Universidad Tecnológica Nacional

Instituto Nacional Superior del Profesorado Técnico

Robótica

Materia:	**Laboratorio de Microprocesadores**
Curso:	**3-631**
Año:	**2017**

Trabajo Práctico Nro. 1

Argumento: ***Proyecto - Estación Meteorológica Inalámbrica.***

Integrantes del Equipo: Caccavallo, Sebastian
Gerbec, Rodrigo
Leuenberger, Leonardo

Responsable del Informe: Caccavallo Sebastian

Profesor: Steiner, Daniel.

Fecha de realización del T.P.	/ /
1ra. Fecha de presentación	/ /
2da. Fecha de presentación	/ /
Calificación	
Fecha de Aprobación	/ /
FIRMA: …….	ACLARACIÓN: ……...

Introducción

Se presentará una estación meteorológica inalámbrica hogareña, la misma consta de dos dispositivos individuales, una estación meteorológica interna que provee de valores dentro del recinto donde se encuentra, y una estación meteorológica externa que provee los valores exteriores, entre ambos dispositivos se realiza la telemetría exterior y se muestra en un display alfanumérico en la estación interior, con la finalidad de observar a primera vista los datos meteorológicos internos y externos.
Esta estación funciona de forma inalámbrica con enlaces de radiofrecuencia y funciona con pilas primarias en el caso exterior y batería recargable en el caso interior.

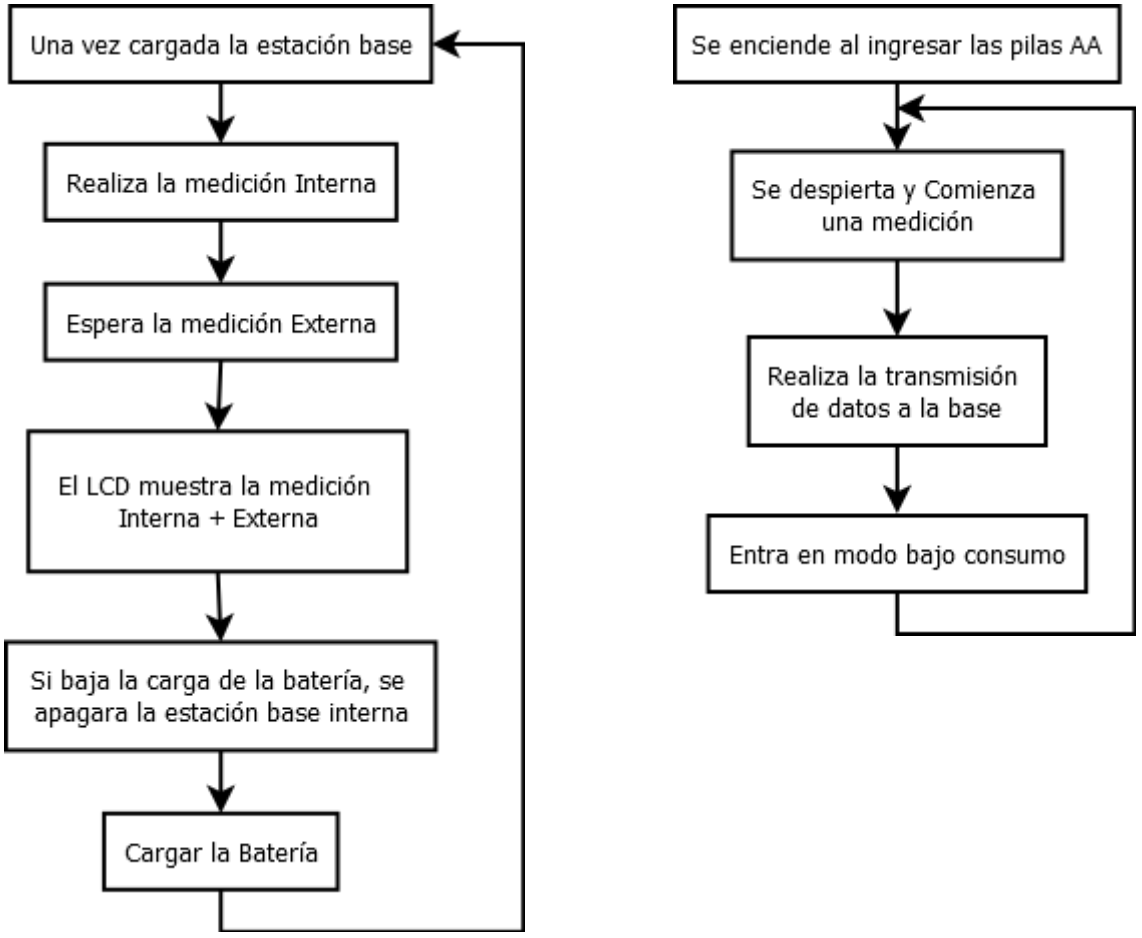
El proyecto surge de la idea de practicar la programación en lenguaje C, los protocolos de comunicación serie y el transporte de los datos vía enlaces inalambricos, en este proyecto se abordará la estructura necesaria para realizar una trama de datos básica, su payload y verificación.
También se realizará la puesta en marcha del sistema mediante electrónica de bajo costo y bajo consumo para afrontar el problema energético que conlleva la utilización de pilas y baterías de forma competitiva con productos comerciales de la actualidad.

Objetivos

- Estación meteorológica con medición interna y externa.
- La estación consiste en un sensor externo y uno interno.
- El sensor externo posee la capacidad de medir Temperatura, Humedad y Luminosidad.
- El sensor interno mide Temperatura y Humedad.
- La telemetría se realiza mediante un enlace inalámbrico simplex UHF, el sensor externo funciona mediante celdas primarias alcalinas y posee electrónica de bajo consumo proporcionando una autonomía de 1 año.
- El sensor interno posee un display de LCD de 2 lineas y 16 caracteres que mostrará la Temperatura y Humedad del sensor interno, la Temperatura, Humedad y Luminosidad del sensor externo.
- El sensor interno cuenta con una batería secundaria de iones de litio recargable mediante un conector micro usb.
- Look&Feel del LCD (VER FIGURA EN EL APARTADO)

Diseño conceptual

Para realizar nuestra estación, utilizaremos microcontroladores modernos de bajo consumo y bajo costo, que permite trabajar con una buena autonomía utilizando pilas primarias para aumentar al máximo el rendimiento del sistema.
La estación meteorológica utilizara la función especial del microcontrolador para entrar en modo bajo consumo y así extender la batería.
Una vez en bajo consumo se despertara de forma automática mediante el WDT del microcontrolador cada 10 segundos, se tomará una muestra de temperatura, humedad y luminosidad, se enviará mediante el enlace de RF a la estación interna y luego se volverá a dormir entrando en bajo consumo.
El enlace se ha simplificado al máximo para reducir el costo, se utiliza modulación ASK en UHF mediante oscilador a cristal SAW de 433.92MHz y se enviaran los datos mediante UART invertida a 600bps.
En la estación interior, se utilizara la interrupción de dato presente en UART para atender la telemetría exterior, realizar el parse de la trama y dejarla disponible para la utilización.
Se tomará humedad y temperatura interior, y junto con la telemetría exterior se mostrará todo en un dsiplay LCD de 2x16 caracteres alfanuméricos.
La estación interna cuenta con baterías recargables y su controlador de carga para ser conectado directamente con un cable estándar micro usb.



Diseño de alto nivel

Diagrama de bloques de Hardware Interior:

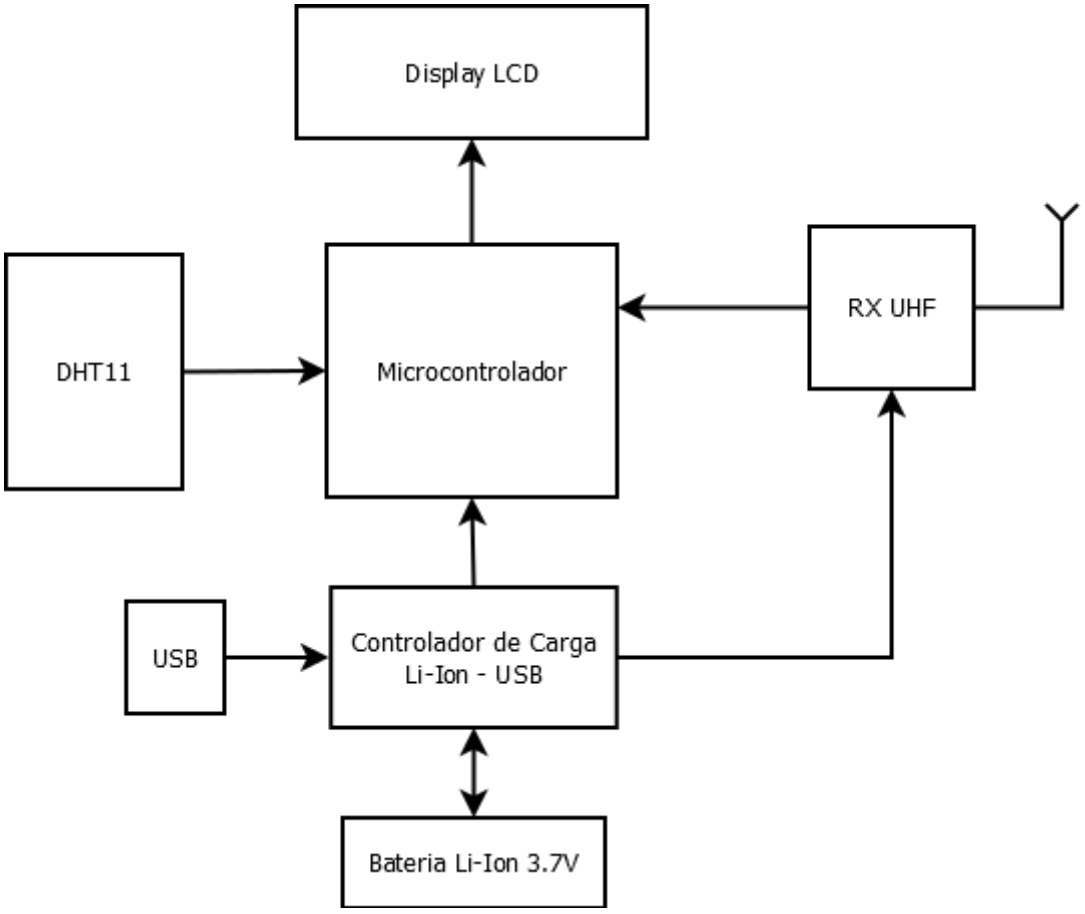


Diagrama de bloques de Hardware Exterior:

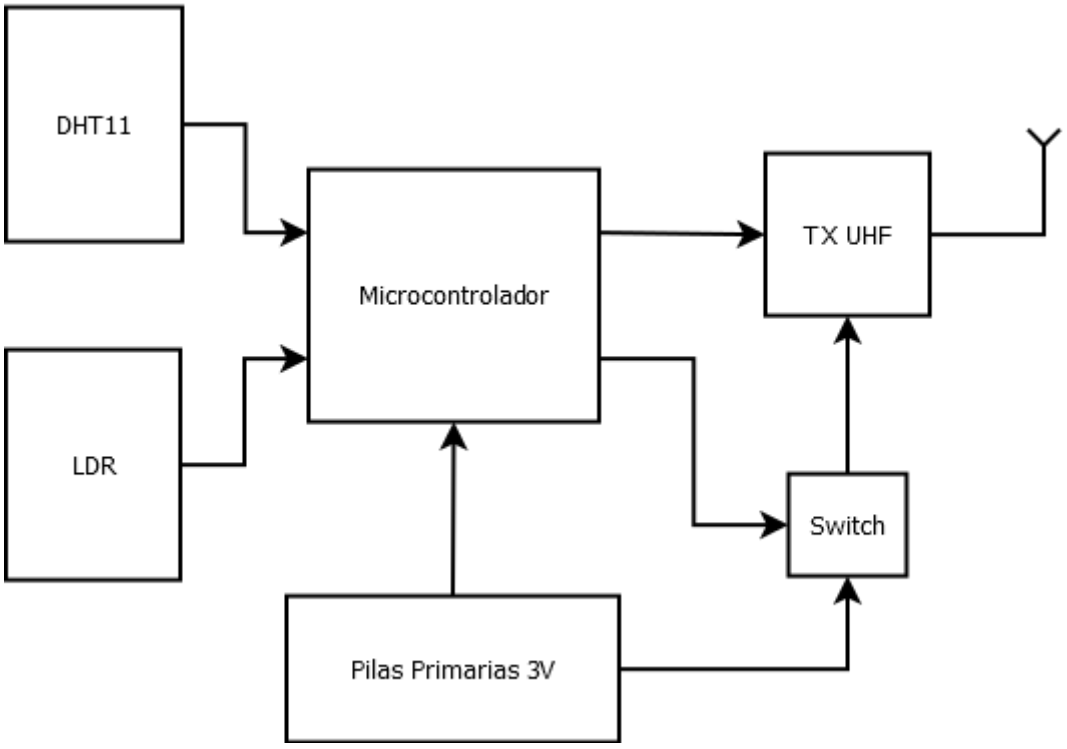
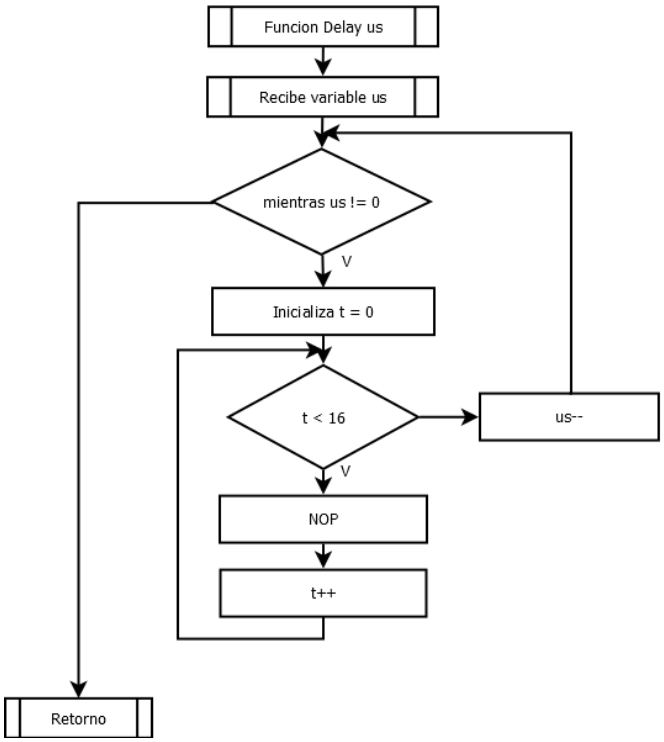
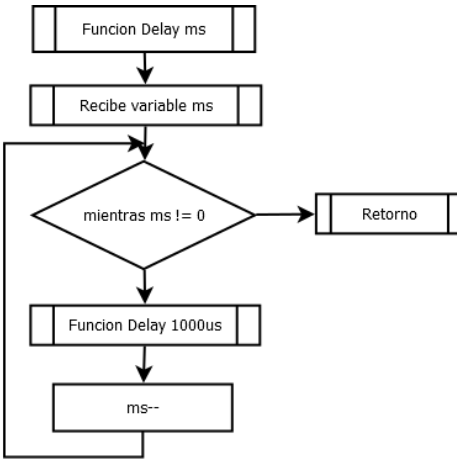
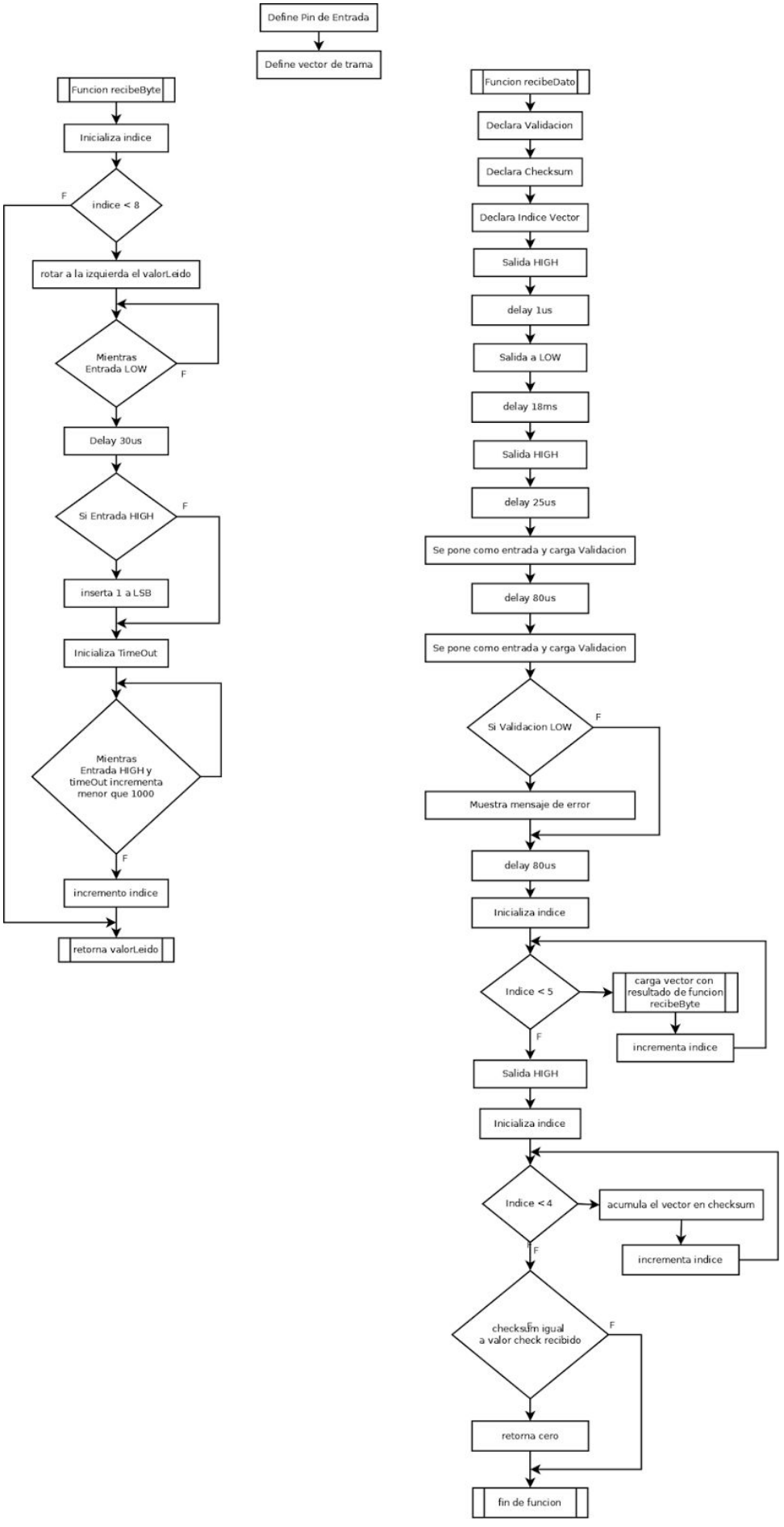
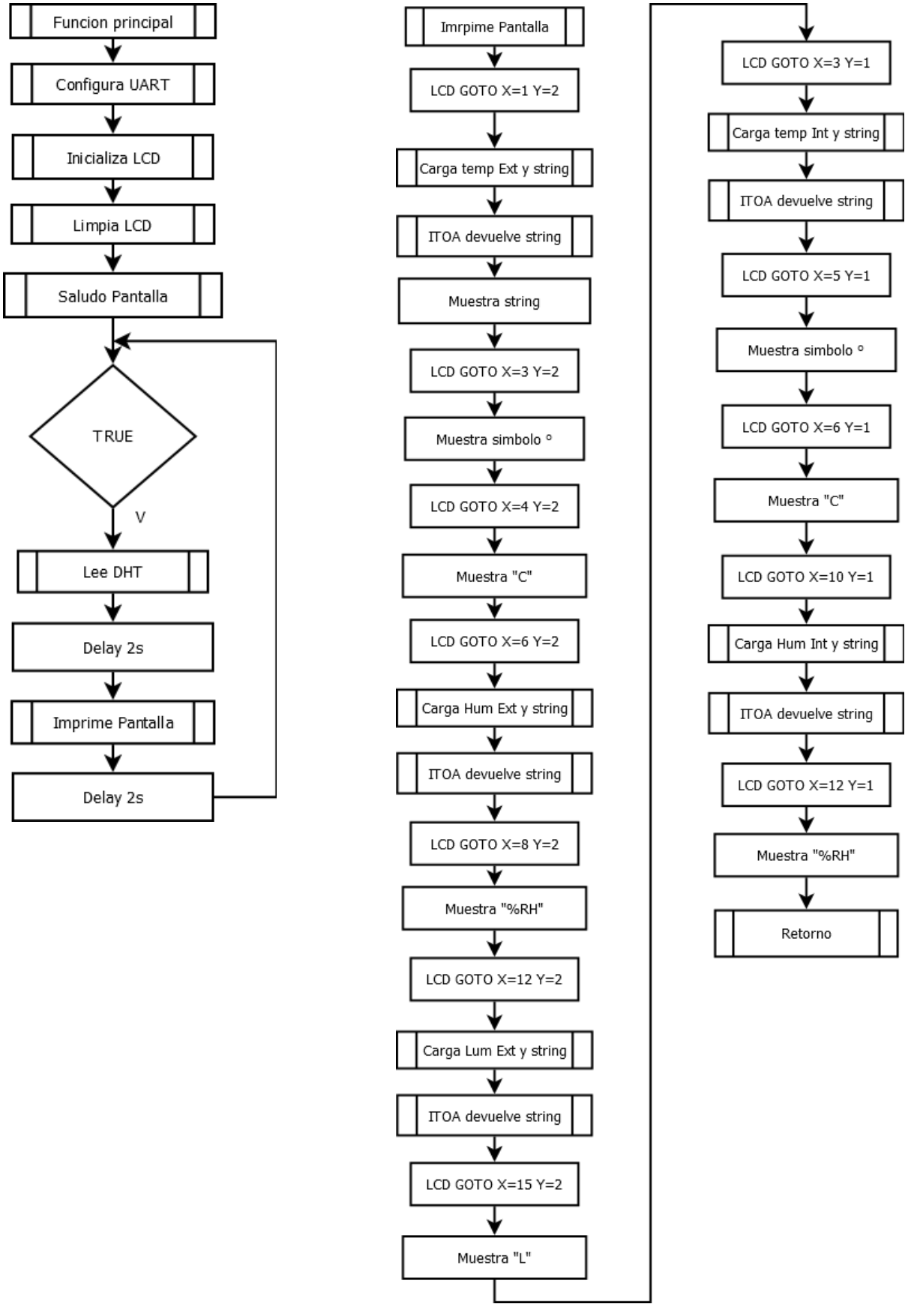


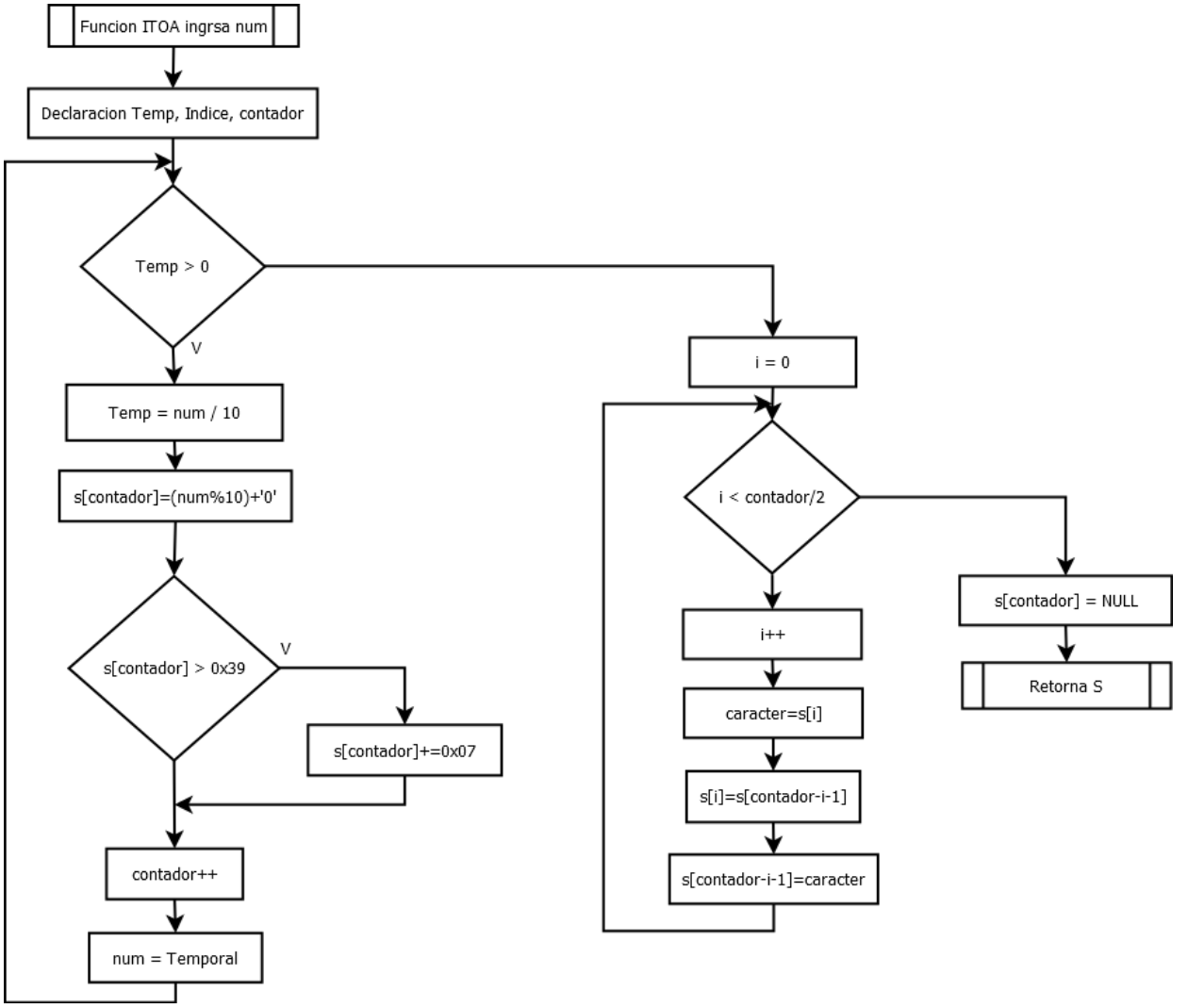
Diagrama de bloques de Firmware Interior:

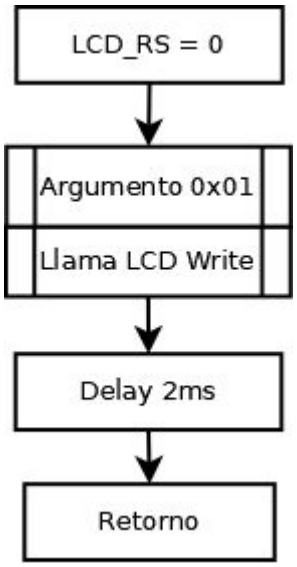


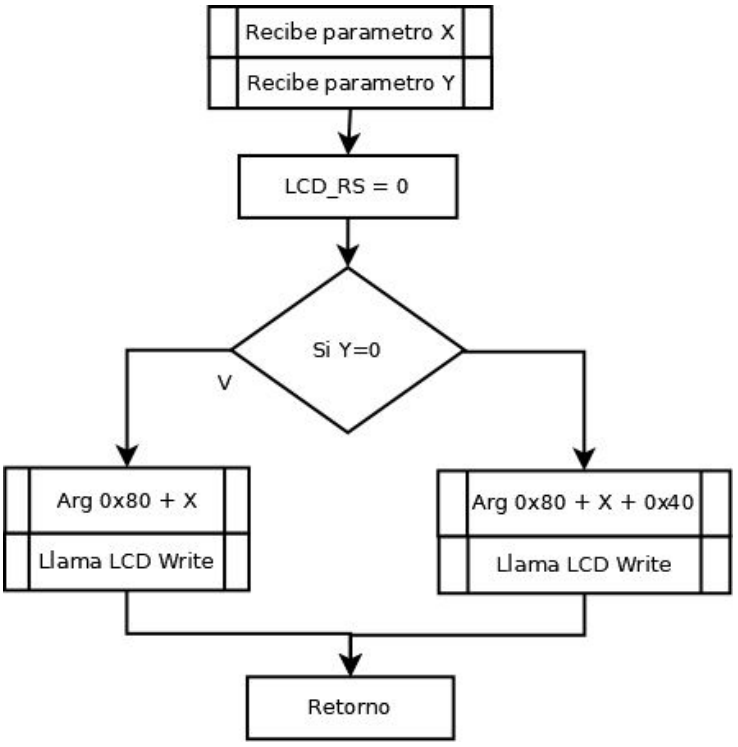


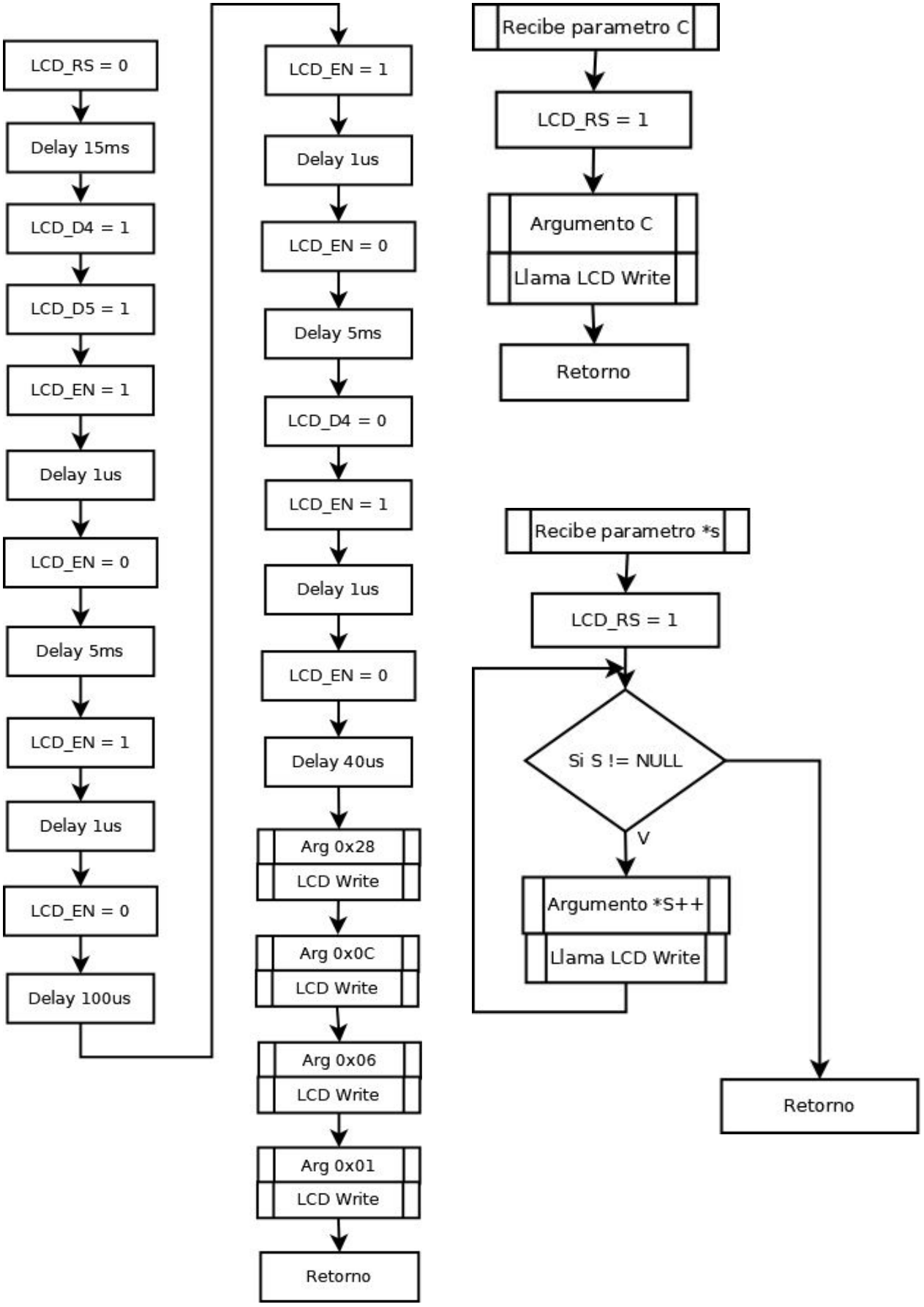

Define Pin de Entrada
Define vector de trama
Funcion recibeByte
Inicializa indice
indice < 8
F
rotar a la izquierda el valorLeido
Mientras Entrada LOW
F
Delay 30us
Si Entrada HIGH
F
inserta 1 a LSB
Inicializa TimeOut
Mientras Entrada HIGH y timeOut incrementa menor que 1000
F
incremento indice
retorna valorLeido
Funcion recibeDato
Declara Validacion
Declara Checksum
Declara Indice Vector
Salida HIGH
delay 1us
Salida a LOW
delay 18ms
Salida HIGH
delay 25us
Se pone como entrada y carga Validacion
delay 80us
Se pone como entrada y carga Validacion
Si Validacion LOW
F
Muestra mensaje de error
delay 80us
Inicializa indice
Indice < 5
carga vector con resultado de funcion recibeByte
incrementa indice
F
Salida HIGH
Inicializa indice
Indice < 4
acumula el vector en checksum
incrementa indice
F
checksum igual a valor check recibido
F
retorna cero
fin de funcion

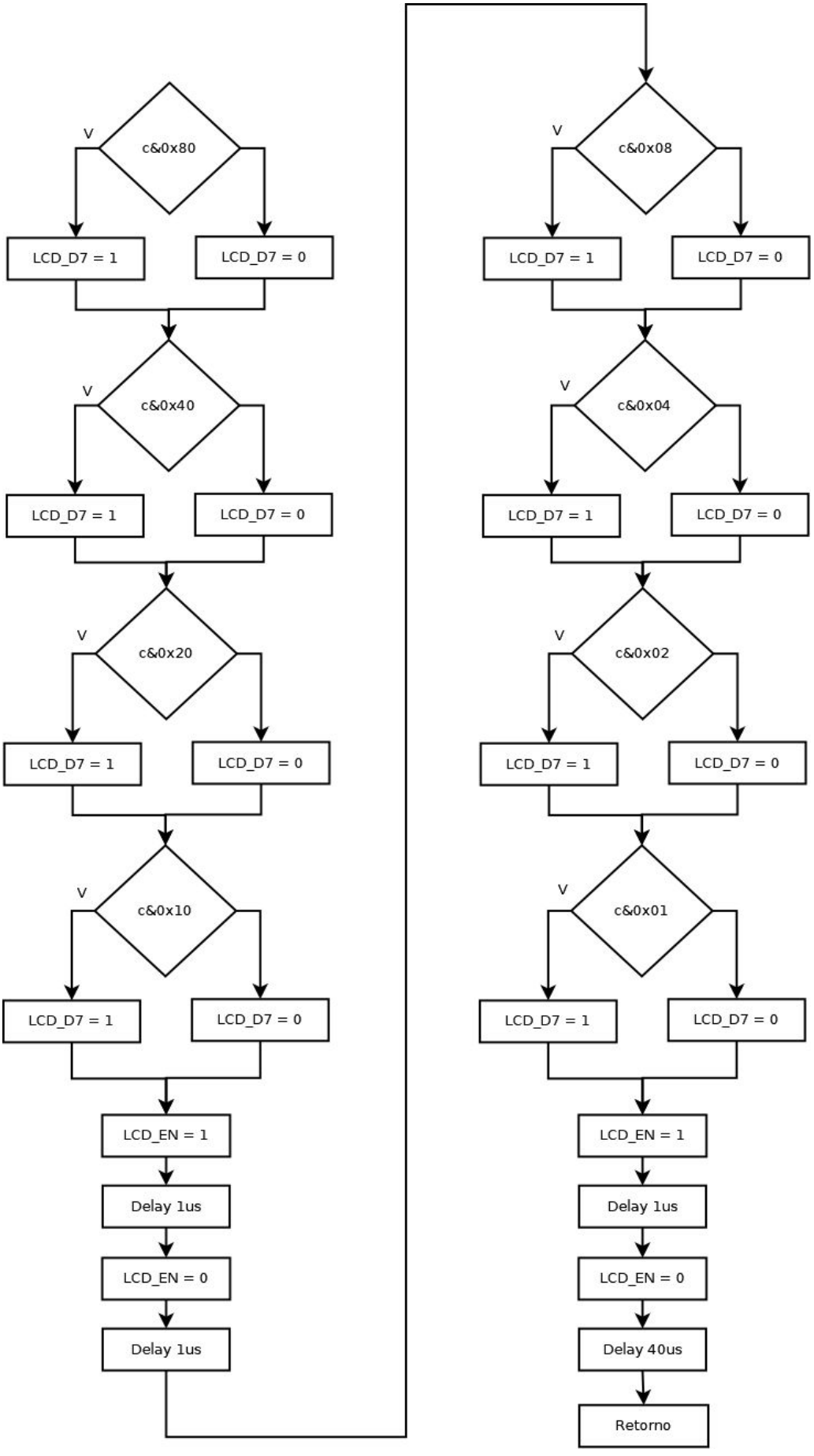

Funcion principal
Configura UART
Inicializa LCD
Limpia LCD
Saludo Pantalla
TRUE
V
Lee DHT
Delay 2s
Imprime Pantalla
Delay 2s
Imrpime Pantalla
LCD GOTO X=1 Y=2
Carga temp Ext y string
ITOA devuelve string
Muestra string
LCD GOTO X=3 Y=2
Muestra simbolo °
LCD GOTO X=4 Y=2
Muestra "C"
LCD GOTO X=6 Y=2
Carga Hum Ext y string
ITOA devuelve string
LCD GOTO X=8 Y=2
Muestra "%RH"
LCD GOTO X=12 Y=2
Carga Lum Ext y string
ITOA devuelve string
LCD GOTO X=15 Y=2
Muestra "L"
LCD GOTO X=3 Y=1
Carga temp Int y string
ITOA devuelve string
LCD GOTO X=5 Y=1
Muestra simbolo °
LCD GOTO X=6 Y=1
Muestra "C"
LCD GOTO X=10 Y=1
Carga Hum Int y string
ITOA devuelve string
LCD GOTO X=12 Y=1
Muestra "%RH"
Retorno

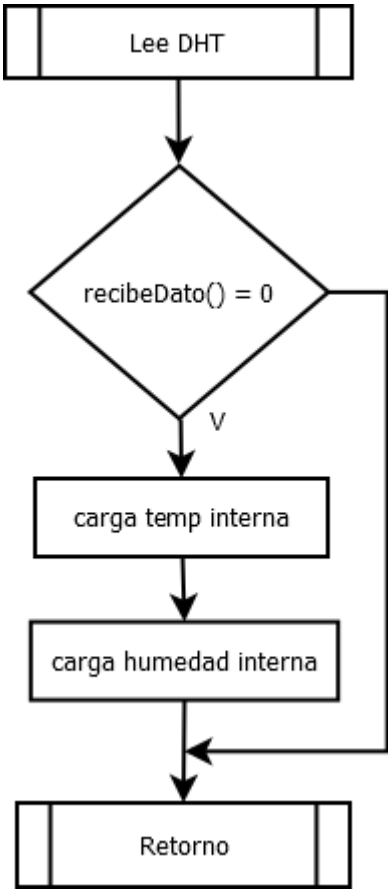

Funcion ITOA ingrsa num
Declaracion Temp, Indice, contador
Temp > 0
V
Temp = num / 10
s[contador]=(num%10)+'0'
s[contador] > 0x39
V
s[contador]+=0x07
contador++
num = Temporal
i = 0
i < contador/2
i++
caracter=s[i]
s[i]=s[contador-i-1]
s[contador-i-1]=caracter
s[contador] = NULL
Retorna S

