3. Diseño y construcción del prototipo

La medición de una variable o cantidad física se realiza por medio de un sensor (o transductor). En este caso, como se estableció en el capítulo anterior, la medición de las variables se debe realizar por medio de un grupo de sensores llamado sonda. Antes de comenzar con la descripción de los tipos de sensores a utilizar, es oportuno tratar los conceptos relacionados.

Sensores y transductores.

Un transductor es un dispositivo que transforma una variable física en otra; un sensor utiliza uno o más transductores para transformar una variable física en otra que sea susceptible de ser medida. Por ejemplo, cuando utilizamos un termómetro de mercurio estamos transformando la variable física temperatura a una variable diferente: la longitud de una columna de mercurio dentro de un tubo graduado que puede ser interpretada fácilmente. Aunque existen diferentes tipos de sensores, los más utilizados son aquellos que tienen una salida de tipo eléctrica: el amplio manejo y la facilidad de almacenamiento de las señales eléctricas es la causa de su gran difusión.

Para elegir un sensor, o cualquier otra cosa, necesitamos evaluar su desempeño de alguna manera: cuando compramos un foco lo hacemos dependiendo de la iluminación que brinde, de su consumo de energía y/o de su precio. En el caso de los sensores, también existen ciertas características que nos ayudan a evaluar su desempeño:

El rango de trabajo o rango dinámico. Es el rango en el que la variable puede ser medida con una exactitud aceptable. Se expresa en las unidades propias de la variable.

Exactitud: nos indica que tanto se acerca el valor que obtenemos de un sensor al valor verdadero. Se define como el error mas grande esperado, es decir, la diferencia entre el valor verdadero y la lectura mas lejana obtenida del sensor para dicho valor. Suele presentarse en términos de la variable medida, pero también puede expresarse en términos de la señal de salida o en porcentaje en relación al rango de trabajo. Por ejemplo, siguiendo con el caso del termómetro, la exactitud puede decirse que es de 1°C o que es de 2% si su rango de trabajo es de 0°C a 50°C.

Precisión: Suponiendo que hacemos varias mediciones, manteniendo constantes tanto el valor de la variable como las condiciones en que se realiza la medición. La precisión nos indica que tanto se esparcirán los valores obtenidos. Existen un concepto similar al de precisión: la repetibilidad. La repetibilidad describe la variación de un grupo de medidas dado un periodo de tiempo. Al ahora considerar que entre cada medición existe un intervalo de tiempo, también se consideran condiciones diferentes, por lo tanto este parámetro puede considerarse más realista que el de precisión y en algunos sensores se indica en lugar de la precisión. Suele expresarse en las unidades de medida de la variable.

Resolución(o discriminación): es la medida más pequeña que podemos obtener (discriminar) de un sensor, en otras palabras, es el incremento más pequeño en el valor de la variable que puede ser detectado. Por ejemplo, para un reloj analógico con segundero, su resolución es de un segundo.

Linealidad. Nos indica que tan lineal es la relación entre la variable de entrada y el valor de salida. El valor obtenido en la salida siempre depende de la señal de entrada, esta dependencia o relación puede ser expresada en forma de una ecuación matemática, mientras dicha expresión se aproxima mas a la ecuación de una recta (y=mx+b) se dice que la respuesta del sensor es mas lineal. En general se busca una relación entrada-salida lineal (o proporcional) debido a que es más fácil interpretar las lecturas obtenidas que en un sistema no lineal.

Estabilidad a largo plazo. Todos los sensores van sufriendo de un corrimiento en su respuesta con el paso del tiempo, por lo que es importante conocer dicha variación para tomarla en cuenta y así obtener resultados mas apropiados. La estabilidad a largo plazo nos indica el valor de este corrimiento en un periodo de tiempo largo, generalmente en un año. Por ejemplo, si decimos que nuestro reloj se atrasa un segundo en un año, estamos indicando que su estabilidad a largo plazo es de un segundo por año(1s/año).

Errores Otro concepto importante a tener en cuenta en una medición es el de error dado que existe inevitablemente en todas las mediciones. Error es la diferencia entre el valor verdadero y el valor obtenido. Tiene dos componentes:

- Error sistemático es aquel que siempre afecta de la misma manera a una lectura, es decir, es una componente constante, por lo tanto, si se conoce su fuente puede ser corregido a través de métodos de compensación. Las principales fuentes de errores sistemáticos son: mala calibración de los sensores y efectos externos al sistema de medición, como cambios en la temperatura, cambios en la presión barométrica, campos magnéticos, etc.
- Error aleatorio es el que aparece por razones desconocidas y de manera aleatoria en cada medición, pueden ser positivos o negativos, de una magnitud pequeña o una magnitud más amplia. Por tratarse de un error impredecible no puede eliminarse, no obstante, sus efectos se disminuyen al realizar varias mediciones y aplicarles algún método estadístico, generalmente un promedio.

Proceso de medición.

Todos los sensores tienen, en su manera más simple, una salida analógica. Ahora bien, aunque se puede guardar un registro de señales analógicas, es preferible hacer un registro digital debido a sus amplias ventajas (provee de una mayor facilidad de procesamiento, transporte de información más sencillo, copia de información sin perdida de calidad, etc.).

Con objeto de convertir las señales analógicas obtenidas del sensor en señales digitales, se utiliza un convertidor analógico a digital (convertidor A/D). No obstante, la amplitud de la salida de muchos de los sensores es muy pequeña o simplemente inapropiada para los convertidores, por lo cual, antes de hacer la conversión es necesario realizar un acondicionamiento de la señal. El acondicionamiento de la señal consiste en amplificaciones, cambios de nivel de la señal, aislamientos, linealización y/o filtrado. Cada sensor, dependiendo de sus características de salida y de los requerimientos del convertidor A/D utilizado necesitará alguno o varios de estos procesos durante el acondicionamiento. En resumen, el sensor transforma la variable a medir en una señal eléctrica, a continuación se acondiciona de acuerdo al convertidor A/D, quien transforma la señal a una señal digital, y finalmente se le da un formato a la información y se almacena (ver figura 3.1).

Actualmente se ha alcanzado una alta integración en los circuitos integrados, lo que ha permitido reducir los elementos necesarios para implementar las etapas de acondicionamiento y conversión analógica a digital.

Esta reducción ha llegado incluso a la fabricación de sensores que tienen integradas ambas etapas en el mismo encapsulado, obteniendo a su salida información de tipo digital que se puede manipular por medio de un microcontrolador (Ver figura 3.1). Este tipo de sensores se les suele llamar sensores inteligentes (en inglés smart sensor), sensores de silicio o sensores digitales.

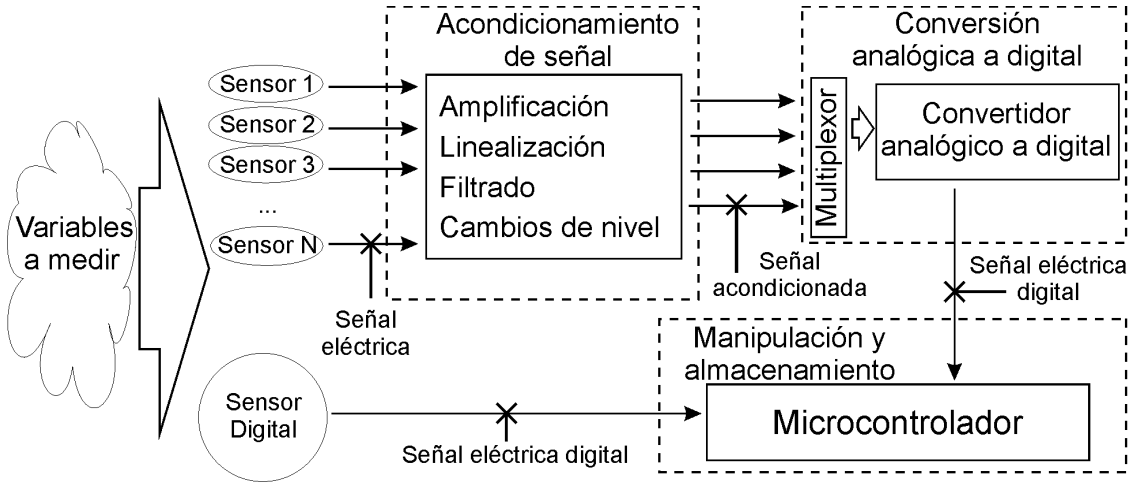


Figura 3.1.: Proceso de medición por medio de sensores analógicos y digitales

La Organización Meteorológica Mundial, por medio de la Guía de Instrumentos Meteorológicos y Métodos de Observación es quién establece los requisitos mínimos necesarios que deben cubrir los sensores de una sonda de globo cautivo; dichas especificaciones se resumen en la tabla 3.1.

Tabla 3.1: Especificaciones de la OMM para globos cautivos

Variable	Rango de operación	Resolución
Presión	850 a 1050 hPa	±0.5 hPa
Temperatura	–20 °C a +40 °C	±0.1 °C
Humedad Relativa	10% a 100%	±2%
Velocidad de Viento	0.5 a 15 m/s	±0.5 m/s
Dirección de Viento	0° a 360°	±1°

Adicionalmente son deseables las siguientes características:

- Que sean robustos como para soportar sacudidas.
- De dimensiones reducidas.
- Tener el menor peso posible.
- Niveles de alimentación bajos
- Consumo de corriente bajo.

3.1. Elección de los sensores y dispositivos.

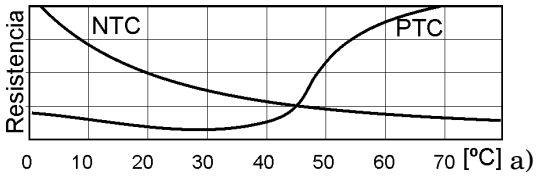
Una vez que se conocen las características generales de los sensores así como los requerimentos del proyecto, es necesario conocer de manera breve las diferentes opciones que existen en el mercado. Esto con objeto elegir el dispositivo más adecuado al proyecto.

3.1.1. Temperatura

Existen diversos métodos para medir la temperatura, sin embargo, en su mayoría implican la detección de un cambio en la resistencia eléctrica o en la tensión de salida del sensor o transductor. A continuación se describe de manera breve los principales tipos de sensores de temperatura.

Termistores (thermally sensitive resistors o resistencias térmicamente sensitivas) Estos dispositivos se fabrican de un material semiconductor de tipo cerámico que cambia su resistencia eléctrica en respuesta a un cambio en su temperatura. Se caracterizan por ser pequeños, con tiempo de respuesta rápida, excelente sensibilidad, estructura robusta y bajo costo. Sin embargo, su respuesta es poco lineal, como se ve en la figura 3.2. Existen dos tipos de termistores:

- Coeficiente de temperatura positivo (PTC), Aumentan su resistencia con la temperatura.
- Coeficiente de temperatura negativo (NTC), Disminuyen su resistencia con la temperatura.



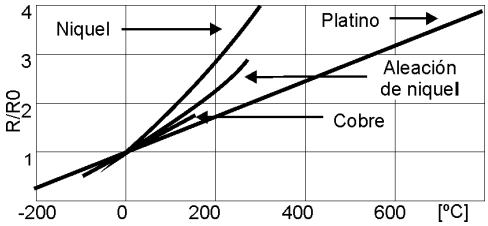
a) Curva de comportamiento R vs T

b) Ejemplo de un termistor

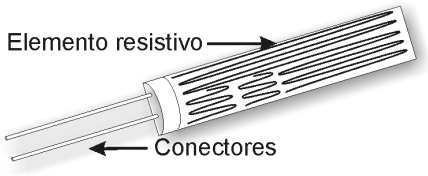
Figura 3.2.: Características de los termistores

RTD. (Sensores de tipo resistivo) También aprovechan el cambio de resistencia eléctrica con la temperatura, pero en los metales semiconductores. En materiales como en el platino, el cobre y el níquel, el comportamiento es bien conocido y altamente repetible.

Los RTD utilizan una red o una espiral de estos metales como elemento sensor, ver figura 3.3b. El platino es el preferido dado que tiene una mejor estabilidad a largo plazo, es químicamente inerte, resiste a la oxidación y su rango de trabajo mayor. La respuesta de un RTD es bastante lineal en un amplio rango de temperatura, como se ve en la figura 3.3a, lo cual evita el uso de circuitos de linealización. Sin embargo, su precio es alto y son vulnerables a vibraciones o golpes.



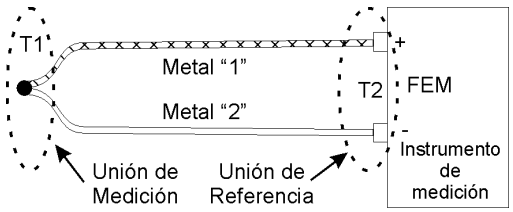
a)Curvas de comportamiento R vs T



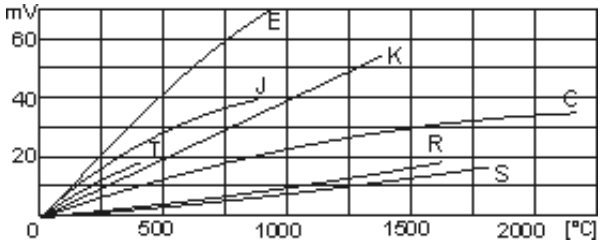
b) Estructura de un RTD

Figura 3.3.: Características de un RTD

Termopares Están formados de dos conductores eléctricos de distintos metales unidos en sus extremos. Cuando las uniones se encuentran a temperaturas diferentes (T1 y T2) se crea una fuerza electromagnética (FEM) que puede ser medida. La unión que se expone a la temperatura a medir se llama unión de medición; la otra unión se denomina unión de referencia y es en donde se obtiene la FEM, como se observa en la figura 3.4 a. La señal obtenida es muy pequeña, por lo que es fácilmente afectada por el ruido y debe ser sometida a un proceso de acondicionamiento que incluya: amplificación, filtrado, linealización y compensación. La curva de comportamiento de varios termopares se muestra en la figura 3.4 b.



a) Circuito de conexión



b) Curvas de comportamiento

Figura 3.4.: Características de los termopares

Sensores en Circuito Integrado La base de los circuitos integrados son los transistores, uniendo las terminales base y colector y haciendo pasar una corriente constante por el circuito se obtiene una tensión entre las terminales base y emisor (VBE), que es dependiente de la temperatura en aproximadamente 2mV/°C. Aprovechando esta característica se obtienen sensores de temperatura bastante lineales en un rango de -55°C a 150°C con un costo relativamente bajo. Podemos ver un sensor de este tipo en la figura 3.5.

Figura 3.5.: Sensor de temperatura en circuito integrado

En el mercado existen diversos sensores de temperatura, en la tabla 3.2 se muestran algunos sensores que cumplen con los requerimientos básicos (rango: –20° C a +40° C; Resolución: ±0.1 C).

Tabla 3.2: Comparación de algunos sensores de temperatura

SENSOR	NT03 10269 AMETHERM	P1K0.232.6W.Y.010 IST	DM-310 LABFACILITY	SHT75 SENSIRION
Tipo	Termistor	RTD	RTD	Circuito Integrado
Rango	-50 A 150 °C	Hasta 600°C	-70 a 500°C	- 40 A 123 °C
Exactitud	±10°C	±0.15°C	±0.15°C	± 0.3 °C
Resolución	-	-	-	0.01 °C
Estabilidad	-	0.1% / 1000 Hr	0.05 / año	ND
Alimentación	-	-	-	3 V
Tipo de Salida	Resistencia	Resistencia	Resistencia	Digital serial
T [a]	10 s	8 s	ND	5 s
Precio[b]	USD $0.912	USD $36.97	USD $15.16	USD $44.94 [c]

[a] Tiempo de respuesta 63% de señal
[b] http://mexico.newark.com/
[c] Precio de sensor de temperatura y humedad

De la tabla 3.2 podemos observar que el SHT75 tiene el menor tiempo de respuesta y su resolución es diez veces menor que la requerida. Por otro lado, al tener una salida digital no requiere de circuitos de acondicionamiento ni de conversión A/D, esto simplifica el circuito y ahorra espacio. Su precio es apenas superior al precio del RTD de la marca IST e incluye un sensor de humedad relativa dentro del mismo encapsulado, lo cual lo hace también idóneo en cuanto a precio.

3.1.2. Humedad Relativa

La humedad es la cantidad de vapor de agua contenida en el aire, generalmente se expresa en términos de humedad relativa. La humedad relativa (HR) es la cantidad de agua que hay en el aire, en relación a la cantidad de agua que tendría en el estado de saturación, suponiendo la misma presión y temperatura. Normalmente se expresa en términos de porcentaje.

Otra forma de definirla es como la presión parcial de vapor en relación a la presión de vapor de saturación. Es decir:

$$HR = (Pv/Psat) * 100\%$$

Donde:

Pv = presión parcial de vapor, es la presión ambiental debida a la presencia de vapor de agua en el aire.

Psat = presión de saturación, es la máxima presión de vapor que un gas puede soportar antes de que el vapor se condense.

Básicamente se utilizan dos tipos de sensores electrónicos de humedad: los sensores de tipo resistivo y los sensores de tipo capacitivo.

Los sensores de tipo resistivos miden el cambio en el valor de la resistencia eléctrica con respecto a la humedad, en algún polímero conductivo, sal, o algún sustrato tratado. La relación entre humedad relativa y resistencia es de tipo exponencial inversa.

Los sensores de tipo capacitivo detectan un cambio en la humedad relativa del ambiente y la reflejan como una variación en el valor de su capacitancia. Se encuentran formados por dos placas conductoras separadas por un aislante que absorbe y expele vapor de agua del ambiente conforme cambia la humedad, esto se refleja como un cambio en la constante dieléctrica.

Comúnmente se utiliza un polímero como dieléctrico y las placas conductoras son de platino como se muestra en la figura 3.6. La superficie se encuentra cubierta por una capa que aísla al sensor de polvo y basura, pero que permite el paso de la humedad para una óptima medición.

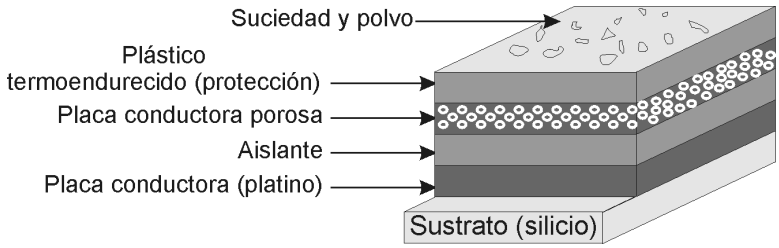


Figura 3.6.: Composición de un sensor de humedad de tipo capacitivo.

Debido a la linealidad de los sensores de tipo capacitivo estos son los más utilizados y por ende los de más fácil adquisición. En la tabla 3.3 se presenta una comparación de algunos sensores de humedad relativa.

Tabla 3.3: Comparación de algunos sensores de humedad

SENSOR	P14 IST	HIH-4021-004 HONEYWELL	SHT75 SENSIRION
Rango	0 A 100%	0 A 100%	0 A 100%
Exactitud	1.5% [a]	3.5%	1.8% [b]
Resolución	0.25 pF	-	0.05% HR
Estabilidad	n/d	1.2% HR/año	0.5% HR/Año
Alimentación	<12V	5V	3 V
Consumo	-	0.20 mA	0.55mA
Tipo de Salida	Capacitancia	Tensión Analógica	Serial Digital
T [c]	5 s	5s	8 s
Precio[d]	USD $39.70	USD $23.40	USD $44.94 [e]

[a] 15% a 90% HR
[b] 20% A 80%
[c] Tiempo de respuesta 63% de señal
[d] http://mexico.newark.com/
[e] Sensor de temperatura y humedad

Como ya se mencionó, el SHT75 tiene la ventaja de contar con una salida de tipo digital por lo que no requiere de circuitos adicionales. Su exactitud es mejor que la presentada por el sensor de Honeywell (1.8 % casi la mitad) y presenta una mejor estabilidad a largo plazo. Su tiempo de respuesta es mayor que los otros sensores, esto debido a que cuenta con una capa que lo protege de polvo, suciedad y agua. Aunque su precio casi duplica al del sensor Honeywell, el ahorro de componentes, espacio, peso y la inclusión de un sensor de temperatura integrado lo compensan perfectamente haciéndolo la mejor opción. Una ventaja adicional que tiene el SHT75 sobre el HIH (Honeywell) es que su alimentación puede ser incluso menor a 3V, lo cual es de vital importancia para la aplicación.

3.1.3. Velocidad del viento.

La intensidad del viento es una cantidad vectorial tridimensional y por lo tanto está conformada por una dirección de tres componentes (dirección del viento) y una magnitud (llamada velocidad de viento), que es una cantidad escalar.

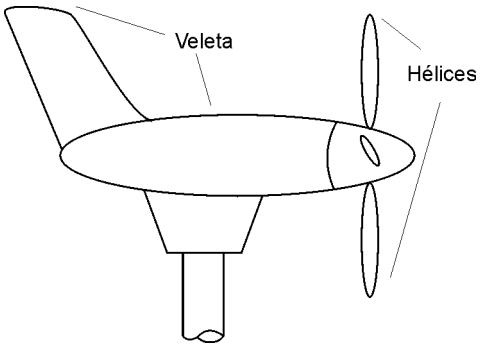
Dos de las componentes que conforman la dirección del viento se encuentran en un plano paralelo a la superficie terrestre (plano XY) y la tercera se encuentra perpendicular a dicho plano, es decir es vertical. Sin embargo, para la mayoría de los propósitos meteorológicos la componente vertical es despreciada, por lo que se considera a la intensidad de viento como una cantidad vectorial de dos dimensiones.

Los instrumentos que miden la magnitud de la intensidad de viento se conocen como anemómetros. Existen varios tipos de anemómetros de acuerdo a su principio de operación y a sus características de uso como son:

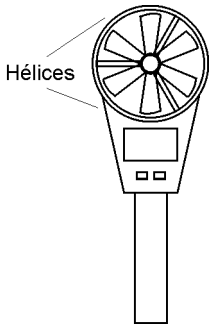
Tipos de anemómetros
- De rotación
- Dependientes de la temperatura
- De empuje
- De compresión
- Láser
- Sónicos

Para nuestro caso utilizaremos un anemómetro de rotación de tipo hélice, el cual cuenta con palas o álabes que están dispuestas sobre un eje horizontal de cara al viento gracias a una veleta (ver figura 3.7a). El viento hace girar los álabes a una velocidad proporcional a la velocidad del viento.

Existen modelos muy simples que no cuentan con veleta para orientarse y deben orientarse manualmente (ver figura 3.7b), sin embargo cuentan con la ventaja de ser muy ligeros y económicos. Se utilizan principalmente dos tipos de transductores para convertir la velocidad de rotación del eje en pulsos eléctricos: los de tipo óptico y los de tipo magnético. Estos pulsos son interpretados y transmitidos a un sistema de despliegue y/o almacenamiento de datos.



a) Anemómetro de tipo hélice con veleta



b) Anemómetro tipo hélice portátil sin veleta

Figura 3.7.: Anemómetros de tipo hélice

3.1.4. Dirección del viento.

Se entiende como dirección del viento a la dirección de donde sopla el viento y se mide por medio de instrumentos llamados veletas. Una veleta es un dispositivo giratorio, compuesto por una lámina rectangular u otra forma adecuada (cola) y un contrapeso (punta) fijos sobre una varilla horizontal que gira libremente y con la menor fricción sobre un eje vertical ubicado en el centro de gravedad. Como la cola tiene un área mayor de exposición al viento que la punta, la veleta gira hasta posicionarse de modo que el viento aplique la menor fuerza posible. De esta manera la veleta queda apuntando en la dirección de donde proviene el viento. La unidad de medida son los grados tomando como cero el norte geográfico e incrementándose en sentido horario. Una de las partes importantes en una veleta son los puntos de referencia, por ejemplo en las veletas que se encuentran arriba de algunas de casas tienen indicaciones de los puntos cardinales. En el caso de las veletas electrónicas en suelo, el punto de referencia (cero o norte) se encuentra fijo en la misma estructura. Sin embargo, en este caso al tener el sistema en un hilo a varios cientos de metros de altura no se puede tener ningún punto fijo, por lo cual se optó por utilizar una brújula electrónica.

