Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Sistema sensor autònom sense fils de baix consum per a aplicacions industrials

TITULACIÓN: Grau en Enginyeria de Sistemes de Telecomunicació

AUTOR: Alberto Gómez Blázquez

DIRECTOR: Francesc Josep Robert Sanxis

FECHA: 13 de Septiembre del 2016

Título: Sistema sensor autònom sense fils de baix consum per a aplicacions industrials

Autor: Alberto Gómez Blázquez

Director: Francesc Josep Robert Sanxis

Fecha: 13 de Septiembre del 2016

Resumen

El objetivo de este proyecto es el diseño y desarrollo de una estación meteorológica inalámbrica y de bajo consumo para su instalación en un puerto deportivo.

Para ello aprovecharemos las características de los últimos microcontroladores de Microchip que disponen de la tecnología XLP (Extreme Low Power) que reduce el consumo del sistema cuando este entra en modo Sleep.

El lenguaje de programación será C y utilizando el estilo de programación FSM (Finite State Machine) que hace el código más eficiente, más fácil de depurar y ayuda a la organización del flujo de programación.

Se han escogido 5 sensores para la medición de los parámetros más comunes en una estación meteorológica: DHT22 (humedad y temperatura), BMP180 (presión atmosférica y temperatura), DS18B20 (temperatura del agua), anemómetro (velocidad del viento) y veleta (dirección del viento).

Para la alimentación del sistema utilizaremos una batería que se recargará mediante una placa solar.

La comunicación entre la estación meteorológica y el PC será mediante tecnología Bluetooth y la interfaz gráfica de recepción de los datos está creada con LabView.

El proyecto se realizará en pequeñas fases completamente funcionales, una por cada sensor, realizando simulaciones con la herramienta Proteus y pruebas reales de los sensores con la placa de entrenamiento DM240313 equipada con el microcontrolador PIC18F87K22.

Una vez finalizadas todas las fases, se unificarán en un proyecto común añadiendo una conexión inalámbrica, implementando el modo de bajo consumo del microcontrolador y creando una interfaz gráfica de monitorización con LabView.

Title: Wireless autonomous sensor system low-power for industrial applications

Author: Alberto Gómez Blázquez

Director: Francesc Josep Robert Sanxis

Date: September, 13th 2016

Overview

The objective off this job is the design and development of a wireless low power weather station for installation in a marina.

To do this, we will use the features of the latest microcontrollers from Microchip that have the XLP (Extreme Low Power) technology that reduces power consumption when the system goes into Sleep mode.

The programming language will be C and using the programming style FSM (Finite State Machine) that makes the most efficient, easier to debug and helps the organization flow programming code.

We chosen 5 sensor for the measurement of the parameters most common in a weather station: DHT22 (humidity and temperature) , BMP180 (pressure and temperature) , DS18B20 (water temperature) , anemometer (wind speed) and wind vane (direction of the wind) .

For system power supply we will use a one battery recharged by a solar panel.

The communication between the weather station and the PC will be via bluetooth technology and the graphical interface data reception is created with LabView.

The project will be implemented in small phases fully functional, one for each sensor, performing simulations with Proteus tool and real tests of the sensors with the development board DM240313 equipped with PIC18F87K22 microcontroller.

Upon completion of all phases, they will be unified in a common project by adding a wireless connection, implementing the low power mode of the microcontroller and creating a graphical monitoring interface with LabView.

ÍNDICE DE ACRÓNIMOS

ADC	Analog to Digital Conversor
COM	Communication
E2PROM	Electrically Erasable Programmable Read-Only Memory
FSM	Finite State Machine
I2C	Inter-Integrated Circuit
LCD	Liquid Cristal Display
PC	Personal Computer
PCB	Printed Circuit Board
ROM	Read Only Memory
RS-232	Recommended Standard-232
SCL	Serial Clock
SDA	Serial Data
TTL	Transistor-Transistor Logic
USART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
WDT	Watch Dog Timer
XLP	Extreme Low Power

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El principal objetivo de este trabajo, es la realización de un sistema autónomo e inalámbrico capaz de capturar, procesar y enviar la información de varios sensores a una interfaz de monitorización para su posterior análisis o estudio.

Uno de los principales problemas que nos encontramos en este tipo de sensores autónomos, es la alimentación y consumo energético de estos. Para ello, se ha realizado una primera aproximación a las nuevas técnicas de reducción de consumo energético de la última generación de microprocesadores y de alimentación mediante procesos de Energy Harvesting.

Se ha escogido desarrollar una estación meteorológica, ya que reúne todos los requisitos planteados en este tipo de sistemas.

La realización del trabajo se ha dividido en cuatro capítulos. En el primer capítulo se hace una introducción sobre las estaciones meteorológicas comerciales y las características de estas, además de un resumen de los principales fenómenos meteorológicos y la importancia que tiene el estudio y análisis de su comportamiento. También se hace un resumen de las especificaciones del proyecto así como de la metodología de trabajo que se ha seguido.

El segundo capítulo contiene la planificación y el desarrollo del proyecto por fases. Se realizan pequeños proyectos completamente funcionales con cada uno de los sensores escogidos para la estación meteorológica. Este método de trabajo nos ayuda a estructurar nuestro trabajo de una forma eficiente de tal forma que sea modulable y fácilmente modificable.

En el tercer capítulo, se realiza la integración de todas las fases anteriores y se añade la comunicación sin hilos y los modos de bajo consumo del microprocesador, además de la realización de una interfaz de monitorización creada con LabView y la creación de un prototipo PCB de la estación meteorológica. También se realizan las pruebas de consumo del sistema y se comprueba que los modos de bajo consumo del microprocesador ayudan a reducir el consumo energético.

Finalmente, en el capitulo cuatro se realizan las conclusiones del trabajo y se plantean diversas mejoras o modificaciones que podrían ayudar a reducir aún más el consumo energético o mejorar algunas características de la estación meteorológica.

CAPÍTULO 1. ESPECIFICACIONES

1.1. Estaciones meteorológicas comerciales

Para tener una pequeña idea de los fenómenos meteorológicos a medir, se han estudiado algunas de las estaciones meteorológicas comerciales disponibles.

Observando las que hay disponibles en el mercado podremos incorporar mejoras o eliminar elementos que no nos interesen.

1.1.1 Estación meteorológica PCE-FWS20

Fig. 1.1 Estación meteorológica PCE-FWS20

Se trata de una estación meteorológica semiprofesional. Nos permite detectar de forma precisa la dirección del viento, la velocidad del viento, la temperatura, la presión atmosférica, la humedad relativa y la pluviosidad. Además dispone de diferentes funciones de alarma (velocidad del aire, presión atmosférica, etc.). Se trata de una estación inalámbrica vía radio con un alcance de unos 100 metros de distancia. El transmisor está alimentado por un módulo solar y dos pilas recargables. Incluye un software de análisis donde se pueden guardar y analizar los valores meteorológicos de forma ilimitada.

Funciones de la estación base:

- Temperatura interior y exterior en grados Celsius o Fahrenheit
- Humedad relativa interior y exterior
- Indicación de la presión atmosférica en inHG o hPa
- Indicación de la pluviometría en mm o inch
- velocidad del viento en mph, km/h, m/s, nudos o Beaufort
- Indicación de la dirección del viento
- Indicador de temperatura Wind Chill (sensación térmica)
- Indicación del punto de rocío
- Previsión del tiempo
- Aviso de tormenta
- Alarma programables para diferentes valores meteorológicos

- Memoriza los valores máximo y mínimo recibidos
- Pantalla LED con iluminación de fondo
- Indicación en 12 o 24 horas
- Calendario
- Ajuste de la franja horaria
- Función de ahorro energético
- Intervalo de medición cada 48 segundos

En la Tabla 1.1 podemos observar los rangos de medición y resolución de la estación meteorológica PCE-FWS20:

Tabla 1.1 Rangos de medición y resolución de la estación PCE-FWS20

	Interior	Exterior	Resolución
Temperatura	De 0ºC a 60ºC	De – 40ºC a 65ºC	0.1ºC
Humedad	De 1% a 99%	De 1% a 99%	1%
Presión atmosférica	De 700 hPa a 1100 hPa	----	0.1 hPa
Velocidad del viento	----	De 0 a 240 Km/h	0.36 Km/h
Pluviometría	----	De 0 a 9999mm	0.1 mm

Precio: 105€

1.1.2 Estación meteorológica WMR300

Fig. 1.2 Estación meteorológica WMR300

Se trata de una estación meteorológica de alta precisión para uso industrial. Cuenta con sensores para medir la velocidad y dirección del viento, la temperatura, la humedad relativa, la presión atmosférica y la pluviosidad. Dispone de un transmisor inalámbrico capaz de transmitir a una distancia de hasta 300 metros. Los sensores están alimentados por una batería conectada a un panel solar lo que permite un gran ahorro de energía. Incluye una pantalla táctil con conexión USB que permite ver, analizar, almacenar y compartir la información meteorológica a través del PC.

Funciones de la estación base:

- Estación meteorológica de calidad profesional y sistema de sensores con certificación NIST
- Pantalla táctil LCD retroiluminada
- Información climatológica detallada que incluye predicción meteorológica, temperatura interior y exterior, humedad, presión barométrica, dirección y velocidad del viento, precipitaciones, sensación térmica y punto de condensación
- Permite grabar y almacenar información meteorológica en intervalos de 1 a 60 minutos.
- Mide la tasa y niveles de lluvia, lluvia acumulada, lluvia caída en las últimas 24 horas en mm y pulgadas
- velocidad del viento en m/s, km/h, mph, nudos
- Indicación de la dirección del viento
- Temperatura interior y exterior en grados Celsius o Fahrenheit
- Humedad relativa interior y exterior
- Hora de salida y puesta del sol, así como la fase de la luna con reloj controlado por radio

En la Tabla 1.2 podemos observar los rangos de medición y resolución de la estación meteorológica WMR300:

Tabla 1.2 Rangos de medición y resolución de la estación WMR300

	Interior	Exterior	Resolución
Temperatura	De 0ºC a 60ºC	De – 40ºC a 65ºC	0.1ºC
Humedad	De 0% a 99%	De 0% a 99%	1%
Presión atmosférica	De 540 hPa a 1100 hPa	----	0.1 hPa
Velocidad del viento	----	De 0 a 288 Km/h	0.36 Km/h
Pluviometría	----	De 0 a 5080 mm	0.254mm

Precio: 679 €

1.2. Fenómenos meteorológicos a medir

1.2.1 Temperatura ambiente

La temperatura es una magnitud física que refleja la cantidad de calor, ya sea de un cuerpo, de un objeto o del ambiente.

Está relacionada con la energía interior de los sistemas termodinámicos, de acuerdo al movimiento de sus partículas, y cuantifica la actividad de las moléculas en la materia: a mayor energía sensible, más temperatura.

La temperatura ambiente varía según el lugar, por lo general se mide con un termómetro expuesto al aire en un lugar protegido de la radiación solar directa.

1.2.2 Humedad relativa

La humedad relativa es la cantidad de humedad en el aire comparada con la cantidad de humedad que puede retener a una temperatura.

$$Humedad\ relativa\ (RH) = \frac{gramos\ de\ vapor\ de\ agua\ presente\ (g/m^3)}{gramos\ de\ vapor\ de\ agua\ en\ saturación\ (g/m^3)}\ x\ 100$$

Fig. 1.3 Cálculo de la humedad relativa

El vapor de agua se forma a causa de la evaporación del agua presente en la naturaleza. El vapor de agua producido es absorbido por el aire en cantidades que dependen de las condiciones ambientales, provocando un aumento del contenido de la humedad. La máxima cantidad de vapor que el aire puede absorber la denominamos “cantidad de saturación” y aumenta en función de la temperatura a un mismo volumen. Si la cantidad de vapor de agua contenida en un volumen de aire saturado con una determinada temperatura aumenta, el vapor condensa pasando al estado líquido, es lo que denominamos “punto de rocío”.

1.2.2 Presión atmosférica

Se conoce como presión atmosférica a la presión que ejerce el aire en cualquier punto de la atmósfera. La presión atmosférica varía por la acción de factores como la temperatura, la altitud o la humedad.

- Altura: La presión atmosférica se debe al peso del aire sobre un punto de la superficie terrestre, por lo tanto mayor altura la cantidad de aire en menor y la presión disminuye y a menor altura aumenta.

- Temperatura: La temperatura también hace variar la presión ya que el aire caliente pesa menos que el frío y tiende a elevarse y origina bajas presiones, en cambio cuando el aire se enfría tiende a bajar y ejercer más presión.

La lluvia está relacionada directamente con la presión atmosférica. Un anticiclón es una zona de alta presión, en la cual la presión atmosférica es superior a la del aire de los alrededores. El aire que se encuentra en un anticiclón desciende desde las capas altas de la atmósfera hacia la superficie terrestre. Este fenómeno que recibe el nombre de subsidencia, hace difícil la formación de nubes provocando la situación de tiempo estable y ausencia de precipitaciones. En cambio una borrasca es una región donde la presión atmosférica es más baja que la del aire circundante, el aire caliente y húmedo al ser más ligero, asciende, y al elevarse este aire se enfría lo que puede provocar la lluvia.

1.2.3 Velocidad del viento

El viento es aire en movimiento que se desplaza de forma horizontal a lo largo de la superficie terrestre por las diferencias de presión y temperatura entre distintas zonas. La circulación se produce de zonas de alta presión a zonas de baja presión intentando igualar ambas. Su fuerza o velocidad viene determinada cuanto mayor es el gradiente entre las presiones. Este trasvase de energía lo denominamos convección.

A causa de la rotación de la tierra, la circulación de los vientos no es la misma en todos los sitios. En el hemisferio norte los vientos se mueven en el sentido de las agujas del reloj en áreas de altas presiones (anticiclón) y en sentido en contrario en áreas de bajas presiones (borrascas o depresión). En el hemisferio sur los vientos se mueven en sentido contrario.

Para expresar los valores del viento en el medio marino, se utiliza la escala Beaufort (Véase Tabla 1.3). La escala Beaufort es una medida empírica basada en el estado de la mar, de sus olas y de la fuerza del viento.

La escala fue creada por Sir Francis Beaufort (oficial naval e hidrógrafo) en el año 1805. Antes de esa fecha los oficiales navales hacían observaciones regulares del tiempo siendo estas muy subjetivas al carecer de una escala.

La escala fue adaptada para uso terrestre a partir del año 1850, asociando los números Beaufort con el número de rotaciones de un anemómetro para medir la velocidad del viento.

Tabla 1.3 Escala Beaufort de viento

Número de Beaufort	Velocidad del viento (Km/h)	Velocidad del viento en Nudos	Denominación	Aspecto del mar	Efectos en tierra
0	0 a 1	<1	Calma	Despejado	Calma, el humo asciende verticalmente
1	2 a 5	1 a 3	Ventolina	Pequeñas olas, pero sin espuma	El humo indica la dirección del viento
2	6 a 11	4 a 6	Flojito (Brisa muy débil)	Crestas de apariencia vítrea, sin romper	Se caen las hojas de los árboles, empiezan a moverse los molinos de

					los campos
3	12 a 19	7 a 10	Flojo (Brisa ligera)	Pequeñas olas, crestas rompientes	Se agitan las hojas, ondulan las banderas
4	20 a 28	11 a 16	Bonancible (Brisa moderada)	Borreguillos numerosos, olas cada vez más largas	Se levanta polvo y papeles, se agitan las copas de los árboles
5	29 a 38	17 a 21	Fresquito (Brisa fresca)	Olas medianas y alargadas, borreguillos muy abundantes	Pequeños movimientos de los árboles, superficie de los lagos ondulada
6	39 a 49	22 a 27	Fresco (Brisa fuerte)	Comienzan a formarse olas grandes, crestas rompientes, espuma	Se mueven las ramas de los árboles, dificultad para mantener abierto el paraguas
7	50 a 61	28 a 33	Frescachón (Viento fuerte)	Mar gruesa, con espuma arrastrada en dirección del viento	Se mueven los árboles grandes, dificultad para caminar contra el viento
8	62 a 74	34 a 40	Temporal (Viento duro)	Grandes olas rompientes, franjas de espuma	Se quiebran las copas de los árboles, circulación de personas muy dificultosa
9	75 a 88	41 a 47	Temporal fuerte (Muy duro)	Olas muy grandes, rompientes. Visibilidad	Daños en árboles, imposible andar

				mermada	contra el viento
10	89 a 102	48 a 55	Temporal duro (Temporal)	Olas muy gruesas con crestas empenachadas. Superficie del mar blanca	Árboles arrancados, daños en la estructura de las construcciones
11	103 a 117	56 a 63	Temporal muy duro (Borrasca)	Olas excepcionalmente grandes, mar completamente blanca, visibilidad muy reducida	Destrucción en todas partes, lluvias muy intensas, inundaciones muy altas
12	+ 118	+64	Temporal huracanado (Huracán)	Olas excepcionalmente grandes, mar blanca, visibilidad nula	Voladura de autos, árboles, casas, techos y personas. Puede generar un huracán o un tifón

1.2.5 Dirección del viento

La dirección del viento depende directamente de la distribución de las presiones, ya que el movimiento de este se produce desde las zonas de altas presiones a zonas de bajas presiones.

Denominamos dirección del viento al punto en el horizonte de donde sopla. Para distinguirlos se le aplica el nombre de los principales rumbos de la brújula. Los cuatro puntos principales corresponden a los cardinales: Norte (N), Sur (S), Este (E) y Oeste (W) pudiéndose considerar hasta 32 puntos entre estos y los intermedios. Se suelen representar en la conocida rosa de los vientos (Véase Tabla 1.4) donde cada punto cardinal está asociado con su equivalencia en grados azimut.

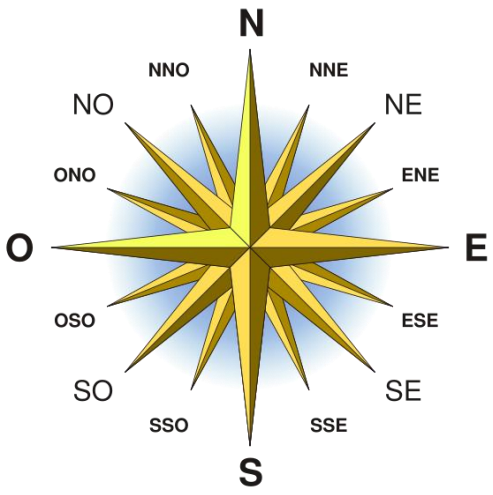


Fig. 1.4 Rosa de los vientos

En la Tabla 1.4 se muestran los puntos cardinales más utilizados y su dirección en grados azimut:

Tabla 1.4 Dirección, nombre y grados azimut del viento

Dirección	Nombre	Grados Azimut
N	Norte	0°
NNE	Nornoreste	22.5°
NE	Noreste	45°
ENE	Estenoreste	67.5°
E	Este	90°
ESE	Estesureste	112.5°
SE	Sureste	135°
SSE	Sursureste	157°
S	Sur	180°
SSW	Sursuroeste	202.5°
SW	Suroeste	225°
WSW	Oestesuroeste	247.5°
W	Oeste	270°
WNW	Oestenoroeste	292.5°
NW	Noroeste	315°
NNW	Nornoroeste	337.5°

1.2.6 Temperatura del agua

El control de la temperatura del agua del mar es un factor determinante para la detección de la formación de huracanes. La elevación de la temperatura del agua está directamente relacionada con la intensidad de estos.

Las temperaturas en la superficie por encima del los 26°C pueden provocar la formación de ciclones tropicales o huracanes. Cuando la temperatura de la superficie del agua aumenta, el agua se evapora con mayor facilidad, lo que contribuye a que las tormentas que se forman en el océano se conviertan en sistemas de mayor tamaño e intensidad.

El aumento de la temperatura del mar se asocia también a la proliferación de especies invasoras o bacterias pudiendo desplazar o eliminar del ecosistema las especies autóctonas.

Otro factor a tener en cuenta es que el aumento de la temperatura puede provocar el deshielo polar, produciendo un aumento en el nivel del mar.

1.3. Especificaciones del proyecto

1.3.1. Medidas y visualización de datos meteorológicos

Una vez estudiadas las diferentes estaciones meteorológicas comerciales disponibles en el mercado, el tipo de fenómenos meteorológicos que miden, la resolución, los rangos de estos y las características de transmisión de datos y alimentación, podemos hacer un resumen de las características que queremos obtener en nuestra estación meteorológica.

- Temperatura exterior e interior con escala de -40º a 50ºC con resolución de 0.1ºC
- Temperatura del agua de -40ºC a 50ºC con resolución de 0.1ºC
- Humedad relativa de 0% a 100% con resolución de 0.1%
- Presión atmosférica con una escala de 960 hPa a 1070 hPa con una resolución de 0.01 hPa
- Velocidad del aire de 0Km/h a 120 Km/h con resolución de 1 Km/h
- Dirección del aire de hasta 16 posiciones
- Grabación de datos meteorológicos para su posterior análisis

1.3.2. Organización y ejecución por fases de proyectos completos

La realización del proyecto se realiza por pequeñas fases funcionales. Se pretende así realizar un proyecto modulable y ampliable en el que realizar modificaciones o ampliaciones sea sencillo de implementar.

1.3.3. Uso del simulador Proteus

Para ayudar en el diseño y desarrollo de estos pequeños módulos utilizaremos el simulador virtual ISIS Proteus 8, que nos permite depurar nuestro código, encontrar errores más rápidamente, realizar modificaciones en el cableado más

fácilmente o añadir dispositivos para comprobar su funcionamiento sin tener que montarlos físicamente o disponer de ellos.

1.3.4. Uso del estilo de programación FSM

Para la programación del código, utilizaremos el estilo de programación FSM o Maquina de estado (Véase Fig. 1.5). Este estilo, tiene una serie de ventajas a la hora de programar microcontroladores con respecto a otros estilos más clásicos.

- Hacen el código más eficiente, más fácil de depurar y ayudan a la organización del flujo de programación.
- Está basado en que hay un número finitos de estados para un sistema determinado.
- Requieren una variable de estado que puede modificarse por interrupciones externas o internas, cambios de variables, finalización de contadores y que mantienen un control del estado en el que se encuentra en microcontrolador y dirige el flujo de programa.

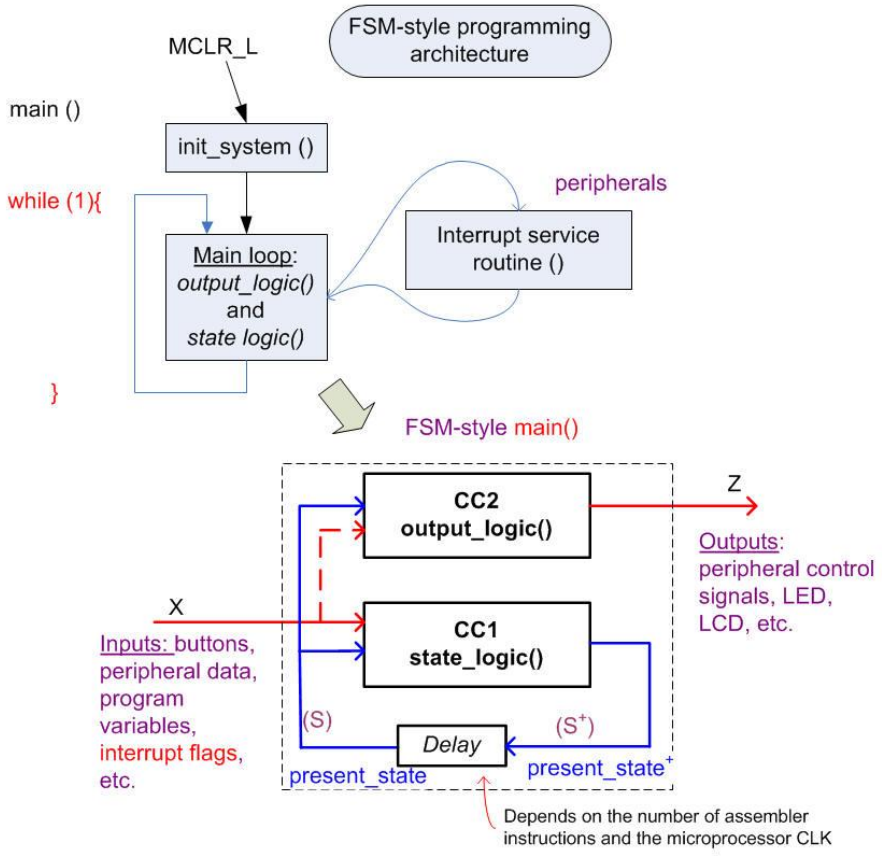


Fig. 1.5 Estilo de programación FSM

La arquitectura FSM se divide en dos principales bloques:

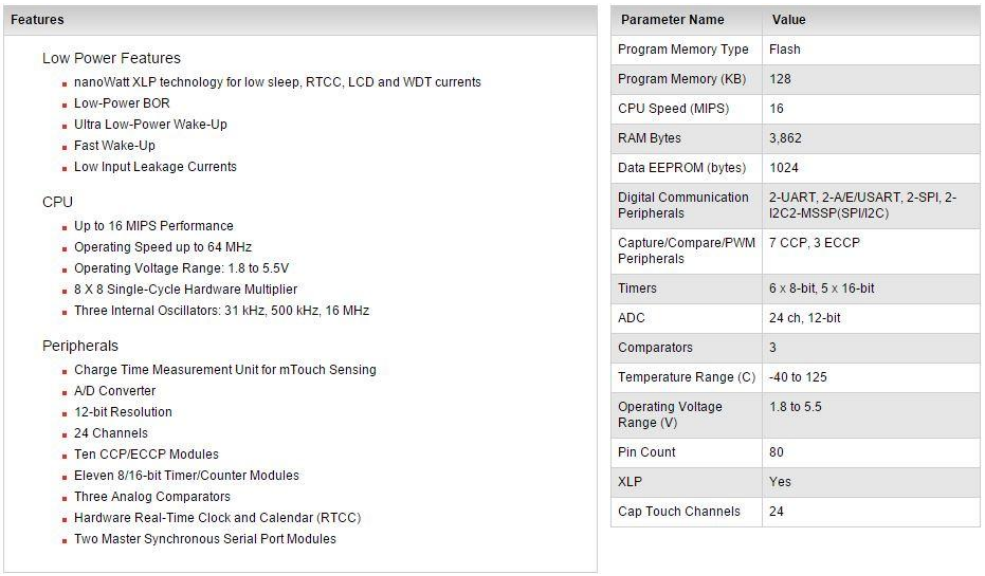
- Init_system (): Se trata de la configuración inicial del sistema, se ejecuta una única vez al inicio de este y solo se vuelve a ejecutar si se produce un Reset del sistema. En ella se configuran los bits de control del microcontrolador, las interrupciones, los pines I/O, la velocidad del oscilador, las condiciones iniciales de las variables o la configuración de los periféricos del microcontrolador.

- Main_loop: Se trata de la ejecución principal del sistema. En ella encontramos las funciones state_logic () que controla el estado futuro del sistema mediante interrupciones o cambios de variables y la función output_logic () donde se encuentran las rutinas principales de los estado. También encontramos la función Interrupt_service_routine () donde se gestionan la interrupciones de los periféricos del microcontrolador como Timers, ADC o USART.

1.3.5. Selección del PIC18F87K22 y la placa de entrenamiento DM240313

El microcontrolador que utilizaremos en el desarrollo del proyecto se trata del PIC18F87K22. Este dispone de la última tecnología de Microchip nanoWatt XLP adecuado para el diseño de aplicaciones de baja potencia que funcionen con baterías donde el consumo reducido de energía y la máxima eficiencia energética son los principales objetivos.

En la figura Fig. 1.6 se pueden ver las principales características del microcontrolador PIC18F87K22.



Features

Low Power Features
- nanoWatt XLP technology for low sleep, RTCC, LCD and WDT currents
- Low-Power BOR
- Ultra Low-Power Wake-Up
- Fast Wake-Up
- Low Input Leakage Currents

CPU
- Up to 16 MIPS Performance
- Operating Speed up to 64 MHz
- Operating Voltage Range: 1.8 to 5.5V
- 8 X 8 Single-Cycle Hardware Multiplier
- Three Internal Oscillators: 31 kHz, 500 kHz, 16 MHz

Peripherals
- Charge Time Measurement Unit for mTouch Sensing
- A/D Converter
- 12-bit Resolution
- 24 Channels
- Ten CCP/ECCP Modules
- Eleven 8/16-bit Timer/Counter Modules
- Three Analog Comparators
- Hardware Real-Time Clock and Calendar (RTCC)
- Two Master Synchronous Serial Port Modules

Parameter Name	Value
Program Memory Type	Flash
Program Memory (KB)	128
CPU Speed (MIPS)	16
RAM Bytes	3,862
Data EEPROM (bytes)	1024
Digital Communication Peripherals	2-UART, 2-A/E/USART, 2-SPI, 2-I2C2-MSSP(SPI/I2C)
Capture/Compare/PWM Peripherals	7 CCP, 3 ECCP
Timers	6 x 8-bit, 5 x 16-bit
ADC	24 ch, 12-bit
Comparators	3
Temperature Range (C)	-40 to 125
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	80
XLP	Yes
Cap Touch Channels	24

Fig. 1.6 Características del microcontrolador PIC18F87K22

Para comprobar el funcionamiento de los sensores y la conexión inalámbrica utilizaremos la placa de entrenamiento DM240313 (Véase Fig. 1.7). Se trata de una plataforma de desarrollo que nos permitirá comprobar el consumo de energía del sistema en los diferentes modos de funcionamiento disponibles del microcontrolador (Idle, sleep, deep sleep).

La placa incorpora diferentes dispositivos (LCD, sensor de temperatura, potenciómetro, LEDs, etc) para la creación de prototipos de aplicaciones de bajo consumo.

Dispone de puertos de expansión Pictail y serial para añadir dispositivos externos con facilidad además de una pequeña protoboard.