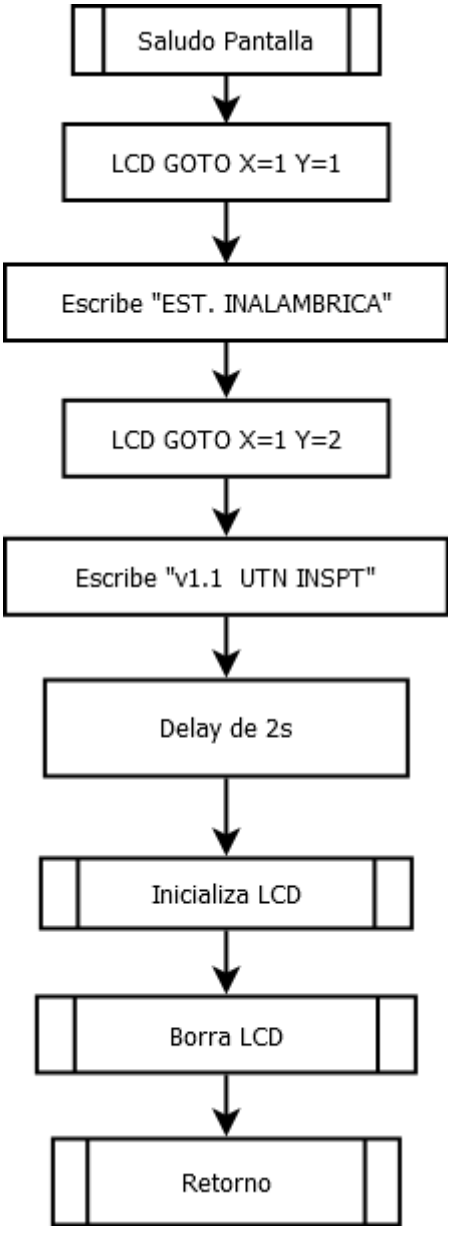

LCD_RS = 0
Argumento 0x01
Llama LCD Write
Delay 2ms
Retorno


Recibe parametro X
Recibe parametro Y
LCD_RS = 0
Si Y=0
V
Arg 0x80 + X
Llama LCD Write
Arg 0x80 + X + 0x40
Llama LCD Write
Retorno


LCD_RS = 0
Delay 15ms
LCD_D4 = 1
LCD_D5 = 1
LCD_EN = 1
Delay 1us
LCD_EN = 0
Delay 5ms
LCD_EN = 1
Delay 1us
LCD_EN = 0
Delay 100us
LCD_EN = 1
Delay 1us
LCD_EN = 0
Delay 5ms
LCD_D4 = 0
LCD_EN = 1
Delay 1us
LCD_EN = 0
Delay 40us
Arg 0x28
LCD Write
Arg 0x0C
LCD Write
Arg 0x06
LCD Write
Arg 0x01
LCD Write
Retorno
Recibe parametro C
LCD_RS = 1
Argumento C
Llama LCD Write
Retorno
Recibe parametro *s
LCD_RS = 1
Si S != NULL
V
Argumento *S++
Llama LCD Write
Retorno


V
c&0x80
LCD_D7 = 1
LCD_D7 = 0
V
c&0x40
LCD_D7 = 1
LCD_D7 = 0
V
c&0x20
LCD_D7 = 1
LCD_D7 = 0
V
c&0x10
LCD_D7 = 1
LCD_D7 = 0
LCD_EN = 1
Delay 1us
LCD_EN = 0
Delay 1us
V
c&0x08
LCD_D7 = 1
LCD_D7 = 0
V
c&0x04
LCD_D7 = 1
LCD_D7 = 0
V
c&0x02
LCD_D7 = 1
LCD_D7 = 0
V
c&0x01
LCD_D7 = 1
LCD_D7 = 0
LCD_EN = 1
Delay 1us
LCD_EN = 0
Delay 40us
Retorno


Lee DHT
recibeDato() = 0
V
carga temp interna
carga humedad interna
Retorno


Saludo Pantalla
LCD GOTO X=1 Y=1
Escribe "EST. INALAMBRICA"
LCD GOTO X=1 Y=2
Escribe "v1.1 UTN INSPT"
Delay de 2s
Inicializa LCD
Borra LCD
Retorno

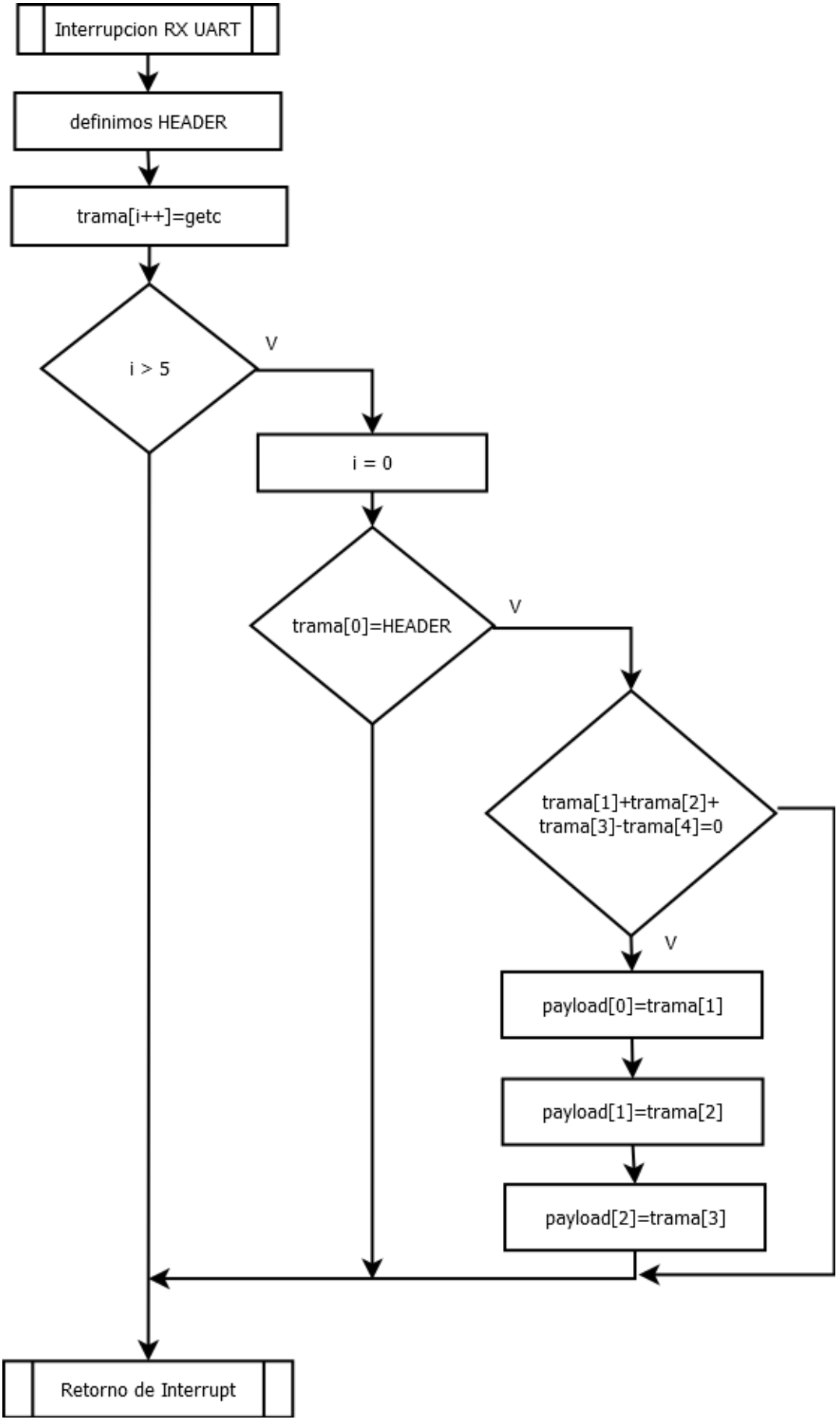
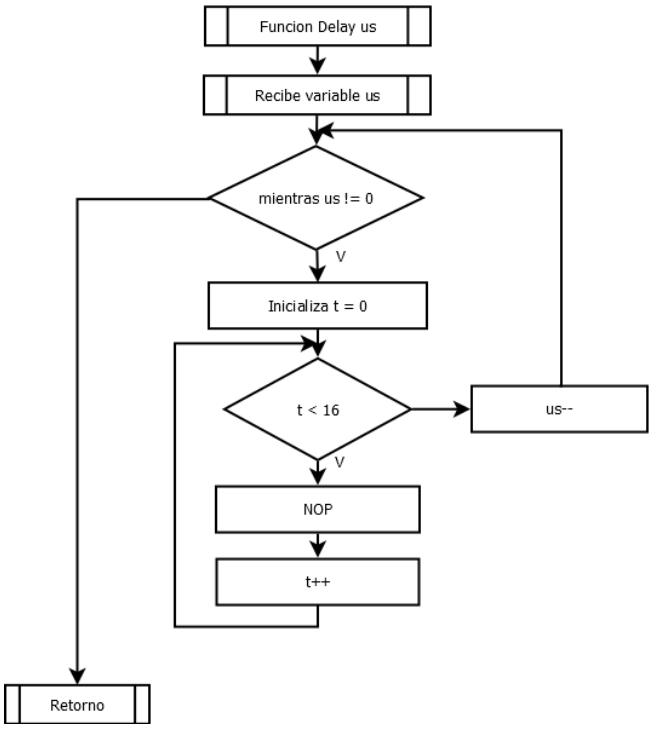
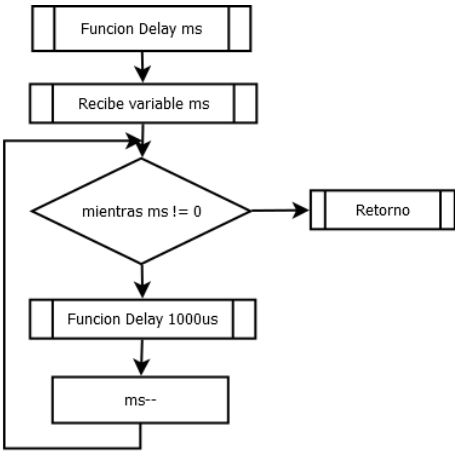
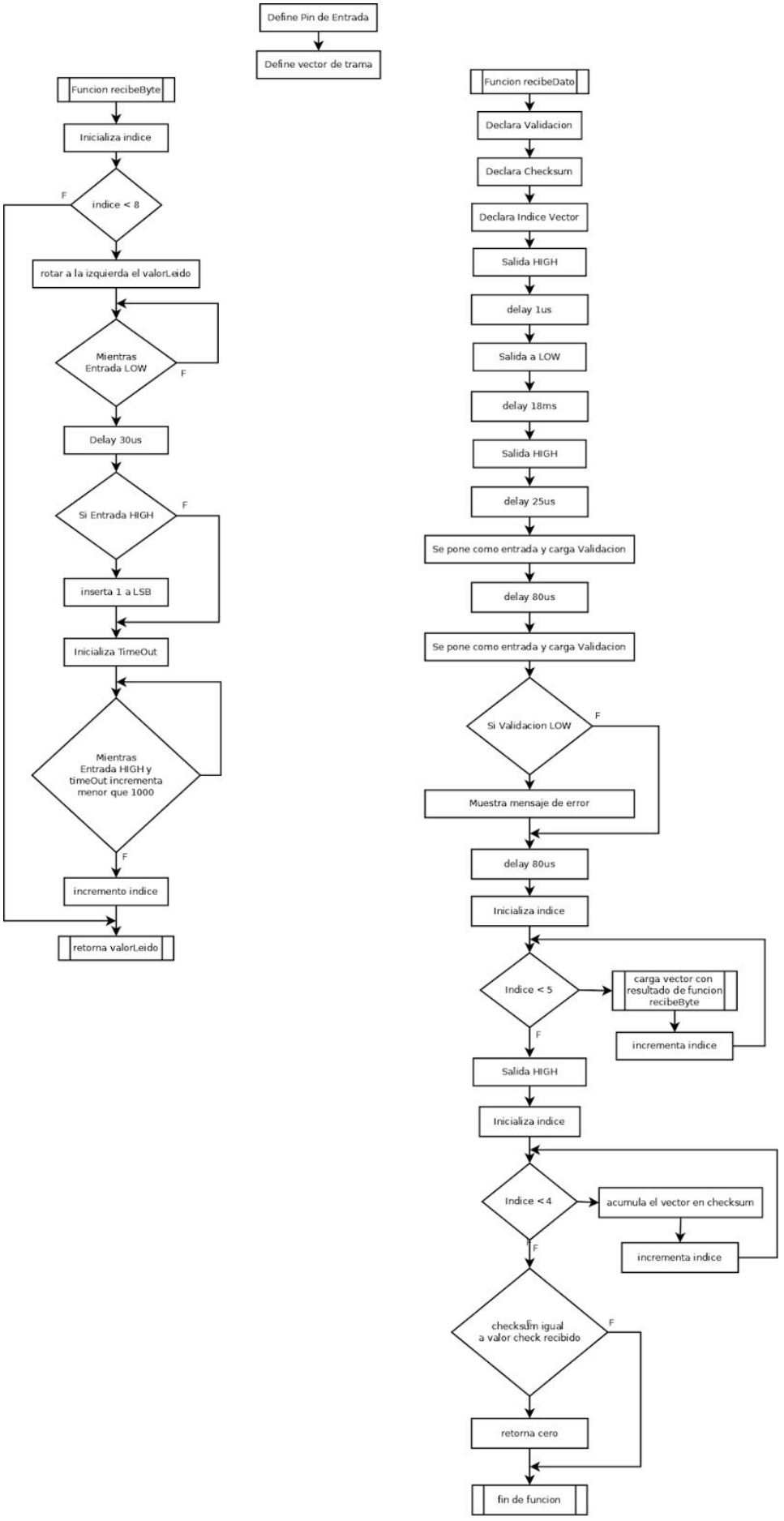
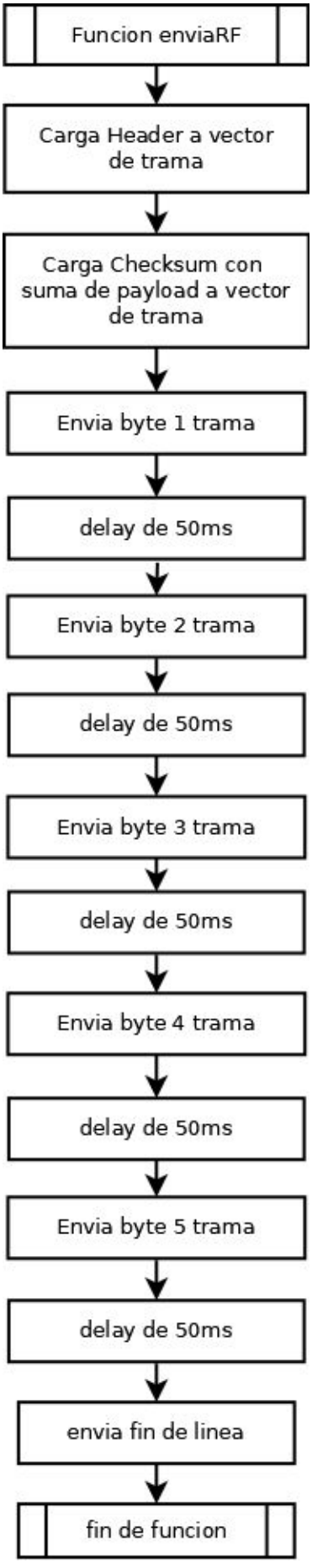

