Tellstick-15.c

Ett program skrivet i C för att med Tellstick tända och släcka en eller flera lampor.

Det nya i detta program är att strömbrytare 1 tänds och släcks efter solens upp- och nedgång, tänds vid solens nedgång och släcks vid solens uppgång.

Kompilering av programmet:

gcc -o tellstick-15 -Wall -ltelldus-core -W -O2 -s -pipe -lm tellstick-15.c

Programmet kommer att skifta loggfil kl 00:01, så kallad loggroll.

Programmet startas med kommandot:

/home/pi/Development/Cprog/Tellstick/tellstick-15 -f </dev/null &>/dev/null &

Avsluta programmet med shellkommandot:

kill -9 PID

där PID, process ID, är det id som sätts på programmet. PID kan hittas med hjälp av

shellkommandot:

ps -ef

Programmet tellstick-15.c

```
/* tellstick-15.c
*  2014-11-28 Jan Pihlgren
*
*  Loggroll körs varje natt klockan 00:01.
*  Programmet justerar sig till att mäta absolut minut. (diff)
*  Programmet utnyttjar en indatafil där tänd och släcktider anges.
*  Brytare nr 1 kommer alltid att tändas vid solens nedgång och släckas vid solens uppgång.
*  PATH/tellstick-13 -f
*
*  Indatafilens organisation:
*  löpnr tändtid släcktid brytarnr/enhetsnr
*  ex:
*  1 15:00 17:00 2   ordningsnr, tändtid, släcktid, brytarenr
*
*  Programmet stoppas med kommando:
```

```
 *  kill -9 PID eller sudo kill -9 PID
 *  där PID hittas med kommando ps -ef och leta efter startkommandot ovan.
 *
 * Tänker införa möjlighet att kommunicera med programmet när det körs.
 *
 *        Kompilera:
 *   (gcc -o tellstick-15 -Wall -ltelldus-core -W -O2 -s -pipe -lm tellstick-15.c)
*/
#include <telldus-core.h>
#include <time.h>
#include <sys/time.h>
#include <math.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include "struktur.h"
#include "sunrise.h"

/*
struct strombr {
        char nr[2];                                     // Ordningsnummer
        char ontid[10];
        char offtid[10];
        char switchnr[2];                       // Brytarens/enhetens nr
};
struct strombr brytare[25];             // Plats för 25 brytare/enheter
*/

char indatafile[] = "/home/pi/Script/tidschema_ny.data";
char logfile[] = "/home/pi/temp/tellstick.log";
int antpost =0;
int postnr;
char typ[2]=" ";

void loggroll();
void readdata_ny();
```

```
int sunrise( int year, int month, int day);

int main(int argc, char *argv[]) {

   time_t tid;
   struct tm *info;
   char buffer[80];
   char sec[80];
   char klockan[10];
   int i;
   int go = 1; // fortsätt programmet
   int diff;       // skillnads
   char version[] = "Version: tellstick-15";
   char rolltid[] = "00:01";
   int year=2014;
   int month=12;
   int day=1;

         time(&tid);
         info = localtime(&tid);
         strftime(buffer,80,"%Y",info);
         year=atoi(buffer);
         strftime(buffer,80,"%m",info);
         month=atoi(buffer);
         strftime(buffer,80,"%d",info);
         day=atoi(buffer);

         loggroll();                                   /* Rulla loggfilerna */
         readdata_ny();                                /* läs in data för tändtider och släcktider i struct
brytare */
         sunrise(year,month,day);       /* Sätt tiden för solens upp- och nedgång för brytare 0 */
         printf("Datum = %d-%d-%d:\n",year,month,day);
         printf("brytare 0,Tändtid = %s  Släcktid = %s\n",brytare[0].ontid,brytare[0].offtid);



         FILE *fp;
         if (argc > 1) {
```

```
                if (strcmp(argv[1], "-f") == 0) {                       /* Läsa från datafil om tänd- och
släcktider ? */
                                /* Hämtas data från fil */
                                postnr = 0;                                                     /* Första
tändtiden, post nr 1 */
