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.

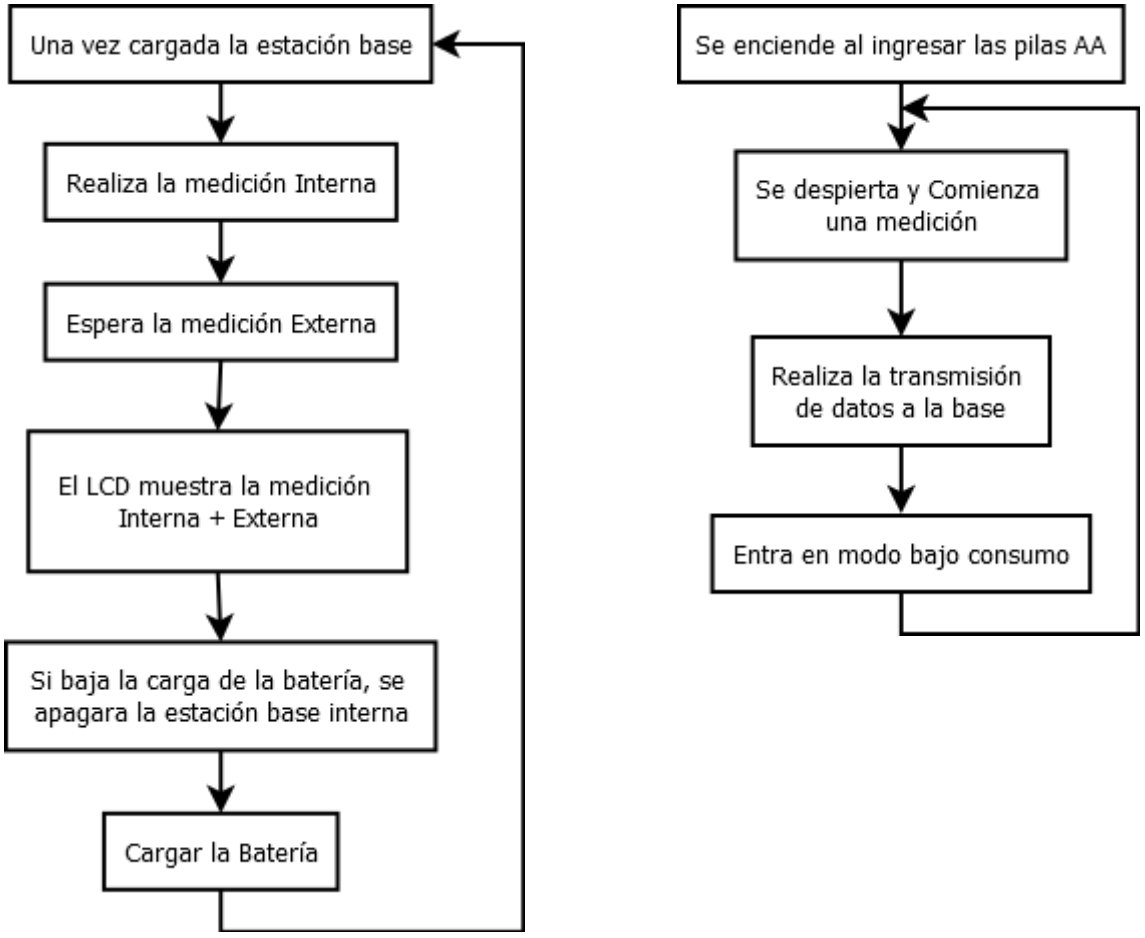
El proyecto surge de la idea de practicar la programación en lenguaje C, los protocolos de comunicación serie y el transporte de los datos vía enlaces inalambricos, en este proyecto se abordará la estructura necesaria para realizar una trama de datos básica, su payload y verificación.
También se realizará 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:

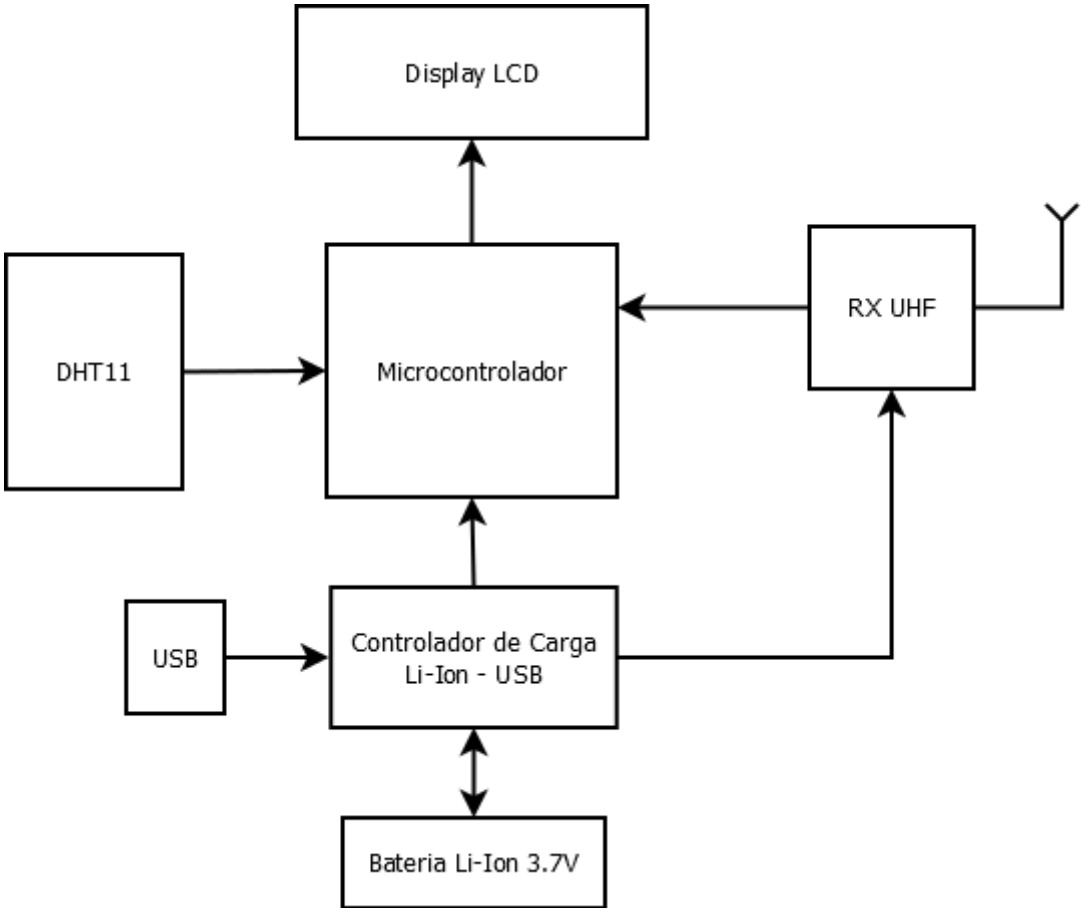


Diagrama de bloques de Hardware Exterior:

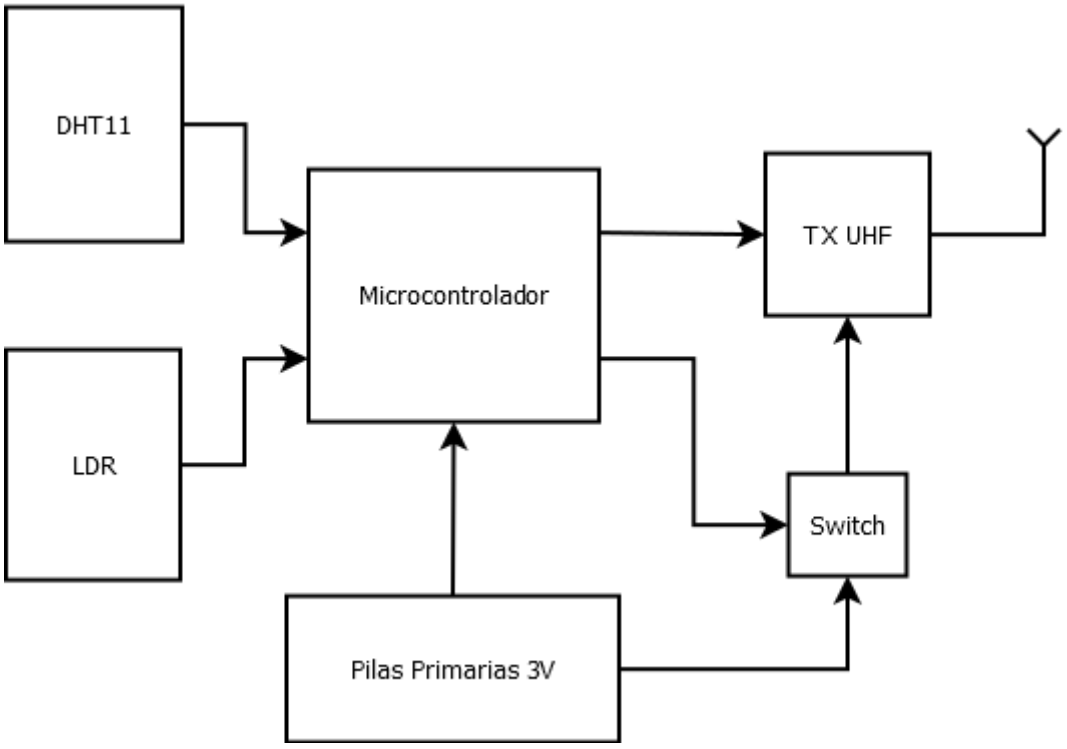
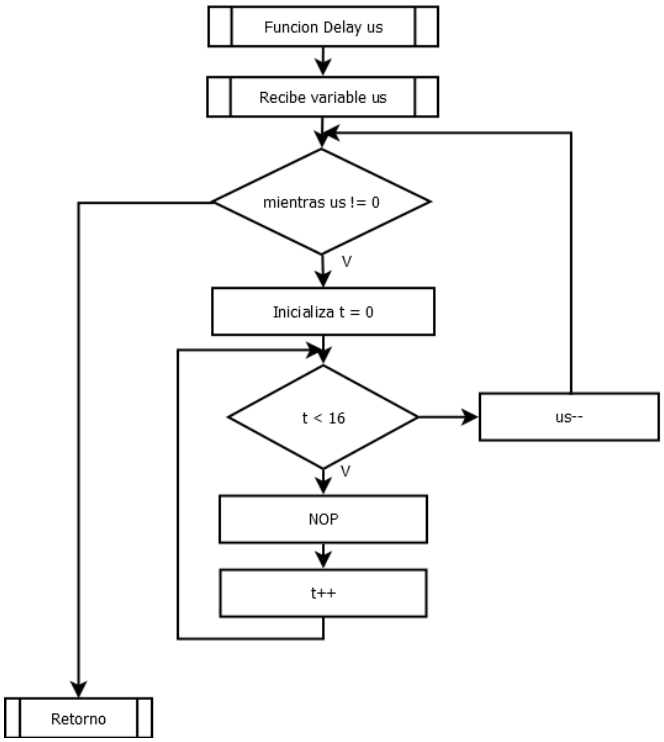
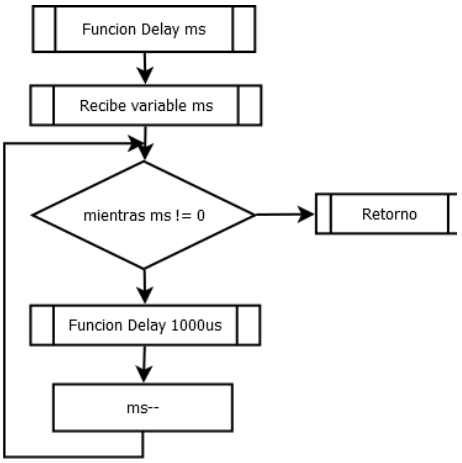
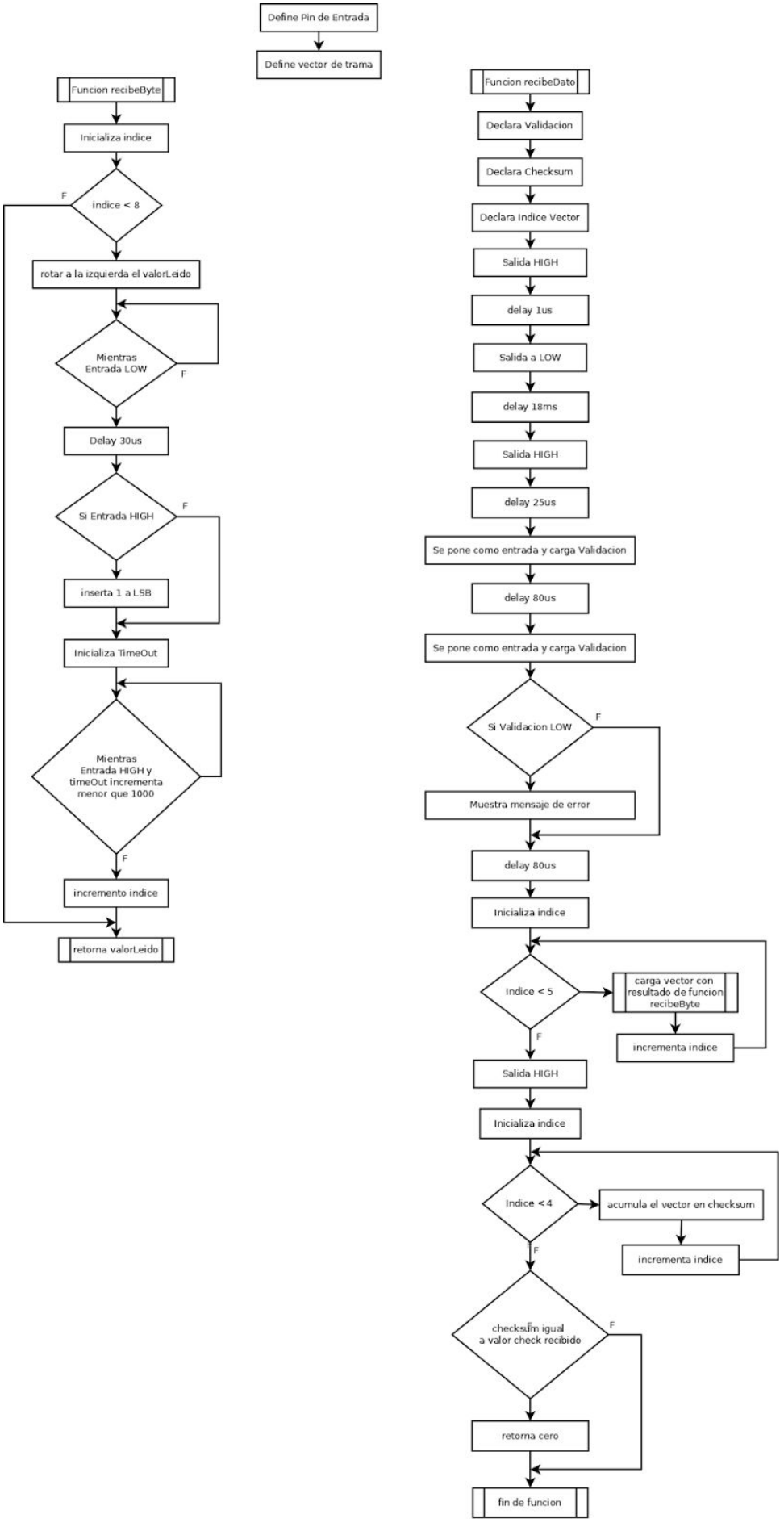
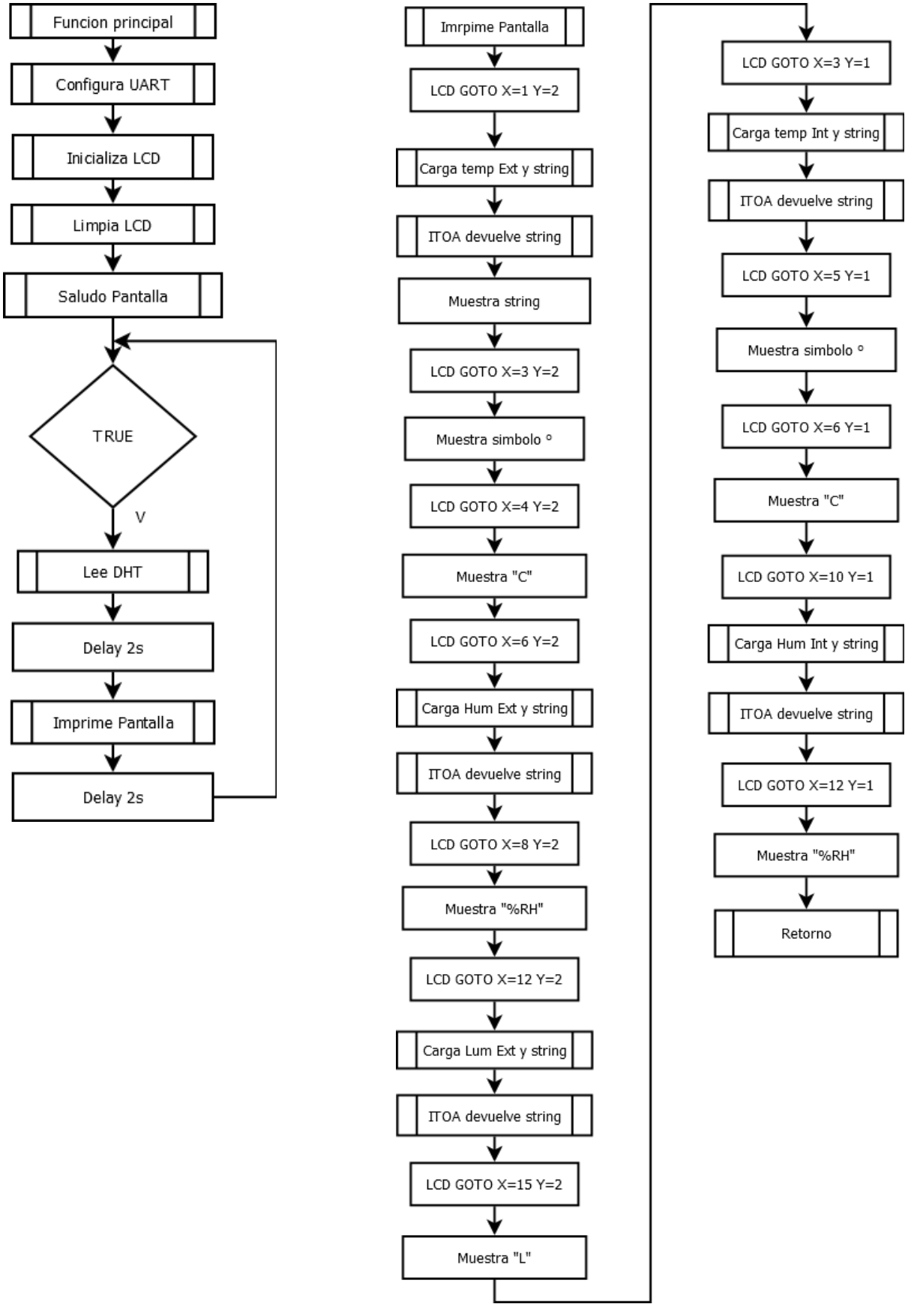


