Universidad de Cuenca

Facultad de Ingeniería

Escuela de Electrónica y Telecomunicaciones

Prototipo de un dispositivo de adquisición, almacenamiento y transmisión de datos meteorológicos usando los transductores de la estación DAVIS 6162 Wireless Vantage Pro2 Plus

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Autores :

Marco Paúl Gutama Morocho C.I.: 010548957-9

Joel José Vázquez Patiño C.I.: 010544075-4

Director :

Ing. Remigio Clemente Guevara Baculima C.I.: 010278201-8

Co-Director :

Ing. Darwin Fabián Astudillo Salinas, PhD C.I.: 010390703-6

Cuenca - Ecuador

2017

Resumen

Estudiar el clima y el tiempo meteorológico es importante desde el punto de vista científico y de planificación, por lo que es necesario contar con datos meteorológicos fiables. Estos datos deben ser recolectados en diferentes puntos de interés, frecuentemente de difícil acceso y dispersos en una amplia área geográfica. Se usan estaciones meteorológicas que en muchos casos no cuentan con un sistema de transmisión de largo alcance, por lo que el personal va a su ubicación para descargar los datos. Esto implica gastar recursos en una actividad que puede ser automática. Estos hechos demuestran la necesidad de equipos que reduzcan costos de recolección de datos y ayuden a evitar la pérdida de los mismos.

En este trabajo se diseñó e implementó un dispositivo prototipo para adquirir y transmitir automáticamente las observaciones realizadas por las estaciones. El prototipo utiliza los transductores de una estación Davis 6162 Wireless Vantage Pro2 Plus. Para gestionar la adquisición, el almacenamiento y la transmisión de los datos se utilizó un microcontrolador con un sistema operativo de tiempo real. Los datos son registrados en una memoria SD y transmitidos mediante un módulo XBee. La referencia de fecha y hora se obtiene de un reloj de tiempo real sincronizado con un GPS. Además, el prototipo permite verificar su funcionamiento y configurar periodos de muestreo y transmisión. Como resultado, de acuerdo a pruebas realizadas, el prototipo es capaz de adquirir, almacenar y transmitir los datos correctamente.

Palabras clave : Estación meteorológica, Sistema operativo de tiempo real, XBee, OSA.

Abstract

Studying climate and weather is important for science and planification, therefore, reliable meteorological data is required. Data should be collected in places of interest often difficult to access and dispersed in a wide geographical area. Meteorological stations are used to collect data, but they often lack long range transmition systems, therefore, staff goes to the stations to download the data. This implies spending resources in an activity that can be done automatically. These facts show the need for devices able to reduce the collection cost, to avoid the data lost and to increasy the reliability of the observations.

In this work, a prototype device is designed and implemented to adquire and transmit automaticaly the observations of the stations. The prototype uses the transducers of a Davis 6162 Wireless Vantage Pro2 Plus station. A microcontroller with a Real Time Operative System was used to manage the adquisition, storage and transmission of the data. Data is saved in a SD memory and transmitted by an XBee module. Time and date references are provided by a Real Time Clock sincronized to a GPS. Also, the prototype allows to configure sampling and transmition periods and to test the correct performance. The carried out tests showed that the prototype adquires, stores and transmits the data correctly.

Keywords : Meteorological station, Real time operating system, XBee, OSA

Índice general

Índice de figuras

Índice de tablas

Cláusulas de licencia y autorización para publicación

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Marco Paúl Gutama Morocho, en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Prototipo de un dispositivo de adquisición, almacenamiento y transmisión de datos meteorológicos usando los transductores de la estación DAVIS 6162 Wireless Vantage Pro2 Plus", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 114 de la Ley Orgánica de Educación Superior.

Cuenca, 20 de noviembre de 2017

Marco Paúl Gutama Morocho
C.I.: 010548957-9

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Joel José Vázquez Patiño, en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Prototipo de un dispositivo de adquisición, almacenamiento y transmisión de datos meteorológicos usando los transductores de la estación DAVIS 6162 Wireless Vantage Pro2 Plus", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 114 de la Ley Orgánica de Educación Superior.

Cuenca, 20 de noviembre de 2017

Joel José Vázquez Patiño
C.I.: 010544075-4

Cláusulas de Propiedad Intelectual

Claúsula de propiedad intelectual

Marco Paúl Gutama Morocho, autor del trabajo de titulación "Prototipo de un dispositivo de adquisición, almacenamiento y transmisión de datos meteorológicos usando los transductores de la estación DAVIS 6162 Wireless Vantage Pro2 Plus", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 20 de noviembre de 2017

Marco Paúl Gutama Morocho
C.I.: 010548957-9

Claúsula de propiedad intelectual

Joel José Vázquez Patiño, autor del trabajo de titulación "Prototipo de un dispositivo de adquisición, almacenamiento y transmisión de datos meteorológicos usando los transductores de la estación DAVIS 6162 Wireless Vantage Pro2 Plus", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 20 de noviembre de 2017

Joel José Vázquez Patiño
C.I.: 010544075-4

Dedicatoria

Dedicado con mucho cariño a mi mamá, Flora, quien ha sido padre y madre a la vez, por todo su esfuerzo para ayudarme a alcanzar esta meta, por su paciencia y motivación a siempre superarme. A mis hermanos, Wilson, Isabel y Estefanía, por su comprensión, apoyo incondicional y ayuda que me brindaron en los momentos difíciles. Finalmente a mi sobrino, Matías Ismael, quien con su ternura me enseña muchas cosas, y con sus travesuras me alegra los días.

Marco Paúl

A mi papá Oswaldo, a mi mamá Carmen, a mi hermano Ángel, a mi hermana Cecilia y a mi sobrino Eliot. A todos ellos, por ayudarme en los momentos difíciles y por compartir conmigo los momentos alegres.

Joel José

Agradecimientos

Agradecemos al Ing. Remigio Guevara y al Ing. Fabián Astudillo, PhD por la dirección y ayuda brindadas para completar este trabajo.

Gracias a los ingenieros Remigio Guevara, Sebastián Abril e Iván Palacios de la Red Sísmica del Austro por su asesoría y ayuda técnica, su ayuda en el diseño inicial y la construcción del contenedor y las placas del diseño inicial y por las sugerencias en el diseño de la placa para que se adapte al contenedor. Además, gracias por facilitarnos el uso de algunos dispositivos.

Agradecemos al Programa para el Manejo del Agua y del Suelo (PROMAS) por facilitar el uso de sus estaciones, especialmente al Ing. Marco Ramírez, investigador, por ayudarnos con la instalación de las estaciones.

Agradecemos al Club de Robótica de la Universidad de Cuenca (CRUC) por permitirnos ser miembros y facilitarnos el uso de sus instrumentos y su espacio físico.

Agradecemos a Angel Vázquez-Patiño por los comentarios realizados acerca de la redacción del documento final y las observaciones acerca de la presentación de las figuras.

Agradecemos a nuestros familiares por el apoyo constante a lo largo de la carrera.

Marco Paúl Gutama Morocho

Joel José Vázquez Patiño

Abreviaciones y Acrónimos

ADC *Analog-Digital Converter*. 15, 16, 19, 20, 23, 25, 27, 41–46, 48–50, 52, 62, 67–70, 95

CCP *Captura/Comparación/PWM*. 47
CFE *Suma Acumulada de Errores de Pronóstico*. 90–93, 95
CNC *Control Numérico Computarizado*. 74, 76
CPU *Central Proccessing Unit*. 15, 23, 126, 127
CS *Chip Select*. 16

EMA *Estación Meteorológica Automática*. 11
EMC *Compatibilidad Electromagnética*. 68, 69, 71

FFD *Full Function Device*. 18

GPL *General Public License*. 124, 125
GPS *Global Positioning System*. 15, 20, 21, 25, 31, 50, 51, 53, 57–59, 63–65, 74, 87, 88, 90, 109, 111
GSM *Global System for Mobile communications*. 28, 81

HR *Humedad Relativa*. 8

I2C *Inter-Integrated Circuit*. 16, 39, 41, 42, 49, 50, 57
ISS *Integrated Sensor Suite* . 12

LAN *Local Area Network*. 17

MAD *Desviación Media Absoluta*. 90–93, 96
MED *Minimal Erythema Dose*. 13, 44
MEO *Medium Earth Orbit*. 20
MSE *Error Cuadrático Medio*. 90–93, 95, 96

