

```

1  /*****
2                                     Marco Assignment 2
3  Author: Jonathan Ambrose & Andrew Fester
4  Date: 17/04/08 17:44
5  Version: Final Release
6  Copyright: JAAF Software
7  Description: A program
8
9
10 Compile: gcc -pedantic -Wall -lcomedi -lncurses -lm irsw.c -o ir
11 Run Code: ./jamlight
12 *****/
13
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include <fcntl.h>
17 #include <unistd.h>
18 #include <errno.h>
19 #include <getopt.h>
20 #include <ctype.h>
21 #include <comedi.h>
22 #include <comedilib.h>
23 #include <unistd.h>
24 #include <math.h>
25 #include <curses.h>
26 #include <ctype.h>
27
28 /***** # defines*****/
29
30 #define EYESOCKET 0x80
31 #define EYE 3
32 #define WRITEFILE "w"
33 #define READFILE "r"
34 #define LOGFILE "./path.txt"
35 #define BRIGHTLIGHT 2150
36
37 /****Menu #defines****/
38
39 #define MENU 0
40 #define REPLAY 1
41 #define LINEFOLLOW 2
42 #define LIGHTFOLLOW 3
43 #define NORMAL 4
44 #define ARTIFICIAL 5
45
46 /****Replay menu****/
47
48 #define REPLAYMENU 0
49 #define RECORD 1
50 #define SRECORD 2
51 #define REPLAYING 3
52 #define ERASE 4
53 #define HPLAY 5
54 #define HREPLAY 6
55
56 /****AI States*****/
57
58 #define LINE 0
59 #define RANDOM 1
60 #define LIGHT 2
61 #define CONTROLLER 3
62
63 /*****Prototypes*****/
64
65 void welcome_menu (void);
66 int init(comedi_t **device,unsigned int *digitaldevice,unsigned int *subdevice);
67 int wlfollow(unsigned int digitaldevice, comedi_t *device, int read, int aref,
68             int *scanrange,int *scantimer);
69 int lightfollow(unsigned int *digitaldevice, unsigned int *subdevice,
70               comedi_t **device, int *read, int *aref,int *range,int *lighttimer);
71 int bumpers(unsigned int digitaldevice, comedi_t *device, int read,
72            int aref, int *hit);
73 void joy_call (comedi_t *device,unsigned subdevice, int range, int aref,
74              int horizontal,int vertical,int *hori,int *vert,
75              int *maxhori, int *maxvert);
76 void driving(comedi_t *device, unsigned int subdevice, int horizontal,
77             int vertical,int range, int aref, int *maxvert, int *maxhori,
78             char *filename,int *motrrgt,int *motrlft,int vert, int hori);
79 /*****
80
81 int main(void)
82 {
83     int i;
84     char *filename = "/dev/comedi0";
85     comedi_t *device;
86     unsigned int subdevice = 0;
87     unsigned int digitaldevice = 2;
88
89     int maxhori;
90     int maxvert;
91     float horidiv=0, vertdiv=0;
92     int range = 0;
93     int ret;
94     int horizontal= 1;
95     int vertical= 0;
96     int aref = AREF_GROUND;
97     /*IR setup*/
98
99     int read = 0x00;
100    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 wanderl=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 printf("%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 printf("-----MENU-----\n");
152 printf("What do you want to do?\n");
153 printf("R - Replay menu\n");
154 printf("L - Line follow\n");
155 printf("T - Light follow\n");
156 printf("B - Bumper\n");
157 printf("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         printf("-----MENU-----\n");
171         printf("What do you want to do?\n");
172         printf("R - Replay menu\n");
173         printf("L - Line follow\n");
174         printf("T - Light follow\n");
175         printf("N - Controller\n");
176         printf("A - ai\n");
177         printf("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         break;
188
189     case REPLAY:
190
191         switch(submenu)
192         {