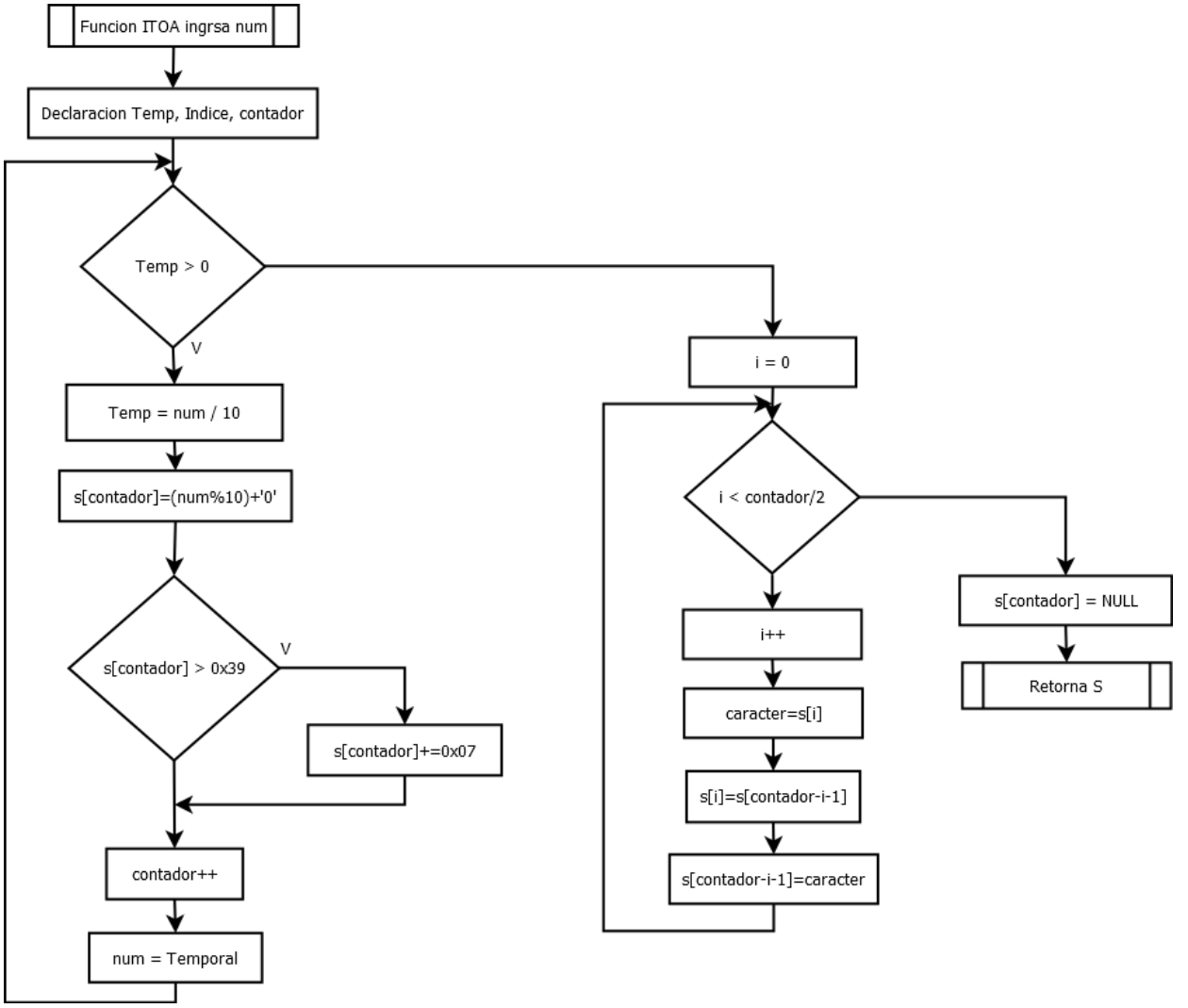
Diagrama de bloques de Firmware Interior:

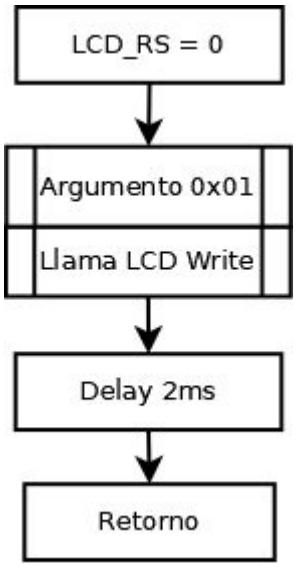


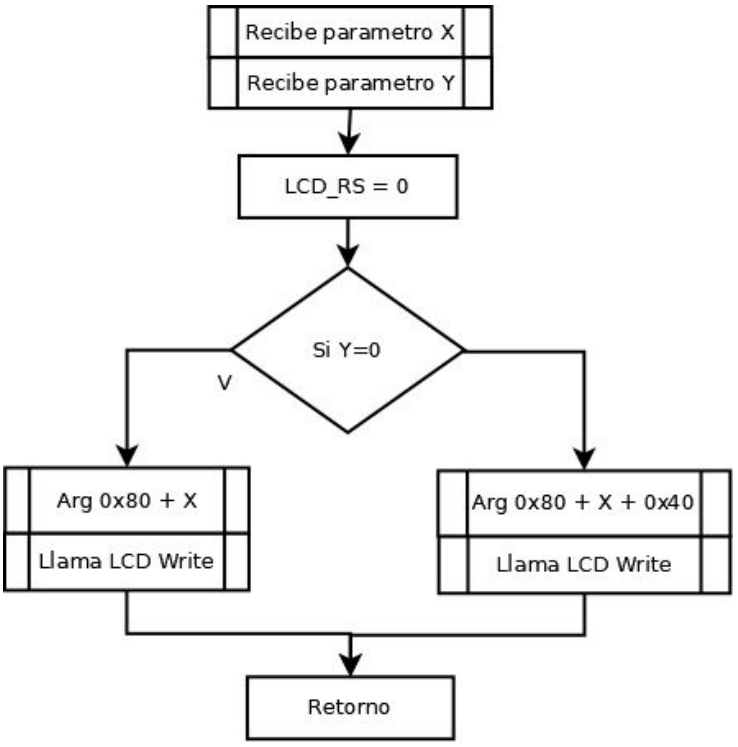


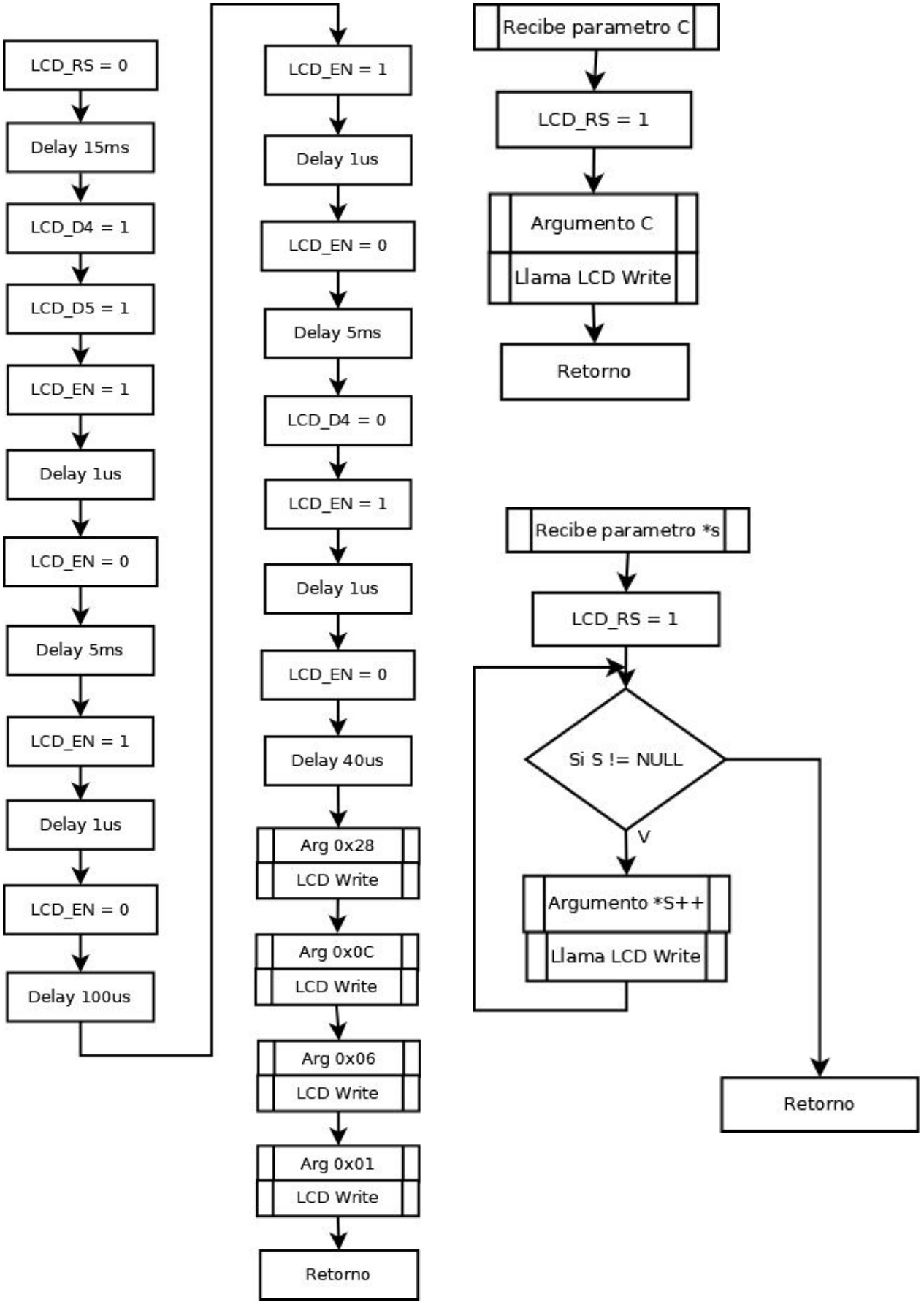

Define Pin de Entrada
Define vector de trama
Funcion recibeByte
Inicializa indice
indice < 8
F
rotar a la izquierda el valorLeido
Mientras Entrada LOW
F
Delay 30us
Si Entrada HIGH
F
inserta 1 a LSB
Inicializa TimeOut
Mientras Entrada HIGH y timeOut incrementa menor que 1000
F
incremento indice
retorna 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

