```
1  /*****************************************************************************
2  *                                Duckshoot Program                           *
3  *                           Copyright 2007. All Rights Reserved.             *
4  *                                                                            *
5  * Author:      Jonathan Ambrose                                              *
6  * Date:        First Release to Manufacturing: 18th December                 *
7  * Version:     Final Release To Manufacturing (RTM)                          *
8  * Description: A program that manipulates and controls a bit pattern that is sent*
9  *              to an LED display.The bit pattern iscontroled by four switches.  *
10 *              One switch acts as a gun shooting out the top LED of the rack.   *
11 *              Two switches must be used to determine four speeds the LEDs      *
12 *              rotate to.                                                    *
13 *              The bottom switch controls the direction of the rotating pattern. *
14 *                                                                            *
15 * Compile: gcc -Wall -pedantic -ansi -lcomedi duckshoot1.c -o duckshoot      *
16 * Run Code:  ./duckshoot1                                                    *
17 *                                                                            *
18 *List of Functions and Sub sections of functions:                            *
19 *   Main Program                                                             *
20 *   Functions                                                                *
21 *       INTIALISE RACK                                                       *
22 *           WRITE LIGHTS                                                     *
23 *       LIGHTS                                                               *
24 *           SPEED SETTING                                                    *
25 *           DIRECTION                                                        *
26 *       SHOOTING DUCKS                                                       *
27 *           MACHINE GUNNING PENALTIES                                        *
28 *       TEST END OF GAME                                                     *
29 *       WELCOME MESSAGE                                                      *
30 *                                                                            *
31 *****************************************************************************/
32
33 /* # Include Section
34 *****************************************************************************/
35 #include <stdio.h>
36 #include <stdlib.h>
37 #include <fcntl.h>
38 #include <unistd.h>
39 #include <errno.h>
40 #include <getopt.h>
41 #include <ctype.h>
42 #include <comedi.h>
43 #include <comedilib.h>
44 /*****************************************************************************/
45
46 /* Function Prototype Section
47 * Prototypes for all functions called by this module, with the exception of runtime
   routines.
48 *****************************************************************************/
49 void welcome_message (void);
50 void lights (unsigned int *usleeptime, unsigned int subdevice, unsigned int *data);
51 void shootswitch (unsigned int subdevice, unsigned int *data, unsigned int *usleeptime,
52       unsigned int *fired);
53 int initialiserack(unsigned int subdevice);
54 comedi_t *device;
55 int testendofgame (unsigned int *data);
56
57 /*****************************************************************************/
58
59
60
61
62
63
64
65
66
67
68
69
70
```

```
 71
 72    /*****************************************************************************
 73    *MAIN PROGRAM                                                                *
 74    *****************************************************************************/
 75
 76    int main(void)
 77
 78    {
 79
 80      unsigned int subdevice= 2;/*used for marco rack*/
 81
 82      unsigned int usleeptime;/*Asssigned to USLEEP*/
 83
 84      unsigned int data; /*variable to store light pattern to goto the LED's*/
 85
 86      unsigned int fired = 0;
 87
 88      welcome_message(); /*print contents displayed in the Welcome Message Function*/
 89
 90      data = initialiserack(subdevice);
 91
 92        while(1)
 93
 94            {   /*function edits number of ducks written on the rack, taking into account
 95                penultys*/
 96
 97                shootswitch (subdevice, &data, &usleeptime, &fired);
 98
 99                /*function edits ducks, in which they are displayed on the rack in the sense
100                of their direction and speed.*/
101
102                lights (&usleeptime,subdevice,&data);
103
104                if (testendofgame(&data)) /*if testend() returns 1 then program will
105                                            end normally*/
106
107                return(0); /*return 0 to break out of while loop in order to exit
108                             the program*/
109
110            }
111
112        return(0);
113
114    }
115
116    /*****************************************************************************
117                                          FUNCTIONS
118    *****************************************************************************/
119
120
121    /*****************************************************************************
122    *                                  INTIALIZE RACK                            *
123    *                                                                            *
124    * Function name : int initialiserack(unsigned int subdevice)                 *
125    * Returns : return value data (Ducks)                                        *
126    * Created by : Jon Ambrose but some code from Ian Johnson                    *
127    * Date created : 19/12/07                                                    *
128    * Description : This function sets up the board and tests to see             *
129    *               if it is working. Then the initial value for the lights.     *
130    *****************************************************************************/