193         case REPLAYMENU:
194             printf("-----REPLAY-----\n");
195             printf("What do you want to do?\n");
196             printf("R - Replay menu\n");
197             printf("\t 1 - Record\n");
198             printf("\t 2 - Stop recording\n");
199             printf("\t 3 - Replay\n");
200

```

```

201    printw("\t 4 - Erase\n");
202    printw("L - Line follow\n");
203    printw("T - Light follow\n");
204    printw("N - Controller\n");
205    printw("A - ai\n");
206    printw("M - Menu\n");
207    refresh();
208
209     submenu = toupper(getchar());
210     if(submenu == 'R')state = REPLAY;
211     else if(submenu == 'L'){ state = LINEFOLLOW;submenu = 0;}
212     else if(submenu == 'T'){ state = LIGHTFOLLOW;submenu = 0;}
213     else if(submenu == 'N'){ state = NORMAL;submenu = 0;}
214     else if(submenu == 'M'){ state = MENU;submenu = 0;}
215     else if(submenu == '0')submenu = REPLAYMENU;
216     else if(submenu == '1')submenu = RECORD;
217     else if(submenu == 'S')submenu = SRECORD;
218     else if(submenu == '3')submenu = REPLAYING;
219     else if(submenu == '4')submenu = ERASE;
220     else if(submenu == '5')submenu = HPLAY;
221     else if(submenu == '6')submenu = HREPLAY;
222
223     break;
224 case RECORD:
225
226     log = fopen(LOGFILE , WRITEFILE);
227     submenu = 5;
228
229
230
231     break;
232 case 'S':
233     submenu = REPLAYMENU;
234     fclose(log);
235     break;
236 case REPLAYING:
237
238     log = fopen(LOGFILE , READFILE);
239     error = 0;
240     submenu = HREPLAY;
241
242     break;
243 case ERASE:
244     log = fopen(LOGFILE , WRITEFILE);
245     fclose(log);
246     submenu = REPLAYMENU;
247
248     break;
249 case HPLAY:
250     driving(device,subdevice,horizontal,vertical,range,aref,&maxxori,
251             &maxvert,filename, &motrrgt,&motrlft,vert,hor);
252
253     fprintf(log,"%d %d\n",motrrgt , motrlft);
254     printw("-----RECORDING-----\n");
255     printw("What do you want to do?\n");
256     printw("R - Replay menu\n");
257     printw("\t 1 - Record\n");
258     printw("\t 2 - Stop recording\n");
259     printw("\t 3 - Replay\n");
260     printw("\t 4 - Erase\n");
261     printw("L - Line follow\n");
262     printw("T - Light follow\n");
263     printw("N - Controller\n");
264     printw("A - ai\n");
265     printw("M - Menu\n");
266     refresh();
267
268     submenu = toupper(getch());
269
270     if(submenu == 'R')state = REPLAY;
271     else if(submenu == 'L'){ state = LINEFOLLOW;submenu = 0;}
272     else if(submenu == 'T'){ state = LIGHTFOLLOW;submenu = 0;}
273     else if(submenu == 'N'){ state = NORMAL;submenu = 0;}
274     else if(submenu == 'M'){ state = MENU;submenu = 0;}
275     else if(submenu == '0')submenu = REPLAYMENU;
276     else if(submenu == '1')submenu = RECORD;
277     else if(submenu == 'S')submenu = SRECORD;
278     else if(submenu == '3')submenu = REPLAYING;
279     else if(submenu == '4')submenu = ERASE;
280     else if(submenu == '5')submenu = HPLAY;
281     else if(submenu == '6')submenu = HREPLAY;
282     else submenu = HPLAY;
283
284     break;
285 case HREPLAY:
286     printw("-----REPLAYING-----\n");
287     printw("What do you want to do?\n");
288     printw("R - Replay menu\n");
289     printw("\t 1 - Record\n");
290     printw("\t 2 - Stop recording\n");
291     printw("\t 3 - Replay\n");
292     printw("\t 4 - Erase\n");
293     printw("L - Line follow\n");
294     printw("T - Light follow\n");
295     printw("N - Controller\n");
296     printw("A - ai\n");