NMEA *National Marine Electronics Association*. 21, 57, 58

OMM *Organización Meteorológica Mundial*. 7, 12, 43

PAN *Personal Area Network*. 17, 18
PCB *Printed Circuit Board*. 68, 73–75, 163, 164
PGA *Programmable Gain Amplifier*. 20
PROMAS *Programa para el Manejo del Agua y del Suelo*. 2, 3, 12, 88

RAM *Random Acces Memory*. 15, 59, 127
RFD *Reduced Function Device*. 18
ROM *Read-Only Memory*. 15
RTC *Real Time Clock*. 15, 20, 21, 49–51, 53, 57, 59, 65, 72, 109, 129
RTOS *Real Time Operating System*. 22, 23, 29, 30, 41, 50, 53, 57, 59, 62, 65, 109, 111, 123–126

SCI *Serial Communication Interface*. 16
SD *Secure Digital*. 21, 22, 26, 28, 31, 79, 88
SIM *Sensor Interface Module*. 12
SPI *Serial Peripheral Interface*. 16, 21, 41, 49, 121
SS *Slave Select*. 16

UART *Universal Asynchronous Receiver Transmitter*. 16, 65
USART *Universal Synchronous/Asynchronous Receiver/Transmitter*. 16, 21, 41
UV *Ultravioleta*. 10, 13, 19, 30, 33–35, 44, 51, 74, 90, 96, 112

WAN *Wide Area Network*. 17

Capítulo 1

Introducción

En este capítulo se trata la utilidad de una estación meteorológica y se presenta la justificación, el alcance, el aporte y los objetivos de este trabajo.

1.1. Presentación

El análisis de datos meteorológicos es una herramienta usada tanto para planificación de actividades como para el diseño de modelos climáticos. Muchas instituciones científicas y técnicas recolectan información meteorológica en diferentes puntos de interés muchas veces ubicados en puntos de difícil acceso y dispersos en un área geográfica de interés amplia. Su labor es facilitada si son disponibles en menor tiempo, mayor cantidad y con mayor precisión. Estos datos pueden ser recolectados de forma manual o automática [11], pero generalmente estas instituciones se valen de estaciones automáticas. Sin embargo, muchas de estas estaciones no cuentan con un sistema de envío automático de datos por lo que se requiere que el personal vaya a las estaciones para descargar la información.

Una de las instituciones que recolecta y usa datos meteorológicos es la Universidad de Cuenca a través del *Programa para el Manejo del Agua y del Suelo* (PROMAS). Este departamento de investigación tiene como parte de su misión, "contribuir al manejo sostenible de los recursos agua y suelo, usando nuevas tecnologías para alcanzar soluciones amigables con el ambiente y cumplir mejor sus objetivos científicos y de impacto social" [12]. En la actualidad, los datos de algunas de las estaciones del PROMAS son recolectados de forma manual en el sitio donde se encuentran instaladas.

Estas estaciones están compuestas por instrumentos destinados a medir diferentes variables meteorológicas. El uso de estas estaciones ha aumentado debido al avance tecnológico en el campo de la electrónica que ha permito un aumento de la disponibilidad y una disminución del precio de las mismas. Además, los componentes electrónicos han reducido su precio y han aumentando su disponibilidad. Estos factores permiten que haya cada vez más desarrollos locales que reducen la dependencia de tecnologías propietarias y aminoran los tiempos de adquisición y mantenimiento de los equipos.

1.2. Justificación

El PROMAS recolecta información meteorológica de aproximadamente 130 estaciones ubicadas en diferentes puntos de interés en la provincia del Azuay. Actualmente, estos datos son recopilados en un tiempo que va de 30 a 45 días dependiendo del lugar donde se encuentran las estaciones [13]. Este tiempo se puede reducir drásticamente con el uso de un sistema automático de envío de datos. Además, la recolección manual de los datos implica gastos en movilización, logística y tiempo. Un sistema de envío automático de los datos a las oficinas del PROMAS permitiría dirigir estos recursos a otras actividades.

Tomando en cuenta lo mencionado, se crea el proyecto "Aplicación de Tecnologías Inalámbricas al Monitoreo Climatológico en la Cuenca del Río Paute". Este proyecto está orientado a crear un sistema que permita que los datos se obtengan automáticamente y de forma remota en los servidores del PROMAS, así se reduce "de manera drástica el retardo" en la obtención de los datos [13]. Para eso se requiere la creación de un dispositivo capaz de recolectar los datos y de enviarlos para su registro en el servidor. En este trabajo se planteó reemplazar el dispositivo de registro de datos de la estación con uno desarrollado localmente, con el fin de tener la capacidad de formar una red de mayor cobertura que permita una adquisición de datos de forma remota y automáticamente.

Por este motivo, se diseño e implementó el prototipo de un dispositivo que permite la adquisición y transmisión automáticas de las medidas (observaciones) realizadas por los transductores meteorológicos de las estaciones. Entre las estaciones disponibles en el medio local están las del fabricante DAVIS INSTRUMENTS. Se escogió el Davis 6162 Wireless Vantage Pro2 Plus. El prototipo es capaz de usar todos los transductores con que cuenta la estación y, además, guarda la información recogida en una memoria como respaldo, con referencia de fecha y hora. También permite configurar los tiempos de muestreo y periodos de transmisión, y verificar el funcionamiento en el momento y en el lugar de instalación.

En el presente trabajo se usan los transductores de la estación Davis 6162 Wireless Vantage Pro2 Plus del fabricante DAVIS INSTRUMENTS. Los datos obtenidos de estos transductores son procesados y enviados mediante un sistema inalámbrico. Por lo tanto, este dispositivo es capaz de tomar los datos, volverlos digitales, almacenarlos y transmitirlos. De esta forma se cuenta con datos de las mismas variables que tenía la estación original y además se tiene la capacidad de transmitirlos a largas distancias.

1.3. Alcance

Este trabajo se centró en diseñar e implementar un dispositivo de costo económico y consumo de energía bajos, compatible con los transductores de la estación Davis 6162 Wireless Vantage Pro2 Plus del fabricante DAVIS INSTRUMENTS y con la capacidad de usar un sistema de transmisión de datos. Este dispositivo se encarga de obtener las señales, acondicionarlas y procesarlas para que se puedan codificar los datos de forma que reduzcan el tamaño de los paquetes de datos registrados con el fin de realizar una transmisión usando menos energía.

Los transductores de la estación comercial se conectan al dispositivo creado y los datos son adquiridos y almacenadas como paquetes listos a ser enviados, como se puede ver en la Figura 1.1. Se cuenta con almacenamiento de respaldo local de los datos y un archivo mediante el

cual se configuran parámetros de adquisición, muestreo y transmisión. La estación es capaz de medir radiación solar y ultravioleta, dirección y velocidad de viento, precipitación, temperatura y humedad, sin embargo, puede ser configurada para tomar muestras de solo algunas de las variables.

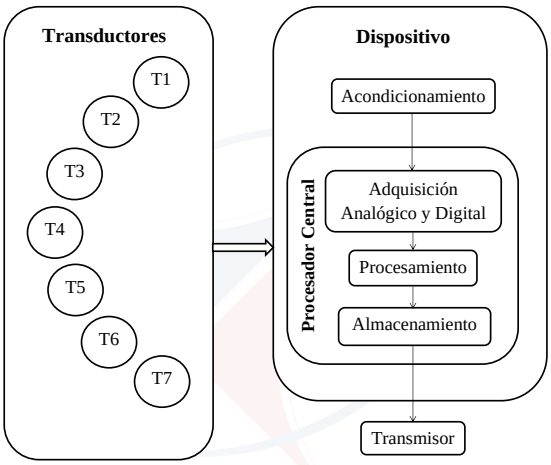


Figura 1.1: Esquema general del diseño del dispositivo construido.

El diseño incluye la posibilidad de conectar un módulo de comunicación que permita protocolos de comunicación avanzados como el manejo de paquetes y el enrutamiento. Sin embargo, la programación de estos protocolos no es parte del alcance de este trabajo. Se realiza el envío de los datos desde un dispositivo XBee (que usa un protocolo basado en IEEE 802.15.4) a otro. Se incluyen los datos necesarios para conocer hora y fecha, identificadores de estación y transductor a más de los datos recogidos.

