Apéndice B.

Código fuente

del nodo de localización

```
#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include <string.h>
#include <avr/interrupt.h>
#include <compat/ina90.h>

#define F_CPU 7372800
#define BAUDRATE 115200
#define RXBUFFER_SIZE_MAX 55
#define TXBUFFER_SIZE_MAX 150
#define INQUIRY_SIZE_MAX 50
#define NUM_DISPOSITIVOS 5

#define INQUIRY 0
#define ESPERA 1
#define PROCESAR 2
#define RESPUESTA_OK 3
#define TRAMA 4
#define RESTAURAR 5
#define IDLE 6
#define ERROR 7
#define PROCESAR_TRAMA_DATOS 8
#define PROCESAR_TRAMA_CONTROL 9
#define CIERRE 11
#define ESPERA_FIN 12
#define ESPERA_CIERRE 13
#define ESPERA_INQUIRY 15

typedef unsigned char uint8;
typedef unsigned int uint16;

void ioinit(void);
void wt12_init(void);
void USART_flush(void);
void USART_init(unsigned int baud);
void USART_transmit(char data);
void timer1Init(void);
void EEPROM_write(unsigned int uiAddress, unsigned int data);
unsigned char EEPROM_read(unsigned int uiAddress);
void delay_ms(double ms);
void delay_us(double us);
void delay_loop_2(unsigned int count);
void delay_loop_1(unsigned int count);
void tx_cad(const char *cad);
void transmite_cmd(const char *cad, uint8 link);
void iniciar_cadena(char *cad,uint8 tam);

/*****************************/
/*            Variables globales*/
/*****************************/
char TXbuffer[TXBUFFER_SIZE_MAX];
char RXbuffer[RXBUFFER_SIZE_MAX];
uint8 RXbuffer_indice;
uint8 estado;
static volatile uint8 salir;
static volatile uint8 salir_inquiry;
static volatile uint8 q;
uint8 fin;
uint8 fin_cierre;
static volatile uint8 num_disp;
char inquiry_partial[NUM_DISPOSITIVOS][INQUIRY_SIZE_MAX];
char direccion_bt[NUM_DISPOSITIVOS][18];
```

```
char rssi[NUM_DISPOSITIVOS][4];
char resp[2];
char trama[50];
const char cmd0[]={"SET CONTROL MUX 1\n"};                          //0
const char cmd1[]={"SET BT LAP 9e8b05"};                            //1
const char cmd2[]={"SET BT PAGEMODE 3 1000 0"};                     //2
const char cmd3[]={"SET BT POWER 3 3 3"};                           //3
const char cmd4[]={"SET CONTROL CONFIG 0 001F"};        //4
const char cmd5[]={"SET CONTROL ECHO 4"};                           //5
const char cmd6[]={"RESET"};                                                                    //6
const char answer0[]={"OK\r"};
const char answer1[]={"INQUIRY 5"};
const char answer2[]={"CIERRE\r"};
uint8 desp_control;
uint8 desp_datos;
uint8 salir_reposo;

/*****************************/
/*          Funcion principal              */
/*****************************/
int main(void)
{
        uint8 bit=0;
        uint8 k=0;
        uint8 contador=0;

        cli();
        ioinit();
        delay_ms(100);
        sei();

        while(1)
        {
                /*transmite_cmd(cmd[1],0xFF);
                delay_ms(5000);*/
                /*PORTB=(1<<PB0);
                delay_ms(1000);
                PORTB=(0<<PB0);
                delay_ms(1000);*/


                switch(estado)
                {
                        case RESTAURAR:
                                for(contador=0;contador<NUM_DISPOSITIVOS;contador++)
                                {
                                        iniciar_cadena(inquiry_partial[contador],INQUIRY_SIZE_MAX);
                                        iniciar_cadena(direccion_bt[contador],18);
                                        iniciar_cadena(rssi[contador],4);
                                }
                                iniciar_cadena(TXbuffer,TXBUFFER_SIZE_MAX);
                                iniciar_cadena(RXbuffer,RXBUFFER_SIZE_MAX);
                                RXbuffer_indice=0;
                                salir=0;
                                fin=0;
                                num_disp=0;
                                desp_control=0;
                                desp_datos=0;
                                salir_reposo=0;
                                salir_inquiry=0;
                                q=0;
                                num_disp=0;
                                estado=IDLE;
```

```
                break;
        case IDLE:
                while(salir_reposo==0 && estado==IDLE)
                {
                        delay_ms(1000);
                        transmite_cmd(cmd2,0xFF);
                };
                salir_reposo=0;
                break;
        case RESPUESTA_OK:
                delay_ms(200);
                transmite_cmd(answer0,0x00);
                estado=INQUIRY;
                break;
        case INQUIRY:
                delay_ms(200);
                transmite_cmd(answer1,0xFF);//INQUIRY 3
                delay_ms(200);
                estado=ESPERA;
                break;
        case ESPERA:
                while(salir==0);
                salir=0;
                estado=ESPERA_INQUIRY;
                break;
        case ESPERA_INQUIRY:
                while(salir_inquiry==0&&q!=num_disp);
                salir_inquiry=0;
                estado=PROCESAR;
                break;
        case PROCESAR:
                for(contador=0;contador<(num_disp);contador++)
                {
                        bit=16;
                        k=0;
                        while(bit<33)

direccion_bt[contador][k++]=inquiry_partial[contador][bit++];
                        direccion_bt[contador][k]=0x00;
                        bit=44;
                        k=0;
                        while(bit<47)
                                rssi[contador][k++]=inquiry_partial[contador][bit++];
                        rssi[contador][k]=0x00;
                }
                switch(num_disp)
                {
                        case 0:
                                resp[0]='0';
                                resp[1]=0x00;
                                break;
                        case 1:
                                resp[0]='1';
                                resp[1]=0x00;
                                break;
                        case 2:
                                resp[0]='2';
                                resp[1]=0x00;
                                break;
                        case 3:
                                resp[0]='3';
                                resp[1]=0x00;
                                break;
```

```
                                        case 4:
                                                resp[0]='4';
                                                resp[1]=0x00;
                                                break;
                                        case 5:
                                                resp[0]='5';
                                                resp[1]=0x00;
                                                break;
                                        default:
                                                break;
                                }
                                estado=TRAMA;
                                break;
                        case TRAMA://f
                                PORTB=(1<<PB0);
                                strcpy(TXbuffer,"RSSI ");
                                strcat(TXbuffer,resp);
                                strcat(TXbuffer," ");
                                contador=0;
                                for(contador=0;contador<num_disp;contador++)
                                {
                                        strcat(TXbuffer,direccion_bt[contador]);
                                        strcat(TXbuffer," ");
                                        strcat(TXbuffer,rssi[contador]);
                                        strcat(TXbuffer," ");
                                }
                                strcat(TXbuffer,"\r");
                                transmite_cmd(TXbuffer,0x00);
                                PORTB=(0<<PB0);
                                estado=ESPERA_FIN;
                                break;
                        case ESPERA_FIN:
                                while(fin==0)
                                {
                                        delay_ms(100);
                                };
                                fin=0;
                                estado=CIERRE;
                                break;
                        case CIERRE:
                                delay_ms(100);
                                transmite_cmd(answer2,0x00);
                                estado=ESPERA_CIERRE;
                                break;
                        case ESPERA_CIERRE:
                                while(fin_cierre==0)
                                {
                                        delay_ms(100);
                                };
                                fin_cierre=0;
                                estado=RESTAURAR;
                                break;

                        default:
                                break;
                }

        }

        return 0;
}
/*****************************/
/*                  INTERRUPCIONES              */
```

```
/***************************/

ISR(USART_RXC_vect)
{
	uint8 j;
	uint8 i;

	cli();
	RXbuffer[RXbuffer_indice]=UDR;
	if(RXbuffer[1]==0xFF&&((RXbuffer[RXbuffer_indice]^RXbuffer[1])==0xFF)&&RXbuffer_indice!=2)
	{
		RXbuffer_indice=0;
		desp_control=RXbuffer[3];
		j=0;
		while(j<desp_control)
		{
			trama[j]=RXbuffer[4+j];
			j++;
		}
		trama[j]=0x00;
		if(strstr(trama,"INQUIRY_PARTIAL")!=NULL)
		{
			strcpy(inquiry_partial[num_disp],trama);
			num_disp++;
			estado=ESPERA;
		}
		else
		{
			if(strstr(trama,"INQUIRY 00:")!=NULL)
			{
				q++;
				estado=ESPERA_INQUIRY;
			}
			else
			{
				if(strstr(trama,"INQUIRY ")!=NULL)
				{
					salir=1;
					estado=ESPERA;
				}
				else
				{
					if((strstr(trama,"NO CARRIER 0 ERROR 0")!=NULL))
					{
						fin_cierre=1;
						estado=ESPERA_CIERRE;
					}
				}
			}
		}
	}
	else
	{
		if(RXbuffer[1]==0x00&&((RXbuffer[RXbuffer_indice]^RXbuffer[1])==0xFF)&&RXbuffer_indice!=2)
		{
			RXbuffer_indice=0;
			desp_datos=RXbuffer[3];

			i=0;
			while(i<desp_datos)
			{
				trama[i]=RXbuffer[4+i];