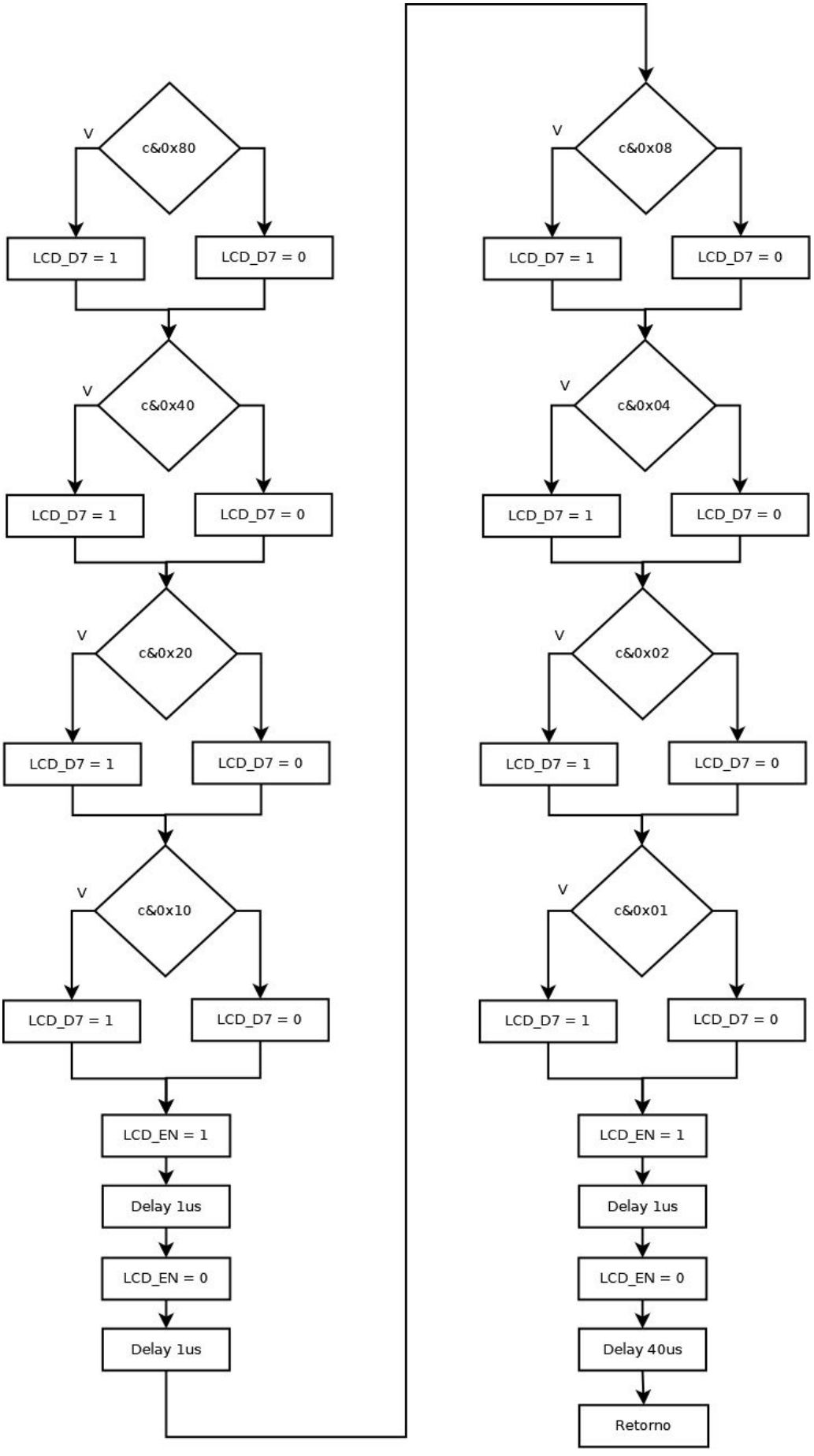

Funcion principal
Configura UART
Inicializa LCD
Limpia LCD
Saludo Pantalla
TRUE
V
Lee DHT
Delay 2s
Imprime Pantalla
Delay 2s
Imrpime Pantalla
LCD GOTO X=1 Y=2
Carga temp Ext y string
ITOA devuelve string
Muestra string
LCD GOTO X=3 Y=2
Muestra simbolo °
LCD GOTO X=4 Y=2
Muestra "C"
LCD GOTO X=6 Y=2
Carga 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

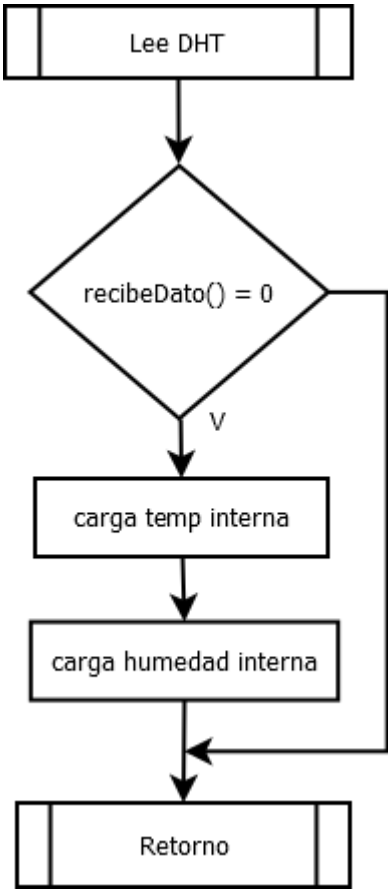

Funcion ITOA ingrsa num
Declaracion Temp, Indice, contador
Temp > 0
V
Temp = num / 10
s[contador]=(num%10)+'0'
s[contador] > 0x39
V
s[contador]+=0x07
contador++
num = Temporal
i = 0
i < contador/2
i++
caracter=s[i]
s[i]=s[contador-i-1]
s[contador-i-1]=caracter
s[contador] = NULL
Retorna S

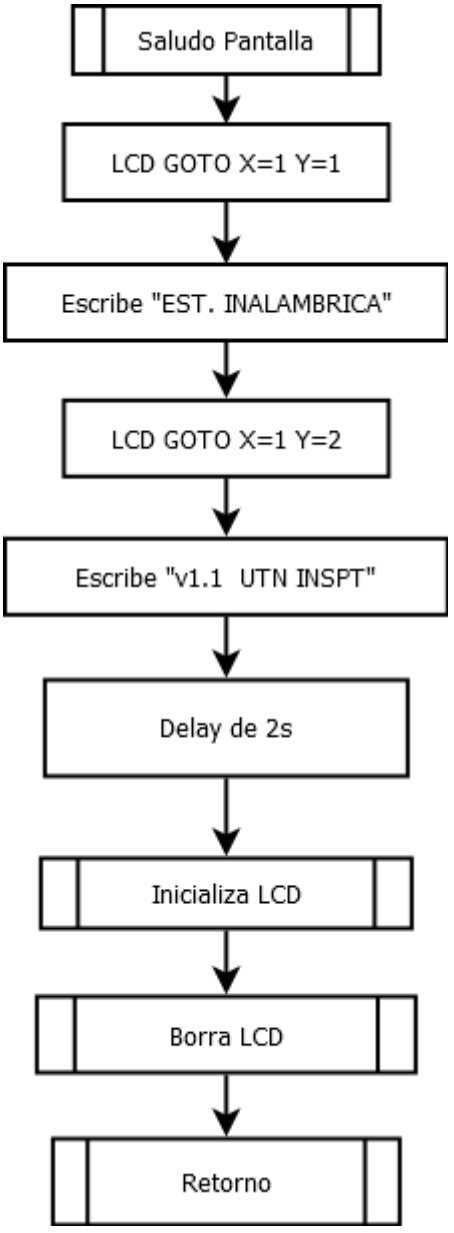

LCD_RS = 0
Argumento 0x01
Llama LCD Write
Delay 2ms
Retorno


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

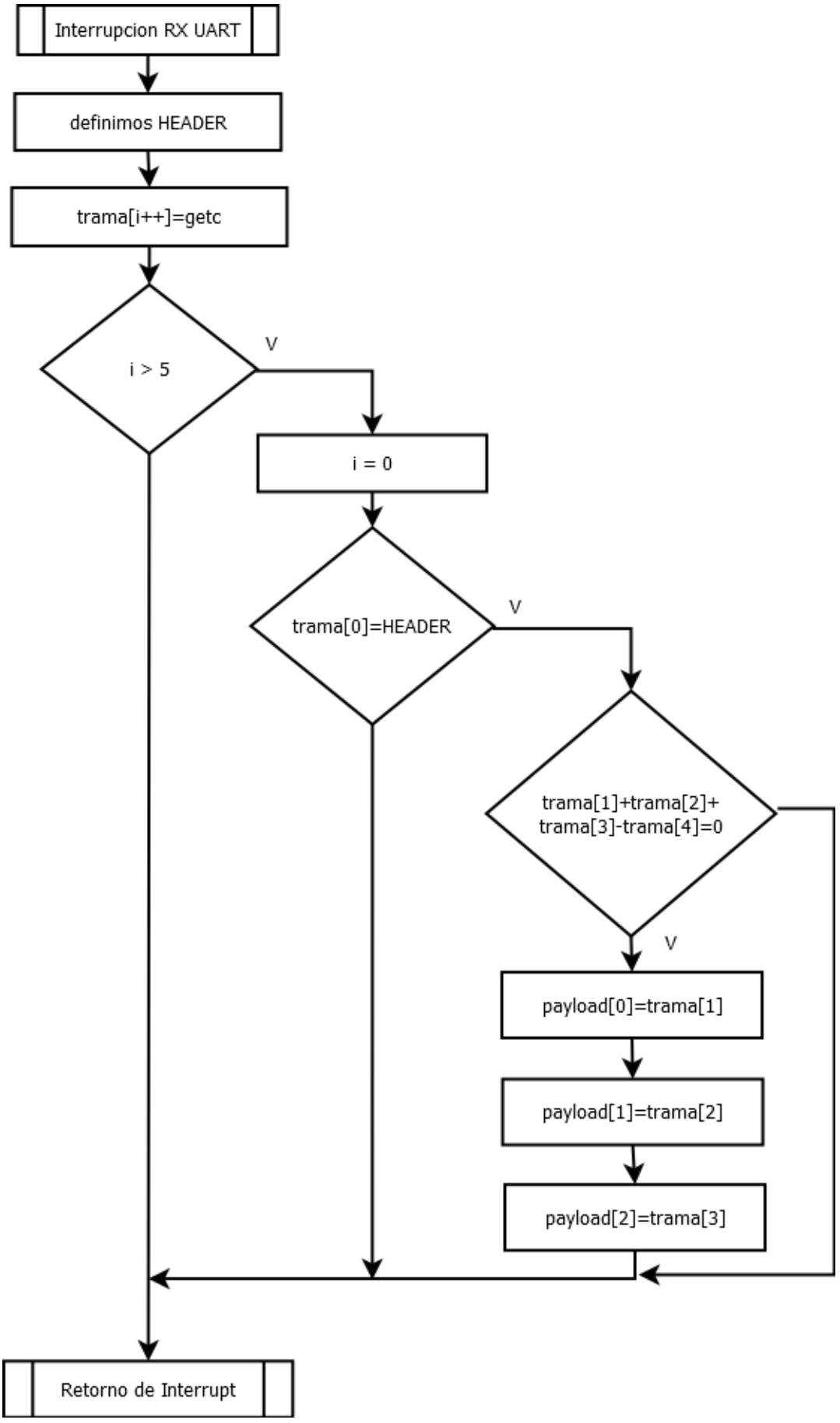
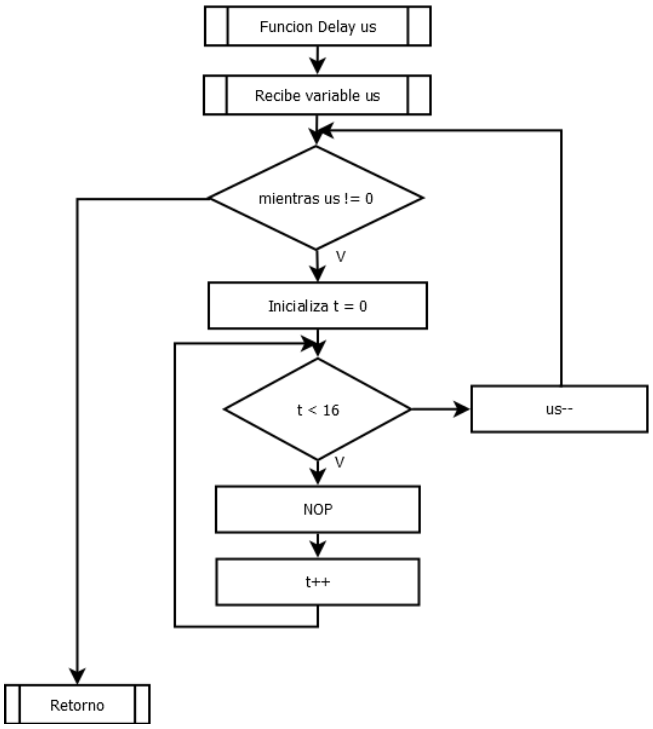
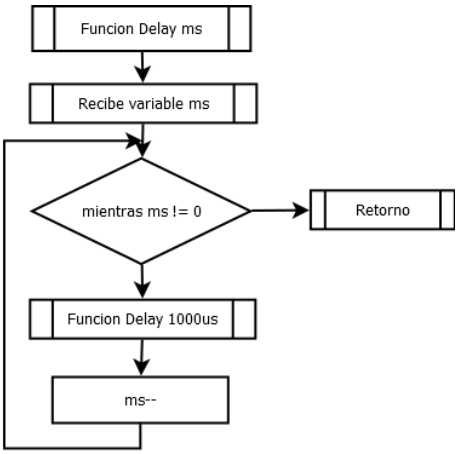
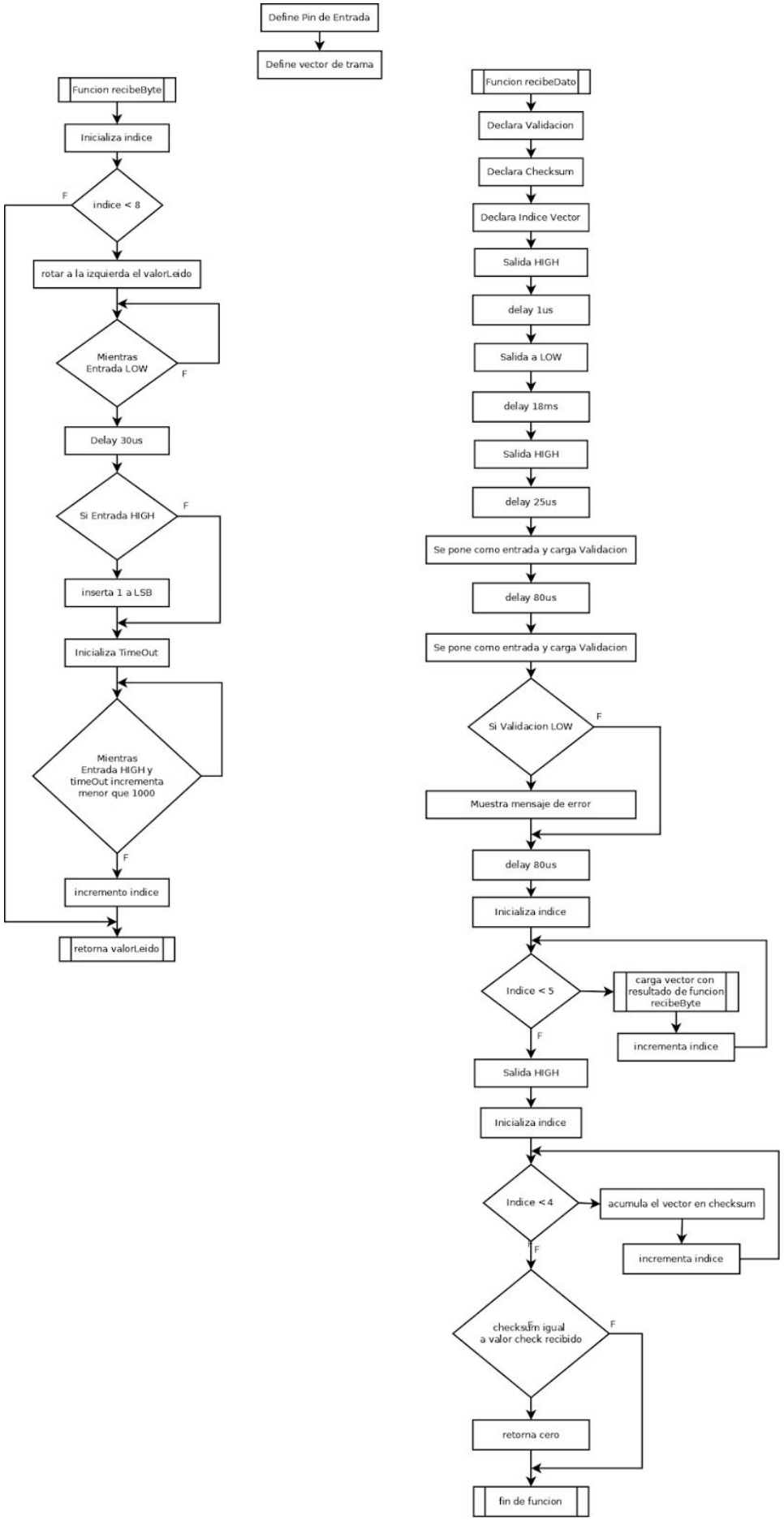
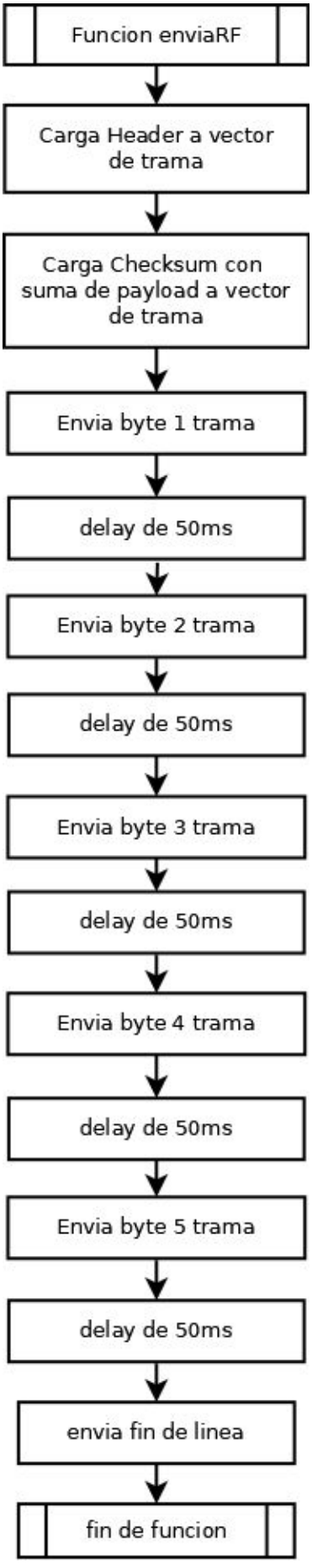

