Université Paul Sabatier

Systèmes Temps Réel

Compte Rendu de TP Sujet -

Auteurs :
David TOCAVEN
Lucien RAKOTOMALALA

Encadrant :
Hamid DEMMOU

17 décembre 2017

Table des matières

Introduction

Chapitre 1

TP 1 : Iniiation a un OS temps Réel basé sur Linux

1.1 Mesures sous linux

1.1.1 Programme *carrelinux − comedi.c*

Ce premier programme est un générateur de signal carré. Il va nous permettre d'analyser les réponses temps réel de en étant basé sur Linux. Ntre première analyse du programme donne :

— Fonction *Void out* envoie un signal carré. La fréquence semble être défini ailleurs dans le programme. L'amplitude du signal est un niveau logique de LOW à $HIGH$.

— Initialisation d'une structure de temps dans le main. La librairie Comedio

dans la main est init une structure de temps, est ensuite ouvert la carte entrée sortie.la carte est paramétrée en sortie sur les ports 0 et 1. ensuite, l'algorithme attend. initialisation d'une horloge qui va attendre un temps correspondant a la demi-période du signal carré généré

Pour mesurer les modifications de période, nous avons crée deux variables *timespec* : une qui mesure le temps précédent le sleep, une qui mesure a la fin de l'instance $while(1)$. La mesure de la δ est : $\delta = t_{debut} - t_{fin}$.

Mise en place d'un *gnuplot* pour afficher les 5000 dernières périodes.

Observation :

— pour aucune charges de linux, les périodes restent à $50\mu s$.

— pour un simple

1.2 Mesures sous RTAI

1.2.1 L'environnement RTAI Kernel

Dans cette partie, nous avons refait la fonctionnalité précédente, c'est-à-dire un générateur de signal carré à fréquence fixe de 100 microsecondes, sous la forme d'un processus temps réel.

Chapitre 2

Chapitre 3

Chapitre 4

Conclusion

Annexes

Annexe 1 - TP 1

Code partie 1

```c
/* compile using  "gcc -o swave swave.c -lrt -Wall"  */
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <sched.h>
#include <signal.h>
#include <sys/io.h>
#include </usr/local/src/comedilib/include/comedilib.h>

#define NSEC_PER_SEC    1000000000
#define DIO 2
#define SIZETAB 5000

comedi_t *cf;


/* array who contain all outputs periodes */
   unsigned int deltat[SIZETAB];

/* the struct timespec consists of nanoseconds
 * and seconds. if the nanoseconds are getting
 * bigger than 1000000000 (= 1 second) the
 * variable containing seconds has to be
 * incremented and the nanoseconds decremented
 * by 1000000000.
 */
static inline void tsnorm(struct timespec *ts)
{
   while (ts->tv_nsec >= NSEC_PER_SEC) {
      ts->tv_nsec -= NSEC_PER_SEC;
      ts->tv_sec++;
   }
}

/* increment counter and write to parallelport */
void out()
{
   static unsigned char state=0;
   comedi_dio_write(cf,DIO,0,state);
  state=!state;
}

/* for print the datas into a file */
void IntHandler(int sig)
{
  FILE *file;
  int i;
```

```
  file = fopen("/home/m2istr_13/Documents/TP_RTAI/M2ISTR_RTAI/tp1_mesures_TR/I/
      delta.res","w");
  if(file==NULL)
  {
    fprintf(stderr,"Erreur de creation du fichier\n");
  }
  for(i=0;i<SIZETAB;i++)
  {
    fprintf(file,"%d %d\n",i,deltat[i]);
  }
  fclose(file);
  exit(0);
}

/*****************************  MAIN
   ****************************************/
int main()
{
   struct timespec t, /* for output signal */
        t1,        /* for get the time at the beginning of the loop */
          t2;          /* for get the time at the end of the loop */

   /* default interval = 50000ns = 50us
    * cycle duration = 100us
    */
   int interval=50000,
       i = 0; /* for moving into deltat */

   /*    attatch the Ctr+C action to print file  */
   signal(SIGINT,IntHandler);
   cf=comedi_open("/dev/comedi0");
   if(cf==NULL)
   {
      comedi_perror("Comedi fails to open");
      return -1;
   }

   // Configure le device ANALOG_OUTPUT pour envoyer les donnees signaux
   comedi_dio_config(cf,DIO,0,COMEDI_OUTPUT);
   comedi_dio_config(cf,DIO,1,COMEDI_OUTPUT);

   /* get current time */
   clock_gettime(0,&t);

   /* start after one second */
   t.tv_sec++;

   while(1){
      /* wait untill next shot */
      clock_gettime(0,&t1);
      clock_nanosleep(0, TIMER_ABSTIME, &t, NULL);
      /* do the stuff */
      out();
      /* calculate next shot */
      t.tv_nsec+=interval;
      tsnorm(&t);
      clock_gettime(0,&t2);
      deltat[i] = t2.tv_nsec - t1.tv_nsec + t2.tv_sec - t1.tv_sec;
      i=(i+1)%SIZETAB;
   }
   return 0;
```