                }
        }
//  else{
                // strcmp ?
//}
        fp=fopen(logfile, "a");
        fprintf(fp,"%s\n",version);
        fprintf(fp,"brytare %s\n ",brytare[postnr].switchnr);
        fprintf(fp,"Tändning sker klockan %s\n",brytare[postnr].ontid);
        fprintf(fp, "Släckning sker klockan %s\n",brytare[postnr].offtid);
        fclose(fp);

        while(go != 0){
                time(&tid);
                info = localtime(&tid);
                strftime(buffer,80,"%H:%M",info);
                strcpy(klockan,buffer);
                strftime(sec,80,"%H:%M:%S",info);
                strftime(buffer,80,"%Y",info);
                year=atoi(buffer);
                strftime(buffer,80,"%m",info);
                month=atoi(buffer);
                strftime(buffer,80,"%d",info);
                day=atoi(buffer);

                sunrise(year,month,day);        /* Sätt tiden för solens upp- och nedgång för brytare 0
*/
//              printf("Datum = %d-%d-%d:\n",year,month,day);
//              printf("brytare 0,Tändtid = %s  Släcktid = %s\n",brytare[0].ontid,brytare[0].offtid);

                fp=fopen(logfile, "a");
                for(i = 0; i < antpost; i++){
                        if(strcmp(klockan,brytare[i].ontid) == 0){
```

```
                                tdTurnOn(atoi(brytare[i].switchnr));
                                fprintf(fp,"Tänder %s Kl %s\n",brytare[i].switchnr,klockan);
                        }
                        if(strcmp(klockan,brytare[i].offtid) == 0){
                                tdTurnOff(atoi(brytare[i].switchnr));
                                fprintf(fp,"Släcker %s Kl %s\n",brytare[i].switchnr,klockan);
                        }
                }
                fclose(fp);

                if(strcmp(rolltid,klockan) == 0){
                        loggroll();                                             // Rulla
loggfilerna
                        readdata_ny();                                          /* läs in
data igen */
                        sunrise(year,month,day);                        /* Återställ brytare 0 till
solens upp och nergång */

                        fp=fopen(logfile, "a");
                        fprintf(fp,"%s\n",version);
                        if(strcmp(argv[1],"-f") == 0){
                                fprintf(fp,"Ant poster = %d\n",antpost);
                                fprintf(fp,"Tändning kl %s. Släckning kl
%s\n",brytare[0].ontid,brytare[0].offtid);
                        }
                        fclose(fp);
                }
   /* Här ska programmet kontrollera om det finns något inkommande kommando som ska åtgärdas
*/
                diff = 60 - atoi(sec);    // Rucka klockan så tiden blir exact på minuten.
                sleep(diff);
//              go=0;
        }
   tdClose();

        return 0;
}
```

```
void loggroll(){

	char oldfile1[] = "/home/pi/temp/tellstick.old1";
	char oldfile2[] = "/home/pi/temp/tellstick.old2";
	char oldfile3[] = "/home/pi/temp/tellstick.old3";

	/*  Radera oldfile3 */
		remove(oldfile3);

	/* Flytta filer */
		rename(oldfile2, oldfile3);  // flytta oldfile2 till oldfile3
		rename(oldfile1, oldfile2);  // flytta oldfile1 till oldfile2
		rename(logfile, oldfile1);       // flytta logfile till oldfile1
	return;
}

void readdata_ny(){
	FILE *data;

	data = fopen(indatafile,"r");
	antpost=0;
	while(1){
		fscanf(data,"%s",brytare[antpost].nr);
		fscanf(data,"%s",brytare[antpost].ontid);
		fscanf(data,"%s",brytare[antpost].offtid);
		fscanf(data,"%s",brytare[antpost].switchnr);
		antpost++;
		if(feof(data)){
			antpost--;
			break;
		}
	}
	fclose(data);
	return;
}
```