Los transductores usan sistemas físicos diferentes por lo que el sistema electrónico de cada conector es específico para cada transductor. Todos se conectan a un microcontrolador que se encarga de procesar los datos. Un archivo contenido en la memoria local permite activar los transductores y configurar el tiempo de muestreo, entre otros parámetros. El sistema realiza un procesamiento previo al envío de los datos, pero éstos deben ser procesados en el receptor para obtener los valores en sus unidades correspondientes.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar e implementar un equipo de adquisición de información meteorológica compatible con los transductores de la estación Davis 6162 Wireless Vantage Pro2 Plus del fabricante DAVIS INSTRUMENTS con referencia de tiempo real y que permita la conexión de un dispositivo de transmisión de datos.

1.4.2. Objetivos específicos

1. Estudiar los transductores con que cuenta la estación Davis 6162 Wireless Vantage Pro2 Plus.
2. Realizar el acondicionamiento de las señales adquiridas desde los transductores de la Davis 6162 Wireless Vantage Pro2 Plus.
3. Diseñar e implementar un sistema electrónico para la adquisición, almacenamiento y transmisión de los datos, utilizando componentes de montaje superficial.
4. Evaluar el dispositivo contrastando sus mediciones con respecto a las mediciones obtenidas por la estación DAVIS 6162 Wireless Vantage Pro2 Plus.

Capítulo 2

Marco Teórico

Existen muchos elementos que componen y posibilitaron desarrollar este trabajo. Este capítulo trata los conceptos relacionados con el desarrollo del trabajo. Inicia describiendo las variables meteorológicas que se miden con el prototipo. Luego se menciona la estación y los transductores usados en este trabajo. Además, se exponen las características generales de los dispositivos electrónicos empleados, de sus sistemas de comunicación y de algunos periféricos. Algunas consideraciones para el sistema de alimentación también son presentadas. Finalmente, se mencionan algunas características de un sistema operativo de tiempo real, programa usado para gestionar los recursos del microcontrolador.

2.1. Antecedentes

El clima y el tiempo meteorológico (o simplemente tiempo) han sido siempre importantes porque influyen directamente en las actividades humanas. Los ciclos climáticos han permitido el desarrollo de la agricultura, el turismo, la industria pesquera y otras actividades que requieren conocerlo y se benefician de predicciones lo más exactas posibles. Conocer la evolución de variables meteorológicas a corto plazo permite que se den alertas de tempestades o inundaciones, de forma que se pueda actuar a tiempo para reducir, e incluso eliminar las pérdidas de cualquier tipo (sobre todo humanas) que se pueden presentar en tales eventos. La navegación, tanto aérea como marítima, también se beneficia de estas predicciones [14].

El clima se define como "síntesis de las condiciones meteorológicas en un lugar determinado, caracterizada por estadísticas a largo plazo de los elementos meteorológicos en dicho lugar", mientras que el tiempo es el "estado de la atmósfera en un instante dado definido por los diversos elementos meteorológicos" [15]. La climatología y la meteorología se encargan respectivamente de estos fenómenos. Las variables que se estudian son, entre otras: temperatura, precipitación, presión atmosférica, humedad, velocidad y dirección del viento, radiación y variables derivadas [16, 17].

Con el transcurso de los años, la predicción del clima y el pronóstico del tiempo se han vuelto más fiables. "Hoy en día, una predicción meteorológica a cinco días es tan fiable como lo era una predicción a dos días hace veinte años" [14]. Ésto se debe, en parte, a que se pueden obtener más datos con mayor precisión. Para mejorar la calidad de los datos se aplican normas internacionales en la precisión, obtención, intercambio y procesamiento de los mismos.

La organización más importante en meteorología es la *Organización Meteorológica Mundial* (OMM), institución autónoma, vinculada a las Naciones Unidas, cuyo fin es promover la investigación científica sobre la atmósfera y el cambio climático y facilitar el intercambio mundial de datos meteorológicos [18]. La OMM ha publicado varios documentos, el reglamento técnico [19] es uno de ellos; en este documento se encuentran recomendaciones sobre la manera adecuada de recolectar datos meteorológicos.

2.2. Variables meteorológicas

La Guía de Instrumentos y Métodos de Observación Meteorológicos (OMM-Nº 8) de la OMM [17] define prácticas recomendadas para la medición de variables meteorológicas y los sistemas de observación. A continuación se describe las variables de temperatura, humedad, viento de superficie, precipitación y radiación. Se expone la definición dada por la segunda

edición del Vocabulario Meteorológico Internacional [15].

Para conocer los instrumentos usados en este trabajo, referirse a la Sección 4.2.

2.2.1. Temperatura

La temperatura se define como la magnitud que caracteriza el movimiento aleatorio medio de las moléculas en un cuerpo [15]. Si dos cuerpos que se encuentran a diferentes temperaturas entran en contacto, ambos cuerpos terminarán a la misma temperatura. Un cuerpo que tiene mayor movimiento molecular interno se encuentra a mayor temperatura. Esta variable se puede medir en Kelvin (K) o en grados Celsius (°C).

La variable de interés en este trabajo es la temperatura del aire cerca a la superficie de la tierra. Su rango operacional depende de las características del lugar en donde se mide. Por ejemplo, si se quiere medir en una zona tropical, difícilmente se alcanzan temperaturas menores a 0 °C [20]. La medición de esta variable se ve afectada por factores como la radiación, el viento, la precipitación y la presencia de elementos circundantes por lo que el sensor debe ser cubierto por una protección que minimice su impacto. También, se recomienda que se realice la medición a una altura de entre 1.2 y 2 metros, en un emplazamiento horizontal con césped corto o con vegetación característica de la zona y donde el equipo no se vea afectado por la sombra o la radiación absorbida o generada por cuerpos cercanos.

Existen varios métodos para medir la temperatura y se valen de diversas propiedades físicas. Unos aprovechan propiedades térmicas o eléctricas de los materiales, otros se valen del espectro electromagnético para sus mediciones e incluso existen unos que usan el ultrasonido para determinar la velocidad media de las moléculas del aire. Los instrumentos que se usan son termómetros de líquido en cápsula de vidrio, termógrafos mecánicos y termómetros eléctricos.

Los termómetros eléctricos son los más usados actualmente en meteorología porque permiten registrar los datos, almacenarlos y transmitirlos a distancia. Más frecuentemente se usan resistencias eléctricas, termómetros de semiconductor o termistores y termopares. (Ver Sección 4.2.2.3).

2.2.2. Humedad

La Humedad se define como el vapor de agua contenido en el aire [15]. Generalmente se mide la *Humedad Relativa* (HR). Esta variable se define como la relación porcentual a una presión y temperatura dadas entre el peso molecular en gramos del vapor de agua y el peso molecular

en gramos que el aire tendría si estuviese saturado de agua en las mismas condiciones [21]. Es decir, es la relación, expresada en porcentaje, entre el vapor de agua presente en el aire y el valor de saturación, la cantidad máxima de vapor de agua que puede contener el aire sin que ésta se condense. Su unidad es el porcentaje (%). Las recomendaciones del emplazamiento son similares a las de temperatura por lo que se les pueden encontrar juntos.

Existen materiales que cambian sus propiedades en función de la humedad, esta característica es usada para desplazar una escala o para obtener una respuesta eléctrica. También se pueden aprovechar materiales con variación de sus propiedades eléctricas. Cada vez son más usados materiales de este tipo en resistores y capacitores debido a que se aplican en mediciones y registros electrónicos que pueden ser transmitidos remotamente.

Un tipo de sensor muy común es el capacitivo. En este tipo de sensores se usa corriente de alta frecuencia para excitar el elemento y medir su inductancia que varía porque el material dieléctrico se ve afectado por el vapor de agua. Generalmente, el circuito necesario para la medición viene integrado en el sensor. (Ver Sección 4.2.2.3).

2.2.3. Viento

El viento es el movimiento del aire con respecto a la superficie de la tierra [15]. Este movimiento se da en tres dimensiones, pero la componente vertical generalmente no es considerada en las estaciones a nivel de tierra. De esta forma el viento es considerado una cantidad vectorial de dos dimensiones representada por la velocidad y dirección en un tiempo dado [20].

Los instrumentos que miden la velocidad y dirección de viento son conocidos como anemómetros. Los anemómetros de cazoletas y de hélice usan un rotor que tiene una velocidad angular proporcional a la velocidad del viento. Las veletas se alinean con la dirección del viento y tienen un transductor que transmite esta posición, generalmente representada como una tensión eléctrica.

