

Orononne.php

<?

// Envoi d'un message au Lego MindStorm

// Les constantes

define('COMMANDE','/usr/local/bin/controle -c');

define('BOITE_ORDRE',3);

define('BOITE_REPONSE',4);

define('BOITE_MOTEUR1',5);

define('BOITE_MOTEUR2',6);

define('COMMANDE_MOTEURS',5);

// Recuperation de parametres

\$cmd=\$_REQUEST['cmd'];

if(!isset(\$cmd)) die('Pas de commande !');

// Utilisation de l'executable

if(\$cmd==COMMANDE_MOTEURS){

 \$m1=\$_REQUEST['m1'];

 \$m2=\$_REQUEST['m2'];

 \$sortie=shell_exec(COMMANDE.' -- '.BOITE_MOTEUR1." \$m1");

 \$sortie=shell_exec(COMMANDE.' -- '.BOITE_MOTEUR2." \$m2");

 }

\$sortie=shell_exec(COMMANDE.' '." \$cmd ".BOITE_ORDRE.BOITE_REPONSE);

echo trim(\$sortie);

?>

Controle.c

```

/*****
/*   Executable permettant de recuperer des cliches de la webcam   */
*****/

/** Fichiers d'inclusion **/

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <getopt.h>
#include <signal.h>
#include <sys/stat.h>
#include <math.h>
#include <libusb-1.0/libusb.h>
#include <termios.h>
#include <sys/types.h>

#include "../Socket/libsck.h"

/** Constantes **/

#define CHEMIN_SOCKET          "/tmp/webcontrol"
#define MAX_CONNEXIONS        10
#define MAX_ATTENTE            1000

#define MAX_NOMBRE              128
#define MAX_CHAINE              1024

#define MODE_SERVEUR            0
#define MODE_CLIENT             1

#define DEVICE "/dev/ttyACM0"
#define SPEED      B9600

/** Fonctions de communication par liaison série   **/
```

```

void envoyerInstruction(int fd,unsigned char instruction){
write(fd,&instruction,sizeof(instruction));
}

```

```

unsigned char recevoirReponse(int fd){
unsigned char answer;
int nb_bytes=read(fd,&answer,sizeof(buf));
if (nb_bytes!=1)
{
    perror("read_file");
    exit(-1);
}
return answer;
}

```

/** Affichage de la syntaxe **/

```

void usage(char *nom){
fprintf(stderr,"Usage:\n");
fprintf(stderr,"  %s -s <serial device>\n",nom);
fprintf(stderr,"  %s -c [-a] <integer>\n",nom);
exit(-1);
}

```

// Serial port initialisation

```

int init_serial(char *device,int speed){
    struct termios new;
    int fd=open(device,O_RDWR|O_NOCTTY);
    if(fd<0){perror(device); exit(-1);}
    tcgetattr(fd,&saveterm); /* save current port settings */
    bzero(&new,sizeof(new));
    new.c_cflag=CLOCAL|CREAD|speed|CS8;
    new.c_iflag=0;
    new.c_oflag=0;
    new.c_lflag=0; /* set input mode (non-canonical, no echo,...) */
    new.c_cc[VTIME]=0; /* inter-character timer unused */
    new.c_cc[VMIN]=1; /* blocking read until 1 char received */
    tcflush(fd, TCIFLUSH);
    tcsetattr(fd,TCSANOW,&new);
    return fd;
}

```

```

// Arrêter la communication liaison série
void close_serial(int fd)
{
    tcsetattr(fd, TCSANOW, &saveterm);
    close(fd);
}

/** Procedure principale **/

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

    // Recuperation des parametres
    int mode=-1;
    int answer=0;
    int flag;
    static struct option long_options[]={
        {"server", 0, 0, 's'},
        {"client", 0, 0, 'c'},
        {0, 0, 0, 0}
    };
    while(1){
        flag=getopt_long(argc, argv, "sc", long_options, NULL);
        if(flag==-1) break;
        switch(flag){
            case 's': mode=MODE_SERVEUR; break;
            case 'c': mode=MODE_CLIENT; break;
            case 'a': answer=1; break;
            default: usage(argv[0]); break;
        }
    }
    if(mode<0) usage(argv[0]);
    if(mode==MODE_SERVEUR && answer==1) usage(argv[0]);
    char *device;
    if(mode==MODE_SERVEUR){
        device=argv[optind];
        if(argc!=optind+1) usage(argv[0]);
    }
    if(mode==MODE_CLIENT){
        if(argc!=optind+1) usage(argv[0]);
        instruction=atoi(argv[optind]);
    }
}

```

```

#ifdef DEBUG
if(mode==MODE_SERVEUR){
    fprintf(stdout,"Serial device '%s'\n",DEVICE);
}
if(mode==MODE_CLIENT){
    fprintf(stdout,"Instruction to send %c\n",instruction);
    fprintf(stdout,"Must read reponse '%s'\n",answer==1?"yes":"no");
}
#endif

// Gestion des signaux de terminaison
// signal(SIGINT,close_serial);

// Mode Client
if(mode==MODE_CLIENT){
    char message[MAX_CHAINE];
    int ds=connexionServeur(CHEMIN_SOCKET);
    sprintf(message,"%c %c\n",instruction,answer==1?'1':'0');
    int taille=strlen(message);
    if(write(ds,message,taille)==taille){
        taille=read(ds,message,MAX_CHAINE);
        message[taille]='\0';
        if(taille>0) printf("%s\n",message);
    }
}

// Mode serveur
if(mode==MODE_SERVEUR){

    // Initialise la connexion liaison série
    int fd=init_serial(DEVICE,SPEED);
#ifdef DEBUG
    //fprintf(stdout,"Liaison série connected to %s \n", address);
#endif

    // Lecture des commandes et envoi des commandes liaison série
    unlink(CHEMIN_SOCKET);
    int ds=initialisationServeur(CHEMIN_SOCKET,MAX_CONNEXIONS);
    chmod(CHEMIN_SOCKET,0777);
    while(1){
        char message[MAX_CHAINE];
        unsigned char instruction;
        unsigned char wait;
        int dialogue=attenteClient(ds,MAX_ATTENTE);

```

```

if(dialogue<0) continue;
FILE *flux=fdopen(dialogue,"a+");
if(fgets(message,MAX_CHAINE,flux)==NULL)
    { perror("main.fgets"); exit(-1); }
if(sscanf(message,"%c %c",&instruction,&wait)!=3) break;
envoyerInstruction(fd,instruction);
if(wait=='1'){
    unsigned char result=recevoirReponse(fd);
    sprintf(message,"%s\n",result);
}
else strcpy(message,"");
fprintf(flux,message);
fclose(flux);
}
close_serial(fd);
}
return 0;
}

```