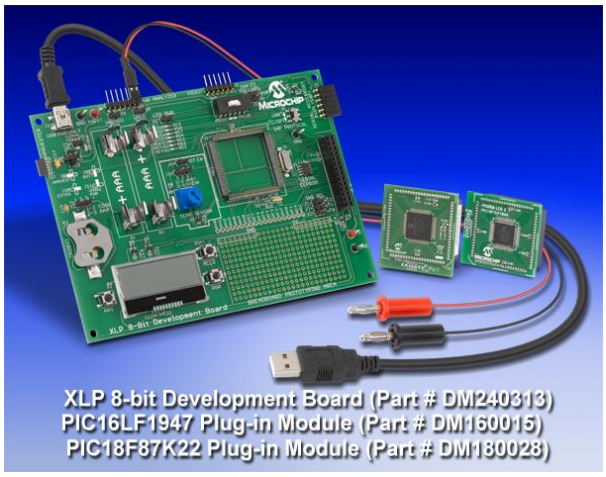


Fig. 1.7 Placa de entrenamiento Microchip DM240313

Características de la placa de entrenamiento DM240313:

- Osciladores externos de 10 MHz y 32.768 kHz
- Diferentes fuentes de alimentación seleccionables (USB, AAA, CR2032, Harvester, Externa)
- Puerto comunicaciones SAP (SPI, USART, I2C)
- Tres botones (RB0, RB1, MCLR)
- 7 LEDs
- LCD SPI 16x2
- Serial EEPROM 24AA256 256 KB/25AA256 256 KB
- Sensor de temperatura MCP9700 (-40ºC a 125ºC)
- Potenciómetro 100KΩ conectado a la entrada AN3
- Protoboard 24x11
- Power Analizer para medir el consumo de la placa o el microcontrolador
- Puerto de expansión Pictail
- Interfaz de programación PicKit
- Conexión a pins I/O del microcontrolador

1.3.6. Experimentación con un sistema de transmisión sin hilos

Una de las principales características del proyecto es la comunicación inalámbrica de la estación meteorológica con el PC. Para llevar a cabo esta comunicación se ha barajado varios de los sistemas disponibles en el mercado. Se ha escogido finalmente una comunicación por bluetooth por la simplicidad a la hora de implementar el hardware, la compatibilidad de los dispositivos y el precio de los componentes.

1.3.7. Estudio de los modos de bajo consumo del microcontrolador

Al ser un sistema autónomo, del que una vez puesto en marcha no se requiera un alto grado de mantenimiento, el modo de bajo consumo del microcontrolador nos será de gran ayuda para conseguir reducir al máximo el consumo del sistema.

El microcontrolador dispone de varios modos de funcionamiento (Run, Idle, Sleep). Se estudiará el más adecuado para cada situación y se realizarán mediciones de consumo para comprobar que realmente utilizándolos se consigue un ahorro de energía.

1.3.8. Organización de los materiales en un ePortfolio

Para que la información sobre el proyecto quede guardada y ordenada para futuras consultas, se ha creado un ePortfolio donde quedan recogidos todos los proyectos así como todos los datos necesarios para poder implementarlos.

En la página https://sites.google.com/site/tfgalbertogomez/ se pueden consultar las características del proyecto, así como descargar los archivos necesarios para su estudio o consulta.

1.4. Diagrama general del sistema

En la Fig. 1.8 se puede ver el diagrama general del sistema. Está formado por un sistema de alimentación compuesto de una placa solar de 5.5V que alimenta a una batería de NiMH de 3.7V que es la que suministra la energía necesaria al sistema mediante un regulador de tensión de 3.3V, un sistema de sensores que capta los diferentes factores a medir, un microprocesador a una velocidad de 4Mhz que procesa los datos obtenidos por los sensores y un módulo bluetooth que envía los datos inalámbricamente a una interfaz de control y monitorización realizada con LabVIEW.

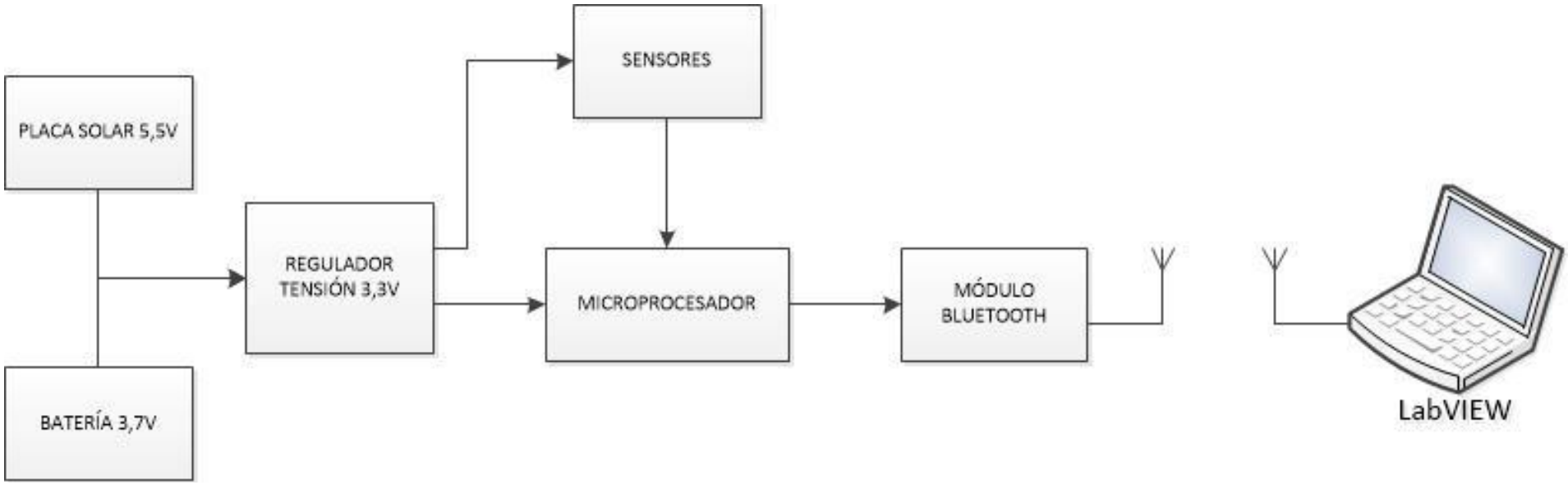


Fig. 1.8 Diagrama completo del sistema

CAPÍTULO 2. PLANIFICACIÓN Y DESARROLLO DEL PROYECTO POR FASES

2.1. Medida de la humedad relativa y temperatura ambiente

En esta primera fase se realizará la captación de la humedad y temperatura mediante el sensor DHT22. Nos servirá como una estructura general que utilizaremos con las siguientes fases para así ir implementando una a una hasta unificarlas todas en el mismo proyecto.

2.1.1. Funcionamiento básico del sensor DHT22

El sensor DHT22 (Véase Fig. 2.1), nos permite la medida de temperatura y humedad relativa (RH). Para comunicarse con el microcontrolador utiliza un protocolo de comunicación propietario muy similar al conocido 1-Wire de Dallas semiconductor y solo requiere un pin de datos para la comunicación.

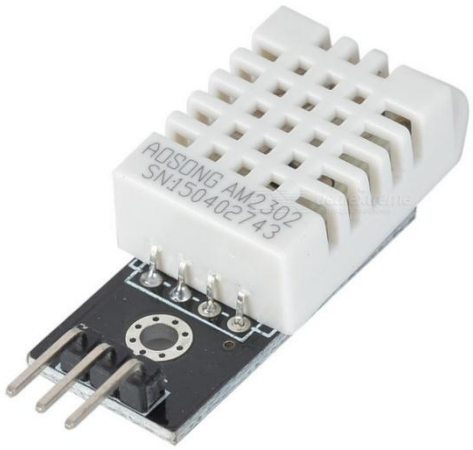


Fig. 2.1 Sensor de humedad y temperatura DHT22

El sensor está compuesto por dos partes, un sensor de humedad capacitivo de polímero y un termistor para medir la temperatura. El sensor también dispone de un circuito integrado que hace la conversión analógica-digital y se encarga de enviar los datos digitales por el bus de comunicación.

Características del sensor DHT22:

- Alimentación de 3.3V a 5.5V DC
- Corriente máxima durante conversión 2.5mA

- Rango de medición de temperatura -40ºC a 125ºC
- Precisión temperatura ±0.5ºC a ±1ºC en condiciones adversas
- Resolución de temperatura 0.1ºC
- Rango de medición de humedad 0% RH a 100% RH
- Precisión humedad relativa ±2% a 25ºC
- Resolución de humedad 0.1% RH
- Velocidad de muestro 0.5Hz

Protocolo de comunicación del DHT22:

El protocolo de comunicación del sensor DHT22 es un protocolo propietario similar 1-wire, aunque en el caso del DHT22 cada sensor tiene que disponer de su propio pin de datos no pudiendo compartir el mismo bus más de un sensor.

El microcontrolador empieza la comunicación configurando el pin como salida y poniendo el nivel en alto (Vcc), después envía la señal de 'Start' estableciendo el nivel bajo durante al menos 18ms y nuevamente a nivel alto entre 20us-40us. Seguidamente se pone el pin como entrada y esperamos la respuesta del sensor que establecerá un nivel bajo durante 80us y un nivel alto durante otros 80us. Ahora el sensor ya está preparado para enviar la información de humedad y temperatura. El sensor enviará 5 bytes de información (40 bits) seguidos. Una vez termina la transmisión de los 40 bits el sensor pone el bus a nivel bajo durante 50us y luego a nivel alto para liberarlo y entrar en modo de bajo consumo hasta que en el microcontrolador envíe de nuevo la señal de 'Start'.

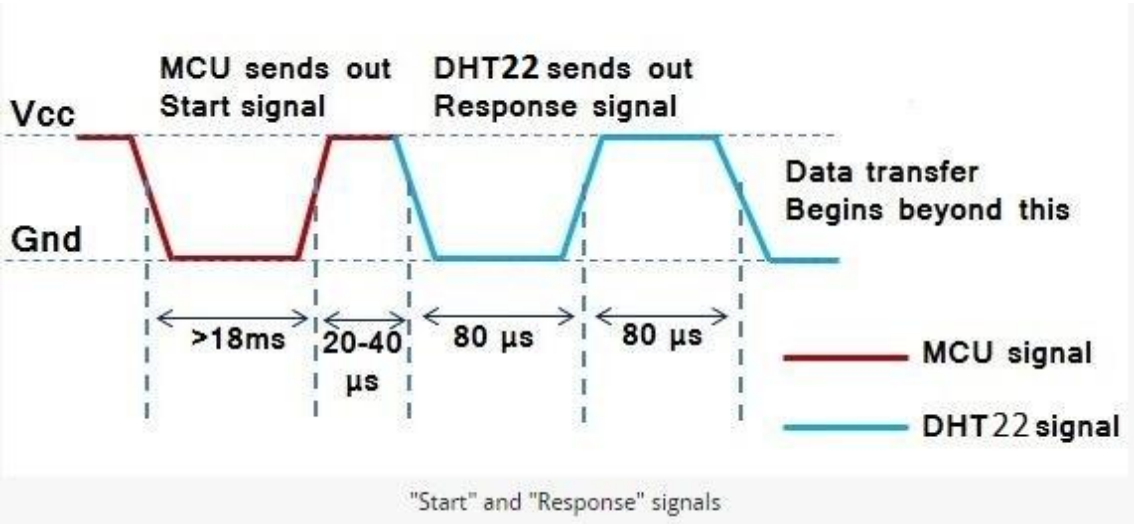


Fig. 2.2 Señal 'Start' y 'Response' del sensor DHT22

La información de temperatura y humedad está estructurada de la siguiente manera en los 40 bits:

- Byte 1: Parte entera de humedad relativa
- Byte 2: Parte decimal de la humedad relativa
- Byte 3: Parte entera de la temperatura
- Byte 4: Parte decimal de la temperatura
- Byte 5: Byte de paridad o checksum

El checksum se utiliza para comprobar que los bits recibidos no contienen errores de transmisión. Se calcula sumando los 4 bytes de humedad y temperatura y el resultado de esta suma tiene que ser igual al byte de checksum para que la transmisión de datos sea correcta.

El sensor puede medir temperaturas negativas. Para expresar una temperatura negativa utiliza el bit más significativo de la parte entera enviando un '1' y envía un '0' si quiere expresar números positivos.

El protocolo para enviar los bits por parte del sensor es la siguiente:

- Formato Bit '0': 50us en nivel bajo y 26us en nivel alto
- Formato Bit '1': 50us en nivel bajo y 70us en nivel alto

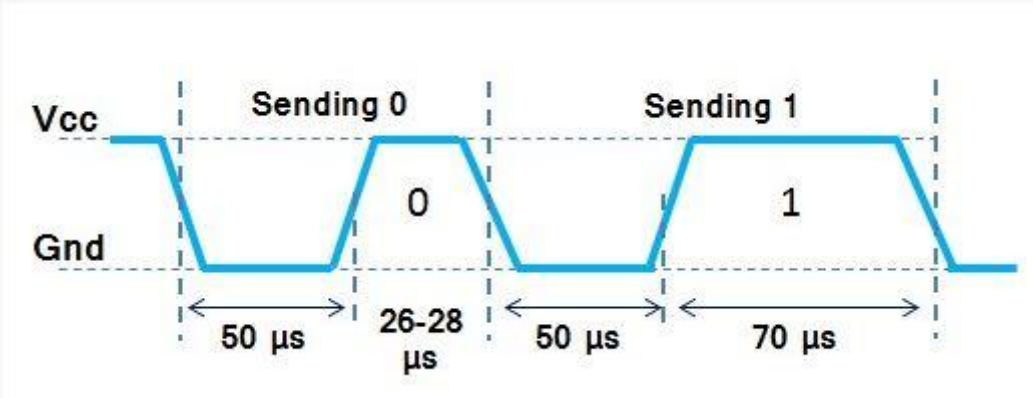


Fig. 2.3 Protocolo datos del sensor DHT22

La secuencia completa de la transmisión sería la siguiente:

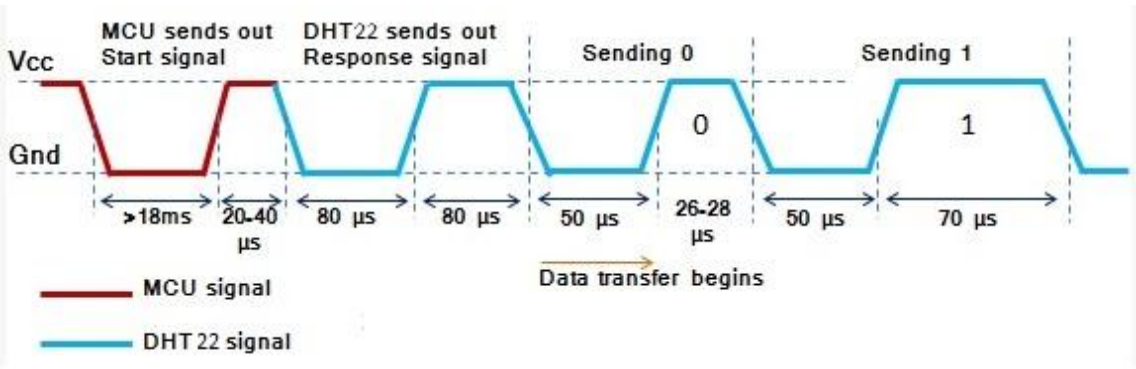


Fig. 2.4 Secuencia completa de transmisión del sensor DHT22

2.1.2. Diagrama de bloques y diseño del circuito

EL sensor DHT22 utiliza un único cable para la transmisión de datos además de los cables de alimentación Vcc y Gnd. En la Fig. 2.5 se muestra el esquema de conexión que se ha utilizado.

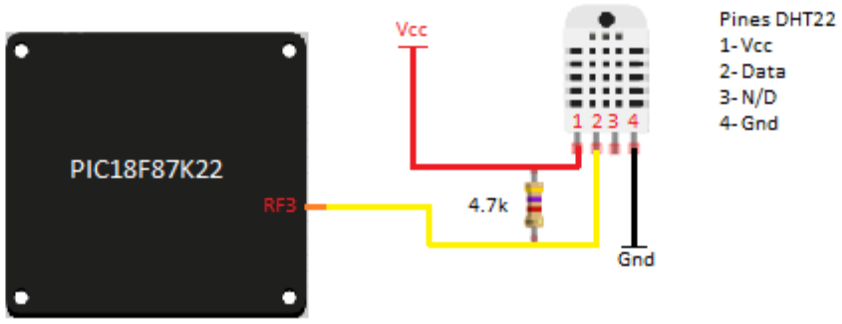


Fig. 2.5 Esquema de conexión del sensor DHT22

Conectamos el pin1 a Vcc, el pin2 es el pin de datos y necesita una resistencia de pull-up de al menos 3.3K para garantizar el correcto funcionamiento del bus de datos. Esta resistencia permite mantener el bus en alto cuando no está siendo utilizado para la comunicación. El pin de datos lo conectaremos al pin RF3 del microcontrolador y será este el que determine si el pin actúa como entrada o como salida dependiendo de si está enviando instrucciones o recibiendo datos. El pin 3 no se utiliza y el pin 4 lo conectamos a Gnd.

2.1.3. Diseño del programa con el estilo FSM

A continuación se muestra el diagrama FSM de estados (Véase Fig. 2.6) para la medida de la humedad y temperatura. Tenemos 5 estados: Idle, Setup, Read DHT22, Error DHT22 y Stop.

- Idle: Es el modo espera, se muestra por pantalla el mensaje de bienvenida. En este estado no hay comunicación con el sensor
- Setup: Se configura el módulo USART para la comunicación RS232 con el PC y se configura el pin de datos del sensor en alto.
- Read DHT22: Se inicia la comunicación con el sensor DHT22 y el envío de datos a la pantalla LCD y al PC.
- Error DHT22: Si el sensor no está disponible se envía un mensaje de error.
- Stop: Se detienen los módulos de comunicación USART y se pone a nivel bajo el pin de datos del sensor.

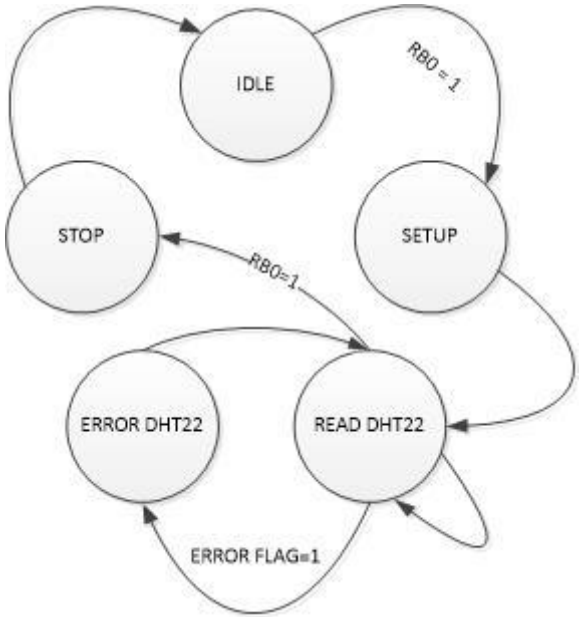


Fig. 2.6 Diagrama FSM para la lectura de humedad y temperatura

Para poner en funcionamiento el sensor hay que pulsar el botón RB0, en este momento se pasa al estado 'Setup' donde se configura el sistema y seguidamente al estado 'Read DHT22'. Si no detecta el sensor se activa la bandera de error de sensor y se muestra un mensaje de error por el LCD y se envía por USART el mensaje de error. Si detecta el sensor muestra cada 2 segundos (0,5Hz) la información de humedad y temperatura actual y se queda en el estado 'Read DHT22' hasta que pulsamos de nuevo el botón RB0 y el sensor y el envió de los datos se detiene y se vuelve al modo de espera 'Idle'.

Para leer los datos de humedad y temperaturas del sensor DHT22 se ha creado una función siguiendo las indicaciones del datasheet para crear el protocolo de comunicación. La Fig. 2.7 muestra el diagrama de flujo para la lectura de los datos del sensor. En primer lugar el microcontrolador envía la señal 'Start' al sensor, si el sensor responde se inicia la lectura de los 5 bytes de información y si no se envía el mensaje de error. Cuando llegan todos los datos se realiza el checksum y si este es correcto se comprueba el signo de la temperatura y se muestran los datos por el LCD y se envían vía USART. El Delay de 2 segundos se debe a que el sensor tiene una velocidad de muestreo de 0,5Hz y no puede proporcionar datos fiables antes de este tiempo.

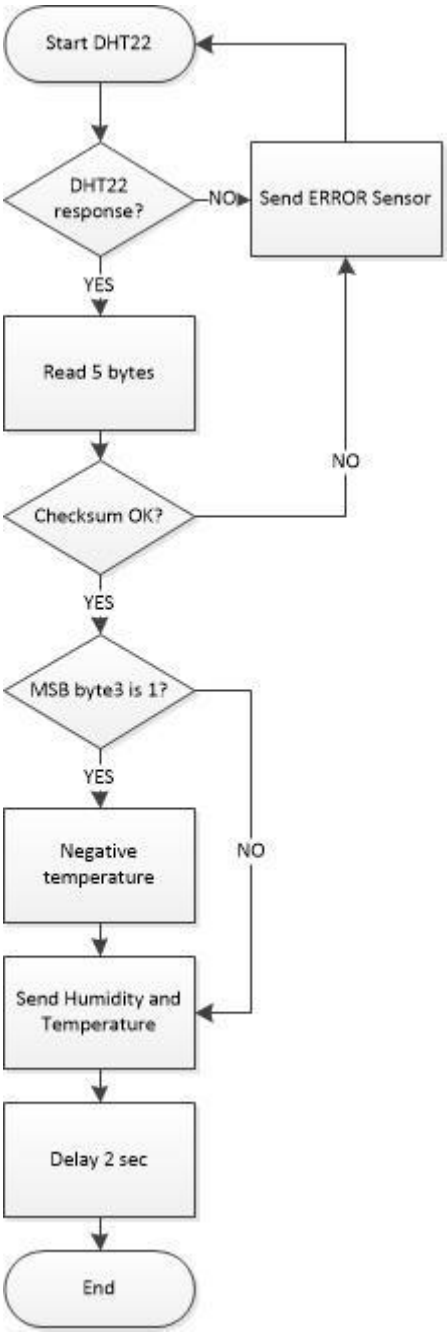


Fig. 2.7 Diagrama de flujo para la lectura del sensor DHT22

2.1.4. Simulación del diseño en Proteus

Para comprobar el correcto funcionamiento del código se ha simulado con Proteus. Se ha creado un esquemático (Véase Fig. 2.8) con el microcontrolador, el sensor DHT22, una pantalla LCD 16x2 y un terminal virtual que emula un puerto COM de comunicaciones, 7 leds, un botón conectado al pin RB0 del microcontrolador que genera una interrupción cuando se pulsa y un botón de reset del microcontrolador.

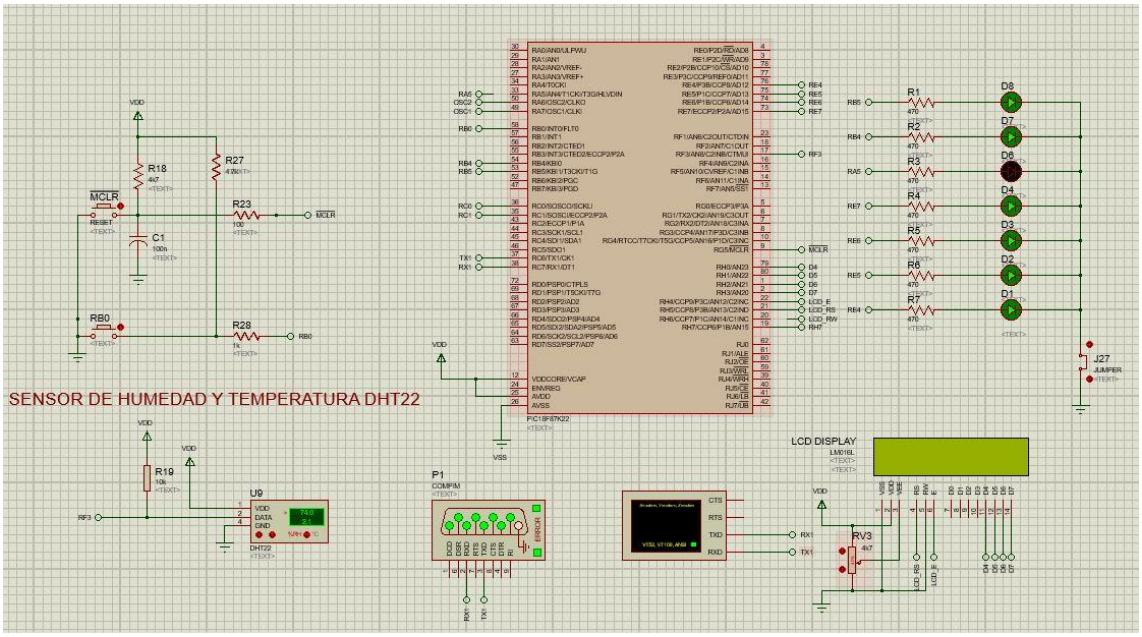


Fig. 2.8 Esquema Proteus sensor DHT22

En la Fig. 2.9 se puede ver el correcto funcionamiento del sistema.

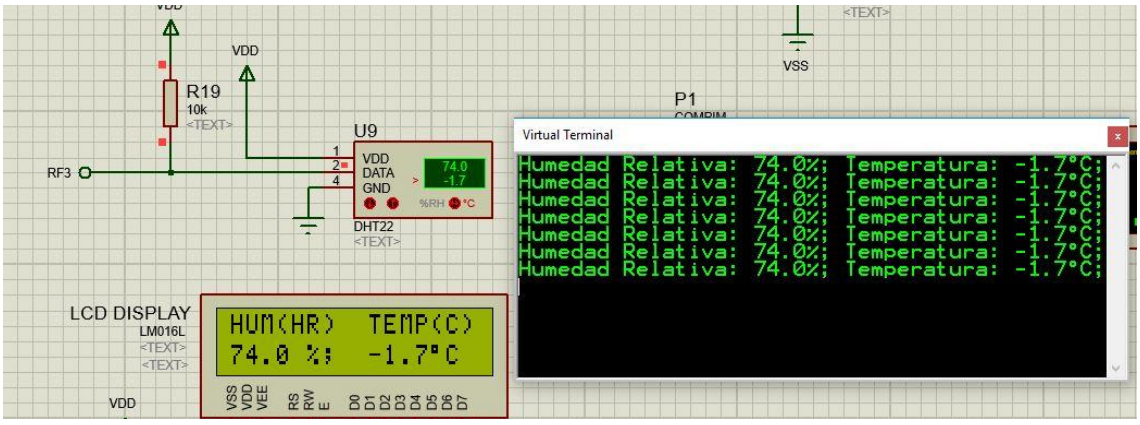


Fig. 2.9 Simulación del sensor DHT22

2.1.5. Prototipo sobre la placa de entrenamiento DM240313

Utilizaremos la placa de entrenamiento DM240313 con el microchip PIC18F87K22 para comprobar el funcionamiento real del sensor y la comunicación USART con el PC. La tarjeta cuenta con varios pines I/O (Véase ANEXO F Esquema placa de entrenamiento Microchip DM240313), para conectar el sensor DHT22 utilizaremos el pin RF3 para el bus de datos como en la simulación con el Proteus. Además añadiremos una pantalla LCD 16x2 como en la simulación para ver los valores de humedad y temperatura del sensor. Para la comunicación USART con el PC utilizaremos un adaptador USB-USART conectado a los pines Rx1 (pin RC7) y Tx1 (pin RC6) de la placa de pruebas (Véase Comunicaciones RS232).

La Fig. 2.10 muestra la conexión física del sensor y el conversor USB-USART con la placa de pruebas y el correcto funcionamiento del sistema.

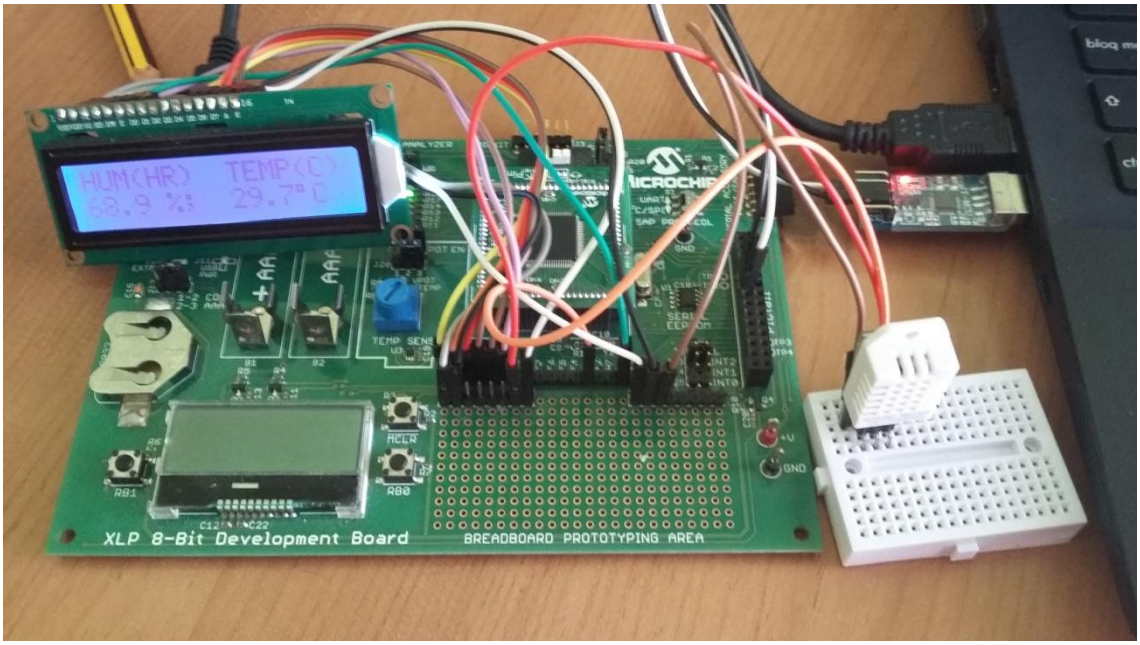


Fig. 2.10 Prueba del sensor DHT22 sobre la placa DM240313

Y la recepción de los datos en el HyperTerminal de Windows

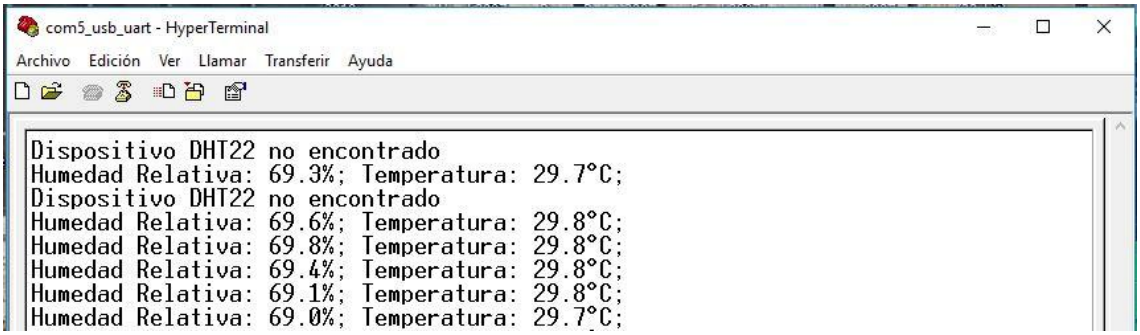


Fig. 2.11 Recepción de datos de humedad y temperatura

2.1.6. Comunicaciones RS232

Para la comunicación del microcontrolador con el PC utilizaremos el módulo USART (Universal Synchronous Asynchronous Receiver Transmitter) del microcontrolador.