Las brújulas electrónicas se basan en el mismo principio que las brújulas tradicionales: orientándose por medio del campo magnético terrestre. La diferencia es que una brújula electrónica no tiene partes móviles y cuenta con una forma de adquirir y almacenar los datos. Un cuadro comparativo de algunas brújulas electrónicas que se encuentran disponibles se muestra en el cuadro 3.4 que se muestra a continuación:

Tabla 3.4: Comparación de algunas brújulas electrónicas

SENSOR	HMC1052	HM55B	V2Xe PNI
Rango	0 A 360°	0 A 360°	0 A 360°
Exactitud	0.1%	-	2°
Resolución	-	5°	0.01°
Alimentación	1.8 - 20 V	4.8 - 5.2 V	3 V
Consumo	500 mA	7 mA	0.75 mA
Tipo de Salida	Analógica (V)	Digital Propio	Serie digital SPI
Precio[a]	USD $35	USD $30	USD $75

[a] http://mexico.newark.com/

Podemos notar que la brújula HCM1052 maneja un amplio rango de alimentación además de tener un precio menor, sin embargo su consumo de corriente es grande y su salida es de tipo analógica, lo que implica la inclusión de un convertidor analógico digital. En el caso de la brújula hm55b tiene un consumo menor y su salida ya es de tipo digital, sin embargo su resolución es relativamente pobre (5 grados) y únicamente trabaja a 5V. En cambio, la brújula V2Xe tiene una salida de tipo serie digital SPI, su resolución es excelente (0.01 grados) y utiliza una alimentación de 3V con un consumo muy bajo. Esta combinación de características la hacen el dispositivo ideal a utilizarse.

3.1.5. Presión Atmosférica

La presión atmosférica se define como la fuerza por unidad de área que ejerce el peso de la atmósfera. La unidad básica es el pascal [Pa], pero en general se utiliza el hectopascal [hPa] que es igual a 1 milibar [mbar] (unidad utilizada en meteorología). Básicamente se utilizan dos tipos de transductores para medir la presión atmosférica: los capacitivos y los piezoresistivos.

En los transductores capacitivos, un diafragma de metal o silicio funciona como elemento sensor y es uno de los electrodos de un capacitor. El otro electrodo es rígido y normalmente está formado por una capa de metal sobre un sustrato de cerámica o vidrio. La presión aplicada sobre el transductor causa una flexión en el diafragma, lo cual hace variar el espacio entre los electrodos y esto provoca un cambio en el valor de la capacitancia.

Los transductores de presión piezoresistivos se basan en un cambio en la resistencia eléctrica de un material cuando es presionado o estirado. El material piezoresistivo se incorpora al diafragma para convertir la flexión provocada en el diafragma en un cambio de resistencia eléctrica, este tipo de transductores son muy sensibles a la temperatura por lo que requieren de una compensación.

La tabla 3.5 presenta las características de algunos sensores que se encuentran en el rango requerido (al menos hasta 1050 mBar > presión a nivel del mar):

Tabla 3.5: Comparación de algunos sensores de presión

SENSOR	MPX4115A MOTOROLA	XPC15ATC HONEYWELL	NPP-301A-100AT NOVASENSOR	MS5540B INTERSEMA
Rango	150 a 1150 mBar	138 a 1034 mBar	0 A 1100mBar	10 a 1100 mBar
Exactitud	±1.5 mbar		±2.0 mBar	±1.5 mBar
Resolución	NA	NA	NA	0.1 mBar
Estabilidad	-	-	2.0 mBar/año	1 mBar/año
Alimentación	5 V	3V	3V	3V
Tipo de Salida	Tensión (V)	Tensión (mV)	Tensión (mV)	Digital Serial
Precio [a]	USD$16.68	USD$38.50	USD$26.70	USD$20.61 [b]

[a] http://mexico.newark.com/
[b] Precio del fabricante

Los sensores de presión mostrados son muy similares en cuanto rango, exactitud y estabilidad, sin embargo el sensor MS5540 de la compañía Intersema además de requerir únicamente una alimentación de 3 V, tiene una salida de tipo serie digital, la cual no requiere de un circuito externo de acondicionamiento ni un convertidor A/D. Además el precio del sensor MS5540 es de los más bajos. Claramente el sensor MS5540 es la mejor opción, combinando las mejores características con un precio bajo.

3.1.6. Altura.

Un método para aproximar la altura a la cual se encuentra el globo es igualarla con la longitud de hilo que es soltado por el malacate, sin embargo dado que la mayoría de las veces el hilo tiende a colgarse formando una curva en lugar de una recta, esta aproximación es muy burda. Existen otras aproximaciones que mejoran la anterior con ayuda de teodolitos y algunos cálculos trigonométricos. Sin embargo un método mucho más práctico para calcular la altura es utilizar la presión atmosférica. Debido a que el aire es compresible, las zonas mas cercanas a la superficie son más densas que las que se encuentran a mayores altitudes, esto sumado a las variaciones de temperatura provoca una relación no lineal de la presión atmosférica y la altura.

3.1.7. Almacenamiento.

Una parte importante del sistema es el respaldo de los datos en una memoria no volátil: en caso de que por alguna razón se pierda la comunicación, la memoria permite la recuperación posterior de los datos. Las memorias de tipo EEPROM (ROM programable y borrable eléctricamente) cuentan con características óptimas: se borran y escriben de manera eléctrica, pueden ser reprogramadas al menos unas 100,000 veces, suelen comunicarse mediante protocolos seriales y existe una amplia variedad. La memoria EEPROM 24lc512 de Microchip tiene la ventaja de trabajar con niveles bajos de tensión así como un bus de tipo serie, lo que reduce su tamaño. Sus características principales son las siguientes:

- Amplio rango de alimentación: 2.5 -5.5 V
- Bajo consumo: 5 mA a 5.5 V
- Comunicación serial I2C
- 1,000,000 de ciclos de escritura
- Disponibilidad en el país

3.1.8. Base de tiempo.

La base de tiempo es una señal que nos indica el periodo mínimo en el que puede realizarse una medición. Para la mayoría de los propósitos meteorológicos, un periodo mínimo de un segundo es suficiente.

Suponiendo una velocidad de ascenso media de 1.5 m/s (siendo esta una velocidad típica) se tendría un dato al menos cada 2 m, con lo cual resultaría un conjunto de datos bastante detallado.La base de tiempo puede ser generada por algun oscilador externo o incluso por el propio microcontrolador por medio de un oscilador interno. Una de las opciones que existen para generar una base de tiempo es un reloj de tiempo real, que es un dispositivo capáz de llevar la fecha y hora actual y además generar una señal a una frecuencia determinada. El DS1307 es un reloj de tiempo real que al contar con un bus de comunicacion de tipo serie se encuentra en un encapsulado pequeño adecuado a la aplicación. Sus características principales son:

- Bajo consumo de corriente: 1.5 mA.
- Comunicación de tipo serie: I2C.
- Disponibilidad en el país.
- Encapsulado pequeño: DIP8.
- Bajo costo.

3.1.9. Transmisión de datos.

Actualmente existen diferentes opciones para transmitir información via radio frecuencia (ver tabla 3.6). Sin embargo la mayoría tienen un alcance muy limitado, y no suficientemente bueno para esta aplicación, o son extremadamente caros.

Tabla 3.6: Comparación entre diversos tipos de módulos de transmisión RF

Módulo	TWS434	nRF2401A	Xbee PRO
Frecuencia	433.92 MHz ASK	2.4 GHz GFSK	2.4 GHz
Alimentación	3 V – 12 V	3 V	3V
Alcance (línea de vista)	120 m	410 m	1600m
Tamaño	10.5 x 15 mm	13 x 17 mm	24.38 x 32.94 mm
Consumo	-	-	215 mA
Precio	MX$95[a]	MX $450[b]	MX $570[c]

[a] Precio www.robodacta.com.mx
[b] Precio www.robodacta.com.mx
[c] Precio www.agelectronica.com.mx

El módulo Xbee-PRO (figura3.8) cuenta con características que lo hacen idóneo: su alimentación es de 3V, transmite de manera transparente el protocolo RS232 haciendo muy sencillo su manejo, su alcance es mayor a 1 km y su precio no es exagerado.

- Alimentación baja: 3V
- Comunicación bidireccional serie RS232
- Frecuencia de transmisión: 2.4 GHz
- Potencia de transmisión:63 mW
- Consumo de corriente: 215 mA en transmisión, 50 mA en recepción.
- Peso 3g.

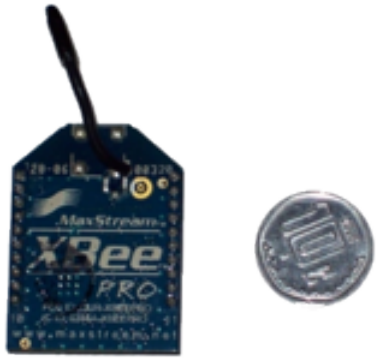


Figura 3.8.: Módulo de transmisión XBee PRO

3.1.10. Adquisición de datos

Actualmente existen muchos adquisitores de datos comerciales, sin embargo, la gran mayoría están totalmente enfocados a la captura de señales de tipo analógicas. Dado que los sensores elegidos en su mayoría utilizan un bus de tipo serie, es necesario realizar el diseño de un adquisitor propio que capture estas señales por medio de un microcontrolador. Un microcontrolador es un circuito integrado programable que realiza las funciones básicas de un sistema de cómputo (procesamiento y almacenamiento de información). Con capacidad reducida y con periféricos de entrada-salida que le permite conectarse fácilmente a otros dispositivos. Los componentes típicos de un microcontrolador son: una unidad central de procesamiento, memoria, puertos de entrada/salida, contadores y convertidores A/D (ver figura 3.54).

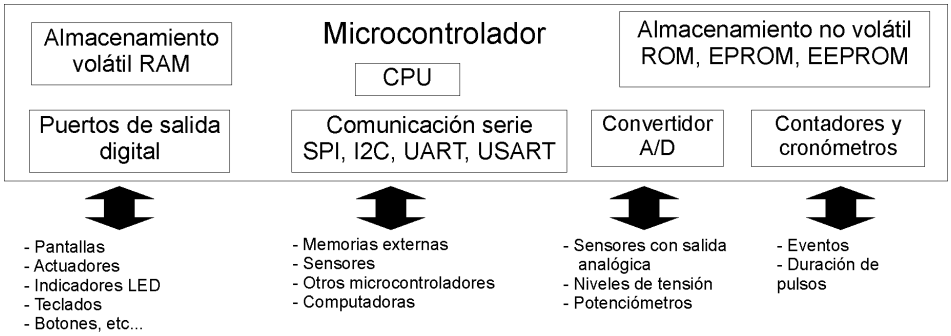


Figura 3.9.: Componentes típicos de un microcontrolador

Elección de microcontrolador Uno de los puntos mas importantes en el diseño de sistemas basados en microcontroladores es la elección del microcontrolador adecuado al proyecto en cuestión, para lo cual es necesario tener en claro las características que debe cubrir. En este caso debe cumplir con lo siguiente:

- Tamaño reducido pero de fácil reemplazo y con los pines suficientes para poder manejar los buses de comunicación y otras señales que se requieren.
- Al menos una interrupción que permita sincronizar el muestreo de datos mediante el reloj de tiempo real.
- Capacidad de manejo de protocolos de tipo serie.
- Amplia memoria de programa, ya que se requiere programar protocolos de comunicación y operaciones con punto flotante, lo cual es muy demandante en cuanto a número de instrucciones.
- Nivel bajo de alimentación (al menos 3V)

Se decidió utilizar un microcontrolador de la familia PIC de la compañía Microchip debido a que reúnen ciertas características que los hacen una buena elección:

- Buena relación precio/capacidades.
- Sencillez de manejo.
- Módulos de hardware para manejo de protocolos de tipo serie.
- Bajos niveles de alimentación (desde 2.2 V).
- Amplia información en cuanto a funcionamiento como ejemplos de uso, tanto en libros como en internet.
- Un gran número de herramientas de desarrollo (hardware y software), algunas incluso gratuitas.
- Gran disponibilidad, dada su gran aceptación es fácil su adquisición a nivel mundial.

La compañía Microchip fabrica una muy extensa variedad de microcontroladores y uno de los parámetros a tomar en cuenta para la elección es el número de pines. Realizando el conteo mostrado en la tabla 3.7, se puede observar que se requiere de al menos 21. La configuración que cubre este número de líneas es un microcontrolador de 28 pines. Otro parámetro a tomar en cuenta es el tamaño de la memoria de programa debido a que a mayor memoria se puede utilizar un mayor número de instrucciones y esto permite programar más rutinas o realizar rutinas más complejas.

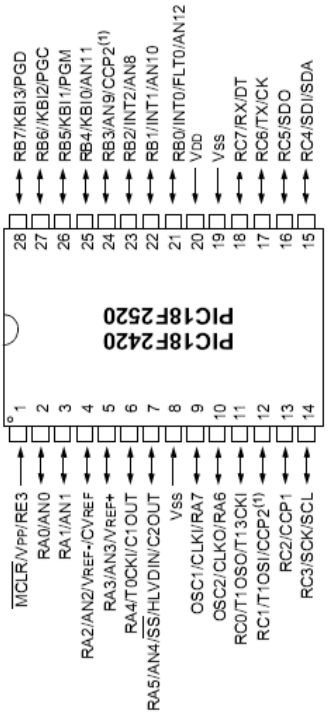
Tabla 3.7: Conteo de pines necesarios a utilizar

Uso	Pines necesarios
Bus serie sht75	2 pines
Bus SPI	5 pines
Bus I2C	2 pines
Contador de eventos	1 pin
Sincronización de muestreo	1 pin
Bus RS 232	2 pines
Led indicador	1 pin
Boton calibración	1 pin
Circuito para oscilador	2 pines
Circuito de reinicio	1 pin
Canales analógicos	1 pin (al menos)
Alimentación	2 pines
Total	21 pines

El microcontrolador que cumple con las especificaciones requeridas es el PIC18f2520 en su versión de 28 pines (figura 3.10b). Sus características principales se enlistan en la figura 3.10a.

- Amplio rango de operación: de 2.0 V a 5.5 V
- Gran capacidad de memoria: 32KB (16384 palabras de instrucción)
- 25 pines de entrada/salida
- Contadores: 1 de 8 bits y 3 de 16 bits
- Tres interrupciones externas
- Módulo de hardware para utilizar I2C Y SPI
- Módulo de hardware para utilizar RS232
- Convertidor analógico de 10 bits
- Memoria EEPROM de 256 bytes
- 2 puertos de 8 líneas (puerto B y puerto C)
- 1 Puerto de 6 líneas (puerto A)

a) Principales características



b) Patigrama

Figura 3.10.: PIC 18F2520

Además de la selección del microcontrolador adecuado se debe realizar la elección de las herramientas con las cuales se desarrollará el programa del microcontrolador. Básicamente las podemos dividir en hardware y software:

Software. El fabricante proporciona el software MPLAB de manera gratuita. Este permite realizar la programación de todos los PIC's en lenguaje ensamblador, la simulación de los programas desarrollados, así como la programación en el microcontrolador de manera directa utilizando las herramientas de hardware proporcionadas por la misma compañía.

El desarrollo de programas en lenguaje ensamblador es la forma más "básica" de trabajar con un microcontrolador. Implica un alto conocimiento del funcionamiento interno del microcontrolador y, dependiendo de la complejidad del sistema, la programación es sumamente complicada en especial cuando se trabaja con operaciones de punto flotante. Afortunadamente, actualmente se cuenta con lenguajes de un nivel mas alto como C o Basic para PIC. Estos lenguajes contienen rutinas que facilitan el desarrollo y evitan trabajar en lenguaje ensamblador; lo que trae consigo importantes ventajas:

- No se requiere un conocimiento a fondo del microcontrolador,
- Portabilidad del programa desarrollado para diferentes PIC (esto permite cambiar de PIC, en caso de ser nesario, sin preocuparse de si al migrar de dispositivo seguirá funcionando de la misma manera),
- La complejidad del programa es menor y por ende su depuración es más sencilla,
- Se ahorra tiempo en el desarrollo debido a la disminución de la complejidad de los programas
- Son mucho más fáciles de aprender que el lenguaje ensamblador.

Su principal desventaja es que, dado que maneja rutinas "genéricas" y no hechas específicamente para una aplicación, el programa resultante puede ocupar más espacio en memoria que el que se ocuparía programando rutinas específicas en lenguaje ensamblador. Sin embargo, esta desventaja es relativa ya que cuando el programa es demasiado complejo, las rutinas requeridas suelen ser más difíciles de realizar. En este caso, a menos de que se cuente con una amplia experiencia en programación en lenguaje ensamblador, la rutina resultante realizada en ensamblador puede ocupar más memoria que una rutina genérica de un lenguaje de nivel alto.

El lenguaje PIC C de la compañía CCS es una de las opciones más ampliamente utilizadas debido a que se basa en el lenguaje C para computadoras, siendo éste último un lenguaje de gran difusión y la migración a PIC C es sencilla. El lenguaje PIC-C soporta una amplia variedad de PIC's y puede adaptarse al entorno proporcionado por el software MPLAB e incluso ser compilado y programado directamente en este, facilitando ampliamente estas tareas.

Hardware para programación de PIC. Existen diversas herramientas tanto proporcionadas por el fabricante como otras gratuitas cuyo diseño se encuentra fácilmente en internet y puede armarse incluso en casa. Lo más adecuado es utilizar las herramientas fabricadas por Microchip con lo cual se garantiza la compatibilidad y facilidad de manejo en conjunto con el software de desarrollo MPLAB. Por lo anterior, se eligió el programador PICSTART® Plus, que es un programador de bajo costo capaz de programar la mayoría de los microcontroladores PIC con encapsulado de tipo DIP y opera por medio del puerto serie de la computadora.

En resumen, las herramientas utilizadas son:

Entorno de desarrollo: MPLAB

Lenguaje: PIC-C de CCS

Programador: PICSTART® Plus

3.2. Etapa de medición

3.2.1. Medición de temperatura y humedad

3.2.1.1. El sensor SHT75

El SHT75 requiere de una alimentación en el rango entre 2.4 V y 5.5 V y trabaja con un bus de comunicación de tipo serie. Su circuito de funcionamiento únicamente requiere de la conexión de cuatro pines: dos son de alimentación: GND y VCC; una línea de reloj (CLK), que se utiliza para sincronizar la comunicación con el microcontrolador; y una línea de datos bidireccional (DATOS) para enviar y recibir información. Como componentes externos se necesita un par de resistencias conectadas a VCC (pull-up) para las líneas de reloj y datos, ver figura 3.11.

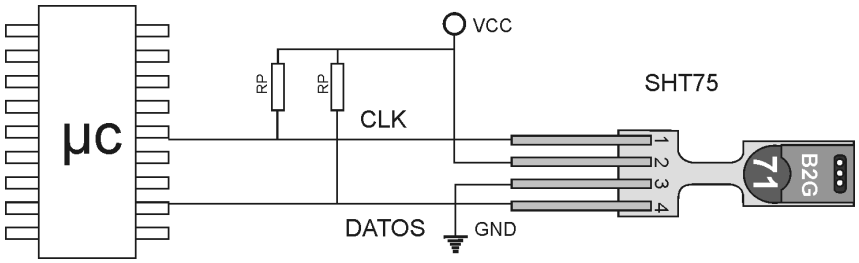


Figura 3.11.: Circuito de conexión del sensor SHT

Bus de comunicación del SHT75

Debido a que el protocolo de comunicación está hecho especialmente por el fabricante para este sensor, no existen módulos de hardware implementados en los microcontroladores, ni rutinas de software implementados en PIC-C. Por lo cual se hace necesario no sólo explicar los detalles del protocolo sino también los detalles de su programación.

Parámetro	mín	tip	máx
TCLKX	100ns		
TV		250ns	
TSU	100ns		
THO	0	10ns	
TR/TF		200ns	

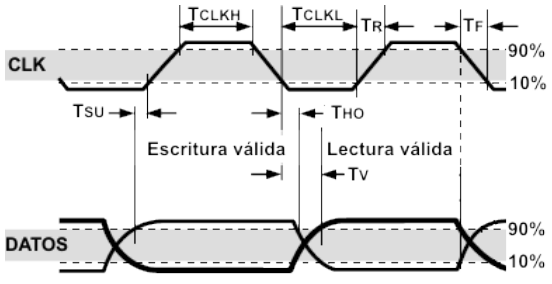
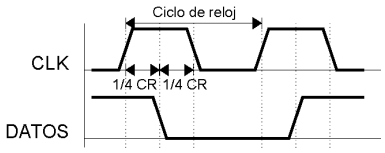


Figura 3.12.: Diagrama de tiempos del bus de comunicación del SHT75

Una vez que se polariza el sensor, se debe esperar al menos 11 ms que permitan la inicialización y la estabilización de las señales, ya que transcurre este tiempo, el sensor entra en modo de reposo esperando la secuencia adecuada para iniciar la comunicación en el bus de datos. Existen cuatro secuencias que el SHT75 debe de interpretar para su correcto funcionamiento: el inicio de transmisión, el envío de datos, la recepción de datos y el reinicio de comunicación.

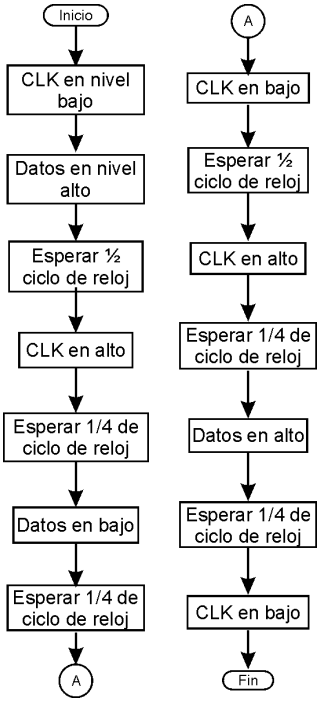
Secuencia de inicio de transmisión Consiste en llevar la línea de datos (DATOS) de un nivel alto a un nivel bajo y regresar a un nivel alto, esto mientras se realizan dos ciclos de reloj como se muestra en el diagrama de la figura 3.13a. Una forma práctica de desarrollar programas es mediante el uso de diagramas de flujo que descomponen el problema, en este caso la señal, en pasos. El diagrama de flujo correspondiente a la secuencia de inicio se muestra en la figura3.13b. Para traducir el diagrama de flujo a un lenguaje de programación específico cada bloque de proceso se convierte en varias instrucciones que realicen las acciones indicadas. El código completo correspondiente en PIC-C se encuentra en la figura 3.13.c



a) Diagrama de tiempos

```
output_high(datos);
output_low(clk);
delay_us(t_clk*2);
output_high(clk);
delay_us(t_clk);
output_low(datos);
delay_us(t_clk);
output_low(clk);
delay_us(t_clk*2);
output_high(clk);
delay_us(t_clk);
output_high(datos);
delay_us(t_clk);
output_low(clk);
```

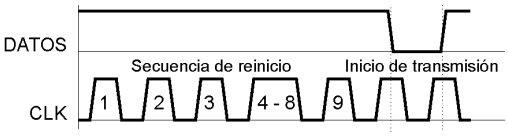
c) Código en C



b) Diagrama de flujo

Figura 3.13.: Secuencia de inicio de transmisión del SHT75

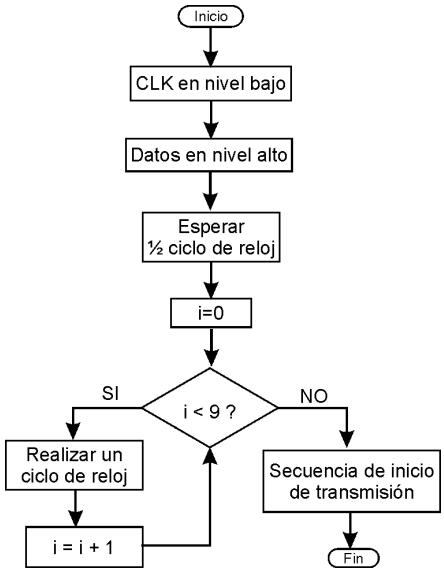
Secuencia de reinicio La secuencia de reinicio consiste en realizar 9 ciclos de reloj mientras se mantiene la línea de datos en un nivel alto como se muestra en el diagrama de tiempos de la figura 3.14.a. Seguido a esto se ejecuta la secuencia de inicio de transmisión. Una forma de realizar esto es colocar la línea DATOS en nivel alto y repetir nueve veces un grupo de instrucciones que realicen el ciclo de reloj. Para llevar la cuenta de los ciclos de reloj se puede utilizar un contado (típicamente i) como se muestra en el diagrma de flujo de la figura 3.14b. El equivalente de este conjunto ciclo repetitivo+acumulador en lenguaje PIC-C es un ciclo de tipo FOR, el código resultante en PIC-C se describe en la figura 3.14.c.



a) Diagrama de tiempos

```
output_low(clk);
output_high(datos);
delay_us(t_clk*2);
for(i=0;i<9;i++)
{
    output_high(CLK);
delay_us(t_clk*2);
output_low(clk);
delay_us(t_clk*2);
}
ini_trans();
```

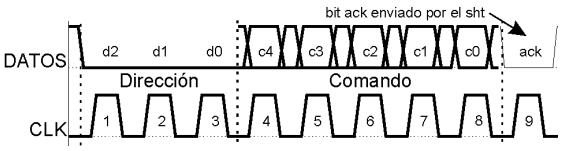
c) Código en C



b) Diagrama de flujo

Figura 3.14.: Secuencia de reinicio de comunicación del SHT 75

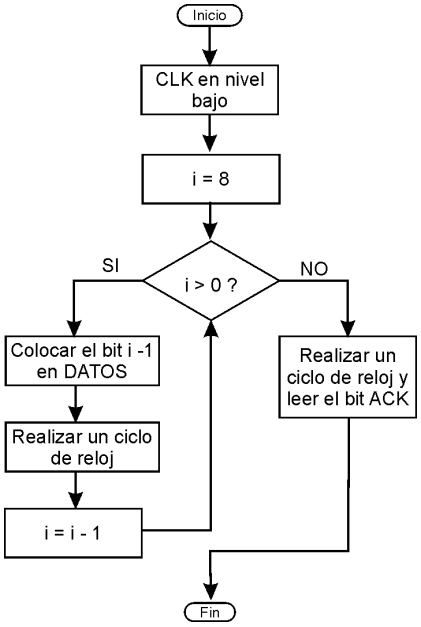
Envío de información al SHT75 Las secuencias de envío de datos (o de comando) constan de un byte con la siguiente estructura: los tres bits mas significativos (d0, d1, d2) representan la dirección y los cinco bits restantes (C0, C1, C2, C3, C4) representan la acción a realizar (bits de comando). Los bits de dirección siempre serán cero ('000'); los bits de comando pueden ser: '00011' para realizar una medición de temperatura y '00101' para realizar una medición de humedad relativa. Cuando el SHT75 recibe un byte correctamente responde colocando la línea DATOS en un nivel bajo en el noveno pulso de reloj. Una vez terminado el este pulso libera la línea para recibir más datos como se muestra en la figura 3.15 a. A esto se le llama bit de confirmación de recepción (ACK).



a) Diagrama de tiempos

```
output_low(clk);
for (i=8;i>0;i--){
if (bit_test(envia,i-1))
  output_high(datos);
else
  output_low(datos);
delay_us(t_clk*2);
output_high(clk);
delay_us(t_clk*2);
output_low(clk);}
output_float(datos);
delay_us(t_clk*2);
output_high(clk);
ack=input(datos);
delay_us(t_clk*2);
output_low(clk);
```

c) Código en C.



b) Diagrama de flujo

Figura 3.15.: Envío de información al SHT75

Se requiere ejecutar varias veces un pulso de reloj al mismo tiempo que se manipula la linea de datos: primero colocando el byte de comando y luego obteniendo el bit de confirmación. Por lo tanto se utilizará un ciclo que realice ocho repeticiones en cada una de las cuales debe colocar un bit y al final se ejecutara un noveno pulso para obtener el bit de confirmación. Según el diagrama de tiempos de la figura 3.15, el primer bit que se debe colocar es el mas significativo (bit 7) y el último es el menos significativo (bit 0); en consecuencia el acumulador utilizado deberá ir disminuyendo en cada ciclo desde el pulso 8 hasta el pulso 1, (i=8 hasta i=1); y entonces el bit a colocar en cada ciclo será el bit (i-1).