				i++;
```

```
                        }
                        trama[i]=0x00;
                        if(strstr(trama,"START")!=NULL)
                        {
                                salir_reposo=1;
                                estado=RESPUESTA_OK;
                        }
                        else
                        {
                                if(strstr(trama,"OK")!=NULL)
                                {
                                        fin=1;
                                        estado=ESPERA_FIN;
                                }
                                else
                                {
                                        if(strstr(trama,"REPETIR")!=NULL)
                                        {
                                                estado=TRAMA;
                                        }
                                }
                        }
                }
                else
                        RXbuffer_indice++;
        }

        sei();
}

/*****************************/
/*              Inicialización              */
/*****************************/
/*****************************/
/*                      Puertos             */
/*****************************/
void ioinit(void)
{
        DDRB = (1<<PB1)|(1<<PB0);
        PORTB = (0<<PB1)|(0<<PB0);
        DDRD = (1<<PD6);
        PORTD = (1<<PD6);
        USART_init((unsigned int)((F_CPU/16)/BAUDRATE-1));              // ocillator fq/(16*baud) rate -1
        or f_cpu/16/BAUDRATE-1
        delay_ms(100);
        USART_flush();
        delay_ms(100);
        wt12_init();
        estado=RESTAURAR;
}
/*****************************/
/*              Bluegiga            */
/*****************************/
void wt12_init(void)
{
        delay_ms(200);
        tx_cad(cmd0);
        delay_ms(200);
        transmite_cmd(cmd1,0xFF);
        delay_ms(200);
        transmite_cmd(cmd2,0xFF);
        delay_ms(200);
        transmite_cmd(cmd3,0xFF);
```

```
	delay_ms(200);
	transmite_cmd(cmd4,0xFF);
	delay_ms(200);
	transmite_cmd(cmd5,0xFF);
	delay_ms(200);
//	transmite_cmd("SET",0xFF);
//	delay_ms(2000);
}

void iniciar_cadena(char *cad,uint8 tam)
{
	uint8 byte=0;
	while(byte<tam)
	{
		cad[byte++]=0x00;
	}
}
/*****************************/
/* Funciones de la USART     */
/*****************************/
void USART_flush( void )
{
	uint8 dummy;
	while ( UCSRA & (1<<RXC) ) dummy = UDR;
}

void USART_init(unsigned int baud)
{
	UBRRH = 0x02;
	//Formato de comunicaciones: 8-N-1
	UCSRC = (0<<URSEL)|(0<<USBS)|(1<<UCSZ1)|(1<<UCSZ0);

	//Configuración del baudrate
	UBRRH = (uint8) baud>>8;
	UBRRL = (uint8) baud;

	//Habilitar Recepcion y Transimision
	UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
}

void USART_transmit(char data)
{
	if(data=='\n')
	{
		USART_transmit(0x0D);
	}
	//Esperar a que el buffer de transmision este vacio
	while (((UCSRA)&(1<<UDRE))==0);
	//Transmision del dato
	UDR=data;
}

void tx_cad(const char *cad)
{
	uint8 pos=0;
	while(cad[pos]!=0x00)
		USART_transmit(cad[pos++]);

}

void transmite_cmd(const char *cad,uint8 link)
{
	char outbuf[150]={0};
```

```
//          uint8 *outbuf=NULL;
            uint8 pos=0;
            uint8 len=strlen(cad);
            uint8 m=0;

            outbuf[pos++]=0xBF;
            outbuf[pos++]=link;
            outbuf[pos++]=0x00;
            outbuf[pos++]=len;
            memmove(outbuf+pos,cad,len);
            pos+=len;
            outbuf[pos++]=link^0xFF;

            while(m<pos)
                        USART_transmit(outbuf[m++]);
}

/*****************************/
/*                                 DELAY                 */
/*****************************/
void delay_loop_1(unsigned int count)    /* time delay for us */

{
            __asm__ volatile (
                        "1: dec %0" "\n\t"
                        "brne 1b"
                        : "=r" (count)
                        : "0" (count)
            );
}

void delay_loop_2(unsigned int count)
{
            __asm__ volatile (
                        "1: sbiw %0,1" "\n\t"
                        "brne 1b"
                        : "=w" (count)
                        : "0" (count)
            );
}

void delay_us(double us)
{
            unsigned char ticks;
            double tmp = ((F_CPU) / 3e6) * us;

            if ( tmp < 1.0 )
                        ticks = 1;
            else if ( tmp > 255)
            {
                        delay_ms( us / 1000.0);
                        return;
            }
            else
                        ticks = (uint8)tmp;
            delay_loop_1( ticks );
}


void delay_ms(double ms)
{
            unsigned int ticks;
            double tmp = ((F_CPU) / 4e3) * ms;
```

```
	if ( tmp < 1.0 )
		ticks = 1;
	else if ( tmp > 65535 )
	{
		//	__ticks = requested delay in 1/10 ms
		ticks = (unsigned int) ( ms * 10.0);
		while( ticks )
		{
			// wait 1/10 ms
			delay_loop_2(((F_CPU) / 4e3) / 10);
			ticks --;
		}
		return;
	}
	else
		ticks = (unsigned int)tmp;
	delay_loop_2(ticks);
}
```