Universidad de Alcalá

Estación meteorológica basada en LPC1768.

ESCUELA POLITECNICA SUPERIOR

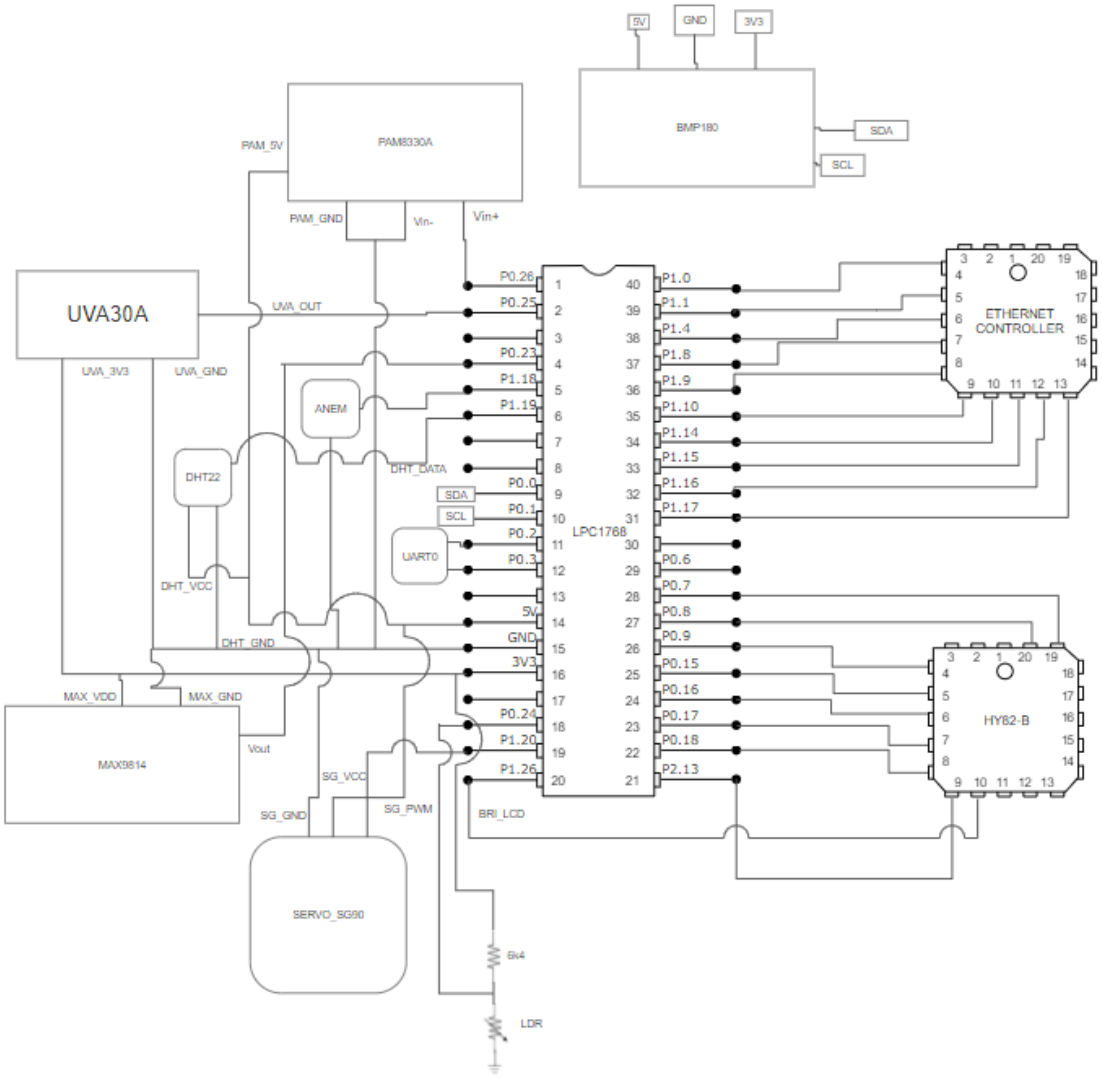
Autor: Palomo Alonso Alberto

Asignatura: Sistemas electrónicos digitales avanzados.

ESCUELA POLITÉCNICA SUPERIOR - UAH.

6 de noviembre de 2019.

Índice de contenido:


5V
GND
3V3
BMP180
SDA
SCL
PAM8330A
PAM_5V
PAM_GND
Vin-
Vin+
UVA30A
UVA_OUT
UVA_3V3
UVA_GND
ANEM
DHT22
DHT_DATA
DHT_VCC
DHT_GND
UART0
LPC1768
ETHERNET CONTROLLER
HY82-B
MAX_VDD
MAX_GND
MAX9814
Vout
SG_VCC
SG_GND
SG_PWM
SERVO_SG90
BRI_LCD
6k4
LDR

46

Abstract.

Microcontrollers, also known as its acronym MCUs (Micro Controller Unit), are embedded systems which, nowadays, its computational power can overcome microprocessors designed in early 90s as P54C (an old intel processor family) which CPU clock were around 100 MHz and were expensive. Only 20 years later, MCUs overcame, with a remarkable lower cost, its computational power.

In this document, I will talk about an LPC1768 based embedded system which holds a Cortex M3 processor. This MCU has a 100 MHz CPU clock speed and has a Thumb/Thumb-2 subset architecture developed by ARM. In this case, the project to document has been developed in a Mini-DK2 card manufactured by NXP, which contains the following features:

- 100M Ethernet network interface.
- TFT color LCD with SPI interface.
- Two USB interfaces.
- Some sets of buttons, one of which can produce interrupts.
- Serial ISP download JTAG (the code can also be flashed via USB).
- SD card with SPI interface.
- Generic input and output ports (GPIO).
- A 12 bit ADC and 10 bit DAC.
- One memory protection unit divided by regions.
- One real time clock.
- Interrupt controller.
- A quadrature encoder interface.
- System of pulse width modulator (PWM) and other for motor control.
- An I2C bus and 3 SPI buses.
- Four timer modules.
- Four universal asynchronous receiver transmitter buses.
  (Reference from: www.hotmcu.com/lpc1768minidk2-development-board-p-55.html)

- NXP sponsors the user a manual (UM10360) which can be obtained by typing the following link: https://www.nxp.com/docs/en/user-guide/UM10360.pdf
- The IDE is MDK (Keil uVision4) for the Cortex M3 processor, it can be bought in ARM website.

The project to build in this embedded system consists in a weather station with the capacity to measure some weather conditions such as temperature or air pressure. It will be able to communicate with a computer via UART and it will contain a website reachable via Ethernet interface.

Capítulo 1: Introducción y objetivos.

Introducción.

Una estación meteorológica basada en LPC1768 puede llegar a necesitar un sistema de control medianamente complejo. Pasar de instrucciones máquina que utiliza el procesador de ARM (Cortex M3) a controlar sensores de manera simple y eficaz necesita varias capas de interfaces, esto es, la acción que realizan los sistemas operativos en cualquier tipo de máquina que necesite interacción sencilla con el usuario. Es por ello que definiré más adelante una estructura del sistema operativo que actúa como interfaz, tanto para los sensores como para el usuario, cuyo código estará contenido en memoria flash.

Para empezar, hablaré un poco sobre las características que tiene la placa que vamos a utilizar.

La placa **Mini-DK2** contiene en su interior un procesador de ARM llamado Cortex M3 que funciona a 100MHz de velocidad de reloj. Además, tiene un oscilador de cristal para los periféricos denominado XTAL de 12MHz. A continuación, en la *'Tabla 1.- Características de la placa Mini-DK2'* resumo la gran mayoría de sus características.

Característica.	*Número de elementos.*	*Notas.*
100M interfaz Ethernet	1	
Interfaz TFT de colores (LCD) con interfaz SPI.	1	
Puertos de propósito general (GPIO).	4 x 32	
Interfaz USB.	2	
Bus I2C	1	
Bus SPI	3	
Interfaz UART.	4	
Reloj en tiempo real.	2	
Oscilador de cristal.	1	12MHz
Módulos temporizadores y de captura.	4	
Módulo PWM.	1	
Módulo PWM especializado para motores.	1	
Acceso directo a memoria (DMA)	1	
Unidad de protección de memoria.	1	8 regiones
Conversor de analógico a digital.	1	12 bits
Conversor de digital a analógico.	1	10 bits
Memoria RAM.	1	64kB
Memoria flash.	1	512kB
Interfaz de encoder de cuadratura	1	
Slot de SD con interfaz SPI	1	
Interrupciones externas.	4	
Leds	N/A	
Interfaz de carga de código vía JTAG por ISP.	1	
Controlador de interrupciones anidadas (NVIC).	1	
Entrada de interrupción no enmascarable (NMI).	1	

Tabla 1.- Características de la placa Mini-DK2.

Para abordar la práctica hablaré de la estructura definida de ese 'sistema operativo' en el **Capítulo 3: Estructura del sistema** y más adelante explicaré cada parte específicamente. Cabe destacar que el código empleado utiliza una estructura jerárquica para controlar el sistema y utiliza variables como señales de comunicación entre diferentes zonas de código.

Más adelante explicaré qué sensores tiene el proyecto, qué miden cada uno de ellos y cómo funcionan a rasgos generales. Todo ello explicado en el **Capítulo 2: Componentes del sistema**.

Objetivos a cumplir.

Los objetivos establecidos para el cumplimiento de la práctica se pueden resumir en la siguiente tabla (Tabla 2.- Objetivos a cumplir):

Objetivo general.	**Objetivos específicos.**
Crear un menú táctil con el TFT de la tarjeta.	Crear un statechart.
	Manejar las librerías para la generación de caracteres y zonas en pantalla.
	Vincular todos los datos y señales de forma coherente.
	Permitir al usuario modificar variables del sistema.
	Permitir al usuario interactuar con el sistema.
	Calibrar la pantalla de manera estática o dinámica.
Controlar un servo motor para la visualización de temperatura.	Manejar el brillo de manera manual o automática con el módulo PWM.
	Ser capaz de alternar dos modos PWM para controlar el servo y el brillo a la vez.
	Ser capaz de dar valores coherentes a las variables PWM.
	Interconectar el módulo PWM con los datos.
Crear un servidor web accesible vía Ethernet	Manejar las librerías TCP-IP.
	Configurar variables para realizar una conexión TCP-IP.
	Escribir con lenguaje de marcado una página web.
	Crear funciones que interactúen con la página web mediante callbacks.
	Mantener una conexión TCP.
	Proporcionar datos a la página web.
	Recibir datos de solicitudes del usuario mediante la web.
Medir la velocidad del viento.	Realizar un análisis físico de cómo realizar las medidas.
	Realizar un análisis a nivel de electrónica de las medidas.
	Configurar un módulo temporizador para medir frecuencia de pulsos.
	Traducir los pulsos a medidas.
	Conexionar los datos recibidos sin que haya interferencias con todo el sistema anterior y posterior, incluyendo gestión de recursos de la tarjeta.