Interrupcion RX UART
definimos HEADER
trama[i++]=getc
i > 5
V
i = 0
trama[0]=HEADER
V
trama[1]+trama[2]+
trama[3]-trama[4]=0
V
payload[0]=trama[1]
payload[1]=trama[2]
payload[2]=trama[3]
Retorno de Interrupt

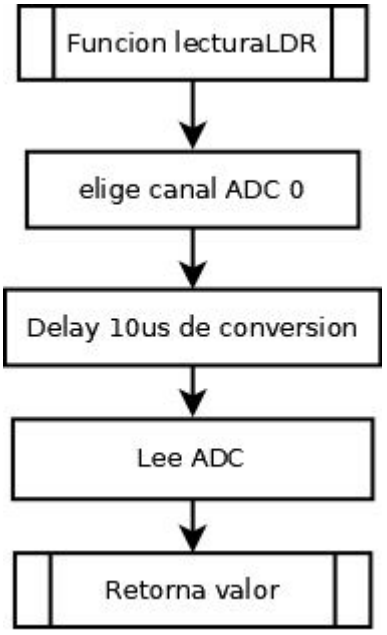
Diagrama de bloques de Firmware Exterior:

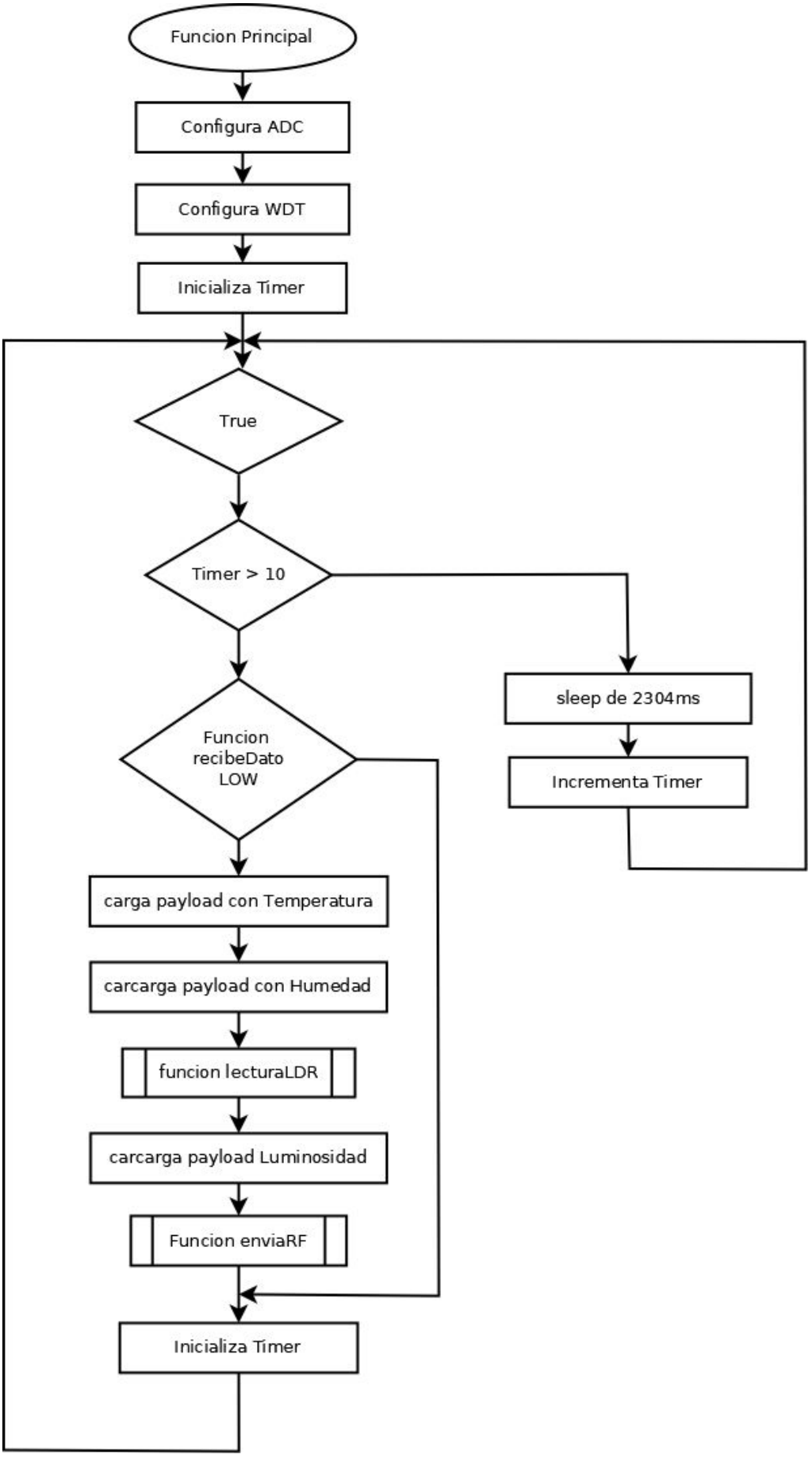





Define Pin de Entrada
Define vector de trama
Funcion recibeByte
Inicializa indice
indice < 8
F
rotar a la izquierda el valorLeido
Mientras Entrada LOW
F
Delay 30us
Si Entrada HIGH
F
inserta 1 a LSB
Inicializa TimeOut
Mientras Entrada HIGH y timeOut incrementa menor que 1000
F
incremento indice
retorna valorLeido
Funcion recibeDato
Declara Validacion
Declara Checksum
Declara Indice Vector
Salida HIGH
delay 1us
Salida a LOW
delay 18ms
Salida HIGH
delay 25us
Se pone como entrada y carga Validacion
delay 80us
Se pone como entrada y carga Validacion
Si Validacion LOW
F
Muestra mensaje de error
delay 80us
Inicializa indice
Indice < 5
carga vector con resultado de funcion recibeByte
incrementa indice
F
Salida HIGH
Inicializa indice
Indice < 4
acumula el vector en checksum
incrementa indice
F
checksum igual a valor check recibido
F
retorna cero
fin de funcion


Funcion enviaRF
Carga Header a vector de trama
Carga Checksum con suma de payload a vector de trama
Envia byte 1 trama
delay de 50ms
Envia byte 2 trama
delay de 50ms
Envia byte 3 trama
delay de 50ms
Envia byte 4 trama
delay de 50ms
Envia byte 5 trama
delay de 50ms
envia fin de linea
fin de funcion


Funcion lecturaLDR
elige canal ADC 0
Delay 10us de conversion
Lee ADC
Retorna valor


Funcion Principal
Configura ADC
Configura WDT
Inicializa Timer
True
Timer > 10
sleep de 2304ms
Incrementa Timer
Funcion recibeDato LOW
carga payload con Temperatura
carcarga payload con Humedad
funcion lecturaLDR
carcarga payload Luminosidad
Funcion enviaRF
Inicializa Timer

Implementación

Especificaciones de Hardware:

Dispositivo Interno:

- Sensor de Temperatura y Humedad, DHT11 (comunicación onWire propietaria).
- Receptor modular RWS simplex UHF operando en la banda de 70cm en 433.92MHz, modulación ASK OOK utilizando protocolo RS232, Antena del tipo microstrip IFA 50 Ohms.
- Display LCD 2x16 con controlador HD44780, para mostrar Temp, Hum, (Interna y Externa), Luminosidad (Externa).
- Cargador de Batería tipo Secundaria Li-Ion con micro USB basado en controlador TP4056, Celda 18650, 4800mAh (típico) .
- Microcontrolador de 8bit Silabs C8051F832.
- Circuito impreso hasta 2 Layer con PTH y tecnologia SMD.
- Aceptación IPC-356, IPC-600, UL-94.

Dispositivo Externo:

- Sensor de Temperatura y Humedad, DHT11 (comunicación onWire propietaria).
- Sensor de Luminosidad basado en LDR con acondicionamiento por firmware.
- Transmisor modular TWS simplex UHF operando en la banda de 70cm en 433.92MHz, modulación ASK OOK utilizando protocolo RS232, Antena del tipo microstrip IFA 50 Ohms.
- Batería del tipo primaria alcalina ZnMnO2 tamaño AA, 2500mAh (típico).
- Microcontrolador de 8bit Silabs C8051F832.
- Circuito impreso hasta 2 Layer con PTH y tecnologia SMD.
- Aceptación IPC-356, IPC-600, UL-94.

Especificaciones de Firmware:

El firmware debe contar con estándares de programación, encabezado del programa, descripción, autores, versionado, fechas y referencias.
Las funciones deben contar con los valores de ingreso y egreso a la misma como así el tipo de datos y la descripción de la función.
El programa se realizará en lenguaje C.

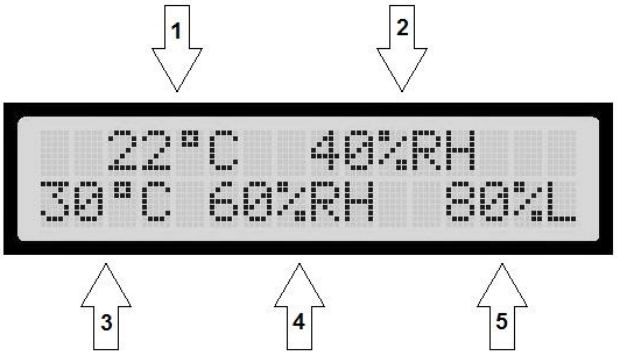
- Desarrollo de Bibliotecas necesarias (Estandarizado)
- Pruebas Unitarias de cada Biblioteca con el hardware asociado
- Desarrollo de programa principal
- Manejo de errores
- Creación de diagramas de flujo o estado
- Pruebas Integrados del firmware con todas las Bibliotecas y devices

- ➔ Biblioteca para sensor DHT11
  - ◆ Manejo de Bus OneWire 5 bytes (4 bytes de datos 1 byte de checksum)
  - ◆ Formateo de datos de **salida** Temperatura: entero sin signo de 8bit
  - ◆ Formateo de datos de **salida** Humedad: entero sin signo de 8bit
- ➔ Biblioteca de control del LCD
  - ◆ Mapeo de los 2 renglones y 16 caracteres.
  - ◆ Control de memoria mediante bus de 4bit y control.
  - ◆ **Entrada** mediante string con control xy de caracteres.
- ➔ Biblioteca bitbanging para protocolo manchester en los enlaces UHF
  - ◆ BitBanging de del protocolo.
  - ◆ Realización de Checksum
  - ◆ **Salida** de datos en formato hexadecimal.
  - ◆ **Entrada** de datos en formato hexadecimal.
- ➔ Biblioteca del sensor de luminosidad
  - ◆ Utilización del conversor analogico digital provisto por el microcontrolador.
  - ◆ Configuración de baja velocidad a resolución de 8bit.
  - ◆ **Salida** datos en formato entero.
- ➔ Manejo de Baja Energía
  - ◆ Apagado de dispositivos periféricos en desuso mediante transistores NMOS.
  - ◆ Reloj WDT para función WakeUp del MCU cada 2.3s
  - ◆ Si los datos a medir (sensores) no cambian respecto a la medición anterior, no se enviaran datos repetidos mejorando el consumo.

Especificaciones Técnicas:

Rango de temperatura 0-50°c (precisión ±2°c)
Rango de humedad 20-90%RH (precisión ±5%)
Rango de luminosidad 0-100% (precisión ±20%)
Rango de trabajo hasta 20m
Frecuencia de trabajo 433MHz
Batería de sensor 2xAA Alcalina - Vida útil de 2 años
Batería de cuerpo 1x18650 - Vida útil Recargable via USB
Medidas cuerpo 113x67x27.5mm
Medidas Sensor 91x70x32.5mm
Gabinete exterior IPX3
Este equipo no dispone de protección ATEX, por lo que no debe ser usado en atmósferas potencialmente explosivas (polvo, gases inflamables).