|| }

Code partie 2

Script du bash go

Listing 4.1 – go

```
sudo /sbin/insmod /usr/realtime/modules/rtai_hal.ko
sudo /sbin/insmod /usr/realtime/modules/rtai_lxrt.ko
sudo /sbin/insmod /usr/realtime/modules/rtai_fifos.ko

sudo modprobe comedi
sudo modprobe kcomedilib


sudo /sbin/insmod /usr/realtime/modules/rtai_comedi.ko
sudo modprobe ni_pcimio


# Remplacer test.ko par le module compile
sudo /sbin/insmod ./squelet.ko
```

Script du bash stop

Listing 4.2 – stop

```
# Remplacer test par le module compile
sudo /sbin/rmmod squelet
```

Script du Makefile

Listing 4.3 – Makefile

```
# Remplacer foo par le fichier objet (.o) compile
obj-m   := squelet.o

KDIR    := /lib/modules/$(shell uname -r)/build
PWD             := $(shell pwd)
EXTRA_CFLAGS := -I/usr/realtime/include -I/usr/include/ -I/usr/local/include  -D__IN_RTAI_


default: clean
        $(MAKE) -lcomedi -lm -C $(KDIR) SUBDIRS=$(PWD) modules
        chmod +x squelet.o

clean :
        rm -f   $(obj-m)
```

Source de l'application RTAI

Listing 4.4 – squelet.c

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <rtai.h>
#include <rtai_sched.h>
#include <rtai_fifos.h>
#include <rtai_proc_fs.h>
#include <comedilib.h>

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Squelette de programme RTAI et carte ni-6221");
MODULE_AUTHOR("RAKOTOMALALA FARHI TOCAVEN");

// times unit
#define ms  1000000
#define microsec 1000
// min-max output signal values
#define CARRE_HIGH 1.22
#define CARRE_LOW 0

#define CAN 0
#define CNA 1
#define DIO 2
#define CHAN_0 0
#define CHAN_1 1
#define CAN_RANGE 1 // [-5, +5]
#define PERIOD 100 // microsecond
#define SIZETAB 5000
#define deltat 0    // FIFO

static RT_TASK Tache1_Ptr; // Pointeur pour la tache 1
static RT_TASK IT_handler_Ptr; // Pointeur pour la tache de reprise de main

comedi_t *cf; // la carte
RTIME curTime; // un timer pour mesurer les dif de temps

/****************** tache1 ***************************/
void Tache1 (long int x)
{
  int voie    = x,
      composant   = DIO,
      i     =0,   /*position dans deltat*/
    delta_i   = 0;
  while (1)
  {
    curTime = rt_get_time(); // read beginning time
    delta_i=(int)curTime; // keep current time
    comedi_dio_write(cf,composant,voie,CARRE_HIGH);
    rt_task_wait_period();
    comedi_dio_write(cf,composant,voie,CARRE_LOW);
    rt_task_wait_period();
    curTime = rt_get_time(); // read final times
    delta_i = (int)curTime - delta_i; // calc period

    i=(i+1)%SIZETAB; // increment counter modulo SIZETAB for count the number of
        values
```

```
    if( i > SIZETAB)
    {
      if(rtf_put(deltat,&delta_i,SIZETAB)<0)
      {
        printk("probleme ecrite fifo : /dev/rtf0\n");
      }
    }
  }
}

/****************** init ************************/

int init_module(void)
{
  RTIME now ;
    // Initialisation de la carte d'E/S
  cf = comedi_open("/dev/comedi0");
  if(cf == NULL)
  {
    comedi_perror("Comedi fails to open");
    return -1;
  }
  // init FIFO pour mesures
  // Configurer le device DIGITAL_INPUT pour recevoir les donnees signaux
  // et DIGITAL_OUTPUT pour envoyer les donnees signaux
  comedi_dio_config(cf,DIO,0,COMEDI_OUTPUT);
    comedi_dio_config(cf,DIO,1,COMEDI_OUTPUT);
  rt_set_oneshot_mode();
  // Lancement du timer