Para la comunicación entro los dos dispositivos se necesitan 2 hilos, una para la transmisión (Tx) y otro para la recepción (Rx).

En el caso de la comunicación con el PC es necesario seguir la norma RS232 ya que los niveles de tensión con los que opera el módulo USART son TTL, un nivel bajo será 0V y un nivel alto será 5V mientras que los niveles de tensión

que utiliza la comunicación serial del PC operan entre -12V y 12V, además de trabajar con lógica negativa, lo que quiere decir que un nivel bajo será 12V mientras que un nivel alto será -12V.

Por lo tanto, para la comunicación entre PIC-PC será necesario un conversor de niveles.

Utilizaremos un conversor USB-USART (Véase Fig. 2.12) que ya lleva incorporado el conversor de niveles CP2102 y la conexión a puerto USB.

Fig. 2.12 Conversor USB-USART

La conexión de los dispositivos tiene que ser cruzada. Esto quiere decir que el pin Tx (RC6) del microcontrolador tiene que ir al pin Rx del conversor y el pin Rx (RC7) al Tx.

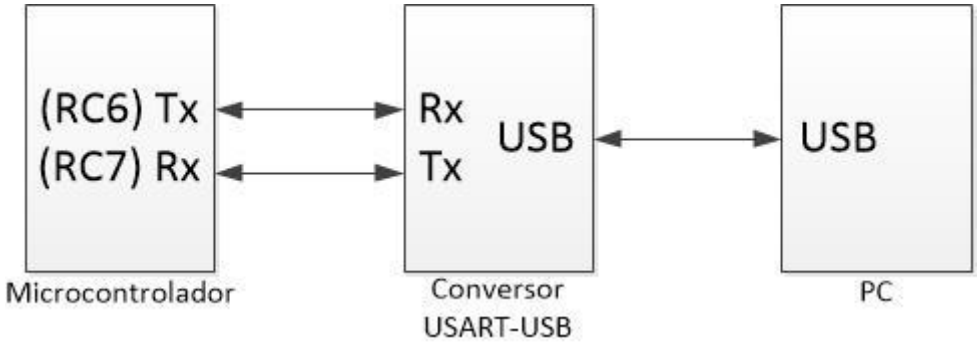


Fig. 2.13 Esquema de conexión conversor USB-USART

Hay dos modos de transmisión serie que se describen a continuación:

- Síncrona: Es una transmisión Half-Duplex. Uno de los hilos será utilizado para una señal de reloj de sincronización por parte del dispositivo maestro, mientras que el otro hilo será para la transmisión y recepción de datos no pudiendo ser esta simultanea.

- Asíncrona: Es una transmisión Full-Duplex. Uno de los hilos se utiliza para la transmisión y otro para la recepción. No es necesario una señal de reloj de sincronización, en su lugar es necesario configurar los parámetros de transmisión igual en emisor y receptor.

Para la comunicación con el PC utilizaremos una transmisión asíncrona y configuraremos el módulo con los siguientes parámetros de transmisión:

- Modo asíncrono
- Interrupción en recepción
- 8 bits de datos
- Recepción continua
- Bit de parada
- Velocidad de transmisión de 9600 baudios

2.2. Medida de la presión atmosférica

El siguiente proyecto será la medición de la presión atmosférica y la temperatura ambiente con el sensor BMP180. Se ha seguido el mismo método de trabajo que con el sensor DHT22, sin embargo el sensor BMP180 no está disponible para su simulación con Proteus y se ha pasado directamente a las pruebas con la placa de entrenamiento DM240313.

2.2.1. Funcionamiento básico del sensor BMP180

El sensor BMP180 (Véase Fig. 2.14), es un sensor digital que nos permite la medida de la presión atmosférica y la temperatura ambiente. El protocolo de comunicación que utiliza es el conocido I2C (Véase ANEXO A Protocolo de comunicación I2C) que se trata de un bus de datos serial desarrollado por Philips en el año 1992.

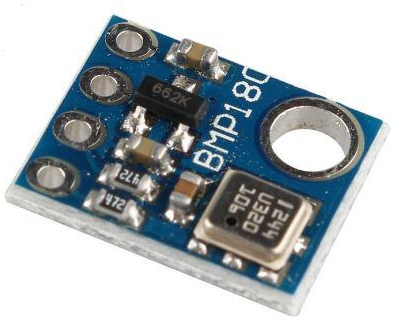


Fig. 2.14 Sensor de presión atmosférica y temperatura BMP180

El BMP180 consiste en un sensor piezo-resistivo, un convertidor analógico-digital, una unidad de control con memoria E2PROM y una interfaz I2C serie. La E2PROM almacena 176 bits de datos de calibración que se utilizan para compensar el offset, la dependencia de la temperatura y otros parámetros del sensor. Estos datos de calibración hay que leerlos una única vez al inicio de la lectura del sensor y nos servirán para calcular la presión y temperatura reales. La temperatura tiene una resolución de 16 bits y la presión de entre 16 y 19 bits según el modo de consumo y resolución escogidos.

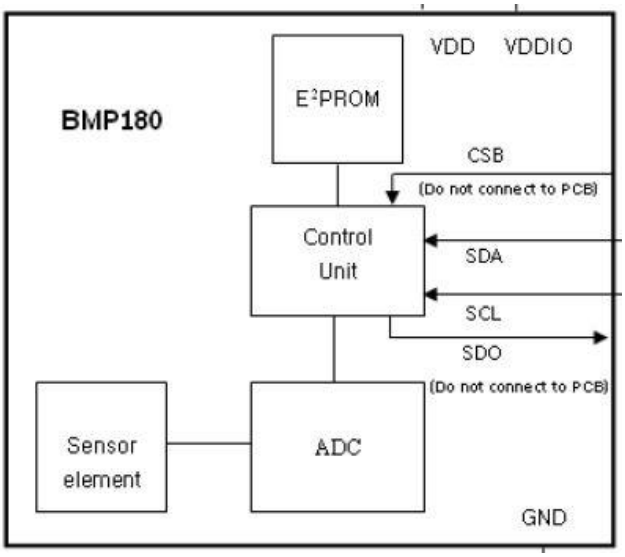


Fig. 2.15 Arquitectura interna del sensor BMP180

Características del sensor BMP180:

- Alimentación de 1.8V a 3.6V
- Bajo ruido 0.06 hPa en modo ultra bajo consumo
- Interfaz I2C
- Completamente calibrado
- Corriente máxima durante conversión 650 uA
- Modo de ultra bajo consumo 3 uA @ 1 Sample/s (25ºC)
- Rango de medición de presión atmosférica 300 hPa-1100 hPa
- Resolución presión atmosférica 0.01 hPa
- Precisión presión atmosférica 950 hPa-1050 hPa de ±0.12 hPa
- Rango de medición de temperatura 0 ºC - 65 ºC
- Resolución temperatura 0.1 ºC
- Precisión temperatura ±1 ºC

2.2.2. Diagrama de bloques y diseño del circuito

Para la comunicación con el microcontrolador, el sensor utiliza el protocolo I2C. Son necesarios 2 cables para la comunicación, el cable de datos (SDA) y el cable del reloj de sincronización (SCL), además de los cables de alimentación Vcc y Gnd. En la Fig. 2.16 se muestra el esquema de conexión del sensor BMP180.

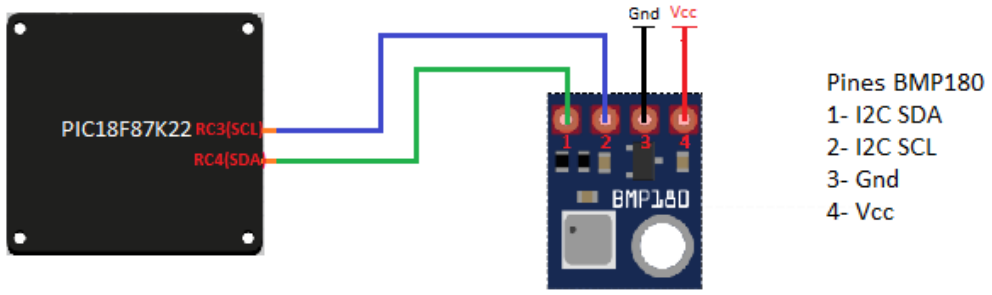


Fig. 2.16 Esquema de conexión del sensor BMP180

El pin 1 es el bus de datos SDA y va conectado al pin RC4 del microcontrolador, el pin 2 es el reloj de sincronización del bus I2C y va conectado al pin RC3. Ambos pines necesitan una resistencia de pull-up para funcionar correctamente. El pin 3 va conectado a Gnd y el pin 4 es la alimentación del sensor que como máximo es de 3.6V.

Hemos escogido el encapsulado del sensor BMP180 que ya tiene incorporado las resistencias de Pull-Up de 4.7K para el bus I2C y el regulador de tensión para garantizar una tensión fija de 3.3V al sensor, también incorpora condensadores de desacoplo que eliminan las interferencias y el ruido de la alimentación.

2.2.3. Diseño del programa con estilo FSM

Para la lectura y envío de datos de temperatura y presión atmosférica del sensor BMP180 tenemos el diagrama FSM de 6 estados (Véase Fig. 2.17) que se detallan a continuación:

- *Idle:* Es el modo de espera, los periféricos y el sensor están detenidos.
- *Setup:* En este modo se configuran los módulos I2C y USART.
- *Read calibration parameters BMP180:* Se leen los parámetros de calibración del sensor BMP180.
- *Read BMP180:* Se leen la temperatura y la presión no compensadas y junto con los parámetros de calibración se calcula la temperatura y la presión atmosférica reales.
- *Send data:* Se envía la información de temperatura y presión y se muestran por la pantalla LCD.
- *Stop:* Se detienen los módulos I2C y USART.

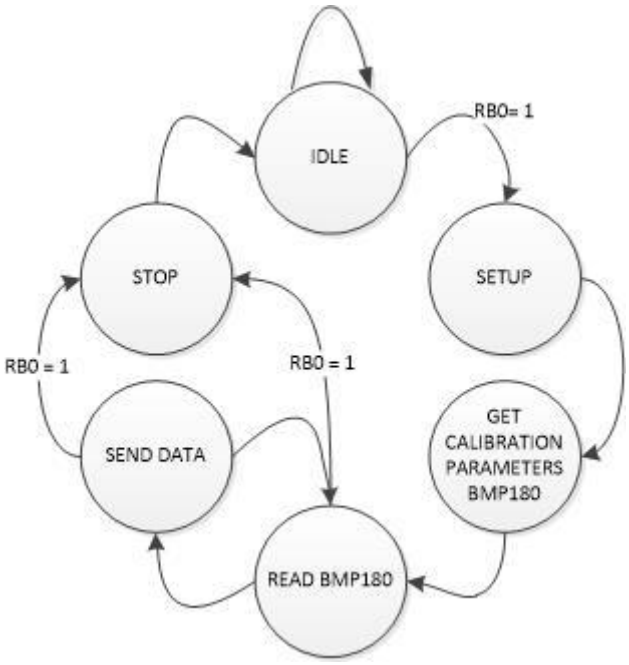


Fig. 2.17 Diagrama FSM para la lectura de presión y temperatura

El sistema se inicia por defecto en modo ‘Idle’, para ponerlo en funcionamiento tenemos que pulsar el botón ‘RB0’, en este momento se pasa al estado ‘Setup’ donde se configuran los módulos I2C y USART. Una vez configurado el sistema se lee una única vez los parámetros de calibración del sensor. En el estado ‘Read BMP180’ se leen la temperatura y la presión no compensadas y se calcula la real junto con los parámetros de calibración, justo después se envía la información y se vuelve a repetir el proceso de obtención de datos. Para volver al modo ‘Idle’ se tiene que pulsar de nuevo el botón ‘RB0’, en ese momento se pasa al estado ‘Stop’ donde se detienen los módulos y se vuelve al modo inicial ‘Idle’.

Para la lectura de los datos de temperatura y presión se ha creado una función siguiendo las indicaciones del fabricante. En la Fig. 2.18 se muestra el diagrama de flujo para la lectura de la temperatura y la presión atmosférica con el sensor BMP180.

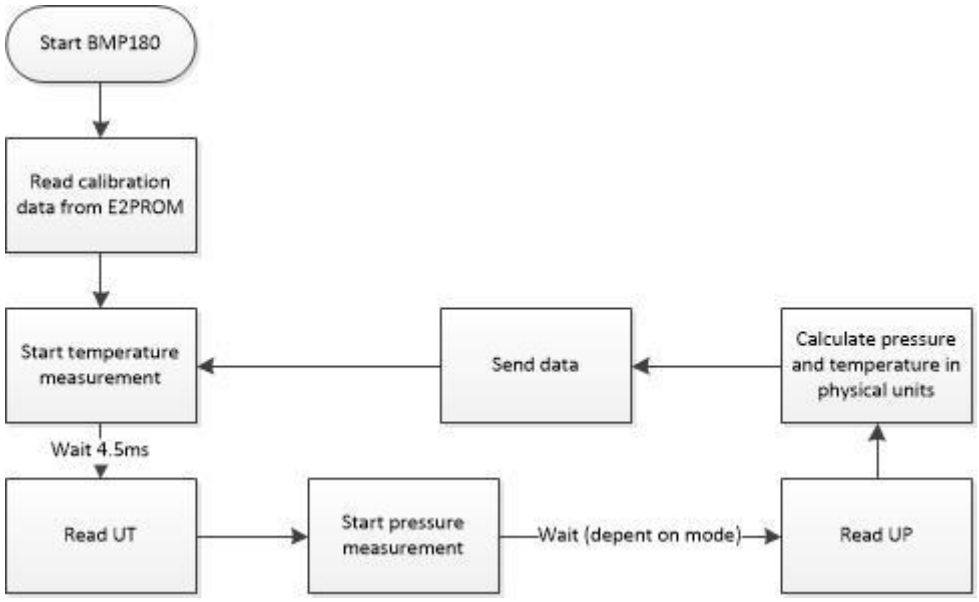


Fig. 2.18 Diagrama de flujo para la lectura del sensor BMP180

Los parámetros de calibración hay que leerlos al inicio antes de la lectura de los datos de temperatura y presión. Estos parámetros se utilizarán después para compensar el las medidas tomadas por el sensor de UT (Uncompensated Temperature) y UP (Uncompensated Pressure) y solo es necesario leerlos una vez al inicio del sistema ya que estos no cambian, son únicos para cada sensor.

La espera después de la lectura de la presión depende del modo de resolución (Véase Fig. 2.19) que hayamos escogido en el sensor. La siguiente tabla muestra un resumen del tiempo de conversión según el modo escogido y el consumo que tiene este.

Mode	Parameter *oversampling_setting*	Internal number of samples	Conversion time pressure max. [ms]	Avg. current @ 1 sample/s typ. [µA]	RMS noise typ. [hPa]	RMS noise typ. [m]
ultra low power	0	1	4.5	3	0.06	0.5
standard	1	2	7.5	5	0.05	0.4
high resolution	2	4	13.5	7	0.04	0.3
ultra high resolution	3	8	25.5	12	0.03	0.25

Fig. 2.19 Resolución y tiempo de conversión del sensor BMP180

Una vez leídos los datos de calibración, los de temperatura y presión no compensada, el fabricante nos proporciona el algoritmo (Véase Fig. 2.20) que tenemos que implementar para obtener los valores reales de temperatura y presión atmosférica. La siguiente imagen muestra el algoritmo implementado y un ejemplo de aplicación para obtener correctamente los resultados. Para implementar el código se ha seguido primeramente este ejemplo hasta obtener los resultados correctos y una vez comprobado se ha pasado a realizar una prueba con el sensor real.

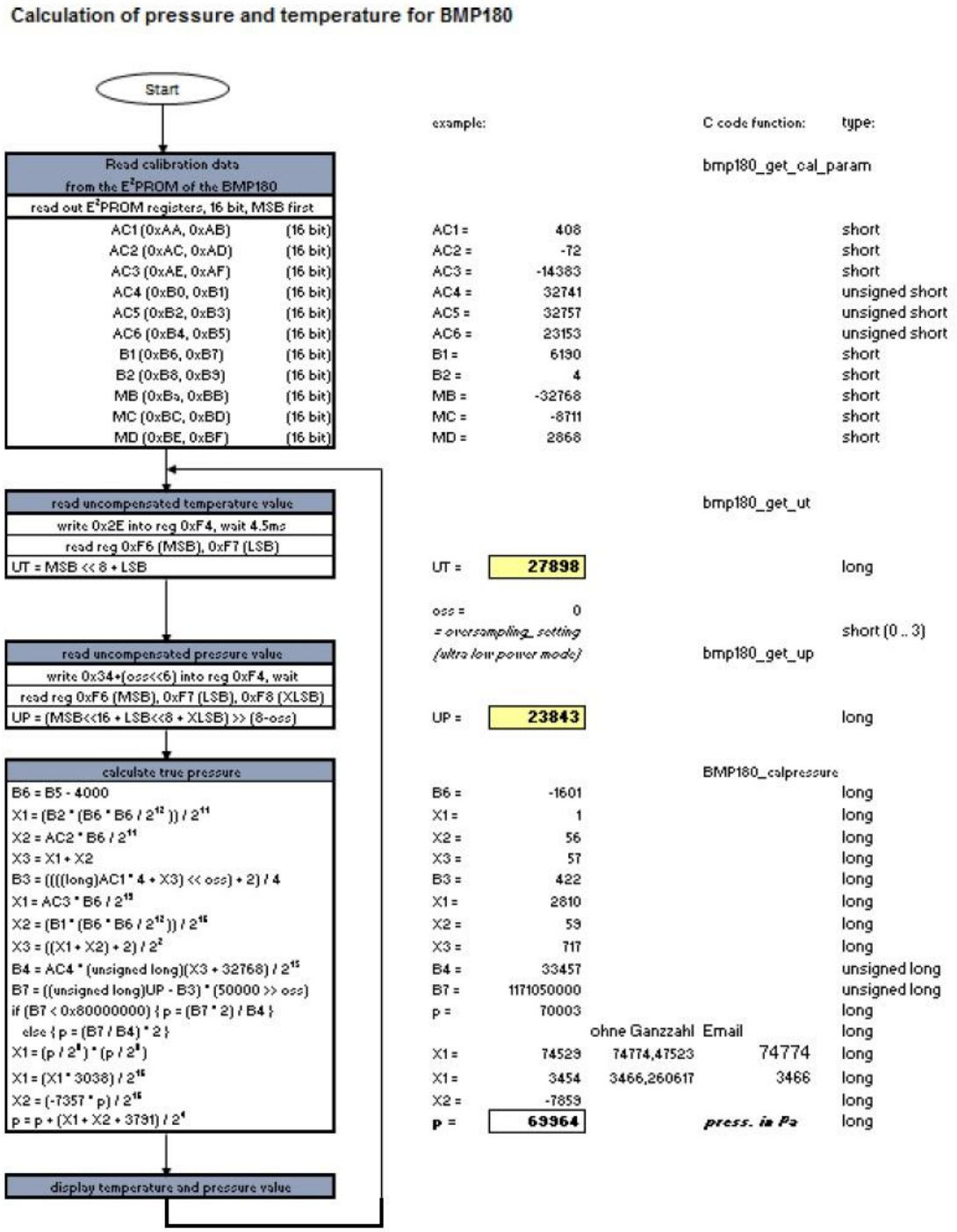


Fig. 2.20 Algoritmo para el cálculo de temperatura y presión atmosférica

2.2.4. Prototipo sobre la placa de entrenamiento DM240313

La prueba con el sensor se ha realizado directamente sobre la placa de entrenamiento ya que en el simulador Proteus no está disponible el sensor BMP180.

Para la comunicación I2C se han utilizado los pines RC3 (SCL) y RC4 (SDA) y para la USART los pines Rx1 (pin RC7) y Tx1 (pin RC6).

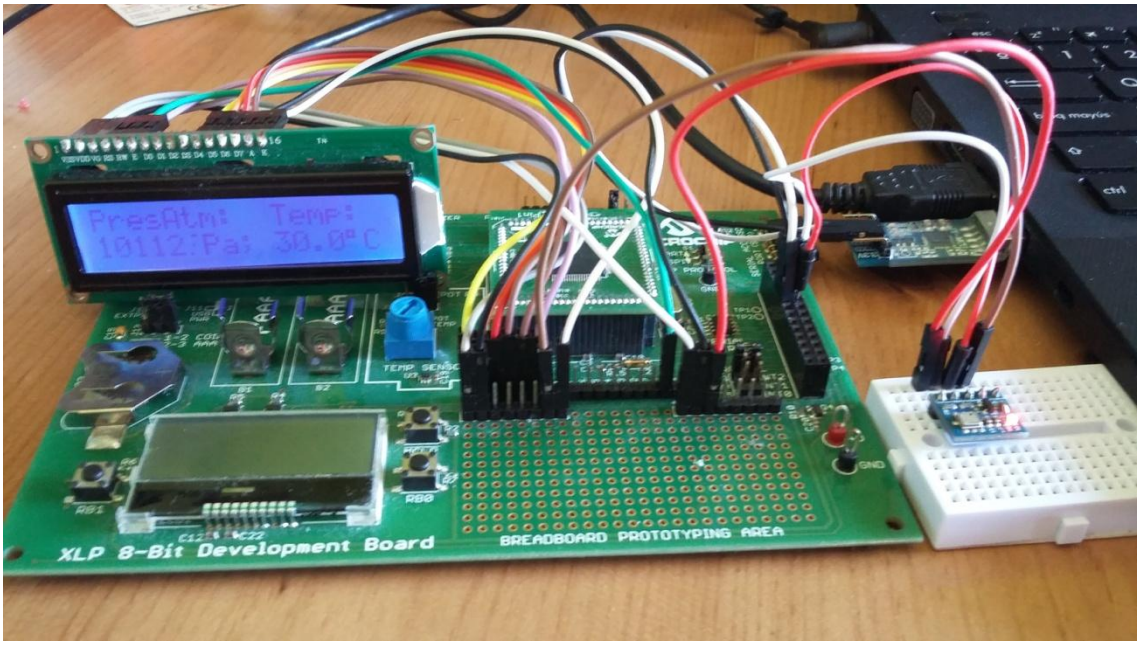


Fig. 2.21 Prueba del sensor BMP180 sobre la placa DM240313

Y la recepción de los datos en el HyperTerminal de Windows

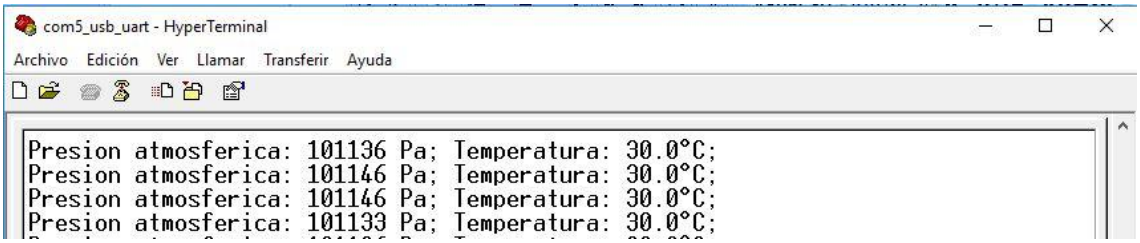


Fig. 2.22 Recepción de datos de presión atmosférica y temperatura

2.3. Medida de la temperatura del agua

En este tercer proyecto se medirá la temperatura del agua mediante el sensor DS18B20 (Véase Fig. 2.23). Se trata de un sensor que utiliza el protocolo de comunicación 1-Wire de Dallas Semiconductor (Véase ANEXO B Protocolo de comunicación 1-W) que utiliza un único cable para la transmisión de los datos.

Se ha escogido el encapsulado del sensor sumergible de manera que el agua no entrará en contacto con la electrónica.

2.3.1. Funcionamiento básico del sensor DS18B20

El sensor DS18B20 es un sensor digital que permite la medición de la temperatura con una resolución configurable de 9 a 12 bits que corresponden a incrementos de 0.5ºC, 0.25ºC, 0.125ºC, y 0.0625ºC. Siendo la de 12 bits la utilizada por defecto. Utiliza el protocolo de transmisión 1-Wire de Dallas semiconductor que utiliza un único cable de datos para la comunicación entre dispositivos. El protocolo permite la utilización del mismo bus por varios sensores ya que cada dispositivo viene con una dirección única de 64 bits. El sensor también puede operar en modo alimentación parásita eliminando así la necesidad de conectarlo a una fuente de alimentación externa, el sensor utiliza la energía que recolecta del bus de datos cuando este no se utiliza almacenándola en un condensador.

Fig. 2.23 Sensor impermeable de temperatura DS18B20

Características del sensor DS18B20:

- Alimentación de 3v A 5.5v
- Consumo de 750nA en modo Standby y 1 mA en modo activo
- Protocolo de comunicación 1-Wire que requiere un único pin para la comunicación
- Rango de medición de temperatura de -55ºC a 125ºC
- Resolución programable de 9 a 12 bits (0.5ºC, 0.25ºC, 0.125ºC y 0.0625ºC)
- Exactitud de ±0.5ºC de-10ºC a 85ºC
- Alarma programable de alta y baja temperatura
- Posibilidad de alimentación en modo parásito sin necesidad de alimentación externa

La Fig. 2.24 muestra la arquitectura interna del sensor:

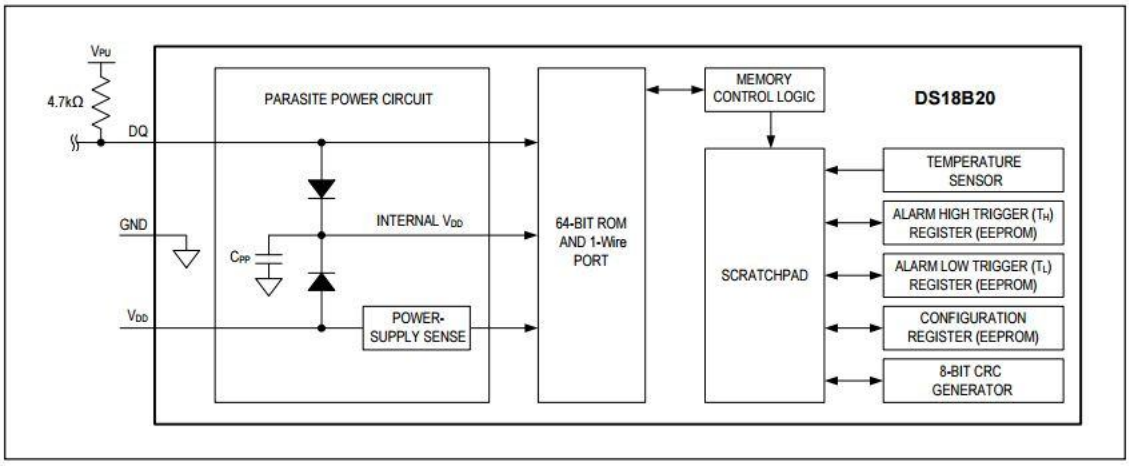


Fig. 2.24 Arquitectura interna del sensor DS18B20

En el encapsulado encontramos la ROM de 64 que contiene la dirección única del sensor, la memoria Scratchpad que contiene el registro de 2 bytes que almacena la salida digital del sensor de temperatura, además proporciona acceso al byte donde se almacena los registros de alarma de temperatura alta y baja (TH y TL) y el byte de registro de configuración donde se almacena la configuración de resolución del sensor. Los registros de configuración y de alarma TH y TL no son volátiles, de esta manera conservan los datos una vez apagado el sensor.

2.3.2. Diagrama de bloques y diseño del circuito

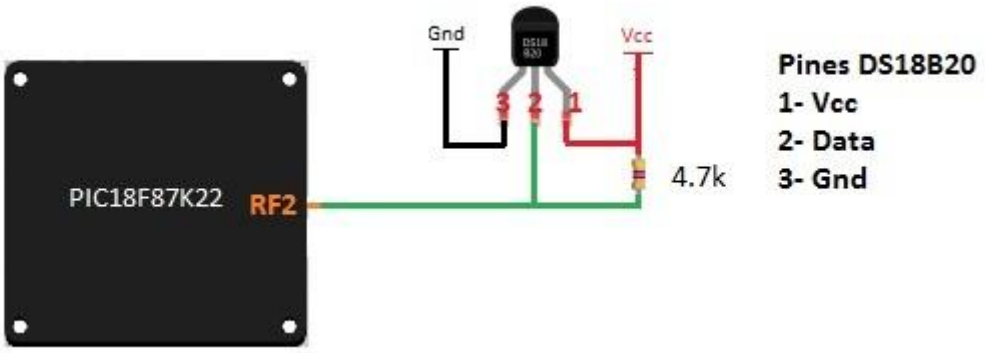


Fig. 2.25 Esquema de conexión del sensor DS18B20

La conexión se realizará con alimentación externa. El pin de datos necesita una resistencia de pull-up de 4,7K para mantener el nivel del bus en alto cuando este no se utiliza. El pin de datos irá conectado al pin RF2 del microcontrolador y estará configurado como entrada o como salida según se envíen o se reciban datos.