En la actualidad se usan anemómetros con un interruptor de lengüeta (reed-switch) que se cierra a cada vuelta dada por una cazoleta o una hélice. En el caso de la dirección de viento se usan, generalmente, potenciómetros o conmutadores giratorios como transductores. En este trabajo se usa una cazoleta y un potenciómetro. (Ver Secciones 4.2.2.2 y 4.2.1.3).

2.2.4. Precipitación

La precipitación es un hidrometeoro que consiste en la caída de un conjunto de partículas [15]. La precipitación puede tener diversas formas, entre ellas: lluvia, nieve, granizo y niebla. La unidad usada típicamente es la profundidad lineal en milímetros, esta unidad equivale a litros de agua en un metro cuadrado. Generalmente, se mide la cantidad de lluvia precipitada.

El pluviómetro es el instrumento más usado para medir la cantidad de agua precipitada. Consiste en un recipiente con un cilindro recto y un embudo, se ubica a una altura de entre 0,5 y 1,5 metros para evitar salpicaduras desde el suelo. Existen también otros tipos de medidores, como por ejemplo uno que acumula y almacena el agua entre periodos de observación. En la actualidad son muy comunes los medidores con registro automático.

Existen pluviómetros de tres tipos: de pesaje o pesada, de cubeta basculante o balancín y de flotador. El de pesaje mide todo tipo de precipitación, mientras que los otros se limitan a la lluvia.

El pluviómetro de balancín mide la caída de lluvia. Usa un recipiente con dos compartimentos; el agua de lluvia llega a uno de ellos mientras el otro permanece vacío. Al llenarse uno de los compartimentos el recipiente se inclina y se vacía a la vez que se empieza a llenar el otro compartimento. Este movimiento, conocido como tip, se usa para generar un pulso que permite determinar el tiempo que tarda en caer una cantidad conocida de lluvia. Este instrumento debe estar siempre nivelado para reducir los errores. (Ver Sección 4.2.2.1).

2.2.5. Radiación Solar

La radiación es la emisión o transferencia de energía en forma de ondas o partículas electromagnéticas. La radiación solar es importante debido a que tiene efectos sobre la biología, la medicina, la agricultura y otros aspectos de la naturaleza y las actividades humanas. Una parte de esta radiación atraviesa la atmósfera y llega a la superficie terrestre, otra parte es dispersada y absorbida. El instrumento de medición más común es el piranómetro que mide la radiación solar global, es decir, la suma de la radiación directa y la difusa. (Ver Sección 4.2.1.1).

2.2.6. Radiación ultravioleta

La parte de radiación solar que corresponde a radiación UV ha ganado importancia debido a que la reducción de la capa de ozono ha provocado un aumento en su intensidad en la superficie. El espectro de radiación UV comprende longitudes de onda desde los 100 nm hasta los 400 nm

y se divide comúnmente en tres partes:

UV-A: entre 315 y 400 nm. No es muy activa biológicamente y no se ve afectada por el ozono atmosférico. Es la que continúa al espectro de luz visible;

UV-B: entre 280 y 315 nm. Es la más activa biológicamente e interactúa con el ozono atmosférico. Es la responsable del efecto eritémico (inflamación de la piel caracterizada por manchas rojas); y

UV-C: entre 100 y 128 nm. Es absorbida totalmente por la atmósfera.

Lo que se mide generalmente es la radiación UV-B debido a su importancia biológica. Su distribución es muy variable espacialmente y depende principalmente de la época del año y de la distribución del ozono [22]. (Ver Sección 4.2.1.2).

2.3. Estaciones meteorológicas automáticas

Una *Estación Meteorológica Automática* (EMA) es una "estación meteorológica en la que se realizan y se transmiten observaciones automáticamente" [17]. Es decir, no se requiere de una persona para tomar medidas y registrar los datos. Existen dos tipos: en tiempo real y fuera de línea. Las del primer tipo proporcionan datos de lo que ocurre en el momento. Por otro lado, las del segundo tipo registran los datos en dispositivos internos o externos y deben ser enviados o recolectados bajo petición. Una estación puede funcionar como cualquiera de los dos tipos o incluso ambos.

Muchas EMAs brindan la posibilidad de formar redes. Una red está definida como un conjunto de estaciones de un mismo tipo que se administran como un grupo [16]. La ubicación de las estaciones de una red es dictada por la variación de las características climáticas del lugar.

Se recomienda que las observaciones se realicen siempre a la misma hora del día con el fin de mantener la homogeneidad de los datos. En caso de cambios en la hora o en las estaciones se deben seguir ciertas recomendaciones que aseguran que si existen variaciones en las mediciones se deban a variaciones reales y no al cambio de hora introducido. Si es necesario usar un nuevo instrumento de medida o realizar un cambio, ya sea en los instrumentos o en la ubicación, se debe tener funcionando el sistema actual con el anterior al menos por un año, y de preferencia dos, para evaluar si dicho cambio produce un efecto en los datos [16].

Una EMA está compuesta por sensores, un registrador de datos (*datalogger*) y equipos periféricos: un sistema de energía, un reloj de tiempo real, dispositivos de conexión y visualización, entre otros instrumentos. Los sensores pueden ser analógicos, digitales o "inteligentes". Los

sensores analógicos básicamente transforman la información en niveles continuos de potencial eléctrico, los digitales codifican la información mediante dos niveles (alto y bajo) y los sensores "inteligentes" cuentan con un propio microcontrolador que procesa y transmite la información al procesador principal.

El *datalogger* se encarga de la adquisición de datos y el tratamiento para que puedan ser entendidos por un computador o mostrados en una pantalla. Consta de al menos un sistema de procesamiento central encargado de gestionar los sensores y acondicionar su señal, procesar los datos y conectarse con los periféricos. Tiene también un sistema de almacenamiento y un sistema de comunicación.

La ubicación de las estaciones es importante para garantizar que el registro de datos sea representativo y homogéneo. La estación debe estar ubicada en un lugar que permita que los instrumentes estén expuestos de forma adecuada a la variable que se desea medir y no debe estar afectada por circunstancias externas al clima, e.g. obstáculos que puedan crear un clima puntual diferente. Existen recomendaciones en cuanto a la instalación y ubicación de las estaciones dadas por la OMM [17] y acotaciones particulares de cada fabricante.

2.4. Estación Davis Vantage Pro2 Plus

Existe muchos modelos de estaciones meteorológicas comerciales disponibles. El PROMAS cuenta, entre otras, con las de los fabricantes HOBO[1] y Davis[2]. También otras empresas, como CAMPBELL SCIENTIFIC[3] hacen disponibles sus *dataloggers*. Otras empresas cuentan con sensores como SIAP+MICROS[4] [23]. Para este proyecto se han escogido los transductores de la estación Davis Vantange Pro2 Plus, en su versión inalámbrica. Esta estación viene con sus propios sensores y *datalogger*. Además, el fabricante ofrece información más detallada en su sito web[5] .

La estación, compuesta por sus sensores y *datalogger*, es conocida como Conjuto de Sensores Integrado o *Integrated Sensor Suite* (ISS). El Módulo de interfaz de sensores (*Sensor Interface Module* (SIM)) es el dispositivo en dónde se conectan los sensores y es el encargado de acondicionar y almacenar las señales para su posterior envío de forma inalámbrica. Todos los sensores se conectan a través de un conector RJ-11. La estación cuenta con una consola inalámbrica que permite configurar las características de los transductores. La consola también tiene un modo gráfico que permite apreciar de manera visual la evolución de las variables. Además, se puede

[1] `http://www.onsetcomp.com/`
[2] `https://www.davisnet.com/`
[3] `https://www.campbellsci.es/`
[4] `http://www.siapmicros.com/`
[5] http://www.davisnet.com/product/wireless-vantage-pro2-with-24-hour-fan-aspirated-radiation-shield/

comunicar a una computadora para descargar los datos y configurar los transductores [24].

La estación Davis Vantage Pro2 Plus tiene sus sensores incluidos: anemómetro, pluviómetro, sensor de temperatura y humedad, sensor de radiación UV y sensor de radiación solar. A continuación se detalla brevemente cada sensor de la estación.

Anemómetro: Está compuesto por una cazoleta con un interruptor de lengüeta que cierra el contacto a cada vuelta para determinar la velocidad del viento, y una veleta, adaptada a un potenciómetro, para determinar su dirección. Por defecto, el brazo del anemómetro debe apuntar hacia el norte para obtener una lectura correcta, pero se puede configurar la dirección norte.