131
132    /*Initialise the marco rack so that the lights are on and they are 10101010 = AA */
133
134    int initialiserack(unsigned int subdevice)
135
136    {
137        unsigned int data = 0xaa;
138
139        unsigned int mask;
140
141        int i;
```

```
142
143         int retval;
144
145         int stype;
146
147         char *filename = "/dev/comedi0";
148
149     /* open the comedi device, exit with error message if something goes wrong. */
150
151       device=comedi_open(filename);
152
153         if(!device)
154
155         {
156           comedi_perror(filename);
157           exit(0);
158         }
159
160         for(i=0;i<8;i++)
161
162
163         retval=comedi_dio_config(device,subdevice,i,COMEDI_OUTPUT);
164
165
166         /* check we have a valid bidirection DIO port subdevice*/
167
168         stype = comedi_get_subdevice_type(device,subdevice);
169
170         if(stype!=COMEDI_SUBD_DIO)
171
172         {
173           printf("%d is not a digital I/O subdevice\n",subdevice);
174           exit(0);
175         }
176
177         data = 0xAA; /*set initial light pattern and store in lights to be send ro rack*/
178                      /*Number of Ducks on the Rack shown - should be AA for alternate */
179
180         /*****************************************************
181         **                   WRITE LIGHTS                   **
182         **                                                  **
183         **Changes that have been made to the light pattern **
184         **so the value stored in lights can be written to  **
185         **the board                                        **
186         *****************************************************/
187
188         mask = 0xff; /*Write Lights*/
189
190         comedi_dio_bitfield(device,subdevice,mask, &data);
191
192         return data;
193
194     }
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
```

```
213  /*****************************************************************************
214  *                                  Lights Function                           *
215  *                                                                            *
216  * Function name : void lights (unsigned int *usleeptime, unsigned int subdevice,     *
217  *                              unsigned int *data)                           *
218  * Returns : return value data (Ducks)                                        *
219  * Created by : Jon Ambrose                                                   *
220  * Date created : 19/12/07                                                    *
221  * Description : This funtction reads the Speed Switches to work out what speed for   *
222  *               the program to run at. It runs 1 of 4 speeds                 *
223  *               This function also reads the direction switch to find out what       *
224  *               direction to rotate the LEDS in. Either up or down.          *
225  *               This function will then write the results and update the LEDS (Data).*
226  *****************************************************************************/
227
228   void lights (unsigned int *usleeptime, unsigned int subdevice, unsigned int *data)
229
230  {
231       int read=0x00;
232
233       int write=0xFF;
234
235       unsigned int speedswitch;
236
237       int delay1=200000; /*USleep 1 Option */
238
239       int delay2=300000; /*USleep 2 Option */
240
241       int delay3=400000; /*USleep 3 Option */
242
243       int delay4=800000; /*USleep 4 Option */
244
245       unsigned int directionswitch;
246
247
248      /*****************************************************
249      **                   SPEED SETTING                  **
250      **                                                  **
251      **test the two speed switches and returns a value  **
252      **to go into usleep.                                **
253      *****************************************************/
254
255      /*Reads the Speed Switch */
256      comedi_dio_bitfield(device,subdevice,read, &speedswitch);
257
258      speedswitch =  speedswitch & 0x00ff00;
259
260      speedswitch >>=8;
261
262      speedswitch =~speedswitch;
263
264      speedswitch = speedswitch & 0x00000006;
265
266
267      switch(speedswitch)
268
269       {
270
271          case 0x00:*usleeptime = delay1;break; /*test if no switches are on DEFAULT speed*/
272
273          case 0x02:*usleeptime = delay2;break; /*test if switch 2 speed switch is on*/
274
275          case 0x04:*usleeptime = delay3;break; /*test if switch 3 speed switch is on*/
276
277          case 0x06:*usleeptime = delay4;break; /*test if switch 2 and 3 speed switch is
278                                                  on*/
279
280       }
281
282
283
```

```
284         /*****************************************************
285         **                  DIRECTION AND ROTATE             **
286         **                                                   **
287         **test the direction switch and rotate in the       **
288         **corresponding direction                           **
289         *****************************************************/
290
291         /*READ Direction Switch*/
292
293         comedi_dio_bitfield(device,subdevice,read,&directionswitch);
294
295         directionswitch = directionswitch & 0x00ff00;
296
297         directionswitch >>=8;
298
299         directionswitch =~directionswitch;
300
301         directionswitch =directionswitch & 0x00000001; /*tests 1st switch for on*/
302
303         /*Move Lights - in association with the direction switch. I.E Whether to
304         make the LEDS rotate UP or DOWN*/
305
306         if (directionswitch == 1)
307
308         {
309             if((*data & 0x01)==1) /*If LED (Duck) is on*/
310
311                 {
312                     *data = *data & 0xFE;
313                     *data = *data >> 1; /*move all the lights down the rack*/
314                     *data = *data | 0x80; /*keep on light that falls off the bottom by
315                                     lighting up the top light*/
316                 }
317
318             else
319
