;
      default:
        break;
      }

    if(*scanrange>20)*scanrange =2;

    /* if((irsw1 == 4) & (irsw2 ==8))

      {
        comedi_data_write(device,1,0,1,aref,4073);
        comedi_data_write(device,1,1,1,aref,4073);
      }

    if((irsw1 == 0) & (irsw2 == 0))

      {
        comedi_data_write(device,1,0,1,aref,2048);
        comedi_data_write(device,1,1,1,aref,2048);
        }*/

    return(0);
  }

  /*****************************************************************************
  * Function name : BUMPERS (TYPE arg1, TYPE arg2…)
  * returns : return value description
  * arg1 : description
  * arg2 : description
  * Created by : Jon Ambrose
  * Date created : 25/03/08
  * Description : This function reads the left and right bumpers, and if either
                  one or the other and or both bumpers are pressed/triggered, then
                  motor values for the robot are set to stop - 2048.
  * Notes : This function is only called for AI for when the robot is following
           a white line and when it hits a obstacle/when bumpers are triggered,
           it gives control over to the joystick.
  *****************************************************************************/

  int bumpers(unsigned int digitaldevice, comedi_t *device, int read, int aref,
              int *hit )
  {
    unsigned int lbsw;
    unsigned int rbsw;

    /*READ LEFT Bumper -uSW1- D0*/

    comedi_dio_bitfield(device,digitaldevice,read,&lbsw);

    lbsw >>=8;
    lbsw =~lbsw;
    lbsw =lbsw & 0x01;


    /*READ RIGHT Bumper -uSW2- D1*/

    comedi_dio_bitfield(device,digitaldevice,read,&rbsw);

    rbsw >>=8;
    rbsw =~rbsw;
    rbsw =rbsw & 0x02;

    /*if right and or left bumpers are triggered then enter the if statement to
      display a error message and stop the motors.*/

    if((lbsw == 1)||(rbsw==2))

      {
        printw("You have hit an Obstacle\n");
        comedi_data_write(device,1,0,1,aref,2048); /* STOP Left Motors */
        comedi_data_write(device,1,1,1,aref,2048); /* STOP Right Motors */
        usleep(100000);
        return 1;
```

```
        *hit = 1;
      }

    return 0;
  }

  /*****************************************************************************
  * Function name : BUMPERS (TYPE arg1, TYPE arg2…)
  * returns : return value description
  * arg1 : description
  * arg2 : description
  * Created by : Jon Ambrose
  * Date created : 25/03/08
  * Description : This function
  * Notes : restrictions, odd modes
  *****************************************************************************/


  int lightfollow(unsigned int *digitaldevice, unsigned int *subdevice, comedi_t **device, int *read, int *aref,int *range,int *
  lighttimer )
  {
    unsigned int eyedata;
    unsigned int data;
    unsigned int data1;
    unsigned int data2;
    unsigned int data3;
    unsigned int smsw;
    unsigned int smswl;
    unsigned int smswr;
    int count = 0;
    int count1 = 0;
    int quad = 0;
    int movecnt=0;

    /*****************************************************************/
    comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
    smsw >>=8;
    smswr =smsw & 0x20;
    smswl = smsw & 0x10;
    data = 9;
    data1 = 10;
    data2 = 6;
    data3 = 5;

    while( smswr != 32)
      {
        comedi_dio_bitfield(*device,2,0xff,&data); usleep(10000);
        comedi_dio_bitfield(*device,2,0xff,&data1); usleep(10000);
        comedi_dio_bitfield(*device,2,0xff,&data2); usleep(10000);
        comedi_dio_bitfield(*device,2,0xff,&data3); usleep(10000);
        comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
        smsw >>=8;
        smswr =smsw & 0x20;
        smswl = smsw & 0x10;
      }
    while( smswl != 16)
      {
        comedi_dio_bitfield(*device,2,0xff,&data3); usleep(10000);
        comedi_dio_bitfield(*device,2,0xff,&data2); usleep(10000);
        comedi_dio_bitfield(*device,2,0xff,&data1); usleep(10000);
        comedi_dio_bitfield(*device,2,0xff,&data); usleep(10000);
        comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
        count++;
        smsw >>=8;
        smswr =smsw & 0x20;
        smswl = smsw & 0x10;

      }
     printw("there are %i sections\n", count);
    while( smswr != 32)

      {
        comedi_dio_bitfield(*device,2,0xff,&data); usleep(50000);
        comedi_dio_bitfield(*device,2,0xff,&data1); usleep(50000);
        comedi_dio_bitfield(*device,2,0xff,&data2); usleep(50000);
        comedi_dio_bitfield(*device,2,0xff,&data3); usleep(50000);
        comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
        count1++;
        smsw >>=8;
        smswr =smsw & 0x20;
        smswl = smsw & 0x10;

      }
    printw("there are %i sections\n", count1);
    if (count !=  count1)
      {
        printw("Counts are not equal \n");
        exit(1);
      }
   /*****************************************************************************/


        comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
        smsw >>=8;
        smswr =smsw & 0x20;
        smswl = smsw & 0x10;
        data = 9;
        data1 = 10;
        data2 = 6;
        data3 = 5;
```

```
        /*READ Right EYE -SMSW- D5*/ /*READ Left EYE -SMSW1- D4*/

        comedi_data_read(*device,*subdevice,EYE,*range,*aref,&eyedata);
        printw("EYE1 = %d\n ",eyedata);



        while( smswl != 16)
      {
          comedi_dio_bitfield(*device,2,0xff,&data3); usleep(20000);
        comedi_dio_bitfield(*device,2,0xff,&data2); usleep(20000);
        comedi_dio_bitfield(*device,2,0xff,&data1); usleep(20000);
        comedi_dio_bitfield(*device,2,0xff,&data); usleep(20000);

        comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
        quad++;
        smsw >>=8;
        smswr =smsw & 0x20;
        smswl = smsw & 0x10;
        /* printf( " l = %d r = %d\n", smswl, smswr);*/

        comedi_data_read(*device,*subdevice,EYE,*range,*aref,&eyedata);
        printw("there are %i quad\n", quad);
        printw("EYE16 = %d\n ",eyedata);


        if(eyedata > BRIGHTLIGHT)
          {

            if(quad >=  0 && quad <= 2.75)
          {
            comedi_data_write(*device,1,0,1,*aref,4073);
            comedi_data_write(*device,1,1,1,*aref,2048);
             usleep(1000);
          }
            else if(quad <= 5.5)
          {
            comedi_data_write(*device,1,0,1,*aref,4073);
            comedi_data_write(*device,1,1,1,*aref,4073);
            usleep(1000);
          }
            else if(quad <= 8.25)
          {
            comedi_data_write(*device,1,0,1,*aref,4073);
            comedi_data_write(*device,1,1,1,*aref,4073);
            usleep(1000);
          }

            else if(quad <= 11)
          {
            comedi_data_write(*device,1,0,1,*aref,2048);
            comedi_data_write(*device,1,1,1,*aref,4073);
             usleep(1000);
          }
          }
        movecnt++;
        if (movecnt == 5)
          {
            comedi_data_write(*device,1,0,1,*aref,2048);
            comedi_data_write(*device,1,1,1,*aref,2048);
            movecnt = 0;
          }

      }

        while( smswr != 32)

      {

        comedi_dio_bitfield(*device,2,0xff,&data); usleep(20000);
        comedi_dio_bitfield(*device,2,0xff,&data1); usleep(20000);
        comedi_dio_bitfield(*device,2,0xff,&data2); usleep(20000);
        comedi_dio_bitfield(*device,2,0xff,&data3); usleep(20000);
        comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
        quad--;
        smsw >>=8;
        smswr =smsw & 0x20;
        smswl = smsw & 0x10;

        comedi_data_read(*device,*subdevice,EYE,*range,*aref,&eyedata);
        printw("there are %i quad\n", quad);
        printw("EYE32 = %d\n ",eyedata);



        /* read the light sensor */
        comedi_data_read(*device,*subdevice,EYE,*range,*aref,&eyedata);
        printw("EYE2 = %d\n ",eyedata);




        if(eyedata > BRIGHTLIGHT)
          {

            if(quad >=  0 && quad <= 2.75)
          {
            comedi_data_write(*device,1,0,1,*aref,4073);
            comedi_data_write(*device,1,1,1,*aref,2048);
            usleep(1000);
```