El diagrama de flujo correspondiente a la secuencia de envio queda como en la figura 3.15.b. La traducción de este diagrama de bloques en lenguaje PIC-C es muy parecido al anterior: colocando las líneas del bus en el nivel requerido y realizando un ciclo de repetición de tipo FOR. Para colocar el bit (i - 1) primero debemos conocer su valor, lo cual lo logramos con la instrucción bit_test(BYTE, BIT). Esta instrucción determina el valor de un bit específico (representado por BIT) en un byte (representado por BYTE). Si su valor es un 1 la línea DATOS debe colocarse en alto, de no ser así se colocará en bajo.

Finalmente para la lectura del bit ACK, únicamente es necesario colocar la línea DATOS en modo de entrada mediante la instrucción *output_float()* y ejecutar un ciclo de reloj. El código completo en PIC-C se muestra en la figura 3.15c. Una vez que se realiza una petición de medición el microcontrolador debe esperar a que el proceso de medición sea terminado para enviar la señal de reloj. Para una resolución de 14 bits, (que es la mejor resolución), se toma un tiempo máximo de 320 ms.

Recepción de datos del SHT75 Mientras se realiza la medición el SHT75 ignora las señales que se le envíen. Cuando los datos se encuentran listos, el módulo coloca la línea DATOS en un nivel bajo indicando que la señal de reloj puede continuar, en seguida transmitirá dos bytes de datos y un byte de comprobación de errores. Al final de cada byte enviado, el microcontrolador debe enviar el bit de confirmación de recepción poniendo la línea DATOS en un nivel bajo. Si el byte de comprobación de errores no es necesario, como en este caso, se puede terminar la comunicación después del segundo byte de datos manteniendo el bit ACK en un nivel alto, como se muestra en el diagrama de tiempos de la figura 3.16.

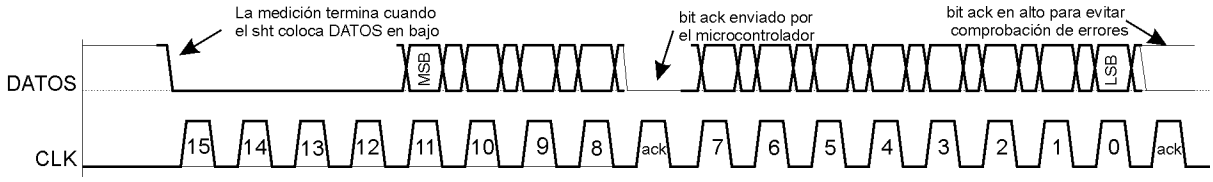


Figura 3.16.: Diagrama de tiempos de la recepción de datos en el SHT75

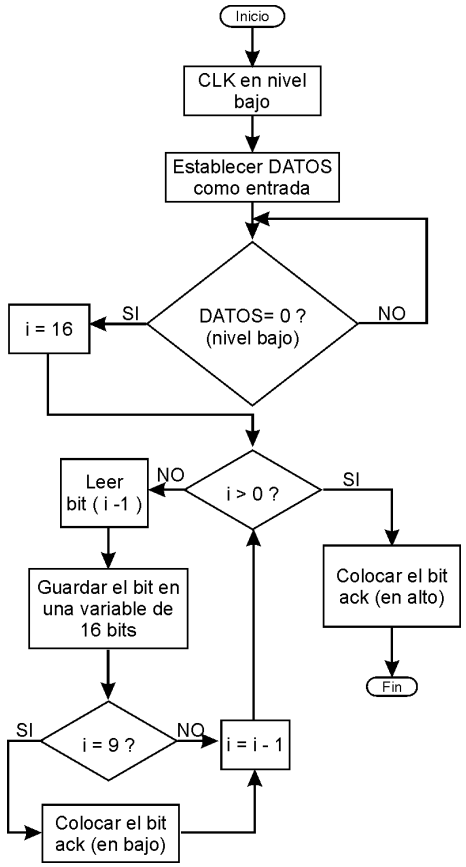
Entonces las acciones a realizar son:

- Esperar a que el módulo coloque la línea DATOS en bajo.
- Leer el primer byte de datos.
- Enviar el bit ACK adecuado.
- Leer el segundo byte de datos.
- Enviar el bit ACK adecuado.

Una forma de realizar esta secuencia es repetir 16 veces un ciclo en el que se realice el pulso de reloj y la lectura de un bit, y agregar de manera extra los pulsos de reloj correspondientes a los bits ACK. Como los bits deben leerse en la parte alto del pulso de reloj, primero debe realizarse sólo una parte de la señal (hasta la mitad de la parte alta del pulso), leer el bit y en seguida realizar el resto de la señal de reloj.

Los bits mas significativos corresponden con los pulsos del 16 al 9 (comenzando a contar desde el 16 en forma descendente), por lo que en cada ciclo se debe revisar si ya se realizó el noveno pulso para así colocar el bit ACK. Tomando en cuenta esto el diagrama de flujo queda como en la figura 3.17b.

```
output_low(clk);
output_float(datos);
while(input(datos));
for(i=16;i>0;i--){
  delay_us(t_clk*2);
  output_high(clk);
  delay_us(t_clk);
  bit_leido=input(datos);
  delay_us(t_clk);
  output_low(clk);
  if (bit_leido)
bit_set(bytes_leidos,i-1);
  else
    bit_clear(bytes_leidos,i-1);
    if(i==9){
  output_low(datos);
   delay_us(t_clk*2);
   output_high(clk);
   delay_us(t_clk*2);
   output_low(clk);
output_float(datos);}}//for
delay_us(t_clk*2);
output_high(clk);
delay_us(t_clk*2);
output_low(clk);}
```



a) Código en C

b) Diagrama de flujo

Figura 3.17.: Recepción de datos del SHT75

En lo que corresponde a la traducción a lenguaje PIC-C, los cambios de nivel en las lineas (como los ciclos de reloj) se realizan mediante las instrucciones *output_low()* y *output_high()* se utiliza un ciclo de tipo FOR como ciclo de repetición principal; para verificar el valor de una línea (pin) se utiliza la instruccion *Input()*, sin embargo para permanecer verificando hasta que cambie de valor se utiliza un ciclo de tipo WHILE en el que el valor de la línea se utiliza como condición: *while(input())*. Con esta instrucción, si la línea se encuentra en nivel alto, la funcion *input()* devolvera el valor 1, haciendo que el ciclo se repita volviendo a verificar la línea hasta que se coloque en un nivel bajo.

La lectura de los bits se realiza de la siguiente manera: cuando el pulso de reloj se encuentra en la parte alta se revisa el estado de la línea DATOS mediante la instruccion *bit_leido=input(datos);* donde *bit_leido* es una variable de un bit.

Siguiendo el diagrama de flujo, se debe terminar de realizar el pulso de reloj y a continuación guardar el valor contenido en la variable *bit_leido* en el bit i -1 de la variable de 16 bits destinada para esto, en este caso la variable se llama *bytes_leidos*. Si la variable *bit_leido* vale 1, dicho valor se coloca en la variable mediante la instrucción: *bit_set (bytes_leidos,i-1)*, en caso contrario un 0 se coloca con la instrucción: *bit_clear (bytes_leidos,i-1)*. Cuando se han obtenido y guardado los 16 bits el ciclo FOR termina y se realiza el ultimo pulso de reloj para colocar el segundo pulso ACK y dar por terminada la comunicación. Dado que el bus es de tipo colector abierto las líneas se encuentran normalmente en un nivel alto y como el bit a colocar es un 1, bastará con ejecutar el pulso de reloj sin necesidad de manipular la línea DATOS. El código completo se encuentra en la figura 3.17a. Una vez que la comunicación termina el dispositivo entra nuevamente en modo de espera. Un detalle importante para la correcta adquisición de datos es su tipo. En este caso son del tipo enteros no signados de 16 bits.

3.2.1.2. Proceso de medición

Inicialización El módulo incluye una configuración predeterminada solo se debe establecer la duración de un ciclo de reloj adecuada para el bus antes de iniciar cualquier comunicación con el sensor. Si la velocidad máxima permitida por el sensor es de 1MHz, entonces la duración mínima de un ciclo es de 1us. Una duración recomendable por experiencia es de un ciclo es de 4us, por lo tanto el código en PIC-C para establecer esto es:

t_clk=1;

donde:

t_clk es la variable utilizada en las funciones anteriores representando a 1/4 del ciclo de reloj (en us). Es decir:

Ciclo de reloj = 4 * t_clk

Medición de datos Para comenzar la comunicación el microcontrolador debe realizar una secuencia de inicio de transmisión, sin embargo es recomendable que envíe antes una secuencia de reinicio de comunicación para asegurar una correcta transmisión de datos.

A continuación debe enviar la secuencia de envió de datos con el comando que se desee ejecutar (medición de temperatura, medición de humedad o acceso al registro de estado) seguido por la secuencia de recepción de los datos resultantes. Sólo se podrá enviar una de las tres opciones, si se requiere medir la temperatura y la humedad es necesario enviar las secuencias dos veces: una para medir la temperatura y otra para medir la humedad. Véase la figura 3.18. El acceso al registro de estado se utiliza para realizar la configuración de SHT75 y el uso de funciones extras.

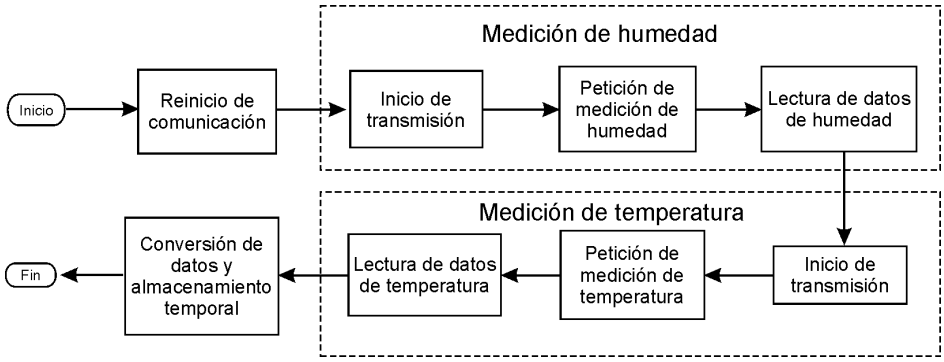


Figura 3.18.: Diagrama de flujo de medición de temperatura y humedad

Como se ilustra en la figura 3.18, una secuencia de medición, ya sea de temperatura o humedad, se puede dividir en dos:

1. El envío de información al SHT75 realizando la petición de medición y;
2. La recepción de los datos resultantes.

Así como en el diagrama de flujo cada bloque representa una secuencia, en el código en PIC-C cada secuencia puede ser representada por una función, lo que facilita una depuración o modificación posterior. Siguiendo el diagrama de flujo, el código en PIC-C utilizando funciones queda de la forma mostrada en la figura 3.19.

```
reinicio_com();                          Realiza la secuencia de reinicio
ini_trans();                Realiza la secuencia de inicio de transmisión
envia_byte(0b00000011);                            Petición de temperatura
temperatura=lee_2bytes();                Lee los dos bytes de temperatura
ini_trans();                           Secuencia de inicio de transmisión
envia_byte(0b00000101);                                 Petición de humedad
humedad=lee_2bytes();                       Lee los dos bytes de humedad
```

Figura 3.19.: Código en PIC-C que realiza la medición de temperatura y humedad

Por medio de este código obtenemos los valores en crudo de las mediciones de temperatura y humedad, y las guardamos en variables de 2 bytes. Sin embargo estos valores aun no se encuentran en las unidades correspondientes por lo que requieren de un proceso para convertirlos.

Conversión de datos a valores de temperatura y humedad El fabricante, mediante hojas de especificación, nos indica que para convertir los datos en valores de ºC para la temperatura y en %HR para la humedad deben realizarse las siguientes operaciones:

Para la temperatura:

$$Temperatura = d1 + (d2 \cdot SO_T) \tag{3.1}$$

Donde:

SO_T es el número de dos bytes obtenido del módulo después de enviar el comando de medición de temperatura.

Y los coeficientes d1 y d2 están dados por:

VDD	d_1 [°C]	d_1 [°F]
5 V	-40.00	-40
4 V	-39.75	-39.55
3.5 V	-39.66	-39.39
3 V	-39.60	-39.28
2.5 V	-39.55	-39.19

SO_T	d_2 [°C]	d_2 [°F]
14 bits	0.01	0.018
12 bits	0.04	0.072

Si tenemos una alimentación de 3V y una resolucion de 14 bits, la ecuación queda de la siguiente forma:

$$Temperatura(ºC) = -39.60 + 0.01 \cdot SO_T \tag{3.2}$$

En forma similar, para obtener la humedad relativa y con objeto de compensar la no linealidad del sensor de humedad se debe utilizar la siguiente fórmula:

$$RH_{LINEAL} = c1 + c2 \cdot SO_{RH} + c3 \cdot (SO_{RH})^2 \tag{3.3}$$

Donde SO es el valor obtenido del sensor y dependiendo de la resolución los coeficientes c1, c2 y c3 son:

SO_{RH}	c_1	c_2	c_3
12 bits	-4	0.0405	$-2.8 * 10^{-6}$
8 bits	-4	0.648	$-7.2 * 10^{-4}$

Por lo que utilizando una resolucion de 12 bits la ecuación queda:

$$RH_{LINEAL} = -4 + (0.0405 \cdot SO_{RH}) - \left[2.8 * 10^{-6} \cdot (SO_{RH})^2\right] \tag{3.4}$$

La humedad relativa cálculada con la fórmula anterior supone que la temperatura es de 25°C. En caso de que la temperatura sea diferente se debe compensar mediante la siguiente fórmula:

$$RH_{REAL} = (T_{°C} - 25) \cdot (t_1 + t_2 \cdot SO_{RH}) + RH_{lineal} \tag{3.5}$$

Donde t1 y t2 dependen de la resolución utilizada:

SO_{RH}	t_1	t_2
12 bits	0.01	0.00008
8 bits	0.01	0.00128

Por lo tanto, para una resolución de 12 bits, la ecuación queda:

$$RH_{REAL} = (T_{°C} - 25) \cdot (0.01 + 0.00008 \cdot SO_{RH}) + RH_{lineal} \tag{3.6}$$

Las ecuaciones 3.2, 3.4 y 3.6 son las que se deben programar; sin embargo, en la lectura de los datos se utilizaron las variables *temperatura* y *humedad* en lugar de SO_T y SO_{RH}. Por lo tanto, al realizar dichos cambios el código queda como en la figura 3.20:

```
temp_en_c=-39.60+(.01*temperatura);
humedad_lineal=-4+(0.0405*humedad)-(0.0000028*humedad*humedad);
humedad_comp=(temp_en_c-25)*(.01+(.00008*humedad))+humedad_lineal;
```

Figura 3.20.: Código en PIC-C de la conversión de valores crudos de temperatura y humedad a valores en ºC y %RH

Ahora se tienen almacenados de manera temporal el valor de temperatura en ºC en la variable temp_en_c y el valor de humedad en %RH en la variable humedad_comp, con lo que se finaliza esta parte de la medición.

Cabe resaltar la facilidad con que se programan las operaciones matemáticas incluso en punto flotante, en comparación con el código que debería escribirse en lenguaje ensamblador para realizar las mismas operaciones.

3.2.2. Medición de velocidad de viento

Para la medición de la velocidad del viento el tipo de anemómetro que se utiliza es el de hélice para esto se utiliza un anemómetro marca Skywatch modelo Fun el cual es muy pequeño y de tipo hélice magnética. Incluye todo un sistema que permite observar la velocidad de viento instantánea en una pantalla; sin embargo, no se requiere observar el valor del viento sino registrarlo y enviarlo a distancia. Por lo tanto, únicamente es de utilidad el transductor compuesto por la hélice y la bobina así que se recurrió a quitárselo para adaptarlo sistema. Esto se decidió debido a que el transductor (hélice y bobina) no se encontró en el mercado individualmente. Al girar, en presencia de viento, se produce un campo magnético variable que induce una diferencia de potencial en la bobina colocada cerca de las hélices. La forma de onda obtenida del transductor es de tipo senoidal.

Para encontrar la relación que existe entre pulsos y velocidad de viento, se hizo lo siguiente: una vez separado el transductor se sustituyó este por un generador de funciones simulando la señal, encontrando que el dispositivo indica una velocidad de 1 m/s cuando se tienen 17 pulsos/s. Entonces la velocidad de viento se puede calcular con la siguiente fórmula:

$$Velocidad(m/s) = \#pulsos\,por\,segundo/17$$

La señal obtenida del transductor es muy pequeña y debido a su forma no puede ser interpretada por el microcontrolador, por lo que, debe pasar por una etapa de acondicionamiento que convierta la señal senoidal en una señal cuadrada y con los niveles de trabajo del microcontrolador. Para convertir la señal se utilizó un amplificador operacional en configuración de comparador. La entrada negativa del amplificador se conecta a 0 V, mientras que en la entrada positiva se conecta la señal de la bobina; así mientras la señal senoidal sea mayor a 0V la salida del amplificador se mantendrá en un nivel alto, cuando sea menor a 0V la señal cambiará a un nivel bajo. Aunque esta señal cuadrada ya puede ser interpretada por el microcontrolador, los niveles de salida de un amplificador son bastante menores al nivel de alimentación y podrían encontrarse debajo del umbral requerido por el microcontrolador. Para solucionar el problema, se introdujo entre el comparador y el microcontrolador un buffer digital, cuyo nivel de salida alto es prácticamente el nivel de alimentación. Ver figura 3.21.

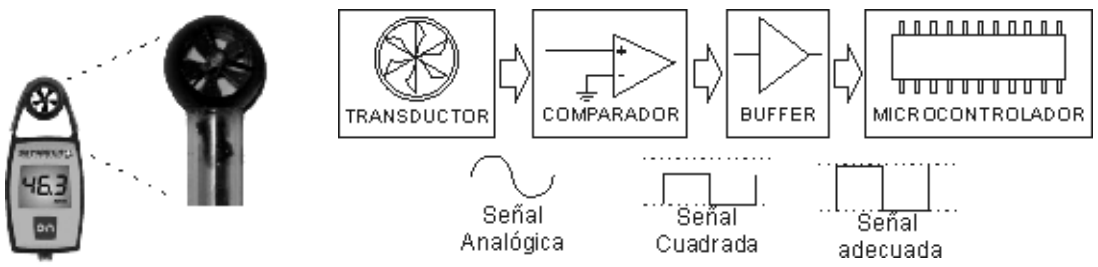


Figura 3.21.: Anemómetro Skywatch y el diagrama de bloques de su acondicionamiento

Una de las actividades más comunes a realizar en un sistema basado en microcontroladores es el conteo de pulsos o eventos, por lo que estos cuentan con un contador que libera de este trabajo al programa principal. Los microcontroladores PIC cuentan con uno o más temporizadores/contadores (únicamente contador de aquí en adelante) de 8 o 16 bits dependiendo del modelo que se esté utilizando. Un contador de 8 bits nos permite llegar hasta una cuenta de 255 , mientras que un contador de 16 bits puede llegar hasta 65535. Dado que el anemómetro es capaz de medir velocidades de hasta 30 m/s y el sensor elegido nos da 17 pulsos en cada m/s, como mínimo el contador debe ser capaz de contar 17*30=510 por lo cual es necesario utilizar un contador de 16 bits. A continuación se explica el funcionamiento y manejo básico de los contadores.

Manejo básico de los contadores Los contadores se utilizan para determinar el número de eventos (pulsos), ya sea en alguna entrada del microcontrolador o en algún oscilador interno . En el caso de que los eventos siempre tengan el mismo periodo se pueden utilizar como una base de tiempo, razón por la cual también se les llama temporizadores. En PIC-C se utilizan tres instrucciones para el manejo de los contadores: *setup_timer_X(), set_timerX()* y *get timerX()*. Donde X es el número del contador.

setup_timer_X(parámetro). Realiza la configuración del contador 0, los posibles parámetros son:

RTCC_INTERNAL Cuenta ciclos de instrucción internos

RTCC_EXT_L_TO_H Cuenta eventos en el flanco ascendente en el pin T0CKI

RTCC_EXT_H_TO_L Cuenta eventos en el flanco descendente en el pin T0CKI

RTCC_DIV_X Activa el divisor del contador por X=2,4,8,16,32,64,128 o 256

RTCC_OFF Desactiva el contador

RTCC_8_BIT Utiliza el contador en modo de 8 bits. Sino se indica este parámetro, se utiliza el contador en modo de 16 bits de manera predeterminada.

Puede indicarse uno o más parámetros separados por el símbolo "|" (OR). Cabe señalar que las opciones disponibles para cada PIC y para cada contador varían, por lo que es recomendable revisar la plantilla ".h" del microcontrolador elegido para saber exactamente las opciones que se pueden utilizar.

set_timerX(valor). Coloca un valor en el contador

valor. Es el valor a colocar y será de 8 o 16 bits (no signado) dependiendo del contador y modo usados.

valor=get_timerX(). Obtiene el valor actual del contador.

valor. Es la variable en la cual se almacena el valor del contador, será de 8 o 16 bits dependiendo del contador y modo usados.

Adicionalmente a estas instrucciones cada contador cuenta con una interrupción en caso de que se desee realizar alguna rutina cuando se desborde.

