```c
/* Copyright (c) 2011, RidgeRun
 * All rights reserved.
 *
From https://www.ridgerun.com/developer/wiki/index.php/Gpio-int-test.c

 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    This product includes software developed by the RidgeRun.
 * 4. Neither the name of the RidgeRun nor the
 *    names of its contributors may be used to endorse or promote products
 *    derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY RIDGERUN ''AS IS'' AND ANY
 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL RIDGERUN BE LIABLE FOR ANY
 * DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <poll.h>
#include <signal.h> // Defines signal-handling functions (i.e. trap Ctrl-C)
#include "gpio-utils.h"

 /****************************************************************
 * Constants
 ****************************************************************/

#define POLL_TIMEOUT (3 * 1000) /* 3 seconds */
#define MAX_BUF 64

/****************************************************************
 * Global variables
 ****************************************************************/
int keepgoing = 1;  // Set to 0 when ctrl-c is pressed

```

```
55  /****************************************************************
56   * signal_handler
57   ****************************************************************/
58  void signal_handler(int sig);
59  // Callback called when SIGINT is sent to the process (Ctrl-C)
60  void signal_handler(int sig)
61  {
62      printf( "Ctrl-C pressed, cleaning up and exiting..\n" );
63      keepgoing = 0;
64  }
65
66  /****************************************************************
67   * Main
68   ****************************************************************/
69  int main(int argc, char **argv, char **envp)
70  {
71      struct pollfd fdset[2];
72      int nfds = 2;
73      int gpio_fd, timeout, rc;
74      char buf[MAX_BUF];
75      unsigned int gpio;
76      int len;
77
78      if (argc < 2) {
79          printf("Usage: gpio-int <gpio-pin>\n\n");
80          printf("Waits for a change in the GPIO pin voltage level or input on stdin\n");
81          exit(-1);
82      }
83
84      // Set the signal callback for Ctrl-C
85      signal(SIGINT, signal_handler);
86
87      gpio = atoi(argv[1]);
88
89      gpio_export(gpio);
90      gpio_set_dir(gpio, "in");
91      gpio_set_edge(gpio, "both");  // Can be rising, falling or both
92      gpio_fd = gpio_fd_open(gpio, O_RDONLY);
93
94      timeout = POLL_TIMEOUT;
95
96      while (keepgoing) {
97          memset((void*)fdset, 0, sizeof(fdset));
98
99          fdset[0].fd = STDIN_FILENO;
100         fdset[0].events = POLLIN;
101
102         fdset[1].fd = gpio_fd;
103         fdset[1].events = POLLPRI;
104
105         rc = poll(fdset, nfds, timeout);
106
107         if (rc < 0) {
108             printf("\npoll() failed!\n");
```

```
109                 return -1;
110             }
111
112             if (rc == 0) {
113                 printf(".");
114             }
115
116             if (fdset[1].revents & POLLPRI) {
117                 lseek(fdset[1].fd, 0, SEEK_SET);  // Read from the start of the file
118                 len = read(fdset[1].fd, buf, MAX_BUF);
119                 printf("\npoll() GPIO %d interrupt occurred, value=%c, len=%d\n",
120                      gpio, buf[0], len);
121             }
122
123             if (fdset[0].revents & POLLIN) {
124                 (void)read(fdset[0].fd, buf, 1);
125                 printf("\npoll() stdin read 0x%2.2X\n", (unsigned int) buf[0]);
126             }
127
128             fflush(stdout);
129         }
130
131         gpio_fd_close(gpio_fd);
132         return 0;
133     }
134
135
```