2.3.3. Diseño del programa con estilo FSM

Para la lectura del sensor DS18B20 utilizamos los mismos estados que en los proyectos anteriores. Tenemos los estados Idle, Setup, Read DS18B20, Send data y Stop.

- Idle: Es el modo de espera, los módulos de comunicaciones USART y el sensor están desconectados.
- Setup: Configuramos los módulos de comunicación USART y ponemos en alto (Vcc) el puerto RF2 donde va conectado el sensor.
- Read DS18B20: Iniciamos la comunicación con el sensor para obtener los datos de la temperatura del agua.
- Send Data: Enviamos los datos del sensor al PC y los mostramos por el LCD.
- Stop: Se detiene la comunicación con el sensor, el módulo USART y se pone a nivel bajo el pin del sensor.

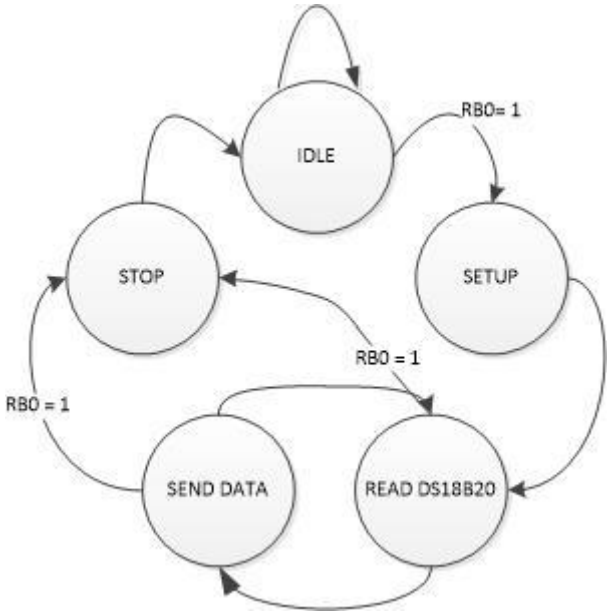


Fig. 2.26 Diagrama FSM para la lectura de la temperatura del agua

El estado inicial es el estado ‘Idle’, si pulsamos el botón RB0 pasamos al modo ‘Setup’, seguidamente se pasa a la lectura del sensor en el estado ‘Read DS18B20’. Una vez leído el sensor en el estado ‘Send data’ se muestran los datos de temperatura por el LCD y se envían los datos al PC. Si pulsamos el botón RB0 se detiene la lectura y el envío de datos y se vuelve al modo ‘Idle’.

El sensor DS18B20 utiliza el protocolo 1-Wire. Siguiendo las indicaciones del fabricante hemos creado una función para la lectura de los datos.

En la Fig. 2.27 se muestra el diagrama de flujo podemos se muestra la lectura del los datos del sensor.

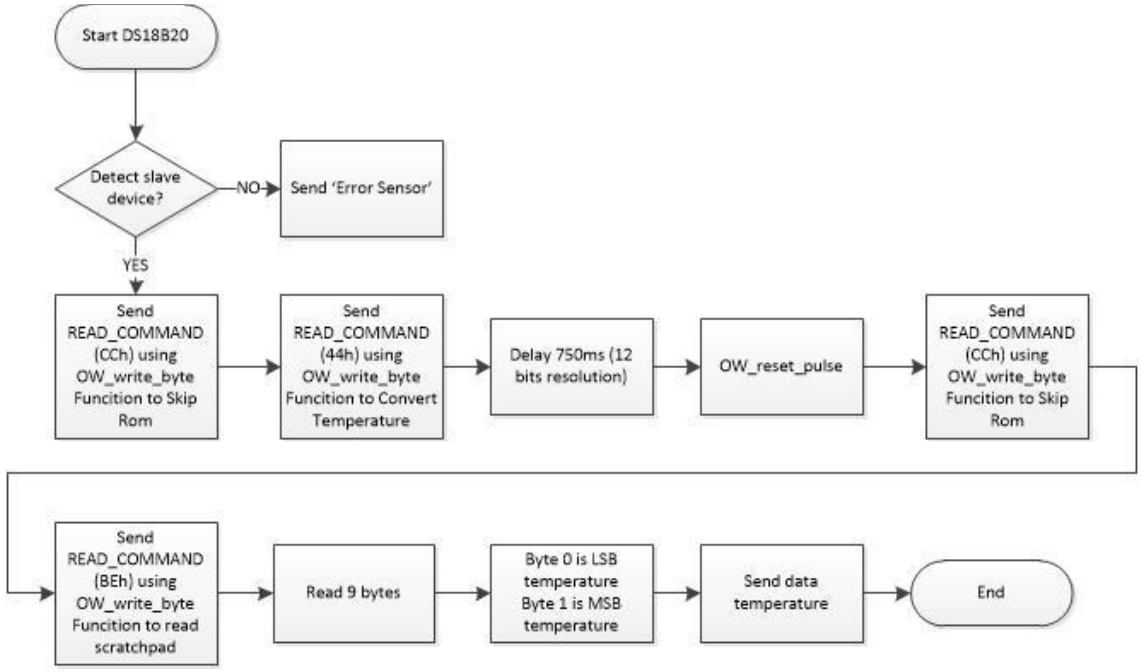


Fig. 2.27 Diagrama de flujo para la lectura del sensor DS18B20

La lectura se inicia con la detección del sensor, si este no devuelve respuesta al microcontrolador se envía el mensaje de error de sensor. Si detectamos el sensor accedemos a la Scratchpad (Véase Fig. 2.28) mediante el comando (CCh) para indicar al que queremos obtener los datos de temperatura mediante el comando (44h). La lectura y conversión de la temperatura en la resolución por defecto de 12 bits tarda 750ms, cuando termina tenemos que acceder al Scratchpad del sensor para leer los 9 bytes de información (comando BEh). En la siguiente figura se puede ver cómo está distribuida la información en el Scratchpad del sensor. Los datos que nos interesan son los dos primeros bytes, que es donde se almacenan los datos de la temperatura.

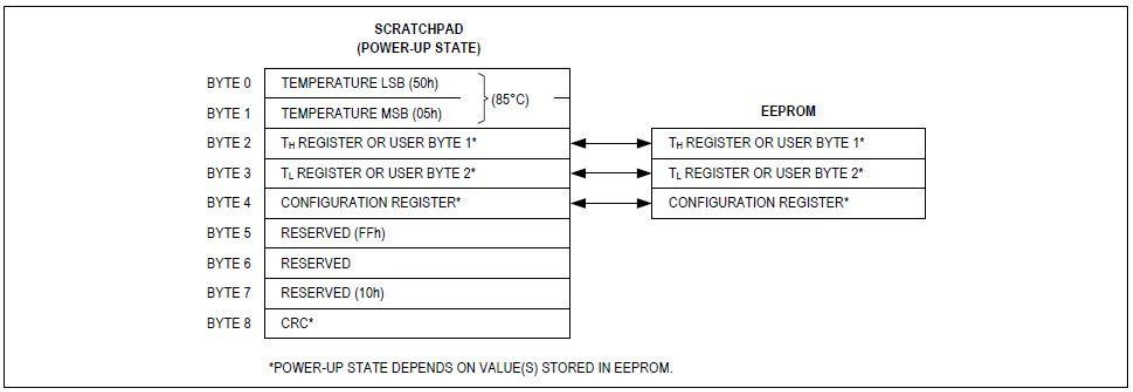


Fig. 2.28 Scratchpad del sensor DS18B20

La temperatura está calibrada en grados Celsius y se almacena como un número en complemento a dos. Los cuatro bits más significativos indican el

signo de la temperatura siendo ‘0000’ para números positivos y ‘1111’ para negativos. Los 12 siguientes bits indican la temperatura. Siendo los 8 bits más significativos la parte entera de la temperatura y los 4 bits menos significativos la parte decimal. En la siguiente figura se puede ver un ejemplo de cómo interpretar los 16 bit de datos con una resolución de 12 bits (0.625 ºC).

Fig. 2.29 Formato de datos del sensor DS18B20

- Hallamos el complemento a dos: 0000 0000 1010 0010
- Parte entera: 0000 1010 = 10
- Parte decimal 0010 = 2 ; 2*0.0625 = 0.125
- Número a mostrar: -10.125

Si la resolución es de 12 bits (0.0625 ºC) se utilizan los 4 bits de la parte decimal, si es de 11 bits se utilizan los 3 bits más significativos, así hasta llegar a la resolución de 9 bits (0.5ºC) donde solamente utilizamos el bit más significativo rellenando el resto de ‘0’.

2.3.4. Simulación del diseño en Proteus

Se ha creado un esquemático con Proteus (Véase Fig. 2.30) para simular la lectura del sensor DS18B20 que incluye el microcontrolador, el sensor, botones RB0 y Reset, una pantalla LCD 16x2, 7 leds de control y el terminal virtual para la comunicación USART (RS232).

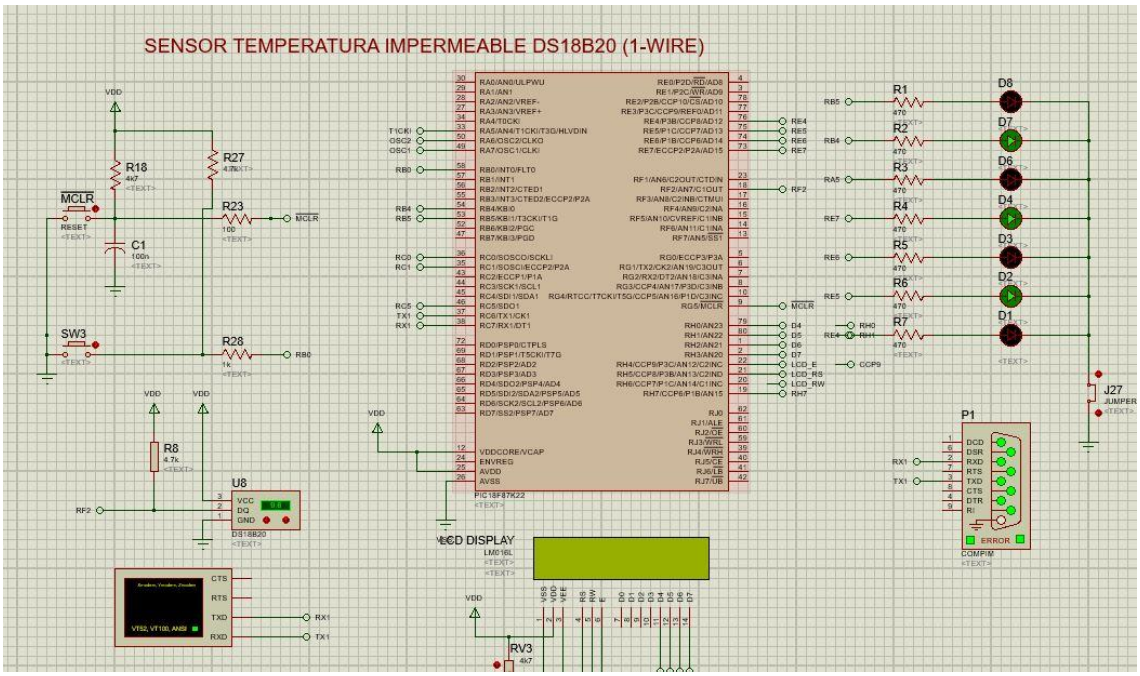


Fig. 2.30 Esquema Proteus sensor DS18B20

En la Fig. 2.31 se puede comprobar el funcionamiento del sistema.

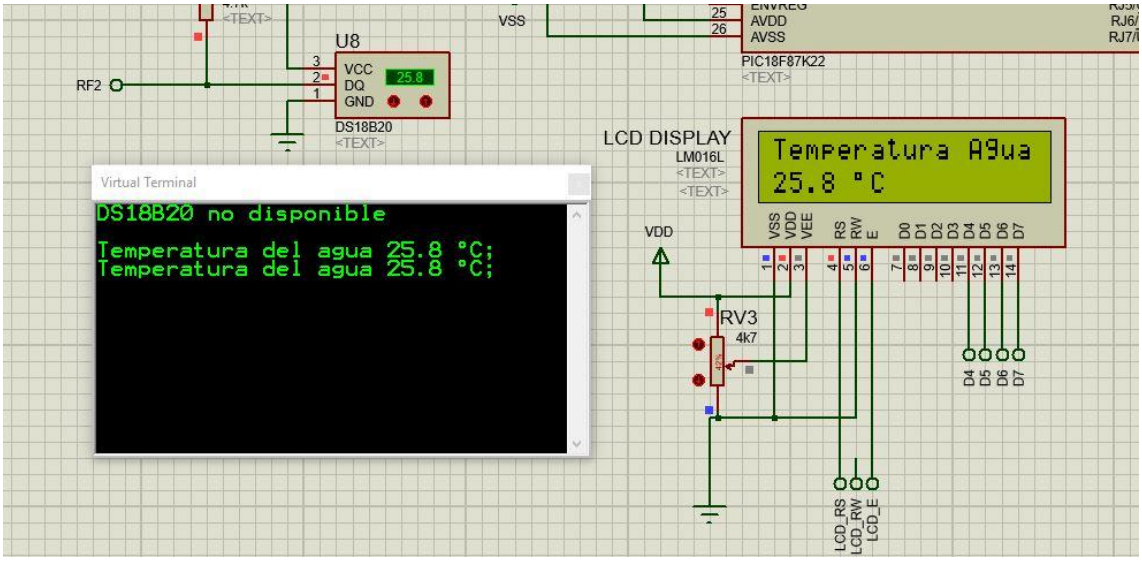


Fig. 2.31 Simulación del sensor DS18B20

2.3.5. Prototipo sobre la placa de entrenamiento DM240313

Una vez realizada la simulación, utilizamos la placa de entrenamiento para comprobar el funcionamiento del sensor real.

Conservamos el montaje de los proyectos anteriores, cambiando únicamente el sensor. Utilizaremos el pin RF2 de la placa para conectar el bus de datos del sensor, añadiendo una resistencia de Pull-Up de 4.7k entre Vcc y el pin RF2.

En la Fig. 2.32 podemos ver la conexión del sensor a la placa y la prueba de funcionamiento.

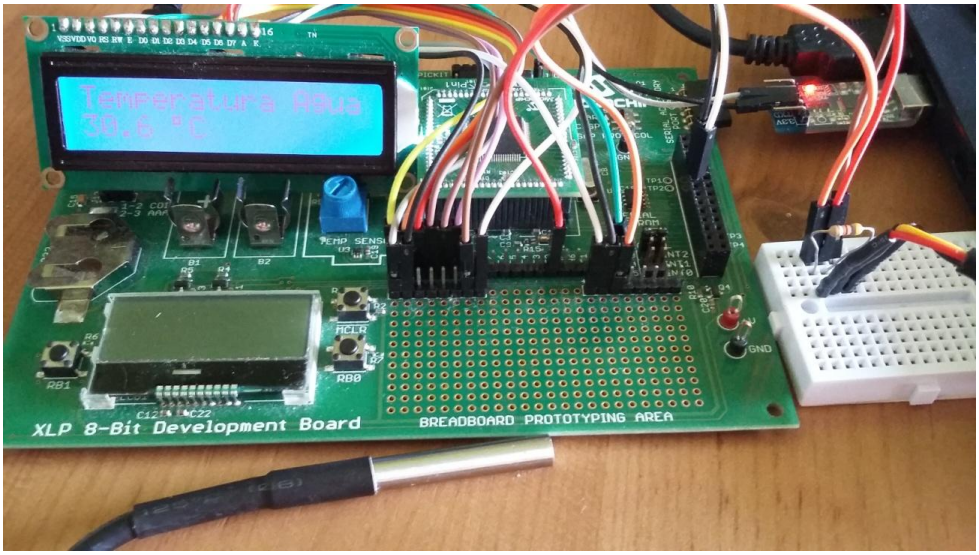


Fig. 2.32 Prueba del sensor DS18B20 sobre la placa DM240313

Y la recepción de los datos en el HyperTerminal de Windows

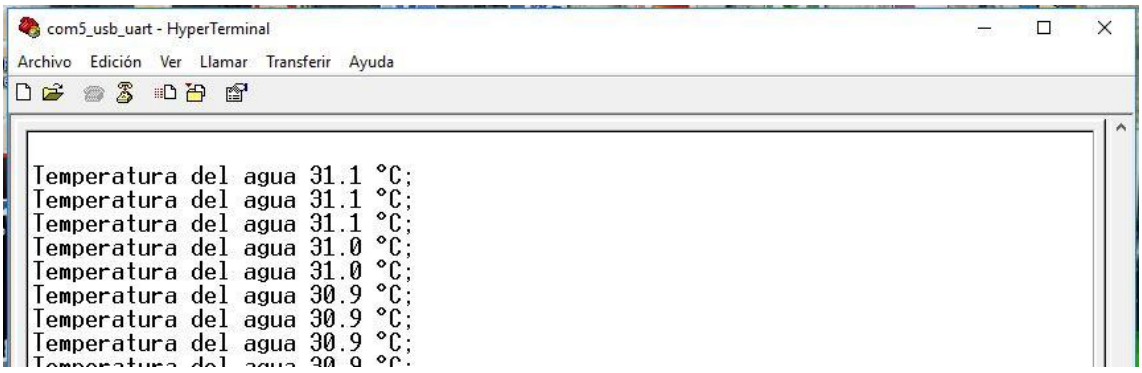


Fig. 2.33 Recepción de datos de temperatura del agua

2.4. Medida de la dirección y velocidad del viento: Veleta y anemómetro

En esta última fase mediremos la dirección y la velocidad del viento mediante una veleta y un anemómetro.

2.4.1. Funcionamiento básico de la veleta

Una veleta (Véase Fig. 2.34), es un dispositivo giratorio que se utiliza para conocer la dirección del viento. Esta dispone de una placa que gira libremente y un señalador que indica la dirección del viento junto con una cruz que muestra los puntos cardinales.

Fig. 2.34 Veleta

La veleta que utilizaremos se compone de 8 interruptores cada uno conectado a una resistencia diferente. Un imán de aleta puede cerrar dos interruptores a la vez lo que permite hasta 16 posiciones diferentes.

Se trata de un simple divisor resistivo que nos da un valor de tensión diferente según la posición de la veleta. Este valor lo mediremos con el conversor analógico digital (ADC) del microcontrolador para obtener la dirección del viento.

En la Fig. 2.35 podemos ver cómo está configurado el divisor resistivo de la veleta. Tan solo tenemos que añadir una resistencia de valor conocido y la alimentación para obtener los valores de tensión diferentes a la salida según la dirección del viento.

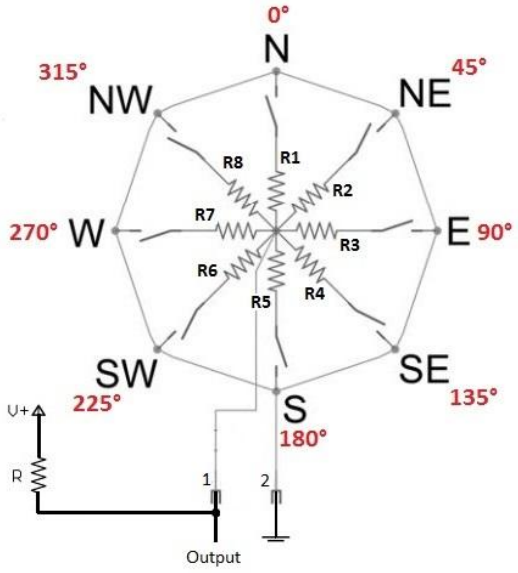


Fig. 2.35 Configuración interna de la veleta

Cuando el imán cierra 2 interruptores a la vez, el resultado es de dos resistencias en paralelo lo que nos permite tener hasta 16 direcciones diferentes. En la Tabla 2.1 se muestra un ejemplo de lo que obtendríamos a la salida si alimentamos el sensor con 3.3V y añadimos una resistencia R de 10K.

Tabla 2.1 Tensión de salida de la veleta según la dirección del viento

Grados	Dirección	Resistencia	Valor	Salida (V=3,3v, R= 10K)
0°	N	R1	33K	2,53 V
22,5°	NNE	R1/R2	6,57K	1,31 V
45°	NE	R2	8,2K	1,49 V
67,5°	ENE	R2/R3	891	0,27 V
90°	E	R3	1K	0,3 V
112,5°	ESE	R3/R4	688	0,21 V
135°	SE	R4	2,2K	0,6 V
157,5°	SSE	R4/R5	1,41K	0,41 V
180°	S	R5	3,9K	0,93 V
202,5°	SSO	R5/R6	3,14K	0,79 V
225°	SO	R6	16K	2,03 V
247,5°	OSO	R6/R7	14,12K	1,93 V
270°	O	R7	120K	3,05 V
292,5°	ONO	R7/R8	42,12K	2,67 V
315°	NO	R8	64,9K	2,86 V
337,5°	NNO	R8/R1	21,88K	2,26 V

2.4.2. Funcionamiento básico del anemómetro

Un anemómetro (Véase Fig. 2.36), consiste en un elemento giratorio con varias aspas con cazoletas. Cuando el viento sopla estas aspas comienzan a girar, lo que nos permite calcular la velocidad del viento.

Fig. 2.36 Anemómetro

El anemómetro que utilizaremos será uno de tipo de cierre por contacto. Consta de un interruptor que cierra el circuito cada vez que da una vuelta. Según los datos del fabricante con una velocidad de 2.4 Km/h se produce un cierre por segundo.

Para implementar la medida de la velocidad del viento, utilizaremos una de las interrupciones por pulsación de las que dispone el microcontrolador. Estas interrupciones detectan si ha habido una pulsación en el pin de entrada. Contando las interrupciones durante un periodo de tiempo calcularemos la velocidad instantánea del viento.

Para crear la base de tiempo, utilizaremos uno de los Timers de los que dispone el microcontrolador. Los Timers se pueden utilizar como temporizador que una vez alcanzan el tiempo configurado producen una interrupción en el microcontrolador.

Queremos una resolución de 1Km/h de velocidad. Sabiendo que una vuelta del anemómetro corresponde a 2.4 Km/h hacemos la siguiente conversión:

$$\frac{1\,pulso}{2.4\,s} = 1\,Km/h \qquad \textbf{(2.1)}$$

Implementando una base de tiempo de 2,4 segundos con el Timer 0 (Véase ANEXO C Configuración Timer 0 PIC18F87K22), si contamos los pulsos que transcurren durante ese periodo, obtendremos la velocidad del viento en Km/h.

2.4.3. Diagrama de bloques y diseño del circuito

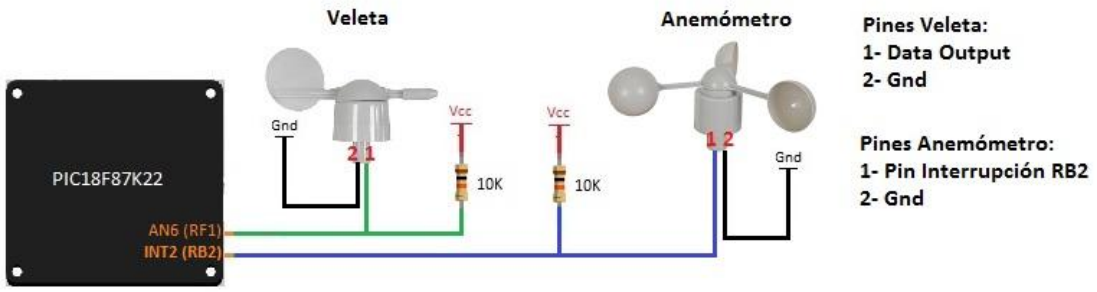


Fig. 2.37 Esquema de conexión del anemómetro

Para conectar la veleta utilizaremos el pin RF1 (AN6) que corresponde al canal analógico 6 del ADC del microcontrolador. Es necesaria una resistencia de 10K para crear el divisor de tensión y obtener diferentes valores de tensión según la dirección del viento.

El anemómetro lo conectaremos al pin RB2. Este es un pin especial del microcontrolador que genera una interrupción en el programa cada vez que detecta una pulsación (transición de Vcc a Gnd). Es necesaria una resistencia de Pull-Up de10K para mantener en alto el pin y poder detectar correctamente las pulsaciones.

2.4.4. Diseño del programa con estilo FSM

Añadiremos a los estados ‘Idle’, ‘Setup’, y ‘Stop’ los estados de lectura de la veleta ‘Read windvane’ y del anemómetro ‘Read wind speed’.

- Idle: Modo de espera. No se realizan lecturas de los sensores. Las interrupciones en el pin RB2, el módulo USART y el ADC están desactivados.
- Setup: Se inicia el módulo de comunicación USART.
- Read Windvane: Se inicia el módulo ADC y se realiza la lectura de la veleta. Cuando finaliza se guarda el valor y se desactiva el módulo ADC.
- Read Windspeed: Se configura el Timer 0, se activan las interrupciones en RB2 y se leen los pulsos en el pin durante 2,4 segundos. Una vez transcurrido el periodo de 2,4, se detiene el Timer 0, se desactivan las interrupciones en RB2 y se calculada la velocidad del viento.
- Send data: Se envían los datos de dirección y velocidad del viento y se muestran por la pantalla LCD.
- Stop: Se detiene el módulo de comunicaciones USART

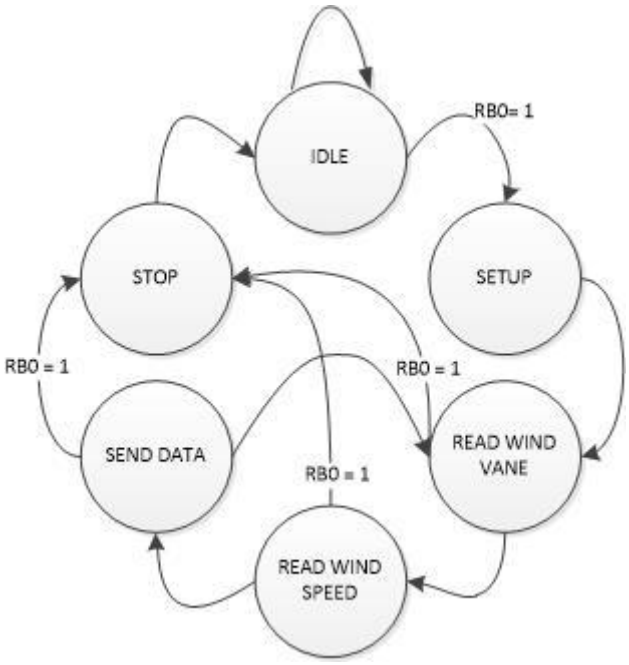


Fig. 2.38 Diagrama FSM para la lectura de la dirección y velocidad del viento

El estado inicial es el modo 'Idle' como en los anteriores casos, para iniciar la lectura de los datos de dirección y velocidad del viento hay que pulsar el botón RB0. Para parar la lectura se pulsa de nuevo el botón RB0 y se vuelve al estado inicial 'Idle'.

Para la lectura de la veleta y el anemómetro se han creado sendas funciones. En la Fig. 2.39 se muestra el diagrama de flujo la lectura de la veleta.

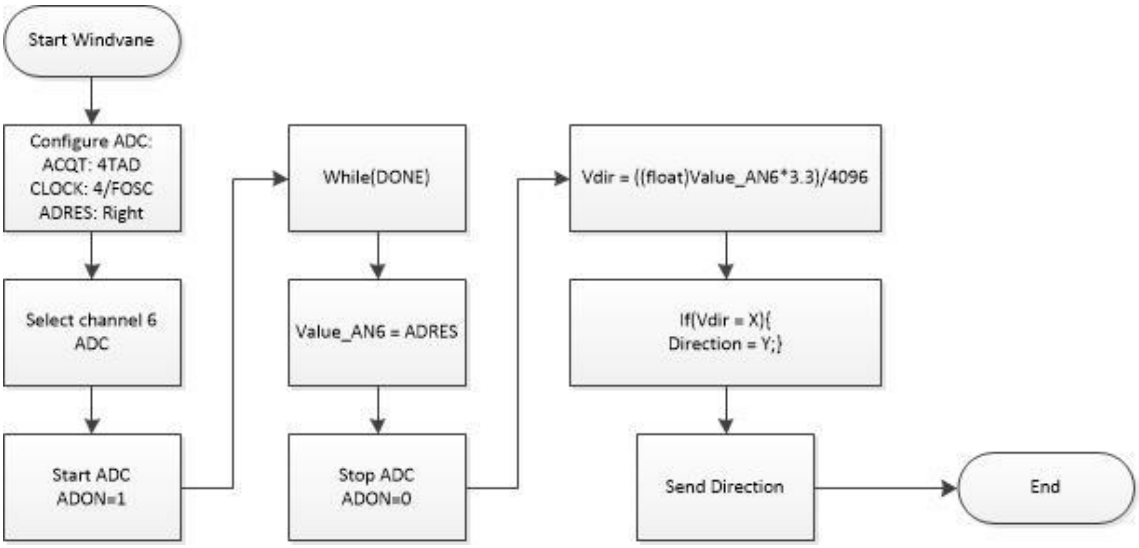


Fig. 2.39 Diagrama de flujo para la lectura de la veleta

Para leer la dirección del viento lo primeo que hacemos es configurar la velocidad de muestro, el tiempo de adquisición y el justificado del resultado del ADC del microcontrolador (Ver Anexo), seguidamente seleccionamos el canal 6 (AN6) del ADC para realizar la lectura. Iniciamos la lectura y esperamos a que finalice. Guardamos el valor y detenemos el ADC. Comprobamos el valor, determinamos la dirección del viento y enviamos los datos.

La Fig. 2.40 corresponde al diagrama de flujo de la lectura de la velocidad del viento.