Interrupcion RX UART
definimos HEADER
trama[i++]=getc
i > 5
V
i = 0
trama[0]=HEADER
V
trama[1]+trama[2]+
trama[3]-trama[4]=0
V
payload[0]=trama[1]
payload[1]=trama[2]
payload[2]=trama[3]
Retorno de Interrupt

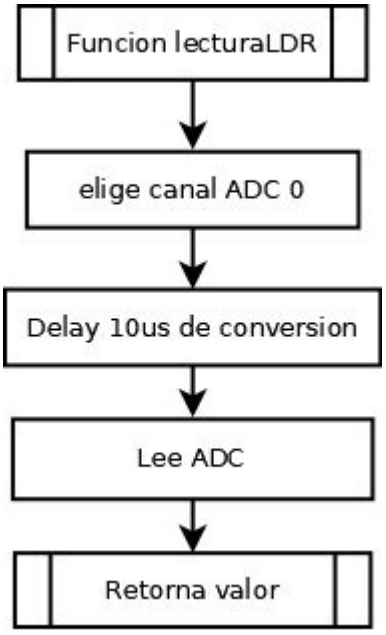
Diagrama de bloques de Firmware Exterior:

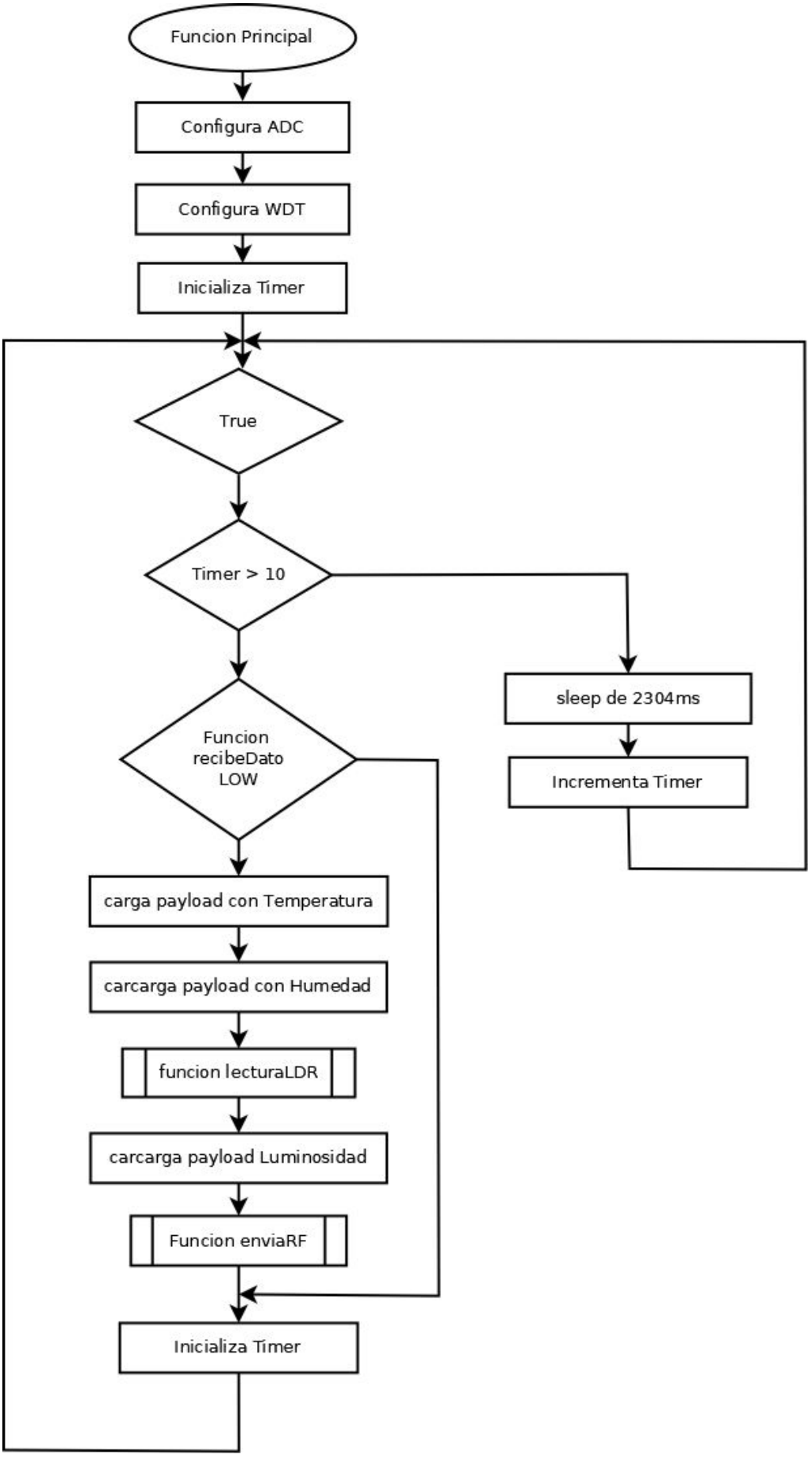





Define Pin de Entrada
Define vector de trama
Funcion recibeByte
Inicializa indice
indice < 8
F
rotar a la izquierda el valorLeido
Mientras Entrada LOW
F
Delay 30us
Si Entrada HIGH
F
inserta 1 a LSB
Inicializa TimeOut
Mientras Entrada HIGH y timeOut incrementa menor que 1000
F
incremento indice
retorna 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.

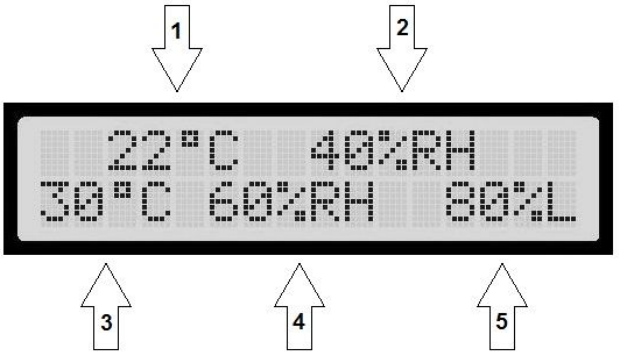
- Desarrollo de Bibliotecas necesarias (Estandarizado)
- Pruebas Unitarias de cada Biblioteca con el hardware asociado
- Desarrollo de programa principal
- Manejo de errores
- Creación de diagramas de flujo o estado
- Pruebas Integrados del firmware con todas las Bibliotecas y devices

- ➔ 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:

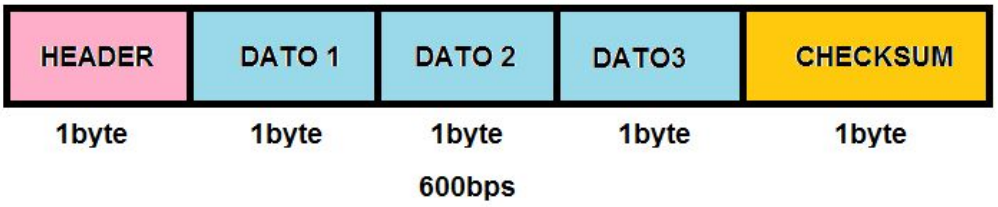


Diagrama Electronico Interior:

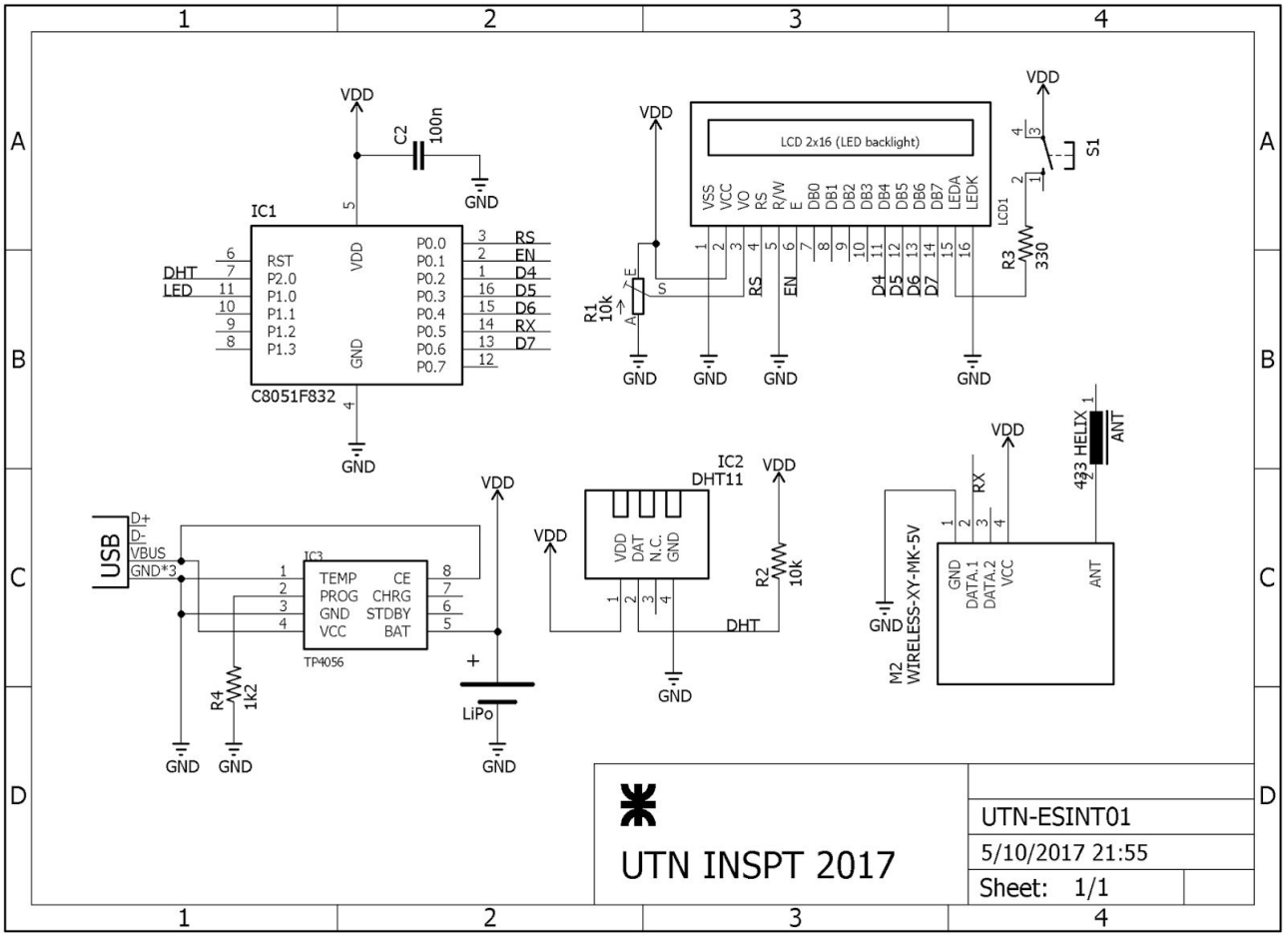
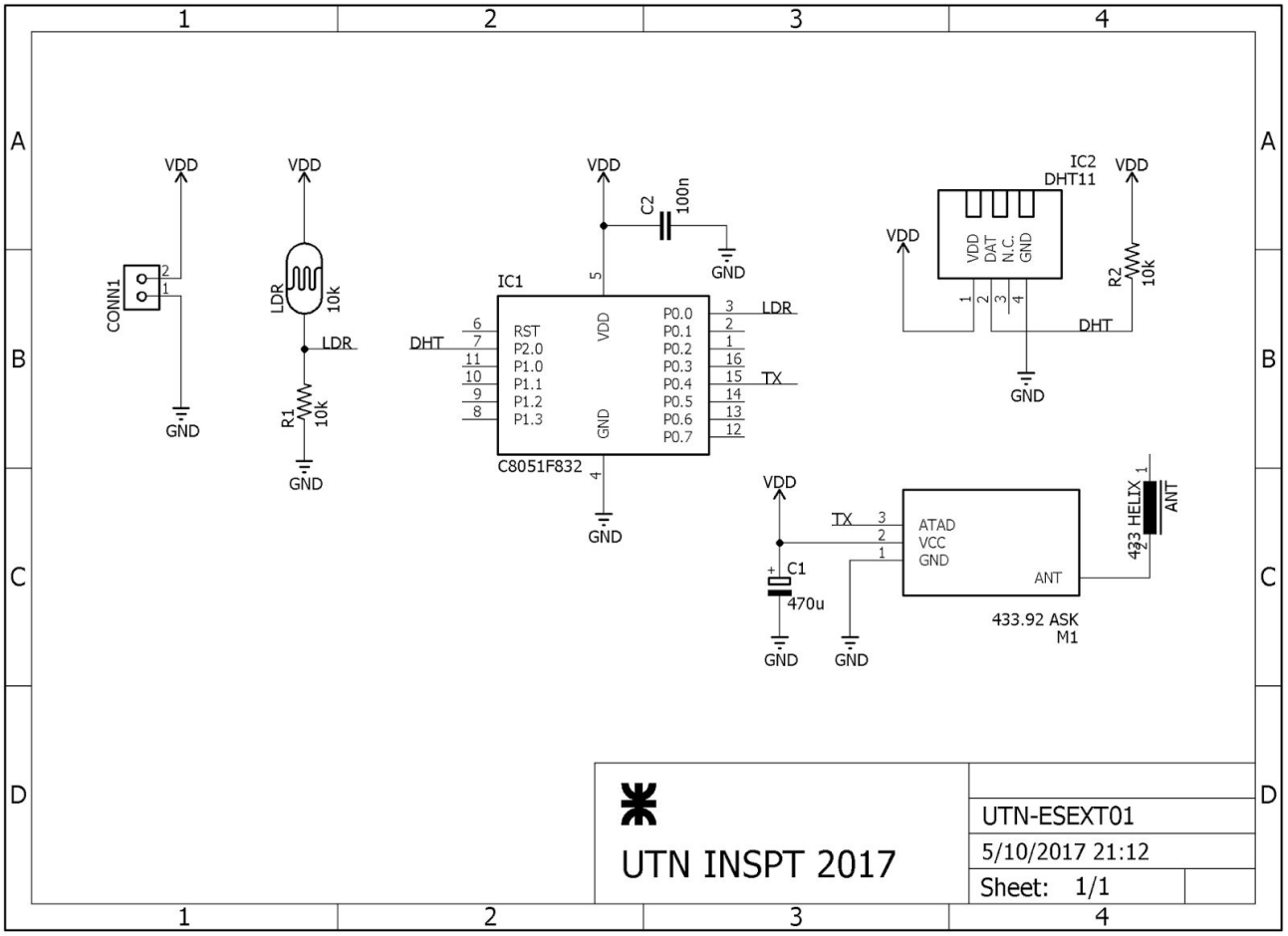
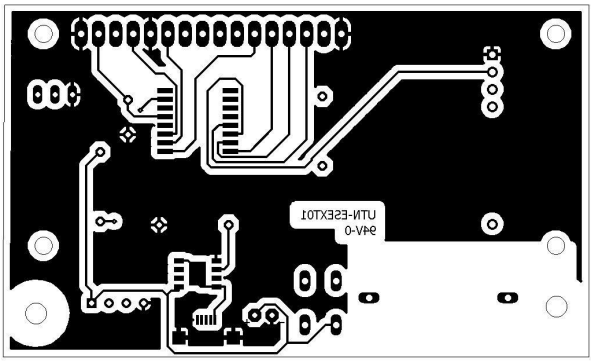
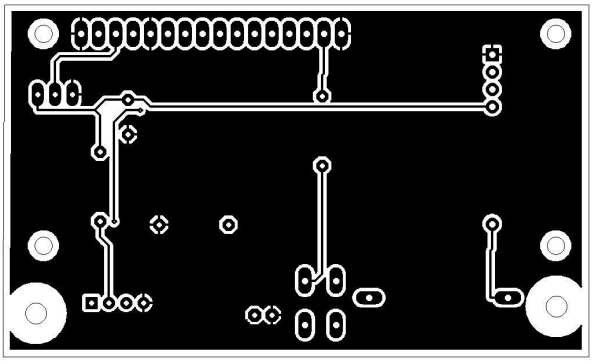


Diagrama Electronico Exterior:

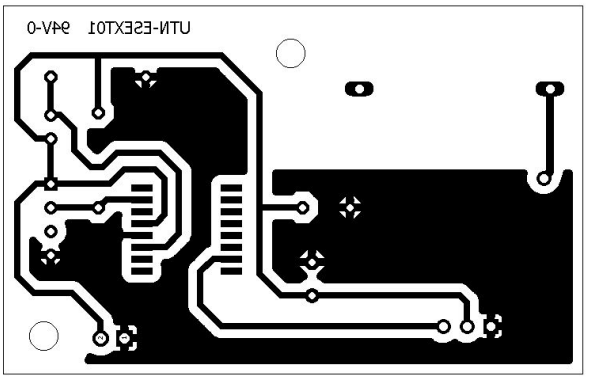


Circuito Impreso Interior:

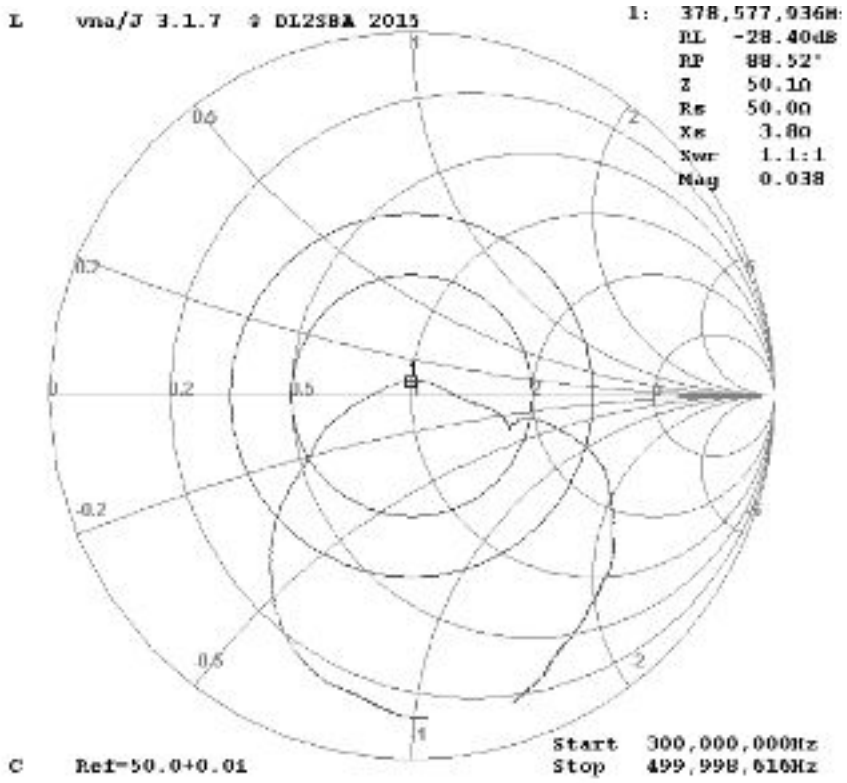




Circuito Impreso Exterior:



Análisis de antena tipo ANT-433-HETH:



Código del proyecto

Firmware Interior (Silabs C8051F832):

```
1.  //*****************************************************************************************************
2.  // Programa para la estación meteorológica Interna inalámbrica
3.  // El programa recibe Temperatura, Humedad y Luminosidad por interrupción
4.  // El sistema mide Temperatura y Humedad interior
5.  // El enlace se realiza mediante una comunicación ASK OOK UHF sobre UART Invertido a 600bps
6.  // Se muestra medición interna y externa en un LCD de 2x16
7.  // Microcontrolador Silabs C8051F832
8.  //*****************************************************************************************************
9.  #include <REG51F800.H>
10. #include <reg52.h>
11. #include <intrins.h>
12. #include <stdio.h>
13. //*****************************************************************************************************
14. // Variables
15. //*****************************************************************************************************
16. #define HEADER 200                                      //Definición de valor Header para el payload
17. sbit DATA = P1^0;                                       //Pin del bus de un hilo para el DHT11
18. static int datoInt[2];                                  //Variable donde se alojan los datos internos
19. static int payload[5];                                  //Variable donde se aloja el payload
20. static int datoExt[3];                                  //Variable donde se alojan los datos externos
21. //*****************************************************************************************************
22. // Configuración de pines del LCD
23. //*****************************************************************************************************
24. extern bit RS;                                          //Definición de pines
25. extern bit EN;                                          //Definición de pines
26. extern bit D4;                                          //Definición de pines
27. extern bit D5;                                          //Definición de pines
28. extern bit D6;                                          //Definición de pines
29. extern bit D7;                                          //Definición de pines
30. sbit RS = P2^0;                                         //Definición de pines
31. sbit EN = P2^1;                                         //Definición de pines
32. sbit D4 = P2^4;                                         //Definición de pines
33. sbit D5 = P2^5;                                         //Definición de pines
34. sbit D6 = P2^6;                                         //Definición de pines
35. sbit D7 = P2^7;                                         //Definición de pines
36. //*****************************************************************************************************
37. // delay_us Bloqueante Micro Segundos
38. //*****************************************************************************************************
39. void delay_us(unsigned int us_count){                   //Función para delay_ms de micro-segundos
40.    int t=0;
41.    while(us_count!=0){                                  //Mientras que el contador es distinto de cero
42.       for(t=0;t<16;t++){                                //16MIPS dividido en 16 para 1us
43.          _nop_();                                       //Ejecuta función NOP de ensamblador
44.       }
45.       us_count--;                                       //Decremento del valor de delay_ms
46.    }
47. }
48. //*****************************************************************************************************
49. // delay_ms Bloqueante Mili Segundos
50. //*****************************************************************************************************
51. void delay_ms(unsigned int us_count){                   //Función para delay_ms de micro-segundos
52.    while(us_count!=0){                                  //Mientras que el contador es distinto de cero
53.       delay_us(1000);                                   //Ejecuta funcion delay_ms micro segundos
54.       us_count--;                                       //Decremento del valor de delay_ms
55.    }
56. }
57. //*****************************************************************************************************
58. // Funcion que escribe en puerto
59. //*****************************************************************************************************
60. void lcdPort(char a){                                   //Funcion para escribir el puerto en 4bit
61.     if(a&1) D4=1;                                      //Evalúa si a AND 0001, D4 a HIGH
62.       else  D4=0;                                      //Caso contrario D4 a LOW
63.     if(a&2) D5=1;                                      //Evalúa si a AND 0010, D5 a HIGH
64.       else  D5=0;                                      //Caso contrario D5 a LOW
```

```
65.     if(a&4) D6=1;                                   //Evalúa si a AND 0100, D6 a HIGH
66.       else  D6=0;                                   //Caso contrario D6 a LOW
67.     if(a&8) D7=1;                                   //Evalúa si a AND 1000, D7 a HIGH
68.       else  D7=0;                                   //Caso contrario D7 a LOW
69.   }
70.   //*************************************************************************************************************
71.   // Funcion que envia comando
72.   //*************************************************************************************************************
73.   void lcdCmd(char a){                              //Función para realizar comandos en LCD
74.     RS=0;                                           //Control RS a LOW
75.     lcdPort(a);                                     //Llamado a lcdPort
76.     EN=1;                                           //Control EN a HIGH
77.     delay_ms(5);                                    //Espera de 5ms
78.     EN=0;                                           //Control EN a LOW
79.   }
80.   //*************************************************************************************************************
81.   // Borrado de LCD
82.   //*************************************************************************************************************
83.   lcdClear(){                                       //Función para borrar el LCD
84.     lcdCmd(0);                                      //Se envia comando LOW
85.     lcdCmd(1);                                      //Se envía comando HIGH
86.   }
87.   //*************************************************************************************************************
88.   // Función para ir a la posición específica
89.   //*************************************************************************************************************
90.   void lcdGotoxy(char b, char a){                   //Función para posicionar el cursor
91.     char temp,z,y;                                  //Declaración de variables
92.     if(a==1){                                       //Si se encuentra en renglón 1
93.       temp=0x80+b;                                  //Se incrementa la columna
94.        z=temp>>4;                                   //Se realiza el desplazamiento
95.        y=temp&0x0F;                                 //Se aplica la máscara de bits
96.        lcdCmd(z);                                   //Envía comandos
97.        lcdCmd(y);                                   //Envía comandos
98.     }else{                                          //Si no está en renglón 1
99.        if(a==2){                                    //Si se encuentra en renglón 2
100.          temp=0xC0+b;                              //Se incrementa la columna
101.          z=temp>>4;                                //Se realiza el desplazamiento
102.          y=temp&0x0F;                              //Se realiza la máscara
103.          lcdCmd(z);                                //Envía comandos
104.          lcdCmd(y);                                //Envía comandos
105.       }
106.    }
107.  }
108.  //*************************************************************************************************************
109.  // Inicializa LCD
110.  //*************************************************************************************************************
111.  void lcdInit(){                                   //Función para inicializar el LCD
112.    lcdPort(0x00);                                  //Se envía 0000 0000
113.    delay_ms(200);                                  //Delay de 200ms
114.    lcdCmd(0x03);                                   //Se envía 0000 0011
115.    delay_ms(50);                                   //Delay de 50ms
116.    lcdCmd(0x03);                                   //Se envía 0000 0011
117.    delay_ms(110);                                  //Delay de 110ms
118.    lcdCmd(0x03);                                   //Se envía 0000 0011
119.    lcdCmd(0x02);                                   //Se envía 0000 0010
120.    lcdCmd(0x02);                                   //Se envía 0000 0010
121.    lcdCmd(0x08);                                   //Se envía 0000 1000
122.    lcdCmd(0x00);                                   //Se envía 0000 0000
123.    lcdCmd(0x0C);                                   //Se envía 0000 1100
124.    lcdCmd(0x00);                                   //Se envía 0000 0000
125.    lcdCmd(0x06);                                   //Se envía 0000 0110
126.  }
127.  //*************************************************************************************************************
128.  // Función para escribir caracter
129.  //*************************************************************************************************************
130.  void lcdWriteChar(char a){                        //Funcion para escribir un caracter
131.    char temp,y;                                    //Declaración de variables
132.    temp=a&0x0F;                                    //Se realiza la máscara de bits 0000 1111
133.    y=a&0xF0;                                       //Se realiza la máscara de bits 0000 1111
134.    RS=1;                                           //Control RS a HIGH
135.    lcdPort(y>>4);                                  //Se desplaza 4 lugares a la derecha
136.    EN=1;                                           //Control EN a HIGH
```

```
    delay_ms(5);                                    //Delay de 5ms
    EN=0;                                           //Control EN a LOW
    lcdPort(temp);                                  //Se envía el valor
    EN=1;                                           //Control EN a HIGH
    delay_ms(5);                                    //Delay de 5ms
    EN=0;                                           //Control EN a LOW
}
//*************************************************************************************************
// Función para escribir string
//*************************************************************************************************
void lcdWriteString(char *a){                       //Funcion para escribir string
   int i;                                           //Declaración de variables
   for(i=0;a[i]!='\0';i++)                          //Iteración de caracteres hasta el NULL
      lcdWriteChar(a[i]);                           //Se carga cada caracter para formar string
}
//*************************************************************************************************
// 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
}
//*************************************************************************************************
// Funcion para inicializar el puerto serie 9600 @ 11.059MHz
//*************************************************************************************************
void serialInit(void){                              //Función para configurar UART
    TMOD=0x20;                                      //Timer 1 en modo 2 - Auto recarga para generar Baudrate
    SCON=0x50;                                      //Serial modo 1, 8N1
    TH1=0xFD;                                       //Carga el baudrate en el timer a 9600bps
    TR1=1;                                          //Dispara el timer
}
//*************************************************************************************************
// Función Interrupción UART que realiza el parse de datos, validación de Header y Checksum
// Si validación y Checksum son validos, carga el vector DATO para ser utilizado
//*************************************************************************************************
char i=0;                                           //Variable para el contador de bytes de entrada
void serial_ISR(void) interrupt 4{                  //Función de servicio de interrupción
     if(RI==1){                                     //Si hay dato pendiente en UART
      payload[i++]=SBUF;                            //Guarda byte de entrada en payload. incrementa índice
      RI=0;                                         //Pone a cero el flag
    }
    if(i>5){                                        //Si índice mayor a 5 se asume que se completa el payload
       if(payload[0]==HEADER){                      //Validación Header sea 200 (seteado en el transmisor)
          if(payload[1]+payload[2]+payload[3]-payload[4]==0){//Validación, 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
          }
       }
       i=0;                                         //Una vez completado 5 bytes, se reinicia el contador
    }
    RI=0;                                           //Pone a cero el flag
}
//*************************************************************************************************
// Trama DHT11 - Segun Datasheet
// ____            ____      ____                        ___
```

```
209. //     |___________|    |____|    |.....................................|
210. //     |   18ms    |    |80us|80us|              5x8 bit                |
211. //     |    PIC    |    |                  DHT11 respuesta              | Fin de
212. //     |  request  |    |  2x80us, intRH, decRH, intT, decT, checksum   | Trama
213. // ____          ________
214. //     |________|        |...                        0 bit
215. //      <-50us-> <-27us->
216. // ____          ____________________________
217. //     |________|                          |...       1 bit
218. //      <-50us-> <---------70us---------->
219. //*************************************************************************************************************
220. unsigned int trama[5];                                    //Vector donde se alojan los datos
221. //*************************************************************************************************************
222. // Función de recepción de Byte
223. // Lee el valor leído en la trama y lo separa realizando shift
224. // Retorna el valor en forma de byte, es utilizado en la función recibeDato()
225. //*************************************************************************************************************
226. unsigned int recibeByte(){                                 //Función que recibe un Byte
227.    unsigned int valorLeido = 0;                           //Valor de retorno de la función
228.    int i=0;                                               //Inicialización del índice
229.    for(i=0; i<8; i++){                                    //Iteración para recepción de bits
230.       valorLeido <<= 1;                                   //Registro de desplazamiento de bits
231.       while(DATA==0);                                     //Espera a DATA = 0
232.       delay_us(30);                                       //Demora de 30us (Del Datasheet)
233.       if(DATA==1){                                        //Pregunta si DATA = 1
234.           valorLeido |= 1;                                //Realiza toggle del valor leído
235.       }
236.       while(DATA==1);                                     //Espera a DATA = 1
237.    }
238.    return valorLeido;                                     //Retorna el valor leído
239. }
240. //*************************************************************************************************************
241. // Función de recepción de dato para el DHT11
242. // Recibe los valores de temperatura y humedad (parte entera y decimales por separado)
243. // Recibe el checksum enviado por el DHT11 y lo compara con el leído en el programa
244. //*************************************************************************************************************
245. unsigned int recibeDato(){                                 //Funcion que recibe el Dato
246.    int validacion = 0;                                    //Variable de Validación
247.    int checksum = 0;                                      //Variable de detección de cambios de secuencia
248.    int j=0;                                               //Variable para el lazo for
249.    DATA = 1;                                              //Set DATA = 1
250.    DATA = 0;                                              //Set DATA = 0
251.    delay_ms(18);                                          //Demora de 18ms (Del Datasheet)
252.    DATA = 1;                                              //Set DATA = 1
253.    delay_us(25);                                          //Demora de 25ms (Del Datasheet)
254.    validacion = DATA;                                     //Mueve valor de DATA a Validacion
255.    delay_us(80);                                          //Espera 80us (Del Datasheet)
256.    validacion = DATA;                                     //Mueve valor de DATA a Validacion
257.    if(!validacion){                                       //Si Validacion = 0, Error de secuencia
258.       printf( "Error en Checksum \r");                    //Muestra leyenda de error
259.    }
260.    delay_us(80);                                          //Espera 80us (Del Datasheet)
261.    for(j=0; j<5; j++){                                    //Lazo de carga de bytes de datos
262.        trama[j] = recibeByte();                           //Carga del vector de datos
263.    }
264.    DATA = 1;                                              //Set DATA = 1
265.    for(j=0; j<4; j++){                                    //Lazo de carga de bytes de verificación
266.        checksum += trama[j];                              //Carga de bytes de verificación
267.    }
268.    if(checksum == trama[4]){                              //Si la secuencia de verificación es correcta
269.       return 0;                                           //Se retorna 0 y se realiza la lectura
270.    }
271. }
272. //*************************************************************************************************************
273. // Lee DHT11
274. //*************************************************************************************************************
275. void leeDHT11(void){                                      //Funcion que lee el DHT11
276.    if(recibeDato()==0){                                   //Consulta si hay dato en la entrada del DHT11
277.       datoInt[0]=trama[2];                                //Carga el valor de temperatura DHT en byte 1 del payload
278.       datoInt[1]=trama[0];                                //Carga el valor de humedad DHT en byte 2 del payload
279.    }
280. }
```

```
281. //*****************************************************************************************************
282. // Saludo de Pantalla
283. //*****************************************************************************************************
284. void saludoPantalla(void){                                  //Función que realiza un saludo inicial
285.    lcdGotoxy(1,1);                                         //Posiciona el cursor en la pantalla
286.    lcdWriteString(" Est.Inalambrica");                     //Imprime mensaje renglon 1
287.    lcdGotoxy(1,2);                                         //Posiciona el cursor en la pantalla
288.    lcdWriteString(" v1.1  UTN INSPT");                     //Imprime mensaje renglón 2
289.    delay_ms(2000);                                         //delay_ms de saludo
290.    lcdInit();                                              //Inicializa LCD
291.    lcdClear();                                             //Borra LCD
292. }
293. //*****************************************************************************************************
294. // Impresion de Pantalla
295. //*****************************************************************************************************
296. void impPantalla(void){                                    //Funcion que imprime pantalla
297.    char string[4];                                         //Declaración de vector para mostrar en LCD
298.    lcdGotoxy(1,2);                                         //Posiciona el cursor en la pantalla
299.    itoa(datoExt[0],string);                                //Funcion que convierte entero en ascii
300.    lcdWriteString(string);                                 //Muestra en el LCD
301.    lcdGotoxy(3,2);                                         //Posiciona el cursor en la pantalla
302.    lcdWriteString(0xDF);                                   //Imprime unidad de medida
303.    lcdGotoxy(4,2);                                         //Posiciona el cursor en la pantalla
304.    lcdWriteString("C");                                    //Imprime unidad de medida
305.    lcdGotoxy(6,2);                                         //Posiciona el cursor en la pantalla
306.    itoa(datoExt[1],string);                                //Funcion que convierte entero en ascii
307.    lcdWriteString(string);                                 //Muestra en el LCD
308.    lcdGotoxy(8,2);                                         //Posiciona el cursor en la pantalla
309.    lcdWriteString("%RH");                                  //Imprime unidad de medida
310.    lcdGotoxy(12,2);                                        //Posiciona el cursor en la pantalla
311.    itoa(datoExt[2],string);                                //Funcion que convierte entero en ascii
312.    lcdWriteString(string);                                 //Muestra en el LCD
313.    lcdGotoxy(15,2);                                        //Posiciona el cursor en la pantalla
314.    lcdWriteString("L");                                    //Imprime unidad de medida
315.    lcdGotoxy(3,1);                                         //Posiciona el cursor en la pantalla
316.    itoa(datoInt[0],string);                                //Funcion que convierte entero en ascii
317.    lcdWriteString(string);                                 //Muestra en el LCD
318.    lcdGotoxy(5,1);                                         //Posiciona el cursor en la pantalla
319.    lcdWriteString(0xDF);                                   //Imprime unidad de medida
320.    lcdGotoxy(6,1);                                         //Posiciona el cursor en la pantalla
321.    lcdWriteString("C");                                    //Imprime unidad de medida
322.    lcdGotoxy(10,1);                                        //Posiciona el cursor en la pantalla
323.    itoa(datoInt[1],string);                                //Funcion que convierte entero en ascii
324.    lcdWriteString(string);                                 //Muestra en el LCD
325.    lcdGotoxy(12,1);                                        //Posiciona el cursor en la pantalla
326.    lcdWriteString("%RH");                                  //Imprime unidad de medida
327. }
328. //*****************************************************************************************************
329. // Programa principal, Realiza la lectura de DHT11, lectura de batería y recibe los datos por UART
330. // Lee la temperatura y humedad interna, realiza un pronóstico aproximado
331. //*****************************************************************************************************
332. void main(){                                               //Funcion principal
333.    saludoPantalla();                                       //Función de saludo de pantalla
334.    P1=0x00;                                                //Usado para aplicación
335.    P3=0x03;                                                //Usado para el serie
336.    serialInit();                                           //Inicializa puerto serie
337.    EA=1;                                                   //Habilitacion de interrupcion Global
338.    ES=1;                                                   //Habilitacion de interrupcion Serie
339.    while(1){                                               //Loop principal infinito
340.       leeDHT11();                                          //Función que lee DHT11
341.       delay_ms(2000);                                      //delay_ms de Actualización
342.       impPantalla();                                       //Imprime pantalla LCD
343.       delay_ms(2000);                                      //delay_ms de Actualización
344.    }
345. }
```

Firmware Exterior (Silabs C8051F832):

```
1.  //*****************************************************************************************************************
2.  // Programa para la estación meteorológica Interna inalámbrica
3.  // El programa recibe Temperatura, Humedad y Luminosidad por interrupción
4.  // El sistema mide Temperatura y Humedad interior
5.  // El enlace se realiza mediante una comunicación ASK OOK UHF sobre UART Invertido a 600bps
6.  // Se muestra medición interna y externa en un LCD de 2x16
7.  // Microcontrolador Silabs C8051F832
8.  //*****************************************************************************************************************
9.  #include <reg52.h>
10. #include <REG51F800.H>
11. #include <intrins.h>
12. #include <stdio.h>
13. //*****************************************************************************************************************
14. // Variables
15. //*****************************************************************************************************************
16. #define HEADER 200                                          //Definición de valor Header para el payload
17. sbit DATA = P1^0;                                           //Pin del bus de un hilo para el DHT11
18. static int datoInt[2];                                      //Variable donde se alojan los datos internos
19. static int payload[5];                                      //Variable donde se aloja el payload
20. static int datoExt[3];                                      //Variable donde se alojan los datos externos
21. sbit P0MDIN = 0xF1;                                        //Port 0 Input Mode
22. sbit P0SKIP = 0xD4;                                        //Port 0 Skip
23. sbit ADC0CF = 0xBC;                                        //ADC0 Configuration
24. sbit ADC0CN = 0xE8;                                        //ADC0 Control 0
25. sbit AD0BUSY = 0xF8;                                       //ADC0 Busy
26. sbit ADC0L = 0xBD;                                         //ADC0 Data Word Low Byte
27. sbit ADC0H = 0xBE;                                         //ADC0 Data Word High Byte
28. //*****************************************************************************************************************
29. // delay_us Bloqueante Micro Segundos
30. //*****************************************************************************************************************
31. void delay_us(unsigned int us_count){                      //Función para delay_ms de micro-segundos
32.    int t=0;
33.    while(us_count!=0){                                     //Mientras que el contador es distinto de cero
34.       for(t=0;t<16;t++){                                   //16MIPS dividido en 16 para 1us
35.          _nop_();                                          //Ejecuta función NOP de ensamblador
36.       }
37.       us_count--;                                          //Decremento del valor de delay_ms
38.    }
39. }
40. //*****************************************************************************************************************
41. // delay_ms Bloqueante Mili Segundos
42. //*****************************************************************************************************************
43. void delay_ms(unsigned int us_count){                      //Función para delay_ms de micro-segundos
44.    while(us_count!=0){                                     //Mientras que el contador es distinto de cero
45.       delay_us(1000);                                      //Ejecuta funcion delay_ms micro segundos
46.       us_count--;                                          //Decremento del valor de delay_ms
47.    }
48. }
49. //*****************************************************************************************************************
50. // Configuracion ADC pin P0.0
51. //*****************************************************************************************************************
52. void configuraADC(void){
53.    #define VREF 3                                          //Tensión de referencia interna
54.    P0MDIN=0xFE;                                            //Selecciona pin P0.0 como canal ADC
55.    P0SKIP=0x01;                                            //Decodificación Crossbar
56.    ADC0CF=0xF8;                                            //Habilita ADC 0
57.    ADC0CN=0x80;                                            //Habilita conversión en registro
58. }
59. //*****************************************************************************************************************
60. // Funcion para inicializar el puerto serie 9600 @ 11.059MHz
61. //*****************************************************************************************************************
62. void serialInit(void){                                     //Función para configurar UART
63.     TMOD=0x20;                                            //Timer 1 en modo 2 - Auto recarga para generar Baudrate
64.     SCON=0x50;                                            //Serial modo 1, 8N1
65.     TH1=0xFD;                                             //Carga el baudrate en el timer a 9600bps
66.     TR1=1;                                                //Dispara el timer
67.  }
68. //*****************************************************************************************************************
```

```
// Trama DHT11 - Según 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
      }
      while(DATA==1);                                       //Espera a DATA = 1
   }
   return valorLeido;                                       //Retorna el valor leído
}
//*************************************************************************************************************
// Función de recepción de dato para el DHT11
// Recibe los valores de temperatura y humedad (parte entera y decimales por separado)
// Recibe el checksum enviado por el DHT11 y lo compara con el leído en el programa
//*************************************************************************************************************
unsigned int recibeDato(){                                  //Funcion que recibe el Dato
   int validacion = 0;                                      //Variable de Validación
   int checksum = 0;                                        //Variable de detección de cambios de secuencia
   int j=0;                                                 //Variable para el lazo for
   DATA = 1;                                                //Set DATA = 1
   DATA = 0;                                                //Set DATA = 0
   delay_ms(18);                                            //Demora de 18ms (Del Datasheet)
   DATA = 1;                                                //Set DATA = 1
   delay_us(25);                                            //Demora de 25ms (Del Datasheet)
   validacion = DATA;                                       //Mueve valor de DATA a Validacion
   delay_us(80);                                            //Espera 80us (Del Datasheet)
   validacion = DATA;                                       //Mueve valor de DATA a Validacion
   if(!validacion){                                         //Si Validacion = 0, Error de secuencia
      printf( "Error en Checksum \r");                      //Muestra leyenda de error
   }
   delay_us(80);                                            //Espera 80us (Del Datasheet)
   for(j=0; j<5; j++){                                      //Lazo de carga de bytes de datos
       trama[j] = recibeByte();                             //Carga del vector de datos
   }
   DATA = 1;                                                //Set DATA = 1
   for(j=0; j<4; j++){                                      //Lazo de carga de bytes de verificación
       checksum += trama[j];                                //Carga de bytes de verificación
   }
   if(checksum == trama[4]){                                //Si la secuencia de verificación es correcta
      return 0;                                             //Se retorna 0 y se realiza la lectura
   }
}
//*************************************************************************************************************
// Lee DHT11
//*************************************************************************************************************
void leeDHT11(void){                                        //Funcion que lee el DHT11
   if(recibeDato()==0){                                     //Consulta si hay dato en la entrada del DHT11
      datoInt[0]=trama[2];                                  //Carga el valor de temperatura DHT en byte 1 del payload
      datoInt[1]=trama[0];                                  //Carga el valor de humedad DHT en byte 2 del payload
```

```
141.    }
142. }
143. //*************************************************************************************************
144. // Función que realiza la lectura del sensor LDR y acondiciona el valor
145. //*************************************************************************************************
146. int lecturaLDR(void){                                    //Función que realiza la lectura del ADC para el LDR
147.    unsigned int adval;                                   //Variable donde se guarda el valor del ADC
148.    AD0BUSY=1;                                            //Inicia conversión de ADC
149.    while(AD0BUSY);                                       //Espera que finalice la conversión
150.    adval=ADC0L;                                          //Lee el ADC de 0 a 511
151.    return(adval);                                        //Retorna el valor del ADC canal 0
152. }
153. //*************************************************************************************************
154. // Función para enviar byte por puerto serie
155. //*************************************************************************************************
156. void sendByte(unsigned char serialdata){
157.    SBUF=serialdata;                                      //Carga el dato a enviar por uart
158.    while(TI==0);                                         //Espera a la transmisión completa
159.    TI=0;                                                 //Borra el flag de transmisión
160. }
161. //*************************************************************************************************
162. // Función que realiza el parse de datos y armado del payload para enviar por UART
163. //*************************************************************************************************
164. void enivaRF(void){                                      //Declaración de función para enviar datos
165.    payload[0]=HEADER;                                    //Carga el Header en byte 0 del payload
166.    payload[4]=payload[1]+payload[2]+payload[3];           //Realiza suma de datos y carga en el byte 4 del payload
167.    sendByte(payload[0]);                                 //Envía el byte 0 del payload por UART
168.    delay_ms(50);                                         //Delay de espera entre bytes enviados por UART
169.    sendByte(payload[1]);                                 //Envía el byte 1 del payload por UART
170.    delay_ms(50);                                         //Delay de espera entre bytes enviados por UART
171.    sendByte(payload[2]);                                 //Envía el byte 2 del payload por UART
172.    delay_ms(50);                                         //Delay de espera entre bytes enviados por UART
173.    sendByte(payload[3]);                                 //Envía el byte 3 del payload por UART
174.    delay_ms(50);                                         //Delay de espera entre bytes enviados por UART
175.    sendByte(payload[4]);                                 //Envía el byte 4 del payload por UART
176.    delay_ms(50);                                         //Delay de espera entre bytes enviados por UART
177.    printf("\r");                                         //Envía retorno de línea como final de payload
178. }
179. //*************************************************************************************************
180. // Programa principal
181. // Lee la temperatura y humedad externa, lee el sensor de luminosidad, envía por RF
182. //*************************************************************************************************
183. void main(){                                            //Funcion principal
184.    serialInit();
185.    configuraADC();                                       //Función que configura ADC
186.    while(1){                                             //Loop principal infinito
187.       if(recibeDato()==0){                               //Consulta si hay dato en la entrada del DHT11
188.          payload[1]=trama[2];                            //Carga el valor de temperatura DHT en byte 1 del payload
189.          payload[2]=trama[0];                            //Carga el valor de humedad DHT en byte 2 del payload
190.          payload[3]=lecturaLDR();                        //Carga el valor de luminosidad en el byte 3 del payload
191.          enivaRF();                                      //Llamado a la función que envía datos
192.       }
193.       delay_ms(2304);                                    //Delay del timer de sleep*
194.    }
195. }
```

Pruebas y Simulación

Por motivos de costo y disposición en el país, se ha realizado de forma paralela la prueba del proyecto en un microcontrolador Microchip de bajo costo y conseguido en la mayoría de los locales de electrónica del país. Al tratarse de un código en lenguaje C, es muy sencilla la portabilidad hacia otro microcontrolador de diferente core y arquitectura.
Las pruebas de laboratorio han sido sobre placa de prototipo tipo Protoboard o Breadboard, donde se han incluido los componentes del proyecto para las pruebas de sensores, alcance de comunicación inalámbrica, consumo de la estación externa (esta utiliza pilas primarias alcalinas).

Resultados de prueba:

Se ha logrado una distancia entre dispositivos de aproximadamente 20m indoor (con paredes y ventanas con alambre tegido), que para los fines prácticos de este proyecto ha sido satisfactorio.

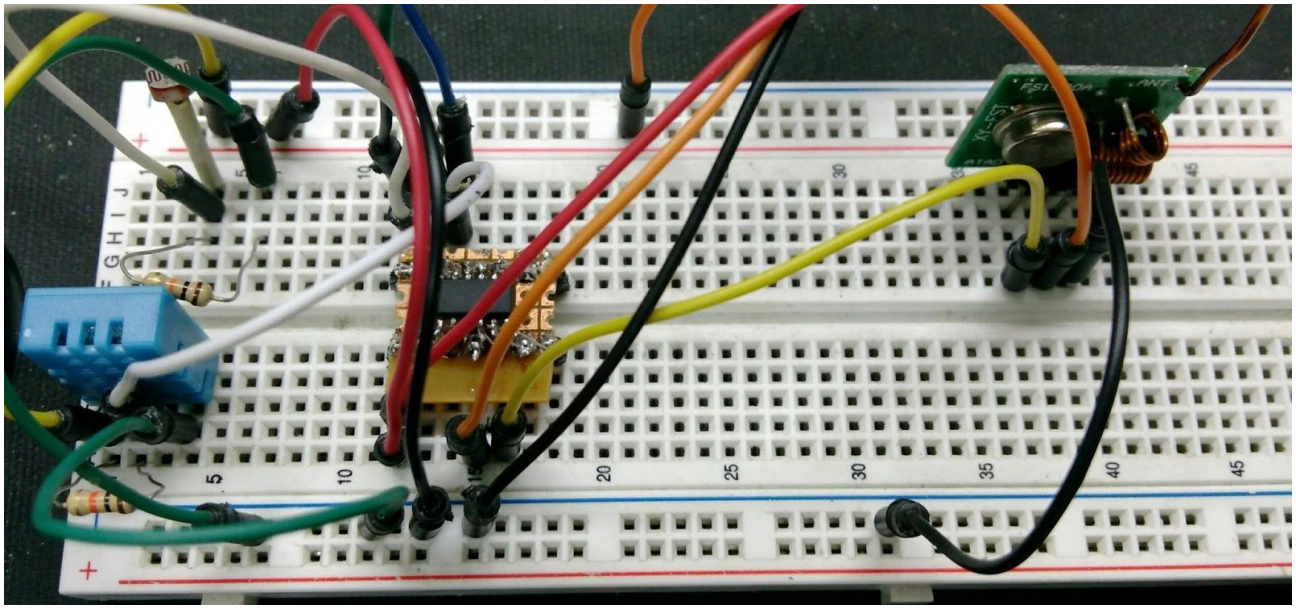
Los sensores se han contrastado contra una estación meteorológica comercial "Sinometer WS-1150", la temperatura posee un desvío de +2°C y la humedad de +8%.

El consumo en el dispositivo exterior es de 30nA en modo Sleep, 2uA en modo run y 14mA en modo TX.

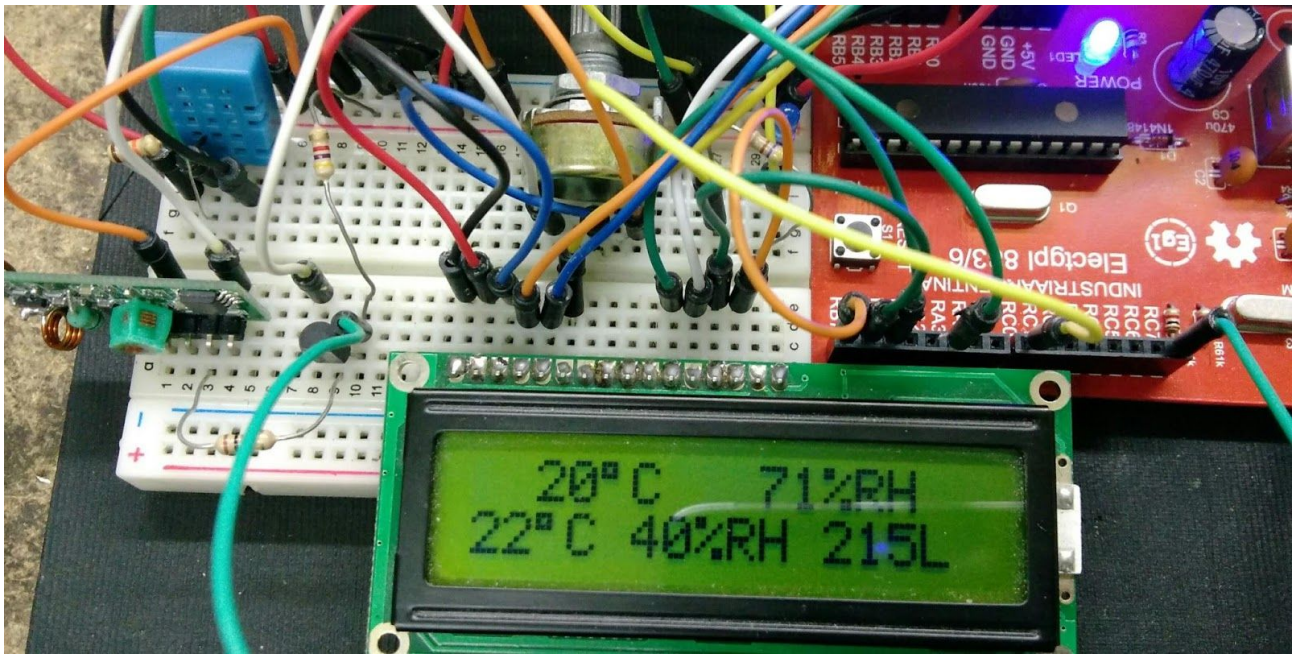
La autonomía del dispositivo interior es de 30 Horas sin recargar, la carga demora 2 Horas para completar la totalidad, mediante un puerto USB de 2A.