Pluviómetro: Usa un balancín que activa un interruptor de lengüeta magnético cada vez que cambia de posición al volcarse el agua (tip). La resolución es de 0,01 pulgadas (in) o 0,2 mm (si se tiene puesto un adaptador para tener las medidas en unidades internacionales) [25].

Sensor de temperatura y humedad: Está protegido del medio ambiente para evitar el error inducido por la exposición directa al sol o por el viento. Internamente cuenta con un circuito integrado SHT11 [5], el cual es un sensor digital de temperatura y humedad. Usa una comunicación 2-wire Serial Interface (Interfaz serial de dos vías). La humedad se mide con un capacitor y la temperatura mediante un sensor *band-gap* (brecha de banda), que usa la diferencia de potencial inducida por la temperatura en un diodo semiconductor. Este sensor cuenta con un ventilador que evita el calor por radiación en el protector [26].

Sensor de radiación solar: Mide la Radiación solar global: radiación directa del sol más la difusa. Es también conocido como piranómetro. Este sensor requiere ser calibrado. Usa un fotodiodo de silicio de amplio espectro de respuesta con un amplificador que genera una salida de 0 a 3 V. La lectura de salida de este sensor tiene una equivalencia de 1,67 mV por W/m^2 [27].

Sensor de radiación ultravioleta: Mide la radiación UV global, compuesta por la radiación UV que llega de forma directa y la que es reflejada y dispersada por la atmósfera, siendo esta última la componente más significativa. La consola puede presentar la radiación UV en las escalas de índice UV y Dosis Eritemática Mínima (*Minimal Erythema Dose (MED)*) [6]. Este sensor requiere ser calibrado. Su salida está en el rango de 0 a 2,5 V con una equivalencia de 150 mV por Índice UV y 364 mV por MED/h. [28].

La Tabla 2.1 lista el intervalo de actualización, la resolución, el rango y el error de los datos medidos de forma directa por la estación y el número de producto del sensor que se usa para cada una.

[6] El índice UV es una medida de intensidad de radiación UV. Un W/m^2 corresponde a 40 índices UV. La MED es una medida de irradiación o densidad de energía. Un Wh/m^2 equivale a 3600/210 MED

Variable	N° de producto	Intervalo actualización	Resolución	Rango	Error
Dirección de viento	6410	2,5 a 3 segundos	1°	0 a 360°	3°
Radiación Solar	6450	50 s a 1 min	1 W/ m^2	0 a 1800 W/ m^2	5 % de escala
Índice UV	6490	50 s a 1 min	0,1	0 a 16	5 % de escala
Temperatura	6832	10 a 12 segundos	0,1° F 0,1° C	-40° a 150° F -40° a 65 ° C	1° F 0,5° C
Humedad	6832	50 s a 1 min	1 %	1 a 100 %	3 % HR $\leq$ 90 % 4 % HR >90 %
Dosis UV	6490	50 s a 1 min 5 min si está oscuro	0,1 MED < 20 1 MED > 20	0 a 199 MEDs	5 % de escala
Tasa de lluvia	7857	20 a 24 segundos	0,01 in 0,1 mm	0 a 96 in/h 0 a 24 mm/h	<5 % o 0,04 in/h; 1 mm/h
Velocidad de viento	6410	2,5 a 3 segundos	1 mph 1 nudo 0,4 m/s 1 km/h	2 a 200 mph 2 a 173 nudos 3 a 322 km/h 1 a 809 m/s	<2 mph/nudos 1 m/s 3 km/h

Tabla 2.1: Número de producto, intervalo de actualización, resolución, rango y error de los datos tomados por la estación Davis Vantage Pro2 Plus.

2.5. Sistemas embebidos

Un sistema embebido es aquel que tiene su software y hardware incluidos, dedicados a una aplicación específica, en una parte de un producto, o como parte de un sistema más grande [29]. Los sistemas embebidos se encuentran en todas partes, por ejemplo, electrodomésticos, aparatos médicos, periféricos de computación, vehículos y en hogares inteligentes. Todos ellos usan un microcontrolador, que cuenta con memoria de programa, memoria de datos , temporizadores y varios periféricos de comunicación, los cuales están integrados en un solo chip. Adicionalmente, cuentan con algunos complementos como dispositivos ADC, *displays*, teclados o memorias [29].

En este trabajo se ocupan un microcontrolador, ADCs, un módulo GPS, un reloj de tiempo real (RTC), dos multiplexores y una memoria SD, a más de componentes electrónicos comunes como resistores, capacitores y diodos y, como un componente del software, un sistema operativo de tiempo real.

2.5.1. Microcontrolador

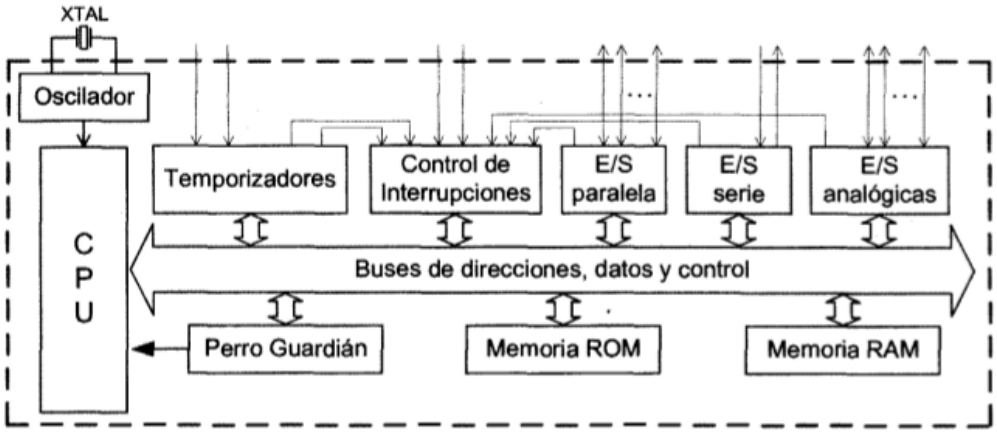


Figura 2.1: Diagrama de bloques general de un microcontrolador, fuente: [1].

Un microcontrolador se encarga de un pequeño grupo de tareas para lo cual ejecuta permanentemente un programa grabado en su memoria, trabaja con datos almacenados temporalmente e interactúa con el exterior usando buses de entrada y salida [1]. La Figura 2.1 muestra un diagrama de bloques con los componentes comúnmente encontrados en un microcontrolador.

Un microcontrolador cuenta con los siguientes elementos [1]: oscilador, unidad de procesamiento central (*Central Proccessing Unit* (CPU)), memoria de acceso aleatorio (*Random Acces Memory* (RAM)), memoria de sólo lectura (*Read-Only Memory* (ROM)), puertos o buses de en-

trada y salida y ADCs. También cuentan con sistemas como el de *reset*, modo de bajo consumo y protección ante copia.

2.6. Comunicación con los periféricos

Existen protocolos de comunicación seriales y paralelos. Se trata sobre la comunicación serial por ser la más utilizada en la actualidad y por ser la que se usa en este trabajo.

Existen varios protocolos de comunicación serial que se han desarrollado. La transmisión en serie de la información consiste en enviar los bits de un mensaje en forma sucesiva a través de un mismo terminal o pin. La velocidad de transmisión se mide como la cantidad de bits que se transmiten en un tiempo dado, más comúnmente bits por segundo (bps). Los equipos que se comunican deben conocer el inicio y el final de la cadena de datos y existen dos formas de hacerlo: transmisión síncrona y transmisión asíncrona. La primera usa una señal de reloj, mientras que la segunda usa bits adicionales para indicar el inicio y el fin [1]. Los siguientes son protocolos de comunicación serial con los periféricos.

Universal Synchronous/Asynchronous Receiver/Transmitter (USART): También conocido como *Serial Communication Interface (SCI)*, usa dos hilos y puede ser configurado de forma síncrona bidireccional simultánea o asíncrona bidireccional no simultánea [1]. Existe una forma más simple que puede ser solo asíncrona y es conocida como *Universal Asynchronous Receiver Transmitter (UART)*.

