```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <termios.h>
#include <libwebsockets.h>
#include "serial.h"
#include "alphanumeric.h"

#define MAX_FRAME_SIZE  1024
#define WAIT_DELAY      50

#define         SERIAL_DEVICE           "/dev/ttyUSB0"

//FONCTION DE CONVERSION D'UNE CARACTERE EN HEXA
 short int conversionHexa(char character)
 {
        switch (character)
        {
                case '0':
                case 0:
                        return ZERO;
                        break;
                case '1':
                case 1:
                        return ONE;
                        break;
                case '2':
                case 2:
                        return TWO;
                        break;
                case '3':
                case 3:
                        return THREE;
                        break;
                case '4':
                case 4:
                        return FOUR;
                        break;
                case '5':
                case 5:
                        return FIVE;
                        break;
                case '6':
                case 6:
                        return SIX;
                        break;
                case '7':
                case 7:
                        return SEVEN;
                        break;
                case '8':
```

```
case 8:
            return EIGHT;
            break;
case '9':
case 9:
            return NINE;
            break;
case 'A':
case 'a':
case 10:
            return A_CHAR;
            break;
case 'B':
case 'b':
case 11:
  return B_CHAR;
            break;
case 'C':
case 'c':
case 12:
            return C_CHAR;
            break;
case 'D':
case 'd':
case 13:
            return D_CHAR;
            break;
case 'E':
case 'e':
case 14:
            return E_CHAR;
            break;
case 'F':
case 'f':
case 15:
            return F_CHAR;
            break;
case 'G':
case 'g':
            return G_CHAR;
            break;
case 'H':
case 'h':
            return H_CHAR;
            break;
case 'I':
case 'i':
            return I_CHAR;
            break;
case 'J':
case 'j':
            return J_CHAR;
		break;
	case 'K':
	case 'k':
		return K_CHAR;
		break;
	case 'L':
	case 'l':
		return L_CHAR;
		break;
	case 'M':
	case 'm':
		return M_CHAR;
		break;
	case 'N':
	case 'n':
		return N_CHAR;
		break;
	case 'O':
	case 'o':
		return O_CHAR;
		break;
	case 'P':
	case 'p':
		return P_CHAR;
		break;
	case 'Q':
	case 'q':
		return Q_CHAR;
		break;
	case 'R':
	case 'r':
		return R_CHAR;
		break;
	case 'S':
	case 's':
		return S_CHAR;
		break;
	case 'T':
	case 't':
		return T_CHAR;
		break;
	case 'U':
	case 'u':
		return U_CHAR;
		break;
	case 'V':
	case 'v':
		return V_CHAR;
		break;
	case 'W':
	case 'w':
		return W_CHAR;
```

```
			break;
		case 'X':
		case 'x':
			return X_CHAR;
			break;
		case 'Y':
		case 'y':
			return Y_CHAR;
			break;
		case 'Z':
		case 'z':
			return Z_CHAR;
			break;
		case ' ':
			return 0;
			break;
		case '!':
			return EXCLAMATION;
			break;
		case '#':
			return (uint16_t) 0xFFFF;
			break;
		case '$':
			return DOLLAR;
			break;
		case '%':
			return PERCENT;
			break;
		case '^':
			return CARROT;
			break;
		case '&':
			return AMPERSAND;
			break;
		case '*':
			return ASTERISK;
			break;
		case '(':
			return LPAREN;
			break;
		case ')':
			return RPAREN;
			break;
		case '-':
			return MINUS;
			break;
		case '_':
			return UNDERSCORE;
			break;
		case '+':
			return PLUSYSIGN;
			break;
```

```
case '=':
          return EQUALS;
          break;
case '>':
          return RARROW;
          break;
case '<':
          return LARROW;
          break;
case ',':
          return COMMA;
          break;
case '/':
          return FSLASH;
          break;
case '\\':
          return BSLASH;
          break;
case '\'':
          return SINGLEQUOTE;
          break;
case '"':
          return DOUBLEQUOTE;
          break;
case 0X5B:
          return LBRACKET;
          break;
case 0X5D:
          return RIBRACKET;
          break;
case 0X7D:
          return LECURLY;
          break;
case 0X7B:
          return RICURLY;
          break;
case '|':
          return PIPE;
          break;
case '~':
          return TILDE;
          break;
case '`':
          return APOSTROPHE;
          break;
case '@':
          return ATSIGN;
          break;
case '?':
          return QUESTIONMARK;
          break;
case ':':
```

```
                        return COLON;
                        break;
                case ';':
                        return SEMICOLON;
                        break;
                case '.':
                        return PERIOD;
                        break;
        }
}
 //FIN FONCTION

int sd;
static int callback_http(
  struct libwebsocket_context *this,
  struct libwebsocket *wsi,enum libwebsocket_callback_reasons reason,
  void *user,void *in,size_t len)
{
return 0;
}

static int callback_my(
  struct libwebsocket_context * this,
  struct libwebsocket *wsi,enum libwebsocket_callback_reasons reason,
  void *user,void *in,size_t len)
{
int length;
int j,i=0;
char c, s,d;
switch(reason){
  case LWS_CALLBACK_ESTABLISHED:
    printf("connection established\n");
            // Declenchement d'un prochain envoi au navigateur
    //libwebsocket_callback_on_writable(this,wsi);
    break;
  case LWS_CALLBACK_RECEIVE:
            // Ici sont traites les messages envoyes par le navigateur
    printf("received data: %s\n",(char *)in);
//traitement du message reçu
   length = strlen(in);
   for(i=length-1; i>=0; i--)
   {
        c= ((char *)in)[i];
        //conversion en hexa
        short int code = conversionHexa(c);
        unsigned char* p =(unsigned char*)&code;
        //fin conversion
        //envoi du code hexa au port serie
        if(write(sd, &p[0], sizeof(char))!=1){perror("main.write");exit(-1); }
        if(write(sd, &p[1], sizeof(char))!=1){perror("main.write");exit(-1); }
   }
```

```
    /*fin recuperation */
                // Declenchement d'un prochain envoi au navigateur
    //libwebsocket_callback_on_writable(this,wsi);
    break;
  case LWS_CALLBACK_SERVER_WRITEABLE:
                // Ici sont envoyes les messages au navigateur
    break;
  default:
    break;
  }
return 0;
}

static struct libwebsocket_protocols protocols[] = {
  {
    "http-only",   // name
    callback_http, // callback
    0,             // data size
    0              // maximum frame size
  },
  {"myprotocol",callback_my,0,MAX_FRAME_SIZE},
  {NULL,NULL,0,0}
  };



//MAIN
int main(void) {
int port=9000;
//OUVERTURE ET CONFIG DE LAISON SERIE

sd=serialOpen(SERIAL_DEVICE,SERIAL_BOTH);
serialConfig(sd,B9600);

struct lws_context_creation_info info;
memset(&info,0,sizeof info);
info.port=port;
info.protocols=protocols;
info.gid=-1;
info.uid=-1;
struct libwebsocket_context *context=libwebsocket_create_context(&info);
if(context==NULL){
  fprintf(stderr, "libwebsocket init failed\n");
  return -1;
  }
printf("starting server...\n");
while(1){
  libwebsocket_service(context,WAIT_DELAY);
  }
libwebsocket_context_destroy(context);
return 0;
}
```