Proceso de medición

Inicialización La inicialización del contador consiste básicamente en dos acciones:

- Configurarlo para que se comporte de la manera requerida.
- Colocar en cero el valor del contador.

En este caso se requiere leer el número de pulsos externos (RTCC_EXT_H_TO_L), sin activar el divisor (RTCC_DIV_1) y utilizando un entero de 16 bits (Configuración predeterminada). Por lo que las instrucciones de inicialización quedan de la siguiente manera (para el timer0):

```
setup_timer_0(RTCC_EXT_H_TO_L-RTCC_DIV_1);

set_timer0(0);
```

Medición de datos Para conocer la velocidad es necesario leer el valor del contador en intervalos de un segundo y reiniciandolo a cero en cada lectura para obtener la lectura correcta de eventos en el siguiente intervalo. Si guardamos el número de eventos en una variable llamada *no_pulsos (*16 bits nosignada), las instrucciones de medición serían las siguientes:

```
no_pulsos=get_timer0();

set_timer0(0);
```

Conversión de datos Recordemos que la velocidad de viento en m/s la podemos obtener por medio de: $Velocidad(m/s) = \#pulsos\,por\,segundo/17$

o de otra manera: $Velocidad(m/s) = \#pulsos\,por\,segundo * 0.0589$

Por lo que, el código en PIC-C para calcular la velocidad está dada por la instrucción:

```
velocidad=no_pulsos*0.0589;
```

Donde:

no_pulsos Es una variable de tipo entero de 16 bits donde se almacena la lectura de eventos por segundo.

velocidad Es la variable de tipo flotante donde se almacena el valor de la velocidad actual.

3.2.3. Medición de dirección de viento

Módulo V2Xe

El módulo V2Xe es una brújula electrónica que obtiene la dirección midiendo el campo magnético por medio de sensores magneto-inductivos. Incluye una interfase serie compatible con el protocolo de comunicación SPI para intercomunicarse con otros dispositivos. Las ventajas que ofrece este módulo son: Alimentación de 3V; tiene un bajo consumo de corriente (consumo típico 2mA); las mediciones son compensadas en temperatura y su tamaño es pequeño (25.4 mm x 25.4 mm). Su resolución es de 0.01° y su exactitud es de 2°. Puede verse una foto en la figura 3.22

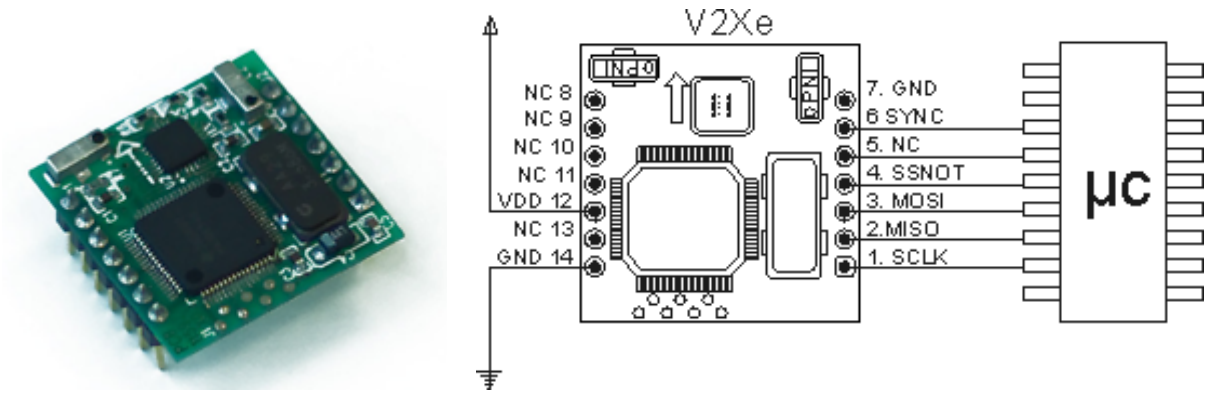


Figura 3.22.: Módulo V2Xe

Interfase de comunicación del módulo V2Xe

La interfase de comunicación consta de 6 pines divididos en dos partes:

$$Interfase\,de\,comunicación \begin{cases} -Sincronización\ de\ datos \begin{cases} -SYNC \\ -DRDY \end{cases} \\ -Bus\ de\ comunicaciones\ tipo\ SPI \begin{cases} -MOSI \\ -MISO \\ -SSNOT \\ -SCLK \end{cases} \end{cases}$$

SYNC funciona como una entrada para el módulo y le indica que es necesario reiniciar la comunicación. Esta línea normalmente se encuentra en un estado bajo, pero si el dispositivo maestro ejecuta un pulso provocará que el módulo reinicie la comunicación en los casos en que se pierda la sincronización en el bus. La pérdida de comunicación suele darse al energizar o inicializar los dispositivos.

La línea DRDY funciona como una salida del módulo y es colocada en bajo después de un pulso de reinicio de comunicación y en alto indicando que los datos se encuentra listos después de que es recibido un comando (Ver figura 3.23). Las funciones de estas dos líneas facilitan la comunicación, sin embargo, puede prescindirse de ellas y utilizar únicamente el bus SPI sin ningún problema.

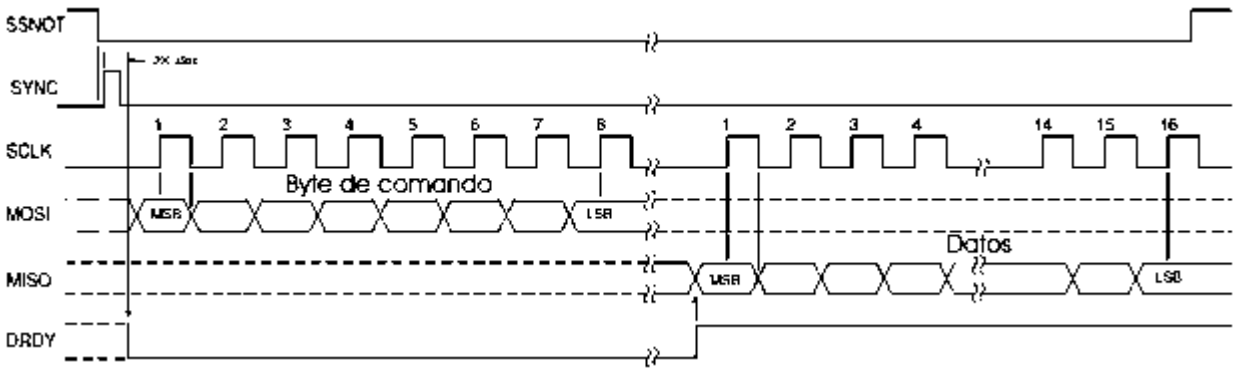


Figura 3.23.: Protocolo de comunicación del módulo V2Xe

Con el fin de facilitar la comprensión del funcionamiento del dispositivo, a continuación se explicará brevemente el estandar de comunicaciones SPI (Serial Peripherical Interfase) así como las instrucciones en PIC-C para utilizarlo.

Interfase de Periféricos Serie (SPI por sus siglas en inglés)

La SPI es una interfase serie síncrona de intercambio de datos basada en la estructura Maestro - Esclavo en la que pueden conectarse varios dispositivos en un único bus (la comunicación se entabla con un único dispositivo esclavo a la vez). Esto implica que tanto el dispositivo maestro como el esclavo en turno funcionan como transmisores y como receptores en todo momento, siempre y cuando exista la señal de reloj para sincronizarlos. Las líneas que componen el bus se describen a continuación, cabe resaltar que ninguna de estas líneas requiere de componentes externos.

- SS o CS- Es la línea de Selección de dispositivo (Slave Select o Chip Select). Debido a que se pueden conectar diversos dispositivos esclavos en el mismo bus SPI, mediante esta línea se elige el dispositivo esclavo que intercambiará datos con el dispositivo maestro.
- SCLK - En esta línea se encuentra la señal de reloj generada por el dispositivo maestro que se utiliza para sincronizar la transferencia de datos.
- MOSI - Salida del Maestro, Entrada del Esclavo (Master Out Slave In). Transporta los datos desde el dispositivo maestro hasta el dispositivo esclavo
- MISO - Entrada del maestro, salida del esclavo (Master in Slave out). Transporta los datos desde el dispositivo esclavo hasta el dispositivo maestro.

Como primera acción, el dispositivo maestro selecciona el dispositivo esclavo con el que realizará el intercambio de información poniendo en un nivel bajo la línea SS correspondiente a dicho dispositivo. A continuación, generara la señal de reloj para poder sincronizar los datos a transmitir. Únicamente el dispositivo maestro controla la señal de reloj. Los bits deben de colocarse en uno de los flancos de la señal de reloj y son leídos en el flanco contrario. Ver figura 3.24.

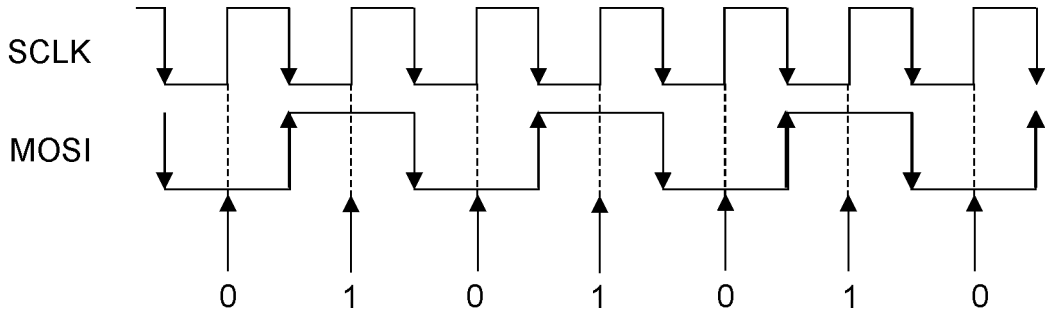


Figura 3.24.: Lectura de datos en el protocolo SPI

Cuando no hay flujo de datos, la señal de reloj debe permanecer estable ya sea en un nivel alto o en un nivel bajo, dependiendo de la configuración. A esto se le llama estado de descanzo (idle) del reloj. Como resultado de la combinación del flanco donde se coloquen los datos y del estado de descanso del reloj, se tienen cuatro configuraciones o modos posibles. Ver figura 3.19. Al estado de descanso también se le llama polaridad del reloj y suele abreviarse con POL, siendo igual a cero para indicar un nivel bajo y uno para indicar un nivel alto. De manera similar al flanco en que se leen los bits se llama fase y suele abreviarse con PHA de acuerdo a lo mostrado en la figura 3.25.

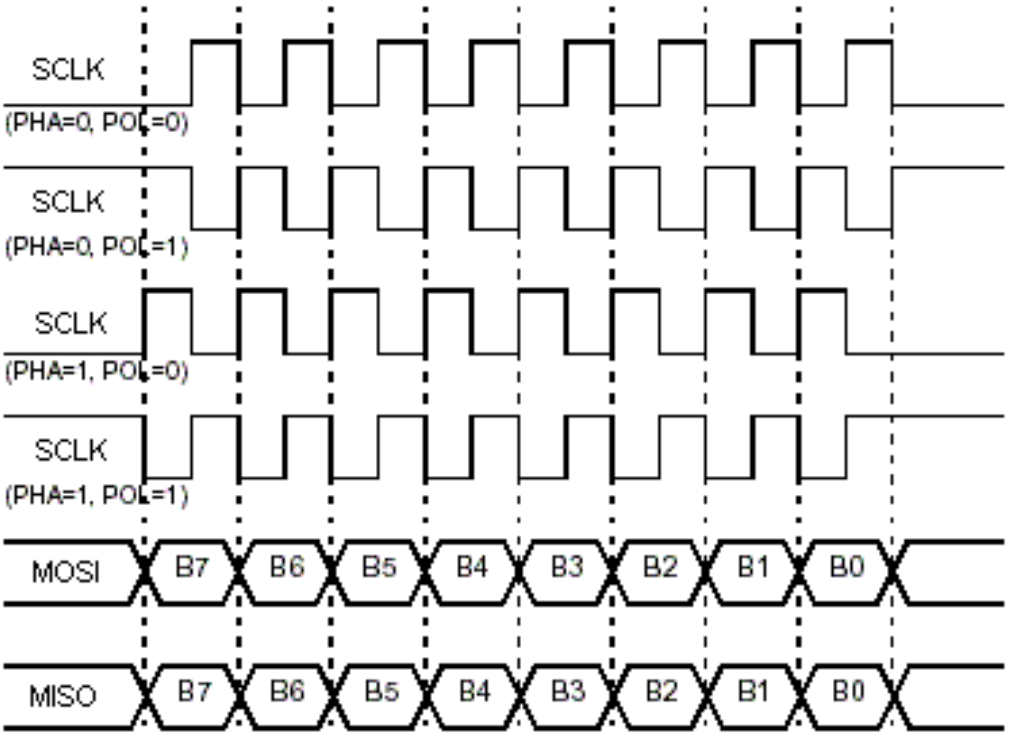


Figura 3.25.: Modos del SPI

Aunque varios dispositivos pueden conectarse al mismo bus SPI estos pueden trabajar con un modo diferente cada uno, por lo que el dispositivo maestro debe comunicarse en el modo correspondiente a cada esclavo con el fin de entablar la comunicación correctamente.

Manejo del protocolo SPI en PIC-C

Para utilizar el protocolo SPI en PIC-C, básicamente se utilizan 4 instrucciones y una directiva de preprocesamiento: *setup_spi(), spi_write(), spi_read(), spi_xfer()* y *#use spi.*

La instrucción *setup_spi()* configura el hardware del puerto SPI para trabajar con las opciones deseadas. Su sintaxis es la siguiente:

setup_spi(MODO);

Donde *MODO* puede ser alguna de las siguientes opciones (las opciones específicas disponibles para cada dispositivo se encuentran indicadas en el archivo .h de dicho dispositivo):

- SPI_MASTER, SPI_SLAVE, SPI_SS_DISABLED
- SPI_L_TO_H, SPI_H_TO_L
- SPI_CLK_DIV_4, SPI_CLK_DIV_16,
- SPI_CLK_DIV_64, SPI_CLK_T2

Para el envío y recepción de datos por medio del hardware para SPI se utilizan las instrucciones *spi_write()* y *spi_read()* su sintaxis es la siguiente:

spi_write(DATO);

DATO=spi_read();

Donde *DATO* es una variable o valor de 8 bits que es enviará por el bus o en el que se guardará el byte leído en el bus según corresponda.

En el caso de que el PIC utilizado no cuente con hardware para manejar un bus SPI o se quiera utilizar pines diferentes a los asignados por el hardware, PIC-C cuenta con una implementación por software que puede ser utilizada mediante la directiva #Use spi, su sintaxis es:

#use spi (opción1,opción2,opción3,...opciónN)

donde las opciones posibles se muestran a continuación (tabla 3.8) :

Tabla 3.8: Opciones de configuración de la directiva *#use spi*

Opción	**Acción que realiza**
MASTER	Establece el dispositivo como maestro
SLAVE	Establece el dispositivo como esclavo
BAUD=n	Velocidad en bits por segundo
CLOCK_HIGH=n	Tiempo en alto del reloj en us
CLOCK_LOW=n	Tiempo en bajo del reloj en us
DI=pin	Pin asignado a los datos de entrada
DO=pin	Pin asignado a los datos de salida
CLK=pin	Pin asignado al reloj
MODE=n	Modo en el que trabaja al SPI
Continua	Continua

Opción (Continuación)	**Acción que realiza** (Continuación)
	MODE=0 IDLE=0 Y SAMPLE_RISE
	MODE=1 IDLE=1 Y SAMPLE_FALL
	MODE=2 IDLE=1 Y SAMPLE_FALL
	MODE=3 IDLE=1 Y SAMPLE_RISE
ENABLE=pin	Pin opcional que se activará durante la transferencia de datos
LOAD=pin	Pin opcional en el que se ejecutará un pulso una vez que los datos son transferidos
DIAGNOSTIC=pin	Pin opcional que se colocará en alto cuando un dato es muestreado
SAMPLE_RISE	Muestrea en el flanco ascendente
SAMPLE_FALL	Muestrea en el flanco descendente (predeterminado)
BITS=n	Máximo número de bits en una transferencia
LOAD_ACTIVE=n	Estado activo para el pin LOAD (0 o 1)
ENABLE_ACTIVE=n	Estado activo para el pin ENABLE (0 o 1)
IDLE=n	Estado inactivo del reloj (0 o 1)
ENABLE_DELAY=n	Tiempo en us de retardo despues de que el pin ENABLE es activado
LSB_FIRST	Envía primero el bit menos significativo
MSB_FIRST	Envía primero el bit mas significativo
STREAM=id	Especifica un nombre para esta configuración
FORCE_HW	Utiliza el hardware para esta configuración

Para este caso el envío y recepción de información se hace utilizando la instrucción *spi_xfer()*, funciona tanto para leer como para enviar datos con la misma sintaxis que *spi_write()* y *spi_read()*:

spi_xfer(DATO) Envía un byte por el bus

DATO= spi_xfer() Lee un byte en el bus y lo guarda en la variable *DATO*.

Proceso de medición

Lo primero que debe hacerse es especificar en que modo se realizará la comunicación. Para el V2Xe el modo debe cumplir con las siguientes condiciones:

- El estado de descanso del reloj en nivel bajo;
- Los bits deben colocarse en el flanco descendente del reloj y deben ser validados en el flanco ascendente, como se muestra en la figura 3.26. Tanto el tiempo en bajo como el tiempo en alto deben ser mayores a 100ns.

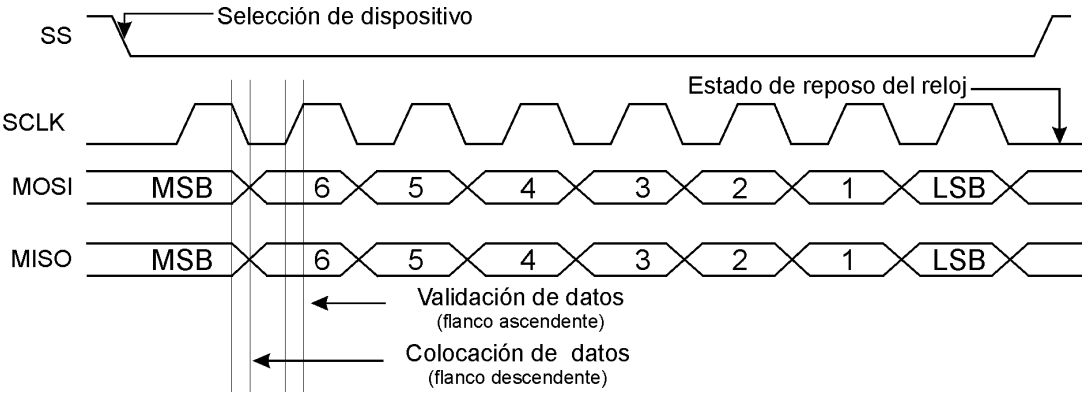


Figura 3.26.: Configuración del protocolo SPI para el V2Xe

El módulo V2Xe utiliza un amplio grupo de comandos para comunicarse, configurar e indicarle cuando debe realizar una medición, sin embargo no se necesita conocer todos para lograr el funcionamiento adecuado. Los comandos básicos que se utilizaran se muestran en la tabla siguiente (3.9):

Tabla 3.9: Comandos del módulo V2Xe

Acción	Código	Acción	Código
Establecer los datos de salida	0x03	Guardar configuración	0x09
Petición de datos	0x04	Inicio de Calibración	0x0A
Respuesta a petición de datos	0x05	Fin de calibración	0x0B
Establecer configuración	0x06	Respuesta a la petición de calibración	0x0D
Respuesta a petición de configuración	0x08	Establecer datos de calibración	0x0E

La estructura de los paquetes de información que conforman cada uno de los comandos consta de 4 partes: un byte de sincronización, un byte de comando, un grupo de bytes de datos (opcionales) y un byte de fin, ver figura 3.27.

Byte de sincronización	Byte de comando	Bytes de datos (opcionales)			Byte de fin
Byte 1	Byte 2	Byte 3	···	Byte n-1	Byte n
0xAA	0x**	0x**	···	0x**	0x00

Figura 3.27.: Estructura de información en el V2Xe

Una vez que se conoce la forma en la que se comunica el módulo, se puede continuar con la descripción del proceso de medición. A grandes rasgos, el proceso de medición se compone de: inicialización del módulo, petición de datos, lectura de datos, conversión y almacenamiento temporal y calibración. Siguiendo el diagrama de flujo de la figura 3.28. Antes de realizar cualquier rutina de medición o calibración con el módulo, se debe llevar a cabo la inicialización de acuerdo a las necesidades.

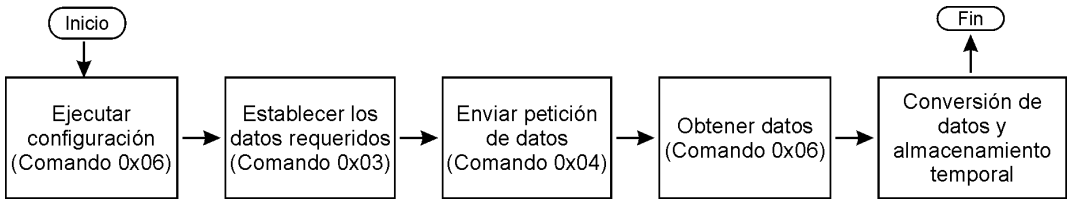


Figura 3.28.: Diagrama de flujo de medición de dirección de viento

Inicialización La inicialización se divide en dos partes: la configuración de los parámetros internos y el establecimiento de los datos que el módulo enviará al recibir una petición de medición.

Lo primero que se debe realizar es la configuración de los parámetros internos mediante el comando 0x06. Los parámetros a configurar son:

Declinación. Establece el ángulo de declinación que se utilizará para calcular el norte geográfico

Norte verdadero. Establece si la dirección será referida al norte geográfico o al norte magnético. Si el valor es verdadero, se utilizará el valor de declinación para calcular la dirección.

Frecuencia de calibración. Es la frecuencia de muestreo durante la calibración

Frecuencia de muestreo. Es la frecuencia de muestreo si se habilita el filtrado digital que incluye el módulo.

Periodo. Es el periodo que se utilizará en el circuito especializado que se encarga de realizar las mediciones.

BigEndian. Indica el formato que se manejará en las variables (flotantes y enteros). Si se establece en verdadero, los datos se manejarán en el formato big endian, de lo contrario se manejaran en formato little endian.

Factor de amortiguación. Indica el número de muestras que se promediaran para calcular la dirección. El módulo calculará el promedio de los últimos n valores indicados en este parámetro y lo enviará como la dirección actual.

Cada uno de estos parámetros internos tiene asignado un identificador, un formato y un valor predeterminado como se puede ver en la tabla 3.10.

Tabla 3.10: Parámetros de configuración

Parámetro	Identificador	Formato	unidades(rango)	Predeterminado
Declinación	0x01	Flotante	grados (-180° a 180°)	0º
Norte verdadero	0x02	Booleano	verdadero o falso	falso
Frecuencia de calibración	0x03	Entero 8(u)	Hz (1 a 8)	8
Frecuencia de muestreo	0x04	Entero 8(u)	Hz (0 a 8)	0
Periodo	0x05	Entero 8(u)	1 a 8	5
BigEndian	0x06	Booleano	Verdadero o falso	verdadero
Amortiguación	0x07	Entero 8(u)	1 a 8	1

Dado que el módulo cuenta con valores predeterminados, no es obligatorio configurar todos los parámetros sino únicamente los que exija la aplicación. Los parámetros a configurar se deben indicar después del código de comando enviando el identificador del parámetro de configuración seguido por el valor que se le quiere dar. Ver figura 3.29.

Byte de sincronización	*Byte de comando*	*Bytes de datos*		*Byte de fin*
		Identificador	*Valor*	
0xAA	0x06	0x**	***	0x00

Figura 3.29.: Sintáxis general del comando 0x06

Los valores predeterminados son los adecuados para la aplicación por lo que se puede omitir esta parte de la inicialización. Una vez realizada la configuración adecuada de los parámetros, se debe establecer cuales son los datos que se requieren mediante el comando 0x03. Los datos que podemos obtener en una petición son:

XRaw y YRaw. Son los valores crudos obtenidos de los transductores.

XCal y YCal. Son los valores de los transductores una vez que se han corregido.

Dirección. Es la dirección a la cual se apunta, será -1 si existe algun error.

Magnitud. Es la magnitud del vector cuyas componentes son XCal y YCal

Temperatura. Es la temperatura en °C obtenida del sensor de temperatura interno.

Distorsión. Indica si la magnitud varía en un 50% de la magnitud de calibración

Estado de calibración. Indica si el módulo requiere ser calibrado.