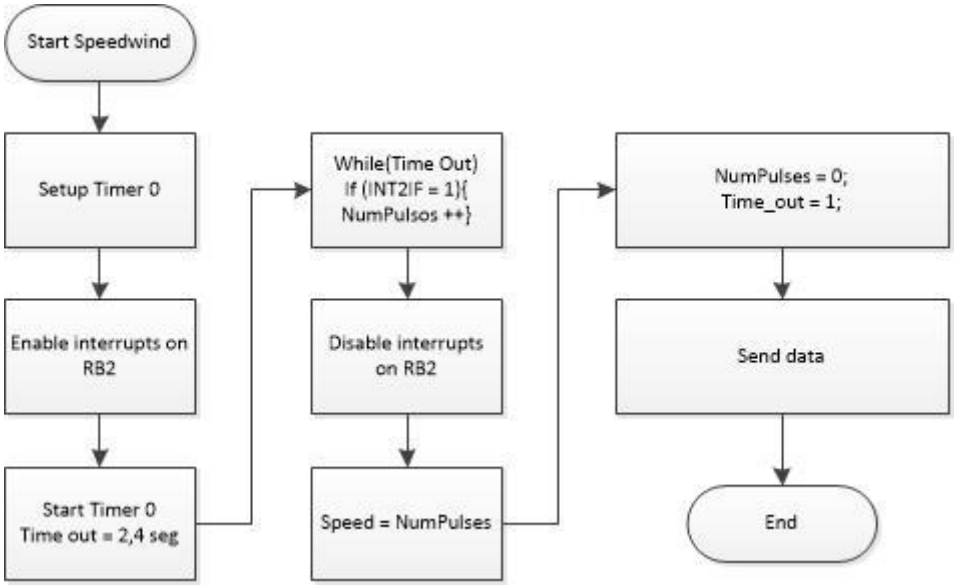


Fig. 2.40 Diagrama de flujo para la lectura del anemómetro

Se inicia la configuración del Timer 0 para una base de 2,4 segundos (Ver Anexo), se habilitan las interrupciones en el puerto RB2, se inicia el temporizador y mientras transcurre el tiempo se van contando las pulsaciones en el puerto RB2. Una vez finaliza el tiempo de 2,4 segundos (Time_out) se deshabilitan las interrupciones y se calcula la velocidad del viento sabiendo que cada pulsación corresponde a 1Km/h. Se resetean las variables de número de pulsos y de finalización de tiempo (Time_out) y se envían los datos.

2.4.5. Simulación del diseño en Proteus

Para la simulación de la veleta se ha creado un circuito que simula el esquema eléctrico de la veleta que nos indica el fabricante. Se ha creado un divisor de tensión con interruptores y resistencias (Véase Fig. 2.41) donde se puede seleccionar la dirección del viento cerrando o abriendo los interruptores. El divisor resistivo lo conectamos a pin analógico AN6 (RF1).

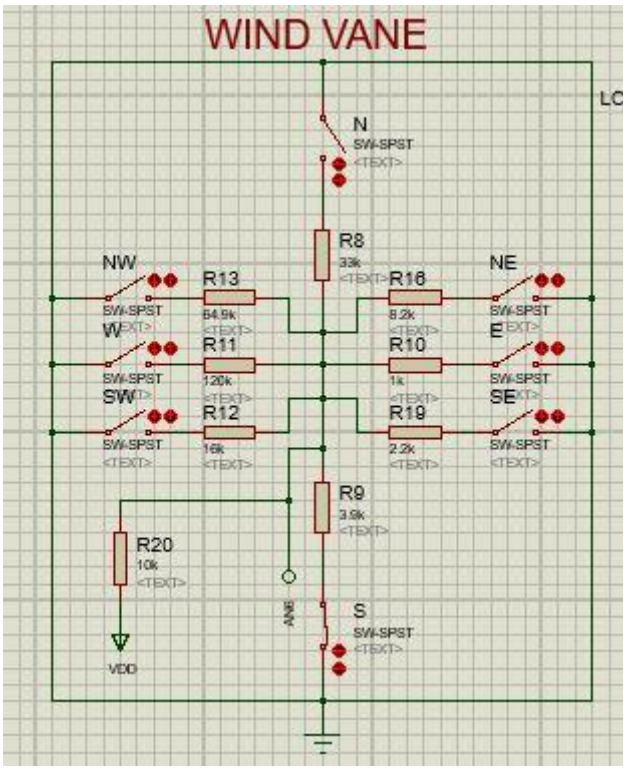


Fig. 2.41 Esquema de la veleta en Proteus

Para seleccionar una de las posiciones intermedias solo será necesario cerrar los dos interruptores contiguos. Para obtener una de las posiciones intermedias como NNE, cerramos los interruptores N y NE obteniendo como resultado una resistencia en paralelo de R8 y R16.

El anemómetro se ha simulado como un pulsador. Se ha conectado un pulsador con una resistencia de Pull-Up de 4.7K al pin RB2 (INT2) del microcontrolador. También se ha añadido un generador de pulsos con el que introducimos una frecuencia de pulsos estable para calcular con mayor precisión la velocidad del viento.

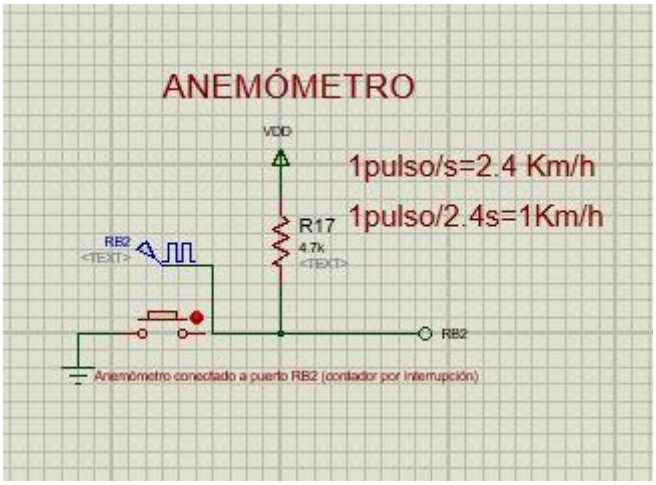


Fig. 2.42 Esquema del anemómetro en Proteus

En la Fig. 2.43 se puede ver el funcionamiento del sistema. Se ha introducido una frecuencia de 4.16Hz (10 pulsos / 2.4 segundos) en el generador de pulsos para obtener una velocidad de viento constante de 10 Km/h.

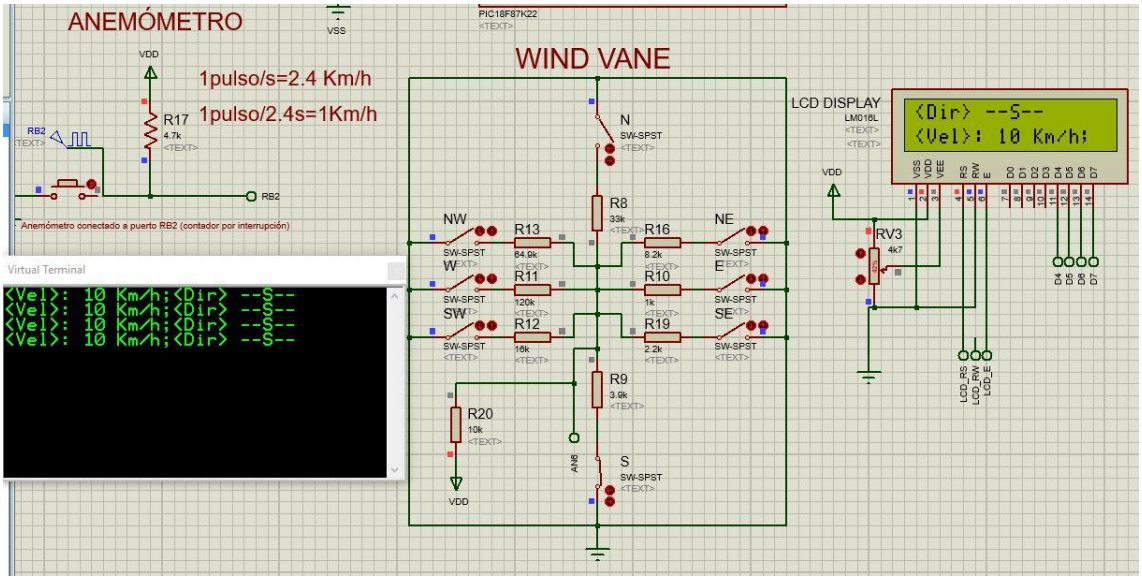


Fig. 2.43 Simulación de la lectura de dirección y velocidad del viento

2.4.6. Prototipo sobre la placa de entrenamiento DM240313

Para realizar la simulación en la placa de entrenamiento, utilizamos los pines RF1 para la veleta junto con una resistencia de 10k para formar el divisor de tensión y RB2 con una resistencia de Pull-Up de 4.7K para el anemómetro.

En la Fig. 2.44 se puede ver el funcionamiento de la veleta junto al anemómetro y los datos de velocidad y dirección del viento.

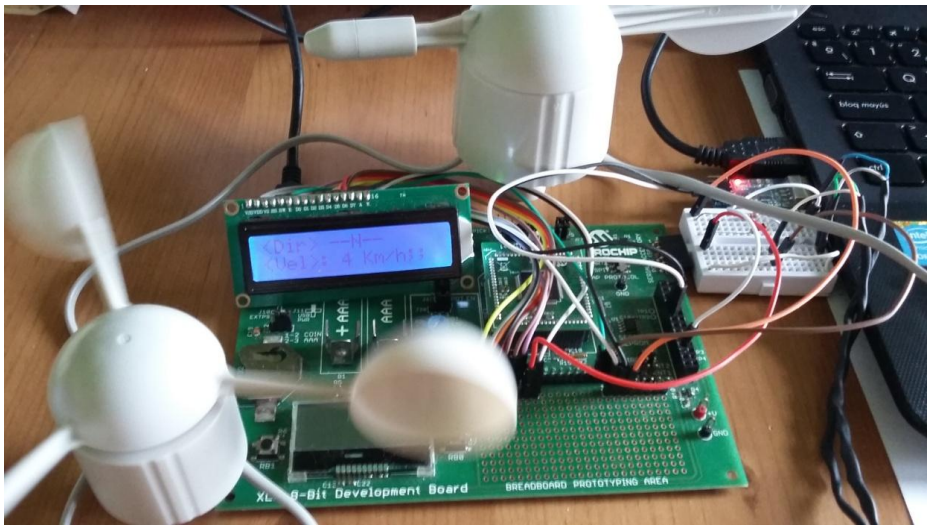


Fig. 2.44 Prueba de la veleta y anemómetro sobre la placa DM240313

Y la recepción de los datos en el HyperTerminal de Windows.

Fig. 2.45 Recepción de datos de dirección y velocidad del viento

CAPÍTULO 3. COMUNICACIÓN SIN HILOS, MODO SLEEP, MONITORIZACIÓN REMOTA Y MEDIDAS DE CONSUMO

3.1. Proyecto completo, modo Sleep y comunicación sin hilos

La última fase será unir todos los proyectos anteriores en uno común, añadirle un módulo bluetooth para la comunicación inalámbrica, implementar el modo Sleep del microcontrolador para reducir el consumo de energía del sistema y realizar el sistema de control y monitorización con LabView.

3.1.1. Diagrama de bloques y diseño del circuito

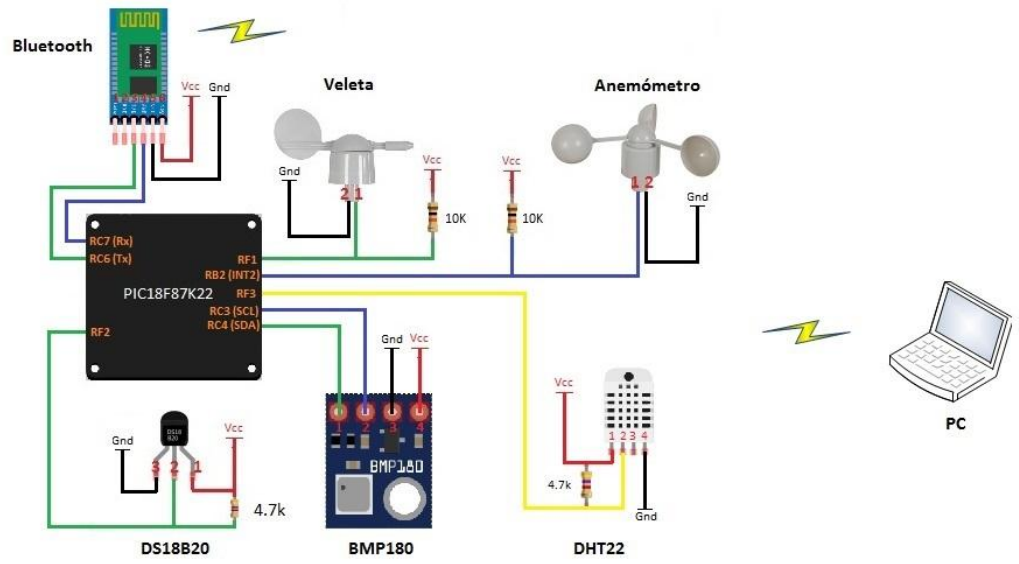


Fig. 3.1 Esquema de conexión del proyecto completo

Para el proyecto final se han incorporado todos los sensores, se ha implementado la conexión inalámbrica mediante el módulo bluetooth HC-05 y se ha eliminado el pulsador RB0 ya que ahora el control del sistema será remoto.

3.1.2. Diseño del programa con estilo FSM

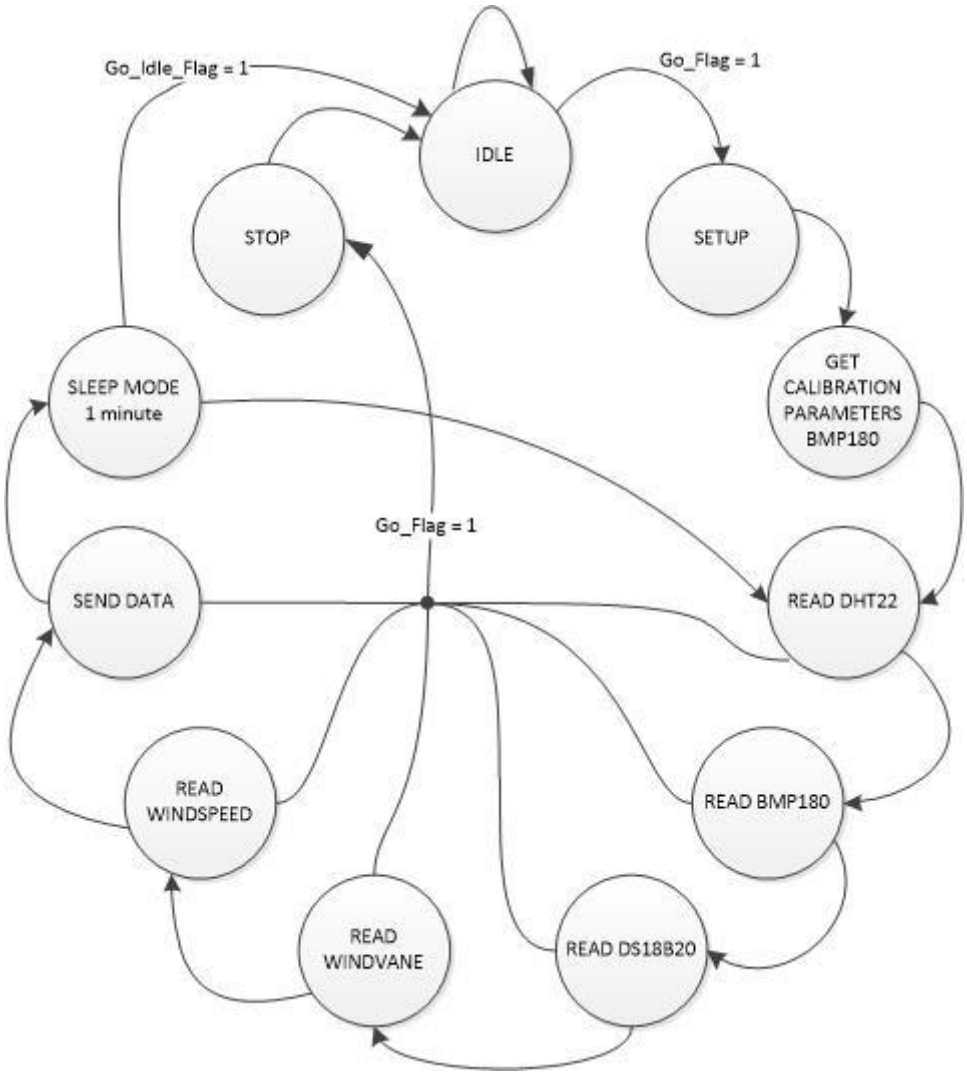


Fig. 3.2 Diagrama FSM del proyecto completo

El proyecto final consta de los siguientes estados que se detallan a continuación:

- Idle: Es el modo reposo, los sensores están apagados, se activa el módulo USART para poder recibir los comandos de control.
- Setup: Se configura el módulo I2C y se ponen en alto los puertos de los sensores DHT22 y DS18B20.
- Get Calibration Parameters BMP180: Se leen los parámetros de calibración del sensor BMP180.
- Read DHT22: Se lee la humedad y temperatura ambiental.
- Read BMP180: Se lee la presión atmosférica.

- Read DS18B20: Se lee la temperatura del agua.
- Read Windvane: Se lee la dirección del viento.
- Read Windspeed: Se lee la velocidad del viento.
- Send data: Se juntan los datos y se envían al PC de forma inalámbrica
- Sleep mode: El microporcesador entra en modo de bajo consumo
- Stop: Se detiene el módulo USART y se ponen a nivel bajo los pins de los sensores DHT22 y DS18B20

El control del sistema ahora pasa a ser remoto. Enviando diferentes caracteres por el terminal podremos iniciar el sistema y detenerlo. Enviando el carácter '3' por el terminal se activa la bandera 'Go_flag' que pone en funcionamiento el sistema. Si enviamos un '1' mientras está en modo Sleep se activa la bandera 'Go_Idle_Flag' y el sistema vuelve al estado Idle.

Se han conservado todas las funciones de los diferentes sensores, solamente se ha creado una nueva función que configura el WDT y hace entrar en modo Sleep al microporcesador.

En la Fig. 3.3 se puede ver el diagrama de flujo para la función creada para el estado 'Sleep mode'.

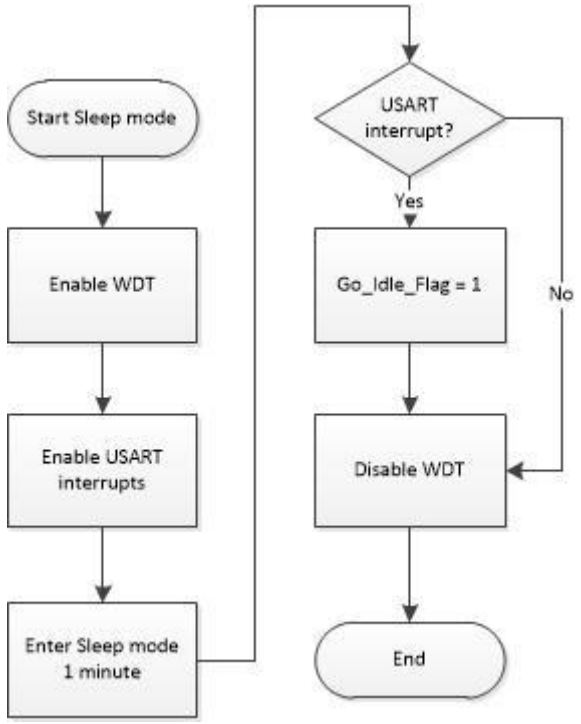


Fig. 3.3 Diagrama de flujo del modo Sleep

Para entrar en el modo Sleep, primero se habilita el WDT con un tiempo de 1 minuto. Se activan las interrupciones USART para el modo Sleep. Se ejecuta la instrucción Sleep durante 1 minuto, si durante ese tiempo se recibe una interrupción de módulo USART se sale del modo Sleep, se activa la bandera 'Go_Idle_flag' para volver al modo 'Idle' y se inhabilita el WDT. En caso

contrario, cuando se agota el tiempo del WDT se sale del modo Sleep, se inhabilita el WDT y se sigue con la ejecución del programa leyendo de nuevo los sensores.

3.1.3. Comunicación sin hilos: El módulo bluetooth HC-05

Para implementar la conexión inalámbrica, vamos a utilizar el módulo bluetooth HC-05 (Véase Fig. 3.4). Se trata de un módulo que convierte las señales TTL de módulo USART a señales inalámbricas bluetooth.

Fig. 3.4 Módulo Bluetooth HC-05

La conexión es totalmente transparente, ya que el PC tratará la conexión como si de un puerto COM se tratase.

Características Bluetooth HC-05:

- Compatible con el protocolo Bluetooth V2.0
- Voltaje de alimentación: 3.3VDC – 6VDC
- Voltaje de operación: 3.3VDC
- Corriente de operación: < 40 mA
- Frecuencia: 2.4GHz banda ISM
- Alcance: Aproximado de 10 m
- Modulación: GFSK (Gaussian Frequency Shift Keying)
- Seguridad: Autentificación y encriptación
- Modo operación: Maestro-Esclavo
- Baud rate ajustable: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- Configurable mediante comandos AT
- Configuración por defecto para el puerto COM: 9600, N, 8,1, Modo esclavo
- Password: 1234

La conexión es sencilla, solamente hay que conectar los pines Rx y Tx de forma cruzada al puerto USART del microcontrolador como en el caso del convertidor USB-USART utilizado hasta ahora para la conexión.

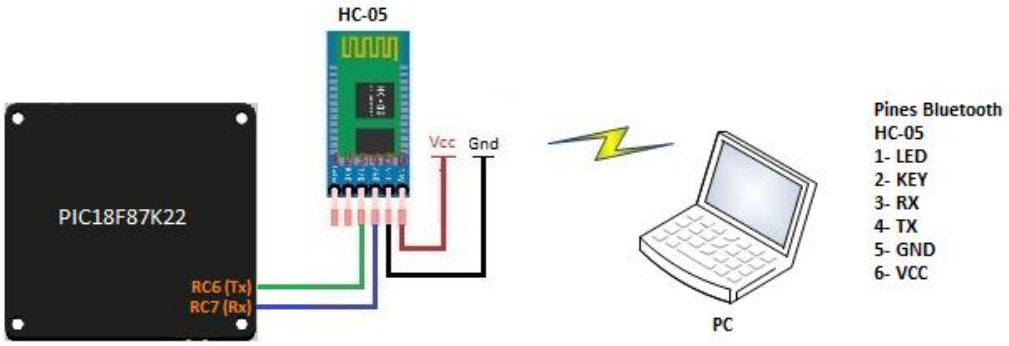


Fig. 3.5 Esquema de conexión del módulo Bluetooth HC-05

Una vez instalado, hay que vincularlo al PC. Para ello agregaremos el módulo como un nuevo dispositivo en Windows. Una vez agregado y configurado los parámetros de transmisión igual que en el módulo USART, el módulo bluetooth, permanece en espera hasta que lo vinculemos. En este modo el led del módulo parpadea continuamente. En cambio, cuando el módulo está vinculado, el led parpadea cada 2 segundos aproximadamente.

3.1.4. Reducción del consumo de energía: El modo Sleep

En el modo Sleep, el microcontrolador entra en un estado de bajo consumo. Cuando entramos en este modo, los valores almacenados en los registros de memoria no se borran y el programa se detiene en la última instrucción realizada. Cuando un evento hace despertar al microcontrolador continúa la ejecución del programa con la instrucción siguiente a la ejecución del modo Sleep.

Hay diferentes eventos que producen el despertar del microcontrolador del modo Sleep.

- Interrupción: Si se produce una interrupción externa o producida por algún periférico del microcontrolador.
- Reset: Si se produce un reset del microcontrolador. En este caso el programa se ejecuta desde el inicio y no desde la última instrucción.
- Desbordamiento perro guardián (WDT): Si el periodo de tiempo programado del WDT finaliza.

Configuración del modo Sleep:

El microcontrolador tiene dos modos de funcionamiento de bajo consumo, el modo ‘Idle’ y el modo ‘Sleep’. En el modo ‘Idle’ el oscilador principal está desactivado pero los periféricos siguen en funcionamiento, y en el modo ‘Sleep’ se detiene el oscilador principal y los periféricos.

Para seleccionar el modo ‘Sleep’ es necesario cambiar el bit ‘IDLEN’ en el registro ‘OSCCON’. Por defecto está configurado a ‘1’ (modo ‘Idle’ activado). Para seleccionar el modo ‘Sleep’ hay que ponerlo a ‘0’.

REGISTER 3-1: OSCCON: OSCILLATOR CONTROL REGISTER(1)

R/W-0	R/W-1	R/W-1	R/W-0	R(1)	R-0	R/W-0	R/W-0
IDLEN	IRCF2(2)	IRCF1(2)	IRCF0(2)	OSTS	HFIOFS	SCS1(4)	SCS0(4)
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 7 **IDLEN:** Idle Enable bit
1 = Device enters an Idle mode when a SLEEP instruction is executed
0 = Device enters Sleep mode when a SLEEP instruction is executed

Fig. 3.6 Registro OSCON del PIC18f87K22

Para despertar el microcontrolador del modo Sleep, utilizaremos el desbordamiento de perro guardián (WDT).

La principal función del WDT es resetear el microcontrolador una vez ha transcurrido un tiempo predeterminado. Esto es útil cuando tenemos un fallo de software o hardware que bloquea el micro e impide el correcto funcionamiento del sistema. Para cancelar el reset hay que ejecutar la instrucción CLRWDT antes de que finalice el período asignado, de lo contrario se producirá el reset del micro y la ejecución del programa se reiniciará sin completarse.

Otra de las características que nos ofrece el WDT es la posibilidad de crear una interrupción que despierte al microcontrolador del modo sleep y el programa vuelva a ejecutarse desde el siguiente punto donde se ejecuto la función Sleep sin producir un reset.

El WDT obtiene su base de tiempo del oscilador LF-INTOSC de 31.25Khz. Tiene una base de tiempo mínima de 4ms que mediante un postscaler de 16 bits podemos conseguir un tiempo de hasta 70 minutos.

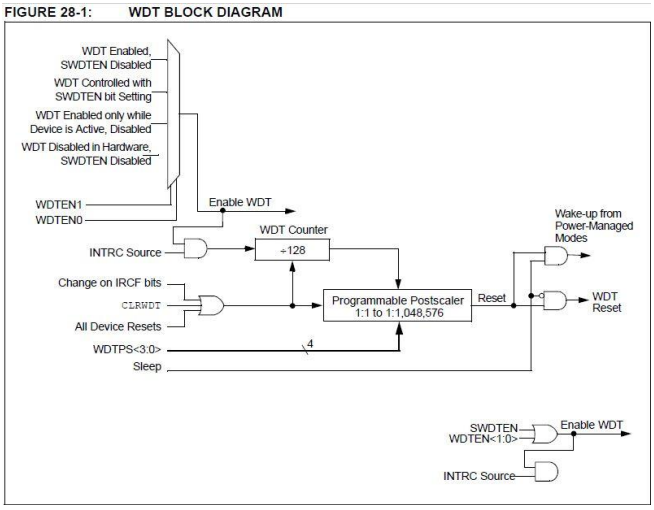


Fig. 3.7 Diagrama de bloques del WDT

Para obtener el tiempo del WDT utilizamos la fórmula: tiempo WDT = (128/31Khz) * Postscaler

Para 1 minuto el postscaler será 1:16384, entonces 1 min= (128/31Khz) * 16384

El WDT tiene 4 modos de funcionamiento:

- Apagado
- Encendido
- Encendido en modo Activo y desconectado en modo Sleep
- Encendido y controlado por software por bit control SWDTEN

Para utilizar el WDT como reloj para despertar al microcontrolador del modo Sleep tenemos que utilizar el último modo de funcionamiento, el modo controlado por software mediante el bit de control SWDTEN nos permite poner en marcha y detener el WDT en el lugar que más nos interese de nuestro código.

Para habilitarlo tenemos que seleccionar la siguiente configuración en el registro CONFIG2H

300003	CONFIG2H 3A	WDTEN	ON	Watchdog Timer	WDT controlled by SWDTEN bit setting
		WDTPS	16384	Watchdog Postscaler	1:16384

Fig. 3.8 Registro CONFIG2H del PIC18F87K22

WDTEN: ON. WDT controlado por bit de control SWDTEN
WDTPS: 16384. Postscaler del WDT 16384. Aproximadamente 1 minuto de tiempo

3.1.5. Prototipo sobre la placa de entrenamiento DM240313

En la Fig. 3.9 se puede ver el montaje de todos los sensores y el módulo bluetooth sobre la placa de entrenamiento DM240313.

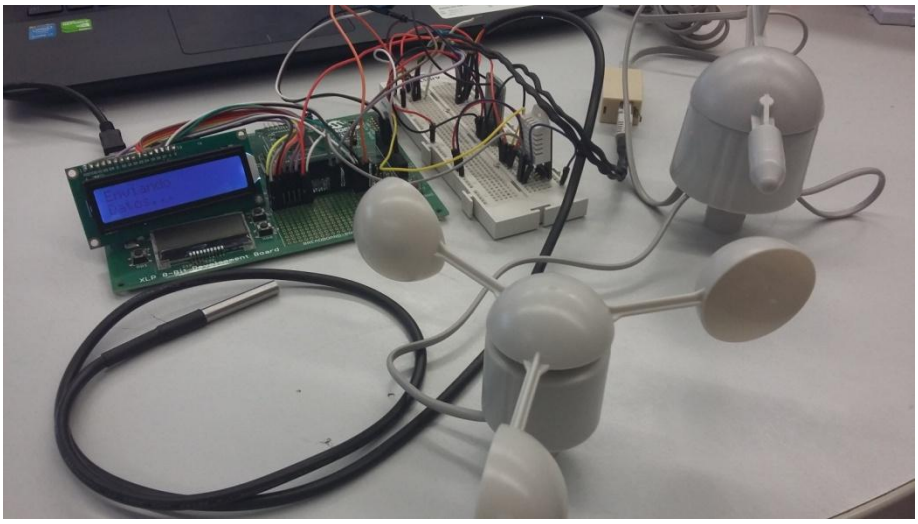


Fig. 3.9 Prueba proyecto completo sobre la placa DM240313

Y la recepción de los datos de los sensores en el HyperTerminal de Windows.