320                 *data = *data >> 1; /*move all the lights down the rack*/
321
322         } /*backward if the switch is in the off position*/
323
324        else
325
326         {
327             if((*data & 0x80)==0x80) /*If LED (Duck) is off*/
328
329                 {
330
331                     *data = *data & 0x7F;
332                     *data = *data <<1; /*move all the lights down the rack*/
333                     *data = *data | 0x01; /*keep the light that just fell off the bottom by
334                                     lighting up the top light*/
335
336                 }
337
338             else
339
340                 *data = *data << 1; /* move lights up the rack*/
341
342         }
343
344         usleep(*usleeptime); /*Run the USLEEP time - varibles are defined above */
345
346         comedi_dio_bitfield(device,subdevice,write,data);
347
348         *data = *data & 0xFF;
349
350     }
351
352
353
354
```

```
355  /*****************************************************************************
356  *                                  SHOOTING DUCKS FUNCTION                   *
357  *                                                                            *
358  * Function name : void lights (unsigned int *usleeptime, unsigned int subdevice,     *
359  *                              unsigned int *data)                           *
360  * Returns : return value data (Ducks)                                        *
361  * Created by : Jon Ambrose                                                   *
362  * Date created : 19/12/07                                                    *
363  * Description : When top switch is on it will shootout the top led or add a penulty  *
364  *               duck if there is no duck to shoot.                           *
365  *               Machine gunning is where the shoot switch is continuously down.      *
366  *               This also adds a penulty duck.                               *
367  *****************************************************************************/
368
369
370      void shootswitch (unsigned int subdevice, unsigned int *data,unsigned int *usleep,
     unsigned int *fired)
371
372  {
373
374      unsigned int shootswitch;
375
376      int read = 0x00;
377
378      int write = 0xFF;
379
380      /*READ the shoot switch*/
381
382      comedi_dio_bitfield(device,subdevice,read,&shootswitch); /* its telling the device
383                                                                  to read the switch*/
384
385      shootswitch = shootswitch & 0x00ff00;
386
387      shootswitch >> = 8;
388
389      shootswitch = ~ shootswitch;
390
391      shootswitch = shootswitch & 0x80;
392
393      /*Shoot or Relight the Lights*/
394
395      if ((shootswitch & 0x80) == 0x80)
396
397          {
398              if (*fired == 0)
399              {
400                  if ((*data&0x80) == 0x80)
401
402                      {
403                          *data= *data ^ 0x80; /* ^ is inverting the bit with 0.
404                                                  Invert the top duck (1 -> 0     OR
405                                                  0 -> 1) meaning 7f = 0111 1111 */
406                      }
407
408              else
409
410                      {
411                          *data= *data ^ 0x80;
412                      }
413
414                  *fired = 1;
415
416              }
417
418         else
419
420
421
422
423
424
```

```
425         /*****************************************************
426         **                  Machine Gunning Penaltys         **
427         **                                                   **
428         **If the Shoot switch is left down then it will     **
429         **give you a penalty which adds a duck to the next **
430         **non lit duck. To stop this the switch has to be  **
431         **"reloaded", which means the switch needs to be   **
432         **turned off, and then the while loop will re-run  **
433         *****************************************************/
434
435                 {
436
437                     if ((*data & 0x01) == 0) /*if bottom duck is not on*/
438
439                         {
440                             *data = *data | 0x01; /*then invert it*/
441                         }
442
443                     else if ((*data & 0x02) == 0)
444
445                         {
446                             *data = *data | 0x02;
447                         }
448
449                     else if ((*data & 0x04)== 0)
450
451                         {
452                             *data = *data | 0x04;
453                         }
454
455                     else if ((*data & 0x08) == 0)
456
457                         {
458                             *data = *data | 0x08;
459                         }
460                     else if ((*data & 0x10)== 0)
461
462                         {
463                             *data = *data | 0x10;
464                         }
465
466                     else if ((*data & 0x20) == 0)
467
468                         {
469                             *data = *data | 0x20;
470                         }
471
472                     else if ((*data & 0x40)== 0)
473
474                         {
475                             *data = *data | 0x40;
476                         }
477
478                     else
479
480                         {
481                             *data = *data | 0x80; /*if the program gets to this phase
482                                                     then the game will be lost*/
483                         }
484
485                 }
486         /*write the lights*/
487         comedi_dio_bitfield(device,subdevice,write,data);
488
489         *data = *data & write;
490
491             }
492
493         else
494
495             *fired = 0;
```

```
496
497     }
498
499     /*****************************************************************************************
500     *                                    TEST END OF GAME FUNCTION                           *
501     *                                                                                        *