Así como los parámetros internos, los posibles datos de salida cuentan con un identificador, formato y otras características que se muestran en la tabla 3.11.

Tabla 3.11: Identificadores de datos

Dato	Identificador	Formato	Unidades	Rango
XRaw	0x01	Entero signado 32bits	cuentas	-32768 a 32767
YRaw	0x02	Entero signado 32bits	cuentas	-32768 a 32767
XCal	0x03	Flotante 32 bits		0 a 1.0
YCal	0x04	Flotante 32 bits		0 a 1.0
Dirección	0x05	Flotante 32 bits	grados	0.0 a 359.9°
Magnitud	0x06	Flotante 32 bits		0 a 1.0
Temperatura	0x07	Flotante 32 bits	°C	
Distorsión	0x08	Booleano	1/0	
Calibración	0x09	Booleano	1/0	

La sintaxis del comando 0x03 requiere que se indique el número de datos seguido por el identificador de cada uno de los datos, como se muestra en la figura 3.30.

Byte de sincronización	*Byte de comando*	*Bytes de datos*					*Byte de fin*
		Número de valores	*Identificador 1*	*Identificador 2*		*Identificador N*	
0xAA	0x03	N	0x**	0x**		0x**	0x00

Figura 3.30.: Estructura del comando 0x03

La única variable que nos interesa es la dirección así que la sintaxis de este comando queda de la manera mostrada en la figura 3.31.

```
spi_write(0xaa);//byte de sincronizacion
spi_write(0x03);//comando
spi_write(0x01);//numero de datos
spi_write(headingve);
spi_write(0x00);//caracter de fin
```

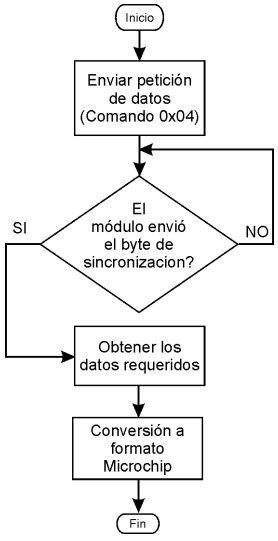
Figura 3.31.: Secuencia en PIC-C para configurar el módulo

Petición y lectura de datos Cuando la inicialización se ha realizado, se puede comenzar a obtener datos mediante el comando de petición de datos (código 0x04).Este comando no contiene parámetros, por lo que únicamente se envían el byte de sincronización, el byte de código y el byte de fin. Después de enviar la petición se debe esperar un tiempo que permita el muestreo y procesamiento de datos en el módulo. Mientras se encuentre realizando la medición se ignorarán los pulsos de reloj. Al terminar, el módulo envía la secuencia de respuesta a petición de datos: iniciando con el byte de sincronización (0xAA), seguido del código de comando (0x05), la cantidad de datos a leer "N" (los mismos solicitados) y finalmente los códigos y los valores de los datos requeridos. Ver figura 3.32.

Byte de sincronización	Byte de comando	Bytes de datos						Byte de fin
		Número de valores	Identificador 1	Valor 1	.....	Identificador N	Valor N	
0xAA	0x05	N	0x**	***		0x**	***	0x00

Figura 3.32.: Estructura del comando 0x05

En consecuencia, para conocer el momento en el que termina la medición se deben enviar pulsos de reloj constantemente y leer la línea de datos hasta recibir el byte de sincronización (0xAA). Por lo tanto el diagrama de flujo de una lectura de datos quedaría como se muestra en la figura 3.33 a. Dado que únicamente nos interesa la dirección, la cantidad de datos es 1 y el código correspondiente según la tabla 3.11 es 0x05, los cuales son los 2 primeros parámetros tanto en la secuencia de petición de datos como de respuesta a petición. Finalmente el código en PIC-C equivalente se muestra en figura 3.33 b.



a) Diagrama de flujo

```
//petición de datos datos
dato_v2xe=spi_read(0xaa);
dato_v2xe=spi_read(0x04);
dato_v2xe=spi_read(0x00);
while (dato_v2xe!=0xaa)//Espera
    dato_v2xe=spi_read(0x00);
//Obtención de datos
dato_v2xe= spi_read(0);
dato_v2xe= spi_read(0);
dato_v2xe= spi_read(0);
en_ieee[0]=spi_read(0x00);
en_ieee[1]=spi_read(0x00);
en_ieee[2]=spi_read(0x00);
en_ieee[3]=spi_read(0x00);
direccion=ieee_en_mic();
dato_v2xe= spi_read(0);
```

b) Código en PIC-C equivalente

Figura 3.33.: Secuencia de petición de datos y lectura

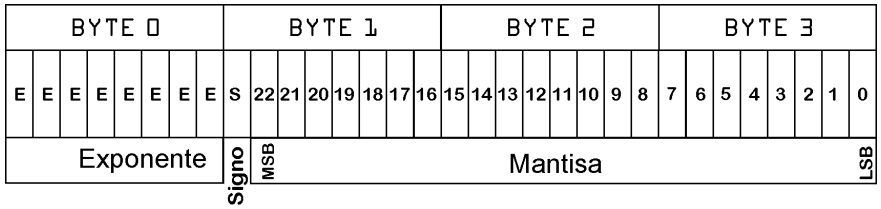
Los bytes son obtenidos uno a uno, por lo que se acomodaran en una matriz de bytes (llamada en_ieee) para una manipulación mas sencilla. El primer byte obtenido (byte0) se coloca en la posición cero de la matriz, mientras que el último se coloca en la posición 3.

Conversión de datos El V2Xe no envía datos en formato BCD o simplemente enteros, sino que envía datos de diferentes tipos (enteros signados y no signados, flotantes y booleanos). Debido a lo anterior, antes de manipular los datos debe tomarse en cuenta el formato que estructura a cada uno de estos tipos, la tabla 3.12 resume las características de los tipos que utiliza el módulo.

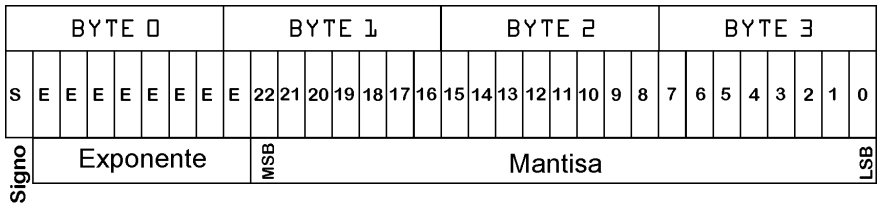
Tabla 3.12: Principales tipos de datos manejados por el módulo V2Xe

Tipo de dato	Formato	Tamaño (bits)
Entero no signado	Número no signado	8,16 y 32
Entero Signado	Complemento a 2	8,16 y 32
Flotante	Estandard IEEE 32 bits	32
Booleano	Número no signado	8

El formato que utiliza PIC-C para los tipos de enteros es el mismo que utiliza el módulo, sin embargo para el tipo flotante PIC-C utiliza el formato Microchip, ilustrado en la figura 3.34a, mientras que el módulo utiliza el formato IEEE (figura 3.34b). Como se puede observar, la diferencia es la ubicación del bit de signo: en el formato IEEE se encuentra en el bit 31, mientras en el formato de PIC-C se encuentra en el bit 23.



a) Formato de tipo flotante de Microchip



b) Formato de tipo flotante IEEE

Figura 3.34.: Formatos de tipo flotante

Por lo tanto se requiere una rutina que realice la conversión del formato IEEE al formato de Microchip efectuando corrimientos de bits. Se debe correr el byte0 y el byte1 en un bit a la izquierda, obteniendo en el acarreo el bit de signo; este bit debe introducirse en el byte1 por medio de un corrimiento a la derecha, ver figura 3.35a.

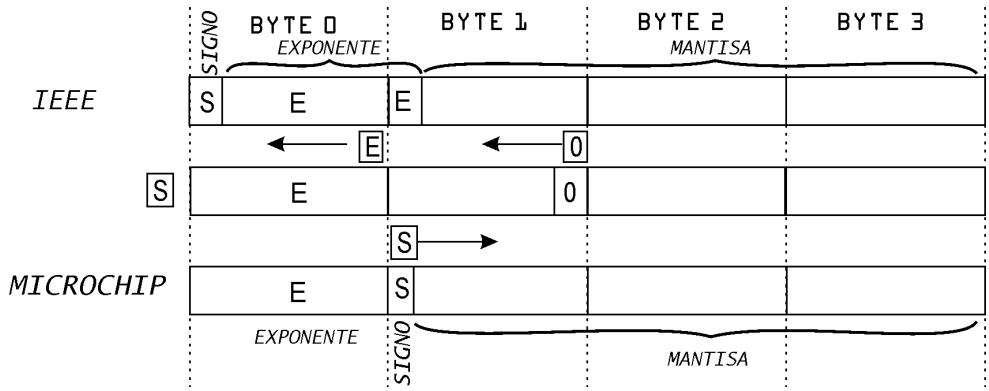
En PIC-C los corrimientos de bits se realizan mediante *shift_left (direccion,bytes,valor)* para un corrimiento de un bit a la izquierda y *shift_right (direccion,bytes,valor)* para un corrimiento de un bit a la derecha, donde:

dirección es un puntero a una dirección de memoria.

bytes son el número de bytes contiguos que se correrán

valor es el valor (0 o 1) que se colocará al hacer el corrimiento

Utilizando estas funciones y recordando que los bytes se encuentran en la matriz en_ieee(), la rutina de conversión quedaría como en la figura 3.35b.



a) Esquema de corrimientos necesarios para la conversión IEEE a Microchip

```
temp = shift_left(&en_ieee[1], 1, 0);
temp = shift_left(&en_ieee[0], 1, temp);
shift_right(&en_ieee[1], 1, temp);
flo=make32(en_ieee[3],en_ieee[2],en_ieee[1],en_ieee[0]);
pf=&flo;
return(*pf);
```

b) Código en PIC-C para la conversión

Figura 3.35.: Conversión de formato flotante IEEE a formato flotante Microchip

temp es una variable de un bit que sirve para almacenar el bit de acarreo. Así después del primer corrimiento a la izquierda se guardara el acarreo en temp y se colocara al realizar el segundo corrimiento, almacenando este nuevo acarreo en la misma variable.

Una vez hecho esto se tiene el bit de signo en *temp* por lo que sólo resta colocarlo mediante un corrimiento a la derecha en el byte1. Para convertir esta matriz de 4 bytes en un número flotante se utiliza la función *make32();* su objetivo es convertir cuatro bytes en una variable de 4 bytes. Simplemente se colocan los bytes como parámetros y la función devuelve la variable resultante. Finalmente se regresa el valor del flotante obtenido que ya es un flotante en formato Microchip y contiene la dirección que nos envía el V2Xe.

Calibración del módulo V2Xe Este proceso es independiente al proceso de medición pero es fundamental para obtener una buena exactitud. Se utiliza para corregir las distorsiones creadas por el ambiente cercano al módulo. Se debe realizar una calibración cuando se utiliza por primera vez, cuando se cambie el sitio de medición o cuando el módulo indique que es requerida una calibración (dirección=-1). Los tres comandos de calibración no requieren de parámetros por lo que siempre se envía la secuencia básica de 3 bytes: un byte de sincronización, un byte de comando y un byte de fin. La figura 3.36 ilustra específicamente cada comando.

Byte de sincronización	Byte de comando	Byte de fin
0xAA	0x0A	0x00

a) Secuencia de inicio de calibración.

Byte de sincronización	Byte de comando	Byte de fin
0xAA	0x0B	0x00

b) Secuencia de fin de calibración

Byte de sincronización	Byte de comando	Byte de fin
0xAA	0x09	0x00

c) Secuencia guardar configuración

Figura 3.36.: Comandos de calibración

Para lograr la calibración se debe seguir el siguiente algoritmo:

1. Configurar el parámetro Periodo en 4 mediante el comando 0x06.
2. Colocar el módulo en las condiciones que operará (instalarlo en el sistema). Una vez que el módulo se encuentre listo debe indicarse mediante la presión de un botón.
3. Enviar el comando de inicio de calibración (0x0A).
4. Girar lentamente el módulo hasta realizar al menos dos vueltas (720º). Cada vuelta debe durar al menos 30 segundos para asegurar una correcta calibración. Cuando la rotación es finalizada debe presionarse nuevamente el botón para indicarlo.
5. Enviar el comando de fin de calibración (0x0B)
6. Enviar el comando guardar configuración (0x09). Con lo que se guardan tanto los datos de configuración como los de calibración.

El código en PIC-C consiste el envío de un grupo de bytes para configurar el Periodo=4 (Paso 1), seguido por grupos de 3 instrucciones que envían los 3 bytes de la secuencia correspondiente (Pasos 3, 5 y 6 respectivamente). Adicionalmente las instrucciones para detectar la presión de un botón consisten de dos ciclos de tipo while: el primero detiene el proceso hasta que se presiona el botón y una vez hecho esto, el segundo detiene el proceso hasta que se suelta el botón; para evitar los llamados "rebotes" del botón, que causan falsos disparos, se cuenta con un retardo de 50 ms entre cada ciclo. Ver figura 3.37

```
void calibracion(){
int temp;
output_float(pin_calibracion);
temp=spi_read(0xaa);
temp=spi_read(0x06);//temp=spi_read(0x05);
temp=spi_read(0x04);
temp=spi_read(0x00);
while(input(pin_calibracion));
delay_ms(50);
while(!input(pin_calibracion));
output_low(indicador);
temp=spi_read(0xaa);//
temp=spi_read(0x0a);
temp=spi_read(0x00);
while(input(pin_calibracion));
delay_ms(50);
while(!input(pin_calibracion));
temp=spi_read(0xaa);//
temp=spi_read(0x0b);
temp=spi_read(0x00);
temp=spi_read(0xaa);/
temp=spi_read(0x09);
temp=spi_read(0x00);}
```

Figura 3.37.: Código en PIC-C de la rutina de calibración

3.2.4. Medición de presión

Dentro del MS5540 el transductor piezorresistivo convierte la presión atmosférica en una señal de tensión de tipo analógico, y el convertidor A/D toma esta tensión y la convierte en un valor digital de 16 bits. Además es posible obtener la temperatura interna, con fines de compensación, por medio de un transductor de tipo resistivo conectado al mismo convertidor A/D. Para una medición de presión se toma la salida de voltaje diferencial del sensor; para una medición de temperatura se utiliza un puente resistivo. En ambos casos, los transductores únicamente se alimentan en el momento de la medición para optimizar el uso de energía. Ver figura 3.38 .

Como la mayoría de los transductores, el piezorresistivo es muy dependiente de la temperatura por lo cual cada módulo es calibrado de fábrica utilizando dos parejas de temperatura y presión. Como resultado se obtienen 6 coeficientes con los cuales se debe realizar operaciones matemáticas que compensan los datos para variaciones de temperatura y variaciones en el proceso de fabricación. Los coeficientes se almacenan en la memoria interna del módulo y deben ser extraídos por el microcontrolador.

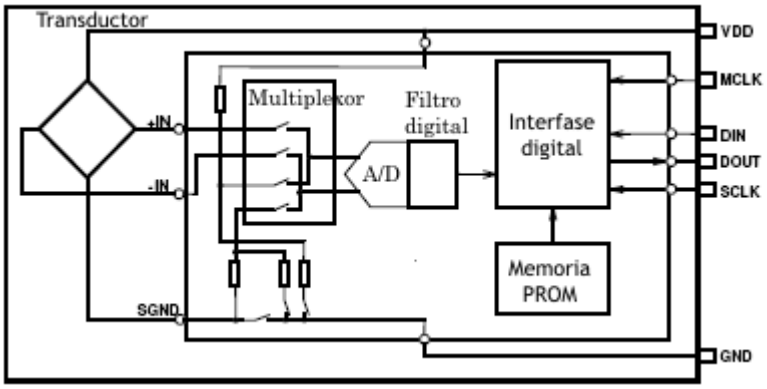


Figura 3.38.: Diagrama de bloques del ms5540b

Interfase de comunicación del MS5540B

El MS5540B se comunica por medio de un bus serie compatible con el estandar SPI antes descrito. Aunque no cuenta con una línea de SS, ésta se implementa fácilmente. Adicionalmente el módulo requiere de un oscilador para sincronizar los procesos internos. El circuito básico de conexión es bastante sencillo y se muestra en la figura 3.39.

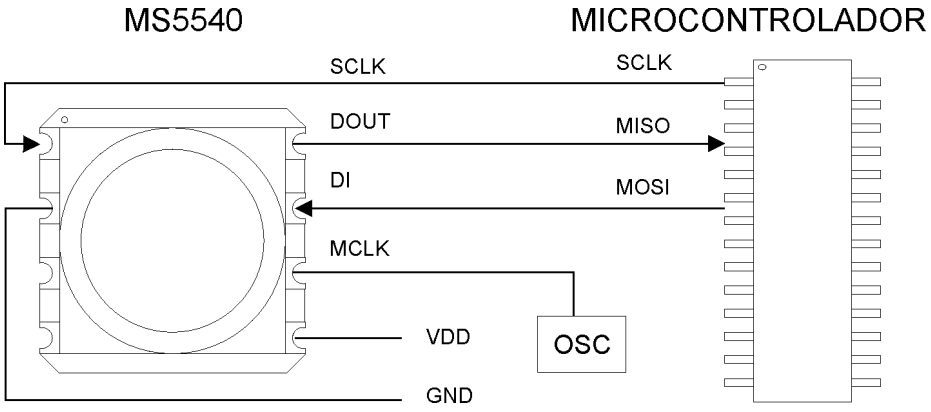


Figura 3.39.: Interfase de comunicación del MS5540b

Proceso de medición

Para el caso específico del MS5540b, cada bit es muestreado por el módulo en el flanco de subida de la señal de reloj; mientras que los bits que salen del módulo deben ser muestreados en el flanco de bajada. En el estado de descanso del reloj se debe mantener la línea un nivel bajo. La velocidad máxima del reloj es de 500 kHz. Ver figura 3.40

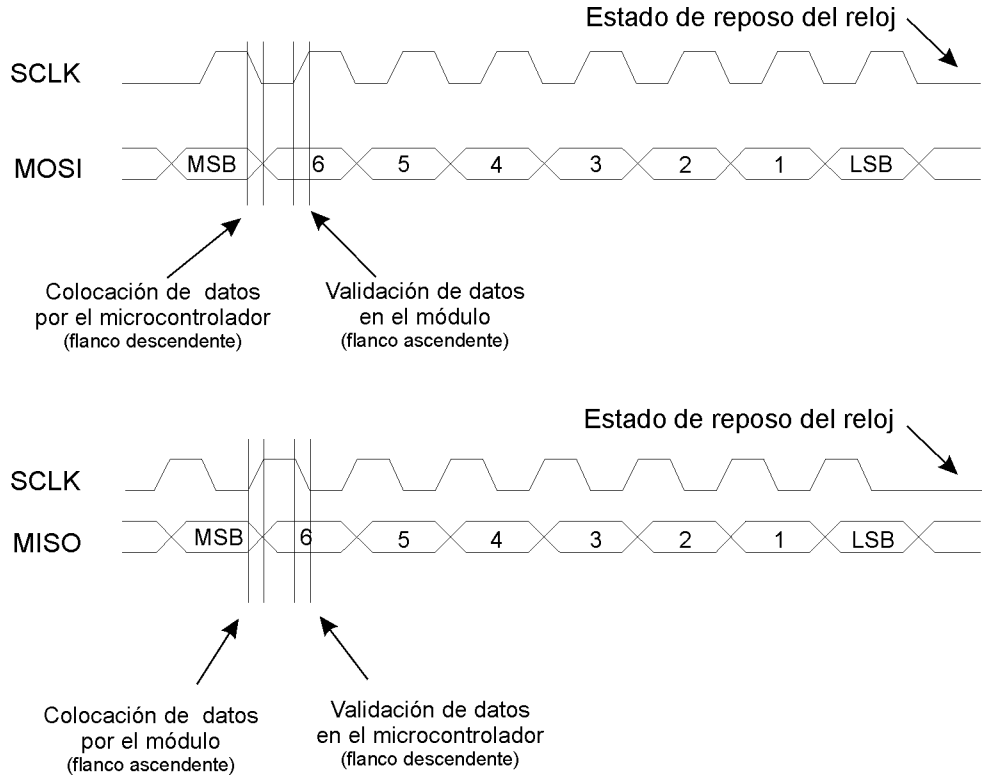


Figura 3.40.: Diagrama de tiempos del bus SPI del MS5540b

Cada secuencia de comunicación con el módulo consta de tres partes (ver figura 3.41):

- Una secuencia de inicio, que consta de tres bits en un nivel alto;
- Una secuencia de comando, que indica la acción que debe realizar el módulo y que consta de 4 bits en el caso de una petición medición y de 6 bits en el caso de lectura de datos de calibración;
- Y finalmente una secuencia de paro, que consta de tres bits en un nivel bajo.

Este módulo utiliza 6 diferentes secuencias: una para medir presión, una para medir temperatura y cuatro para leer las palabras de calibración; además utiliza una secuencia de reinicio que no sigue la estructura antes mencionada. En la figura 3.41 se muestra esta estructura por medio de diagramas de tiempos de las secuencias completas.

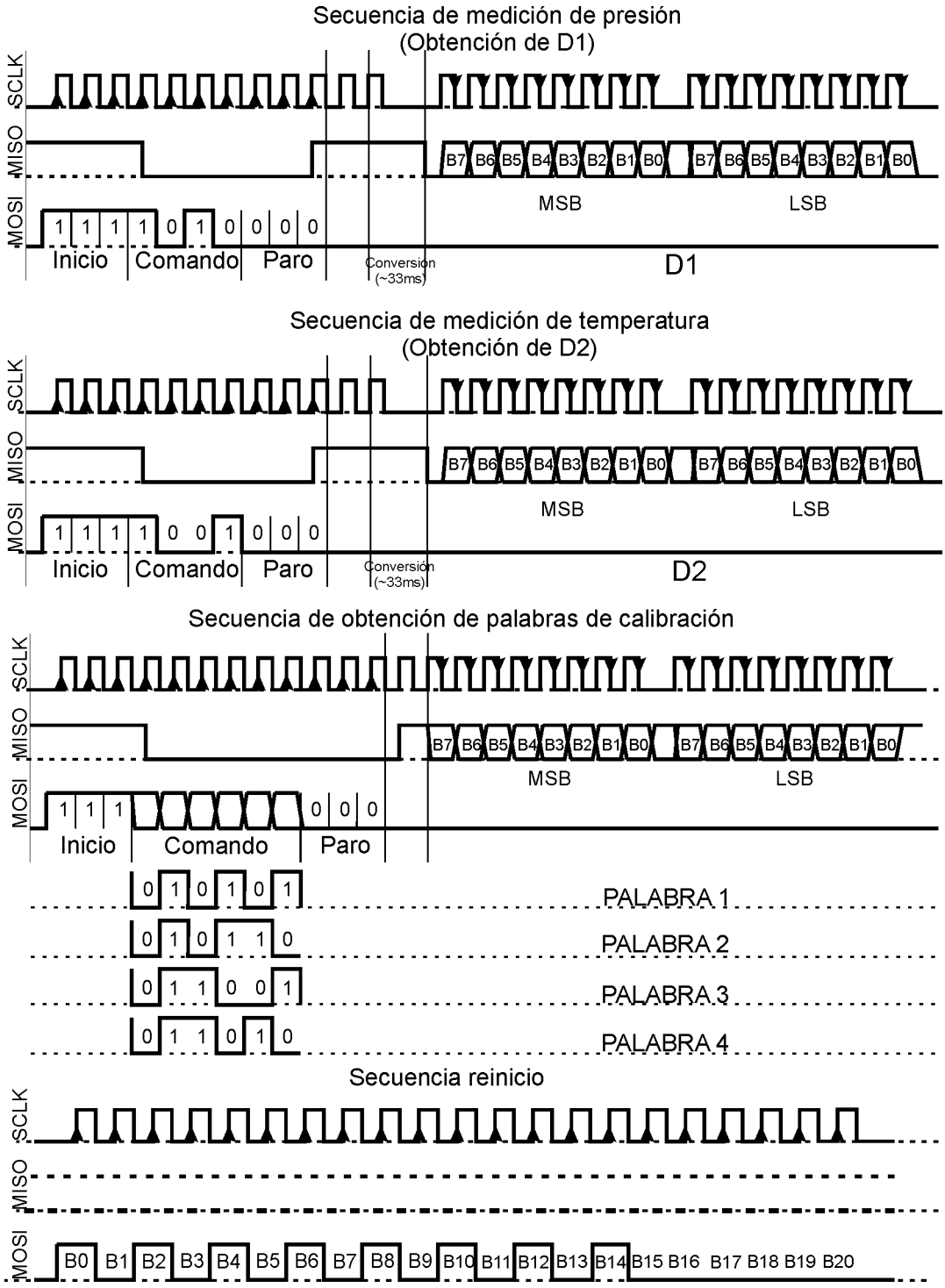


Figura 3.41.: Diagramas de tiempos de los comandos

Ahora bien, el protocolo SPI únicamente permite enviar información en grupos de 8 bits, por lo que hay que ajustar las secuencias y rellenar con ceros los espacios faltantes hasta completar las secuencias en grupos de 8 bits.