*Tener un reloj en tiempo real.**	Examinar si las medidas son precisas.
	Configurar el RTC de la tarjeta.
	Exportar los datos de manera coherente.
	Ser capaz de configurar la fecha, día y hora de manera manual.
	Mostrar por el TFT la hora actual.
	Conexionar los datos recibidos sin que haya interferencias con todo el sistema anterior y posterior, incluyendo gestión de recursos de la tarjeta.
Tener un sistema de control de bloqueo del sistema.	Configurar un Watchdog timer.
	Seleccionar una zona del código donde se ejecute con frecuencia, para cualquier estado de funcionamiento, la alimentación del Watchdog sin interferir con el sistema anterior y posterior.
	Elegir un valor adecuado para las variables del Watchdog timer.
	Examinar que este temporizador no salte de manera espuria.
Medir temperatura.	Comprender y manejar el protocolo OneWire.
	Comprender documentación en mandarín o en su defecto comprender documentación mal planificada en inglés*.
	Configurar un pin para un timer en modo captura.
	Gestionar los recursos para no quedarse sin timers ni interferir con el sistema anterior y posterior.
	Estructurar y traducir los datos recibidos adecuadamente.
Reproducir audio.	Configurar adecuadamente el DAC.
	Exportar vía DMA los datos solicitados.
	Gestionar la memoria de manera eficaz para tener 3 segundos de audio almacenados en memoria.
	Gestionar los recursos para no quedarse sin timers ni interferir con el sistema anterior y posterior.
	Configurar adecuadamente otro timer para reproducir el audio guardado en memoria.
	Hacer sonar el audio cada vez que una medida supera un umbral.
	Establecer dicho umbral para todas las medidas.
	Implementar un botón de reproducción del audio cada vez que el usuario lo desee.
	Seleccionar las variables adecuadas, así como la frecuencia de exportación del audio.
Grabar audio.	Configurar un ADC para tener dos modos de funcionamiento.
	Configurar un pin de la placa para leer audio.
	Seleccionar adecuadamente las variables, así como la frecuencia de muestreo del audio.
	Ser capaz de alternar ambos modos de funcionamiento del ADC sin interferir con

	las demás medidas y ser capaz de bloquearlas mientras se produce la grabación del audio.
	Exportar los datos de manera adecuada al sistema.
	Entrar en el modo de grabación cuando el usuario lo desee.
Medir el brillo.	Traducir de manera adecuada resistencia a voltaje y de voltaje a código del ADC.
	Escoger una resistencia de pull-up adecuada para el LDR seleccionado.
	Configurar el ADC en modo ráfaga sin que interfiera con la grabación de audio.
	Leer el código del ADC para traducirlo a nivel de brillo.
	Controlar automáticamente el brillo del LCD en función del dato.
	Examinar la validez de las medidas.
	Gestionar la frecuencia de muestreo.
Medir el índice UV.	Traducir de manera adecuada el código recibido al índice.
	Usar el modo del ADC ya configurado para esta medida sin interferir con el sistema anterior y posterior.
	Examinar la validez de las medidas.
	Exportar los datos al sistema.
Medir presión atmosférica.	Crear un controlador que sea capaz de leer registros de un sensor vía I2C.
	Examinar la validez de las medidas.
*Medir posición GPS[1].**	Conseguir leer un sensor GPS vía UART.
*Conexión WiFi.**	Conseguir conexionar un módulo WIFI vía UART a la placa.
	Ser capaz de modificar variables.
	Comprender los protocolos utilizados para conexionar un dispositivo WIFI con el módulo.
	Buscar una interfaz adecuada.
Conexión UART con un PC.	Conexionar el módulo UART del lpc1768 con el controlador de un ordenador convencional.
	Modificar variables mediante la conexión serie asíncrona.
	Conseguir exportar datos vía conexión seria asíncrona.

Tabla 2.- Objetivos a cumplir.

Como podemos observar, esta lista de objetivos a cumplir de manera obligatoria para el funcionamiento del sistema, contiene todo lo que debemos de realizar a grandes rasgos y será utilizada como guía para los siguientes capítulos.

[1] Los objetivos marcados con * representan objetivos opcionales pero que incluyen en su implementación modelos marcados como obligatorios.

Posibles mejoras al sistema en un futuro.

Pese a que los objetivos generales son muy completos, los objetivos opcionales son sólo suplementarios y añaden funciones extras a esta estación meteorológica para hacer pequeñas mejoras del sistema y que sea aún más completo. No tabularé los objetivos opcionales, que no prometo para nada cumplir, pero que puede que unas versiones futuras del proyecto aparezcan, son los que enumero a continuación y evalúo su complejidad:

- Añadir memoria extra con la interfaz SPI en la tarjeta SD.
  (Complejidad media)
- Añadir funcionalidad a la página web para modificar más variables del sistema.
  (Complejidad baja)
- Hacer un seguimiento histórico de los datos.
  (Complejidad baja)
- Exportar los datos a la comunidad.
  (Complejidad media)
- Mejorar la interfaz TFT.
  (Complejidad baja)
- Comunicarse con más estaciones del mismo tipo a gran distancia.
  (Complejidad alta)
- Realizar los objetivos marcados como * en el apartado anterior.
  (Complejidad media)

Capítulo 2: Componentes del sistema.

Mini-DK2

El componente fundamental de sistema es la tarjeta antes mencionada en el *Capítulo 1: Introducción y objetivos a cumplir.* Será la unidad de control y todos los sensores y aparatos electrónicos del sistema se conectan a este componente. Su función es ejecutar el sistema operativo antes mencionado en el Capítulo 1 y proporcionar control sobre todo el sistema. En la *Figura 1* podemos ver el esquema de pines de la tarjeta con la que vamos a trabajar. En la *Figura 2* podemos ver una imagen de la tarjeta.

Si desea ver las características generales de este componente puede recurrir a la *Tabla 1* o si desea más información, puede recurrir a su fabricante; NXP.

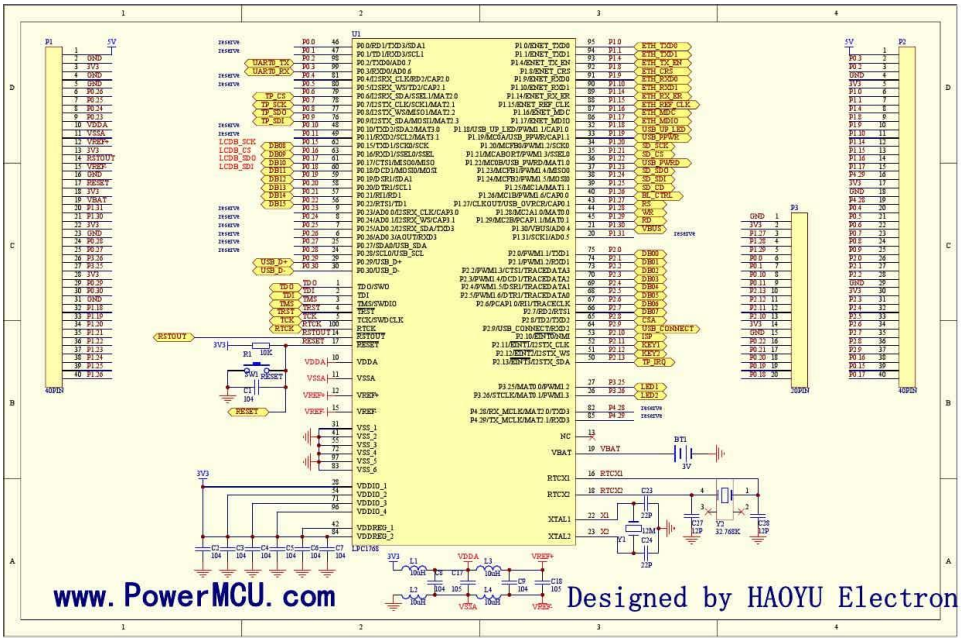


Figura 1.- Esquema general de la tarjeta Mini-DK2.

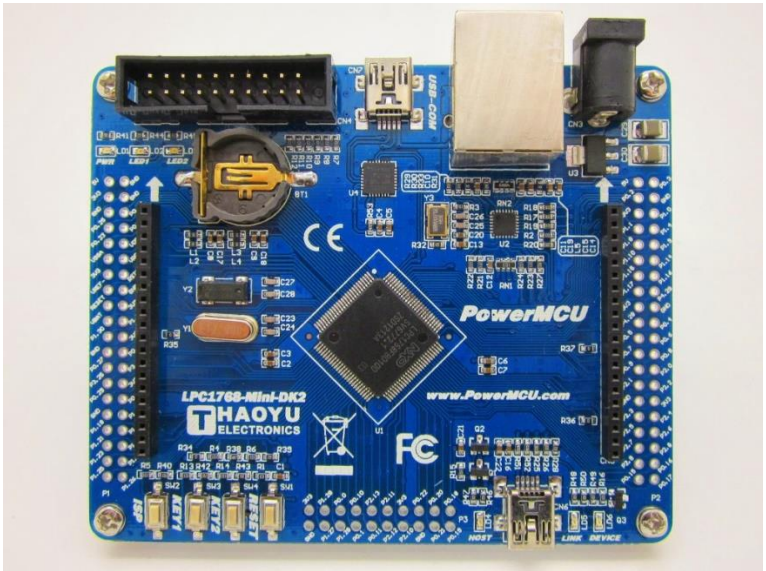


Figura 2.- Imagen de la tarjeta Mini-DK2.

Anemómetro.

Este componente nos permitirá medir la velocidad del viento. Se trata de un eje rotatorio conectado a tres cuencas de baja densidad que forman un ángulo de 120° entre ellas y permiten rotar sobre el eje si existe viento.

Tiene dos pines de salida que son cortocircuitados por zonas cada 90°, por lo que cada vuelta se cortocircuita dos veces.

Su funcionamiento consiste en generar cortocircuitos entre los dos pines de salida a medida que el eje va rotando, por lo que podemos generar pulsos cuadrados a medida que el eje rota si lo conectamos adecuadamente.

En la siguiente figura, *Figura 3* podemos ver una imagen de este anemómetro del cuál no he encontrado documentación.

Figura 3.- Anemómetro.

LDR.

Este componente nos permitirá medir la cantidad de brillo que hay en la zona de manera que este varía la resistencia entre sus dos patillas en función de la luz que recibe. Esto sigue una relación lineal con una tolerancia entre los LUX en el ambiente y la resistencia entre ambas patillas. Conectándolo adecuadamente podemos traducir mediante un divisor resistivo y **escogiendo un valor de pull-up adecuado** esta resistencia nos dará los LUX que mida.

En la *Figura 4* incluyo una imagen del LDR y en la *Figura 5* una gráfica en la mejor calidad que he encontrado que representa la relación resistencia-LUX del componente.

Figura 4.- LDR.

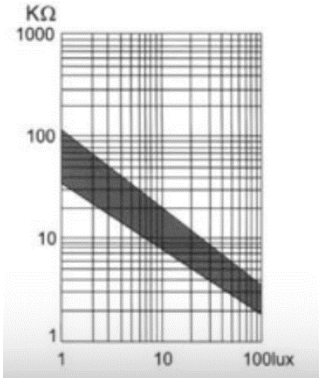


Figura 5.- Gráfica R-LUX

Sensor UV.

Este sensor nos permitirá medir el índice UV que reciba, este sensor traduce el índice UV que recibe a voltaje directamente sin necesidad de aplicar un circuito conversor resistivo como en los dos casos anteriores, en cambio, viene en forma de tarjeta pequeña con 3 pines de interés:

- Alimentación (+3.3V)
- Masa. (-0.0V)
- Salida de voltaje. [0,3.3V]

Este sensor mide la índice UV y saca un índice del 0 al 10 en forma de la siguiente ecuación:

$$Vo = \frac{Vcc \cdot IndiceUV}{IndiceUVmax}$$

En la *Figura 6* se muestra el sensor UV a utilizar (VMA30A)

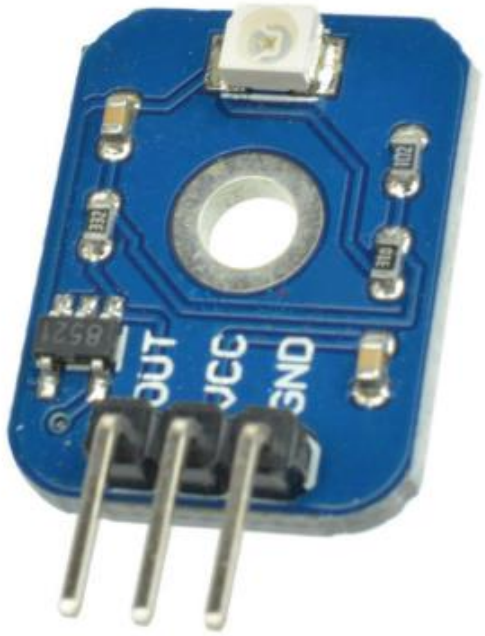


Figura 6.- Sensor UV30A.

Servo motor.

Este componente nos permitirá representar la temperatura que hay entre dos valores límite a elegir. El mínimo significará izquierda, mientras que el máximo de esa temperatura será a la derecha. Este servo controla el ángulo de posición en función de una señal modulada por pulso (PWM) y, en función de su ciclo de trabajo o tiempo a nivel alto, tendrá un ángulo u otro. En la *Figura 7* aparece una imagen del servo motor a utilizar (SG90) y en la *Figura 8* una gráfica del fabricante que sólo va a servir para identificar un problema que surge a raíz de utilizar la tarjeta Mni-DK2 con este servo.

Figura 7.- SG90, servo.

Como podemos observar en la *Figura 8* **la señal PWM generada ha de tener un valor a nivel alto de +5.0V**. Es por ello que esta gráfica sólo nos va a dar una referencia del periodo de la señal a generar, pero no del tiempo en alta, dado que estos servos se guían por la potencia recibida que depende de la amplitud del pulso y el voltaje. Habrá que hacer una corrección en potencia del pulso.

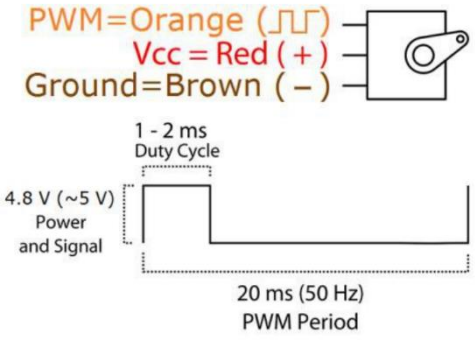


Figura 8.- Pulso a recibir.