Fotos de Prueba:

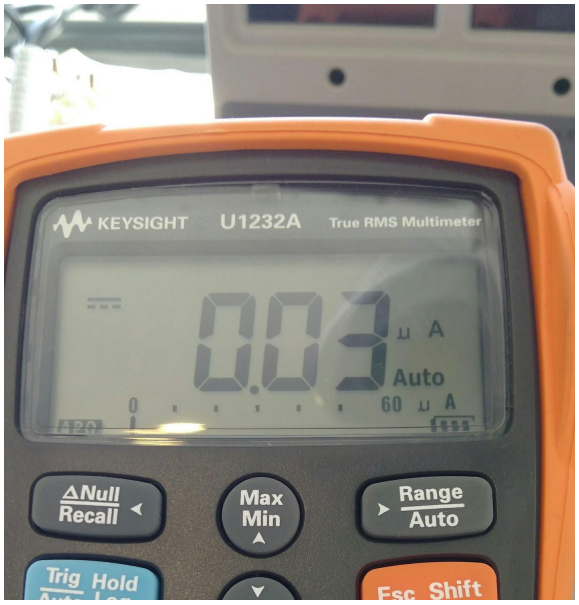
Estación Externa (DHT11, LDR, MCU, TWS433)



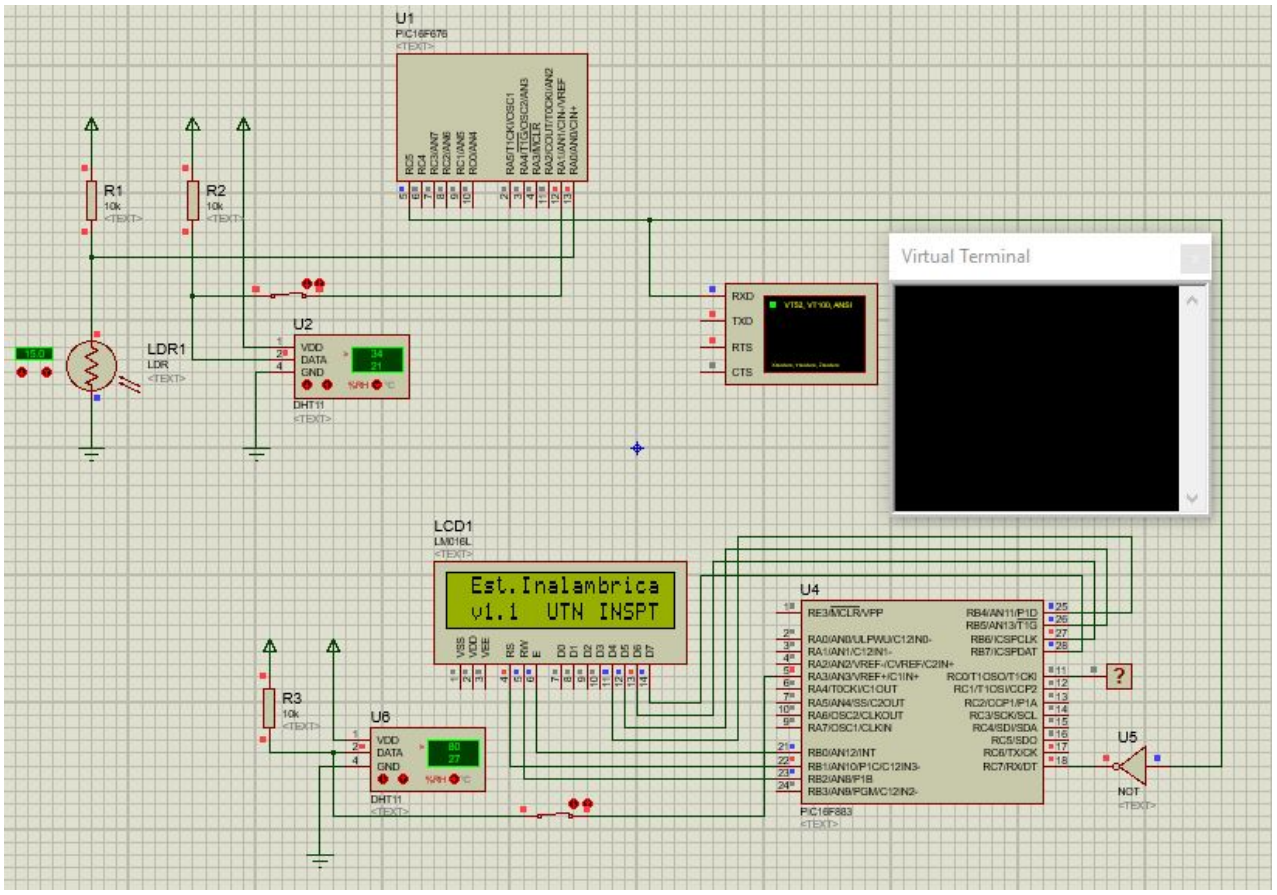
Estación Interna (RWS433, DHT11, LCD, MCU)

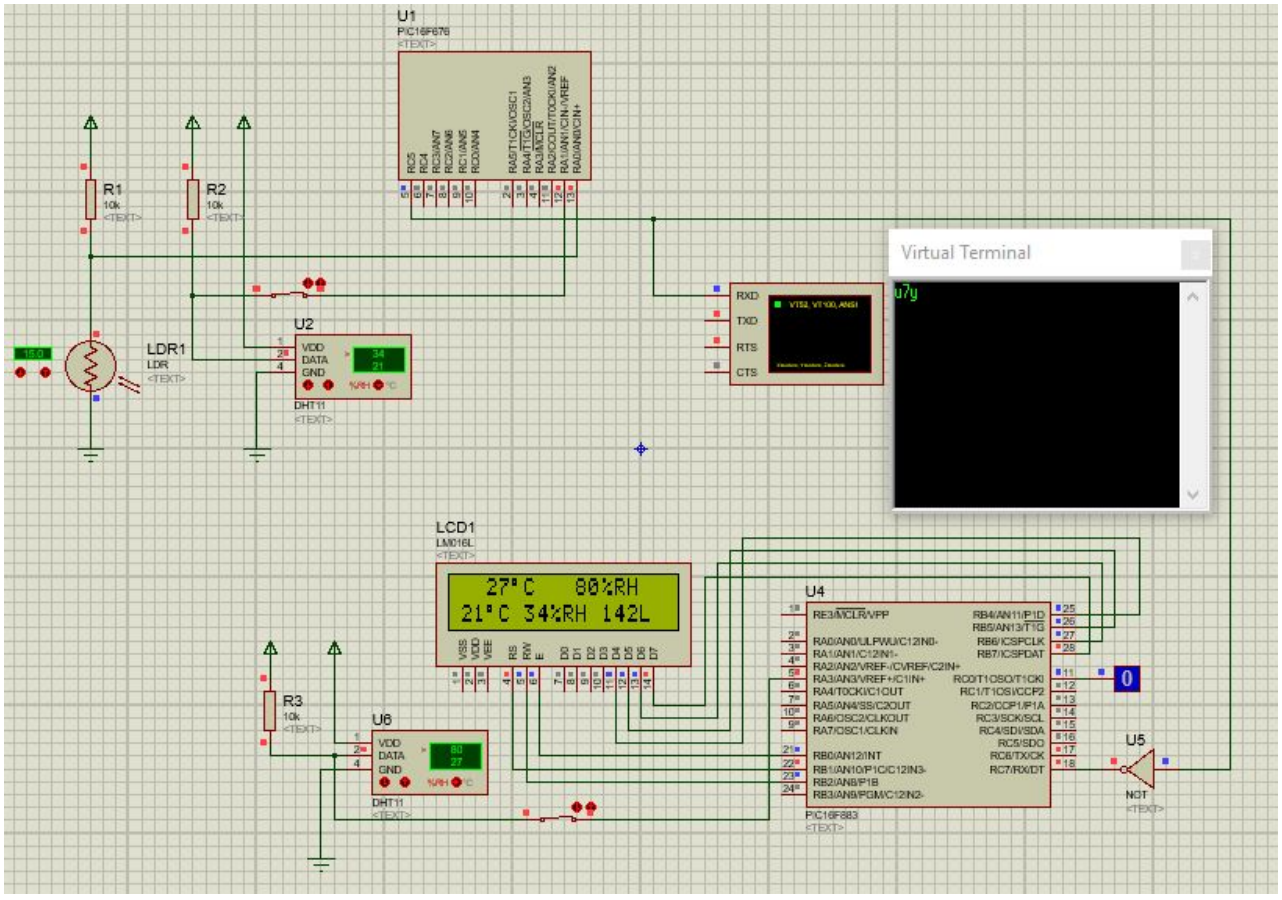


Medición Estación Externa en Sleep 30nA

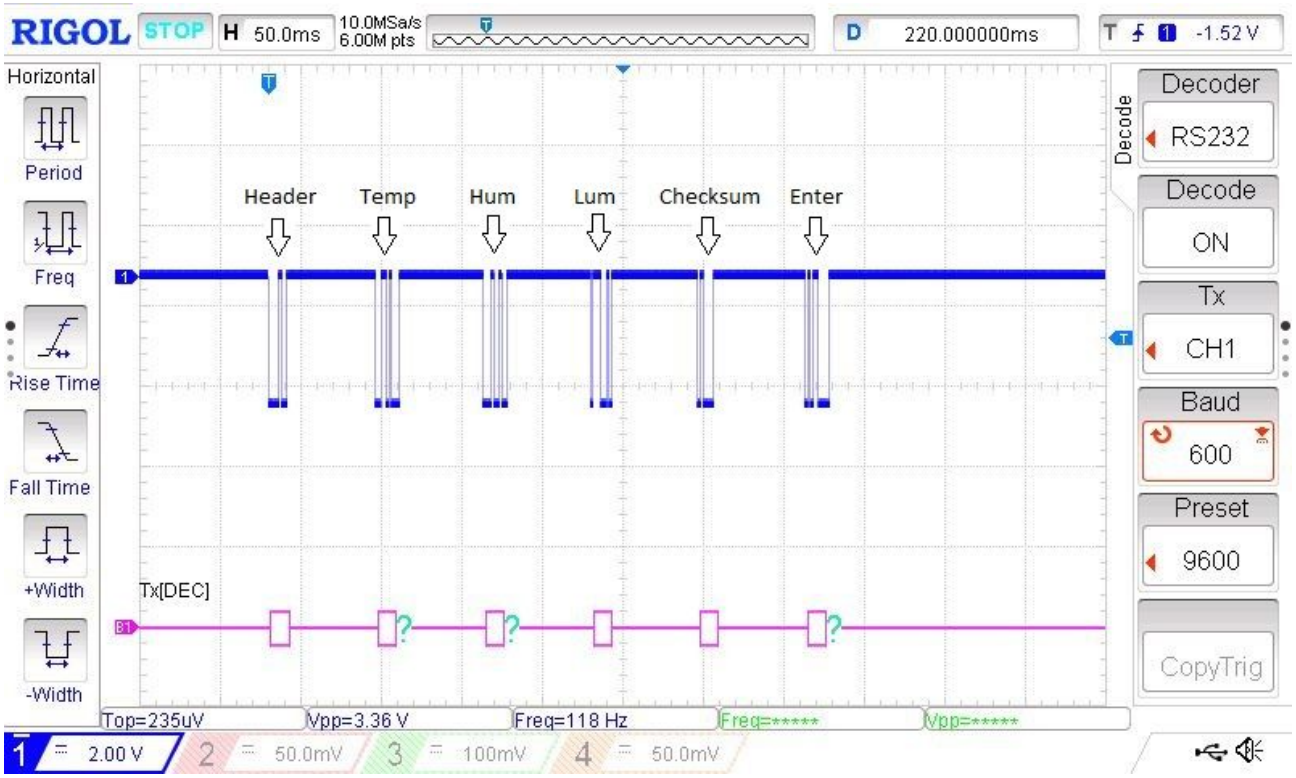


Capturas de Simulación en Proteus:

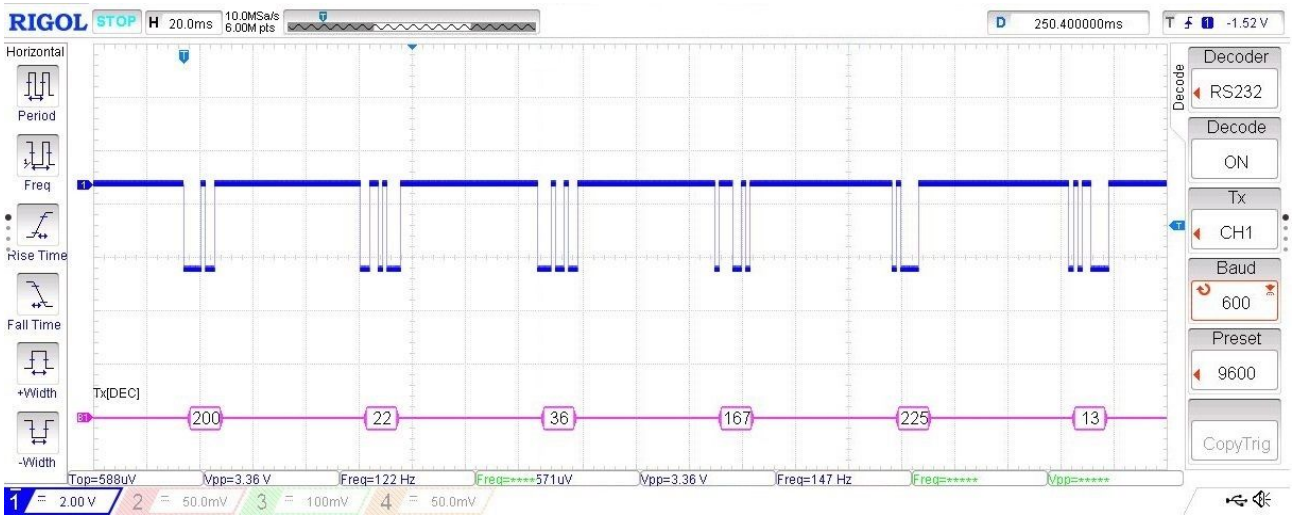




Capturas de Trama UART:



Se observa la trama completa (preámbulo, payload, verificación)



Se observa el Header: 200 (Establecido en el firmware), Temp: 22, Hum: 36, Lum; 167, Checksum (22+36+167): 225 y el retorno de carro (enter): 13.

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