Look & Feel de pantalla:



1) Temperatura Interior
2) Humedad Interior
3) Temperatura Exterior
4) Humedad Exterior
5) Cantidad de Luz Exterior

Trama implementada para el enlace:

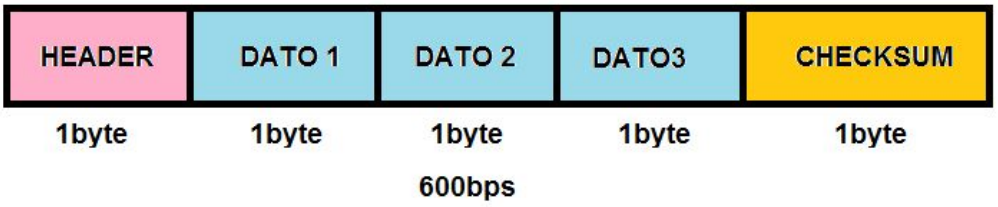


Diagrama Electronico Interior:

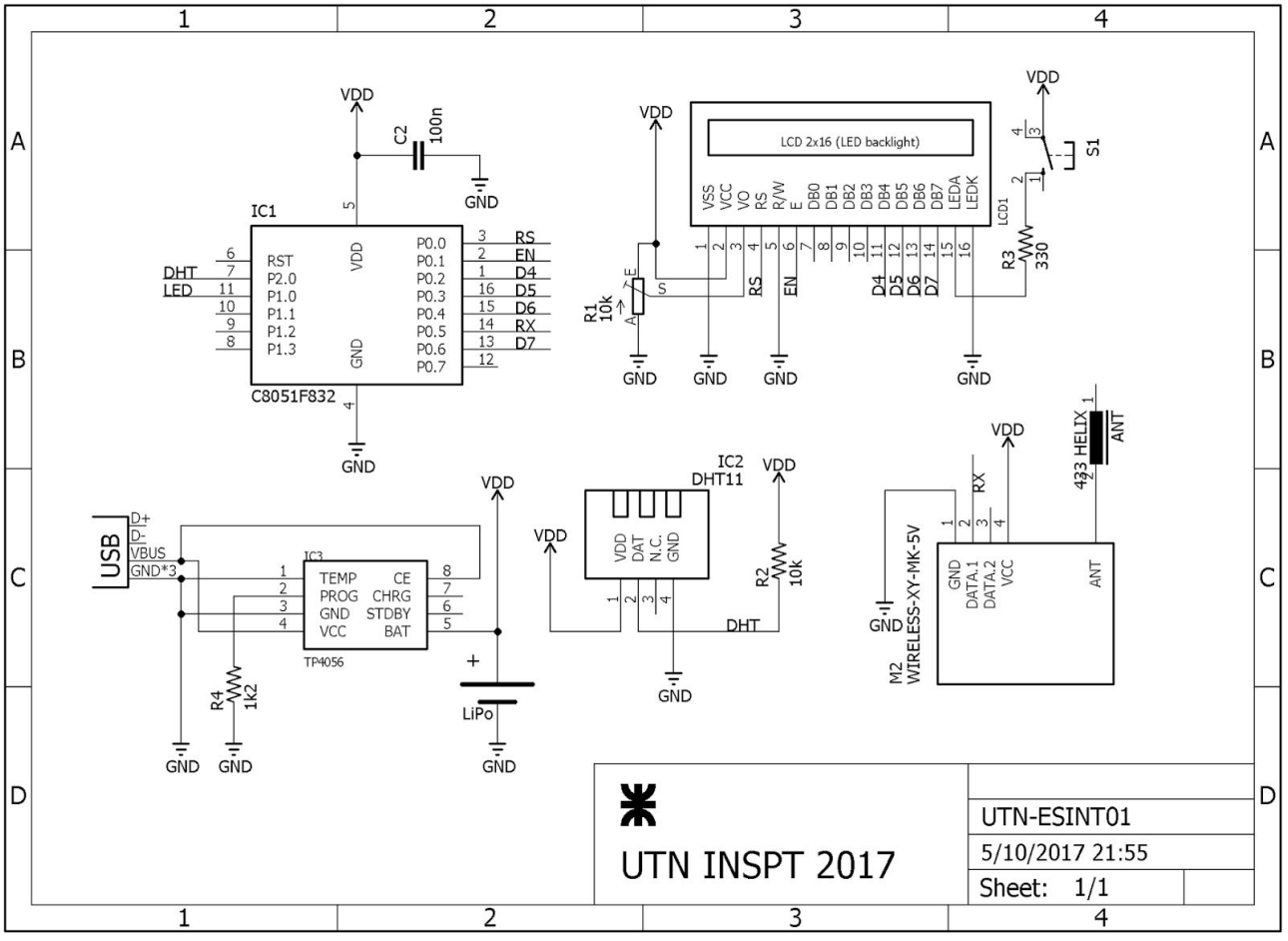
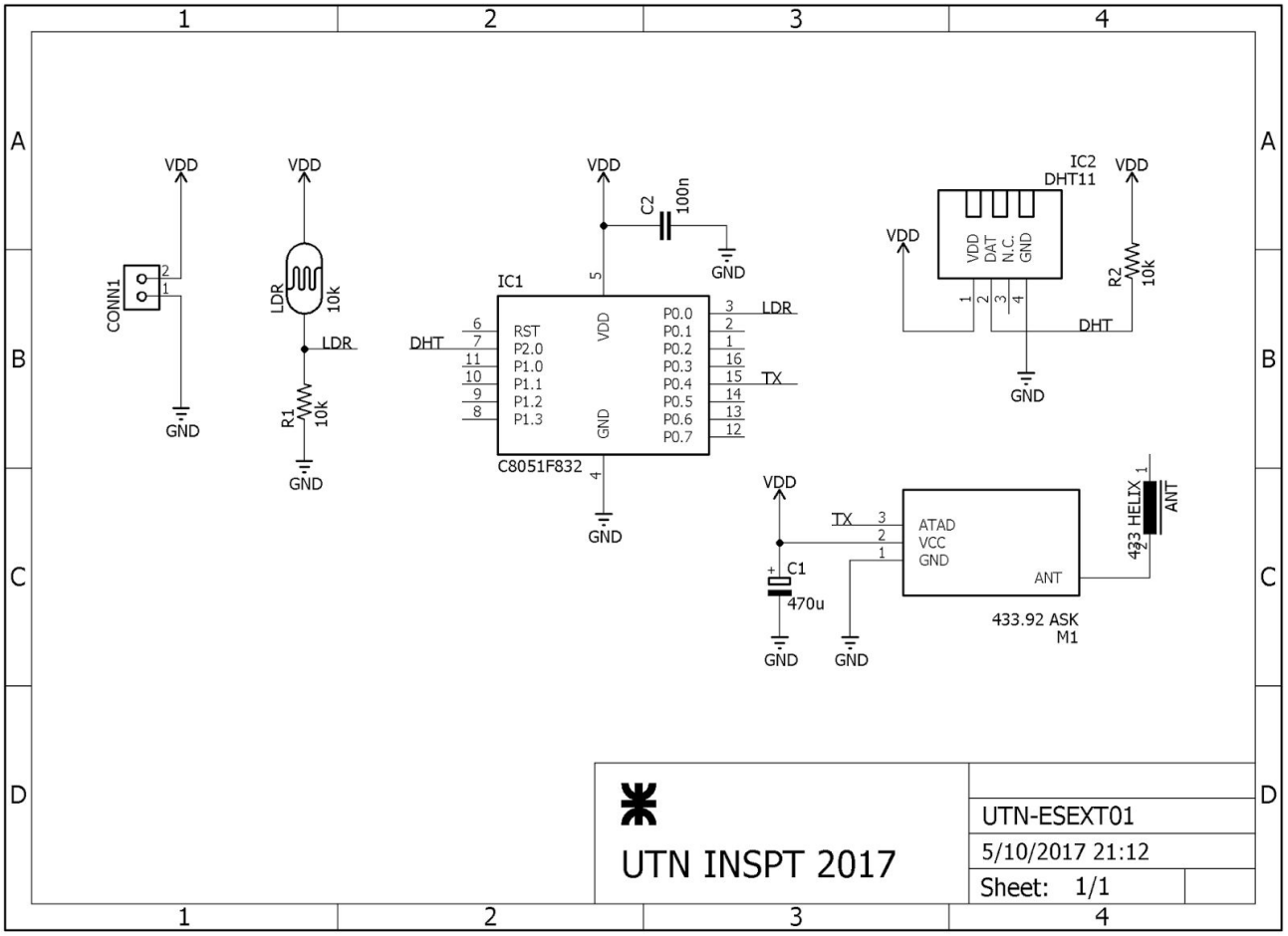
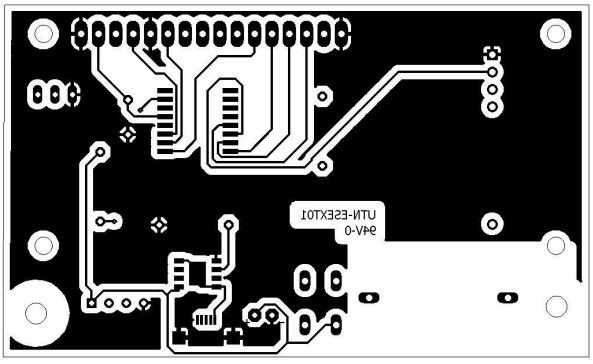
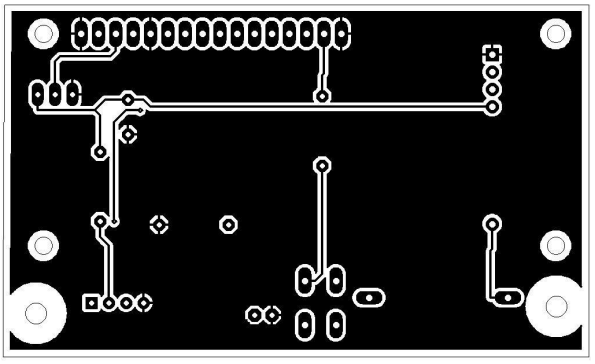


Diagrama Electronico Exterior:

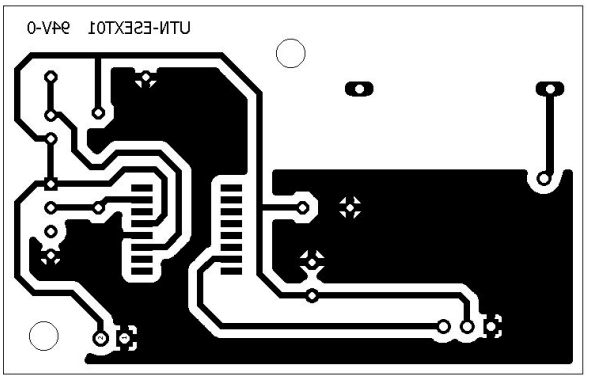


Circuito Impreso Interior:

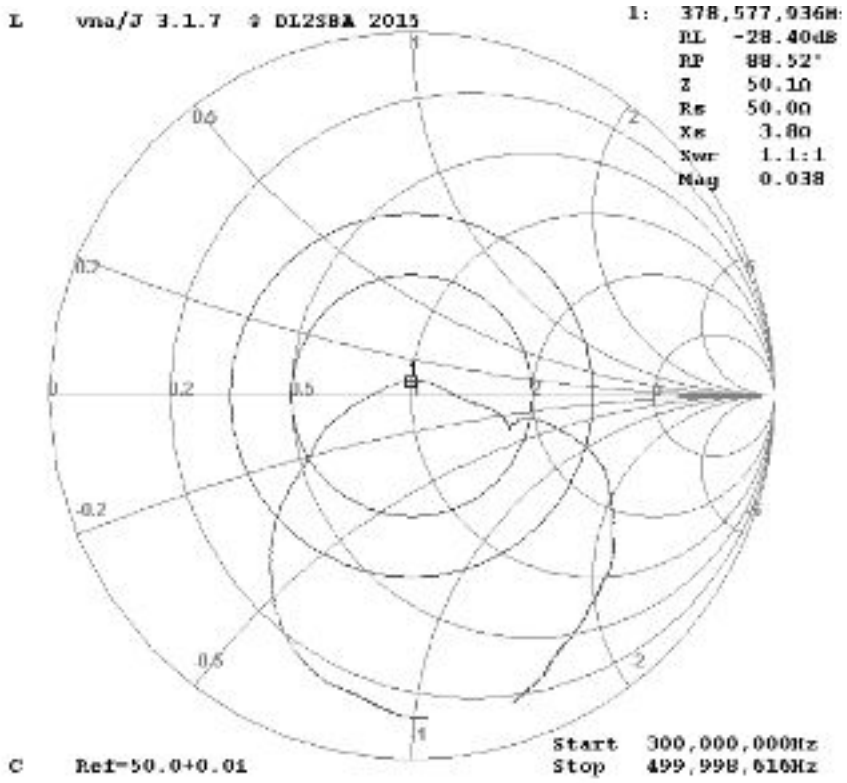




Circuito Impreso Exterior:



Análisis de antena tipo ANT-433-HETH:



Código del proyecto

Firmware Interior:

```
//*****************************************************************************************************
// Programa para la estación meteorológica Interna inalámbrica
// El programa recibe Temperatura, Humedad y Luminosidad por interrupción
// El sistema mide Temperatura y Humedad interior
// El enlace se realiza mediante una comunicación ASK OOK UHF sobre UART Invertido a 600bps
// Se muestra medición interna y externa en un LCD de 2x16
// Microcontrolador Silabs C8051F832
//*****************************************************************************************************
#include <REG51F800.H>                                  //Header de Silabs C8051F832
#include <intrins.h>                                    //Biblioteca para funciones de assembler para el Delay
#include <stdio.h>                                      //Biblioteca estandar de entrada y salida
//*****************************************************************************************************
// Delay Bloqueante Micro Segundos
//*****************************************************************************************************
void delay_us(unsigned int us_count){                   //Función para delay de micro-segundos
   int t=0;
   while(us_count!=0){                                  //Mientras que el contador es distinto de cero
      for(t=0;t<16;t++){                                //16MIPS dividido en 16 para 1us
         _nop_();                                       //Ejecuta función NOP de ensamblador
      }
      us_count--;                                       //Decremento del valor de delay
   }
}
//*****************************************************************************************************
// Delay Bloqueante Mili Segundos
//*****************************************************************************************************
void delay_ms(unsigned int us_count){                   //Función para delay de micro-segundos
   while(us_count!=0){                                  //Mientras que el contador es distinto de cero
      delay_us(1000);                                   //Ejecuta funcion delay micro segundos
      us_count--;                                       //Decremento del valor de delay
   }
}
//*****************************************************************************************************
// Variables
//*****************************************************************************************************
#define HEADER 200                                      //Definición de valor Header para el payload
sbit DATA = P2^0;                                       //Pin del bus de un hilo para el DHT11
sbit STATUS = P1^0;                                     //Pin para analizar el status de la interrupt
static int datoInt[2];                                  //Variable donde se alojan los datos internos
static int payload[5];                                  //Variable donde se aloja el payload
static int datoExt[3];                                  //Variable donde se alojan los datos externos
//*****************************************************************************************************
// Configuracion de pines del LCD
//*****************************************************************************************************
sbit LCD_RS = P0^0;                                     //Register select
sbit LCD_EN = P0^1;                                     //Enable
sbit LCD_D4 = P0^2;                                     //Data bits
sbit LCD_D5 = P0^3;                                     //Data bits
sbit LCD_D6 = P0^4;                                     //Data bits
sbit LCD_D7 = P0^6;                                     //Data bits
//*****************************************************************************************************
// Función de escritura LCD
//*****************************************************************************************************
void lcd_write(unsigned char c){
   if(c & 0x80) LCD_D7 = 1; else LCD_D7 = 0;
   if(c & 0x40) LCD_D6 = 1; else LCD_D6 = 0;
   if(c & 0x20) LCD_D5 = 1; else LCD_D5 = 0;
   if(c & 0x10) LCD_D4 = 1; else LCD_D4 = 0;
   LCD_EN = 1;
   delay_us(1);
   LCD_EN = 0;
   delay_us(1);
   if(c & 0x08) LCD_D7 = 1; else LCD_D7 = 0;
   if(c & 0x04) LCD_D6 = 1; else LCD_D6 = 0;
   if(c & 0x02) LCD_D5 = 1; else LCD_D5 = 0;
   if(c & 0x01) LCD_D4 = 1; else LCD_D4 = 0;
```

```
    LCD_EN = 1;
    delay_us(1);
    LCD_EN = 0;
    delay_us(40);
}
//*****************************************************************************************************
// Función que Limpia y enciende el LCD
//*****************************************************************************************************
void lcd_clear(void){
    LCD_RS = 0;                                         //Escribir bytes de control
    lcd_write(0x01);
    delay_ms(2);
}
//*****************************************************************************************************
// Función que escribe un caracter en pantalla
//*****************************************************************************************************
void lcd_putch(unsigned char c){
    LCD_RS = 1;                                         //Escribir bytes de control
    lcd_write(c);
}
//*****************************************************************************************************
// Escribir una cadena de caracteres en la pantalla LCD
//*****************************************************************************************************
void lcd_puts(char *s){
    LCD_RS = 1;                                         //Escribir bytes de control
    while(*s)
       lcd_write(*s++);
}
//*****************************************************************************************************
// Funcion para ir a la posición específica
//*****************************************************************************************************
void lcd_goto(unsigned char pos,unsigned char line){
    LCD_RS = 0;                                         //Escribir bytes de control
    if(line==0)
       lcd_write(0x80 + pos);
    else
       lcd_write(0x80 + pos+ 0x40);
}
//*****************************************************************************************************
// Función que Inicializar la pantalla LCD en modo de 4 bits
//*****************************************************************************************************
void lcd_init(void){
    LCD_RS = 0;                                         //Escribir bytes de control
    delay_ms(15);                                       //Retraso de encendido
    LCD_D4 = 1;                                         //Inicialización según datasheet
    LCD_D5 = 1;                                         //Inicialización según datasheet
    LCD_EN = 1;
    delay_us(1);
    LCD_EN = 0;
    delay_us(1);                                        //Habilitación / Deshabilitación del LCD a demanda
    delay_ms(5);                                        //Delay de configuración de datasheet
    LCD_EN = 1;
    delay_us(1);
    LCD_EN = 0;
    delay_us(1);                                        //Habilitación / Deshabilitación del LCD a demanda
    delay_us(100);                                      //Delay de configuración de datasheet
    LCD_EN = 1;
    delay_us(1);
    LCD_EN = 0;
    delay_us(1);                                        //Habilitación / Deshabilitación del LCD a demanda
    delay_ms(5);                                        //Delay de configuración de datasheet
    LCD_D4 = 0;                                         //Establecer el modo de 4 bits
    LCD_EN = 1;
    delay_us(1);
    LCD_EN = 0;
    delay_us(1);                                        //Habilitación / Deshabilitación del LCD a demanda
    delay_us(40);                                       //Delay de configuración de datasheet
    lcd_write(0x28);                                    //Modo de 4 bits, 1/16 de servicio, fuente 5x8, 2 líneas
    lcd_write(0x0C);                                    //Enciende Display
    lcd_write(0x06);                                    //Cursor de avance del modo de entrada
    lcd_write(0x01);                                    //Borrar la visualizaci restablecer el cursor
}
```

```
//*****************************************************************************************************
// Función ITOA para convertir entero en ascii
//*****************************************************************************************************
char *itoa(long int num, char *s){                          //Función ITOA (Entero to ASCII)
   unsigned long int temp=1;                                //Declaración de valor temporal
   unsigned int i, cnt=0;                                   //Declaración de índices y contadores
   char c;                                                  //Declaración de variable de carácter de salida
   while(temp>0){                                           //Rutina de Conversión (Queda invertida)
      temp=(num/10);                                        //Conversión de carácter a carácter
      s[cnt]=(num%10)+'0';                                  //utilizando divisiones y resto
      if(s[cnt]>0x39)                                       //sumando el offset de la tabla ASCII
         s[cnt]+=0x7;
      cnt++;
      num=temp;
   }
   for(i=0;i<(int)(cnt/2);i++){                             //Rutina para invertir el numero convertido
      c=s[i];                                               //Intercambio de variables
      s[i]=s[cnt-i-1];
      s[cnt-i-1]=c;
   }
   s[cnt]='\0';                                             //Carácter nulo, fin de la conversión ITOA
   return s;                                                //Retorno del valor ASCII
}
//*****************************************************************************************************
// Función Interrupción UART que realiza el parse de datos, validación de Header y Checksum
// Si validación y Checksum son válidos, carga el vector DATO para ser utilizado
//*****************************************************************************************************
int i=0;                                                    //Variable para el contador de bytes de entrada
void UART0_ISR(){                                           //Función de servicio de interrupción
   payload[i++]=SBUF0;                                      //Guarda byte de entrada en payload. incrementa índice
   if(i>5){                                                 //Si índice mayor a 5 se asume que se completa el payload
      if(payload[0]==HEADER){                               //Validación que el Header sea 200
         if(payload[1]+payload[2]+payload[3]-payload[4]==0){ //Checksum, si datos leídos son igual a checksum
            datoExt[0]=payload[1];                          //Cargamos los datos en el vector
            datoExt[1]=payload[2];                          //Cargamos los datos en el vector
            datoExt[2]=payload[3];                          //Cargamos los datos en el vector
            STATUS ^= 1;                                    //Status de dato recibido OK
         }
      }
      i=0;                                                  //Una vez completados los 5 bytes, reinicia el contador
   }
}
//*****************************************************************************************************
// Trama DHT11 - Segun Datasheet
//  ____            ____    ____                                  ___
//      |__________|    |____|    |.................................|
//      |   18ms   |    |80us|80us|              5x8 bit            |
//      |    PIC   |    |                 DHT11 respuesta           | Fin de
//      | request  |    |  2x80us, intRH, decRH, intT, decT, checksum  | Trama
// ____          ________
//      |________|        |...                      0 bit
//       <-50us-> <-27us->
// ____          ____________________________
//      |________|                          |...       1 bit
//       <-50us-> <---------70us---------->
//*****************************************************************************************************
unsigned int trama[5];                                      //Vector donde se alojan los datos
//*****************************************************************************************************
// Función de recepción de Byte
// Lee el valor leído en la trama y lo separa realizando shift
// Retorna el valor en forma de byte, es utilizado en la función recibeDato()
//*****************************************************************************************************
unsigned int recibeByte(){                                  //Función que recibe un Byte
   unsigned int valorLeido = 0;                             //Valor de retorno de la función
   int i=0;                                                 //Inicialización del índice
   for(i=0; i<8; i++){                                      //Iteración para recepción de bits
      valorLeido <<= 1;                                     //Registro de desplazamiento de bits
      while(DATA==0);                                       //Espera a DATA = 0
      delay_us(30);                                         //Demora de 30us (Del Datasheet)
      if(DATA==1){                                          //Pregunta si DATA = 1
          valorLeido |= 1;                                  //Realiza toggle del valor leído
      }
```

```
211.       while(DATA==1);                                    //Espera a DATA = 1
212.    }
213.    return valorLeido;                                    //Retorna el valor leído
214. }
215. //*****************************************************************************************************
216. // Función de recepción de dato para el DHT11
217. // Recibe los valores de temperatura y humedad (parte entera y decimales por separado)
218. // Recibe el checksum enviado por el DHT11 y lo compara con el leído en el programa
219. //*****************************************************************************************************
220. unsigned int recibeDato(){                               //Funcion que recibe el Dato
221.    int validacion = 0;                                   //Variable de Validación
222.    int checksum = 0;                                     //Variable de detección de cambios de secuencia
223.    int j=0;                                              //Variable para el lazo for
224.    DATA = 1;                                             //Set DATA = 1
225.    DATA = 0;                                             //Set DATA = 0
226.    delay_ms(18);                                         //Demora de 18ms (Del Datasheet)
227.    DATA = 1;                                             //Set DATA = 1
228.    delay_us(25);                                         //Demora de 25ms (Del Datasheet)
229.    validacion = DATA;                                    //Mueve valor de DATA a Validacion
230.    delay_us(80);                                         //Espera 80us (Del Datasheet)
231.    validacion = DATA;                                    //Mueve valor de DATA a Validacion
232.    if(!validacion){                                      //Si Validacion = 0, Error de secuencia
233.       printf( "Error en Checksum \r");                   //Muestra leyenda de error
234.    }
235.    delay_us(80);                                         //Espera 80us (Del Datasheet)
236.    for(j=0; j<5; j++){                                   //Lazo de carga de bytes de datos
237.        trama[j] = recibeByte();                          //Carga del vector de datos
238.    }
239.    DATA = 1;                                             //Set DATA = 1
240.    for(j=0; j<4; j++){                                   //Lazo de carga de bytes de verificación
241.        checksum += trama[j];                             //Carga de bytes de verificación
242.    }
243.    if(checksum == trama[4]){                             //Si la secuencia de verificación es correcta
244.       return 0;                                          //Se retorna 0 y se realiza la lectura
245.    }
246. }
247. //*****************************************************************************************************
248. // Lee DHT11
249. //*****************************************************************************************************
250. void leeDHT11(void){                                     //Función que lee el DHT11
251.    if(recibeDato()==0){                                  //Consulta si hay dato en la entrada del DHT11
252.       datoInt[0]=trama[2];                               //Carga el valor de temperatura DHT en byte 1 del payload
253.       datoInt[1]=trama[0];                               //Carga el valor de humedad DHT en byte 2 del payload
254.    }
255. }
256. //*****************************************************************************************************
257. // Saludo de Pantalla
258. //*****************************************************************************************************
259. void saludoPantalla(void){                               //Función que realiza un saludo inicial
260.    lcd_goto(1,1);                                        //Posiciona el cursor en la pantalla
261.    lcd_putch(" Est.Inalambrica");                        //Imprime mensaje renglón 1
262.    lcd_goto(1,2);                                        //Posiciona el cursor en la pantalla
263.    lcd_putch(" v1.1  UTN INSPT");                        //Imprime mensaje renglón 2
264.    delay_ms(2000);                                       //Delay de saludo
265.    lcd_init();                                           //Inicializa LCD
266.    lcd_clear();                                          //Borra LCD
267. }
268. //*****************************************************************************************************
269. // Impresión de Pantalla
270. //*****************************************************************************************************
271. void impPantalla(void){                                  //Función que imprime pantalla
272.    char string[4];                                       //Declaración de vector para mostrar en LCD
273.    lcd_goto(1,2);                                        //Posiciona el cursor en la pantalla
274.    itoa(datoExt[0],string);                              //Funcion que convierte entero en ascii
275.    lcd_puts(string);                                     //Muestra en el LCD
276.    lcd_goto(3,2);                                        //Posiciona el cursor en la pantalla
277.    lcd_putch(0xDF);                                      //Imprime unidad de medida
278.    lcd_goto(4,2);                                        //Posiciona el cursor en la pantalla
279.    lcd_putch("C");                                       //Imprime unidad de medida
280.    lcd_goto(6,2);                                        //Posiciona el cursor en la pantalla
281.    itoa(datoExt[1],string);                              //Funcion que convierte entero en ascii
282.    lcd_puts(string);                                     //Muestra en el LCD
```

```
283.    lcd_goto(8,2);                                  //Posiciona el cursor en la pantalla
284.    lcd_putch("%RH");                               //Imprime unidad de medida
285.    lcd_goto(12,2);                                 //Posiciona el cursor en la pantalla
286.    itoa(datoExt[2],string);                        //Funcion que convierte entero en ascii
287.    lcd_puts(string);                               //Muestra en el LCD
288.    lcd_goto(15,2);                                 //Posiciona el cursor en la pantalla
289.    lcd_putch("L");                                 //Imprime unidad de medida
290.    lcd_goto(3,1);                                  //Posiciona el cursor en la pantalla
291.    itoa(datoInt[0],string);                        //Funcion que convierte entero en ascii
292.    lcd_puts(string);                               //Muestra en el LCD
293.    lcd_goto(5,1);                                  //Posiciona el cursor en la pantalla
294.    lcd_putch(0xDF);                                //Imprime unidad de medida
295.    lcd_goto(6,1);                                  //Posiciona el cursor en la pantalla
296.    lcd_putch("C");                                 //Imprime unidad de medida
297.    lcd_goto(10,1);                                 //Posiciona el cursor en la pantalla
298.    itoa(datoInt[1],string);                        //Funcion que convierte entero en ascii
299.    lcd_puts(string);                               //Muestra en el LCD
300.    lcd_goto(12,1);                                 //Posiciona el cursor en la pantalla
301.    lcd_putch("%RH");                               //Imprime unidad de medida
302. }
303. //*************************************************************************************************
304. // Configuracion puerto serie a 9600 baud con cristal externo de 16MHz.
305. //*************************************************************************************************
306. void configuraUART(void){
307.    SCON0  = 0x50;                                  //SCON0: mode 1, 8-bit UART, enable rcvr
308.    TMOD  |= 0x20;                                  //TMOD:  timer 1, mode 2, 8-bit reload
309.    TH1    = 76;                                    //TH1:   reload value for 600 baud @ 16MHz
310.    TR1    = 1;                                     //TR1:   timer 1 run
311.    TI0    = 1;                                     //TI0:   set TI to send first char of UART
312. }
313. //*************************************************************************************************
314. // Programa principal, Realiza la lectura de DHT11, lectura de batería y recibe los datos por UART
315. // Lee la temperatura y humedad interna, realiza un pronóstico aproximado
316. //*************************************************************************************************
317. void main(){                                       //Función principal
318.    configuraUART();                                //Función que configura UART
319.    lcd_init();                                     //Inicializa LCD
320.    lcd_clear();                                    //Borra LCD
321.    saludoPantalla();                               //Función de saludo de pantalla
322.    while(1){                                       //Loop principal infinito
323.       leeDHT11();                                  //Función que lee DHT11
324.       delay_ms(2000);                              //Delay de Actualización
325.       impPantalla();                               //Imprime pantalla LCD
326.       delay_ms(2000);                              //Delay de Actualización
327.    }
328. }
```

