```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <signal.h>
#include <sched.h>

int fd;

volatile long interrupt_counter;
long ref,data_counter;
short data[8001*10];
volatile char data_flag,data_done;
long timer_value,sf;
long u,x,oldx;
short write_buffer[3];
short read_buffer[3];
float Kp=1.0;

void led_key(void)
{
    read(fd,read_buffer,3);
    if ((read_buffer[2] & 0x0001) == 0) {
        write_buffer[2] |= 0x0001;
    }
    else {
        write_buffer[2] &= 0xfffe;
    }
    if ((read_buffer[2] & 0x0002) == 0) {
        write_buffer[2] |= 0x0002;
    }
    else {
        write_buffer[2] &= 0xfffd;
    }
    if ((read_buffer[2] & 0x0004) == 0) {
        write_buffer[2] |= 0x0004;
    }
    else {
        write_buffer[2] &= 0xfffb;
    }
    write(fd,write_buffer,3);
}

void seven_segment_display(short ad0_value)
{
    char digit[3];
    short ad0_volt;
    short tmp;

    ad0_volt = (ad0_value-2048)*1000/2048;
    if (ad0_volt < 0 ) {
        ad0_volt = -ad0_volt;
        write_buffer[2] |= 0x0080;
        write(fd,write_buffer,3);
    }
    else {
        write_buffer[2] &= 0xff7f;
        write(fd,write_buffer,3);
    }

    digit[0] = ad0_volt/100;
    digit[1] = (ad0_volt % 100)/10;
    digit[2] = ad0_volt % 10;

    tmp = 0xe0 + digit[2];
    write_buffer[2] = (write_buffer[2] & 0x00ff) + (tmp<<8);
    write(fd,write_buffer,3);
    tmp = 0xf0 + digit[2];
```

```
    write_buffer[2] = (write_buffer[2] & 0x00ff) + (tmp<<8);
    write(fd,write_buffer,3);

    tmp = 0xd0 + digit[1];
    write_buffer[2] = (write_buffer[2] & 0x00ff) + (tmp<<8);
    write(fd,write_buffer,3);
    tmp = 0xf0 + digit[1];
    write_buffer[2] = (write_buffer[2] & 0x00ff) + (tmp<<8);
    write(fd,write_buffer,3);

    tmp = 0xb0 + digit[0];
    write_buffer[2] = (write_buffer[2] & 0x00ff) + (tmp<<8);
    write(fd,write_buffer,3);
    tmp = 0xf0 + digit[0];
    write_buffer[2] = (write_buffer[2] & 0x00ff) + (tmp<<8);
    write(fd,write_buffer,3);
}

void usrsignal(int sig)
{
    interrupt_counter++;
    if (interrupt_counter >= sf*4)
    {
        interrupt_counter=0;
        if (data_flag==1)
        {
            data_counter=0; data_flag=2;
        }
        ref=410;
    }
    if (interrupt_counter >= sf*2)
    {
        ref=0;
    }
    read(fd,read_buffer,1);
    seven_segment_display(read_buffer[0]);
    led_key();
    x= read_buffer[0] -2048;

    if (data_flag==2)
    {
        if (data_counter<=sf*4)
        {
            data[data_counter++]=(short)x;
        }
        else
        {
            data_done=1;
        }
    }

    /* Your control algorithm */
    u = Kp*(float)(ref-x);
    oldx = x;
    if (u > 2047) u=2047;
    if (u < -2048) u=-2048;
    /* end of your control algorithm */

    u = u+2048;
    write_buffer[0] = u;
    write(fd,write_buffer,1);
}
void FileSave(void)
{
    FILE *fp;
    int i;

    fp=fopen("data","w+b");
    printf("Data Capture Started \n");
    data_flag=1;
    while(data_done!=1)usleep(1);
```

```
143         printf("Data Capture Finished \n");
144         for (i=0;i<sf*4;i++) {
145             fprintf(fp,"%d %d\n",i,data[i]);
146         }
147         fclose(fp);
148     }
149     int main(void)
150     {
151         unsigned long id;
152         unsigned char c;
153         char reply;
154         struct sched_param param, new_param;
155 
156         printf("start policy = %d\n", sched_getscheduler(0));
157         /* 0 <- SCHED_OTHER, 1 <- SCHED_FIFO, 2 <- SCHED_RR */
158         param.sched_priority=59;
159         printf("max priority = %d, min priority = %d, my priority = %d\n",
160             sched_get_priority_max(SCHED_FIFO), sched_get_priority_min(SCHED_FIFO),
            param.sched_priority);
161 
162         if (sched_setscheduler(0, SCHED_FIFO, &param) != 0){
163             perror("sched_setscheduler failed\n");
164             return 0;
165         }
166 
167         fd = open("/dev/pci_adda",O_RDWR);
168         if (fd < 0){
169             printf("/dev/pci_adda open error\n");
170             exit(1);
171         }
172 
173         interrupt_counter=0;
174         data_counter=0;
175         data_flag=0;
176         data_done=0;
177         sf=1000;
178         printf("Sampling Frequency = %ld Hz \n",sf);
179         timer_value=10000000/(sf*100);
180         ioctl(fd,1,&timer_value,1);
181 
182         (void)signal(SIGUSR1,usrsignal);
183         id = getpid();
184         ioctl(fd,0,&id,1);
185 
186         printf ("Type Sampling Frequency and press ENTER: \n");
187         scanf("%ld",&sf);
188         printf("Sampling Frequency = %ld Hz \n",sf);
189         timer_value=10000000/(sf*100);
190         ioctl(fd,1,&timer_value,1);
191         printf("Controller is ready. Power on the system\n");
192         printf ("Would you like to save in a file? (y or n and press ENTER): \n");
193         scanf("%s",&reply);
194         if (reply=='Y' || reply=='y')
195         {
196             FileSave();
197         }
198         printf("Type Ctrl-C to stop\n");
199         while(1) {
200             //seven_segment_display(read_buffer[0]);
201             usleep(10000);
202         }
203         close(fd);
204     }
205 
```