297     printw("M - Menu\n");
298     refresh();
299
300     submenu = toupper(getch());

```

```

301     if(submenu == 'R')state = REPLAY;
302     else if(submenu == 'L'){ state = LINEFOLLOW;submenu = 0;}
303     else if(submenu == 'T'){ state = LIGHTFOLLOW;submenu = 0;}
304     else if(submenu == 'N'){ state = NORMAL;submenu = 0;}
305     else if(submenu == 'M'){ state = MENU;submenu = 0;}
306     else if(submenu == '0')submenu = REPLAYMENU;
307     else if(submenu == '1')submenu = RECORD;
308     else if(submenu == '2')submenu = SRECORD;
309     else if(submenu == '3')submenu = REPLAYING;
310     else if(submenu == '4')submenu = ERASE;
311     else if(submenu == '5')submenu = HPLAY;
312     else if(submenu == '6')submenu = HREPLAY;
313     else submenu = HREPLAY;
314
315
316     fscanf(log,"%d %d",&motrrgt , &motrlft);
317     t++;
318     usleep(100000);
319     printf("%d %d",motrrgt,motrlft);
320     comedi_data_write(device,1,0,1,aref,motrlft);
321     comedi_data_write(device,1,1,1,aref,motrrgt);
322     printf("%d",t);
323     refresh();
324
325
326     /* submenu = REPLAYMENU;*/
327
328     break;
329 case 'M':
330     state = MENU;
331     break;
332 default:
333     printf("-----REPLAY-----\n");
334     printf("What do you want to do?\n");
335     printf("R - Replay menu\n");
336     printf("\t 1 - Record\n");
337     printf("\t 2 - Stop recording\n");
338     printf("\t 3 - Replay\n");
339     printf("\t 4 - Erase\n");
340     printf("L - Line follow\n");
341     printf("T - Light follow\n");
342     printf("N - Controller\n");
343     printf("A - ai\n");
344     printf("M - Menu\n");
345     refresh();
346
347     submenu = toupper(getchar());
348     if(submenu == 'R')state = REPLAY;
349     else if(submenu == 'L'){ state = LINEFOLLOW;submenu = 0;}
350     else if(submenu == 'T'){ state = LIGHTFOLLOW;submenu = 0;}
351     else if(submenu == 'N'){ state = NORMAL;submenu = 0;}
352     else if(submenu == 'M'){ state = MENU;submenu = 0;}
353     else if(submenu == '0')submenu = REPLAYMENU;
354     else if(submenu == '1')submenu = RECORD;
355     else if(submenu == '2')submenu = SRECORD;
356     else if(submenu == '3')submenu = REPLAYING;
357     else if(submenu == '4')submenu = ERASE;
358     else if(submenu == '5')submenu = HPLAY;
359     else if(submenu == '6')submenu = HREPLAY;
360     break;
361 }
362 break;
363 case LINEFOLLOW:
364     printf("-----WHITE-LINE-FOLLOW-----\n");
365     printf("What do you want to do?\n");
366     printf("R - Replay menu\n");
367     printf("L - Line follow\n");
368     printf("T - Light follow\n");
369     printf("N - controller\n");
370     printf("A - ai\n");
371     printf("M - Menu\n");
372     refresh();
373     state = toupper(getch());
374     if(state == 'R')state = REPLAY;
375     else if(state == 'L') state = LINEFOLLOW;
376     else if(state == 'T') state = LIGHTFOLLOW;
377     else if(state == 'N') state = NORMAL;
378     else if(state == 'M') state = MENU;
379     else state = LINEFOLLOW;
380
381
382     printf("%d",t);
383     refresh();
384
385
386     wlfollow(digitaldevice,device, read, aref,&scanrange,&scantimer);
387     bumpers(digitaldevice, device, read, aref,&hit);
388     refresh();
389
390     break;
391
392 case LIGHTFOLLOW:
393     printf("-----LIGHT-FOLLOW-----\n");
394     printf("What do you want to do?\n");
395     printf("R - Replay menu\n");
396     printf("L - Line follow\n");
397     printf("T - Light follow\n");
398     printf("N - controller\n");
399     printf("A - ai\n");