Como resultado se tiene los bytes indicados en seguida en la tabla 3.13 Hay que notar que, siguiendo los diagramas de tiempos de la figura 3.41 se debe adicionar 2 ceros al final para las mediciones de temperatura y humedad y un cero para las lecturas de las palabras.

Comando	Secuencia original con bits de inicio y paro	Rellenando con ceros (valor en hexadecimal)
Petición de lectura de presión	11,1101,0000	0000,1111,0100,0000 (0F40)
Petición de lectura de temperatura	11,1100,1000	0000,1111,0010,0000 (0F20)
Lectura de datos de calibración PALABRA1	1110,1010,1000	0001,1101,0101,0000 (1D50)
Lectura de datos de calibración PALABRA2	1110,1011,0000	0001,1101,0110,0000 (1D60)
Lectura de datos de calibración PALABRA3	1110,1100,1000	0001,1101,1001,0000 (1D90)
Lectura de datos de calibración PALABRA4	1110,1101,0000	0001,1101,1010,0000 (1DA0)
Secuencia de Reinicio	1,0101,0101,0101,0100,0000	0001,0101,0101,0101,0100,0000 (155540)

Tabla 3.13: Comandos del MS5540 compatibles con el protocolo SPI

Para realizar la medición de presión y temperatura se debe seguir el diagrama de flujo de la figura 3.42 que consta de las 3 secciones básicas: inicialización, petición y lectura de datos y conversión de datos. A continuación se explica cada una de estas secciones.

Inicialización

La inicialización consiste en la lectura de las palabras de calibración contenidas en la memoria PROM interna del módulo y su conversión a los coeficientes de compensación que posteriormente se utilizarán en el cálculo de la presión y temperatura. En primera instancia debe realizarse la petición de las palabras de calibración mediante el envío de los comandos 1D50, 1D60, 1D90 Y 1DA0 correspondientes a las palabras 1, 2, 3 y 4 (según la tabla 3.13). En seguida del envío de la petición de una palabra se deben generar 16 ciclos de reloj para leer los 2 bytes que conforman la palabra.

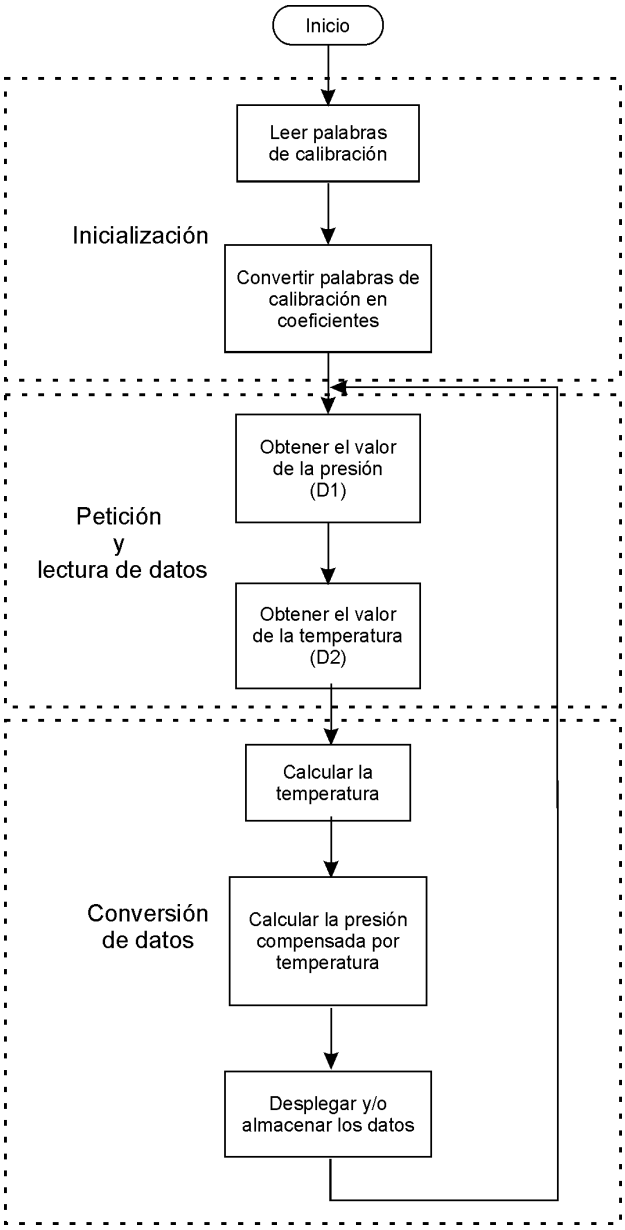


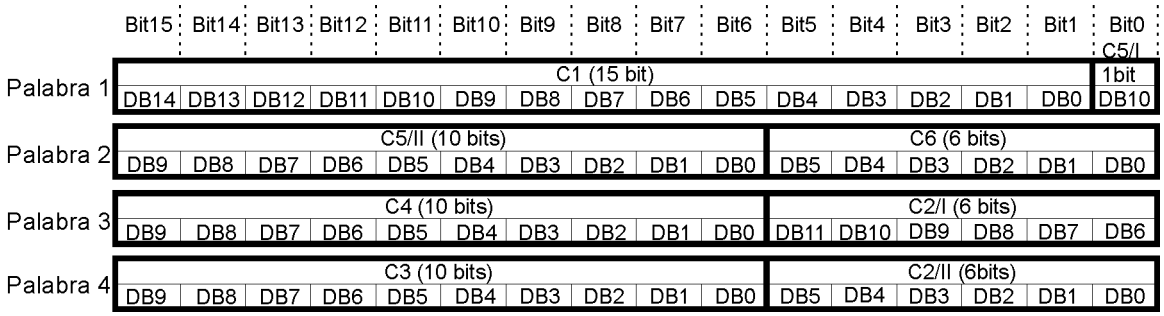
Figura 3.42.: Diagrama de Flujo para la medición de presión y temperatura

En la figura 3.43 se muestra el código en PIC-C que realiza la lectura de las palabras de calibración, primero se envía una secuencia de reinicio para asegurar una correcta sincronización en el bus. Las variables w1, w2, w3 y w4 corresponden a las palabras 1, 2, 3 y 4 respectivamente; primero se obtiene la parte alta (wX_h), luego la parte baja (wX_l) y finalmente se unen ambas partes para formar una variable de 16 bits.

Una vez obtenidas las palabras, tenemos que los 6 coeficientes se encuentran colocados dentro de 4 palabras de 16 bits de acuerdo a lo mostrado en la figura 3.44a, para extraerlos se debe aplicar diversas operaciones de bits como se muestra en la figura 3.44b.

```
//Secuencia de reinicio
spi_write(0x15);
spi_write(0x55);
spi_write(0x40);
spi_write(0);
//lectura palabra 1
spi_write(0x1D);
spi_write(0x50);
w1_h=spi_read(0);
w1_l=spi_read(0);
w1=make16(w1_h,w1_l);
//lectura palabra 2
spi_write(0x1D);
spi_write(0x60);
w2_h=spi_read(0);
w2_l=spi_read(0);
w2=make16(w2_h,w2_l);
//lectura palabra 3
spi_write(0x1D);
spi_write(0x90);
w3_h=spi_read(0);
w3_l=spi_read(0);
w3=make16(w3_h,w3_l);
//lectura palabra 4
spi_write(0x1D);
spi_write(0xA0);
w4_h=spi_read(0);
w4_l=spi_read(0);
w4=make16(w4_h,w4_l);
```

Figura 3.43.: Código en PIC-C correspondiente a la lectura de palabras de calibración



a) Colocación de los coeficientes de compensación dentro de las palabras de calibración.

```
c1=w1>>1;
c2=w3&0x003f;
c2=c2<<6;
c2=c2+(w4&0x003f);
c3=w4>>6;
c4=w3>>6;
c5=w2;
c5=c5>>6;
if (bit_test(w1,0))
  {bit_set(c5,10);}
c6=w2&0x003f;
```

b) Código en PIC-C para extraer los coeficientes de calibración

Figura 3.44.: Colocación de los coeficientes de calibración en la memoria del módulo y código en PIC-C para extraerlos.

Petición y lectura de datos Para realizar la medición de presión se envían tanto la petición de medición de presión como la de medición de temperatura. Como respuesta a cada petición se obtiene una lectura de 16 bits de donde obtenemos 2 valores de 16 bits (D1 y D2). Estos representan los valores de presión no compensada y temperatura. La rutina en PIC-C para obtener D1 y D2 es similar a la de petición de las palabras de calibración, como se puede ver en la figura 3.45. La diferencia entre hacer la petición de una palabra de calibración y hacer una petición de medición radica en que en este último caso se debe esperar a que la medición sea concluida.

En cuanto se envía la petición de medición el módulo coloca su línea de salida de datos (MISO) en un nivel alto, indicando que se encuentra realizando la medición. Una vez que la medición es terminada, la línea MISO vuelve a un nivel bajo en espera de la señal de reloj necesaria para sincronizar el envío de los 2 bytes. Por lo tanto, se debe realizar una revisión continua del pin MISO en espera del término de la medición. La forma más sencilla de hacerlo es utilizando un ciclo de tipo while como se muestra en la figura 3.45.

```
//Petición y lectura de D1
spi_write(0x0f);
spi_write(0x40);
//Espera a terminar medición
while(input(pin_MISO));
w1_h=spi_read(0);
w1_l=spi_read(0);
d1=make16(w1_h,w1_l);
```

```
//Petición y lectura de D2
spi_write(0x0f);
spi_write(0x20);
//Espera a terminar medición
while(input(pin_MISO));
w2_h=spi_read(0);
w2_l=spi_read(0);
d2=make16(w2_h,w2_l);
```

Figura 3.45.: Código en PIC-C para obtener D1 y D2 (medición de presión y temperatura)

Conversión de datos a valores de presión compensada Utilizando los valores y los coeficientes adquiridos durante la inicialización se calcula la presión compensada por medio de la siguiente ecuación:

$$P = X \cdot \frac{10}{2^5} + 250 \cdot 10 \tag{3.7}$$

Donde:

$$X = \frac{(SENS \cdot (D1 - 7168))}{2^{14}} - OFF \tag{3.8}$$

$$SENS = C1 + \frac{(C3 \cdot dT)}{2^{10}} + 24576 \tag{3.9}$$

$$OFF = C2 \cdot 4 + \frac{(C4 - 512) \cdot dT}{2^{12}} \tag{3.10}$$

$$dT = D2 - UT1 \tag{3.11}$$

$$UT1 = 8 \cdot C5 + 20224 \tag{3.12}$$

Gracias a la facilidad que PIC-C brinda para la realización de operaciones, el código queda prácticamente de la siguiente manera (ver en la figura 3.46)

```
ut1=(8*c5)+20224;
dt=d2-ut1;
offst=((c4-512)*dT)/4096;
offst=(c2*4)+offst;
sensitividad=(c3*dT);
sensitividad=sensitividad/1024;
sensitividad=c1+sensitividad+24576;
xsensitividad=((sensitividad*(d1-7168))/16384)-offst;
presion=(xsensitividad*.03125)+250;
```

Figura 3.46.: Código en PIC-C para calcular la presión compensada

3.2.5. Cálculo de la altura

Como se dijo anteriormente la altura es dependiente de la presión y de la temperatura y no es lineal por lo que se utiliza una aproximación de esta relación válida para la troposfera (hasta 11000m aprox.), que es la propuesta en el US Standard Atmosphere 1976:

$h = \frac{T_0}{T_{gradiente}} \left(1 - \left(\frac{P}{P_0}\right)^{T_{gradiente}\frac{R}{g}}\right)$

Donde:

P_0 = 1,013.25 mbar - presión a altitud cero (nivel del mar)

T_0= 288.15 K - Temperatura a altitud cero (nivel del mar)

$T_{gradiente}$ = 6.5/1000 [K/m] - Gradiente de temperatura

R =287.052 [J/(K*kg)] - Constante específica del gas

Sustituyendo los valores de cada variable, la fórmula queda:

$h = \frac{288.15}{6.5/1000}\left(1 - \left(\frac{P}{1{,}013.25}\right)^{6.5/1000\frac{287.052}{9.8}}\right)$

Aunque es cierto que la presión varía debido a las condiciones cambiantes del tiempo y condiciones especiales de clima, puede considerarse una buena aproximación si las mediciones se realizan en periodos cortos de tiempo en los que dichas condiciones no varían mucho.

Una vez realizadas las operaciones en la fórmula propuesta anteriormente se obtiene que la altura está dada por:

$h = 44330\left(1 - \left(\frac{P}{1{,}013.25}\right)^{0.19}\right)$

Lo cual se calcula en PIC-C con la siguiente instrucción:

```
altura=44330*(1-pow(presion/1013.25,0.19));
```

3.3. Etapa de adquisición

La etapa de adquisición consta de 3 partes:

1. Recolector o adquisitor de datos. Es la parte principal y se ocupa de obtener los datos medidos por los sensores y procesarlos para su almacenamiento y transmisión.
2. Base de tiempo. Es la encargada de llevar la cuenta de la hora y la fecha en el sistema.
3. Sistema de almacenamiento de datos. Este se encargará de almacenar de manera temporal los datos para mantenerlos disponibles después de terminada la medición.

3.3.1. Almacenamiento de datos

Ya que la memoria elegida como sistema de almacenamiento utiliza un bus tipo serie diferente a los manejados anteriormente (el I2C), es adecuado explicar brevemente su funcionamiento.

Protocolo serie I2C (Inter-Integrated Circuit)

El bus I2C es un bus de tipo serie diseñado por Philips, se utiliza principalmente para comunicar diversos circuitos integrados que se encuentran en un mismo sistema (normalmente gobernados por un microcontrolador).

Sus principales características son:

- Utiliza únicamente dos líneas: SDA (datos) y SCL(reloj).
- Las líneas SDA y SCL son de tipo colector abierto, por lo que requieren resistencias de pull-up.
- Pueden conectarse varios dispositivos en un único bus.
- Cada dispositivo conectado al bus es reconocido mediante una dirección de 7 o 10 bits.
- Es un bus bidireccional con diversas velocidades máximas de transferencia:
  - Modo Standard 100 kbits/s
  - Modo Fast 400 kbits/s
  - Modo Fast plus 1Mbits/s
  - Modo High Speed 3.4 Mbits/s
- Es un bus multi-maestro

Un circuito típico del bus I2C puede observarse en figura 3.47

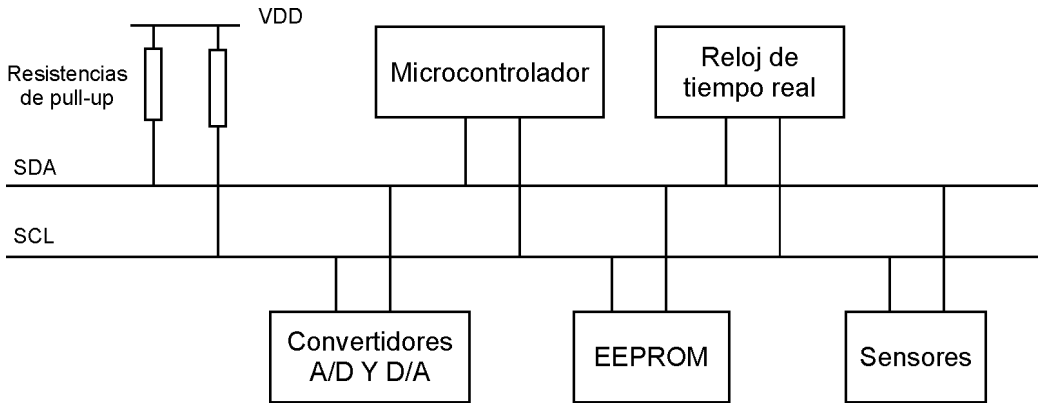


Figura 3.47.: Es posible conectar varios dispositivos a un mismo bus I2C

Aunque este bus tiene amplias capacidades, al ser este un texto introductorio se asumirá el bus en su forma mas sencilla, es decir como un bus sólo un maestro, direcciones de 7 bits y en modo Standard.

Características de las líneas SDA y SCL. Ambas líneas son de tipo bidireccional y se deben conectar a una fuente positiva de voltaje (VDD) mediante una resistencia de pull-up.

Debido a que se pueden conectar dispositivos con diferente tecnología al mismo bus, los niveles lógicos (alto y bajo o 1 y 0) se encuentran definidos por el nivel de la alimentación utilizada (VDD). Los niveles de referencia se encuentran en 30% VDD para el nivel bajo y 70%VDD para un nivel alto, es decir: VIL=0.3VDD y VIH=0.7VDD (ver figura 3.48). Los datos en SDA son validados mientras la señal de reloj (SCL) se encuentre en un nivel alto, el cambio de estado de la línea de datos se debe realizar cuando la señal de reloj se encuentre en un nivel bajo.

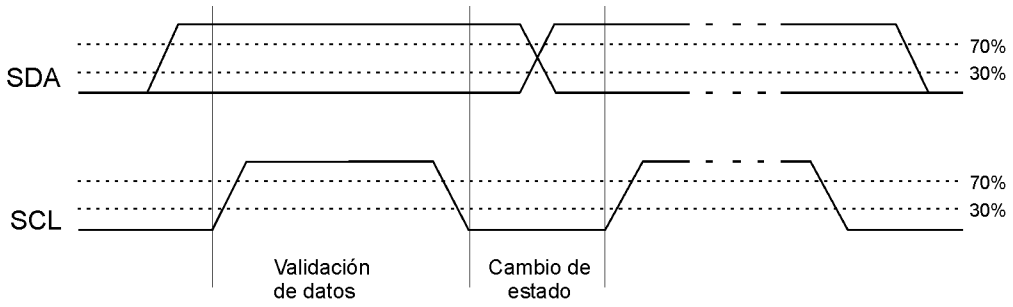


Figura 3.48.: Validación de datos y niveles de referencia para el bus I2C.

Estructura de envío de datos en el bus I2C. En primer lugar se debe tener en claro algunos conceptos básicos:

Maestro. Es el dispositivo que administra la transferencia de datos (inicia y termina la comunicación) y genera la señal de reloj

Esclavo. Es el dispositivo direccionado por el dispositivo maestro.

Transmisor. Dispositivo que envía datos al bus.

Receptor. Dispositivo que recibe datos del bus.

Secuencia de inicio (S). Se define como una transición de un nivel alto a uno bajo en SDA mientras SCL se mantiene en un nivel alto.

Secuencia de fin (P). Se define como una transición de un nivel bajo a uno alto en SDA mientras SCL se mantiene en un nivel alto

Tanto el maestro como el esclavo pueden ser transmisor o receptor, sin embargo el dispositivo maestro siempre es el que inicia la comunicación y genera la señal de reloj.

La transferencia de datos la comienza el maestro con una secuencia de inicio seguida por la dirección del esclavo con el que se entablará la comunicación, a esto se le llama direccionar el dispositivo o hacer una llamada al dispositivo. La dirección tiene una longitud de 7 bits seguidos de un bit llamado el bit de dirección de datos (R/W).

Se tienen dos posibles valores para el bit de dirección de datos:

0 (WRITE): Se indica que el maestro enviará datos al esclavo. El maestro será transmisor y el esclavo receptor.

1 (READ): Se indica un requerimiento de datos al esclavo. El maestro será receptor y el esclavo transmisor.

El esclavo debe responder con el bit de confirmación (ACK) y a continuación el transmisor, ya sea el maestro o el esclavo, colocará la información en grupos de 8 bits iniciando por el bit más significativo (MSB). Un envío adecuado de cada grupo de 8 bits debe ser confirmado por el receptor con el bit ACK (se utilizan 9 ciclos de reloj para cada byte enviado).

Un bit ACK consiste en colocar la línea de datos en un nivel bajo en el noveno ciclo de reloj. El número de bytes que se pueden enviar es ilimitado, marcando el término de la transmisión por medio de la secuencia de fin. Ver figura 3.49

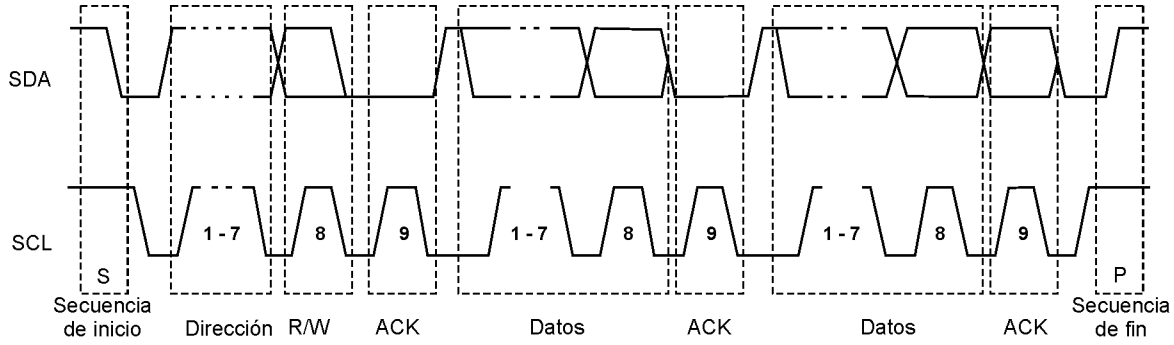


Figura 3.49.: Estructura de la transferencia de datos en el bus I2C

Manejo del protocolo I2C en PIC-C

PIC-C permite programar el uso del hardware para I2C, además de que se pueden implementar por software este protocolo en caso de que el PIC no contenga el hardware o no se quiera utilizar. En primer lugar se debe configurar el bus mediante la directiva de pre-procesamiento #use I2C que tiene la siguiente sintaxis:

#use i2c (opción1, opción2, opcion3,...)

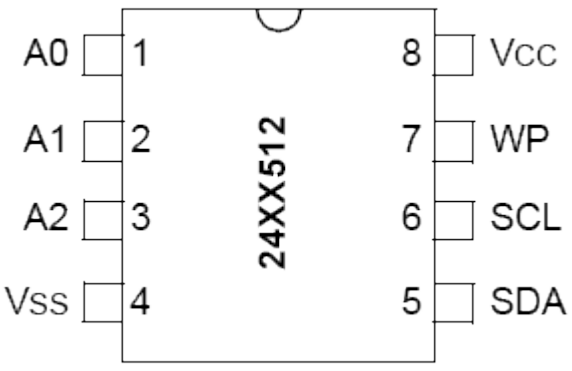
Donde las opciones pueden ser las indicadas a continuación en la tabla 3.14

Tabla 3.14: Opciones de configuración del bus I2C

Opciones	Acción
MULTI_MASTER	Activa el modo multi-maestro
SLAVE	Configura el microcontrolador como esclavo
SCL=pin	Especifica el pin que funcionará como reloj (SCL)
SDA=pin	Especifica el pin que se utilizará para el intercambio de datos (SDA)
ADDRESS=dirección	Especifica la dirección cuando se configura el microcontrolador como esclavo
FAST=velocidad	Utiliza las especificaciones para el modo FAST. Se debe indicar la velocidad del bus.
SLOW	Utiliza las especificaciones para el modo STANDARD
RESTART_WDT	Reinicia el watchdog timer (WDT) mientras se espera una lectura
FORCE_HW	Utiliza el hardware del microcontrolador para manejar el bus.
STREAM=id	Asocia un identificador (id) a la configuración para indicar que bus se utilizará (en caso de que trabaje con mas de un bus I2C)

La memoria 24LC512

Esta memoria tiene la ventaja de poder operar desde una tensión de 2.5 V, además tiene una capacidad de 64 kbytes y permite colocar hasta 8 memorias iguales en el bus (hasta 256 kbytes). Al utilizar el bus I2C su encapsulado es de únicamente 8 pines en el tipo PDIP (ver figura 3.50).



A0, A1 y A2. Pines de configuración de dirección de dispositivo

VSS Tierra

VCC 2.5 a 5.5 V

WP Protección de escritura

SCL Señal de reloj del bus I2C

SDA Señal de datos del bus I2C

Figura 3.50.: Patigrama de la memoria

La conexión es bastante sencilla: se cuenta con dos pines para la alimentación, dos para el bus I2C, uno para protección contra escritura y tres para configurar la dirección del dispositivo.