502     * Function name : int testendofgame(unsigned int *data)                                  *
503     * Returns :       return value data (Ducks)                                              *
504     * Created by :    Jon Ambrose                                                            *
505     * Date created :  19/12/07                                                               *
506     * Description :   This function will test for the different options in terms of          *
507     *                 whether or not you have won the game.                                  *
508     *                 The funtion will find out how many ducks are lit, by working out       *
509     *                 the hex equivalent, and checking it against the if statement.          *
510     *                 I.E Either 0x00 (0) - No ducks lit, or 0xFF (225)                      *
511     *****************************************************************************************/
512
513     int testendofgame(unsigned int *data)
514
515     {
516         if ((*data == 0xFF)|(*data ==0)) /*test extremes of game win or  lose*/
517
518             {
519
520                 if (*data == 0) /* if no LEDS (ducks) are lit on the rack, then the
521                                    game will stop and display the message "You Have One".*/
522
523                     {
524
525                         printf("YOU WIN DUCKSHOOT!!\n\n\n");
526
527                         return 1;
528
529                     }
530
531                 else
532                                 /* if all LEDS (ducks) are lit on the rack, then the*/
533                     {           /*game will stop and display the message "You Have Lost".*/
534
535                         printf("YOU LOSE! , GAME OVER!\n\n\n");
536
537                         return 1;
538
539                     }
540
541             }
542
543         return 0;
544
545     }
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
```

```
567  /*****************************************************************************************
568  *                                  WELCOME MESSAGE FUNCTION                             *
569  *                                                                                       *
570  * Function name : int testendofgame(unsigned int *data)                                 *
571  * Returns :       return value data (Ducks)                                             *
572  * Created by :    Jon Ambrose                                                           *
573  * Date created :  19/12/07                                                              *
574  * Description :   Displays the Instructions and Controls of the game -                  *
575  *                 this will appear at the beginning of the program.                     *
576  *****************************************************************************************/
577
578  void welcome_message(void)
579      {
580
581          printf("*********************************************************\n");
582          printf("*_______________________________________________________*\n");
583          printf("*         WELCOME TO DUCKSHOOT v1. RTM RELEASE          *\n");
584          printf("*               Written by Jon Ambrose                  *\n");
585          printf("*_______________________________________________________*\n");
586          printf("* Introduction: Basically the object of the game is to  *\n");
587          printf("* Shoot the ducks, untill all the LED's are not lit.    *\n");
588          printf("*                                                       *\n");
589          printf("*                                                       *\n");
590          printf("*                                                       *\n");
591          printf("* CONTROLS:                                             *\n");
592          printf("*     SW7  | = Shoot the duck:-                        *\n");
593          printf("*          Switch needs to be returned to 0 after firing*\n");
594          printf("*     SW6  | = Not used                                 *\n");
595          printf("*     SW5  | = Not used                                 *\n");
596          printf("*     SW4  | = Not used                                 *\n");
597          printf("*     SW3  | = Not used                                 *\n");
598          printf("*     SW2  | = SPEED SELECTION                          *\n");
599          printf("*     SW1  | = SPEED SELECTION                          *\n");
600          printf("*     SW0  | = reverse direction of lights              *\n");
601          printf("*              Switch 0 = Direction UP                  *\n");
602          printf("*              Switch 1 = Direction DOWN                *\n");
603          printf("* SPEED SELECTION:                                      *\n");
604          printf("*                  switch USLEEP Times:                 *\n");
605          printf("*    SLOW wait time  USLEEP1 =    0                     *\n");
606          printf("*                                 0  = 0                *\n");
607          printf("*                    _____________                      *\n");
608          printf("*                    USLEEP2 =    0                     *\n");
609          printf("*                                 1  = 2                *\n");
610          printf("*                    _____________                      *\n");
611          printf("*                    USLEEP3 =    1                     *\n");
612          printf("*                                 0  = 4                *\n");
613          printf("*                    _____________                      *\n");
614          printf("*    HIGH wait time  USLEEP4 =    1                     *\n");
615          printf("*                                 1  = 6                *\n");
616          printf("*                    _____________                      *\n");
617          printf("*_______________________________________________________*\n");
618          printf("*********************************************************\n");
619          printf("\n");
620          printf("\n");
621      }
622
```