400     printf("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     printf("%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     printf("-----CONTROLLER-----\n");
419     printf("What do you want to do?\n");
420     printf("R - Replay menu\n");
421     printf("L - Line follow\n");
422     printf("T - Light follow\n");
423     printf("N - controller\n");
424     printf("A - ai\n");
425     printf("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=CONTROLLER;
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=CONTROLLER;
481
482         }
483         randomtimer++;
484         ai = LIGHT;
485
486         break;
487     case LIGHT:
488         /*after moving 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         }

```

```

501     if(randomtimer == 2)/*stop random movementand go back to line */
502     {
503         ai=LINE;
504         randomtimer = 0;
505     }
506     bumpers(digitaldevice, device, read, aref,&hit );
507     if (hit == 1)ai=CONTROLLER;
508
509     break;
510     case CONTROLLER:
511         /*user control with access tot ehmenu for other functionality*/
512         driving(device,subdevice,horizontal,vertical,range,aref,&maxhori,
513             &maxvert,filename, &motrrgt,&motrlft,vert,hori);
514
515         printf("-----MENU-----\n");
516         printf("What do you want to do?\n");
517         printf("R - Replay menu\n");
518         printf("L - Line follow\n");
519         printf("T - Light follow\n");
520         printf("N - Controller\n");
521         printf("A - ai\n");
522         printf("M - Menu\n");
523         refresh();
524         state = toupper(getch());
525         if(state == 'R')state = REPLAY;
526         else if(state == 'L') state = LINEFOLLOW;
527         else if(state == 'T') state = LIGHTFOLLOW;
528         else if(state == 'N') state = NORMAL;
529         else if(state == 'M') state = MENU;
530
531         break;
532         default:
533         break;
534     }
535
536     break;
537 default:
538     printf("-----MENU-----\n");
539     printf("What do you want to do?\n");
540     printf("R - Replay menu\n");
541     printf("L - Line follow\n");
542     printf("T - Light follow\n");
543     printf("N - Controller\n");
544     printf("M - Menu\n");
545     refresh();
546     state = toupper(getchar());
547     if(state == 'R')state = REPLAY;
548     else if(state == 'L') state = LINEFOLLOW;
549     else if(state == 'T') state = LIGHTFOLLOW;
550     else if(state == 'N') state = NORMAL;
551     else if(state == 'M') state = MENU;
552     break;
553 }
554
555     refresh();
556 }
557
558
559     return 0;
560 }
561
562 /*****FUNCTIONS*****/
563
564 /*initialize the rack*/
565
566 int init(comedi_t **device, unsigned int *digitaldevice, unsigned int *subdevice)
567 {
568     int i;
569     int retval;
570     int stype;
571     char *filename = "/dev/comedi0";
572
573     *device=comedi_open(filename);
574
575     if(!*device)
576     {
577         comedi_perror(filename);
578         exit(0);
579     }
580
581     for(i=0;i<8;i++)
582
583         retval=comedi_dio_config(*device,*digitaldevice,i,COMEDI_OUTPUT);
584
585     /* check we have a valid bidirection DIO port subdevice*/
586
587     stype = comedi_get_subdevice_type(*device,*digitaldevice);
588     if(stype!=COMEDI_SUBD_DIO)
589     {
590         printf("%d is not a digital I/O subdevice\n",*subdevice);
591         exit(1);
592     }
593
594     return 0 ;
595 }
596
597
598 }
599
600 /*****

```

```

601 * Function name : WHITE LINE FOLLOW (TYPE arg1, TYPE arg2...)