Amplificador + altavoz.

Se utilizará un altavoz de 8Ω conectado a un amplificador de señal adaptado de 4 a 8Ω (PAM8320A) que contiene el siguiente patillaje:

- Vdc – +5.0V.
- GND - -0.0V.
- Shutdown – N/C.
- Audioin+ - Audio.
- Audioin- - GND.

Esta configuración permite pasar de una señal eléctrica de audio a audio en sí. En la Figura 9 muestro una imagen del amplificador soldado al altavoz de 8Ω.

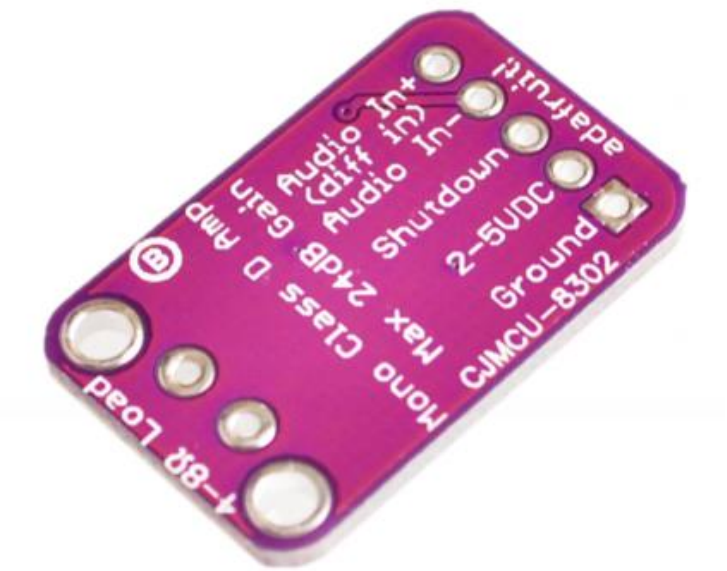
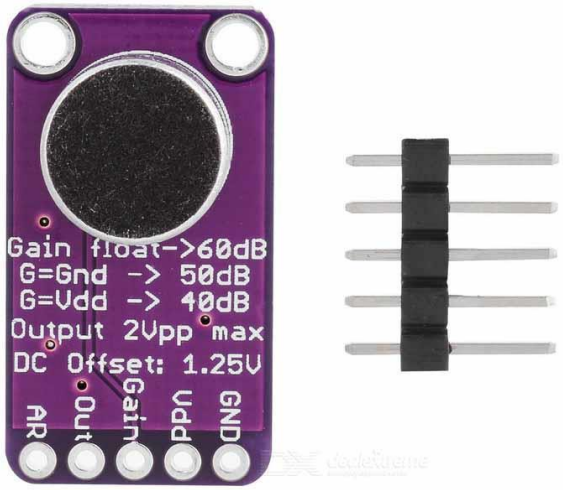


Figura 9.- PAM8320A + Altavoz



Micrófono.

Este micrófono (MAX9814) es un micrófono de alta calidad y bajo coste que nos permitirá hacer vibrar la señal continua con un offset de +1.25V (amplitud máxima de +2Vpp) en función del audio recibido. Directamente podemos convertir la señal por el pin de salida con la siguiente configuración del patillaje:

- AR – N/C
- Gain – N/C
- Out – Salida.
- Vdd - +3.3V
- GND - -0.0V

Figura 10.- MAX9814, micrófono.

Se podría configurar para que obtuviese ganancia personalizada, pero no conectar da una ganancia fija también que no tendremos en cuenta dado que nuestro objetivo no es mejorar la calidad de audio, sino simplemente tener una señal de audio. Lo mismo pasa con el pin AR, que fija el factor de compresión del audio; no conectarlo implica que tenga un valor fijo, dado que ambas entradas al aire no forman efecto de alta impedancia.

En la figura 10 tenemos una imagen de dicho micrófono (MAX9814) con parte de los valores más significativos de la datasheet impresa en la tarjeta.

Sensor de temperatura.

El sensor que medirá la humedad y la temperatura en interiores a utilizar será el DHT22. Este sensor pese a tener 4 patillas, una no hay que conectarla y la otra utiliza el protocolo OneWire. Las otras dos son de diferencia de potencial para alimentación. El conexionado es el siguiente, siendo ordenados los pines de izquierda a derecha:

- Pin 1 - +5V Vcc.
- Pin 2 - OneWire.
- Pin 3 - NC
- Pin 4 - 0V GND.

Cabe destacar que este sensor requiere de una resistencia de pull-up de 4k1Ω.

Figura 11.- Sensor DHT22.

Figura 12.- Módulo BMP180.

Sensor de presión.

El sensor a utilizar es un módulo que incorpora el sensor BMP180, que funciona por protocolo I2C.

Cabe destacar que el módulo contiene un regulador y las resistencias necesarias de pull-up para el funcionamiento de los pines SDA y SCL del protocolo. El conexionado del módulo es el siguiente:

- GND - -0.0V
- Vcc - +5.0V
- SDA - Datos I2C.
- SCL - Reloj I2C.

Este sensor utiliza un driver software que puede ser extraído del fabricante, pese a eso, se ha desarrollado un driver específico para estación meteorológica utilizado los mismos algoritmos que utiliza el driver del fabricante para leer los registros por I2C del sensor.

TFT.

Se utilizará un display táctil (TFT), para poder mostrar y recibir por pantalla datos relativos a la interfaz del usuario y variables ajustables del sistema. El panel se llama HY32-B y utiliza 8 bits para comunicarse. El conexionado de los 10 bits, los 8 anteriormente mencionados y dos de control se muestra a continuación:

Figura 13.- Panel TFT-LCD.

Puerto / pin utilizado.	Función que desempeña.
P0.6	Touchpannel – CS
P0.7	Touchpannel - SCK
P0.8	Touchpannel - SDO
P0.9	Touchpannel - SDI
P0.15	LCD-SCK
P0.16	LCD-CS
P0.17	LCD-SDO
P0.18	LCD-SDI
P1.26	Control del brillo por PWM.
P2.13	Interrupción del touchpannel.

Tabla 3.- Conexionado del TFT.

Capítulo 3: Estructura del sistema.

Durante este capítulo nos dedicaremos a analizar cómo he estructurado de manera general el sistema y cómo pretendo hacer funcionar y gestionar los recursos de la placa para que no interfieran entre ellos.

Cabe mencionar que este es un análisis que hay que hacer desde un principio y en mi caso no lo he podido hacer debido a que he tenido que añadir las cosas por partes separadas sin conocer el funcionamiento de las posteriores. Por ello, he tenido que aplicar la estrategia de: *la utilización de los menores recursos posibles para que funcione y ya añadiré más recursos si me sobran y si quiero mejorar la calidad*, cosa que si pudiese haber evitado lo hubiese hecho.

0. Esquema general.

Mi esquema se basa en ramificar jerárquicamente en torno a la interfaz de usuario, es decir, el menú, los demás módulos de control del sistema según la *Figura 1.*

Fi

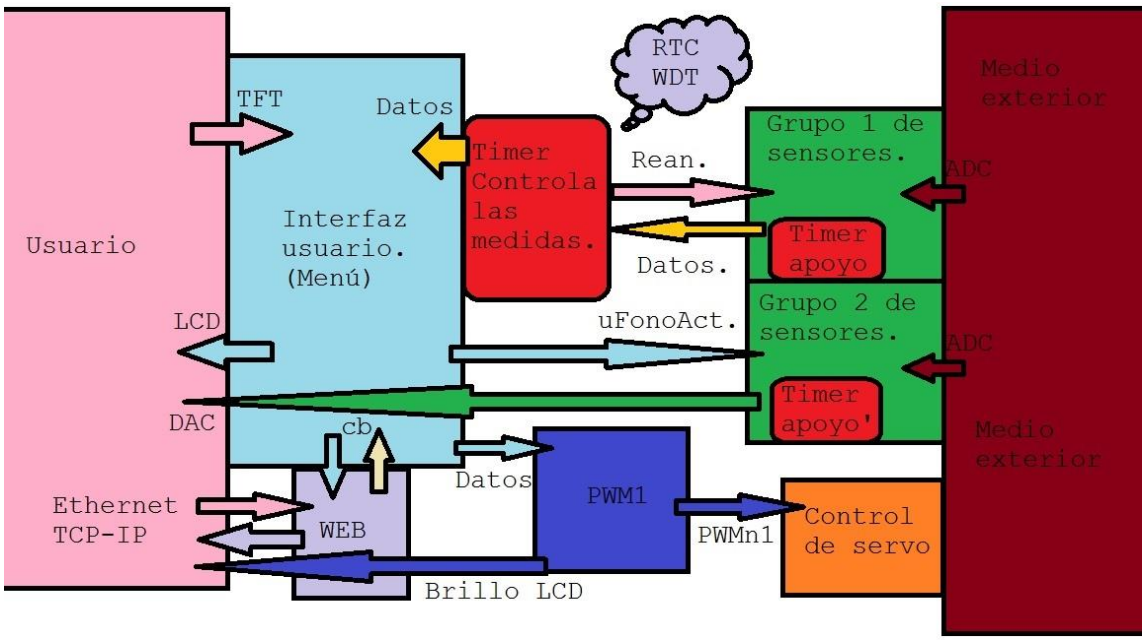


Figura 1.- Esquema de primera versión.

Cabe destacar que el esquema de la *Figura 1* es la primera versión. Es decir, sin que exista conexión UART posible ni WiFi. Añadiré una versión así más adelante que únicamente añade esos dos módulos.

El esquema en general consta de módulos que se comunican todos con la interfaz de usuario indirecta o directamente. Por eso que es un requema ramificado a partir de la interfaz de usuario (que actúa como sistema de control). También llama la atención un timer que controla las medidas. Esto lo explicaré en la sección de este mismo capítulo *Timer controlador de medidas*.

A grandes rasgos, existe un timer que controla las medidas del grupo 1, es por eso que he separado en dos grupos los sensores: los controlados por timer y los controlados por la interfaz de usuario; grupo 2. Evidentemente, los del grupo 2 son los correspondientes a grabar y

reproducir audio y los demás corresponden al grupo 1. El tiempo de muestreo es, por tanto, marcado dentro del timer que controla esos sensores para el grupo 1 y constante para el grupo 2 de sensores. Todos ellos leen del medio exterior los datos con el único ADC de la placa que consta de varios canales multiplexados.

El usuario interactúa con el interfaz de usuario mediante el TFT y el LCD, además es posible comunicarse mediante UART y mediante Ethernet (TCP-IP).

1. PWM1 y control de servo.

El módulo PWM se utiliza para controlar un servomotor que se encargará de monitorizar las medidas de temperatura y presión del sistema. Los datos son almacenados en memoria y el sistema se encarga de, cada vez que se leen los sensores, actualizar el módulo PWM para enviar un pulso de manera adecuada para que el servo se posicione en función de unos valores máximos y mínimos que pueden ser modificados por el usuario en todas sus interfaces.

La estación meteorológica utiliza el servo expuesto en el capítulo 2 y lo actualiza el manejador de interrupción del Timer0 del LPC1768.

2. WEB.

La interfaz web utiliza una pila de protocolos y librerías TCP-IP para proporcionar los recursos que permiten montar un servidor WEB integrado en el LPC1768. Mediante un compilador de CGI compila un archivo con extensión de formato CGI que incorpora un lenguaje de marcado HTML, para configurar la interfaz de usuario WEB. Más adelante se explica el funcionamiento.

3. Usuario.

El usuario puede interactuar con las diferentes interfaces que se presentan para leer los datos de la estación meteorológica. Además, este usuario puede modificar mediante todas sus interfaces los valores modificables del sistema.

- Interfaz WEB: Interfaz mediante un servidor WEB sobre un cable ethernet, se puede acceder mediante un navegador y el usuario puede modificar las variables del sistema excepto la hora actual.
- Interfaz UART: Interfaz mediante el protocolo de comunicación UART sobre un cable USB, se puede acceder mediante una aplicación que monitorice la conexión serie asíncrona como Termite (por ejemplo).
- Interfaz TFT-LCD: Interfaz mediante el protocolo de comunicación SPI sobre los pines de la placa MiniDK-2. El usuario lee por la pantalla del LCD la interfaz creada en el programa mediante la iluminación del array de pixeles. Además, el usuario puede modificar las variables del sistema desde la pantalla.

Las variables que el usuario puede modificar son: máximos y mínimos valores de alarma y presión. Si se superan, suena una alarma. La hora puede ser modificada, también el umbral de brillo, que es un valor expresando en segundos de el tiempo que permanece encendida la pantalla en el modo ULPM[2]. Otra variable a modificar es la selección del servo, es decir, qué variable (si presión o temperatura) es representada.

[2] Ultra Low Power Mode. Ver manual de usuario.

4. Medio exterior.

El medio exterior se utiliza para obtener los dartos, los sensores utilizan el medio exterior para obtener valores que la estación meteorológica lee mediante los diferentes protocolos.

5. Timer controlador de medidas.

Se utilizan varios módulos temporizadores para apoyar a los sensores. A continuación, se exponen las funcionalidades de cada uno de los times si se utilizasen:

- Timer 0: Controla las medidas, marca el tiempo de muestreo de cada uno de los sensores y se ocupa de actualizar el módulo PWM cada vez que interrumpe, en caso de haber cambiado un valor máximo o mínimo.
- Timer 1: Este Timer cumple **tres funciones**: la primera es utilizar el modo capture para medir los pulsos del **anemómetro**; la segunda es utilizar un MR (Match Register) para desactivar el **DAC** una vez haya acabado de reproducir audio; y la tercera es utilizada en caso de **grabar audio**, tener un MR que sirva para activar las cuentas.
- Timer 2: No usado.
- Timer 3: Sirve de apoyo al protocolo OneWire (actúa de contador).

6. Grupo de sensores 1.

Definimos como el primer grupo de sensores al grupo perteneciente al sensor BMP180 y DHT22, debido a que al tener incorporadas esperas activas es recomendable medirlo cuento menos. Cabe recalcar que estos sensores miden variables que no cambian con gran velocidad, por lo que pueden ser medidas cada 5 segundos incluso más si se desease ajustar. Estas variables con la temperatura, la presión y la humedad. En caso de querer medirlas más rápido, cambiar el símbolo CsCAP definido en el archivo Systemsymbols.h y poner un valor de cuentas adecuado para la función que se desee desempeñar.

7. Grupo de sensores 2.

Definimos como segundo grupo de sensores al grupo perteneciente a los sensores LDR y UVA. Debido a que necesitan un tiempo de muestreo más regular, la luz y el índice UV pueden cambiar rápidamente, por lo que es aconsejable medirlo con más frecuencia. Si se desease modificar el valor del tiempo de muestreo, referirse a CsADC definido en el archivo Systemsymbols.h y poner un valor de cuentas adecuado para la función que se desee desempeñar.

8. WDT.

El WatchDogTimer es un contador regresivo que si se deja llegar a 0 provoca un reinicio del sistema. Así, si el sistema es bloqueado, el WDT lo desbloquea reiniciando el sistema. Cabe destacar que la hora debe de ser ajustada de nuevo y todas las variables que el usuario haya modificado.

9. RTC.

El RTC es un reloj en tiempo real que sirve para tener un reloj en la estación meteorológica. Es un contador que se utiliza para mostrar por la página WEB y por pantalla la hora actual de las medidas.

10. UART.

Es un módulo que se encarga de transmitir y recibir información por el protocolo de comunicación seria asíncrona universal. Sirve como interfaz para que el usuario pueda recibir información de la estación y mandar comandos a la misma.

11. ADC.

El módulo ADC, es un módulo integrado en la placa que tiene la capacidad de codificar señales analógicas y convertirlas a un formato digital, esto permite leer señales analógicas, en nuestro caso, el sensor LDR y el UVA.

12. DMA + DAC.

Estos módulos actúan unidos. El DMA se encarga de liberar carga computacional al procesador, transfiriendo el valor de los datos del audio al siguiente módulo, el DAC, mediante una señal de inicio de conversión. Mientras que el DAC es un módulo que genera una señal analógica en función de un valor digital recibido. El funcionamiento reside en activar el DMA cuando se desee producir audio y desactivarlo cuando este termine, produciendo las transferencias de datos con el DMA y no con el procesador.

Se recuerda que el Timer encargado de desactivar la transferencia es el Timer 1.

Capítulo 4: Implementación del sistema.

En este capítulo veremos cómo se implementan tanto en software como en hardware los sistemas y módulos anteriormente mencionados en el *Capítulo 3: Estructura del sistema.* Intentando resolver tantos objetivos como nos sea posible a lo largo del capítulo. Además, se explica el fundamento de la solución adoptada.

En el Anexo I se incluye un esquemático de todo el **conexionado hardware del sistema**, cumpliendo las características del *Capítulo 2: Componentes del sistema.*

En el Anexo II se incluye el **código fuente**. Para leer las descripciones de las funciones y variables, leer en el código comentado de las secciones.

Carpeta MiniDK-2.

En esta carpeta se incluyen todas las carpetas del proyecto.

Carpeta README.

Contiene información del proyecto.

Carpeta Startup.

Contiene los ficheros de inicialización del sistema, así como librerías internas que utiliza el LPC1768 para iniciar. Incluyen las librerías:

- Startup_LPC17XX.s: Inicializa el sistema. La única variable modificada en este archivo es Stack_Size y se le ha otorgado un valor de 0x200.
- Core cm3.c: Inicializa el core.
- System LPC17XX.c: Ajusta variables y define macros internos sobre todo de los relojes de la placa MiniDK-2.

Carpeta Principal.

Contiene el programa principal en su interior, dentro del archivo main.c. En el programa principal se crean variables globales y punteros a dichas variables globales, para utilizarlas de manera más sencilla.

Dentro del programa principal sólo es necesario llamar a la función de configuración del sistema. Luego es necesario crear un bucle infinito para llamar a la función __mainLoop__() que está definida en Statechart.c y a la función __mantenerTCP__() que se encuentra en el archivo HTTP_SOURCE.c. La función __mainLoop__() es la máquina de estados que rige la interfaz de usuario por el LCD y __mantenerTCP__() mantiene la conexión TCP mediante la librería de TCP haciendo un llamado a la función main_TcpNet().

Cabe destacar que no todas las variables importadas se encuentran en este archivo, dado que sólo residen las de mayor importancia como pueden ser los datos almacenados y señalizadores del sistema.

Carpeta Menu.

En esta carpeta se encuentra la librería TouchPanel.c y el archivo responsable de generar la máquina de estados, Satatechart.c. En la *Figura 1.* Podemos observar la máquina de estados implementada en el archivo. La función __mainLoop__() se encarga de generar la máquina de estados, utilizando la variable global ESATADO como una variable de 8 bits que almacena el valor del estado que se quiere asignar. En este archivo se declara la variable MODIFICABLES, que contiene todos los valores modificables por el usuario en cualquiera de sus interfaces mencionados anteriormente.

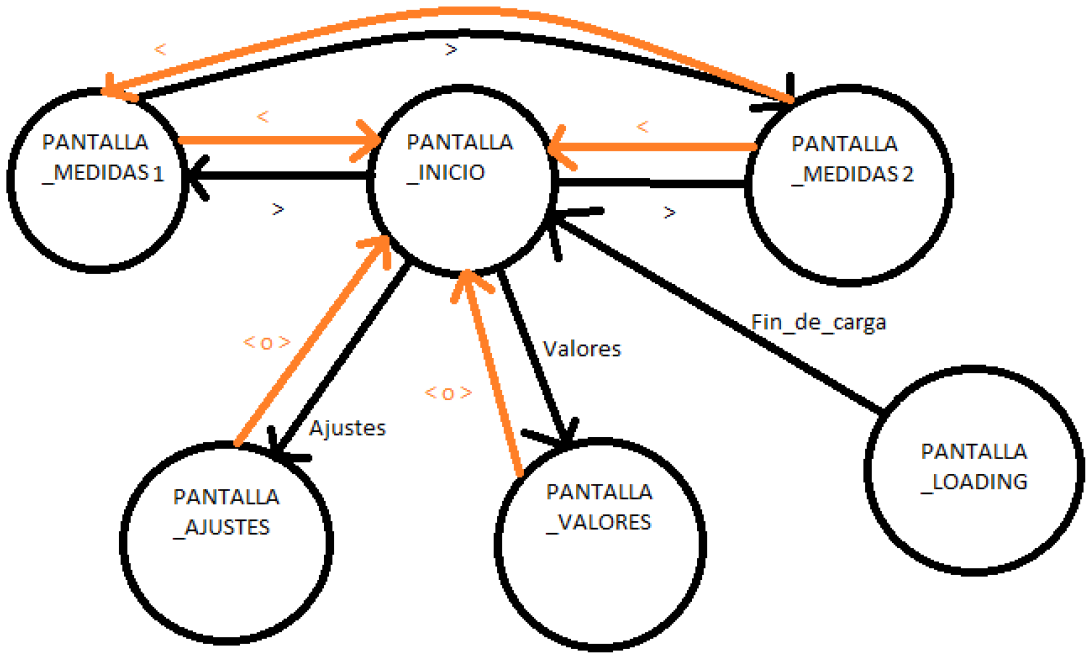


Figura 1.- Modelo de pantallas.

Además, se definen variables como Modo_brillo, Modo_energetico que guardan el estado del brillo, es decir, cualquiera de sus 4 niveles de brillo y el modo de ahorro de energía que se encuentra el sistema. La variable pressedTouchPannel refleja si se ha tocado la pantalla, para que se ilumine cuando esta, en su modo de ultra ahorro de energía; ULPM, se haya apagado.

Las variables internas __brilloFade y __brilloAuto, son variables que guardan si el sistema debe de apagar la pantalla o no (__brilloFade) y si deben de leer el LDR para proporcionar un control automático de brillo (__brilloAuto). El sistema cada iteración de la función principal, comprueba si __brilloAuto con ayuda del SysTickTimer cada 100ms si el estado es de brillo automático y actualiza el módulo PWM para proporcionar el pulso adecuado para generar un nivel de brillo que oscila entre el 20% y el 80% de luminosidad. Esto lo hace refiriéndose a la tabla en el archivo LUT.c, contretamente en el array Brillo2ciclo_LDR, que traduce el brillo al porcentaje del ciclo de trabajo del módulo PWM1.

Se definen variables de tipo zona que son variables que necesita la librería para generar la interfaz. Se le pasa como parámetro el exe X e Y y la longitud en X y en Y que ocupa el recuadro generado. Además, dichas variables admiten color y texto tanto como de la línea como del texto.

En el archivo se encuentran las siguientes funciones:

- __pintaX__(): pinta la pantalla X.
  Dentro de estas funciones puede haber más estados que representen un tipo de color, por ejemplo, existen unas barras de colores que son interfaces gráficas para que el usuario sepa la temperatura que hace en función de los valores máximos y mínimos de la misma, cambiando el color en función del porcentaje.
  Dentro de estas funciones se utilizan actualizadores, estos actualizadores sirven para actualizar el valor de los datos, es decir, mostrarlos por pantalla una vez sean medidos. Cuando un módulo termine de realizar una medida, este módulo debe de hacer un toogle al actualizador correspondiente para indicarle a la máquina de estados que se ha realizado un cambio en el valor de la medida y así poder mostrarla por pantalla.
- __configuraLCD__(): configura el LCD.
- squareButton(): Dibuja un botón cuadrado en función de la zona, los colores y un texto.
- squareBox(): Dibuja una caja en una zona de un color.
- checkTouchPanel(): Comprueba si se ha tocado un botón, además si se ha tocado cualquier zona de la pantalla, resetea el contador que apaga la pantalla en ULPM.
- ZoneNewPressed(): Comprueba si se ha tocado cierta zona de la pantalla, sirve para saber qué botón se ha pulsado. Si se toca la pantalla cuando esta ha sido apagada (esto sucede dentro del manejador del SysTickTimer) esta se recupera con un ciclo del 60% de PWM, es decir al 60% del brillo. Además, recupera las variables anteriormente mencionadas y resetea el contadorLuz, que es el contador encargado de controlar que no se supere el tiempo de apagar pantalla (este tiempo es modificable).

Dentro de cada pantalla, podemos visualizar o modificar variables.

- En la pantalla de medidas 1: Podemos ver los siguientes valores.
  - Velocidad del viento en m/s.
  - Humedad del aire en % relativo.
  - Claridad recibida en LUX.
  - Índice UV en UVs.
  - La altura respecto al nivel del mar en metros.
- En la pantalla de medidas 2:
  - Temperatura en °C.
  - Presión en mBar.
  - Gráfico de barras horizontales de colores para los valores máximos y mínimos de cada una.
- En la pantalla de ajustes: Podemos visualizar la IP del servidor WEB y además modificar las horas, minutos y segundos del reloj, además del día.
- En la pantalla de valores: Podemos modificar los valores de presión y temperatura máximos y mínimos que generan alarma, además de poder modificar si el servo representa la temperatura o la presión. El botón tendrá letras verdes para la presión y letras rojas para la temperatura. En dicho botón se representan los rangos máximos y mínimos de temperatura y presión a representar.

- En la pantalla de inicio: Podemos, además de acceder a otras pantallas, modificar el brillo, siendo 1 el mínimo y 4 el máximo. Además, está la selección de A que es brillo automático, lo que automáticamente le hace cambiar a LPM (Low power mode). También podemos grabar y reproducir audio con los botones de load y play, que lanzan las señales de grabar y reproducir audio. Desde esta pantalla podemos modificar el modo de ahorro de energía.

Importante:

Dentro de la librería TouchPanel.c, podemos encontrar una matriz denominada como matrix. Esta matriz contiene unos valores de calibración para calibrar la pantalla, cada una tiene sus valores y deberían de ser ajustados por el usuario. En caso de querer calibrar manualmente la pantalla cada vez que se inicia el sistema, descimentar la línea 68 del archivo configura.c, donde dice: //TouchPanel_Calibrate();.

Carpeta Setup.

En esta carpeta se encuentra el archivo de configuración configura.c. Se encarga de mandar a pintar las diversas pantallas de carga, de iniciar las variables y de mandar a configurar todos los módulos. Además, inicia el módulo PWM con el 90% de ciclo para el servo y 50% de brillo. Inicia las variables a cero y las modificables con el valor de las macros.

Temperatura: [-10,50]°C.

Presión: [500,1500]mBar.

Tiempo de brillo: [10]seg.

Variable medida:Temperatura.

Carpeta I2C.

En esta carpeta hay una librería de I2C (I2Clib.c) que contiene todo lo necesario para realizar las llamadas a las funciones de comunicación de I2C. (Mandar un byte, recibir un byte, recibir una dirección). Luego está el archivo que se encarga de utilizar la librería de I2C para leer el sensor BMP180.

En el archivo I2C.c podemos encontrar funciones que se encargan de usar la librería para leer los registros de acuerdo con la datasheet del fabricante (ver la hoja de datos de BMP180). Utiliza la lectura de ciertos registros del sensor para calibrarlo y luego llama a medirBMP(), que utiliza un algoritmo que leyendo ciertos registros y trabajando con ellos podemos obtener la temperatura y la presión. Cuando se desee medir, solo hace falta llamar a medirBMP() para que automáticamente guarde en la variable DATOS la presión y temperatura. Además, se calcula la altura sobre el nivel del mar con una fórmula proporcionada por el fabricante.

Nota: La función procesarDato() admite parámetros que en versiones anteriores servían para elegir presión o temperatura, en la versión final se incluyen la altura, temperatura y presión. Se ha decidido tomar

la temperatura con este sensor, dado que es de mayor calidad que el DHT22.

Carpeta Anemómetro.

En esta carpeta se encuentra Anemometro.c que incluye dos funciones, una de configuración y otra de medición. El anemómetro funciona de la siguiente manera: necesita una resistencia de pull up (vale la configuración estándar de la placa), dado que cada cuarto de vuelta del ciclo de giro cortocircuita ambas patillas, por lo que la conexión debe de ser input-masa, la resistencia de pull up debe de estar en el lado de input. Cada vuelta genera 2 pulsos, dado que cortocircuita en dos cuartos de ciclo, por lo que el anemómetro genera 2 pulsos por vuelta.

Dicho esto, podemos utilizar el módulo CAPTURE del Timer 1 para calcular la diferencia de tiempos entre el inicio o fin de ambos flancos (subida o bajada a elegir). Podemos calcular la velocidad que recorre el viento de la siguiente manera:

$$Distancia_{recorrida} = Perímetro_{anemometro} = \pi \cdot D_{anemómetro}$$

$$Tiempo_{medidoCapture} = \frac{Perímetro_{anemómetro}}{Pulsos_{vuelta} \cdot Velocidad_{viento}}$$

Podemos esperar a que se realicen dos pulsos y luego medir como si se hubiese realizado un pulso para reducir carga computacional:

$$Velocidad_{viento} = \frac{Perímetro_{anemómetro}}{Diferencia_{pulsos}} = \pi \cdot \frac{D}{T_{reloj} \cdot (CAPTURE_x - CAPTURE_{x-1})}$$

Medidas realizadas estiman el diámetro del anemómetro en **14 centímetros**.

Carpeta ADC.

En esta carpeta están los archivos necesarios para hacer funcionar todo el módulo del ADC y leer los sensores LDR y UVA30A. Además, se encuentra la lectura del micrófono. Los archivos inherentes son Ldr.c, uFono.c y UVA30A.c.

En el archivo LDR.c contiene toda la configuración del ADC, que se ejecuta en modo BURST. Dicha configuración realiza conversiones a todos los canales activados, que en este caso es el del LDR y el del UVA, podrían haberse metido más sensores sin ningún problema y sin aumentar la carga computacional, solo que se utilizan más pines del ADC en este modo BURST. El valor de CLKDIV es el máximo que se puede poner (0xFF), por dejar tiempo al ADC a que convierta con tranquilidad, podría reducirse dicho valor hasta cierto punto.

La interrupción que hace leer los registros del ADC es la penúltima, dado que hay un 'lag' de una conversión desde que se muestrea hasta que se produce la llamada al Handler del ADC. Cuando el Handler entra en acción, el último canal ya ha sido convertido. El penúltimo canal es del del LDR (1), por lo que la interrupción la provoca este fin de conversión.

En el archivo UVA30A.c sólo se configura el ADC para permitir un canal más en modo BURST, si no se ha configurado el LDR, el configurador del

UVA30A llama al del LDR primero, dado que hay que configurar todo el ADC primero para configurar que el UVA entre en modo BURST. El canal asociado al UVA es el 2.

Una vez teniendo las medidas en modo BURST, cuando se desee grabar el micrófono, se lanzará una señal que lo indique. Esta señal es la función lanzaUFONO(), que configura todo el ADC y el Timer 1 para que se obtengan muestras cada MATCH0, pare la conversión de audio y vuelva a medir en modo BURST. Cabe destacar que las medidas se bloquean mientras el modo audio está activado. Este bloque lo representa la variable YaPuedesMedir, que además se utiliza en el Handler del ADC para saber en qué modo (si BURST o audio) se encuentra el sistema. Una vez alcanzado el número de muestras del audio, se lanza la desactivación del ADC en modo audio y se recupera el contexto del ADC para el modo BURST, además de activar los actualizadores y de marcar que ya se puede medir en modo BURST.

La configuración del micrófono configura el ADC para que cada 8000kHz se obtenga una muestra de audio y se guarde en la variable AUDIO, que ocupa un total de 8 bits por muestra a 16000 bytes. 16kB de audio.

Dentro del manejador de la interrupción del ADC podemos encontrar dos modos:

- YaPuedesMedir = 0: Mete las muestras recibidas por el canal 0 al array definido para el AUDIO, cuando se alcanza el número de muestras MUESTRAS_AUDIO, se reconfigura el ADC para medir en modo BURST.
- YaPuedesMedir = 1: Lee los canales 1 y 2 del ADC por cada interrupción.
- Para obtener el brillo: Se lee el canal 1 del ADC, se calcula el porcentaje relativo de voltios del canal referido a 3.3V; luego se calcula su resistencia y se pasa dicho valor a una look-up-table para traducir dicho valor a LUX.

$$R_{LDR} = R_{pull} \cdot \frac{\frac{ADC_1}{0xFFF}}{1 - \frac{ADC_1}{0xFFF}}$$

- Para obtener el índice UV: Es una función lineal, por lo que se obtiene el porcentaje relativo respecto a 3.3V del canal 2 de entrada y se multiplica por el máximo.

$$Indice_{UV} = Indice_{maximo} \cdot 3.3V \cdot \frac{ADC_2}{0xFFF}$$

La resistencia de pull-up escogida son 70kOhm, debido a que linealiza mucho su respuesta. Es muy importante escoger una resistencia de pull-up adecuada, debido a que una pequeña variación en el error podría producir una gran variación de la resistencia. A continuación, gráficas que muestran dicho comportamiento.

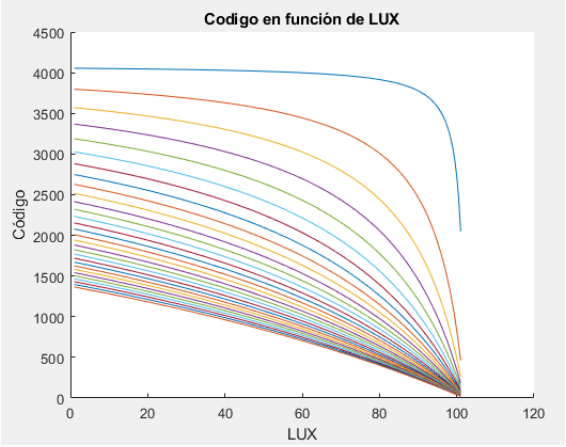


Figura 2.- Código en función de LUX.

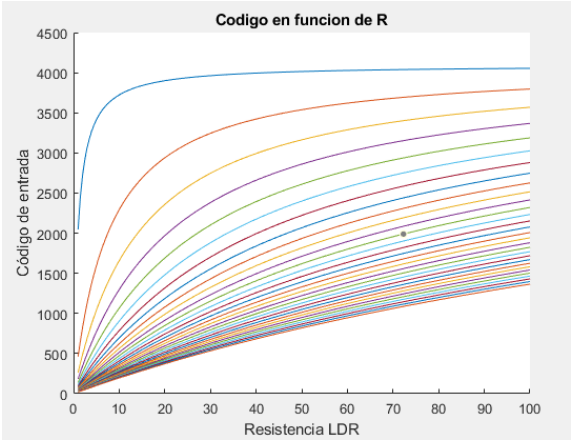


Figura 3.- 2 En función de los ohm.

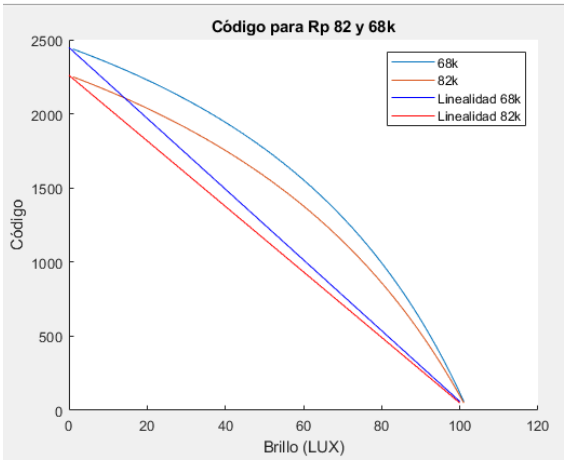


Figura 4.- Linealidad a 68 y 82k.

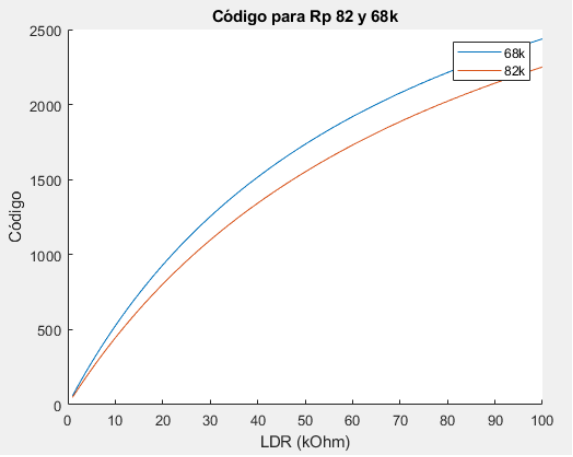


Figura 5.- 4 En función de los ohm.

Se han escogido valores de 68kohm debido a que son valores estándar y tienen una buena respuesta en la linealidad sin sacrificar mucho porcentaje del código que se ha recibido. Perdemos la mitad del rango, que podría arreglarse seleccionado una fuente de voltaje adecuada, pero ganamos linealidad y por ende, más margen de error.

Carpeta OneWire.

Pese a que el código de este apartado es muy claro y este sí está bien comentado, explico por encima las funciones que realiza.

En la datasheet del DTH22 especifican un protocolo de lectura de monofilo, lo que requiere que exista una configuración dinámica del pin de datos, para mandar una señal de inicio de conversión y configurar dicho GPIO como entrada. He utilizado GPIO como forma de lectura utilizando esperas activas del orden de los microsegundos ayudándome de el contador del Timer 3, dado que este no lo toca ningún otro módulo y estaba sin usar. Dado que realizando pruebas este sensor tiene una cantidad de ruido importante, utilizar el modo capture era realmente un desafío. Por lo que he optado utilizar esperas activas, que no dañan a penas la ejecutabilidad del programa dado que la relación ejecución/uso (0.076% del tiempo) es muy baja; se llama cada 5 segundos a la función en el Handler del Timer 0 y ocupa 3.8ms de media en realizarse. Algo parecido pasa con I2C que utiliza librerías de espera activa.

En este archivo se ejecuta el protcolo y se almacena en una variable local a mideTemperatura(), que es la señal de inicio de medida, que se llama Rx. Se divide en 4 etapas, marcadas en el código en el marcador @state:

- Inicio de medida: Se configura el pin como salida de datos y se crea un pull down, luego se configura como entrada.
- Señal de respuesta: Para esta señal, se espera recibir 50us de nivel bajo más otros 50 de nivel alto, con un margen de 10us. Esto se ejecuta sobra la función compruebarespuesta(). No se tiene en cuenta el tiempo que tarda el sensor en volver a poner el tiempo en alto, por lo que hay que esperar a que lo haga, con un valor de timeout de 45us.
- Lectura de datos: Se leen los datos en función de la duración del flanco de subida, si se excede el tiempo de timeout, se aborta la medición.
- Comprobación de checksum: Se lee el último byte y se compara con el algoritmo de checksum del fabricante. Si coincide, se insertan los nuevos datos, si no coincide se aborta la medida y se espera a la siguiente llamada.

Existe la función reinicia cuenta, que devuelve el valor del contador TC, que está reducido por un valor de 25, lo que provoca cuentas cada 1 us, y acto seguido lo reinicia. Por eso podemos obtener el valor del tiempo fácilmente.

Cabe destacar que si no se resuelven los márgenes, se sale de la función con un código de salida de 1 en la función compruebaRespuesta() y un código de 0 para la función LeerByte(). Los valores de timeout de datos son de 100us, si no se ha recibido un cambio en ese periodo, se sale de la medición y queda abortada.

Carpeta PWM.

En esta carpeta se encuentra todo lo referente al módulo PWM, residente en el archivo PWM.c. Existen dos funciones: __configuraPWM__() y modificaPulso(). El código está comentado y explica todos los input y output de las funciones.

- La función de configuración lo que hace es recibir una frecuencia para el pulso de PWM y los puertos y pines que se quieren activar. En nuestro caso sólo los pines 2_1 y 6_1, uno para el servomotor y otro para el brillo del LCD.
- La función modifica pulso, lo que hace es modificar el ciclo del pulso PWM, para cambiar su frecuencia, hay que reconfigurarlo. Las entradas son:
  - PWMn: El pin PWM seleccionado.
  - Modo: Indica si el siguiente argumento es para el servo o para el LCD.
  - Ciclo: Si es modo ciclo, se utiliza este ciclo de trabajo del módulo PWM.
  - Grados: Si es modo servo, se utiliza este argumento como los grados a los que debe inclinarse la aguja del servo.

  PARA EL MODO SERVO:
  - Mínimo: Valor mínimo del pulso a nivel alto en segundos.
  - Máximo: Valor máximo del pulso a nivel alto en segundos.

IMPORTANTE: Cabe destacar que el servomotor utiliza ciclos de 5V, cuando la placa LPC1768 utiliza ciclos de 3.3V. Esto es un problema dado que el servo parece utilizar la potencia del pulso y no precisamente su duración, es por ello por lo que no funciona de 0 a 180° y funciona de 45° a 135°. Esto puede corregirse modificando los valores de duración mínimo y máximo del pulso.

Experimentalmente se ha decidido que los valores por los que hay que multiplicar dichas constantes sean las siguientes:

- KMX: Constante del máximo = 1.3
- KMN: Constante del mínimo = 0.6

Lo que implica proporcionar más potencia al servo con la señal de 3.3V, dado que el mínimo disminuye y el máximo aumenta.

Las fórmulas utilizadas son, en función del ciclo o los grados, las siguientes:

$$MR_x = MR_0 \cdot \frac{Ciclo}{100}$$

$$MR_x = \left(Maximo + (Maximo - Minimo) \cdot \frac{Grados}{180}\right) \cdot \frac{1}{T_{reloj}} - 1$$

Carpeta DAC.

En esta carpeta se encuentra todo lo referente a la señalización y configuración del DAC. Para configurarlo, se utiliza el prototipo de función que se ha utilizado hasta ahora: __configuraDAC__(), que incluye la puesta a nivel alto de los LED de la placa, cuando están encendidos significa que está disponible la escritura de audio. Lo mismo para la lectura de audio. Esta función únicamente configura esos pines dado que el DAC se utiliza con el DMA y por ende se configura junto a él.

Las funciones activarDac() y desactivarDac() realizan la función de activación o desactivación del DAC.

Al activar el DAC, se activa el canal correspondiente del DMA, es decir, se manda una señal de incio. Además, se configura un Timer para llamar a desactivarDac() pasados los 2 segundos. Se apaga el LED de la placa que señaliza escritura de audio.

Para desactivar el DAC, se señala al sistema con el actualizador que se ha acabado el tiempo del DAC y desactiva el canal del DMA, es decir, se manda una señal al DMA de fin. Además, se enciende el led correspondiente a escritura de audio y se escribe un valor de 0 en el DAC, dado que no hay señal de salida una vez finalizada la conversión.

Carpeta Database.

En esta carpeta se encuentran todos los archivos referentes a look-up-tables (LUT.c) y transferencia de memoria (DMA.c). En el archivo LUT.c podemos ver que reside la función goto_LUT que admite los siguientes parámetros:

- Variable: Variable de entrada.
- LUTn: El tipo de tabla que vamos a usar.
- Ret_x: Son un grupo de parámetros por los que hay que señalar una dirección en la que se va a guardar el resultado. Los tipos admisibles son de enteros sin signo de 8 a 64 bits y flotantes.

Las tablas que existen son Brillo_LDR, que traduce el valor en ohmios medido por el ADC a brillo, y Brillo2Ciclo_LDR, que traduce la variable brillo o claridad expresada en LUX a un ciclo de trabajo para controlar el servomotor. Sólo el LDR utiliza estas tablas.

En el archivo de DMA.c existen tres funciones y las tres son de configuración.

La primera función es la función __configuraDMA__(), que genera un tono de 32 muestras y activa la salida analógica (P0.26). Luego hace una llamada a la función __configuraTono__(), que deja configurado el audio pregrabado, que es un tono de 400Hz. Este es activado si se supera el valor mínimo de la alarma. Si se superase el valor máximo, se llamaría a la función __configuraAudio__(), esta función configura el DMA para reproducir el audio grabado. En ambos casos las muestras son de 8 bits y se crea una estructura (LLI0) que contiene las direcciones de origen y destino de las transferencias, así como que el destino se incremente y el número de muestras a tranferir. Además, en ambos se configura que la transferencia es de memoria a periférico y que se transfiere al DAC. Dentro del DAC, hay que configurar que las transferencias son realizadas vía DMA.

La frecuencia a la que se transfieren las muestras son las siguientes:

- Para el tono:

$$T_{dma} = \frac{Num_{muestras}}{400Hz}$$

- Para el audio:

$$T_{dma} = \frac{Duracion_{audio}}{F_s} = 4kHz$$

Las funciones __configuraTono__() y __configuraAudio__() son llamadas si se exceden los valores límite de alarma, para que el DMA lance la transferencia ya sea por audio o por alarma pregrabada. Tras configurar el DMA así, se lanza un Timer (Timer 1), que desactiva el canal pasados 2 segundos.

Carpeta WDT.

Esta carpeta contiene los archivos referentes al WatchDogTimer, en WDT.c. Existen dos funciones:

- __configuraWDT__(): Se encarga de configurar el WDT, la configuración seleccionada es de un timeout de 10 segundos utilizando el reloj de Fclk/4. La acción a realizar tras vencer el temporizador es de reiniciar el sistema, por lo que es muy importante estar alimentando el WTD continuamente.
- alimentaWDT(): Es la función que reinicia el contador del WDT. Se escribe en su registro de alimentación el código 1 y acto seguido el código 2 para reiniciarlo y evitar el reinicio del sistema. El código 1 se define como 0xAA y el código 2 se define como 0x55.

Esto se hace cada ciclo de __mainLoop__(). Si no se entra a dicha función en 10 segundos, este temporizador reinicia el sistema. Esto evita bloqueos indeseados.

Carpeta UART.

Esta carpeta utiliza una librería proporcionada por la signatura SEDA de la UAH. Y el archivo UART0.c.

Dentro del archivo UART0.c podemos observar dos funciones, una de configuración y una de procesado del comando.

Respecto a la configuración, la función hace referencia a la librería para configurar, con 9600 baudios, el UART0 que es la conexión por USB (controlador de UART de la placa). En la librería se tiene una función para el cálculo de los registros de configuración de los baudios con un método iterativo. También se encuentra el manejador de la interrupción de UART0, que se activa cada byte recibido y por cada vez que el buffer de transmisión esté lleno.

Cuando se envía una cadena, hay que usar la función de la librería tx_cadena_UART0.

Para procesar el comando recibido, se guarda cada byte en un buffer (bufferx) y al recibir el carácter 13 (\r) se considera como acabado y se llama a la función de procesar comando: procesarComando(char * string).

Esta función tiene una máquina de estados implementada como se muestra en la figura 6. Donde se espera a recibir un tipo de comando para ejecutar una determinada acción. Los comandos pueden observarse en el Capítulo 6: Manual de usuario, en el apartado de UART.

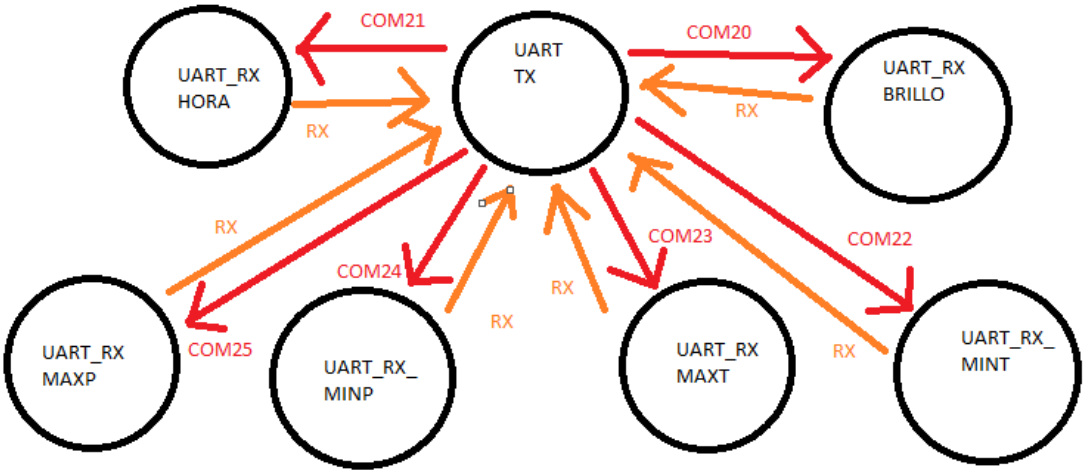


Figura 6.- Diagrama de estados de UART0.

Una vez obtenido el comando de modificación necesario, se procede a introducir el valor de la variable a modificar, el nombre del estado hace alusión a la variable a modificar. (Ver manual de usuario para más información sobre los comandos y el formato de salida).

La cadena a transmitir se guarda en un buffer y luego se transmite (UART0_BUFFER_TX[]) con la función de la librería (tx_cadena_UART0).

Carpeta TCP/IP.

En esta carpeta se encuentran los archivos referentes a la pila de protocolos TCP/IP que forman en servidor web. Se incluyen dos librerías: TCP_CM3.lib (la del fabricante) y EMAC_LPC17XX_LAN8720.c (la que se encarga del servicio sobre ethernet). Además, existe el archivo WEB.inp que guarda la orden de compilación:

index.cgi to WEB.C nopr root (../TCPIP)

Esta orden manda compilar el archivo CGI a otro llamado WEB.C que contiene la página web, convirtiendo el archivo de marcado en un objeto de C.

Luego se encuentran los siguientes archivos:

- HTTP_SOURCE.c: Contiene la configuración de la página web (__configuraWEB__()) que llama a la función init_TcpNet() de la librería, que inicia la conexión TCP. Además, encapsulo en __mantenerTCP__() a la función de la librería main_TcpNet() que es la que hay que llamar con cierta frecuencia (se encuentra en el programa main.c).
- NET_Config.c: Contiene la configuración de la conexión TCP, lo más relativo de este fichero se encuentra definido en el archivo miGlobal.h, que define la IP del servidor, el Gateway y la máscara subred. En este archivo se define con DEFAULT la ip 192.168.1.120 con un Gateway de 192.168.1.20 y una máscara subred de 255.255.255.0. Además, el usuario y la contraseña se definen como user y Alver respectivamente.
- Index.cgi: contiene el lenguaje de marcado en formato CGI y html de la página web a compilar. Esta pasa a convertirse en WEB.C.
- HTTP_CGI.c: Se encarga de la parte automática de la página web, mediante llamadas a callbacks y al método GET.