Serial Peripheral Interface (SPI): Es un protocolo de comunicación sincrónica de 8 bits mediante el que se puede conectar un maestro y varios esclavos. Usa dos pines de datos, SDO y SDI, salida y entrada respectivamente; además, usa un terminal de reloj (SCK). Este protocolo permite conectar a la vez varios esclavos que pueden ser activados o desactivados mediante un terminal con el que cuentan conocido como *Slave Select (SS)* o *Chip Select (CS)*. La señal de reloj es una terminal de salida para el maestro y una de entrada para los esclavos.

I2C: Este protocolo de comunicación, conocido también como I^2C, es una comunicación síncrona. Usa dos terminales conocidos como SDA y SCL que son para datos y el reloj respectivamente. El maestro fija la frecuencia de trabajo y entrega la señal de reloj a los esclavos. El maestro se dirige a un esclavo a la vez para lo cual usa su dirección única. No usa terminales adicionales para incluir a un nuevo esclavo [1].

Single-wire, two way serial protocol: Este protocolo de comunicación usa un pin de datos entre el maestro y el esclavo. Debe usar una resistencia de *pull-up* en esta línea [30]. Se usa para aplicaciones más lentas que las anteriores. Es también conocida como *1-wire bus*.

2-wire Serial Interface: Este protocolo fue usado inicialmente por ATMEL. Es un protocolo genérico industrial síncrono que cuenta con las terminales SDA y SCL que son de datos y de reloj respectivamente. El maestro puede manejar las dos terminales, mientras que los esclavos solo pueden enviar datos cuando el maestro lo requiere [31].

2.7. Comunicación entre dispositivos

La comunicación entre dispositivos requiere de mayor fiabilidad debido a que puede cubrir distancias mayores. Se usan protocolos de comunicación como los de las tecnologías *Personal Area Network (PAN)* (Bluetooth, Z-Wave, ZigBee), *Local Area Network (LAN)* (Ethernet, WiFi) o incluso *Wide Area Network (WAN)* (Wimax, GPRS, LTE). En este trabajo se usa el protocolo ZigBee.

2.7.1. ZigBee

Zigbee es un protocolo de comunicación inalámbrica, usado entre dispositivos, basada en el estándar IEEE 802.15.4. Este estándar está orientado a aplicaciones con una baja tasa de transmisión de datos y con bajo consumo de energía, como monitoreo y control de sensores [2]. La ventaja de ZigBee es que está orientado a una parte no cubierta por otras tecnologías, como *Bluetooth* o WiFi. El primero cuenta con mayores tasas máximas de transmisión, pero menor rango de alcance; mientras que el segundo está sobredimensionado para las aplicaciones mencionadas (ver Figura 2.2).

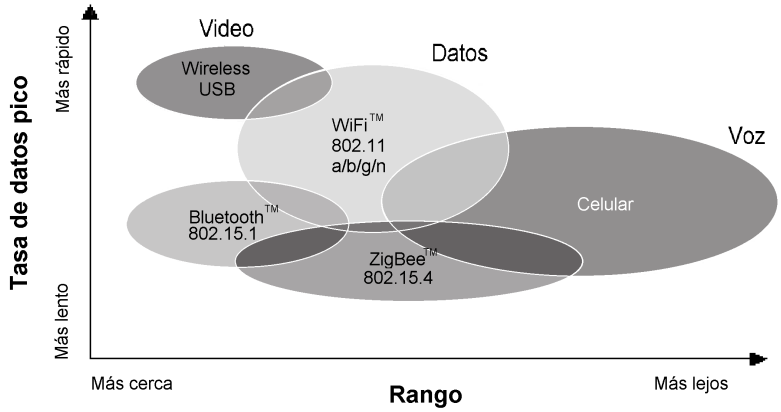


Figura 2.2: Gráfico del lugar que ocupan diferentes tecnologías según su tasa de datos máxima y su rango de alcance, fuente: [2].

En una red ZigBee los dispositivos (radios) pueden comportarse como: coordinador, enruta-

dor o dispositivo final [32]. A su vez el protocolo IEEE 802.15.4 define dos clases de dispositivos: *Full Function Device (FFD)*, que son capaces de cumplir con todo el protocolo, y *Reduced Function Device (RFD)*, que cumplen sólo con parte de las funcionalidades del protocolo [33], lo cual resulta útil, por ejemplo, para ahorrar energía en dispositivos que sólo requieren enviar datos de forma intermitente.

Toda red ZigBee tiene uno y sólo un coordinador y cualquier cantidad de enrutadores, incluso cero. El coordinador se encarga de formar la red, asignar direcciones y manejar otras funciones que definen la red, como la seguridad. Mientras que un enrutador se puede unir a una red, recibir y enviar información, enrutar mensajes entre dispositivos y enviar sus propios mensajes.

Los dispositivos finales son los de menor consumo de energía. Se pueden unir a una red y pueden enviar y recibir información, pero no pueden ser mediadores entre otros dispositivos. Estos son los únicos dispositivos que pueden entrar en un modo de bajo consumo (*sleep*). Siempre necesitan unirse a la red mediante un dispositivo *padre*, sea un enrutador o coordinador, encargado de almacenar sus mensajes cuando está en modo de bajo consumo. Por su reducido número de funciones, son los únicos dispositivos que pueden ser RFD.

ZigBee puede conectar diferentes tipos de productos a la vez y se pueden configurar en diferentes topologías de red [32]: emparejamiento (*peer-to-peer*), árbol (tree), malla (mesh) y estrella(*star*). En la topología *peer-to-peer* se unen sólo dos dispositivos. En su firmware, uno de ellos debe ser coordinador y el otro puede ser enrutador o dispositivo final. Esta es la configuración más simple. En la configuración de árbol, los dispositivos enrutadores y finales se unen al coordinador a través de una rama, pero no entre ramas. En la configuración de malla todos los enrutadores se comunican entre ellos para llevar el mensaje. Es la más robusta debido a que se puede auto configurar para hacer frente a cambios inesperados, como la pérdida de alguno de los enrutadores. En la configuración de estrella los dispositivos se pueden comunicar entre ellos, pero solo a través del coordinador. Cada red tiene una identificación única conocida como dirección *PAN* .

2.8. Otros dispositivos

2.8.1. ADC

Las variables meteorológicas son fundamentalmente analógicas, por lo que se requiere convertirlas en digitales para procesarlas en un microcontrolador. Algunas de estas señales ya se convierten en digitales por sus métodos de adquisición: el pluviómetro (cantidad lluvia) y el

anemómetro (velocidad de viento) usan un interruptor de lengüeta que crea pulsos digitales, y el sensor usado para la temperatura y humedad es digital. Sin embargo, para las señales de dirección de viento, radiación solar y radiación UV las señales siguen siendo analógicas. Es por eso que se necesita de un ADC.

Los ADCs son dispositivos electrónicos que transforman una diferencia de potencial en un conjunto de bits que la representan de forma digital. Los ADCs realizan este proceso mediante dos pasos principales: muestreo (digitalización en tiempo) y cuantificación (digitalización en amplitud) [34]. La Figura 2.3 muestra el diagrama de bloques de un ADC ideal.

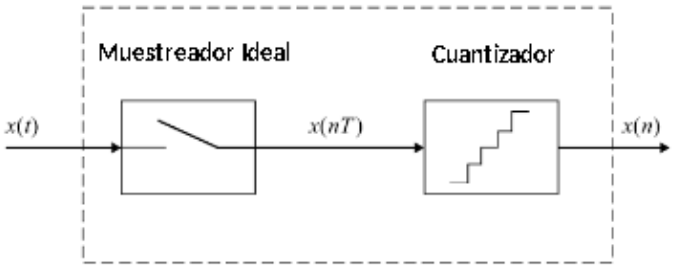


Figura 2.3: Diagrama de bloques de un ADC ideal, fuente: [3].

En el muestreo, idealmente se toma datos con un intervalo fijo y cada valor se mantiene hasta que se toma el siguiente. Este intervalo se conoce como periodo (T). El inverso de T es la frecuencia de muestreo, f_s. Según el teorema de muestreo de Shannon [3], f_s debe ser mayor a dos veces la mayor frecuencia contenida en la señal muestreada, frecuencia conocida como frecuencia de Nyquist. Las componentes de frecuencias mayores a la de Nyquist afectan la señal como ruido mediante un efecto conocido como *aliasing* que se puede observar en la Figura 2.4. Para eliminar tal efecto se usa un filtro que, idealmente, elimina todas las frecuencias mayores a la máxima deseada sin modificar las menores.

