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 refais la fonctionnalité précédente, c'est à dire un générateur de signal carré a fréquence fixe. Cette fréquence doit valoir 100 milisecondes

Chapitre 2

Chapitre 3

Chapitre 4

Conclusion

Annexes

Annexe 1 - TP 1

Code partie 1

```
/* 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 */
```

```c
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

Annexe 2 - TITRE