602 * returns : return value description
603 * arg1 : description
604 * arg2 : description
605 * Created by : Jon Ambrose & Andrew Fester
606 * Date created : date
607 * Description : detailed description
608 * Notes : restrictions, odd modes
609 *****/
610 int wlfollow(unsigned int digitaldevice, comedi_t *device, int read, int aref,
611             int *scanrange, int *scantimer )
612 {
613     unsigned int irsw1;
614     unsigned int irsw2;
615     int i;
616
617
618     /*READ IRSW1 - D2*/
619     irsw1=0; /*set irsw1 to 0*/
620
621     comedi_dio_bitfield(device,digitaldevice,read,&irsw1);
622
623     irsw1 >>=8;
624     irsw1 =~irsw1;
625     irsw1 =irsw1 & 0x04;
626
627     /*READ IRSW2 - D3*/
628     irsw2=0;
629
630     comedi_dio_bitfield(device,digitaldevice,read,&irsw2);
631
632     irsw2 >>=8;
633     irsw2 =~irsw2;
634     irsw2 =irsw2 & 0x08;
635
636
637     switch(irsw1)/*left sensor*/
638     {
639     case 0:
640         switch(irsw2)/*right sensor*/
641         {
642         case 0:
643             /*no line*/
644             comedi_data_write(device,1,0,1,aref,2048);usleep(10000);/*stop ready for line scanning*/
645             comedi_data_write(device,1,1,1,aref,2048);usleep(10000);/*stop ready for line scanning*/
646             for(i=0;i<*scanrange;i++)
647             {
648                 /*scan one direction for line*/
649                 comedi_data_write(device,1,0,1,aref,20);usleep(10000);
650                 comedi_data_write(device,1,1,1,aref,4073);usleep(10000);
651                 *scantimer++;
652             }
653             break;
654         case 8:
655             /*left sensor off line*/
656             comedi_data_write(device,1,0,1,aref,4073);usleep(10000);/*correctional movement*/
657             comedi_data_write(device,1,1,1,aref,2048);usleep(10000);/*correctional movement*/
658             break;
659         default:
660             break;
661         }
662         break;
663     case 4:
664         switch(irsw2)/*right sensor*/
665         {
666         case 0:
667             /*right sensor off line*/
668             comedi_data_write(device,1,0,1,aref,2048);usleep(10000);/*correctional movement*/
669             comedi_data_write(device,1,1,1,aref,4078);usleep(10000);/*correctional movement*/
670             break;
671         case 8:
672             /*both on line*/
673             comedi_data_write(device,1,0,1,aref,4073);usleep(10000);/*carry on forward*/
674             comedi_data_write(device,1,1,1,aref,4073);usleep(10000);/*carry on forward*/
675             break;
676         default:
677             break;
678         }
679         break;
680     default:
681         break;
682     }
683     *scanrange= *scanrange*2;
684
685     switch(irsw1)/*left sensor*/
686     {
687     case 0:
688         switch(irsw2)/*right sensor*/
689         {
690         case 0:
691             /*no line*/
692             comedi_data_write(device,1,0,1,aref,2048);usleep(10000);/*stop ready for line scanning*/
693             comedi_data_write(device,1,1,1,aref,2048);usleep(10000);/*stop ready for line scanning*/
694             for(i=0;i<*scanrange;i++)
695             {
696                 /*scan other direction for line*/
697                 comedi_data_write(device,1,0,1,aref,4073);usleep(10000);
698                 comedi_data_write(device,1,1,1,aref,20);usleep(10000);
699                 *scantimer++;
700             }

```

```

701     break;
702 case 8:
703     /*left sensor off line*/
704     comedi_data_write(device,1,0,1,aref,4073);usleep(10000);/*correctional movement*/
705     comedi_data_write(device,1,1,1,aref,2048);usleep(10000);/*correctional movement*/
706     break;
707 default:
708     break;
709 }
710 break;
711 case 4:
712     switch(irsw2)/*right sensor*/
713     {
714     case 0:
715         /*right sensor off line*/
716         comedi_data_write(device,1,0,1,aref,2048);usleep(10000);/*correctional movement*/
717         comedi_data_write(device,1,1,1,aref,4078);usleep(10000);/*correctional movement*/
718         break;
719     case 8:
720         /*both on line*/
721         comedi_data_write(device,1,0,1,aref,4073);usleep(10000);/*carry on forward*/
722         comedi_data_write(device,1,1,1,aref,4073);usleep(10000);/*carry on forward*/
723         *scanrange =2;
724         break;
725     default:
726         break;
727     }
728     break;
729 default:
730     break;
731 }
732
733 if(*scanrange>20)*scanrange =2;
734
735 /* if((irsw1 == 4) & (irsw2 ==8))
736
737 {
738     comedi_data_write(device,1,0,1,aref,4073);
739     comedi_data_write(device,1,1,1,aref,4073);
740 }
741
742 if((irsw1 == 0) & (irsw2 == 0))
743
744 {
745     comedi_data_write(device,1,0,1,aref,2048);
746     comedi_data_write(device,1,1,1,aref,2048);
747 }*/
748
749 return(0);
750 }
751
752 /*****
753 * Function name : BUMPERS (TYPE arg1, TYPE arg2...)
754 * returns : return value description
755 * arg1 : description
756 * arg2 : description
757 * Created by : Jon Ambrose
758 * Date created : 25/03/08
759 * Description : This function reads the left and right bumpers, and if either
760                one or the other and or both bumpers are pressed/triggered, then
761                motor values for the robot are set to stop - 2048.
762 * Notes : This function is only called for AI for when the robot is following
763          a white line and when it hits a obstacle/when bumpers are triggered,
764          it gives control over to the joystick.
765 *****/
766
767 int bumpers(unsigned int digitaldevice, comedi_t *device, int read, int aref,
768            int *hit )
769 {
770     unsigned int lbsp;
771     unsigned int rbsp;
772
773     /*READ LEFT Bumper -uSW1- D0*/
774
775     comedi_dio_bitfield(device,digitaldevice,read,&lbsp);
776
777     lbsp >>=8;
778     lbsp =~lbsp;
779     lbsp =lbsp & 0x01;
780
781
782     /*READ RIGHT Bumper -uSW2- D1*/
783
784     comedi_dio_bitfield(device,digitaldevice,read,&rbsp);
785
786     rbsp >>=8;
787     rbsp =~rbsp;
788     rbsp =rbsp & 0x02;
789
790     /*if right and or left bumpers are triggered then enter the if statement to
791       display a error message and stop the motors.*/
792
793     if((lbsp == 1)|| (rbsp==2))
794
795     {
796         printf("You have hit an Obstacle\n");
797         comedi_data_write(device,1,0,1,aref,2048); /* STOP Left Motors */
798         comedi_data_write(device,1,1,1,aref,2048); /* STOP Right Motors */
799         usleep(100000);
800         return 1;

```



```

801     *hit = 1;
802     }
803
804     return 0;
805 }
806
807 /*****
808 * Function name : BUMPERS (TYPE arg1, TYPE arg2...)
809 * returns : return value description
810 * arg1 : description
811 * arg2 : description
812 * Created by : Jon Ambrose
813 * Date created : 25/03/08
814 * Description : This function
815 * Notes : restrictions, odd modes
816 *****/
817
818
819 int lightfollow(unsigned int *digitaldevice, unsigned int *subdevice, comedi_t **device, int *read, int *aref, int *range, int *
lighttimer )
820 {
821     unsigned int eyedata;
822     unsigned int data;
823     unsigned int data1;
824     unsigned int data2;
825     unsigned int data3;
826     unsigned int smsw;
827     unsigned int smswl;
828     unsigned int smswr;
829     int count = 0;
830     int count1 = 0;
831     int quad = 0;
832     int movecnt=0;
833
834     /*****
835     comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
836     smsw >>=8;
837     smswr =smsw & 0x20;
838     smswl = smsw & 0x10;
839     data = 9;
840     data1 = 10;
841     data2 = 6;
842     data3 = 5;
843
844     while( smswr != 32)
845     {
846         comedi_dio_bitfield(*device,2,0xff,&data); usleep(10000);
847         comedi_dio_bitfield(*device,2,0xff,&data1); usleep(10000);
848         comedi_dio_bitfield(*device,2,0xff,&data2); usleep(10000);
849         comedi_dio_bitfield(*device,2,0xff,&data3); usleep(10000);
850         comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
851         smsw >>=8;
852         smswr =smsw & 0x20;
853         smswl = smsw & 0x10;
854     }
855     while( smswl != 16)
856     {
857         comedi_dio_bitfield(*device,2,0xff,&data3); usleep(10000);
858         comedi_dio_bitfield(*device,2,0xff,&data2); usleep(10000);
859         comedi_dio_bitfield(*device,2,0xff,&data1); usleep(10000);
860         comedi_dio_bitfield(*device,2,0xff,&data); usleep(10000);
861         comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
862         count++;
863         smsw >>=8;
864         smswr =smsw & 0x20;
865         smswl = smsw & 0x10;
866
867     }
868     printf("there are %i sections\n", count);
869     while( smswr != 32)
870
871     {
872         comedi_dio_bitfield(*device,2,0xff,&data); usleep(50000);
873         comedi_dio_bitfield(*device,2,0xff,&data1); usleep(50000);
874         comedi_dio_bitfield(*device,2,0xff,&data2); usleep(50000);
875         comedi_dio_bitfield(*device,2,0xff,&data3); usleep(50000);
876         comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
877         count1++;
878         smsw >>=8;
879         smswr =smsw & 0x20;
880         smswl = smsw & 0x10;
881
882     }
883     printf("there are %i sections\n", count1);
884     if (count != count1)
885     {
886         printf("Counts are not equal \n");
887         exit(1);
888     }
889     /*****
890
891     comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
892     smsw >>=8;
893     smswr =smsw & 0x20;
894     smswl = smsw & 0x10;
895     data = 9;
896     data1 = 10;
897     data2 = 6;
898     data3 = 5;

```

```

900 /*READ Right EYE -SMSW- D5*/ /*READ Left EYE -SMSW1- D4*/
901
902 comedi_data_read(*device,*subdevice,EYE,*range,*aref,&eyedata);
903 printf("EYE1 = %d\n",eyedata);
904
905
906
907 while( smswl != 16)
908 {
909     comedi_dio_bitfield(*device,2,0xff,&data3); usleep(20000);
910     comedi_dio_bitfield(*device,2,0xff,&data2); usleep(20000);
911     comedi_dio_bitfield(*device,2,0xff,&data1); usleep(20000);
912     comedi_dio_bitfield(*device,2,0xff,&data); usleep(20000);
913
914     comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
915     quad++;
916     smsw >>=8;
917     smswr = smsw & 0x20;
918     smswl = smsw & 0x10;
919     /* printf( " l = %d r = %d\n", smswl, smswr);*/
920
921     comedi_data_read(*device,*subdevice,EYE,*range,*aref,&eyedata);
922     printf("there are %i quad\n", quad);
923     printf("EYE16 = %d\n",eyedata);
924
925
926     if(eyedata > BRIGHTLIGHT)
927     {
928
929         if(quad >= 0 && quad <= 2.75)
930         {
931             comedi_data_write(*device,1,0,1,*aref,4073);
932             comedi_data_write(*device,1,1,1,*aref,2048);
933             usleep(1000);
934         }
935         else if(quad <= 5.5)
936         {
937             comedi_data_write(*device,1,0,1,*aref,4073);
938             comedi_data_write(*device,1,1,1,*aref,4073);
939             usleep(1000);
940         }
941         else if(quad <= 8.25)
942         {
943             comedi_data_write(*device,1,0,1,*aref,4073);
944             comedi_data_write(*device,1,1,1,*aref,4073);
945             usleep(1000);
946         }
947
948         else if(quad <= 11)
949         {
950             comedi_data_write(*device,1,0,1,*aref,2048);
951             comedi_data_write(*device,1,1,1,*aref,4073);
952             usleep(1000);
953         }
954     }
955     movecnt++;
956     if (movecnt == 5)
957     {
958         comedi_data_write(*device,1,0,1,*aref,2048);
959         comedi_data_write(*device,1,1,1,*aref,2048);
960         movecnt = 0;
961     }
962 }
963
964
965 while( smswr != 32)
966 {
967
968     comedi_dio_bitfield(*device,2,0xff,&data); usleep(20000);
969     comedi_dio_bitfield(*device,2,0xff,&data1); usleep(20000);
970     comedi_dio_bitfield(*device,2,0xff,&data2); usleep(20000);