La cuantificación de una señal consiste en asignar a cada valor continuo un valor relacionado tomado de un conjunto finito [35]. Al reducir el conjunto de valores de salida a un número finito se produce un error, conocido como error de cuantificación, cuyo valor máximo es la mitad de un bit y su frecuencia tiene una distribución uniforme. Este número de valores es propio del ADC e igual a 2^n, donde n es el número de bits con que cuenta.

Existen varios tipos de ADC según la electrónica interna que usan para realizar la conversión: contador, aproximaciones sucesivas, convertidor paralelo, convertidor en subrangos y convertidor delta-sigma ($\Delta\Sigma$) [35].

Los convertidores delta-sigma consisten de un modulador de sobremuestreo seguido por un filtro digital de decimación que juntos producen una salida de datos de alta resolución. Su

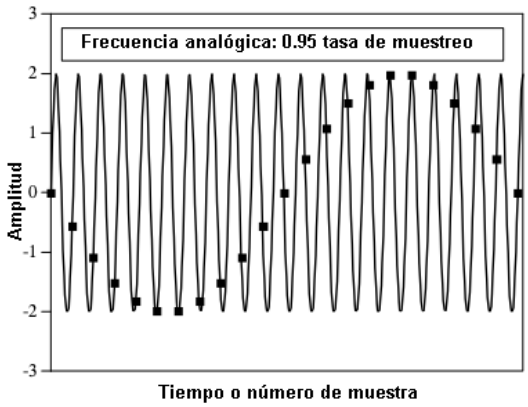


Figura 2.4: Señal muestreada a una frecuencia menor a la adecuada, fuente: [3].

frecuencia de entrada es mayor que su frecuencia de salida. Usa un modulador $\Delta\Sigma$, un filtro digital y un decimador [36]. El filtro digital permite que el filtro *antialiasing* sea más simple.

Los ADCs integrados generalmente vienen con un *Programmable Gain Amplifier (PGA)* que permite que señales de una amplitud muy pequeña sean convertidas con mayor definición. También, cuentan con varios canales y un multiplexor que permite su uso. Además, al ser elementos periféricos incluyen una forma de comunicación.

2.8.2. RTC

Un Reloj de tiempo real (RTC) se usa como referencia de tiempo en los dispositivos embebidos que requieren de cierta sincronización. Comúnmente se usan RTCs en circuitos integrados debido a su bajo consumo de energía. Los RTCs usan un oscilador de cristal externo de 32,768 kHz del cual depende la exactitud de los mismos. Estos relojes funcionan de forma independiente del microprocesador, el cual se comunica con éste para configurar y consultar la hora y fecha. En este trabajo este dispositivo fue usado con el fin de tener una referencia de hora y fecha en los datos.

2.8.3. GPS

El Sistema de Posicionamiento Global (GPS)[1] es un sistema de navegación mundial que otorga la ubicación y hora a un número ilimitado de usuarios que cuenten con un receptor [37]. Usa satélites de órbita circular intermedia (*Medium Earth Orbit* (MEO)) y entre sus

[1] http://www.gps.gov/

aplicaciones están: navegación vehicular, telefonía móvil, agricultura, construcción y minería y redes de comunicación. Los datos entregados por los dispositivos de aplicación siguen el formato NMEA-0183 [38], definido por la *National Marine Electronics Association (NMEA)*[2]. Estos datos luego son procesados para ser legibles por un usuario.

Los datos que entrega el GPS permiten conocer, entre otras cosas, la ubicación geográfica del dispositivo y también la hora y fecha. En el presente trabajo se usó este dispositivo para obtener la hora y fecha con el fin de igualar el RTC.

2.8.4. Multiplexores y demultiplexores

Los multiplexores sirven para tomar varias señales de entrada y enviar una de ellas a la salida [34]. Los multiplexores son usados si existen varias señales que deben ser transmitidas a un solo receptor. Permiten controlar qué señal de entrada pasa a la salida. Por otro lado, un demultiplexor hace lo inverso a un multiplexor: toma una entrada y la envía a una de varias salidas [34]. Así mismo, la salida que tendrá la señal puede ser seleccionada.

Los multiplexores y los demultiplexores pueden ser vistos como interruptores que cierran su contacto en la línea de interés. En ambos, la señal es escogida mediante un código en la entrada de selección. Generalmente el número de terminales de la entrada de selección está determinado por el número de canales de datos que contiene porque la señal de selección es un conteo binario: si se tiene dos canales basta una entrada de selección, pero en caso de tener cuatro, se requiere de dos entradas. Es decir, el número de canales es menor o igual a una potencia en base dos del número de líneas de selección.

En este trabajo, el microcontrolador debe comunicarse con varios dispositivos mediante el puerto serial USART (ver Sección 2.6), sin embargo, cuenta con un solo módulo. Dado que algunos de estos dispositivos deben funcionar simultáneamente, se usó un multiplexor y un demultiplexor en el TX y el RX del puerto del microcontrolador.

2.8.5. Almacenamiento

Se requiere que el sistema pueda respaldar la información un tiempo que permita realizar reparaciones o mantenimiento. Se usa una tarjeta de memoria *Secure Digital (SD)* por contar con suficiente capacidad de respaldo y porque se puede extraer fácilmente para una revisión. Además, estas memorias pueden ser grabadas a través del protocolo SPI [39]. El proceso de lectura y escritura se facilita debido a la disponibilidad de librerías implementadas para su

[2] http://www.nmea.org/

manejo en microcontroladores.

Las memorias SDs son de tamaño reducido, con alta capacidad, no volátiles y regrabables [39]. Su uso está dirigido a dispositivos móviles y son fabricadas cada vez con mayor capacidad. El estándar de uso es mantenido por la SD Association[3].

Existen tres sistemas de archivos compatibles con estas memorias para el almacenamiento de la información: FAT16, FAT32 y exFAT. En este proyecto se usa FAT32 ya que la capacidad de almacenamiento deseada se encuentra en el rango permitido por este sistema de datos: mayor a 2 GB y hasta de 32 GB. En el Apéndice A se muestra la distribución de terminales de las diferentes versiones de tarjetas SD.

2.9. Alimentación del sistema

La alimentación del sistema es importante para la implementación puesto que en sistemas de tiempo real es primordial el uso eficiente de la energía. La energía requerida depende de los componentes del sistema. Por ejemplo, algunos circuitos deben ser alimentados con voltajes negativos y positivos, por lo que requieren de una fuente simétrica. Por lo general, se usan valores de tensión de 5 o 3,3 V, pero hay dispositivos que pueden funcionar en un rango más amplio y otros no permiten cambiar la alimentación sin modificar su comportamiento.

La alimentación puede venir del sistema de tendido eléctrico, o mediante baterías y fuentes de energía alternativas, que son más portátiles. En proyectos que se van a ubicar en lugares aislados es importante tomar en cuenta la autonomía del sistema. En estos casos se recurre a sistemas de energía alternativa, como la fotovoltaica, hidráulica o eólica, y a baterías para la reserva de energía en ausencia de su fuente de generación.

El prototipo diseñado en este trabajo cuenta con un regulador de voltaje de 3,3 V. Está destinado a ser alimentado por un sistema manejado por un controlador de carga. Un panel solar entrega la energía al sistema y una batería sirve de almacenamiento.

2.10. Sistema operativo de tiempo real

Un sistema operativo de tiempo real *(Real Time Operating System (RTOS))* es un programa que permite gestionar recursos e implementar tareas simultáneas en un microprocesador o microcontrolador. Aunque las instrucciones se ejecutan secuencialmente, el RTOS permite

[3]https://www.sdcard.org/index.html

gestionar los tiempos de retardo para dar una sensación de ejecución paralela. Por ejemplo, al utilizar el módulo ADC de un procesador, es necesario un tiempo de retardo para que la conversión sea correcta; el RTOS permitirá aprovechar este tiempo para ejecutar otras tareas.

Los RTOSs están construidos en un núcleo (*kernel*) multitarea que controla y administra las tareas. Cada tarea se ejecuta durante un periodo (*time slice*). Luego de su periodo la tarea es detenida y reemplazada por otra. Este proceso se repite continuamente hasta que las tareas se terminen. Cuando ocurre la conmutación, los registros de la tarea en ejecución se guardan en la memoria, los registros de la nueva tarea se cargan al CPU y la nueva tarea comienza a ejecutarse. Un RTOS provee otros servicios como paso de mensajes entre tareas (*task-to-task message passing*), sincronización de tareas y ubicación de recursos compartidos [39]

La implementación de un RTOS depende de las características del procesador y de la aplicación a la que se vaya a dedicar. Para el caso de microcontroladores, existen RTOSs tanto libres como de pago. En este trabajo se usa un microcontrolador libre para evitar costos asociados a licencias y para tener mayor control y conocimiento del sistema.