El pin de protección contra escritura (WP) inhibe las operaciones de escritura pero permite las de lectura, protegiendo los datos guardados contra un posible borrado accidental. Si desea activarse la protección contra escritura debe conectarse este pin a VCC.

Con objeto de permitir conectar varias memorias de este tipo en el mismo bus, la dirección de cada memoria es configurable por medio de los pines A0, A1 Y A2 (lo cual permite hasta 8 combinaciones). Entonces la dirección de 7 bits de cada memoria se encuentra conformada por dos partes: una fija (1010) y una variable (los bits A1, A2 y A3), ver figura 3.51.

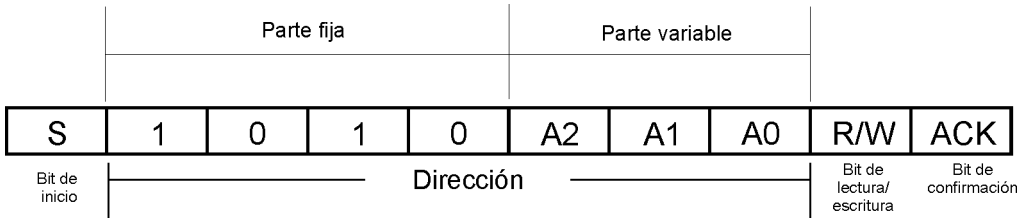


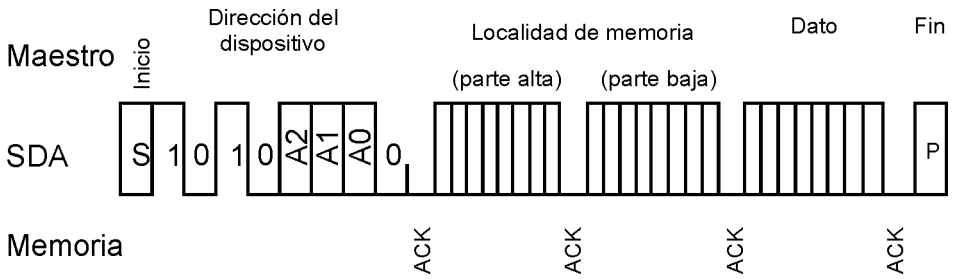
Figura 3.51.: Direccionamiento de la memoria 24LC512

Escritura

Para guardar un byte en la memoria se debe hacer una llamada al dispositivo con el bit R/W=0 para indicar que se realizará una escritura (el microcontrolador será el transmisor y la memoria el receptor). A continuación se enviará la localidad de memoria donde se guardara la información. Como el direccionamiento de las localidades de memoria es de 16 bits (0000 a FFFF) la dirección de la localidad de memoria se debe dividir en dos: parte alta y parte baja, enviando primero la parte alta y luego la parte baja.

En seguida se envía el byte que desea ser guardado y una secuencia de fin (P), en ese momento comienza el proceso interno de escritura durante el cual no se emitirá el bit ACK. Ver figura 3.52a.

Entonces para conocer el momento en el que termina la escritura se hacen llamados a la memoria hasta que esta confirma el llamado (bit ACK=0). En la figura 3.52b se encuentra la rutina en PIC-C que realiza la escritura en la memoria suponiendo A0=A1=A2=0, dicha función requiere la dirección donde se escribirá y el dato a escribir.



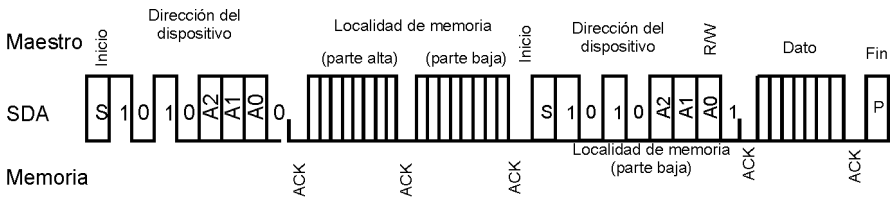
a) Diagrama de tiempos

```
void write_ext_eeprom(
long int,direccion,
BYTE dato){
short int ACK;
i2c_start();
i2c_write(0xa0);
i2c_write(direccion>>8);
i2c_write(direccion);
i2c_write(dato);
i2c_stop();
i2c_start();
status=i2c_write(0xa0);
while(ACK==1)
  { i2c_start();
   ACK=i2c_write(0xa0);}}
```

b) Código en PIC-C

Figura 3.52.: Escritura en la memoria 24LC512

Lectura En el caso de una lectura se debe indicar la localidad de memoria que se quiere leer, por lo que después de la secuencia de inicio (S) se hace un llamado a la memoria en modo escritura (R/W=0) seguido de los 2 bytes correspondientes a la dirección de dicha localidad de memoria (parte alta y parte baja). En seguida debe enviarse nuevamente una secuencia de inicio (S) y hacer otro llamado a la memoria pero ahora en modo de lectura (R/W=1). Una vez recibida esta llamada la memoria emite el bit ACK y transmite los 8 bits requeridos. El microcontrolador no confirmará la transmisión pero debe generar la secuencia de fin para terminar la comunicación. Ver figura 3.53a. En la figura3.53b se muestra la rutina en PIC-C que realiza la lectura en la memoria suponiendo A0=A1=A2=0, en este caso la función únicamente requiere la dirección que se leerá y se devuelve el valor de 8 bits encontrado.



a) Diagrama de tiempos

```
BYTE read_ext_eeprom(long int direccion){
BYTE dato;
  i2c_start();
  i2c_write(0xa0);
  i2c_write(direccion>>8);
  i2c_write(direccion);
  i2c_start();
  i2c_write(0xa1);
  data=i2c_read(0);
  i2c_stop();
  return(dato); }
```

b) Código en PIC-C

Figura 3.53.: Lectura en la memoria 24lc512

3.3.2. Sistema de reloj de tiempo real

El DS1307 es capaz de proveer la fecha y hora entregando información de segundos, minutos, horas, día, dia de la semana, mes y año. Además incluye una salida de onda cuadrada programable que puede ser utilizada para sincronizar datos y una pequeña memoria RAM de propósito general (56 bytes).

Su circuito de operación es sencillo: requiere un cristal de 32,768 kHz y resistencias de polarización en sus salidas de colector abierto, ver figura 3.54.

La información de la fecha y la hora se obtiene y configura por medio de la lectura y escritura de registros internos. Cuando el DS1307 es energizado, el estado de todos los registros es indeterminado, por lo que obligatoriamente debe ser configurado para que pueda funcionar. La tabla 3.15 muestra la configuración de los registros así como su estructura.

Los valores de fecha y hora se encuentran en BCD. El bit 7 del registro 0 (OSC) es el encargado de habilitar el oscilador.

El registro de control (0x07) sirve para configurar la salida de onda cuadrada:

bit 7 (OUT). Establece el estado de la salida cuando ésta se encuentra deshabilitada. Será un nivel bajo cuando OUT=0 y un nivel alto cuando OUT=1

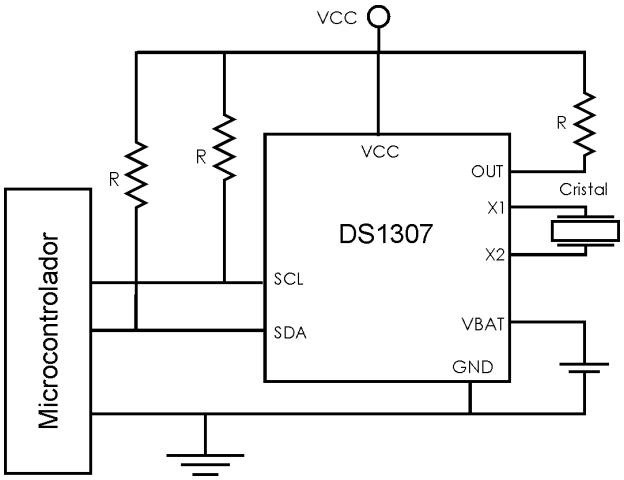


Figura 3.54.: Circuito de operación del DS1307

Tabla 3.15: Registros del reloj de tiempo real DS1307

Dirección	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Función	Rango
0x00	OSC	Decenas			Unidades				Segundos	00-59
0x01	0	Decenas			Unidades				Minutos	00-59
0x02	0	12/24	AM-PM/ Decenas	Decenas	Unidades				Horas	1-12/ 00-23
0x03	0	0	0	0	0	Unidades			Día (semana)	01-07
0x04	0	0	Decenas		Unidades				Día (mes)	01-31
0x05	0	0	0	Decena	Unidades				Mes	01-12
0x06	Decenas				Unidades				Año	00-99
0x07	OUT	0	0	SQWE	0	0	RS1	RS0	Control	-
0x08-0x3F									RAM 56x8	0x00-0xFF

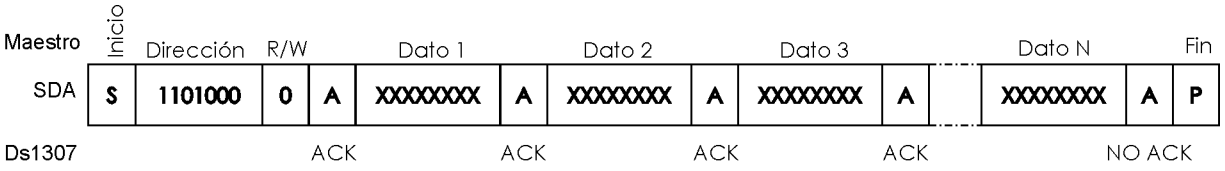
bit 4 (SQWE) Habilita o deshabilita la salida de onda cuadrada (1 habilita, 0 deshabilita)

bits 1,0 (RS1,RS0) Configuran la frecuencia de la onda cuadrada de acuerdo a los siguientes valores

RS1	RS0	Frecuencia de salida
0	0	1 Hz
0	1	4096 kHz
1	0	8192 kHz
1	1	32768 kHz

Configuración del DS1307

Para configurar el reloj de tiempo real DS1307 se debe hacer una llamada al dispositivo mediante la dirección 1101000 y en modo de escritura (quedando 11010000), a continuación el maestro debe transmitir la dirección del registro que se desea modificar seguido del valor que se desea escribir. Cada que el DS1307 lee un valor de registro incrementa en uno el valor del puntero de dirección, lo que permite una escritura secuencial de todos los registros si se inicializa a cero el puntero de dirección. Para terminar la comunicación el maestro debe enviar la secuencia de fin (P). En la figura 3.55a se encuentra el diagrama de tiempos de la configuración de los registros. En la figura 3.55b se encuentra el código en PIC-C que implementa la configuración del reloj de tiempo real con la señal cuadrada a 1hz, que se utilizará como base de tiempo para realizar las mediciones.



a) Diagrama de tiempos

```
i2c_start();
i2c_write(0b11010000);//byte de control del reloj, con escritura
i2c_write(0);//inicializando el puntero a la direccion cero
i2c_write(segundos);//Configura segundos e inicia el oscilador
i2c_write(minutos);//Configura minutos
i2c_write(horas);//Configura horas
i2c_write(num_dia); //Dia de la semana
i2c_write(dia); //Configura dia
i2c_write(mes); //Configura mes
i2c_write(anio); //Configura año
i2c_write(0b00010000); //Establece la señal cuadrada a 1hz
i2c_stop();
```

b) Código en PIC-C

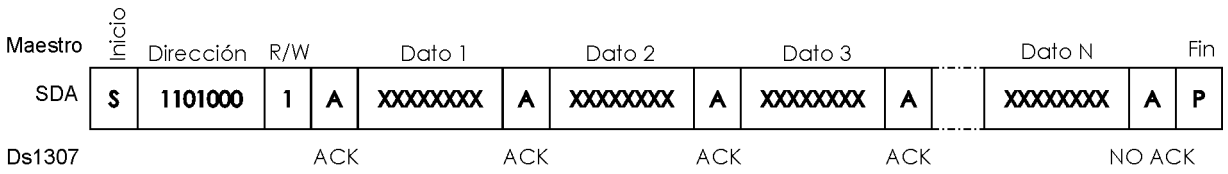
Figura 3.55.: Configuración de los registros de DS1307

Lectura de la hora y fecha

Cuando se hace una llamada al DS1307 en modo lectura, éste envía los datos en los registros a partir del valor tenga el puntero de dirección, ver figura 3.56a. Por lo tanto, para obtener la fecha y la hora el maestro debe colocar el puntero de dirección en el registro 0 (registro de los segundos) mediante una llamada al dispositivo en modo escritura seguida de la dirección (0).

Una vez hecho esto al llamar al DS1307 en modo lectura este enviará el bit ACK seguido por el valor del registro 0, esperará el bit ACK por parte del maestro e incrementará en uno el valor de su puntero de dirección, enviará el valor del registro 1, esperará el bit ACK e incrementará nuevamente el puntero de dirección y así sucesivamente hasta llegar al registro 6.

Una vez que el maestro lea el valor del registro 6 no transmitirá el bit ACK y a continuación generará la secuencia de fin (P) para terminar la lectura. En la figura 3.56b se muestra el código en PIC-C que realiza esta lectura guardando los valores de la fecha y hora en variables para su utilización posterior.



a) Diagrama de tiempos

```
i2c_start();
i2c_write(0b11010000); //llamada en modo escritura
i2c_write(0);          //inicializando el puntero a la direccion cero
i2c_stop();
i2c_start();
i2c_Write(0b11010001); //byte de control del reloj, con lectura
segundos=i2c_read();   //Lectura de todos los registros
minutos=i2c_read();
horas=i2c_read();
num_dia=i2c_read();
dia=i2c_read();
mes=i2c_read();
anio=i2c_read(0);
i2c_stop();
```

b) Código en PIC-C

Figura 3.56.: Lectura de la fecha y hora en el DS1307

3.3.3. Microcontrolador

El microcontrolador es el dispositivo central del sistema ya que controla a los demás dispositivos así como el flujo de la información. En este caso: configura la fecha y hora en el reloj de tiempo real, le indica a cada sensor en que momento debe realizar una medición, obtiene los resultados de la medición, procesa los datos, los guarda en memoria y los envía a una computadora, todo conforme el diagrama de la figura 3.57.

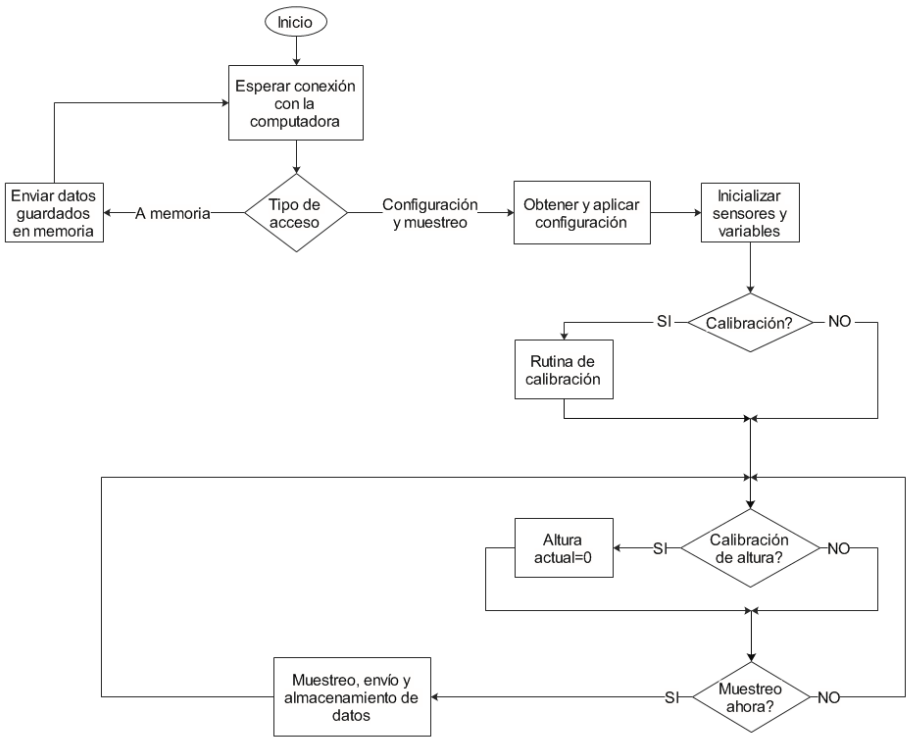


Figura 3.57.: Diagrama de flujo de las tareas realizadas por el microcontrolador

A continuación se describirá la programación de cada una de las etapas, que básicamente consta de una inicialización, la medición de datos y su conversión al tipo deseado (El código en PIC C completo puede consultarse en el anexo A)

Configuración del microcontrolador. Al programar el microcontrolador es necesario indicar las opciones de configuración: el tipo y la velocidad del oscilador que se utilizará, uso de la protección de código, uso del watch dog timer y otras características especiales disponibles.

No se requiere del uso de todas las características especiales así que únicamente se indica el tipo y la velocidad del oscilador (cristal de 8Mhz) mediante las siguientes directivas:

#fuses HS,NOBROWNOUT,NOWDT,NOPUT,NOPROTECT,NOLVP

#use delay(clock=8000000)

Distribución de pines La distribución de pines que se utilizó se encuentra en la figura 3.58. Aunque los canales analógicos no se utilizan se dejaron libres para poder utilizarlos en una posible mejora al diseño. Dado que el hardware para SPI y I2C es el mismo y el microcontrolador sólo cuenta con uno (pines RC3, RC4, RC5 Y RA5), se utilizó el hardware para el bus SPI y el bus I2C se colocó en otros pines implementando por medio de software. El muestreo de variables se hará cada segundo y se ejecutará por medio de una interrupción externa activada por la señal cuadrada proveniente del reloj de tiempo real, por lo cual ocupa el pin RB0.

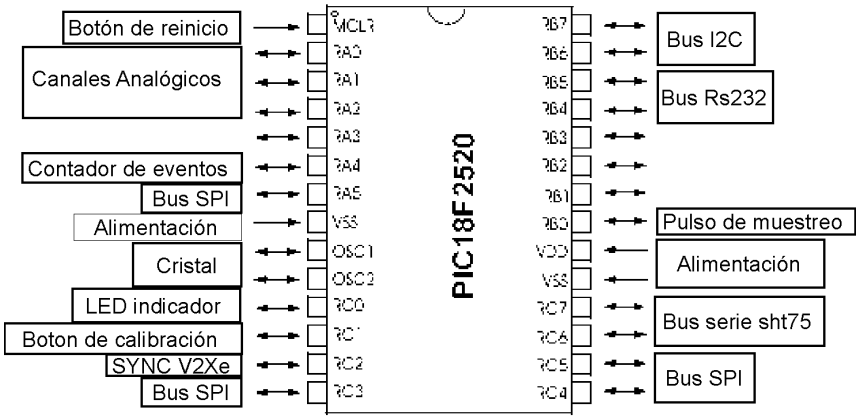


Figura 3.58.: Distribución de pines a utilizar

3.3.4. Configuración y muestreo.

Existen dos tipos de acceso a la sonda:

- Acceso de configuración y muestreo. Se establecen los parámetros que requiere la sonda para iniciar la medición de las variables.
- Acceso a memoria. Se obtienen los datos guardados en la memoria interna de la sonda después de realizar una medición.

Obtención y aplicación de la configuración

Los siguientes parámetros deben configurarse antes de realizar un muestreo:

- La fecha y hora.
- Tiempo de muestreo. Cada cuantos segundos se realizará una medición.
- Si debe realizarse una rutina de calibración de brújula previa al muestreo.
- Sobre-escritura de memoria. Si se comenzará a guardar datos al inicio de la memoria o en la dirección donde se terminó la última prueba

Con el fin de obtener estos parámetros de configuración, el microcontrolador enviará por medio de un puerto RS232 (hacia una computadora vía el módulo de transmisión) un byte y recibirá el parámetro correspondiente según la tabla 3.16

Tabla 3.16: Parámetros de configuración de la sonda

Byte (decimal)	ASCII	Parámetro solicitado
122	z	segundos (Hora)
90	Z	minutos (Hora)
121	y	hora (Hora)
89	Y	día (fecha)
120	x	mes (fecha)
88	X	año (fecha)
119	w	Tiempo de muestreo
87	W	ID de sonda
117	u	Calibración de brújula
85	U	No. sondas
116	t	Sobreescritura de memoria (Localidad de inicio de memoria)

La configuración de cada uno de estos parámetros se realiza por medio del software realizado por medio de Visual Basic. Este software toma la hora y fecha actuales de la computadora para sincronizarlas con la fecha y hora de la sonda pero también es posible configurar con una fecha y hora diferentes. Las demás opciones se configuran de acuerdo a las necesidades de la medición por medio de los campos destinados a esto. Una vez indicados todo los parámetros se presiona el botón configurar con lo que se envían todos los parámetro a la sonda vía RS232.

Los parámetros de fecha y hora son enviados al reloj de tiempo real para su configuración pero también son almacenados en la memoria externa junto con los demás parámetros para su uso posterior en caso de daño de la sonda.

Debido a esto se requiere un parámetro que indique la localidad en la que se puede comenzar a guardar los datos llamado “Localidad de inicio de memoria”. El mapa de memoria completo se muestra a continuación en la figura 3.59, los espacios de memoria vacíos se reservan para su posible uso en un diseño posterior (para almacenar máximos, mínimos o incluso otro diseño de memoria).

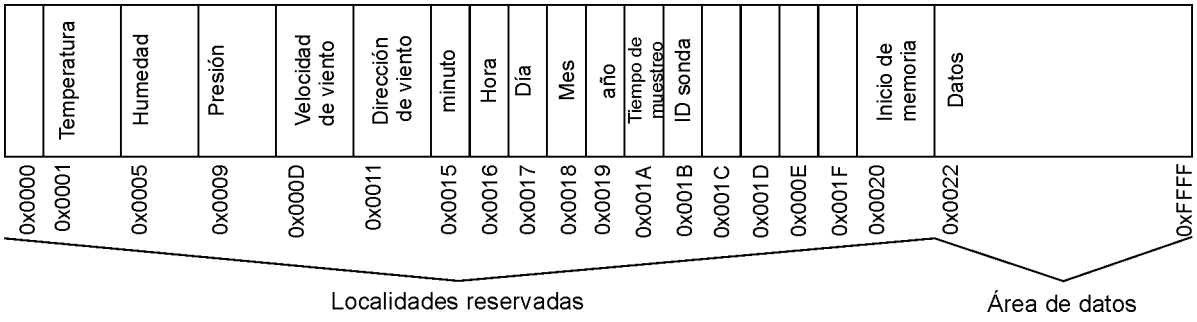


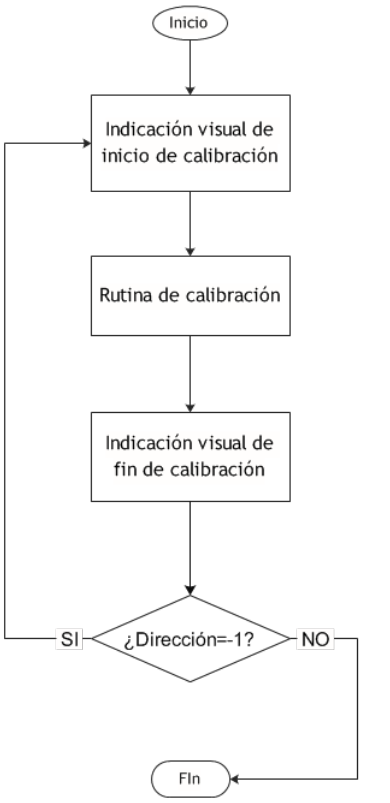
Figura 3.59.: Mapa de memoria

Inicialización de sensores y variables La inicialización se realiza mediante el llamado a rutinas específicas para cada sensor como se indicó en el apartado dedicado a cada sensor, en lo que toca a las variables todas las relativas a alguna medición deben inicializarse a cero para evitar algún error.

Calibración de la Brújula Aunque la calibración de la brújula también ha sido descrita anteriormente falta agregar una indicación de que se espera la presión del botón para iniciar la calibración y una indicación de que la calibración ha finalizado. Para indicar que se espera la presión del botón se debe hacer parpadear un LED varias veces; para indicar el fin el LED se mantendrá encendido por un segundo. Una vez que termina la rutina de calibración se debe verificar que se ha realizado adecuadamente (la dirección debe ser diferente de -1), de manera que todo queda dentro de un ciclo que revise esta condición (como puede verse en la figura 3.60).

Calibración de altura

Con la aproximación indicada anteriormente se puede obtener la altura a nivel del mar en la que se encuentra el sistema; sin embargo, es más cómodo conocer la altura con relación al nivel de suelo local. Con este objeto, una vez obtenida la altura (presión) local se fija este punto como “altura cero local” y se calcula la diferencia entre la altura del sistema (altura) y la altura cero local (altura_base).



a) Diagrama de flujo

```
do
{
output_low(indicador);
delay_ms(300);
output_high(indicador);
delay_ms(300);
output_low(indicador);
delay_ms(300);
output_high(indicador);
delay_ms(300);
output_low(indicador);
delay_ms(300);
output_high(indicador);
delay_ms(300);
output_low(habilitador);
delay_us(300);
setup_spi(spi_master|
   SPI_L_TO_H|SPI_CLK_DIV_64|
   SPI_XMIT_L_TO_H);
calibracion();
output_high(indicador);
delay_ms(1000);
lee_datos_v2xe();
}while (direccion==-1.0);
```

b) Código en PIC-C

Figura 3.60.: Calibración

Entonces la altura relativa al nivel de suelo local debe calcularse como:

$$altura\_relativa = altura - altura\_base$$

Donde:

altura_base es la altura del suelo con respecto al nivel del mar.

altura es la altura de la sonda con respecto a nivel del mar.

altura_relativa es la altura de la sonda con relación al suelo local.

Para que el microcontrolador conozca el momento en que se debe realizar esta calibración de altura, debe esperar que se presione un botón y en ese momento activar una bandera (guarda_altura_base) que indica fijar la altura actual como altura 0 o altura base en la siguiente medición de variables. Para esto tenemos el siguiente código:

```
while(input(pin_calibracion));
delay_ms(5);
while(! input(pin_calibracion));
guarda_altura_base=1;
```

Muestreo de datos

El muestreo de datos sigue el diagrama de flujo de la figura 3.61. Primero debe obtenerse la fecha y hora; en seguida se obtienen las variables físicas por medio de las rutinas explicadas anteriormente para cada sensor; una vez hecho esto las variables son filtradas para evitar cambios excesivamente drásticos (necesario para un buen cálculo de altura) y obtener así una respuesta suavizada; por último se calcula la altura relativa al nivel local.

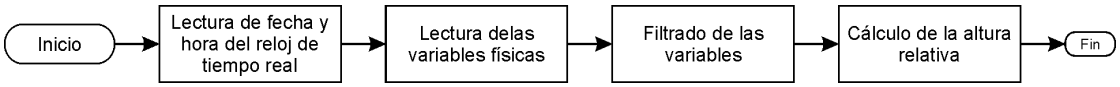


Figura 3.61.: Diagrama de flujo del muestreo de datos

El filtro utilizado es:

$y_n = y_{n-1} \cdot (1-k) + x \cdot k$

Donde:

y_n es el valor de la variable una vez filtrado

y_{n-1} es el valor filtrado anterior

x es el valor de la variable obtenido del sensor

k es un factor de amplificación ($k \varepsilon [0;1]$) el valor de k recomendado por el fabricante para una buena aproximación de la altura es k =1/8

La rutina en PIC-C que realiza este filtro es la siguiente:

```
float filtro(float x, float k, int localidad)
{
  float yn,yn1;
  yn1=read_float_eeprom(localidad);
  yn=(yn1*(1-k))+x*k;
  write_float_eeprom(localidad,yn);
  return(yn);
}
```

en este caso el valor filtrado se debe guardar en una localidad de memoria para leerlo en la siguiente iteración como el valor anterior (Yn-1)

Almacenamiento de datos

El almacenamiento de datos incluye, además de los valores de las variables, la fecha y hora siguiendo la estructura mostrada en la figura 3.62. El total de bytes necesarios es 32, lo cual nos permite almacenar hasta 2048 mediciones

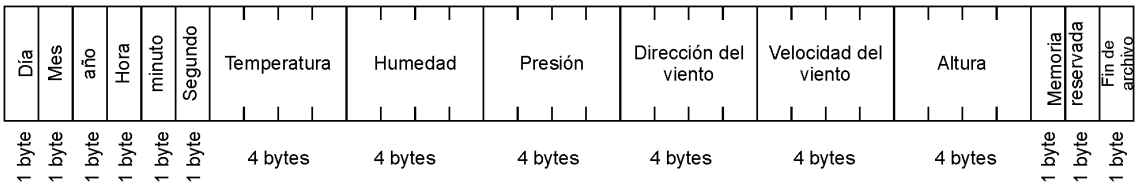


Figura 3.62.: Estructura de almacenamiento de datos

Transmisión de datos

La estructura de transmisión de datos es muy parecida a la estructura de almacenamiento de datos como lo podemos ver en la figura 3.63. Adicionalmente a los valores almacenados en la memoria son enviados algunos otros datos: un carácter de inicio para indicar el inicio del paquete de datos; el ID de la sonda; la dirección de la localidad memoria actual, para conocer el nivel de ocupación de memoria; un indicador de almacenamiento, que se utiliza para saber si es un dato de tiempo real (0) o un dato que debe almacenarse (1); y un carácter de fin para indicar el fin del paquete de datos. Cada dato debe ir separado por un espacio (0x20), el carácter de inicio es '! ' (0x21) y el carácter de fin es una línea nueva (0x10 + 0x13). Dado que el módulo de transmisión trabaja de manera transparente con el protocolo RS232 basta con colocar los datos en el bus y son enviados automáticamente.

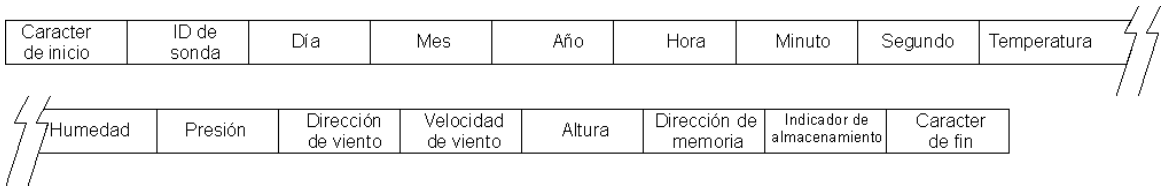


Figura 3.63.: Estructura de envio de datos

Acceso a memoria

En este tipo de acceso se direcciona a la localidad de inicio (0x22) y leer y enviar la secuencia de datos conforme la estructura mostrada en la figura 3.62 hasta encontrar el carácter de fin (0xFF)

Una vez que se envía cada grupo de variables se debe indicar enviando el carácter '+' (0x4B), de la misma manera para indicar el fin de la lectura de la memoria se debe enviar el carácter '*' (0x4A). El diagrama de flujo es el indicado en la figura 3.64.

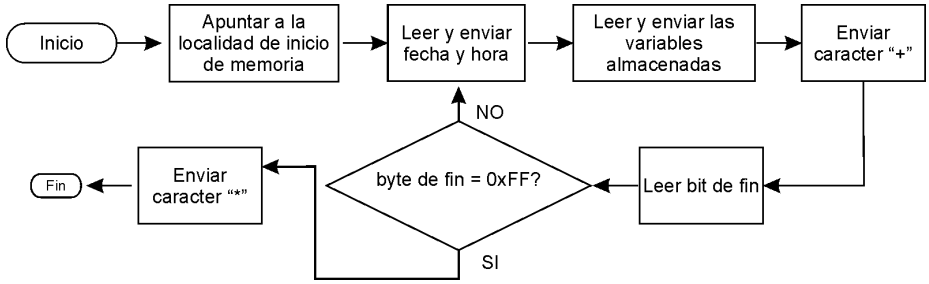


Figura 3.64.: Diagrama de flujo del acceso a memoria

3.4. Etapa de transmisión

La transmisión de datos vía radio frecuencia es de suma importancia, ya que hace posible el almacenamiento y visualización de los datos de manera instantánea. Esta disposición inmediata de los datos trae consigo dos grandes beneficios: una mejor medición y una mayor seguridad para el equipo. Por ejemplo: al visualizar los datos se sabe con bastante seguridad a que altura se encuentra el equipo en cada momento y puede detenerse el ascenso cada cierta altura para tener mejores mediciones en dichas alturas; por otro lado se conoce la magnitud del viento en altura en todo momento, con lo que se previene una posible ruptura del hilo provocada por vientos excesivos.

Otro dato importante que se puede conocer es si el sistema está subiendo, porque aunque el malacate se encuentre liberando hilo puede darse el caso de que el equipo únicamente tenga un desplazamiento horizontal y no uno vertical.

El circuito básico del módulo de transmisión Xbee PRO se compone de la alimentación y las líneas que conforman el bus RS232 (figura 3.65). Cada byte colocado en el bus es automáticamente enviado vía radiofrecuencia por el módulo y de la misma manera cada byte recibido por el módulo vía radiofrecuencia se coloca automáticamente en el bus RS232.

Los parámetros de configuración del bus RS232 pueden ser modificados para adecuarse a las preferencias de cada aplicación, sin embargo el módulo cuenta con una configuración predeterminada (ver figura 3.65).

Dado que este módulo es bidireccional, puede funcionar como transmisor y como receptor, por lo que se coloca uno en la sonda como transmisor y otro se tiene en tierra conectado a una computadora como receptor mediante un puerto serie RS232 (ver figura 3.66).

- Velocidad: 9600 baud.
- Longitud de palabra: 8 bits
- Paridad: ninguna
- Bit de paro: 1.

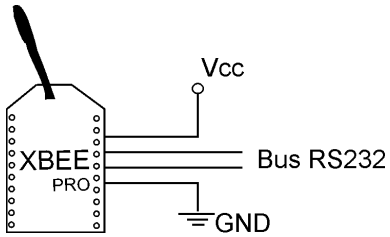


Figura 3.65.: Circuito básico del Xbee PRO

Adicionalmente, el módulo puede ser configurado con una dirección de recepción y una dirección de envío (canal) de manera que únicamente el módulo que transmita en ese canal puede enviarle datos. Esto es muy útil si se pretende utilizar más de un par transmisor-receptor en un área cercana en la que pudiera haber interferencia.

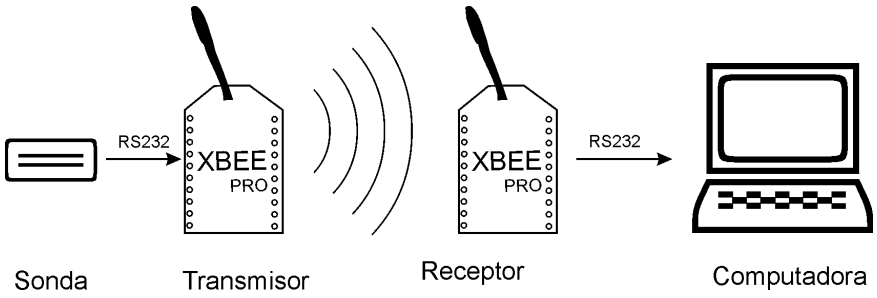
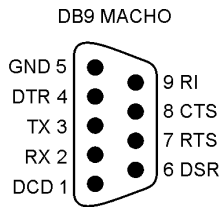


Figura 3.66.: Diagrama de transmisión vía radiofrecuencia

Estándar RS232 Con fines de comunicación entre dos dispositivos existen dos tipos de bus: paralelo y serie. Actualmente están en desuso los buses de tipo paralelo debido a que son costosos porque requieren un mayor número de líneas. Uno de los buses de tipo serie más ampliamente usado es el bus RS232, aunque también tiende al desuso en las computadoras aun existen diversos dispositivos que se comunican por medio de este bus y es fácil encontrar adaptadores (serie-usb) para dotar de este tipo de comunicación a las computadoras que no cuentan con él. El RS232 es un bus asíncrono bidireccional full duplex (los datos pueden viajar en ambos sentidos simultáneamente) que normalmente utiliza un conector de 9 pines (DB9), aunque también existe un conector de 25 pines (DB25). En la figura 3.67 se muestra la asignación de pines en el conector DB9.

Señal		DB9
Detección de portadora	DCD	1
Recepción de datos	Rx	2
Transmisión de datos	Tx	3
Terminal de datos listo	DTR	4
Tierra	GND	5
Equipo de datos listo	DSR	6
Solicitud de envio	RTS	7
Libre para envio	CTS	8
Indicador de timbrado	RI	9

a) Listado de pines



b) Distribución de pines en el conector DB9

Figura 3.67.: Conector DB9 para RS232

Manejo del bus RS232 en PIC-C

Este bus es ampliamente usado, por lo que muchos microcontroladores incluyen un hardware específico para manejarlo o en su defecto se proporcionan rutinas de software para implementarlo. PIC-C incluye diversas funciones para manejar ambas opciones.

En primer lugar se debe especificar la configuración del bus a implementar mediante el uso de la directiva de pre-procesamiento *#use rs232* con la siguiente sintaxis:

#use rs232 (OPCIONES)

Donde OPCIONES van separadas por comas y las principales se muestran en la tabla 3.17

Tabla 3.17: Opciones de configuración de la directiva #use RS232

Opción	Función
STREAM=ID	Establece un identificador para la configuración
BAUD=X	Especifica la velocidad del bus
XMIT=PIN	Establece el pin de recepción de datos Rx
RCV=PIN	Establece el pin de envio de datos Tx
FORCE_SW	Obliga a generar rutinas de software para manejar el bus incluso si se especifican los pines del hardware
RESTART_WDT	Reiniciará el WDT mientras se espera por un caracter
INVERT	Invierte la polaridad de las señales (sin esta opción un nivel positivo es un 1 lógico)

Opción (continuación)	Función (continuación)
PARITY=X	Establece la paridad que se utilizará, X puede ser: N (ninguna), E (par) u O (impar)
BITS=X	Establece el número de bits de datos, X puede tomar valores entre 5 y 9
STOP=X	Establece el número de bits de paro
DISABLE_INTS	Deshabilita las interrupciones mientras se envia o recibe un byte
TIMEOUT=X	Indica el tiempo en milisegundos que se esperará cuando se hace una lectura, sino se recibe ningún caracter se regresará un valor de 0 y se indicará un mensaje de error
UART1	Establece los pines de recepción y transmisión a los correspondientes al hardware (UART1)
UART2	Establece los pines de recepción y transmisión a los correspondientes al hardware (UART2)

3.5. Etapa de visualización y almacenamiento de la información

El microcontrolador se encarga de controlar el flujo y manipular los datos sin embargo se requiere de la interacción con una computadora mediante la cual se configure la sonda de acuerdo a las necesidades de medición, además de que permita la captura y el almacenamiento de datos , y el despliegue de la información en pantalla en tiempo real. Para lograr esta interacción se desarrolló un programa con una interfase intuitiva en visual basic. La recepción se realiza por medio de un módulo Xbee PRO conectado a un puerto serie de la computadora, el diagrama de flujo utilizado es el mostrado en la figura 3.68

La visualización de los datos incluyen el despliegue en un tamaño de texto grande, así como una forma gráfica de la dirección del viento lo que facilita su interpretación en el momento de realizar las mediciones. Además se agregó un parámetro extra (dH) muy útil al realizar las mediciones. dH calcula la diferencia de la altura actual y la altura anterior, indicandonos si la sonda se esta elevando o no (ver figura 3.69). El almacenamiento de los datos se realiza automáticamente en cuanto se reciben los datos.

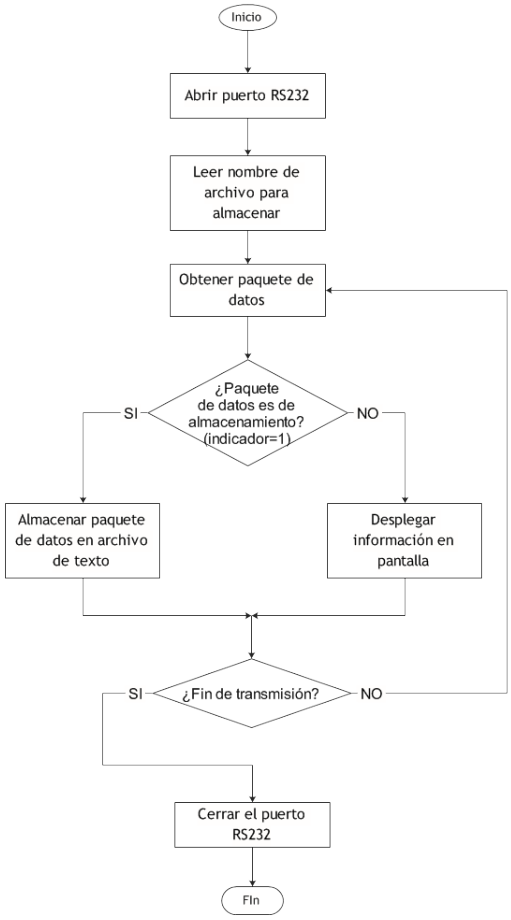


Figura 3.68.: Diagrama de flujo de la recepción de datos

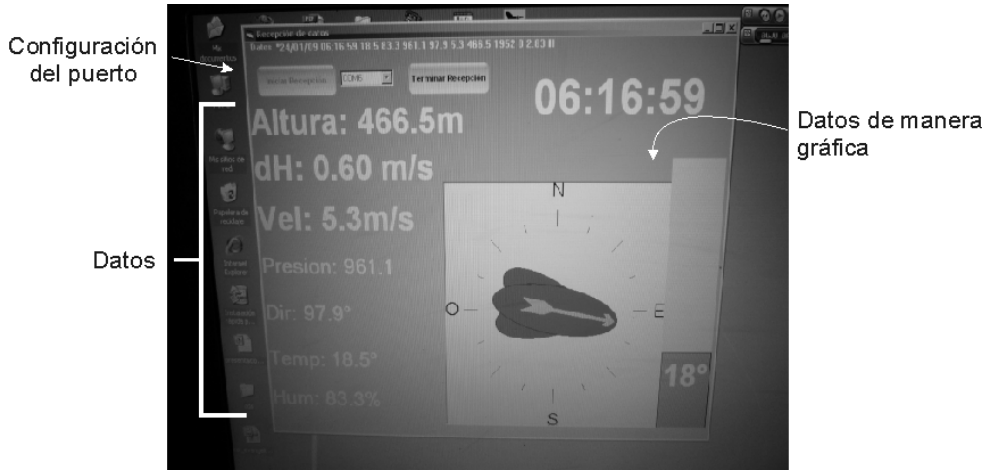


Figura 3.69.: Estructura de la pantalla de despliegue

Este mismo software permite la configuración de la sonda de manera sencilla. Los parámetros configurables son:

a) Hora.

b) Fecha.

c) Intervalo de muestreo.

d) Identificador de la sonda.

e) Realización de calibración de brújula antes de iniciar la medición.

f) Sobreescritura de datos previamente obtenidos por la sonda

Finalmente el programa tiene también la función de recuperar los datos después de una medición en caso de que se pierda la comunicación. Para esto, sólo se requiere establecer la comunicación con la sonda e indicar el archivo en el que se almacenará la información. El software se realizó en el lenguaje de programación Visual Basic debido a la sencillez de programación que proporciona.

3.6. Etapa de ascenso y descenso del sistema

Para poder elevar y descender el sistema son necesarios tres componentes:

- Un globo que eleve el sistema.
- Un hilo que sujete al globo y al sistema a tierra.
- Un malacate eléctrico que nos permita desembobinar o rebobinar el hilo de sujeción.

Globo

Si bien en principio es posible utilizar cualquier tipo de globo siempre y cuando permita el ascenso, la aplicación nos exije las siguientes características para obtener un mejor desempeño:

- Forma tipo zeppelín. Su forma aerodinámica evita el arrastre causado por el viento, facilitando de esta manera el ascenso y proporcionando una gran estabilidad. Esta permite una mejor medición en especial en el caso de los sensores de viento.
- Diseño estable. Otro aspecto importante en cuanto a la forma es que sea estable a velocidades de viento considerables (15 m/s). Si bien la velocidad más alta a la cual se recomienda elevar el globo es con 10 o 12 m/s, es posible que a veces el globo se encuentre sujeto a velocidades mayores debido a un cambio repentino de las condiciones. Si el globo no es lo suficientemente resistente y estable puede causar la pérdida del equipo o incluso algún accidente. En este sentido es preferible utilizar modelos con cuatro aletas a modelos con tres, que son menos estables.
- Capacidad de ascenso. El globo debe tener la capacidad de levantar la carga que se le coloque. Si bien esta capacidad es afectada por diversos factores (presión atmosferica, temperatura, pureza del gas de relleno, etc.), el principal factor es la densidad del gas de relleno en contra de la densidad del aire. Normalmente suelen utilizarse o helio o hidrógeno como gases de relleno, sin embargo aunque el hidrógeno es más ligero (y por tanto tiene un mayor empuje) también es flamable y muy peligroso, por lo cual fue desechado como opción viable. La densidad del aire es de 1.2 kg/m3, mientras que la del hélio es de 0.18 kg/m3. La capacidad de ascenso depende proporcionalmente del volumen de aire desplazado, es decir, si desplazamos 1 m3 de aire tendremos una capacidad ascendente de 1.2 kg. Dado que el gas con el que desplacemos el aire también tiene una masa (en este caso 0.18 kg de helio), debemos restar dicha masa. De lo anterior, es posible calcular la capacidad de ascenso por medio de la resta de densidades del aire y el helio:

- Capacidad de ascenso= 1.2 kg/m3- 0.18 kg/m3
- Capacidad de ascenso = 1.02 kg/m3

El globo elegido cuenta con las siguientes características:

Volúmen máximo: 6 m3

Material: nylon cubierto con urethano

Malacate El malacate utilizado es de aluminio para soportar la corrosión, utiliza un motor de 1/2 hp, cuenta con una guía para enredar el hilo automáticamente en el carrete e incluye un control de velocidad que permite variala durante el ascenso y descenso del sistema, una foto se encuentra en la figura 3.70. Este malacate fue fabricado por la compañía AIR.

Figura 3.70.: Malacate eléctrico

Hilo El hilo utilizado es de la compañía Vaisala, siendo el mas adecuado debido a que esta hecho específicamente para esta aplicación, sus características son:

Longitud: hasta 2 km

Peso: 0.96 g/m

Diametro: 1.25 mm

3.7. Alimentación

Al ser este un sistema que debe funcionar de manera autónoma en altura, inevitablemente debe utilizarse una alimentación basada en baterías. Esta es una de las razones por las que se eligieron sensores capaces de trabajar con una alimentación baja. Los sensores elegidos requieren tensiones alrededor de 3 V, por lo que este es el valor de tensión que nos debe proporcionar la batería a utilizar. Existen diversos tipos de baterías: alcalinas, de litio, de zinc-carbón, etc ver tabla 3.18.

Tabla 3.18: Características de diversos tipos de baterías

Tipo	Tensión	Densidad de energía (MJ/kg)	
Zinc-carbón	1.5	0.13	Baratas
Alcalinas	1.5	0.4-0.59	Densidad de energía moderada
Litio (LiFeS2)	1.5		Caras, utilizadas para una duración extra
Litio (LiMnO2)	3	0.83-1.01	Caras, utilizadas para larga duración. Comúnmente llamadas de litio
Oxido de plata	1.55	0.47	Se utilizan en las baterias de forma de botón

Podemos observar que las baterías de Litio tienen la mejor densidad de energía, por lo cual son las más ligeras, además de una larga duración las cuales son características deseables para la aplicación. Dentro de las baterías de litio se eligió la batería de uso fotográfico CR123 debido a que nos brinda 3V en una sola pieza, es ligera y tiene una larga duración, además de que es de fácil adquisición en el país. Sus características son:

Tensión nominal: 3V

Capacidad Típica: 1500mAh

Peso: 17 g (una AA pesa 23g aprox.)

Tipo: LitioTamaño 17 x 34.5 mm

3.8. Diseño final del prototipo

Con objeto de facilitar el desarrollo del sistema se implementó el funcionamiento de cada sensor de manera independiente. Una vez que se logró el funcionamiento individual, el siguiente paso fue unir todos los sensores en un único sistema. En un sistema de tipo analógico este paso no consiste en mas que conectar la salida de cada etapa de acondicionamiento de cada sensor al canal analógico adecuado en adquisitor de datos. Sin embargo, en este caso al utilizar sensores con salidas de tipo digitales, una vez que se conectaron las señales al microcontrolador (adquisitor de datos) se realizó una etapa de depuración que permitió la correcta lectura de los dispositivos que se encontraban algunos incluso en el mismo bus que otros. En contra parte, debido al uso de un microcontrolador, el circuito obtenido resultó ser relativamente sencillo y con pocos elementos. Tanto el circuito impreso como el esquemático obtenidos se muestran en la parte correspondiente a los anexos. Por otro lado, las dimensiones físicas del prototipo se muestran de manera detallada en el capítulo 5.