strucktur.h

```
/* header för strukturen brytare */
struct strombr {
        char nr[2];                                     // Ordningsnummer
        char ontid[10];
        char offtid[10];
        char switchnr[2];                       // Brytarens/enhetens nr
};
struct strombr brytare[25];             // Plats för 25 brytare/enheter
```

sunrice.h

```
/*
*       sunrise.h
*
*       Copyright (GPL) 2004   Mike Chirico mchirico@comcast.net
*       Updated: Sun Nov 28 15:15:05 EST 2004
*
*       Program adapted by Mike Chirico mchirico@comcast.net
*
*       Reference:
*   http://prdownloads.sourceforge.net/souptonuts/working_with_time.tar.gz?download
*   http://www.srrb.noaa.gov/highlights/sunrise/sunrise.html
*
*       Modified: 2014-11-28 Jan Pihlgren
*
*       Position för MÄRSTA:
*   59 grader 37 minuter 0 sekunder N , 17 grader 51 minuter 0 sekunder E
*   59.62, -17.85
*       Kompilera:
*       (gcc -L/usr/local/lib -Wall -W -lm -o sunrise sunrise.c)
*   (gcc -o sunrise -Wall -W -O2 -s -pipe -lm sunrise.c)
*
*   Exekvera för Märsta:
```

```
*   ./sunrise 2014 11 24 59.62 -17.85
*/

double calcSunEqOfCenter(double t);

/* Convert degree angle to radians */

double  degToRad(double angleDeg)
{
 return (M_PI * angleDeg / 180.0);
}

double radToDeg(double angleRad)
{
 return (180.0 * angleRad / M_PI);
}

double calcMeanObliquityOfEcliptic(double t)
{
 double seconds = 21.448 - t*(46.8150 + t*(0.00059 - t*(0.001813)));
 double e0 = 23.0 + (26.0 + (seconds/60.0))/60.0;

 return e0;              // in degrees
}

double calcGeomMeanLongSun(double t)
{


 double L = 280.46646 + t * (36000.76983 + 0.0003032 * t);
 while( (int) L >  360 )
   {
     L -= 360.0;

   }
 while(  L <  0)
   {
```

```
    L += 360.0;

   }


 return L;              // in degrees
}

double calcObliquityCorrection(double t)
{
 double e0 = calcMeanObliquityOfEcliptic(t);


 double omega = 125.04 - 1934.136 * t;
 double e = e0 + 0.00256 * cos(degToRad(omega));
 return e;              // in degrees
}

double calcEccentricityEarthOrbit(double t)
{
 double e = 0.016708634 - t * (0.000042037 + 0.0000001267 * t);
 return e;              // unitless
}

double calcGeomMeanAnomalySun(double t)
{
 double M = 357.52911 + t * (35999.05029 - 0.0001537 * t);
 return M;               // in degrees
}


double calcEquationOfTime(double t)
{


 double epsilon = calcObliquityCorrection(t);
 double  l0 = calcGeomMeanLongSun(t);
```

```
 double e = calcEccentricityEarthOrbit(t);
 double m = calcGeomMeanAnomalySun(t);
 double y = tan(degToRad(epsilon)/2.0);
 y *= y;
 double sin2l0 = sin(2.0 * degToRad(l0));
 double sinm   = sin(degToRad(m));
 double cos2l0 = cos(2.0 * degToRad(l0));
 double sin4l0 = sin(4.0 * degToRad(l0));
 double sin2m  = sin(2.0 * degToRad(m));
 double Etime = y * sin2l0 - 2.0 * e * sinm + 4.0 * e * y * sinm * cos2l0
                                    - 0.5 * y * y * sin4l0 - 1.25 * e * e * sin2m;


 return radToDeg(Etime)*4.0;          // in minutes of time


          }

double calcTimeJulianCent(double jd)
{

 double T = ( jd - 2451545.0)/36525.0;
 return T;
}

double calcSunTrueLong(double t)
{
 double l0 = calcGeomMeanLongSun(t);
 double c = calcSunEqOfCenter(t);

 double O = l0 + c;
 return O;              // in degrees
}

double calcSunApparentLong(double t)
{
 double o = calcSunTrueLong(t);
```

```
 double  omega = 125.04 - 1934.136 * t;
 double  lambda = o - 0.00569 - 0.00478 * sin(degToRad(omega));
 return lambda;          // in degrees
}

double calcSunDeclination(double t)
{
 double e = calcObliquityCorrection(t);
 double lambda = calcSunApparentLong(t);

 double sint = sin(degToRad(e)) * sin(degToRad(lambda));
 double theta = radToDeg(asin(sint));
 return theta;          // in degrees
}

double calcHourAngleSunrise(double lat, double solarDec)
{
 double latRad = degToRad(lat);
 double sdRad  = degToRad(solarDec);



 double HA = (acos(cos(degToRad(90.833))/(cos(latRad)*cos(sdRad))-tan(latRad) * tan(sdRad)));

 return HA;              // in radians
}

double calcHourAngleSunset(double lat, double solarDec)
{
 double latRad = degToRad(lat);
 double sdRad  = degToRad(solarDec);



 double HA = (acos(cos(degToRad(90.833))/(cos(latRad)*cos(sdRad))-tan(latRad) * tan(sdRad)));

 return -HA;             // in radians
```

```
}

double calcJD(int year,int month,int day)
	{
		if (month <= 2) {
			year -= 1;
			month += 12;
		}
		int A = floor(year/100);
		int B = 2 - A + floor(A/4);