- Callbacks: se encuentra en la función cgi_func, función llamada cuando una línea de index.cgi empieza por la letra c. Se utiliza un espacio y una letra para determinar la acción de dicha callback:
  t para temperatura.
  v para velocidad del viento.
  p para presión.
  h para humedad.
  i para índice UV.
  b para brillo.
  a para altitudo.
  X no usado.
  Y no usado.
  A año.
  M mes.
  D día.
  H horas.
  T minutos.
  S segundos.
  Así podemos sacar por este callback y la función sprintf la variable en cuestión y poder representarla en el formato de la página WEB.
- GET: Se utiliza para enviar comandos a la estación, se define una tabla en la que se pueden escribir valores para enviar a la estación, todos los modificables comentados con anterioridad exceptuando la hora, que es modificable por pantalla y por UART. Primero se lee de la cadena si coincide la variable con las siguientes cadenas:
  Tmin= para temperatura mínimo.
  Tmax= para temperatura máxima.
  Pmin= para presión mínima.
  Pmax= para presión máxima.
  Vart= para seleccionar temperatura.
  Varp= para seleccionar presión.
  Si coincide con dicha string, se pasa a procesar el número que viene a continuación y se guarda en la variable en el campo de la estructura correspondiente MODIFICABLES.[campo].

Carpeta RTC.

Dentro de esta carpeta, se encuentra el archivo RTC.c, lo que contiene la configuración y el manejador de la interrupción del RTC. Se configura con __configuraRTC__() y se incicia el reloj a principios de año de 2020. Se configura para interrumpir cada segundo para actualizar una variable llamada Clock, que contiene una string con el valor de la fecha actual, además tiene un contador de segundos.

Carpeta Timers.

En esta carpeta se encuentra todo lo referido a los Timers, es la carpeta más importante dado que es donde se hacen todas las llamadas a las funciones de las demás carpetas. Dentro de esta carpeta se encuentran las configuraciones y los manejadores de interrupción de los temporizadores y del SysTickTimer.

- __configuraSysTick__(): Es la función que configura el SysTick para interrumpir cada 100ms.
- SysTick_Handler(): En esta función se hace una llamada a una función necesaria para la librería TCP/IP: timer_Tick(). Además, incrementa un contador (contadorLUZ) cada 100ms. Si este alcanza el tiempo límite sin tocar la pantalla (recuerdo que tocar la pantalla reinicia dicho contador) y este se encuentra en ULPM, esta desactiva el brillo automático y pasa a mandar una señal PWM de brillo del 1%. Por lo que se apaga, dejando un pequeño margen de luz del 1%.
- __configuraTimer0__(): Configura el Timer0 para interrumpir cada 0.5 segundos. En esta configuración se activan todos los timer a usar, aunque no el manejador de sus interrupciones.
- Timer0_IRQHandler(): En este manejador, tenemos tres funciones: medir el grupo 1 de sensores cada 0.5 segundos, medir el grupo 2 de sensores cada 5 segundos y actualizar el servo cada 0.5 segundos. Este temporizador controla el tiempo de muestreo.
- Timer1_IRQHandler(): Si interrumpe el MR1, se considera interrupción por fin de salida del DAC y se desactiva el mismo. Si interrumpe el modo capture del (CAP1.0), se considera que ha llegado un pulso del anemómetro y se llama a la función que lo mide. (Hay que tener en cuenta que quien activa el anemómetro es el Timer0 y se desactiva solo tras obtener una medida, si este no manda un pulso, la velocidad saldrá como 0 m/s indicando que no existe viento en ese momento).

Carpeta SDCARD.

Esta carpeta contiene los ficheros diskio.c, ff.c, SPI_MSD_Driver.c en caso de que en un futuro se quiera utilizar la tarjeta SD en la estación meteorológica. En esta versión no se incluye la tarjeta SD.

Carpeta CMSIS.

En esta carpeta se encuentra el software del fabricante CMSIS para utilizar en las librerías, varias de ellas utilizan los archivos de esta carpeta. Entre ellos se encuentran:

- lpc17xx_ssp.c.
- lpc17xx_pinsel.c
- lpc17xx_gpio.c
- lpc17xx_clkpwr.c
- lpc17xx_libcfg_default.c

Carpeta GLCD.

En esta carpeta se encuenta la librería que se encarga de comunicar los pines de la placa con el TFT, utilizando funciones para mandar datos por dichos pines. (Ver esquema del Anexo I).

Capítulo 5: Pruebas, ejecutabilidad y conclusiones.

En este capítulo se explican los bugs que puede llegar a tener la estación meteorológica en esta versión con pruebas que se han realizado. Luego un estudio de ejecutabilidad y una recapitulación de objetivos logrados.

Pruebas.

Pruebas realizadas con la estación concluyen que puede haber los siguientes bugs:

- Aparición de números en las medidas no deseables: Cuando se realiza una medida de una variable que por un momento se ha excedido de un valor alto, puede que, si al no cambiar de pantalla el valor se reduce lo suficiente como para pasar de 3 a 2 dígitos, por ejemplo, el último dígito no sea borrado de la pantalla y parezca que sigue ahí. Esto es resuelto cambiando de pantalla. Puede observarse reduciendo valores de alarma de presión cómo la letra acaba por duplicado. El error viene de las funciones de la librería TouchPanel.c.

- Caída de la conexión TCP: Depende del tiempo de ejecución y de lo que se haya realizado hasta el momento, puede haber una caída de la conexión TCP y que la página WEB se caiga, esto se resuelve reiniciando el sistema. La media medida en la que se produce este fallo oscila entre los 9 o 10 minutos de ejecución del programa, habiendo veces que no sucede y habiendo veces que al minuto 2 está caída.

- Bloqueo del sistema por caída de la conexión TCP: Depende de si se ha caído la conexión TCP o no, puede bloquearse el sistema por un error que denomina la librería como ERROR_FREE_MEMORY, que es un error de liberación de memoria. Inmediatamente este error reinicia el sistema debido al WatchDogTimer. Si no se cae la conexión TCP, este error no sucede. La media medida de este error oscila las 2:30 horas de ejecución, aunque si no se produce la caída no se forma, y aun produciéndose puede tardar bastante más en caerse. Su record medido es de 6 horas después de caerse la conexión TCP a los 8 minutos de ejecución. Este error es resuelto automáticamente con el WatchDogTimer.

Las medidas de la estación son bastante precisas para la calidad y el precio de los sensores utilizados. Aunque las pruebas realizadas no han sido posibles con el sensor UVA30A, se ha probado a simular su comportamiento y el comportamiento es correcto.

Ejecutabilidad.

Con el simulador de Keil uVision4 podemos obtener la diferencia de tiempos entre dos breakpoints y medir las diferentes tareas que componen el sistema, en la Tabla 1 se muestran las tareas, el tiempo máximo que hay para realizarlas, el tiempo mínimo que tardan en ser requeridas y el tiempo máximo que tardan en ejecutarse.

Tarea	Deadline (D)	T.mín (T)	T.ejec (R)	Unidades
RTC	1000	1000	0.41	ms
Timer0	500	500	1.21+0.69	ms
Timer0*	5000	5000	10.32	ms
SysTick	100	100	0.92	ms
UART	1.04	1.04	0.12	ms
ADC (microfono)	0.125	0.125	0.10	ms
WDT / Statechart.	10000	10000	108.729	ms

Tabla 1.- Datos de ejecutabilidad del sistema.

El RTC al interrumpir cada segundo y actualizar el reloj, su D = T es de ese segundo.

El Timer0 interrumpe cada 0.5s, siendo cada 5s el mayor tiempo de ejecución que tiene, incluyendo el protocolo OneWire e I2C que son los que incluyen esperas activas, es por eso por lo que incluyo dos Timer0, el largo cada 5s y el corto cada 500ms.

El SysTick tiene 100ms para mantener abierta la pila de protocolos TCP.

El Timer1 no tiene tiempo mínimo de ejecución, dado que lee los pulsos del anemómetro y cuando necesite y se cierra, esto está incluido en el tiempo de ejecución del Timer0 cada 500ms.

UART tiene que mandar 9600 bits por segundo y lanza 11 bits, 8 de datos y 1 de start y stop y otro de paridad.

WDT reinicia al sistema cada 10s, por lo que hay que el tiempo máximo es de 10s.

El ADC tiene una frecuencia de muestreo de 8000kHz, por lo que su tiempo de muestreo es el máximo que tenemos para ejecutar las funciones dentro del manejador.

Nota: Los valores 0.001 son valores que no se han conseguido medir dado que el simulador los ha considerado despreciables, dado que rondan el orden de las 3 o 4 líneas de código.

Nada es bloqueante excepto el ADC, pero como se ha comentado con anterioridad, cuando se mide con el ADC la voz, se bloquean las medidas, por lo que el Timer0 no ejecuta código. La suma de la duración de todas las tareas no es mayor que los 100ms del SysTick. Luego está la tarea del Statechart, pero al no estar en una interrupción su prioridad es mínima, por lo que la máxima interrupción de Statechart sería de 12ms aproximadamente en caso de que todo interrumpiese a la vez. Cosa que el usuario no notaría para nada.

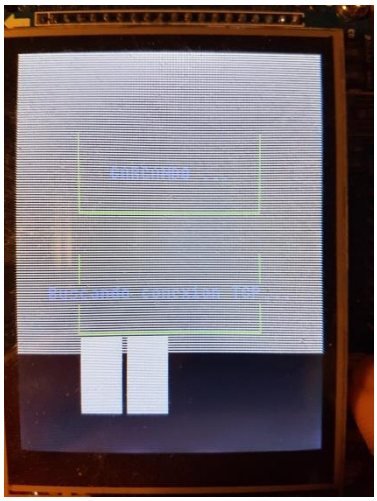
Conclusiones.

La práctica en su totalidad está completa, aunque no se han añadido muchos de los objetivos opcionales (sólo el RTC y el DMA y alguna mejora de la interfaz de pantalla de usuario), por lo general funciona muy bien en cuanto a lectura de sensores y funcionamiento de pantalla s refiere. La UART también funciona muy bien, aunque personalmente me gustaría mejorar los bugs que tiene la pila de protocolos TCP/IP que bloquean el sistema tras 4 o 5 horas de ejecución, de hecho, si comento la línea de configuración de TCP/IP, el sistema no se bloquea nunca al no ser que se mande el comando KILL (ver manual de usuario).

La parte más costosa ha sido el ADC y el DMA junto con la página WEB, dado que he tenido muchos problemas al leer el audio y reproducirlo. La poca documentación sobre DMA que disponía me ha hecho tener que referirme a ejemplos colgados en internet y el largo tiempo que me ha llevado modificar pequeñas variables de la página web para que funcione ha sido determinante para centrarme en la calidad de esta. Además, no disponía ninguna experiencia en html ni en diseño web, por lo que la página es un poco simplista, aunque hace lo que se requiere de ella sin muchos problemas excepto ese bloqueo ocasional.

Capítulo 6: Manual de usuario.

En este capítulo se redacta lo necesario que debe conocer alguien que no sabe el comportamiento interno de la estación y solo desea utilizarla, transparentemente del comportamiento interno que la misma realice. A continuación, para las diferentes interfaces, se resume su manual.



TFT.

La pantalla al ser iniciada muestra por pantalla una interfaz que muestra el porcentaje de carga o pasos realizados para finalizar su configuración. Una vez realizada, se pasa a la pantalla de inicio. La primera barra indica que se ha iniciado el sistema, la segunda que se está iniciando la conexión TCP, la tercera que se están generando variables de alto costo computacional y la cuarta que se están iniciando los módulos del sistema.

Figura 1.- Carga.

Pantalla de inicio:

Figura 2.- Inicio.

En esta pantalla se puede observar el reloj en la parte superior junto a dos flechas que permiten desplazarnos a las pantallas de medidas. Tras iniciar el sistema la fecha predeterminada son las 0:0:0 del día 1/1/2020. Más abajo podemos ver el botón de valores y el de ajustes que nos mandan a dichas pantallas.

Debajo de Nivel de brillo podemos seleccionar el nivel de brillo de la pantalla, del 1 al 4 en orden de brillo ascendente y automático a la derecha del todo. Presionar el botón de automático es sinónimo de pasar a modo de bajo consumo (LPM o LP). Si presionamos los botones HP automáticamente se selecciona el brillo 4, y seleccionando cualquier otro nivel de brillo se pasa al modo de alto consumo (HPM o HP). Si se presiona el botón de ULP se pasa al modo de muy bajo consumo, que, además de tener brillo automático, se apaga la pantalla si en un tiempo determinado y modificable no es utilizada por el usuario.