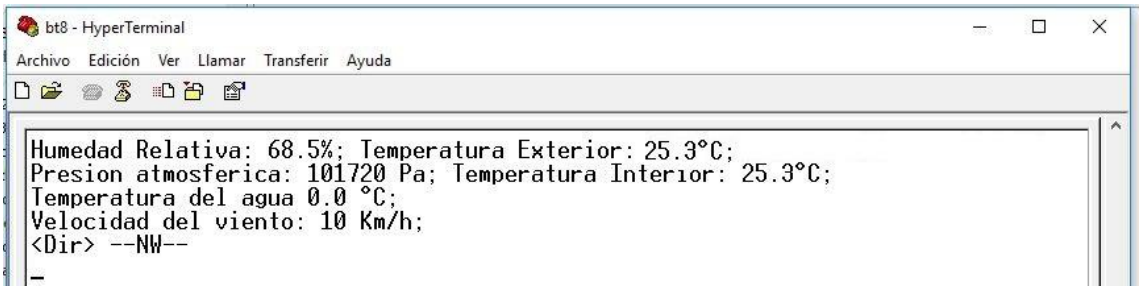


Fig. 3.10 Recepción de datos de los sensores

3.2. Prototipo PCB del proyecto completo

El siguiente paso en el proyecto, es crear una placa PCB con únicamente los componentes necesarios. Para ello, hemos utilizado el Software de edición Eagle.

3.2.1. Esquemático con EAGLE

El editor Eagle consta de dos ventanas de edición. En la primera ventana, la llamada ‘Schematic’ se selecciona los componentes electrónicos y se crean las conexiones lógicas entre ellos. En la segunda ventana ‘Board’, se edita las conexiones a nivel físico entre los componentes cambiando la forma y dirección de las pistas que unen los componentes y la posición de estos en la placa PCB.

El prototipo final de la placa cuenta son las siguientes características:

- Pines de conexión para la batería y la placa solar con diodo schottky de protección
- Microchip PIC18f87k22
- Regulador de tensión MCP170330MB 3.3V
- Pines de conexión para el módulo bluetooth HC-05
- Pines de conexión para los sensores DS18B20, DHT22, BMP180, Anemómetro y Veleta
- Pines de conexión para el programador PicKit3
- Botón de Reset
- 2 leds de control con jumper de conexión
- Led de alimentación con jumper de conexión
- Jumper para la medición del consumo de la placa
- Jumper para la conexión de resistencias de Pull-Up I2C

En la Fig. 3.11 se muestra el esquemático con las conexiones lógicas de los componentes.

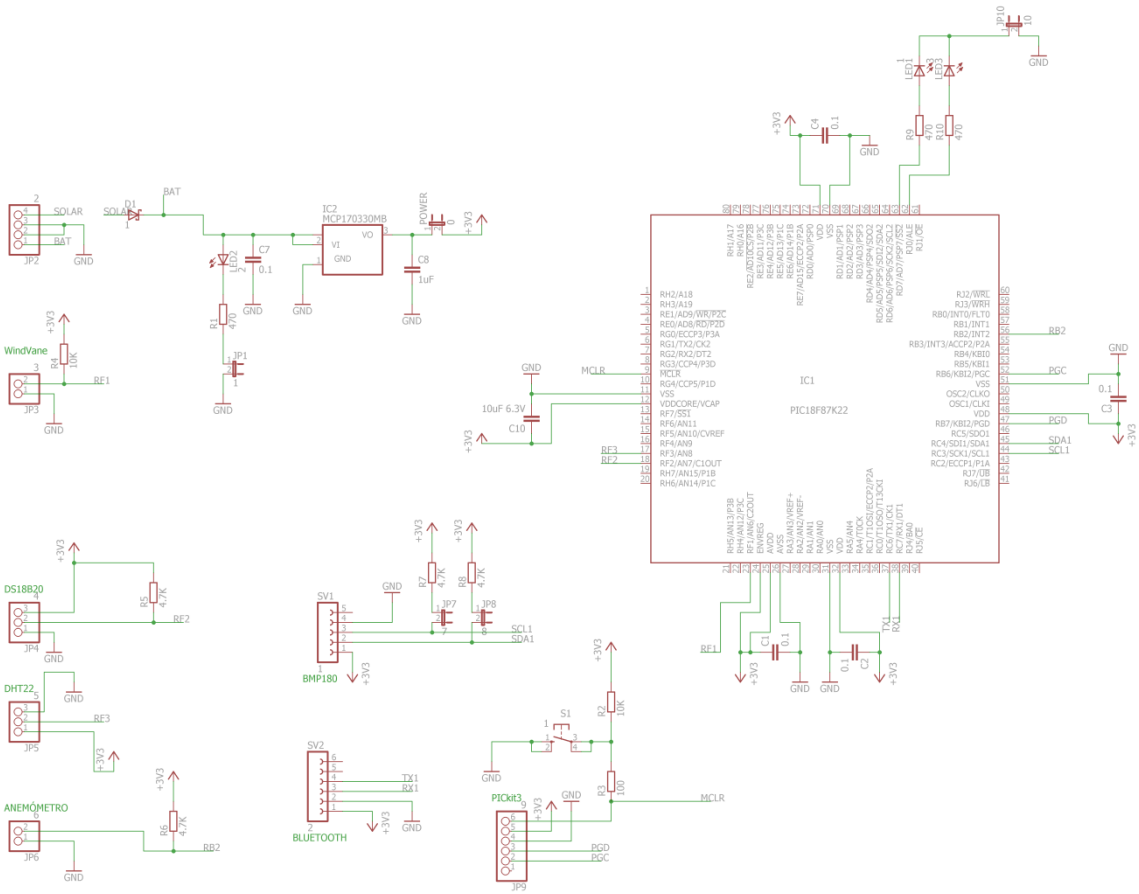


Fig. 3.11 Esquemático conexiones lógicas Eagle

Una vez seleccionados todos los componentes necesarios y creadas las conexiones lógicas, en la ventana 'Board', editamos la posición de los componentes y la forma y anchura de las pistas de conexión.

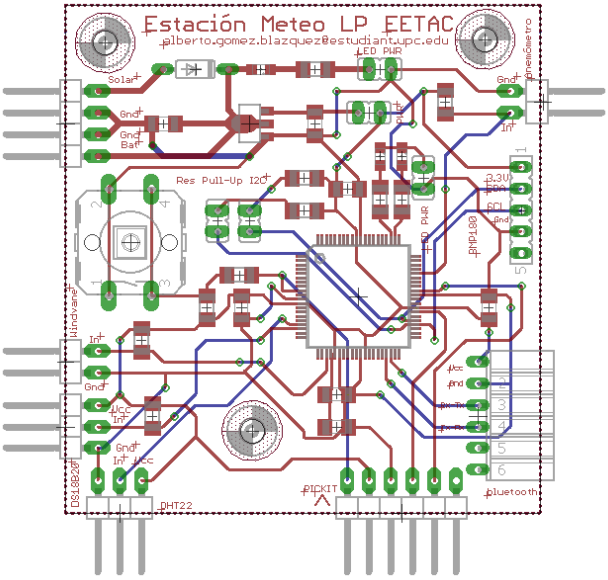


Fig. 3.12 Top y Bottom placa PCB

3.2.2. Montaje

Una vez editada la placa y comprobado las conexiones, se ha creado la placa y se han soldados los componentes electrónicos.

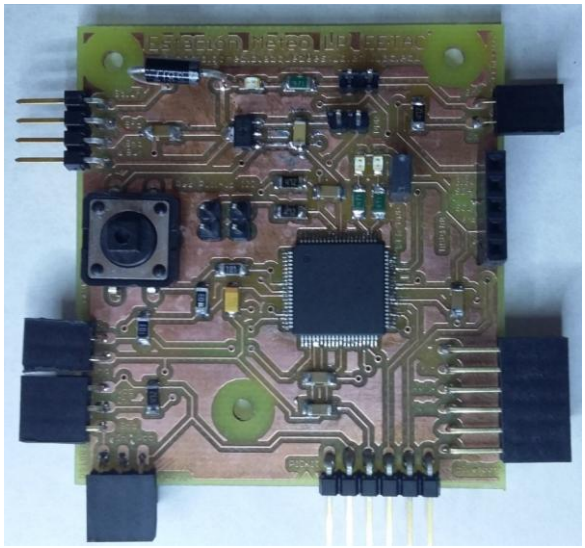


Fig. 3.13 Placa PCB Estación meteorológica

En la Tabla 3.1 se hace un resumen de los componentes empleados y la cantidad para la fabricación de la PCB

Tabla 3.1 Resumen componentes PCB

Componente	Valor	Cantidad
PIC18F87K22 TQFP80	-	1

DIODEDO35-7	-	2
Regulador MCP170330MB	-	1
Switch B3F-40XX	-	1
PINHD-1X4/90	-	1
PINHD-1X2/90	-	2
PINHD-1X3/90	-	2
PINHD-1X6/90	-	1
Jumper	-	5
LED Rojo SML0805	-	1
LED Verde SML0805	-	2
Resistencias R1206	10K	2
Resistencias R1206	4.7K	4
Resistencias R1206	470	3
Resistencias R1206	100	1
Condensador 1206	1uF	1
Condensador 1206	10uF 6.3V	1
Condensador 1206	100nF	5

3.2.3. Instalación de la placa en la estación meteorológica

Una vez finalizado el proceso de fabricación y soldado de la placa PCB, se ha instalado en la estación meteorológica junto con los sensores, la batería y la placa solar.

En la Fig. 3.14se puede ver el montaje final en funcionamiento.

Fig. 3.14 Estación meteorológica

3.3. Interfaz gráfica de monitorización con LabView

El último paso es crear una interfaz gráfica con LabView de National Instruments. Se trata de una plataforma de desarrollo destinado al diseño de sistemas de monitorización, aplicaciones de control industrial o procesamiento digital de señales. Emplea la programación gráfica o lenguaje G para la creación de programas basados en diagramas de bloques.

3.3.1. Panel frontal

El panel frontal consta de las siguientes características:

- Panel de control para la configuración de la conexión con la estación meteorológica, fecha y hora, botonera de control y guardado de datos en tabla Exel en intervalos desde 1 minuto hasta 60 minutos
- Indicadores verticales de temperatura interior, exterior, temperatura del agua con escala de -40ºC a 50ºC y resolución de 0,1ºC
- Indicador vertical de humedad relativa con escala de 0 a 100% y resolución de 0.1 %
- Indicador esférico de presión atmosférica con escala de 960 hPa a 1070 hPa con una resolución de 0.01 hPa
- Rosa de los vientos (dirección del viento) de 16 posiciones e indicador de nombre del viento
- Indicador lineal de velocidad del viento en Km/h con escala de 1 a 120 Km/h y una resolución de 1 Km/h
- Indicador digital del viento en Km/h y Nudos
- Panel de alertas por temperatura mayor de 30ºC, heladas (temperatura inferior a 0ºC), viento (rachas superior a 35 Km/h) y lluvia (presión atmosférica inferior a 980 hPa y humedad superior a 75%)
- Panel indicador de escala de viento Beaufort
- Gráfica interactiva de temperatura interior, exterior, agua y humedad

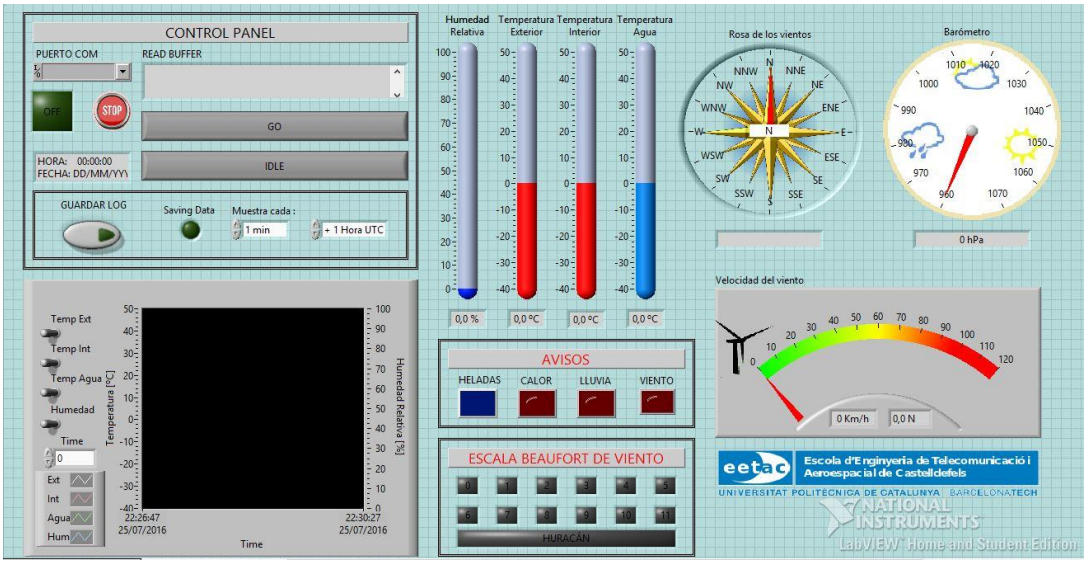


Fig. 3.15 Panel frontal de monitorización y control con LabView

3.4. Medidas de consumo

Una vez terminado el montaje del sistema, se han realizado medidas de consumo del sistema completo para comprobar el consumo en los diferentes modos de funcionamiento y poder calcular la duración de la batería que alimenta el sistema.

Hay tres etapas de funcionamiento en la estación meteorológica. La primera es el estado Idle donde no se realizan mediciones de los sensores pero están alimentados y en modo reposo. El módulo bluetooth también está conectado. Durante los primeros instantes el módulo Bluetooth estará desvinculado del PC elevando el consumo del sistema, una vez vinculado, el consumo se reduce.

La segunda etapa es la de medición y envío de datos mediante bluetooth. En esta etapa los sensores y el módulo bluetooth entran en modo activo elevando el consumo del sistema, sin embargo esta es la etapa más corta.

Por último, tenemos la etapa de modo de bajo consumo del microcontrolador. En esta etapa el microcontrolador entra en modo Sleep durante 1 minuto bajado el consumo del sistema, pero se mantiene la alimentación de los sensores y del módulo bluetooth, ya que si se retira la alimentación al módulo bluetooth, este se desvincula del PC y se produce un fallo en el envío de los datos. Este aspecto que no se ha podido solventar, lo que produce que el consumo de energía en esta etapa no se haya podido reducir al máximo, ya que el módulo bluetooth es el que más energía requiere en todo el sistema.

En la Tabla 3.2 se muestra el consumo aproximado del sistema medido en las tres etapas

Tabla 3.2 Medidas de consumo de la estación meteorológica

Modo de funcionamiento	Consumo medio	Tiempo de funcionamiento
Modo Idle (Bluetooth desvinculado)	46 mA	6 Segundos
Modo IDLE (Bluetooth vinculado)	10 mA	4 Segundos
Medición y envío	24 mA	Aproximadamente 3.5 segundos
Modo Sleep	6 mA	Aproximadamente 65 Segundos

En el tiempo de medición y envío se ha tenido en cuenta los tiempos más grandes de medición, que son los del anemómetro (2.4 segundos) y el tiempo de conversión del DS18B20 (1 segundo), ya que el tiempo de medición de los otros sensores es casi despreciable comparado con estos.

Cuando iniciemos el sistema tendremos el siguiente diagrama de consumo.

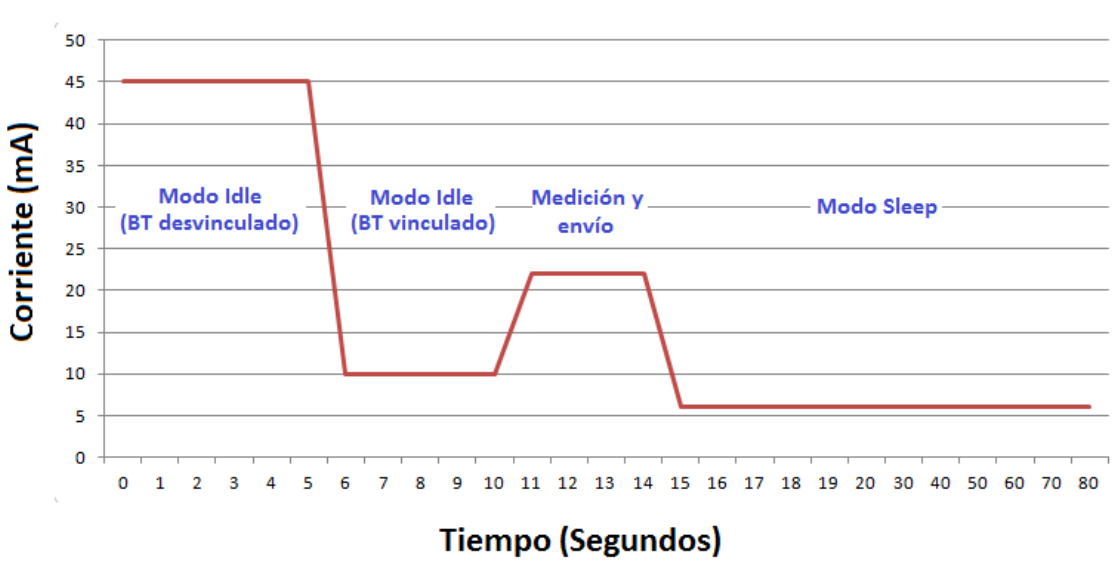


Fig. 3.16 Diagrama de consumo

Una vez vinculado el módulo Bluetooth e iniciado el sistema, se repetirán únicamente las etapas de medición y envío y modo Sleep. Por lo tanto para estimar el consumo de batería por parte del sistema solamente se han tenido en cuenta estas dos últimas.

En la siguiente ecuación se calcula la corriente media consumida en un ciclo de trabajo, que incluye los 3,5 segundos (5.1%) de medición y envío y los 65 segundos (94.9%) de modo Sleep.

$$Consumo\ medio = 0.051 * 22mA + 0.949 * 6mA = 6.81\ mA \qquad \textbf{(3.2)}$$

La batería escogida para alimentar el sistema, es una batería NiMH de 3.7V y 700 mAh de capacidad.

Sabiendo el consumo medio del sistema y la capacidad de la batería podemos estimar la duración de esta.

$$Duración\ de\ la\ batería\ (h) = \frac{700\ mAh}{6.81\ mA} = 102.8\ h = 4\ días\ y\ 6\ horas \qquad \textbf{(3.1)}$$

El sistema podría funcionar más de 102 horas sin interrupción. Sin embargo como se trata de un sistema autónomo se ha incluido una placa solar (Véase **¡Error! No se encuentra el origen de la referencia.**) para la alimentación del istema y recarga de la batería.

Se ha escogido una batería de 700 mAh teniendo en cuenta que pueden producirse largos periodos sin suficiente luz solar, que los días de invierno son más cortos o varios días seguidos de lluvia. De esta manera nos aseguramos un mínimo de casi 4 días en que la estación meteorológica podría funcionar normalmente a pesar de las condiciones adversas.

Fig. 3.17 Placa solar

La palca solar tiene las siguientes características:

- Dimensiones: 100x80x2.5(±0.2) mm
- Voltaje típico: 5.5V
- Corriente típica: 170mA
- Voltaje en circuito abierto: 8.2 V
- Máximo voltaje con carga: 6.4V

La placa solar tiene un emplazamiento fijo en la estación meteorológica. Se ha escogido la inclinación más óptima para estos casos. Cuando se trata de instalaciones fijas, lo más habitual es orientar la placa solar hacia el Sur y con una inclinación β igual a la latitud en la que nos encontramos. En Barcelona, nos encontramos a una latitud de unos 41º.
Para el sistema de alimentación, se ha optado por un diseño simple (Véase Fig. 3.18) pero valido para una primera aproximación de lo que sería un sistema de Energy Harvesting.

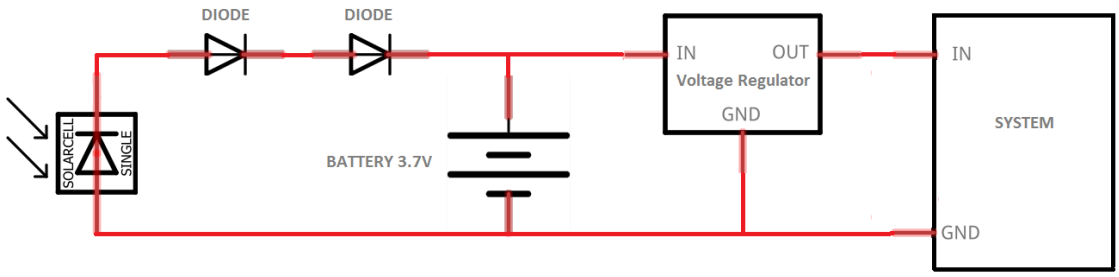


Fig. 3.18 Esquema sistema alimentación

Para evitar que la placa solar trabaje como carga cuando no haya suficiente luz solar, instalaremos 2 diodos de protección entre la batería y la placa solar. Con esto también conseguimos que cuando la luz solar es máxima, la tensión de la placa solar no dañe la batería por una sobretensión igualando las tensiones de la batería y la placa solar.

CAPÍTULO 4. CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO

El objetivo de este trabajo era la realización de un sistema de sensores autónomo, sin hilos y de bajo consumo. Se ha escogido crear una estación meteorológica, ya que agrupa todos los requisitos iniciales del proyecto.

Una vez finalizado el proyecto, podemos decir que se han alcanzado todos los requerimientos iniciales. Se ha conseguido realizar un sistema autónomo, que necesita un mínimo de mantenimiento una vez puesto en marcha.

Se ha comprobado que el sistema de alimentación propuesto, es suficiente para mantener en funcionamiento la estación meteorológica, incluso en periodos de poca luz.

Implementando un sistema inalámbrico bluetooth para la monitorización y control de la estación remotamente, se ha conseguido eliminar los cables de comunicación permitiendo una mayor movilidad y fácil instalación de la estación meteorológica.

Por último, también se ha podido demostrar que los modos de bajo consumo del microprocesador, logran reducir el consumo de energía del sistema y ayudan a prolongar la duración de las baterías.

4.1. Líneas futuras de trabajo

Una vez finalizado el proyecto y puesto en funcionamiento, se han detectado algunos puntos en los que es posible introducir mejoras que aumenten aún más la eficiencia del sistema. A continuación se detallan algunos de los problemas con los que nos hemos encontrado y propuestas para solucionarlos.

- Sistema de comunicación inalámbrico:

Se ha escogido el sistema de comunicación bluetooth por su facilidad de integración y uso que presenta frente a otros sistemas más complejos. A la hora de implementarlo, nos han surgido varios problemas que impiden reducir el consumo del sistema. El principal problema ha sido la vinculación con el PC. Una vez vinculado el módulo bluetooth con el PC, este ya no se puede apagar, ya que se pierde la vinculación y se detiene el envío de datos.

Una de las posibles soluciones, sería implementar un sistema de comunicación inalámbrico de bajo consumo como ZigBee.

- Control de la alimentación de los sensores mediante transistores:

En esta versión del proyecto, los sensores van conectados directamente a la alimentación, sin posibilidad de desconectarlos cuando el sistema está en modo Sleep.

Como solución, se propone el uso de transistores que permitan ser controlados mediante el microcontrolador para encender o apagar los sensores y así poder reducir el consumo de energía mientras no se utilizan.

- Aplicación móvil de control y monitorización:

Actualmente el sistema se monitoriza y controla mediante una aplicación creada con LabView. Esto implica el uso de un PC, lo que pude resultar molesto y pesado.

Se propone crear una aplicación Android para la recepción de datos y el control de la estación meteorológica.

- Sistema de alimentación:

La alimentación se realiza mediante un circuito simple que incluye una placa solar y una batería. Actualmente el sistema, no tiene ningún tipo de protección en caso de que la batería este totalmente cargada o por descarga excesiva.

Como solución se propone la instalación de un circuito de protección que impida la sobrecarga de la batería una vez esté totalmente cargada o la desconecte del sistema en caso de que la tensión disminuya demasiado.

Un ejemplo de circuito que cumple con estos requisitos es el LTC3105 de Linear Technology.

BIBLIOGRAFÍA

[1] Milan Verle, *PIC Microcontrollers – Programming in Basic,* MikroElektronika, Septiembre 2010

[2] Programación en XC8 URL: http://www.todopic.com.ar/foros/index.php?PHPSESSID=jc4mpf6fotrqq9iddllm1thp60&topic=40649.msg338220#msg338220

[3] MPLAB® C18 C Compiler Libraries URL: http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_C18_Libraries_51297f.pdf

[4] Studentcompanion Microcontroller Communicatio with I2C Bus URL: http://www.studentcompanion.co.za/pic-microcontroller-communication-with-i%C2%B2c-bus-xc8/

[5] MicroPic USART con Interrupción URL: http://www.micropic.es/mpforo/index.php?topic=1466.0

[6] DigSys UPC CSD Chapter 3 Microcontrollers URL: http://digsys.upc.es/csd/index.html

[7] DigiKey XLP Deep Sleep Mode URL: http://dkc1.digikey.com/us/en/TOD/Microchip/DeepSleep/DeepSleep.html

[8] Unasguiasmas Wordpress Librería XLCD URL: https://unasguiasmas.wordpress.com/2014/03/20/01-libreria-xlcd-para-el-manejo-de-displays-lcd/

[9] Wikipedia Escala Beaufort URL: https://es.wikipedia.org/wiki/Escala_de_Beaufort

[10] Wikipedia Rosa de los vientos URL: https://ca.wikipedia.org/wiki/Rosa_dels_vents

[11] MicrocontroladoresPic Anemómetro URL: http://www.microcontroladorespic.com/proyectos-con-microcontroladores-pic/Anemometro/anemometro.html

[12] Universidad Tecnológica Metropolitana Protocolo 1-Wire URL: https://es.scribd.com/doc/24421918/Protocolo-1-Wire

[13] Datasheet PIC18F87K22 URL: http://ww1.microchip.com/downloads/en/DeviceDoc/39960d.pdf

[14] Datasheet Development Board DM240313 URL: http://ww1.microchip.com/downloads/en/DeviceDoc/41581A.pdf

[15] Source Code DM240313 URL: http://ww1.microchip.com/downloads/en/DeviceDoc/XLP%208-bit%20Source%20Code%20V1.1.zip

[16] Microchip LowPower Design Guide AN1416 URL: http://www.microchip.com/wwwAppNotes/AppNotes.aspx?appnote=en556618

[17] Microchip Developer Help WDT URL: http://microchip.wikidot.com/8bit:wdt

[18] Datasheet DHT22 URL: http://akizukidenshi.com/download/ds/aosong/AM2302.pdf

[19] Datasheet DS18B20 URL: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[20] Datasheet BMP180 URL: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMP180-DS000-121.pdf

[21] Dos Botones Cargador Solar URL: http://blog.dosbotones.com/2013/04/prototipo-de-cargador-solar-simple-diy.html

[22] Estaciones meteorológica PCE-Instruments URL: https://www.pce-instruments.com/espanol/instrumento-medida/medidor/estacion-meteorologica-kat_71062_1.htm

[23] Brikogeek Estación Meteorológica URL: http://tienda.bricogeek.com/sensores-humedad/672-estacion-meteorologica-con-mastil.html

[24] Astronomia Frentes, borrascas y anticiclones URL: http://www.astromia.com/tierraluna/frentes.htm

[25] Librerías Plib Microchip URL: http://www.microchip.com/mymicrochip/filehandler.aspx?ddocname=en574973

[26] Sensor de clima Argent Data System URL: https://www.sparkfun.com/datasheets/Sensors/Weather/Weather%20Sensor%20Assembly..pdf

[27] Linear Technology LTC3105 URL: http://www.linear.com/product/LTC3105

ANEXO A. Protocolo de comunicación I2C

El protocolo I2C (Inter-Integratet Circuit) es un bus de datos serial desarrollado por Philips en el año 1992. Se utilizaba principalmente para la interconexión entre diferentes partes de un circuito como CPU y dispositivos periféricos (Sensores, memorias, pantallas LCD,..).
I2C está diseñado como un bus maestro-esclavo. La comunicación se realiza a través de dos hilos, el hilo de datos (SDA) y el hilo del reloj de sincronización (SCL). La transferencia de datos siempre la inicia el maestro y es este el que genera la señal de reloj utilizada para la sincronización y el esclavo responde a las peticiones de este. Cada dispositivo dispone de una única dirección de 7 bits que lo distingue en el bus de comunicación de los otros dispositivos. El protocolo también permite tener más de un maestro en el mismo bus de comunicación (Multimaster-Mode). El modo multimaestro permite comunicarse a dos maestros entre ellos asignando el rol de esclavo a uno de ellos. Cuando está en curso una comunicación solo puede existir un maestro, los demás dispositivos actuarán como esclavos hasta el fin de la comunicación.

Características del bus I2C:

- Bus de comunicación síncrono
  - La comunicación es controlada por una señal de reloj común

  Bus formado por dos hilos
  - SDA (Serial Data Line): Bus de datos
  - SCL (Serial Clock Line): Reloj de sincronización
  - Es necesaria una referencia común de masa para todos los dispositivos que formen parte del mismo bus

  Velocidad de transmisión
  - Standard: Hasta 100 kbits/s
  - Fast: Hasta 400 Kbits/s
  - High-Speed: Hasta 3.4 Mbits/s

  Cada dispositivo del bus tiene una dirección única
  Máxima distancia de transmisión aproximada de 3 metros
  Protocolo de arbitraje
  Detección de colisiones
  Protocolo de acceso al bus:
  - Maestro - Esclavo
  - Multimaestro

Conexión física de dispositivos al bus:

- Todos los dispositivos se conectan al mismo bus
- Se necesita de dos líneas de señal: el reloj (SCL) y la línea de datos (SDA).
- Se requieren resistencias de Pull-Up en ambas líneas para mantener el nivel alto en el bus cuando ningún dispositivo accede a él o ningún dispositivo transmite un cero.

- El bus I2C trabaja con lógica positiva, un nivel alto en la línea de datos corresponde a un '1' lógico y un nivel bajo a un '0'.
- El nivel alto debe ser de al menos 0,7 x Vdd y el nivel bajo no puede exceder de 0,3 x Vdd

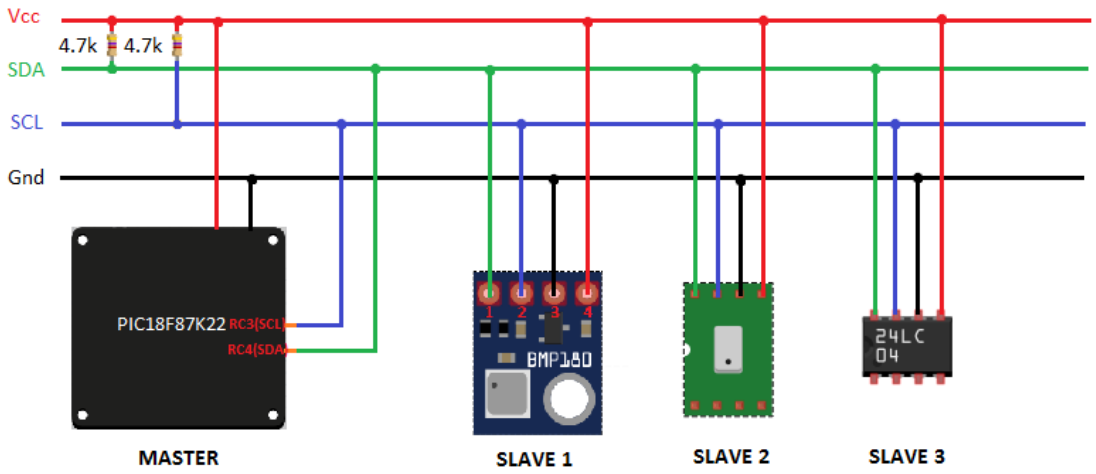


Fig. A.1 Esquema conexión dispositivos I2C

Protocolo de acceso al medio:

- El maestro controla la comunicación

    - Genera la señal de reloj (SCL)
    - Inicia y termina la comunicación
    - Establece el sentido de la comunicación

    Se requiere la confirmación de la recepción de cada byte de datos
    Los bits de datos van por la línea SDA
    Por cada bit de información es necesario un pulso de SCL
    Direccionamiento

    - Cada dispositivo tiene una dirección de 7 bits, el octavo bit (R/W) indica al esclavo si tiene que recibir datos del maestro (low,'0') o enviar datos al maestro (high,'1').
    - En el mismo bus se permiten 2^7 (128) dispositivos a la vez, aunque 16 direcciones están reservadas por lo que en la práctica podemos conectar 112 dispositivos simultáneamente.

El protocolo I2C tiene condiciones especiales de señal de bus. Para iniciar una comunicación el Master envía una señal de Start (S) y para pararla una señal de Stop (P). Cada byte de información requiere una respuesta de confirmación por parte del receptor. Cuando nadie accede al bus tanto SCL como SDA están a nivel alto.

Condiciones de Start y Stop:

Fig. A.2 Condiciones Start y Stop I2C

La condición de Start (S) la inicia el Master llevando la línea SDA a cero mientras SCL permanece en nivel alto. Mientras que para la condición de Stop (P) la línea SDA pasa a nivel alto mientras que la línea SCL permanece en alto.

Transferencia de datos:

Tras la condición de inicio el Master envía la dirección del esclavo (7 bits) junto con el comando de lectura/escritura (R=1/W=0). El esclavo responde con un ACK (acknowledge) si ha recibido la información. Justo después se inicia la transmisión de datos. La transmisión finaliza cuando el Master envía la condición de Stop.

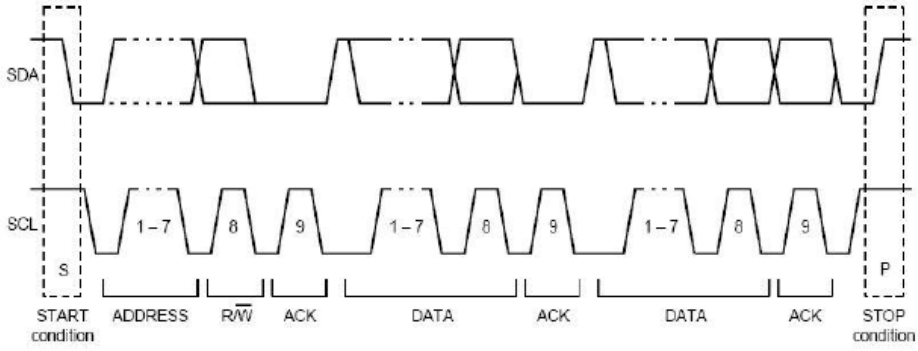
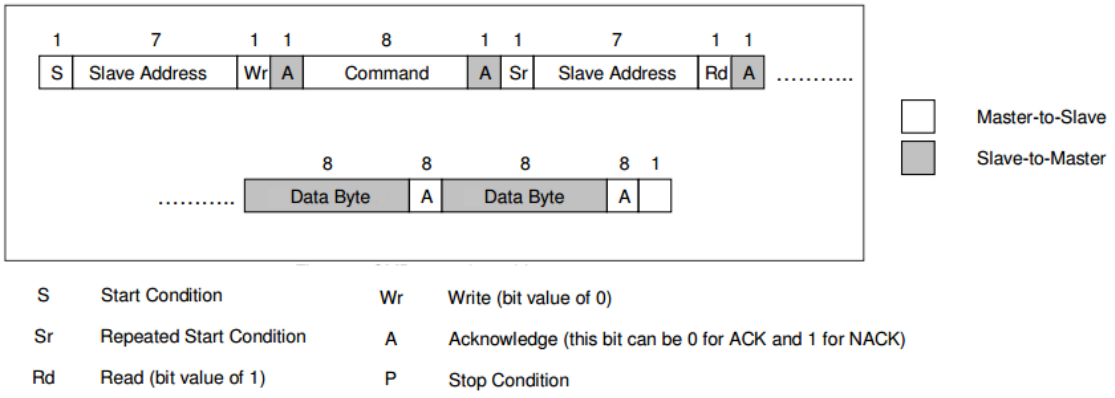


Fig. A.3 Transmisión de datos I2C

Existen diferentes formatos de transmisión:

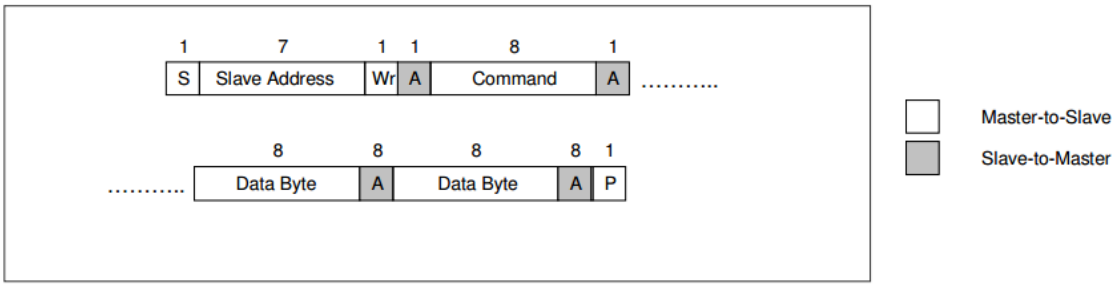
- Leer dato: El Master lee datos del esclavo



S	Start Condition	Wr	Write (bit value of 0)
Sr	Repeated Start Condition	A	Acknowledge (this bit can be 0 for ACK and 1 for NACK)
Rd	Read (bit value of 1)	P	Stop Condition

Fig. A.4 Lectura datos I2C

- Escribir dato: El Master escribe un dato en el esclavo



S	Start Condition	Wr	Write (bit value of 0)
Sr	Repeated Start Condition	A	Acknowledge (this bit can be 0 for ACK and 1 for NACK)
Rd	Read (bit value of 1)	P	Stop Condition

Fig. A.5 Escritura datos I2C

ANEXO B. Protocolo de comunicación 1-Wire

El protocolo 1-Wire es un protocolo diseñado por Dallas Semiconductor. Una red 1-wire está basada en un maestro y uno o varios dispositivos esclavos que utilizan la misma línea para comunicarse además de una tierra común para todos. Es necesaria una resistencia de Pull-Up en la línea de datos para mantener la línea en alto.

Características del bus 1-Wire:

- Bus de comunicación serial asíncrono
- Utiliza lógica CMOS/TTL con un rango de alimentación de 2.8V a 6V
- Maestro y esclavo pueden ser transmisor o receptor pero la transmisión solo puede ser en una dirección (half-duplex)
- Toda la información es leída o escrita empezando por el bit menos significativo (LSB)
- No se requiere una señal de reloj de sincronización ya que cada dispositivo se auto-sincroniza con el flanco de bajada del maestro
- Cada dispositivo dispone de una dirección única de 64 bits
- La velocidad de transmisión alrededor de 15Kbps en modo estándar de 111 Kbps en modo overdrive
- La alimentación de los esclavos se puede realizar a través del propio bus de datos. Este modo de alimentación se denomina modo parásito.

Operaciones del bus 1-Wire:

Hay cuatro operaciones básicas que se pueden realizar en el bus que son Reset, Escribir '0', Escribir '1' y Leer bit.

Operation	Description	Implementation
Reset	Reset the 1-Wire bus slave devices and get them ready for a command.	Drive bus low, delay 480 µs. Release bus, delay 70 µs. Sample bus: 0 = device(s) present, 1 = no device present Delay 410 µs.
Write 0 bit	Send '0' bit to the 1-Wire slaves (Write 0 slot time).	Drive bus low, delay 60 µs. Release bus, delay 10 µs.
Write 1 bit	Send '1' bit to the 1-Wire slaves (Write 1 slot time).	Drive bus low, delay 6 µs. Release bus, delay 64 µs.
Read bit	Read a bit from the 1-Wire slaves (Read time slot).	Drive bus low, delay 6 µs. Release bus, delay 9 µs. Sample bus to read bit from slave. Delay 55 µs.

Fig. B.1 Operaciones básicas protocolo 1-Wire

Para escribir un byte de datos combinaremos las operaciones de escribir '0' o '1'.

El diagrama de tiempo del protocolo 1-Wire, es similar a utilizar PWM (Pulse-Width Modulation) ya que para transmitir un '0' utiliza un pulso ancho y para transmitir un'1' un pulso estrecho durante el time slot.

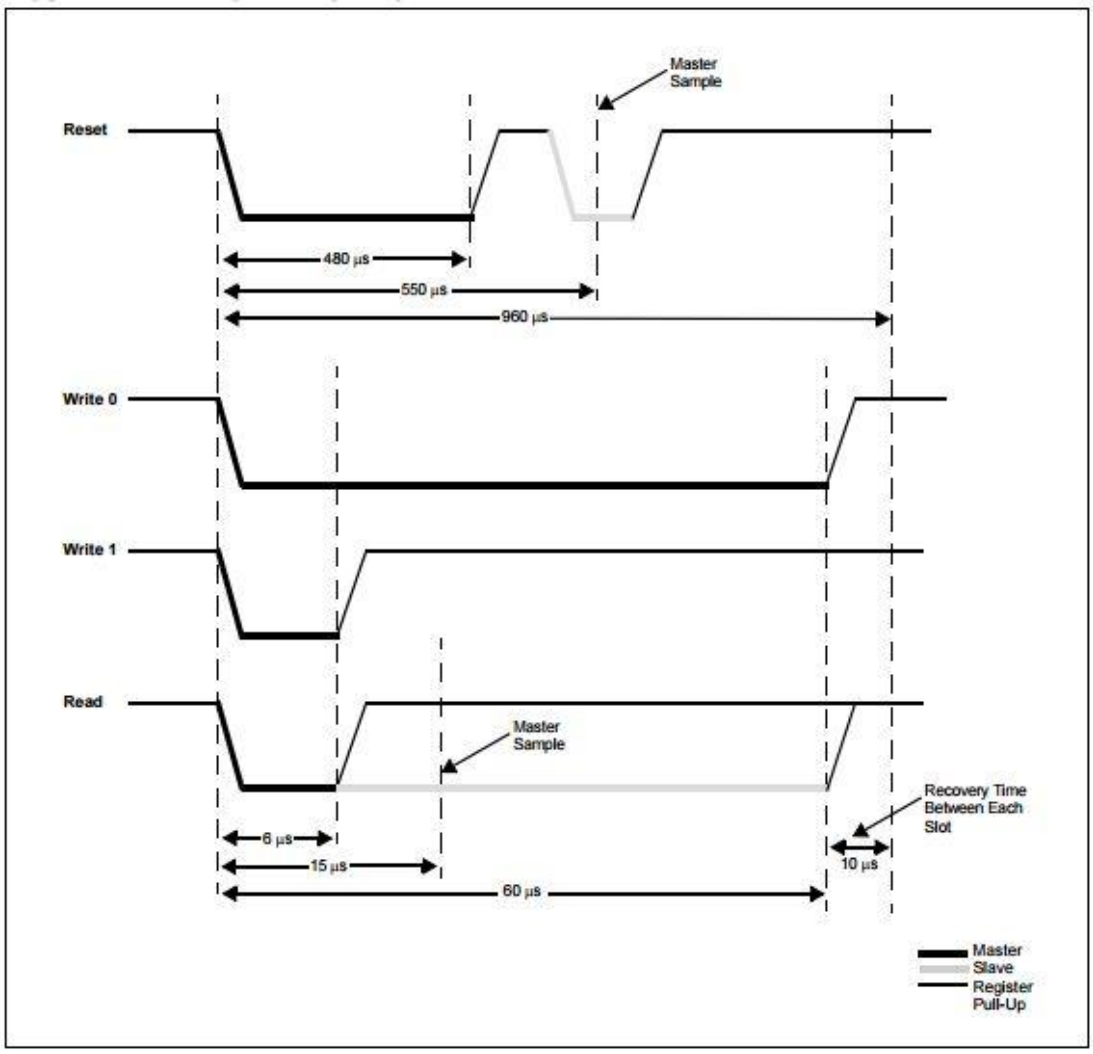


Fig. B.2 Diagrama de tiempo protocolo 1-Wire

La comunicación la inicia el maestro con la operación Reset para sincronizar el bus, en ese momento queda a la espera de que los esclavos lleven el bus a nivel alto indicando así que están disponibles para iniciar la comunicación con el maestro.

Para escribir un dato el maestro lleva el bus a nivel bajo durante 6 us y después dependiendo de si se quiere transmitir un '0' dejará el bus a nivel bajo hasta los 60 us o si quiere transmitir un '1' lo llevará a nivel alto hasta los 60 us. Para leer un dato el master lleva a nivel bajo durante 6 us el bus en ese momento queda a la espera de que es esclavo envié el bit a transmitir. Para transmitir un '1' el esclavo lleva el bus a nivel alto y para transmitir un '0' lo deja

en nivel bajo. El master leerá el bus pasados 15 us y esperará hasta los 60 us para llevar de nuevo el bus a nivel alto y realizar de nuevo la operación de lectura hasta que se termine la transmisión de un byte.

Librerías de Microchip para el protocolo 1-Wire:

Microchip nos proporciona la librería necesaria para la comunicación con los dispositivos compatibles con 1-wire. Las funciones disponibles son la las siguientes:

- drive_OW_low: Configura el pin como salida y lleva el bus a nivel bajo
- drive_OW_high: Configura el pin como salida y lleva el bus a nivel alto
- read_OW: Configura el pin como entrada y lee el estado de este
- OW_write_byte: Transmite un byte al esclavo
- OW_read_byte: Lee un byte transmitido por el esclavo
- OW_reset_pulse: Produce un pulso de reset y detecta la presencia de esclavos
- OW_write_bit: Envía un bit por el bus
- OW_read_bit: Lee un bit enviado por el esclavo

ANEXO C. Configuración Timer 0 PIC18F87K22

Vamos a utilizar el Timer 0 del microcontrolador como temporizador. El Timer 0 es un contador de 8/16 bits que produce una interrupción en el micro cuando se produce un desbordamiento del contador TMR0 register (FFFFh a 0000h) activando la bandera TMR0IF. Una vez se ha producido este desbordamiento hay que limpiar la bandera TMR0IF de forma manual mediante software para que el contador se reinicie y empiece de nuevo.

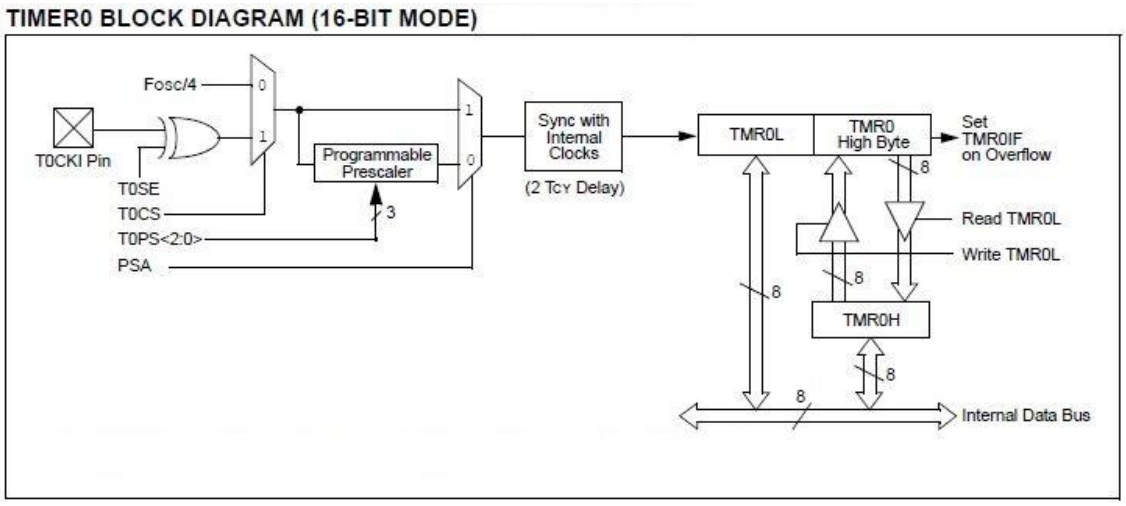


Fig. C.1 Diagrama de bloques Timer 0

Puede funcionar mediante dos relojes: el interno Fosc/4 y con reloj externo conectado al pin T0CKI (RA4), dispone de un prescaler programable de 8 bits y de registros TMR0L y TMRoH de lectura y escritura.

La ecuación para obtener un tiempo determinado sería la siguiente:

$$\text{Tiempo} = \left(\frac{4}{Fosc}\right) * \text{prescaler} * (65536 - \text{TMR0}) \qquad \textbf{(C.1)}$$

Para configurar el Timer0 hay que modificar el registro T0CON: TIMER0 CONTROL REGISTER

T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **TMR0ON**: Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-Bit/16-Bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin input edge
0 = Internal clock (FOSC/4)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = Timer0 prescaler is not assigned; Timer0 clock input bypasses prescaler
0 = Timer0 prescaler is assigned; Timer0 clock input comes from prescaler output

bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

Fig. C.2 Registro T0CON Timer 0

Configuración de tiempo 2,4 segundos:

- Reloj interno: Fosc/4
- Modo: 16 bits
- Reloj interno: Fosc/4
- Prescaler: 128
- Precargar el valor 46786 en TMR0

```
static void interrupt service_routine(void) { //Atiende las interrupciones del microcontrolador
GIE = 0; //Desactivamos las interrupciones mientras atendemos a una de ellas
//Interrupción del Timer0 cada 2.4seg
if (TMR0IF == 1) {
    TMR0 = 46786; //Precargo de nuevo el valor 46786
    Pulsos = NumPulsos;
    NumPulsos = 0;
    Time_out = 0; //Flag de tiempo transcurrido de 2.4 segundos
    TMR0IF = 0;
}
```

Fig. C.3 Interrupción Timer 0

Sabiendo que el oscilador del microntrolador está configurado a 4Mhz tenemos:

$$2.4\ \text{Segundos} = \left(\frac{4}{4\ \text{Mhz}}\right) * 128 * (65536 - 46786) \qquad \textbf{(C.2)}$$

Hay que tener en cuenta que cada vez que el contador TMR0 se desborda, empieza a contar de 0. Por ese motivo, hay que precargar de nuevo el valor de TMR0 de 46786 una vez que se produzca la interrupción y reseteamos la bandera TMR0IF.

```
//Setup Timer 0
void setup_timer_0(void) {

    TMR0IE = 1; // Enable TMR0 interrupts
    PSA = 0; // Prescaler in use
    T0CONbits.T08BIT = 0; //Mode 16-bits
    T0CONbits.T0CS = 0; // clock = FOSC/4
    T0PS2 = 1; T0PS1 = 1; T0PS0 = 0; //Prescaler 128
    TMR0 = 46786; //Preload 46786 at TMR0 count TMR0=(65536-46786)=18570
}
```

Fig. C.4 Configuración para tiempo de 2,4 segundos

ANEXO D. Configuración del ADC del PIC18F87K22

Para capturar el valor de salida de la veleta utilizaremos el canal 6 del ADC. Se trata de un ADC de 12 bits de resolución (4096 niveles) con 24 canales de entrada. Nosotros utilizaremos el canal 6 (AN6) para digitalizar la señal de salida de la veleta.

En la siguiente figura se puede ver el diagrama del ADC que integra el microcontrolador PIC18F87K22

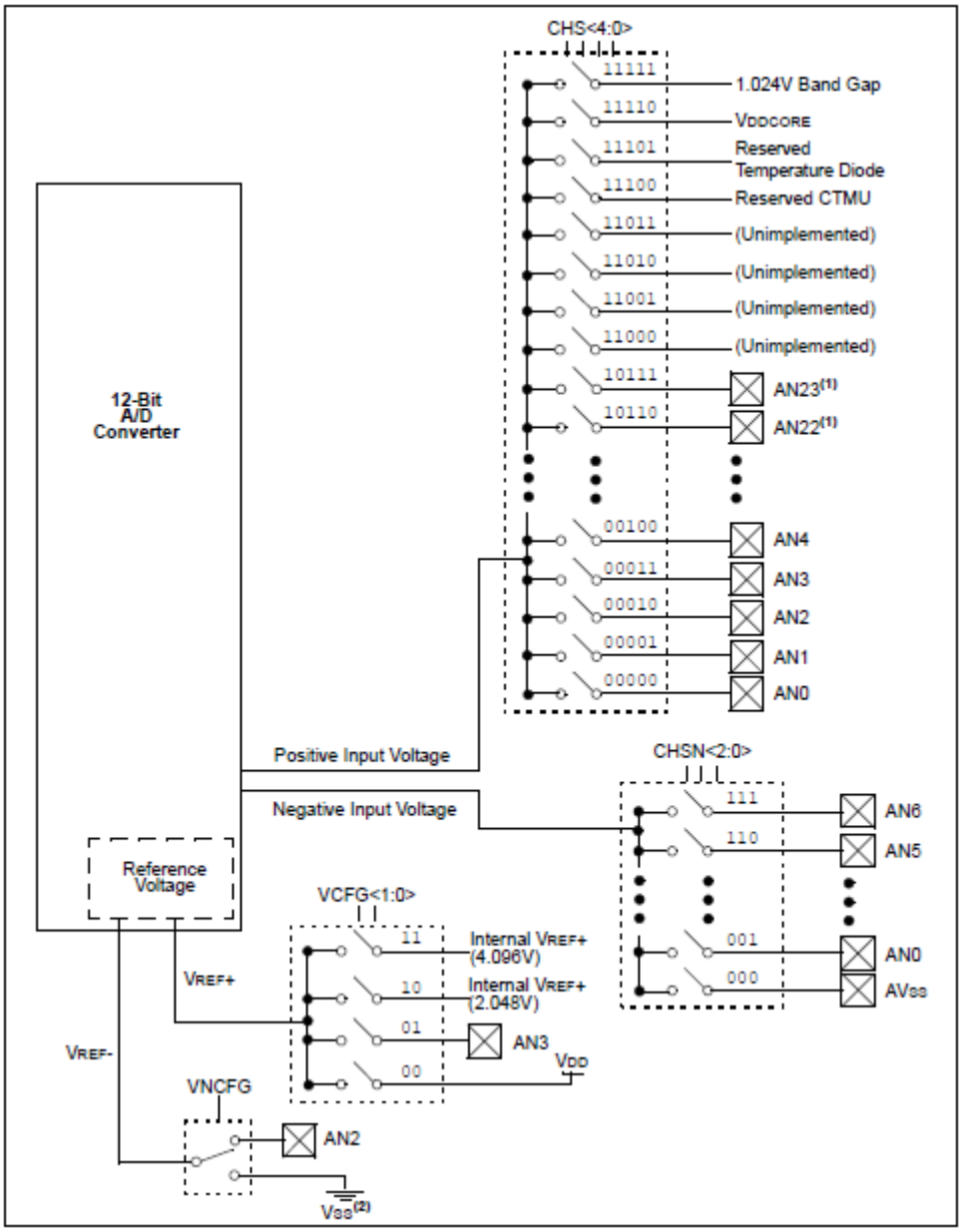


Fig. D.5 Esquema ADC PIC18F87K22

Para configurarlo tenemos diferentes registros en los que podemos seleccionar diferentes parámetros de nuestro ADC:

- ADCON0: Controla la operación del módulo ADC
- ADCON1: Configura el voltaje de referencia y los eventos especiales
- ADCON2: Configura la velocidad de muestreo del ADC, el tiempo de adquisición y la justificación del registro de salida.
- ANCON(0...2): Podemos seleccionar el tipo de puerto que vamos a utilizar entre analógico o digital.

ADCON0:

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CHS4	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Fig. D.6 Registro ADCON0

- CHS<4:0>: Bits para la selección del canal analógico
- GO/DONE: Bit que inicia la conversión
- ADON: Inicia o detiene el ADC

ADCON1:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TRIGSEL1	TRIGSEL0	VCFG1	VCFG0	VNCFG	CHSN2	CHSN1	CHSN0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Fig. D.7 Registro ADCON1

- TRGSEL<1:0>: Bits para la selección de eventos especiales
- VCFG<1:0>: Bits configuración A/D Vref+
- VNCFG<1:0>: Bits configuración A/D Vref-
- CHSN<2:0>: Bits de configuración canal analógico negativo

ADCON2:

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Fig. D.8 Registro ADCON2

- ADFM: Justificado del resultado
- ACQT<2:0>: Bits configuración Tiempo de Adquisición
- ADCS<2:0>: Bits selección de reloj de conversión

Configuración inicial del ADC:

- Canal analógico AN6
- Vref+ = Vdd; Vref - = Vss
- Justificado derecha
- Tiempo de adquisición 4TAD
- Reloj adquisición Fosc/4

```
// Configuración inicial ADC
    ADCON1bits.VCFG1=0; // Vref+ = Vdd
    ADCON1bits.VCFG0=0;
    ADCON1bits.VNCFG=0; // Vref- = Vss
    ADCON2 = 0b10010100; //ADRES justificado derecha, 4TAD, Fosc/4
```

Fig. D.9 Configuración inicial ADC

ANEXO E. Diagrama de pins del PIC18F87K22

Pin Diagrams – PIC18F8XK22

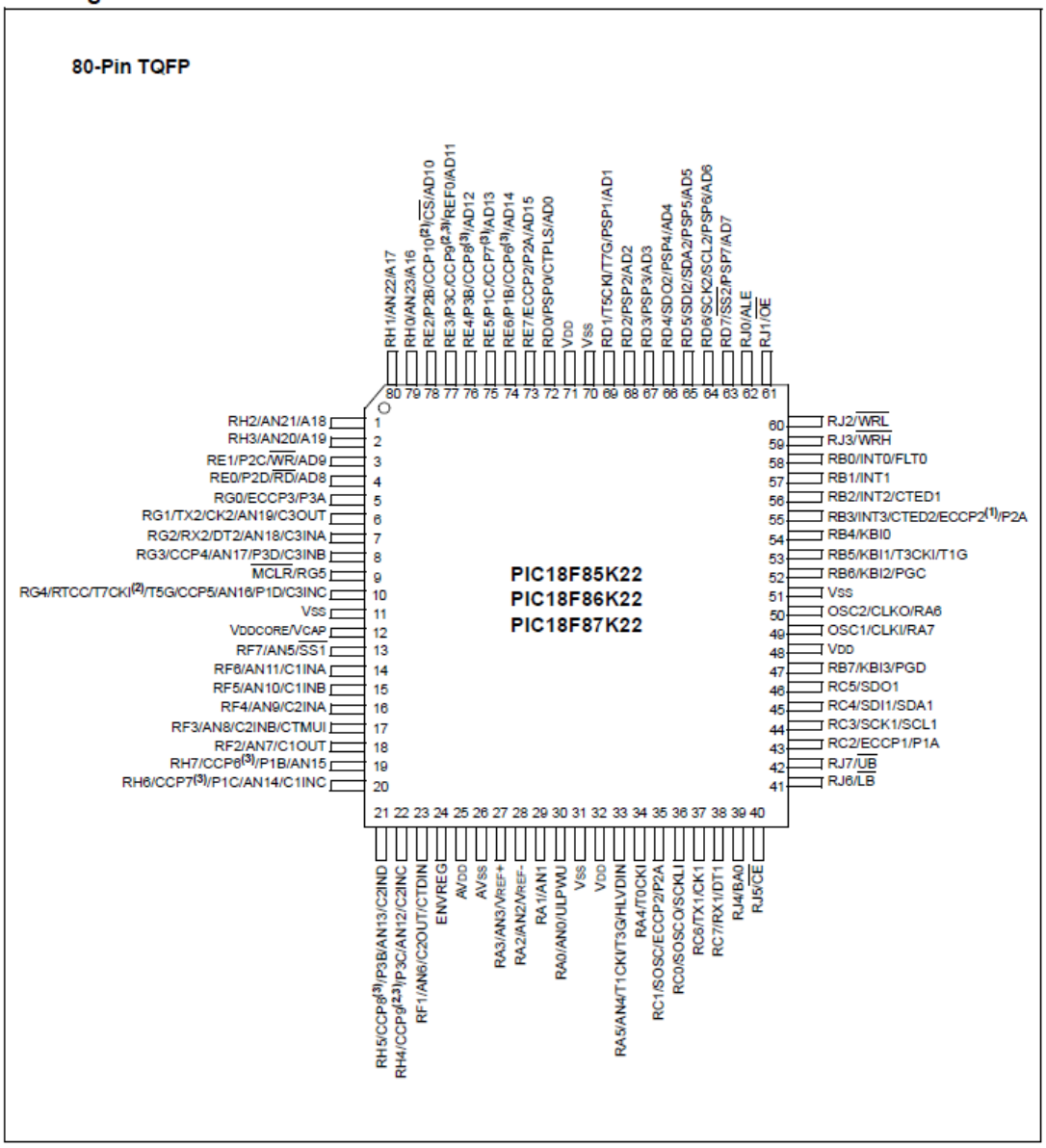


Fig. E.10 Diagrama de pines PIC18F87K22

ANEXO F. Esquema placa de entrenamiento Microchip DM240313

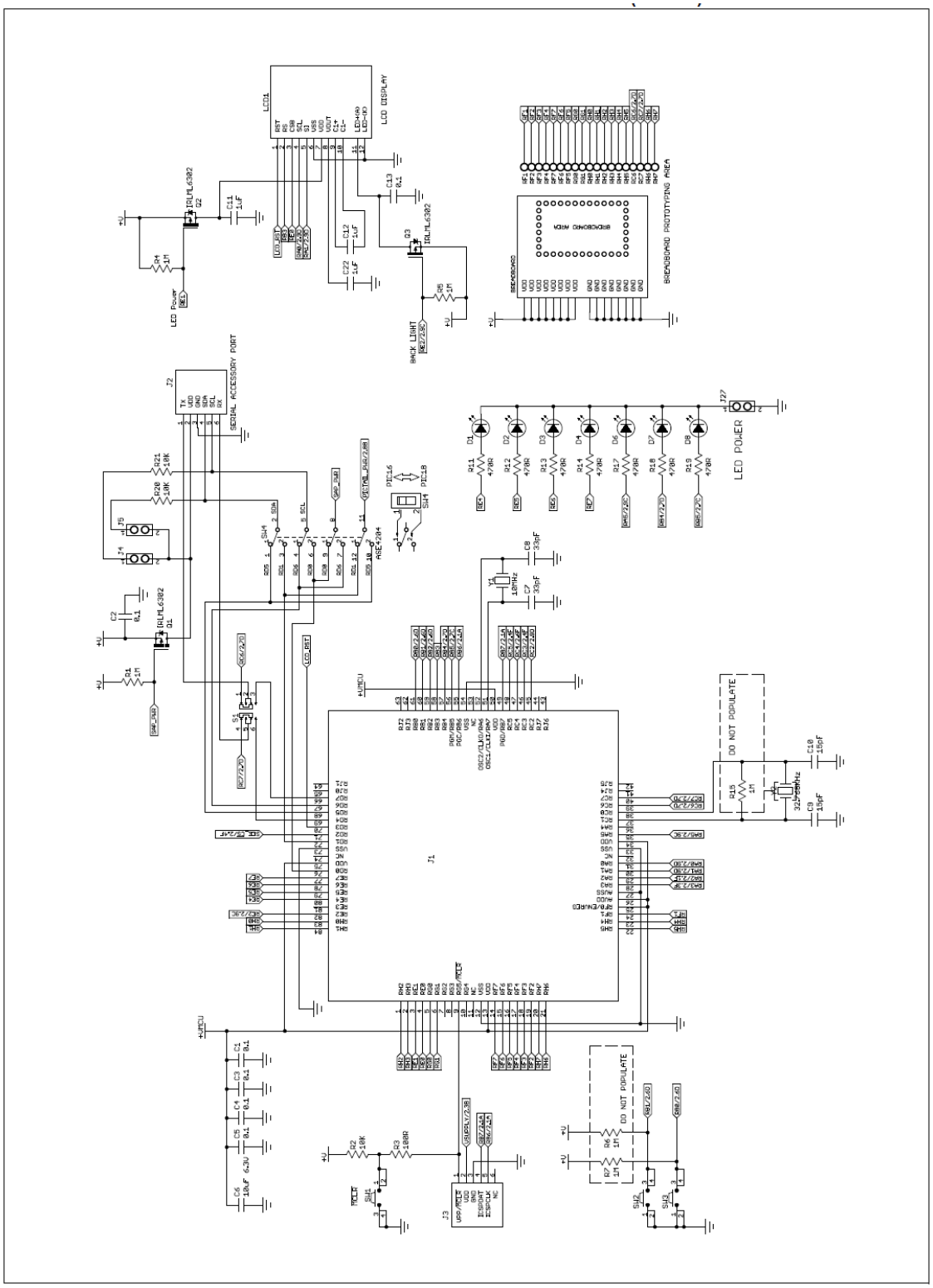


Fig. F.11 Esquema placa entrenamiento DM240313 1/2

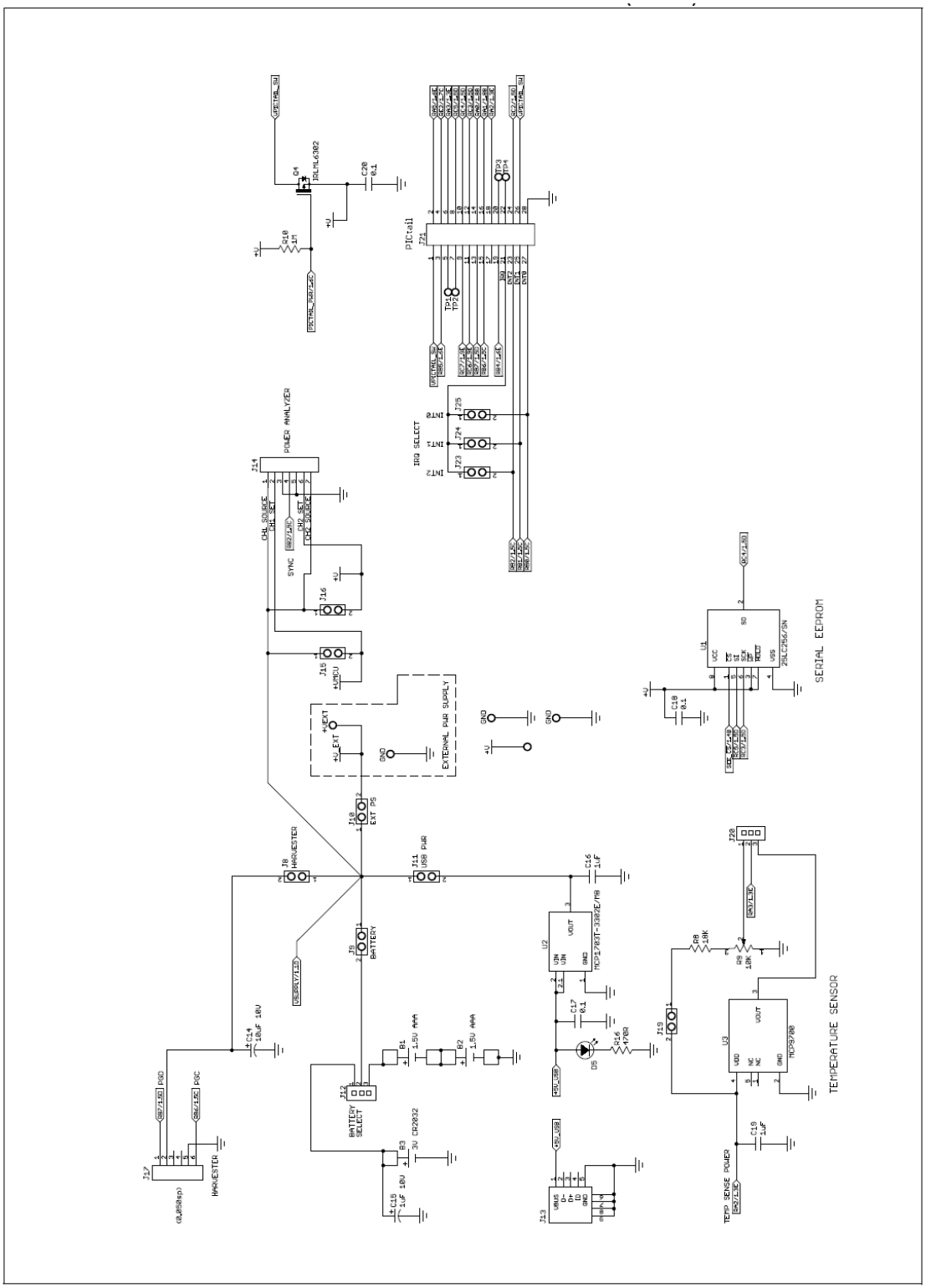


Fig. F.12 Esquema placa de entrenamiento DM240313

ANEXO G. Código principal Estación meteorológica

```
/*
 * File:   main.c
 * Author: Alberto Gómez Blázquez
 * Email: alberto.gomez.blazquez@estudiant.upc.edu
 *
 * Proyecto de estación meteorológica de bajo consumo e inalámbrica (Bluetooth)
 * Microcontrolador PIC18F87K22
 * Fosc = 4MHz
 * Envío de datos mediante protocolo RS232 (USART) y un dispositivo bluetooth HC-05 al PC (Interfaz en LabView)
 * Modo Sleep durante 1 minuto con despertar por desbordamiento de perro guardián (WDT)

 * Descripción de los sensores y el protocolo de comunicación que utilizan
 *
================================================================================
 Sensor---------Protocolo----------------  Descripción-----------------------------
================================================================================
 * DS18B20     1-Wire                    Sensor para medir la temperatura del agua

 * DHT22       Propio (Similar 1-Wire)   Sensor de humedad relativa y temperatura ambiente

 * BMP180      I2C                       Sensor para medir la presión atmosférica y la temperatura ambiente

 * Anemómetro  Interrupciones            Anemómetro que genera 1 pulsación por cada vuelta

 * WindVane    ADC                       Genera una tensión diferente para cada posición. Se mide con el ADC del micro
 *
 */


#include <xc.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
```

```
#include "delay.h"
#include "BMP180.h"
#include "DHT22.h"
#include "1wire.h"
#include "USART.h"
#include "Windvane.h"
#include "Velwind.h"
#include <pic18f87k22.h>
#include <plib/i2c.h>
#include <plib/usart.h>
#include <plib/delays.h>

/*
================================================================
============================
      Definitions

================================================================
============================ */
// Definición de los puertos
#define       PORTBIT(adr, bit) ( (unsigned)(&adr)*8 + (bit) )

//Definición de los estados
#define       Idle                  'A'
#define       Setup                 'B'
#define       Read_Cal_BMP180       'C'
#define       Read_DHT22            'D'
#define       Read_BMP180           'E'
#define       Read_DS18B20          'F'
#define       Read_Windvane         'G'
#define       Read_Vel_wind         'H'
#define       Send_data              'I'
#define       Sleep_mode            'J'
#define       Stop                  'K'

//Definición de los osciladores para configurar correctamente los delays

#define    LF                 0
#define    MF                 1
#define    HF                 2
#define    PLL                3
#define    XT_10              4
#define    _XTAL_FREQ         4000000
unsigned char Freq;
```

```
//Bus i2c
unsigned char sync_mode = 0, slew = 0;

//USART
char Txdata0[50] = "0";
unsigned char Rxdata;
unsigned char config=0,spbrg=0,baudconfig=0, i=0;
unsigned char inicio_usart = 1;


//Sensor BMP180
char TxdataBMP180[25] = "0";

//Sensor DHT22
char TxdataDHT22[15] = "0";
unsigned char DHT_22_flag;

//Sensor DS18B20
char TxdataDS18B20[16];
char TxdataDS18B20_USART[25] = "0";

//Anemómetro
char TxdataAnemometro[5] = "0";
unsigned int Pulsos;
unsigned char Time_out;
unsigned int NumPulsos;

//Windvane
char TxdataWindvane[3] ="0";

/*
=============================================================
============================
      Global variables

=============================================================
============================ */
//Leds de control placa
static bit LED_2 @ PORTBIT(PORTJ, 0);
static bit LED_1 @ PORTBIT(PORTD, 7);

//Variables estado inicial FSM
static char present_state = Idle;

//Variable interruoción RB0
```

```
unsigned char INT0_ST_SP_Flag;

//Variables interrupción USART
unsigned char Go_Idle_Flag;
unsigned char Go_Flag;


/*
============================================================
============================
      Function prototypes

============================================================
============================ */

void init_system(void);
static void interrupt service_routine(void);
char state_logic(void);
char output_logic(void);
void Setup_I2C(void);
void Setup_USART_1(void);
void setup_timer_0(void);


/*
============================================================
============================
      Main function

============================================================
============================ */

void main(void)
{
  //Programa principal
    init_system();
     /* loop forever */
    while (1) {
       output_logic();
       state_logic();
    }
}
```

```
/*
================================================================
=========================
      Functions

================================================================
========================= */
void init_system(void) {
// Configuración bits de control PIC18F87K22

// CONFIG1L
#pragma config RETEN = OFF      // VREG Sleep Enable bit (Disabled -
Controlled by SRETEN bit)
#pragma config INTOSCSEL = LOW  // LF-INTOSC Low-power Enable bit
(LF-INTOSC in Low-power mode during Sleep)
#pragma config SOSCSEL = HIGH   // SOSC Power Selection and mode
Configuration bits (Low Power SOSC circuit selected)
#pragma config XINST = OFF      // Extended Instruction Set (Disabled)

// CONFIG1H
#pragma config FOSC = INTIO2    // Oscillator (Internal RC oscillator)
#pragma config PLLCFG = OFF     // PLL x4 Enable bit (Disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor (Disabled)
#pragma config IESO = OFF       // Internal External Oscillator Switch Over
Mode (Disabled)

// CONFIG2L
#pragma config PWRTEN = OFF     // Power Up Timer (Disabled)
#pragma config BOREN = OFF      // Brown Out Detect (Disabled in
hardware, SBOREN disabled)
#pragma config BORV = 3         // Brown-out Reset Voltage bits (1.8V)
#pragma config BORPWR = ZPBORMV // BORMV Power level
(ZPBORMV instead of BORMV is selected)

// CONFIG2H
#pragma config WDTEN = ON       // Watchdog Timer (WDT disabled in
hardware; SWDTEN bit disabled)
#pragma config WDTPS = 16384    // Watchdog Postscaler (1:16384) 65 seg

// CONFIG3L

#pragma config RTCOSC = SOSCREF // RTCC Clock Select (RTCC uses
SOSC)
#pragma config EASHFT = ON      // External Address Shift bit (Address
Shifting enabled)
```

```
#pragma config ABW = MM        // Address Bus Width Select bits (8-bit address bus)
#pragma config BW = 16         // Data Bus Width (16-bit external bus mode)
#pragma config WAIT = OFF      // External Bus Wait (Disabled)

// CONFIG3H
#pragma config CCP2MX = PORTC  // CCP2 Mux (RC1)
#pragma config ECCPMX = PORTH  // ECCP Mux (Enhanced CCP1/3 [P1B/P1C/P3B/P3C] muxed with RE6/RE5/RE4/RE3)
#pragma config MSSPMSK = MSK7  // MSSP address masking (7 Bit address masking mode)
#pragma config MCLRE = ON      // Master Clear Enable (MCLR Enabled, RG5 Disabled)

// CONFIG4L
#pragma config STVREN = ON     // Stack Overflow Reset (Enabled)
#pragma config BBSIZ = BB2K    // Boot Block Size (2K word Boot Block size)

// CONFIG5L
#pragma config CP0 = OFF       // Code Protect 00800-03FFF (Disabled)
#pragma config CP1 = OFF       // Code Protect 04000-07FFF (Disabled)
#pragma config CP2 = OFF       // Code Protect 08000-0BFFF (Disabled)
#pragma config CP3 = OFF       // Code Protect 0C000-0FFFF (Disabled)
#pragma config CP4 = OFF       // Code Protect 10000-13FFF (Disabled)
#pragma config CP5 = OFF       // Code Protect 14000-17FFF (Disabled)
#pragma config CP6 = OFF       // Code Protect 18000-1BFFF (Disabled)
#pragma config CP7 = OFF       // Code Protect 1C000-1FFFF (Disabled)

// CONFIG5H
#pragma config CPB = OFF       // Code Protect Boot (Disabled)
#pragma config CPD = OFF       // Data EE Read Protect (Disabled)

// CONFIG6L
#pragma config WRT0 = OFF      // Table Write Protect 00800-03FFF (Disabled)
#pragma config WRT1 = OFF      // Table Write Protect 04000-07FFF (Disabled)
#pragma config WRT2 = OFF      // Table Write Protect 08000-0BFFF (Disabled)
#pragma config WRT3 = OFF      // Table Write Protect 0C000-0FFFF (Disabled)
#pragma config WRT4 = OFF      // Table Write Protect 10000-13FFF (Disabled)
```

```
#pragma config WRT5 = OFF      // Table Write Protect 14000-17FFF (Disabled)
#pragma config WRT6 = OFF      // Table Write Protect 18000-1BFFF (Disabled)
#pragma config WRT7 = OFF      // Table Write Protect 1C000-1FFFF (Disabled)

// CONFIG6H
#pragma config WRTC = OFF      // Config. Write Protect (Disabled)
#pragma config WRTB = OFF      // Table Write Protect Boot (Disabled)
#pragma config WRTD = OFF      // Data EE Write Protect (Disabled)

// CONFIG7L
#pragma config EBRT0 = OFF     // Table Read Protect 00800-03FFF (Disabled)
#pragma config EBRT1 = OFF     // Table Read Protect 04000-07FFF (Disabled)
#pragma config EBRT2 = OFF     // Table Read Protect 08000-0BFFF (Disabled)
#pragma config EBRT3 = OFF     // Table Read Protect 0C000-0FFFF (Disabled)
#pragma config EBRT4 = OFF     // Table Read Protect 10000-13FFF (Disabled)
#pragma config EBRT5 = OFF     // Table Read Protect 14000-17FFF (Disabled)
#pragma config EBRT6 = OFF     // Table Read Protect 18000-1BFFF (Disabled)
#pragma config EBRT7 = OFF     // Table Read Protect 1C000-1FFFF (Disabled)

// CONFIG7H
#pragma config EBRTB = OFF     // Table Read Protect Boot (Disabled)



//Configuración de las resistencias de pull-up en los puertos PORTB

   INTCON2bits.RBPU = 0; // 0 = Habilitadas las resistencias de pull-up en PORTB
                         // 1 = Deshabilitadas las resistencias de pull-up en PORTB

//Configuración del oscilador interno
//|IDLEN|IRCF2|IRCF1|IRFC0|OSTS|HFIOFS|SCS1|SCS0|
//|  0  |  1  |  0  |  1  |  1 |   1   |  0 |  0 |
```

```
//HF (4MHz), PLL=OFF,(IDLEN=0)Modo sleep activado, Modo Idle desactivado
   OSCCON = 0b01011100;
   OSCCON2bits.MFIOFS = 0;
   OSCCON2bits.MFIOSEL = 0;
   OSCTUNEbits.PLLEN = 0;
   OSCTUNEbits.INTSRC = 0;

//Configuración de los puertos

// PORTA:
   PORTA = 0x00; // Reset all Flip-Flops at PORTA
   LATA = 0x00; // Reset all Flip-Flops at PORTA
   TRISA = 0b11011010; // RA5 LED D6 /RA2 Alimentación del sensor de temperatura
   ANCON0bits.ANSEL2 = 0; //Puerto RA2 salida digital


// PORTB:
   PORTB = 0x00; // Reset all Flip-Flops at PORTB
   LATB = 0x00; // Reset all Flip-Flops at PORTB
   TRISB = 0b11001111; // RB0 BOTON/ RB2 Anemómetro/ RB4 LED D7/ RB5 LED D8

// PORTC:
   PORTC = 0x00; // Reset all Flip-Flops at PORTC
   LATC = 0x00; // Reset all Flip-Flops at PORTC
   TRISC = 0b10011000; //RC7 Rx(USART), RC6 Tx (USART), RC4=SDA1(I2C), RC3=SCL1(I2C)

// PORTD:
   PORTD = 0X00; // Reset all Flip-Flops at PORTD
   LATD = 0X00; // Reset all Flip-Flops at PORTD
   TRISD = 0b01100010; //RD6=SCL(I2C), RD5=SDA(I2C)

// PORTE:
   PORTE = 0x00; // Reset all Flip-Flops at PORTE
   LATE = 0x00; // Reset all Flip-Flops at PORTE
   TRISE = 0b00001111; // RE7/6/5/4 LED D4D3D2D1

// PORTF:
   PORTF = 0x00; // Reset all Flip-Flops at PORTE
   LATF = 0x00; // Reset all Flip-Flops at PORTE
   TRISF = 0b00000010; // RE7/6/5/4 LED D4D3D2D1
   ANCON0bits.ANSEL6 = 1; //Habilitada entrada analógica en puerto RF1
```

```
    ANCON0bits.ANSEL7 = 0; // I/O Digital en puerto RF2
    ANCON1bits.ANSEL8 = 0; // I/O Digital en puerto RF3

// PORTH:
    PORTH = 0x00; // Reset all Flip-Flops at PORTH
    LATH = 0x00; // Reset all Flip-Flops at PORTH
    TRISH = 0b00000000; // Todo salidas
    ANCON1 = 0b00000000; //Pines I/O se configuran como digitales
    ANCON2 = 0b00000000;

// PORTJ:
    PORTJ = 0x00; // Reset all Flip-Flops at PORTE
    LATJ = 0x00; // Reset all Flip-Flops at PORTE
    TRISJ = 0b00000000; // RE7/6/5/4 LED D4D3D2D1

// Configuración inicial ADC
    ADCON1bits.VCFG1=0; // Vref+ = Vdd
    ADCON1bits.VCFG0=0;
    ADCON1bits.VNCFG=0; // Vref- = Vss
    ADCON2 = 0b10010100; //ADRES justificado derecha, 4TAD, Fosc/4

// Configuración INTCON
    GIE = 1; // Global interrupts allowed
    PEIE = 1; //Interrupts periferic

// The INT2
    INTEDG2 = 0; // INT2 edge selection: '0' --> on falling edge (interrupt after
clicking the pushbutton)
    INT2IE = 1; // Enable interrupts from the INT2 pin

//Configuración de la frecuencia para los delays
    Freq = HF;

//Configuración del reloj para i2c
    SSP1ADD = 0x09;

//Condición inicial de las variables

    INT0_ST_SP_Flag = 0;
    Go_Idle_Flag = 0;
    Go_Flag = 0;
    Time_out = 1;
    NumPulsos = 0;
    present_state = Idle;
```

```
}
//*****************************************************************
*****************
//Future_state_logic routine
//*****************************************************************
*****************

char state_logic(void) {

   char error = 0;

   switch (present_state) {
      case Idle:
         if (Go_Flag == 1) {
            present_state = Setup;
            Go_Flag = 0;
         }
         else
            present_state = Idle;
         break;

      case Setup:
          if (Go_Flag == 1) {
            present_state = Stop;
            Go_Flag = 1;
         } else
            present_state = Read_Cal_BMP180;
         break;

      case Read_Cal_BMP180:
         if (Go_Flag == 1) {
            present_state = Stop;
            Go_Flag = 1;
         } else
            present_state = Read_DHT22;
         break;

      case Read_DHT22:
         if (Go_Flag == 1 ) {
            present_state = Stop;
            Go_Flag = 0;
         } else
            present_state = Read_BMP180;
         break;
```

```
case Read_BMP180:
   if (Go_Flag == 1 ) {
      present_state = Stop;
      Go_Flag = 0;
   } else
      present_state = Read_DS18B20;
   break;

case Read_DS18B20:
   if (Go_Flag == 1) {
      present_state = Stop;
      Go_Flag = 0;
   } else
      present_state = Read_Windvane;
   break;

case Read_Windvane:
   if (Go_Flag == 1) {
      present_state = Stop;
      Go_Flag = 0;
   } else
      present_state = Read_Vel_wind;
   break;


case Read_Vel_wind:
   if (Go_Flag == 1) {
      present_state = Stop;
      Go_Flag = 0;
   } else
      present_state = Send_data;
   break;

case Send_data:
   if (Go_Flag == 1) {
      present_state = Stop;
      Go_Flag = 0;
   } else
   present_state = Sleep_mode;
   break;

case Sleep_mode:
   if (Go_Idle_Flag == 1) {
      present_state = Stop;
      Go_Idle_Flag = 0;
```

```
            } else
            present_state = Read_DHT22;
            break;

        case Stop:
            present_state = Idle;
            break;

        default:
            error = 1;
    }
    return (error);
}

//*****************************************************************
*****************
//Output logic routine
//*****************************************************************
*****************

char output_logic(void) {
    char error = 0;

    switch (present_state) {
        case Idle:
            //Inicio modulo USART
            if (inicio_usart == 1) {
                Setup_USART_1();
            }
            //USART se inicia solo una vez
            inicio_usart = 0;
            //Leds control placa que inidican que estamos en modo Idle
            LED_1 = 1; LED_2 = 1;

            break;

        case Setup:
            //Setup I2C
            //100kHz Baud clock(SSPADD =9) Fosc=4MHz Configurado en
Init_System
            Setup_I2C();
            //Preparo el bus del sensor DHT22
            Setup_port_DHT22();
            //Preparo el bus del sensor DS18B20
            Setup_1_wire_port();
```

```
            break;

        case Read_Cal_BMP180:
            //Cambio estado leds control para indicar que estamos obteniendo y
enviando datos
            LED_1 = 0; LED_2 = 1;
            //Leer parámetros de calibración del sensor BMP180
            Get_CalValue_BMP180();

            break;

        case Read_DHT22:
            //Medimos la humedad y la temperatura ambiente con el sensor
DHT22
            Read_Dev_DHT22();

            break;

        case Read_BMP180:
            //Medimos la temperatura y la presión atmosférica con el sensor I2C
BMP180
            Read_Dev_BMP180();

            break;

        case Read_DS18B20:
            //Leo el sensor impermeable de temperatura DS18B20
            ReadTemp_DS18B20();

            break;

        case Read_Windvane:
            //Capturo dirección del viento con ADC
            Capture_Windvane();

            break;

        case Read_Vel_wind:
            //Cuento los pulsos de anemómetro durante 2.4 seg para medir la
velocidad del viento
            Read_VelWind_Int();

            break;
```

```
        case Send_data:
            //Juntar los datos y enviarlos por USART
            strcpy(Txdata0,TxdataDHT22);
            strcat(Txdata0,TxdataBMP180);
            strcat(Txdata0,TxdataDS18B20);
            strcat(Txdata0,TxdataAnemometro);
            strcat(Txdata0,TxdataWindvane);
            strcat(Txdata0,"\r\n");
            //Enviar datos  por puerto serie
            while (Busy1USART()); //Check if Usart is busy or not
            puts1USART((char *) Txdata0); //transmit the string

            break;

        case Sleep_mode:
            //Entro en modo sleep
            //Leds que indican modo Sleep
            LED_1 = 1; LED_2 = 0;

            //WDT Setup
            //Watchdog timer is ON
            WDTCONbits.SWDTEN = 1;
            BAUDCON1bits.WUE = 1; //Configuración la interrupción EUSART
en modo sleep
            //Entro en modo Sleep
            Sleep();
            NOP();//Se recomienda que la siguiente instrucción después del modo
sleep sea NOP

            //El dato recibido en modo Sleep es indiferente, solo activa la bandera
de dato recibido
            Rxdata = getc1USART(); //Recibo el dato después del modo sleep

            __delay_ms(20);
            // Si el estado de el registro WUE ha cambiado a 0, ha habido una
interrupción por USART
            if (BAUDCON1bits.WUE == 0) {
               Go_Idle_Flag = 1;
            }
            //Limpio la bandera de recepción USART
            PIR1bits.RC1IF = 0;
            //Watchdog timer is OFF
            WDTCONbits.SWDTEN = 0;

            break;
```

```
        case Stop:
            //Cierro comunicación USART
            Close1USART();
            inicio_usart = 1; //Para volver a enviar mensaje de inicio
            //Apago puerto DHT22
            Off_port_DHT22();
            //Apago puerto DS18B20
            Off_1_wire_port();

            break;

        default:
            error = 1;
    }
    return (error);
}

/*
=============================================================
=========================
        Interrupt function

=============================================================
========================== */
static void interrupt service_routine(void) { //Interrupt is the key word here
GIE = 0; //Disable interrupts while attending one of them ...

//Interrupción del Timer0 cada 2.4seg
if (TMR0IF == 1) {
    TMR0 = 46786; //Precargo de nuevo el valor 46786
    Pulsos = NumPulsos;
    NumPulsos = 0;
    Time_out = 0; //Flag de tiempo transcurrido de 2.4 segundos
    TMR0IF = 0;
}

//Interrupción de anemómetro en RB2
if (INT2IF == 1) {
    NumPulsos++;
    INT2IF = 0;
}

//Interrupción USART
if (RC1IF == 1) {
```

```
   Rxdata = getc1USART(); //Se lee el dato que está en el buffer de Rx del
USART
   PIR1bits.RC1IF = 0; //Desactivamos la bandera de recepción en el buffer
de entrada del USART


   switch (Rxdata) {
//Datos recibidos en estado IDLE

          case '3':
             Go_Flag = 1;
             break;
          default:
             Go_Flag = 0;
             break;
       }
   Rxdata = 0;
}

GIE = 1; //Enable interrupts
}

//Funciones I2C
void Setup_I2C(void) {
   sync_mode = MASTER;
   slew = SLEW_OFF;
   OpenI2C(sync_mode, slew);

}

//Timer 0
void setup_timer_0(void) {
//    Configuración Timer0 para tiempo 2.4 segundos con Fosc=4MHz:
//          |Tiempo|Prescaler|T0PS2|T0PS1|T0PS0|  TMR0  |
//          |2.4seg|   128   |  1  |  1  |  0  | 46786  |
//
//    IMPORTANTE!!:Hay que modificar el valor de TMR0 en la interrupción
del timer0
//    Para cualquier otro tiempo:
//      Tiempo = (4/Fosc)*prescaler*(65536-TMR0)
   TMR0IE = 1; // Enable TMR0 interrupts
   PSA = 0; // Prescaler in use
   T0CONbits.T08BIT = 0; //Mode 16-bits
   T0CONbits.T0CS = 0; // clock = FOSC/4
   T0PS0 = 0;
```

```
   T0PS1 = 1;
   T0PS2 = 1; // T0PS(2..0) = "110" programs a 1:128 Prescale value
   TMR0 = 46786; //Preload 46786 at TMR0 count TMR0=(65536-
46786)=18570
}
```