  now =  start_rt_timer(0); // creation timer de periode  50ms
  curTime = start_rt_timer( nano2count(0));
  now = rt_get_time();
    // Lancement des taches
  rt_task_init(&Tache1_Ptr,    /* Le prt de tache */
          Tache1,      /* Nom de la tache */
          1,     /* valeur du parametre X */
          2000,    /* Taille de la pile necessaire pour la tache (memoire tempo
              utilisee) */
          0,     /* Priorite */
          0,       /* Pas de calcul flottant */
          0);      /* choix d'un signal ou non  */


  rt_task_make_periodic( &Tache1_Ptr,  /* Pointeur vers la tache */
        now,     /* Instant de depart */
        nano2count(PERIOD*microsec/2));    /* Periode */


  rtf_create(deltat, SIZETAB);
  return 0;
}

/****************** cleanup ************************/
void cleanup_module(void)
{
  stop_rt_timer();
  rt_task_delete(&Tache1_Ptr);
 }
```

Source du processus de récupération de données

Listing 4.5 – writeToFile.c

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <rtai.h>
#include <rtai_sched.h>
#include <rtai_fifos.h>
#include <rtai_proc_fs.h>
#include <comedilib.h>

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Squelette de programme RTAI et carte ni-6221");
MODULE_AUTHOR("RAKOTOMALALA FARHI TOCAVEN");

// times unit
#define ms  1000000
#define microsec 1000
// min-max output signal values
#define CARRE_HIGH 1.22
#define CARRE_LOW 0

#define CAN 0
#define CNA 1
#define DIO 2
#define CHAN_0 0
#define CHAN_1 1
#define CAN_RANGE 1 // [-5, +5]
#define PERIOD 100 // microsecond
#define SIZETAB 5000
#define deltat 0    // FIFO

static RT_TASK Tache1_Ptr; // Pointeur pour la tache 1
static RT_TASK IT_handler_Ptr; // Pointeur pour la tache de reprise de main

comedi_t *cf; // la carte
RTIME curTime; // un timer pour mesurer les dif de temps

/****************** tache1 ***************************/
void Tache1 (long int x)
{
  int voie    = x,
      composant   = DIO,
      i     =0,   /*position dans deltat*/
    delta_i   = 0;
  while (1)
  {
    curTime = rt_get_time(); // read beginning time
    delta_i=(int)curTime; // keep current time
    comedi_dio_write(cf,composant,voie,CARRE_HIGH);
    rt_task_wait_period();
    comedi_dio_write(cf,composant,voie,CARRE_LOW);
    rt_task_wait_period();
    curTime = rt_get_time(); // read final times
    delta_i = (int)curTime - delta_i; // calc period

    i=(i+1)%SIZETAB; // increment counter modulo SIZETAB for count the number of
        values
```

```
    if( i > SIZETAB)
    {
      if(rtf_put(deltat,&delta_i,SIZETAB)<0)
      {
        printk("probleme ecrite fifo : /dev/rtf0\n");
      }
    }
  }
}

/****************** init ************************/

int init_module(void)
{
  RTIME now ;
    // Initialisation de la carte d'E/S
  cf = comedi_open("/dev/comedi0");
  if(cf == NULL)
  {
    comedi_perror("Comedi fails to open");
    return -1;
  }
  // init FIFO pour mesures
  // Configurer le device DIGITAL_INPUT pour recevoir les donnees signaux
  // et DIGITAL_OUTPUT pour envoyer les donnees signaux
  comedi_dio_config(cf,DIO,0,COMEDI_OUTPUT);
    comedi_dio_config(cf,DIO,1,COMEDI_OUTPUT);
  rt_set_oneshot_mode();
  // Lancement du timer

  now =  start_rt_timer(0); // creation timer de periode  50ms
  curTime = start_rt_timer( nano2count(0));
  now = rt_get_time();
    // Lancement des taches
  rt_task_init(&Tache1_Ptr,    /* Le prt de tache */
          Tache1,      /* Nom de la tache */
          1,     /* valeur du parametre X */
          2000,    /* Taille de la pile necessaire pour la tache (memoire tempo
              utilisee) */
          0,     /* Priorite */
          0,       /* Pas de calcul flottant */
          0);      /* choix d'un signal ou non  */


  rt_task_make_periodic(  &Tache1_Ptr,  /* Pointeur vers la tache */
        now,     /* Instant de depart */
        nano2count(PERIOD*microsec/2));    /* Periode */


  rtf_create(deltat, SIZETAB);
  return 0;
}

/****************** cleanup ************************/
void cleanup_module(void)
{
  stop_rt_timer();
  rt_task_delete(&Tache1_Ptr);
 }
```

Annexe 2 - TITRE