En esta pantalla se encuentra el botón de grabar y reproducir audio (Load en la esquina derecha y Play en la esquina izquierda respectivamente).

Figura 3.- Medidas (1).

Pantalla de medidas 1:

En esta pantalla podemos observar las medidas actuales, es decir, las últimas tomadas por la estación. En esta pantalla podemos observar las variables de altitud en metros, velocidad del viento en metros por segundo, humedad en porcentaje relativo, claridad en LUX e índice UV en UVs.

Además, podemos observar el reloj en la misma posición que la pantalla de inicio con las dos flechas de mover la pantalla, por lo que la interfaz superior sigue siendo la misma para cualquiera de las pantallas en la que nos encontremos.

Pantalla de medidas 2:

En esta pantalla podemos observar las medidas actuales, es decir, las últimas tomadas por la estación. En esta pantalla podemos observar las variables de presión y temperatura recibidas por el sensor BMP180, además de una interfaz gráfica que depende de los valores máximo y mínimo de la alarma de presión y temperatura. Esta interfaz consiste en una serie de barra de colores que cambia el color a medida que va habiendo más barras. Tener la barra vacía significa acercarse al valor mínimo mientras que tenerla llena significa acercarse al valor máximo. Además, se adjunta con el formato numérico.

Figura 4.- Medidas(2)

Figura 5.- Ajustes.

Pantalla de ajustes:

En esta pantalla se puede modificar fecha y la hora pulsando los botones de imncremento y decremento de la fila correspondiente (+ y - respectivamente). Además, podemos visualizar la IP que utiliza el servidor en el mismo momento que estamos visualizando el valor.

El valor por defecto de la dirección IP del servidor es 192.168.1.120, pero si es modificado, se mostrará el valor modificado para que el usuario sepa en todo momento la dirección a la que referirse.

Para retroceder a la pantalla de inicio basta ocn pulsar cualquier flecha de las situadas en las esquinas superiores.

Pantalla de valores:

En esta pantalla se pueden modificar los valores máximo y mínimo de la estación meteorlógica. Tanto de alarma de presión como de alarma de temperatura, valores máximos y mínimos. Pulsando las interfaces de incremento y decremento (+ y – respectivamente) podemos modifcar dichos valore. Al final de la pantalla podemos observar los límites mencionados.

Al lado de Pres (que indica presión) vemos dos números, el de la izquierda es el valor mínimo de alarma y el de la derecha el valor máximo.

Al lado de Temp (que indica temperatura) vemos dos números, el de la izquierda es el valor mínimo de alarma y el de la derecha el valor máximo.

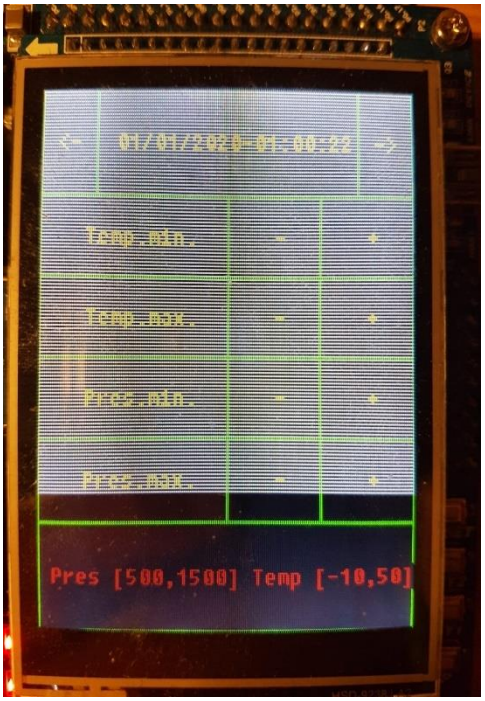


Figura 6.- Valores.

Además, si se toca la última línea que muestra la información de los valores mínimos y máximos, vemos que cambia de color, rojo indicando que se representa temperatura y verde que se representa presión, esto puede verse en el servomotor, que indica a su izquierda en su valor mínimo y a la derecha el máximo.

WEB.

La interfaz web consta de 3 partes, una tabla que se actualiza cada 20 segundos con los datos recibidos de la estación meteorológica, una segunda con la fecha de la medida y una tercera con los valores a modificar.

En la figura 7 podemos ver la interfaz WEB, en la última tabla pueden modificarse los siguientes valores:

- Valor mínimo de la alarma de temperatura. (°C)
- Valor máximo de la alarma de temperatura. (°C)
- Valor mínimo de la alarma de presión. (mBar.)
- Valor mínimo de la alarma de presión. (mBar.)
- Tiempo que tarda en apagarse la pantalla en ULPM (segundos).

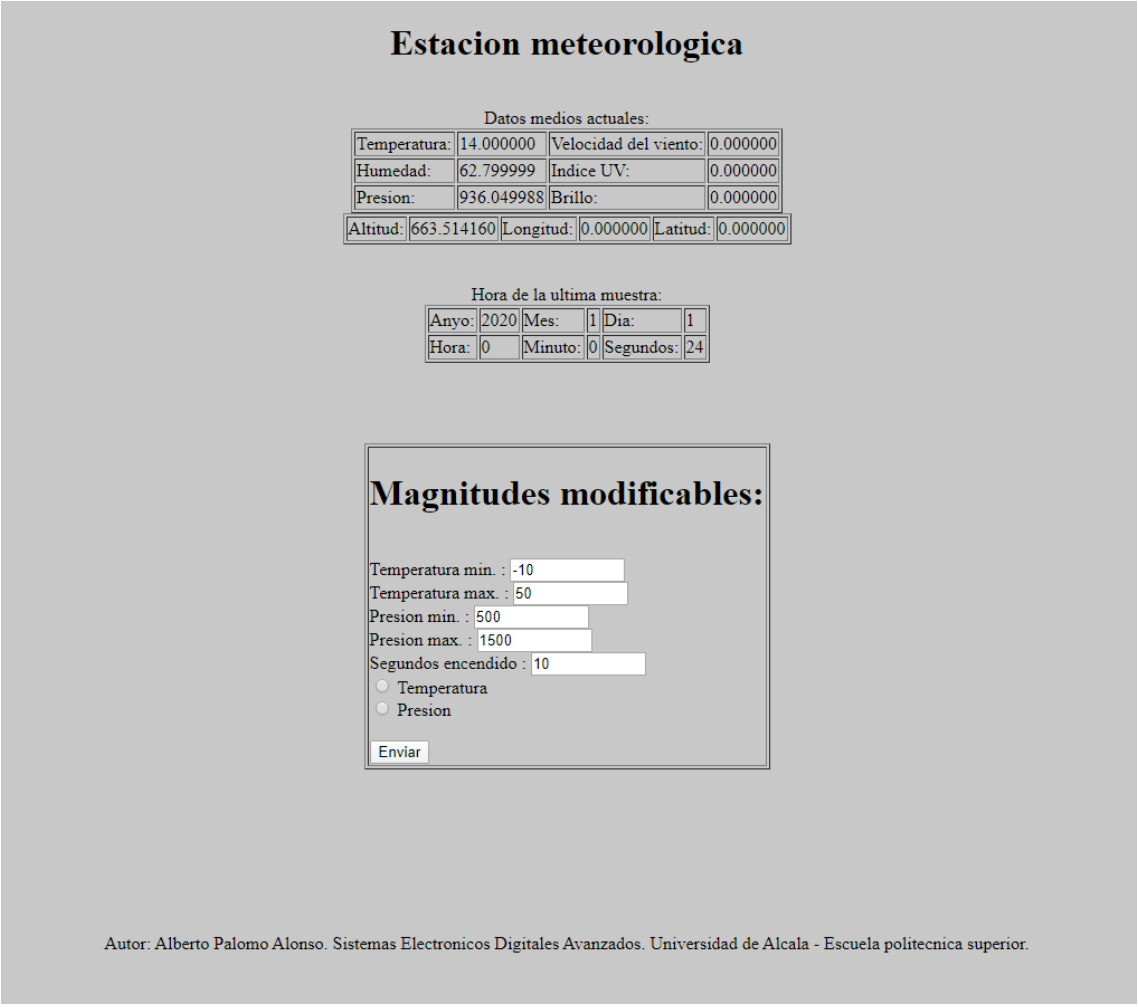


Figura 7.- Interfaz WEB.

Para configurar nuestro dispositivo que accede vía ethernet, debemos dirigirnos a nuestra configuración de IPv4. En Windows, eso es en la siguiente ruta:

- Configuración de Red e Internet/Cambiar opciones del Adaptador/Click detecho Ethernet/Priopiedades (root)/Protocolo de internet versión 4/Propiedades/Usar la siguiente dirección IP.

Por defecto está configurado que se utilice cualquier otra IP que no sea la del servidor, por ejemplo: 192.168.1.10.

Hay que configurar la máscara subred tal que sea: 255.255.255.0

La dirección de la puerta de enlace predeterminada o Gateway debe de ser: 192.168.1.20.

Estos valores pueden ser modificados dentro del código fuente del proyecto.

Para acceder a la web basta con conectar el cable ethernet al dispositivo y acceder a la dirección IP del servidor: 192.168.1.120 por defecto. Una vez entrando nos pedirán usuario y contraseña:

- Usuario: user
- Clave: Alver

Una vez introducido, deberíamos ser capaces de visualizar la interfaz WEB del servidor.

UART.

La interfaz UART es la interfaz USB del dispositivo – placa, con conectar un puerto USB mediante un cable USB-Micro USB al puerto de la tarjeta MINI-DK2 sería suficiente para iniciar la instalación automática del controlador.

Al realizar la instalación, se precisa de un monitor serie para podernos comunicar con un terminal serie, como puede ser Termite. La configuración del puerto serie debe de ser la siguiente:

- 9600 baudios.
- Paridad impar (odd).
- 8 bits por dato.
- 1 bit de stop.
- Append CR. (Sólo CR, el programa está hecho para interpretar sólo el carácter 13.)

Una vez realizada la configuración podemos comunicarnos con los siguientes comandos:

- GIVE [X]: Muestra por pantalla del dato [X] de la lista expuesta a continuación; sin los corchetes:

- IP
- TEMPERATURA
- PRESION
- VIENTO
- LUGAR
- INDICEUV
- HORA
- HUMEDAD
- BRILLO

  Cada uno haciendo referencia a la variable que indica en su propio nombre.

- SET [Y]: Configura la variable [Y] de la lista expuesta a continuación; sin los corchetes:

- BRILLO
- HORA
- MIN TEMP
- MAX TEMP
- MIN PRES
- MAX PRES
- TEMPERATURA
- PRESION

  Donde brillo es el tiempo que transcurre desde que no se toca la pantalla hasta que se apaga en modo ULPM, min representa mínimo y max representa máximo de los valores de alarma de TEMP, temperatura, y PRES, presión. TEMPERATURA activa la representación de la temperatura en el servomotor y PRESION la de presión en el servomotor.

- KILL: Cuelga el sistema en un while(1), activa el WatchDog.
- ABOUT: Muestra información del creador.
- HELP: Muestra información de la UART.
- HELP SET: Muestra la información del comando GIVE.
- HELP GIVE: Muestra la información del comando SET.

Referencias

https://github.com/iTzAlver/EstacionMeteorlogica.git

https://developer.arm.com/ip-products/processors/cortex-m/cortex-m3 [En línea] / aut. ARM.

https://www.amazon.es/Anem%C3%B3metro-Velocidad-Estaci%C3%B3n-Meteorol%C3%B3gica-Arduino/dp/B07BMVYBW9/ref=asc_df_B07BMVYBW9/?tag=googshopes-21&linkCode=df0&hvadid=300930138206&hvpos=1o1&hvnetw=g&hvrand=3634125468818365177&hvpone=&hvptwo=&hvqmt=&hvdev=c&h [En línea] / aut. anemómetro Imagen.

www.intel.com [En línea] / aut. Intel.

www.nxp.com [En línea] / aut. NXP.

www.powermcu.com [En línea] / aut. POWER MCU.

https://cloud.smartdraw.com/

www.blackboard.com

https://uah.blackboard.com/ultra/courses/_17817_1/cl/outline

https://github.com/OCFreaks/LPC1768-Tutorial-Examples/blob/master/DHT11_Interfacing/main.c

Anexos.

A continuación, los anexos de la memoria de la estación meteorológica.

Anexo I – Esquemático.

En este anexo se adjunta un pequeño esquemático con la distribución de pines.