Firmware Exterior:

```
1.  //*************************************************************************************************
2.  // Programa para la estación meteorológica externa inalámbrica
3.  // El programa envía Temperatura, Humedad y Luminosidad cada 30 segundos
4.  // El sistema entra en bajo consumo cuando mientras no realiza mediciones
5.  // El enlace se realiza mediante una comunicación ASK OOK UHF sobre UART Invertido a 600bps
6.  // Microcontrolador Silabs C8051F832
7.  //*************************************************************************************************
8.  #include <REG51F800.H>                              //Header de Silabs C8051F832
9.  #include <intrins.h>                                //Biblioteca para funciones de assembler para el Delay
10. #include <stdio.h>                                  //Biblioteca estandar de entrada y salida
11. //*************************************************************************************************
12. // Delay Bloqueante Micro Segundos
13. //*************************************************************************************************
14. void delay_us(unsigned int us_count){               //Función para delay de micro-segundos
15.      int t=0;
16.    while(us_count!=0){                             //Mientras que el contador es distinto de cero
17.       for(t=0;t<16;t++){                           //16MIPS dividido en 16 para 1us
18.          _nop_();                                  //Ejecuta función NOP de ensamblador
19.       }
```

```
20.       us_count--;                                           //Decremento del valor de delay
21.    }
22. }
23. //**************************************************************************************************
24. // Delay Bloqueante Mili Segundos
25. //**************************************************************************************************
26. void delay_ms(unsigned int us_count){                       //Función para delay de micro-segundos
27.    while(us_count!=0){                                     //Mientras que el contador es distinto de cero
28.       delay_us(1000);                                      //Ejecuta función delay micro segundos
29.       us_count--;                                          //Decremento del valor de delay
30.    }
31. }
32. //**************************************************************************************************
33. // Trama DHT11 - Según Datasheet
34. // ____             ____     ____                          ___
35. //     |___________|    |___|    |.....................................|
36. //     |   18ms    |    |80us|80us|              5x8 bit               |
37. //     |    PIC    |    |                  DHT11 respuesta             | Fin de
38. //     |  request  |    |  2x80us, intRH, decRH, intT, decT, checksum   | Trama
39. // ____          ________
40. //     |________|        |...                        0 bit
41. //      <-50us-> <-27us->
42. // ____          ____________________________
43. //     |________|                          |...       1 bit
44. //      <-50us-> <---------70us---------->
45. //**************************************************************************************************
46. sbit DATA = P2^0;                                          //Pin del bus de un hilo para el DHT11
47. unsigned int trama[5];                                     //Vector donde se alojan los datos
48. //**************************************************************************************************
49. // Función de recepción de Byte
50. // Lee el valor leído en la trama y lo separa realizando shift
51. // Retorna el valor en forma de byte, es utilizado en la función recibeDato()
52. //**************************************************************************************************
53. unsigned int recibeByte(){                                  //Función que recibe un Byte
54.    unsigned int valorLeido = 0;                            //Valor de retorno de la función
55.    int i=0;                                                //Inicialización del índice
56.    for(i=0; i<8; i++){                                     //Iteración para recepción de bits
57.       valorLeido <<= 1;                                    //Registro de desplazamiento de bits
58.       while(DATA==0);                                      //Espera a DATA = 0
59.       delay_us(30);                                        //Demora de 30us (Del Datasheet)
60.       if(DATA==1){                                         //Pregunta si DATA = 1
61.          valorLeido |= 1;                                  //Realiza toggle del valor leído
62.       }
63.       while(DATA==1);                                      //Espera a DATA = 1
64.    }
65.    return valorLeido;                                      //Retorna el valor leído
66. }
67. //**************************************************************************************************
68. // Función de recepción de dato para el DHT11
69. // Recibe los valores de temperatura y humedad (parte entera y decimales por separado)
70. // Recibe el checksum enviado por el DHT11 y lo compara con el leído en el programa
71. //**************************************************************************************************
72. unsigned int recibeDato(){                                  //Funcion que recibe el Dato
73.    int validacion = 0;                                     //Variable de Validación
74.    int checksum = 0;                                       //Variable de detección de cambios de secuencia
75.    int j=0;                                                //Variable para el lazo for
76.    DATA = 1;                                               //Set DATA = 1
77.    DATA = 0;                                               //Set DATA = 0
78.    delay_ms(18);                                           //Demora de 18ms (Del Datasheet)
79.    DATA = 1;                                               //Set DATA = 1
80.    delay_us(25);                                           //Demora de 25ms (Del Datasheet)
81.    validacion = DATA;                                      //Mueve valor de DATA a Validacion
82.    delay_us(80);                                           //Espera 80us (Del Datasheet)
83.    validacion = DATA;                                      //Mueve valor de DATA a Validacion
84.    if(!validacion){                                        //Si Validacion = 0, Error de secuencia
85.       printf( "Error en Checksum \r");                     //Muestra leyenda de error
86.    }
87.    delay_us(80);                                           //Espera 80us (Del Datasheet)
88.    for(j=0; j<5; j++){                                     //Lazo de carga de bytes de datos
89.       trama[j] = recibeByte();                             //Carga del vector de datos
90.    }
91.    DATA = 1;                                               //Set DATA = 1
```

```
92.     for(j=0; j<4; j++){                                    //Lazo de carga de bytes de verificación
93.         checksum += trama[j];                              //Carga de bytes de verificación
94.     }
95.     if(checksum == trama[4]){                              //Si la secuencia de verificación es correcta
96.        return 0;                                           //Se retorna 0 y se realiza la lectura
97.     }
98.  }
99.  //*****************************************************************************************************
100. // Función que realiza la lectura del sensor LDR y acondiciona el valor
101. //*****************************************************************************************************
102. int lecturaLDR(void){                                     //Función que realiza la lectura del ADC para el LDR
103.    unsigned int adval;                                    //Variable donde se guarda el valor del ADC
104.    AD0BUSY = 1;                                           //Inicia conversión de ADC
105.    while (AD0BUSY);                                       //Espera que finalice la conversión
106.    adval = ADC0L + (ADC0H << 8);                          //Lee el ADC de 0 a 511
107.    return(adval);                                         //Retorna el valor del ADC canal 0
108. }
109. //*****************************************************************************************************
110. // Función que realiza el parse de datos y armado del payload para enviar por UART
111. //*****************************************************************************************************
112. #define HEADER 200                                        //Definición de valor Header para el payload
113. static char payload[5];                                   //Variable donde se aloja el payload
114. void enivaRF(void){                                       //Declaración de función para enviar datos
115.    payload[0]=HEADER;                                     //Carga el Header en byte 0 del payload
116.    payload[4]=payload[1]+payload[2]+payload[3];            //Suma de los tres datos y carga el byte 4 del payload
117.    putchar(payload[0]);                                   //Envía el byte 0 del payload por UART
118.    delay_ms(50);                                          //Delay de espera entre bytes enviados por UART
119.    putchar(payload[1]);                                   //Envía el byte 1 del payload por UART
120.    delay_ms(50);                                          //Delay de espera entre bytes enviados por UART
121.    putchar(payload[2]);                                   //Envía el byte 2 del payload por UART
122.    delay_ms(50);                                          //Delay de espera entre bytes enviados por UART
123.    putchar(payload[3]);                                   //Envía el byte 3 del payload por UART
124.    delay_ms(50);                                          //Delay de espera entre bytes enviados por UART
125.    putchar(payload[4]);                                   //Envía el byte 4 del payload por UART
126.    delay_ms(50);                                          //Delay de espera entre bytes enviados por UART
127.    printf("\r");                                          //Envía carácter retorno de línea como final de payload
128. }
129. //*****************************************************************************************************
130. // Configuracion puerto serie a 9600 baud con cristal externo de 16MHz.
131. //*****************************************************************************************************
132. void configuraUART(void){
133.    SCON0  = 0x50;                                         //SCON0: mode 1, 8-bit UART, enable rcvr
134.    TMOD  |= 0x20;                                         //TMOD:  timer 1, mode 2, 8-bit reload
135.    TH1    = 76;                                           //TH1:   reload value for 600 baud @ 16MHz
136.    TR1    = 1;                                            //TR1:   timer 1 run
137.    TI0    = 1;                                            //TI0:   set TI to send first char of UART
138. }
139. //*****************************************************************************************************
140. // Configuración ADC pin P0.0
141. //*****************************************************************************************************
142. void configuraADC(void){
143.    #define VREF 3                                         //Tensión de referencia interna
144.    P0MDIN = 0xFE;                                         //Selecciona pin P0.0 como canal ADC
145.    P0SKIP = 0x01;                                         //Decodificación Crossbar
146.    ADC0CF = 0xF8;                                         //Habilita ADC 0
147.    ADC0CN = 0x80;                                         //Habilita conversión en registro
148. }
149. //*****************************************************************************************************
150. // Programa principal, Realiza la lectura de DHT11, lectura del LDR y Envía los datos por UART
151. // Realiza el timer de Sleep y WakeUp
152. //*****************************************************************************************************
153. void main(){                                             //Función principal
154.     configuraUART();                                     //Función que configura UART
155.     configuraADC();                                      //Función que configura ADC
156.     printf("CONFIGURADO\r");
157.     delay_ms(500);
158.    while(1){                                             //Loop principal repetitivo
159.       if(recibeDato()==0){                               //Consulta si hay dato en la entrada del DHT11
160.          payload[1]=trama[2];                            //Carga el valor de temperatura DHT en byte 1 del payload
161.          payload[2]=trama[0];                            //Carga el valor de humedad DHT en byte 2 del payload
162.          payload[3]=lecturaLDR();                        //Carga el valor de luminosidad en el byte 3 del payload
163.          enivaRF();                                      //Llamado a la función que envía datos
```

```
164.        }
165.        delay_ms(2304);                                    //Delay del timer de sleep*
166.    }
167. }
```

Conclusiones

Se ha logrado implementar una estación meteorológica hogareña de bajo consumo y bajo costo que presenta una propuesta comercial respecto a las competidoras de mismas prestaciones.

- Se implementó el código necesario para realizar el proyecto sobre core 8051 bajo una plataforma moderna que opera en 16MHz y 16MIPS.
- Se realizó todo el tratamiento de la trama, preámbulo, payload, checksum y parseo de datos para realizar el enlace de telemetría entre los dos dispositivos.
- Se analizó la velocidad de transferencia de datos sobre UART para lograr la mejor comunicación y alcance sobre módulos de RF de bajo costo.
- Se comprobó el diseño de la antena con la ayuda de instrumental adecuado “VNA” para su correcta adaptación.

Bibliografía

[1] “Manual 8051 Keil”,
http://www.keil.com/support/man/docs/is51/
[2] “Manual Eagle”,
http://hades.mech.northwestern.edu/images/b/b4/Eagle_Manual.pdf
[3] “Norma ATEX”,
http://www.insht.es/InshtWeb/Contenidos/Normativa/GuiasTecnicas/Ficheros/ATM%C3%93SFERAS%20EXPLOSIVAS.pdf
[4] “Norma IPX3”,
http://hades.mech.northwestern.edu/images/b/b4/Eagle_Manual.pdfhttp://www.f2i2.net/documentos/lsi/rbt/guias/guia_bt_anexo_1_sep03R1.pdf
[5] “Norma IPC610”,
http://www.ipc.org/TOC/IPC-A-610E-Spanish.pdf
[6] “Datasheet DHT11”,
https://akizukidenshi.com/download/ds/aosong/DHT11.pdf
[7] “Datasheet Modulos Transmisor y Receptor RF UHF”,
https://4.imimg.com/data4/AJ/NM/MY-1833510/rf-transmitter-receiver-pair-433-mhz-ask.pdf
[8] “Datasheet LDR”,
http://kennarar.vma.is/thor/v2011/vgr402/ldr.pdf
[9] “Datasheet C8051F832 Silabs”,
https://www.silabs.com/documents/public/data-sheets/C8051F80x-83x.pdf
[10] “Datasheet TP4056”,
https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf