Apéndice C.

Código fuente

de la aplicación principal.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <syslog.h>

#include <sys/socket.h>
#include <sys/stat.h>
#include <sys/types.h>

#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>
#include <bluetooth/hci_lib.h>
#include <bluetooth/rfcomm.h>
#include <bluetooth/sdp.h>
#include <bluetooth/sdp_lib.h>

#define ERROR_LECTURA -1
#define BDADDR_DONGLE "00:02:5B:0A:6E:52"
typedef struct param
{
	int dev_id;
	int dd;
	char bt_address[19];
	uint8_t *lap;
	uint8_t len_inq;
	uint8_t max_rsp;
	uint8_t tipo_inq;
	uint8_t modo_inq;
	uint8_t flags;
}PARAM;

void change_iac(int dd,uint8_t lap[3],uint8_t tipo_inquiry,uint8_t modo_inquiry);
void init(PARAM *arg, int dev_id, int dd);
int leer_cmd(int s, char *buffer);

int main(int argc,char *argv[])
{
	int dev_id, dd;
	PARAM parametros;
	char bd_address[18]={0};
	char buffer[128]={0};
	char buffer_fichero[128]={0};
	int sock[5];
	int num_rsp=0;
	int i=0;
	int contador=0;
	int estado=0;
	int bytes_write=0;
	int salir_lectura=0;
	FILE *fd=NULL;
	char *cmd[]={"START\r\n","OK\r\n","REPITE\r\n"};
	char *nodos[]={"00:07:80:90:C8:1E","00:07:80:93:9A:FF","00:07:80:93:9A:FE","00:07:80:90:C8:1A"};

	inquiry_info *ii=NULL;
	struct sockaddr_rc addr;

	if(argc!=2)
	{
		printf("USO: server + nombre_fichero\n");
		exit(0);
	}
```

```
	dev_id=hci_devid(BDADDR_DONGLE);
	dd=hci_open_dev(dev_id);
	if(dev_id < 0 || dd < 0){
		printf("Error: no se ha podido abrir el dispositivo\n");
		exit(1);
	}

	init(&parametros,dev_id,dd);
	change_iac(dd,parametros.lap,parametros.tipo_inq,parametros.modo_inq);

	while(1){
		ii = (inquiry_info*)malloc(parametros.max_rsp * sizeof(inquiry_info));
#ifdef _DEBUG_
		printf("Dev_id:0x%2.2x Length:0x%2.2x Max_rsp:0x%2.2x
Lap:0x%2.2x%2.2x%2.2x\n",parametros.dev_id,parametros.len_inq,parametros.max_rsp,parametros.lap[2],parame
tros.lap[1],parametros.lap[0]);
#endif
		num_rsp = hci_inquiry(parametros.dev_id, parametros.len_inq, parametros.max_rsp,
parametros.lap, &ii, parametros.flags);
		if( num_rsp < 0 ){
			printf("Error: no hay dispositivos encontrados\n");
			//exit(0);
		}
#ifdef _DEBUG_
		for(i=0;i<num_rsp;i++){
			ba2str(&(ii+i)->bdaddr,bd_address);
			printf("[%d]\t%s %d %d %d %d %d %d %d\n",i+1,bd_address,(ii+i)-
>pscan_rep_mode,(ii+i)->pscan_period_mode,(ii+i)->pscan_mode,(ii+i)->dev_class[0],(ii+i)->dev_class[1],(ii+i)-
>dev_class[2],(ii+i)->clock_offset);
			memset(bd_address,0,sizeof(bd_address));
		}
#endif
		for(i=0;i<num_rsp;i++){
			sock[i] = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);
			addr.rc_family = AF_BLUETOOTH;
			addr.rc_channel = (uint8_t) 1;
			addr.rc_bdaddr=(ii+i)->bdaddr;
			estado = connect(sock[i], (struct sockaddr *)&addr, sizeof(addr));
			if(estado<0 || estado>0){
				perror("Error: No se ha podido establecer conexion\n");
				close(sock[i]);
			}
			else{
				bytes_write=write(sock[i],cmd[0],strlen(cmd[0]));
				if(bytes_write<0){
					while(bytes_write<0&&contador<3){
						bytes_write=write(sock[i],cmd[0],sizeof(cmd[0]));
						contador++;
					}
					contador=0;
					salir_lectura=1;
					printf("Error: No se ha podido enviar %s\n",cmd[0]);
				}
				while(salir_lectura!=1){
					printf("Lectura OK:\n");
					if(leer_cmd(sock[i],buffer)<0)
						salir_lectura=1;
					else{
						if(strstr(buffer,"OK")!=NULL){
							printf("Lectura RSSI:\n");
							memset(buffer,0,sizeof(buffer));
							if(leer_cmd(sock[i],buffer)<0)
```

```
                                                                        salir_lectura=1;
                                                                else{
                                                                        if(strstr(buffer,"RSSI")!=NULL){
                                                                                fd=fopen(argv[1],"a+b");
                                                                                if(fd==NULL){
                                                                                        printf("Error al
abrir el fichero solicitado\n");
                                                                                        exit(1);
                                                                                }
        ba2str(&addr.rc_bdaddr,buffer_fichero);
                                                                                strcat(buffer_fichero," ");
                                                                                strcat(buffer_fichero,buffer);
                                                                                strcat(buffer_fichero,"\n");

        fwrite(buffer_fichero,sizeof(char),strlen(buffer_fichero),fd);

        memset(buffer_fichero,0,sizeof(buffer_fichero));

        memset(buffer,0,sizeof(buffer));
                                                                                fclose(fd);

        bytes_write=write(sock[i],cmd[1],strlen(cmd[1]));
                                                                                if(bytes_write<0){
                                                                                        printf("Error: no se
ha podido enviar %s",cmd[1]);
                                                                                        close(sock[i]);
                                                                                        salir_lectura=1;
                                                                                }
                                                                                else{
                                                                                        printf("Lectura
CIERRE:\n");
        if(leer_cmd(sock[i],buffer)<0)

        salir_lectura=1;
                                                                                        else{

        printf("Cierre\n");

        if(strstr(buffer,"CIERRE")!=NULL){

        memset(buffer,0,sizeof(buffer));

        close(sock[i]);

        salir_lectura=1;
                                                                                                }
                                                                                                else

        memset(buffer,0,sizeof(buffer));
                                                                                        }
                                                                                }
                                                                        }
                                                                }
                                                        }
                                                }
                                        }
                                        salir_lectura=0;
                                }
                        }
                        printf("Liberando memoria\n");
                        bt_free(ii);
```

```
		}

		return 0;
}

/*!	\fn	int leer_cmd(int s, char *buffer)
*	\brief	FunciÃ³n que permite leer datos del mÃ³dulo Bluegiga WT-12
*	\param	int s:	paso por valor del descriptor de ficheros asociado con el socket RFCOMM que soporta la comunicaciÃ³n Bluetooth.
*	\param	char *buffer: paso por referencia de una cadena de caracteres donde se procederÃ¡ a escribir el comando enviado por
*				el nodo de localizaciÃ³n.
*/
int leer_cmd(int s, char *buffer)
{
		int error=0;
		int salir=0;
		int bytes_read=0;
		int i=0;
		char caracter=0x00;

		while(salir!=1){
			bytes_read=recv(s,&caracter,sizeof(caracter),0);
			if(bytes_read<0){
				perror("Error: no se ha podido efectuar la operacion de lectura\n");
				salir=1;
				error=ERROR_LECTURA;
			}
			else{
				buffer[i++]=caracter;
				if(buffer[i-1]=='\r'){
					buffer[i-1]='\0';
					salir=1;
#ifdef _DEBUG_
					printf("Cadena recibida [%s]\n",buffer);
#endif
				}
			}
		}

		return error;
}

void init(PARAM *arg, int dev_id, int dd)
{
		arg->dev_id=dev_id;
		arg->dd=dd;
		arg->lap=(uint8_t *)malloc(3*sizeof(uint8_t));
		arg->lap[0]=0x05;
		arg->lap[1]=0x8b;
		arg->lap[2]=0x9e;
		arg->len_inq=0x05;
		arg->max_rsp=255;
		arg->tipo_inq=0x01;
		arg->modo_inq=0x01;
		arg->flags=IREQ_CACHE_FLUSH;
}

void change_iac(int dd,uint8_t lap[3],uint8_t tipo_inquiry,uint8_t modo_inquiry)
{

		int iac_liac;
		uint8_t iac,type,mode;		uint8_t lap_check[3];
```

```
#ifdef _DEBUG_
		iac_liac = hci_read_current_iac_lap(dd, &iac, lap_check, 1000);
		hci_read_inquiry_mode(dd,&mode,1000);
		hci_read_inquiry_scan_type(dd, &type,1000);

		printf("iac = 0x%2.2x\n",iac);
		printf("lap = 0x%2.2x%2.2x%2.2x\n",lap_check[2],lap_check[1],lap_check[0]);
		printf("Inquiry:\n\tTipo:\t0x%2.2x\n\tModo:\t0x%2.2x\n",type,mode);
#endif
		iac=0x01;

		iac_liac = hci_write_current_iac_lap(dd, iac, lap, 1000);
		if(iac_liac<0)
		{
			printf("Error al modificar el valor del iac\n");
			exit(1);
		}

		if(hci_write_inquiry_scan_type(dd, tipo_inquiry,1000)<0)
		{
			printf("Error: no se ha podido escribir el tipo de inquiry\n");
			exit(1);
		}

		if(hci_write_inquiry_mode(dd, modo_inquiry,1000)<0)
		{
			printf("Error: no se ha podido escribir el modo de inquiry\n");
			exit(1);
		}

#ifdef _DEBUG_
		iac_liac = hci_read_current_iac_lap(dd, &iac, lap_check, 1000);
		hci_read_inquiry_mode(dd,&mode,1000);
		hci_read_inquiry_scan_type(dd, &type,1000);

		printf("iac = 0x%2.2x\n",iac);
		printf("lap = 0x%2.2x%2.2x%2.2x\n",lap_check[2],lap_check[1],lap_check[0]);
		printf("Inquiry:\n\tTipo:\t0x%2.2x\n\tModo:\t0x%2.2x\n",type,mode);
#endif
}
```