971     comedi_dio_bitfield(*device,2,0xff,&data3); usleep(20000);
972     comedi_dio_bitfield(*device,*digitaldevice,*read,&smsw);
973     quad--;
974     smsw >>=8;
975     smswr = smsw & 0x20;
976     smswl = smsw & 0x10;
977
978
979     comedi_data_read(*device,*subdevice,EYE,*range,*aref,&eyedata);
980     printf("there are %i quad\n", quad);
981     printf("EYE32 = %d\n",eyedata);
982
983
984
985     /* read the light sensor */
986     comedi_data_read(*device,*subdevice,EYE,*range,*aref,&eyedata);
987     printf("EYE2 = %d\n",eyedata);
988
989
990
991
992     if(eyedata > BRIGHTLIGHT)
993     {
994
995         if(quad >= 0 && quad <= 2.75)
996         {
997             comedi_data_write(*device,1,0,1,*aref,4073);
998             comedi_data_write(*device,1,1,1,*aref,2048);
999             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_cal (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     printf("MOVE STICK TO TOP RIGHT AND HOLD\n");
1042     printf("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     printf("DONE, RELEASE AND ALLOW TO RETURN TO CENTER!");
1053     refresh();
1054     getchar();
1055
1056     clear();
1057     printf("MOVE STICK TO BOTTOM LEFT AND HOLD\n");
1058     printf("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     printf("DONE, RELEASE AND ALLOW TO RETURN TO CENTER!\n");
1069     refresh();
1070     getchar();
1071     clear();
1072     printf("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     printf("vert %d\n", *vert);
1079     printf("hori %d\n", *hori);
1080     printf("\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 }*/

```

```

1098     {2048 , 20 , 20}, /*{r-lbw, r-bw , r-rbw},*/
1099     {4073 , 4073 , 2048}, /*{l-lfw, l-fw , l-rfw},*/
1100     {4073 , 2048 , 20}, /*{l-l , l-center, l-r },*/
1101     {20 , 20 , 2048}, /*{l-lbw, l-bw , l-rbw},*/
1102 };
1103
1104
1105 ret = comedi_data_read(device,subdevice,horizontal,range,aref,&horidata);
1106 if (ret<0)
1107 {
1108     printf("i have an error here\n");
1109     comedi_perror(filename);
1110     exit(0);
1111 }
1112
1113     usleep(50000);
1114
1115 ret = comedi_data_read(device,subdevice,vertical,range,aref,&vertdata);
1116 if (ret<0)
1117 {
1118     printf("i have an error here\n");
1119     comedi_perror(filename);
1120     exit(0);
1121 }
1122
1123
1124     printf("V = %d          ",vertdata);
1125     printf("H = %d          ",horidata);
1126
1127     vertdiv = *maxvert-vertdata;
1128     horidiv = *maxhori-horidata;
1129
1130
1131
1132     usleep(50000);
1133
1134     printf("Vc = %2.2f          ",vertdiv);
1135     printf("Hc = %2.2f\n",horidiv);
1136
1137     if (vertdiv>10000)
1138     {
1139         vertdata = 0;
1140     }
1141
1142     if (horidiv>10000)
1143     {
1144         horidata=0;
1145     }
1146
1147     /******NEW MOTOR SHIT******/
1148
1149
1150
1151     vertarray = vert/3;
1152     horiarray = hori/3;
1153
1154     vertarray = vertdiv/vertarray;
1155     horiarray = horidiv/horiarray;
1156     if( vertarray > 2) vertarray = 2;
1157     if( horiarray > 2) horiarray = 2;
1158
1159     *motrrgt = array[vertarray][horiarray];
1160     *motrlft = array[vertarray+3][horiarray];
1161
1162     printf("\nrt = %d          ",motrrgt);
1163     printf("lft = %d\n",motrlft);
1164     refresh();
1165     /* motrlft = 2048;
1166     motrrgt = 2048;*/
1167     comedi_data_write(device,1,0,1,aref,*motrlft);
1168     comedi_data_write(device,1,1,1,aref,*motrrgt);
1169
1170 }
1171
1172

```