```
1000            }
1001              else if(quad <= 5.5)
1002            {
1003              comedi_data_write(*device,1,0,1,*aref,4073);
1004              comedi_data_write(*device,1,1,1,*aref,4073);
1005              usleep(1000);
1006            }
1007              else if(quad <= 8.25)
1008            {
1009              comedi_data_write(*device,1,0,1,*aref,4073);
1010              comedi_data_write(*device,1,1,1,*aref,4073);
1011              usleep(1000);
1012            }
1013
1014              else if(quad <= 11)
1015            {
1016              comedi_data_write(*device,1,0,1,*aref,2048);
1017              comedi_data_write(*device,1,1,1,*aref,4073);
1018              usleep(1000);
1019            }
1020            }
1021          movecnt++;
1022          if (movecnt == 5)
1023            {
1024              comedi_data_write(*device,1,0,1,*aref,2048);
1025              comedi_data_write(*device,1,1,1,*aref,2048);
1026              movecnt = 0;
1027            }
1028        }
1029          lighttimer++;
1030          return 0;
1031    }
1032
1033    void joy_cali (comedi_t *device,unsigned subdevice, int range, int aref,int horizontal,int vertical,int *hori,int *vert,int *
      maxhori,int *maxvert)
1034    {
1035
1036      int horidata=0,vertdata=0;
1037      int minhori=0, minvert=0;
1038      int ret=0 ;
1039
1040
1041      printw("MOVE STICK TO TOP RIGHT AND HOLD\n");
1042      printw("PRESS ENTER WHEN READY!\n");
1043      refresh();
1044      getchar();
1045      comedi_data_read(device,subdevice,horizontal,range,aref,&horidata);
1046      comedi_data_read(device,subdevice,vertical,range,aref,&vertdata);
1047
1048      *maxvert = vertdata;
1049      *maxhori = horidata;
1050
1051      clear();
1052      printw("DONE,RELEASE AND ALLOW TO RETURN TO CENTER!");
1053      refresh();
1054      getchar();
1055
1056      clear();
1057      printw("MOVE STICK TO BOTTOM LEFT AND HOLD\n");
1058      printw("PRESS ENTER WHEN READY!\n");
1059      refresh();
1060      getchar();
1061     ret = comedi_data_read(device,subdevice,horizontal,range,aref,&horidata);
1062
1063     comedi_data_read(device,0,horizontal,range,aref,&horidata);
1064     comedi_data_read(device,subdevice,vertical,range,aref,&vertdata);
1065      minvert = vertdata;
1066      minhori = horidata;
1067      /*clear();*/
1068      printw("DONE,RELEASE AND ALLOW TO RETURN TO CENTER!\n");
1069      refresh();
1070      getchar();
1071      clear();
1072      printw("Mv-%d Mh-%d mv-%d mh-%d\n",*maxvert,*maxhori,minvert,minhori);
1073
1074
1075      *vert = *maxvert-minvert;
1076      *hori = *maxhori-minhori;
1077
1078      printw("vert %d\n", *vert);
1079      printw("hori %d\n", *hori);
1080      printw("\n-------------------------------\n");
1081      refresh();
1082    }
1083
1084
1085    void driving(comedi_t *device,unsigned int subdevice, int horizontal, int vertical,int range,int aref,int *maxvert,int *maxhori,
       char *filename,int *motrrgt, int *motrlft, int vert, int hori)
1086    {
1087      int ret;
1088      float vertdiv =0;
1089      float horidiv =0;
1090      int vertarray;
1091      int horiarray;
1092      unsigned int horidata;
1093      unsigned int vertdata;
1094      int array [6][3] =
1095        {
1096          {2048 , 4073 , 4073}, /*{r-lfw, r-fw    , r-rfw},*/
1097          {20   , 2048 , 4073}, /*{r-l  , r-center, r-rb },*/
```

```
        {2048 ,   20 ,   20}, /*{r-lbw, r-bw    , r-rbw},*/
        {4073 , 4073 , 2048}, /*{l-lfw, l-fw    , l-rfw},*/
        {4073 , 2048 ,   20}, /*{l-l  , l-center, l-r  },*/
        {20   ,   20 , 2048}, /*{l-lbw, l-bw    , l-rbw},*/
      };


 ret = comedi_data_read(device,subdevice,horizontal,range,aref,&horidata);
       if (ret<0)
      {
        printf("i have an error here\n");
        comedi_perror(filename);
        exit(0);
      }

        usleep(50000);

        ret = comedi_data_read(device,subdevice,vertical,range,aref,&vertdata);
        if (ret<0)
      {
        printf("i have an error here\n");
        comedi_perror(filename);
        exit(0);
      }


        printw("V = %d          ",vertdata);
        printw("H = %d          ",horidata);

        vertdiv = *maxvert-vertdata;
        horidiv = *maxhori-horidata;



        usleep(50000);

        printw("Vc = %2.2f      ",vertdiv);
        printw("Hc = %2.2f\n",horidiv);

        if (vertdiv>10000)
      {
        vertdata = 0;
      }

        if (horidiv>10000)
      {
        horidata=0;
      }


        /***************NEW MOTOR SHIT*******************/


        vertarray = vert/3;
        horiarray = hori/3;

        vertarray = vertdiv/vertarray;
        horiarray = horidiv/horiarray;
        if( vertarray > 2) vertarray = 2;
        if( horiarray > 2) horiarray = 2;

        *motrrgt = array[vertarray][horiarray];
        *motrlft = array[vertarray+3][horiarray];

        printw("\nrt = %d      ",motrrgt);
        printw("lft = %d\n",motrlft);
        refresh();
        /* motrlft = 2048;
       motrrgt = 2048;*/
        comedi_data_write(device,1,0,1,aref,*motrlft);
        comedi_data_write(device,1,1,1,aref,*motrrgt);

}
```