		double JD = floor(365.25*(year + 4716)) + floor(30.6001*(month+1)) + day + B - 1524.5;
		return JD;
	}

double calcJDFromJulianCent(double t)
{
 double JD = t * 36525.0 + 2451545.0;
 return JD;
}

double calcSunEqOfCenter(double t)
{
		double m = calcGeomMeanAnomalySun(t);

		double mrad = degToRad(m);
		double sinm = sin(mrad);
		double sin2m = sin(mrad+mrad);
		double sin3m = sin(mrad+mrad+mrad);

		double C = sinm * (1.914602 - t * (0.004817 + 0.000014 * t)) + sin2m * (0.019993 - 0.000101 * t) + sin3m * 0.000289;
		return C;		// in degrees
}

double calcSunriseUTC(double JD, double latitude, double longitude)
```

```
{

        double t = calcTimeJulianCent(JD);

                // *** First pass to approximate sunrise


        double  eqTime = calcEquationOfTime(t);
        double  solarDec = calcSunDeclination(t);
        double  hourAngle = calcHourAngleSunrise(latitude, solarDec);
      double  delta = longitude - radToDeg(hourAngle);
        double  timeDiff = 4 * delta;  // in minutes of time
        double  timeUTC = 720 + timeDiff - eqTime;          // in minutes
      double  newt = calcTimeJulianCent(calcJDFromJulianCent(t) + timeUTC/1440.0);


       eqTime = calcEquationOfTime(newt);
       solarDec = calcSunDeclination(newt);


                hourAngle = calcHourAngleSunrise(latitude, solarDec);
                delta = longitude - radToDeg(hourAngle);
                timeDiff = 4 * delta;
                timeUTC = 720 + timeDiff - eqTime; // in minutes



                return timeUTC;
        }

double calcSunsetUTC(double JD, double latitude, double longitude)
 {

        double t = calcTimeJulianCent(JD);

                // *** First pass to approximate sunset
```

```
        double  eqTime = calcEquationOfTime(t);
        double  solarDec = calcSunDeclination(t);
        double  hourAngle = calcHourAngleSunset(latitude, solarDec);
   double  delta = longitude - radToDeg(hourAngle);
        double  timeDiff = 4 * delta;  // in minutes of time
        double  timeUTC = 720 + timeDiff - eqTime;          // in minutes
   double  newt = calcTimeJulianCent(calcJDFromJulianCent(t) + timeUTC/1440.0);



      eqTime = calcEquationOfTime(newt);
      solarDec = calcSunDeclination(newt);



                  hourAngle = calcHourAngleSunset(latitude, solarDec);
                  delta = longitude - radToDeg(hourAngle);
                  timeDiff = 4 * delta;
                  timeUTC = 720 + timeDiff - eqTime; // in minutes

                  // printf("************* eqTime = %f  \nsolarDec = %f \ntimeUTC =
%f\n\n",eqTime,solarDec,timeUTC);



                  return timeUTC;
        }



int sunrise(int year, int month, int day){
        time_t seconds;
        time_t tseconds;
        struct tm *ptm=NULL;
        struct tm tm;

        int dst=-1;
        char buffer[30];

        float JD=calcJD(year,month,day);
```

```
        double latitude = 59.62;        // Minus för söder om ekvatorn, för Märsta
        double longitude = -17.85;    // Minus för öster om Greenwich, för Märsta

   JD = calcJD(year,month,day);

        tm.tm_year= year-1900;
        tm.tm_mon=month-1;  /* Jan = 0, Feb = 1,.. Dec = 11 */
        tm.tm_mday=day;
        tm.tm_hour=0;
        tm.tm_min=0;
        tm.tm_sec=0;
        tm.tm_isdst=-1;

        seconds = mktime(&tm);
        int delta;
        dst =tm.tm_isdst;

        ptm = gmtime ( &seconds );
        delta = ptm->tm_hour;
        tseconds = seconds;

        seconds= seconds + calcSunriseUTC( JD,  latitude,  longitude)*60;
        seconds= seconds - delta*3600;

        strftime(buffer,30,"%m-%d-%Y  %T",localtime(&seconds));
//      printf("Sunrise  %s   ",buffer);
        char rise[35];
        strftime(rise,30,"%H:%M",localtime(&seconds));

        seconds=tseconds;
        seconds+=calcSunsetUTC( JD,  latitude,  longitude)*60;
        seconds= seconds - delta*3600;

        strftime(buffer,30,"%m-%d-%Y  %T",localtime(&seconds));
//      printf("Sunset %s\n",buffer);
        char set[35];
        strftime(set,30,"%H:%M",localtime(&seconds));
```

```
// ------------------------------------
//          printf("Sunset %s  ",set);
//          printf("Sunrise  %s\n",rise);
            strcpy(brytare[0].ontid,set);
            strcpy(brytare[0].offtid,rise);
            return 0;
}
```

Kompilering

gcc -o tellstick-15 -Wall -ltelldus-core -W -O2 -s -pipe -lm tellstick-15.c

tidschema_ny.data

1 04:30 08:11 1

2 05:00 06:00 2

4 16:15 17:05 2