Se optó por usar un RTOS con el fin de que se gestionen mejor los recursos y debido a la cantidad de tareas que debe realizar el microprocesador. De esta forma se puede usar un procesador más sencillo y aprovechar mejor los tiempos de procesamiento y, por ende, la energía.

Capítulo 3

Trabajos Relacionados

Este capítulo presenta aspectos sobresalientes de trabajos relacionados al diseño de *dataloggers* en diversas áreas. Se analiza el diseño de su sistema embebido, el microprocesador usado, la forma en que almacena y transmite los datos obtenidos de los sensores y el tipo de sistema de alimentación.

La tecnología ha permitido que se desarrollen estaciones cada vez más complejas y de mejores prestaciones, que han sido usadas con el fin de realizar mediciones de forma automática más fiables y con menores interrupciones. Estas estaciones, que generalmente forman parte de una red, están adecuadas para resistir la intemperie y funcionan de forma autónoma. Estos adelantos tecnológicos han permitido que se recolecten más datos y de mejor calidad.

El acceso a la tecnología y las condiciones en la que se emplazan las estaciones son diferentes dependiendo del lugar donde se encuentren. Por lo tanto, distintas soluciones tienen lugar de acuerdo a las restricciones locales.

En [40] se desplegó una red de sensores inalámbricos en el volcán Reventador (Ecuador). Estos sensores son más livianos y pequeños que los tradicionales y cuentan con un sistema de transmisión a distancia de datos. Se instalaron 16 nodos equipados a lo lago de 3 km en una red multisaltos (*multihop*) con un enlace de radio de larga distancia entre ellos. Se usó una antena omnidireccional de 8dBi a 2.4 GHz, un sismómetro, un micrófono y una interfaz de hardware personalizada en cada nodo de la red.

El microcontrolador usado fue un MSP430 del fabricante Texas Instruments y se usó una memoria flash externa de 1 MB. Para la transmisión se usa el transceptor Chipcon CC2420 que funciona con el protocolo IEEE 802.15.4. Se usó un ADC AD7710 de 24 bits. Cada nodo tiene 2 o 4 canales sismoacústicos que almacenan los datos temporalmente en una memoria flash local. La energía de cada nodo proviene de dos baterías alcalinas tipo D, que fueron cambiadas dos veces en tres semanas de uso.

El dispositivo corre con un sistema operativo. Los paquetes de datos son divididos en bloques de 256k con su *time stamp* y luego se dividen en fragmentos más pequeños con un identificador. Luego, un modem de radio Freewave [1] permitió una conexión de larga distancia entre la red de sensores y la computadora del observatorio.

Para proteger al circuito electrónico de condiciones ambientales adversas se usa una caja del fabricante Pelican[2] impermeable y hermética y se usaron conectores resistentes al medioambiente.

Se usó el sistema para tomar muestras de eventos limitados en el tiempo, no de forma continua. Una interfaz de usuario permite establecer tiempos de muestreo y umbrales para los eventos. También se usa una tabla con información sobre el estado de los nodos. Con el fin de reducir los falsos positivos, se registra un evento cuando ha sido captado por varios nodos a la vez. La hora es tomada de un GPS y se acepta una tolerancia de tiempo de 10 ms. Como resultado se obtuvieron datos el 61 % del tiempo.

[1] `http://www.freewave.com/`
[2] `http://www.pelican.com/`

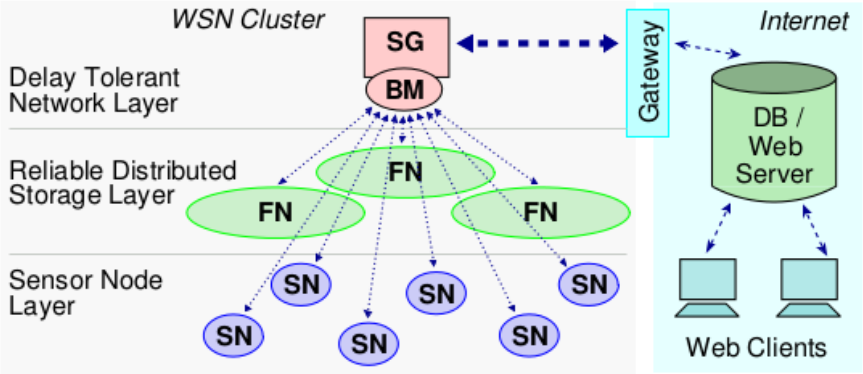


Figura 3.1: Arquitectura de red de LUSTER [4].

El trabajo descrito es aproximado a lo que se pretende implementar, con ciertas diferencias debido a su aplicación. En el monitoreo sísmico es preferible muestrear con una frecuencia elevada únicamente eventos, mientras que en el monitoreo climatológico se deben registrar los valores de las variables todo el tiempo. También existe diferencia en la cantidad de tipos de sensores que lleva la estación. En las sismológicas se usaron únicamente sensores sísmicos y micrófonos de baja frecuencia, pero en las estaciones meteorológicas se requiere registrar varias variables. La alimentación proviene de baterías que deben ser cambiadas. El presente trabajo no afrontará la necesidad de cambios de baterías para mantener el sistema funcionando, salvo casos poco probables de daño en el panel solar o en la batería. Significa una mayor inversión en el sistema de alimentación a cambio de un sistema que genera menores costos de mantenimiento.

Otro trabajo relacionado a redes de sensores es LUSTER [4], una red de sensores tolerante a fallas usada principalmente para vigilar la luz ambiental en matorrales. Esta red también puede medir otros parámetros biológicos como la temperatura, la humedad del suelo y la concentración de CO_2.

Es una red jerárquica, organizada como se muestra en la Figura 3.1. Una capa de almacenamiento distribuido la hace tolerante a fallas y a retrasos. Tiene una interfaz para validar el funcionamiento de la red en tiempo de despliegue. Cuenta con sensores de luz reconfigurables. El hardware se encuentra en recipientes completamente herméticos con cables sellados para evitar problemas relacionados a la humedad.

El almacenamiento distribuido está provisto por una red de sensores que replican la información de todos los nodos que se encuentran a un salto. Estos datos son almacenados en una tarjeta SD extraíble que puede ser leída de forma remota o en el lugar, además se la puede reemplazar sin dejar de tomar datos. La red se puede organizar de dos formas: áreas sobrepuestas, las cuales lo vuelven tolerante a pérdidas de información y a mensajes corruptos porque cada nodo sensor tiene más de un nodo de almacenamiento, o áreas limitadas, que reducen la capacidad

de almacenamiento y su ancho de banda necesarios porque no se replica la información.

Esta red usa el protocolo IEEE 802.15.4 y una pasarela comunica entre este protocolo y el IEEE 802.11 para hacer disponible la información en una base de datos visible en la web.

El sistema [4] es fiable en su aplicación puesto que cuenta con equipos redundantes; sin embargo, esto implica un aumento en el costo de la red y aumento en el trabajo necesario para implementar los equipos. En el caso del trabajo actual, se cuenta también con una memoria SD que permite tener un respaldo de los datos que se puede consultar en el lugar en el caso de fallas en el sistema de transmisión, así brinda un respaldo. También se puede comprobar el funcionamiento de cada nodo mediante un modo, llamado depurador, que permite verificar los datos que recibe y envía la estación mediante una conexión serial.

El trabajo descrito en [23] consiste en el diseño e implementación de un dispositivo que reemplaza a un *datalogger* y es usado con sensores comerciales. Se planteó una arquitectura en la que los sensores son conectados a placas independientes y todas ellas se conectan a una placa principal. Las placas de los sensores son adaptable a sensores analógicos o digitales mediante el cambio en su programa.

Las placas de los sensores procesan los datos y los guardan temporalmente como *buffer* y respaldo. Usan un ADC de 22 bits para tomar los datos y un banco de memorias para respaldarlos hasta por 6 días. Estas placas usan un microprocesador de prestaciones menores a las del microprocesador de la placa principal, la cual realiza las peticiones de datos a las placas de cada sensor.

Los datos recibidos en el *datalogger* son almacenados en un *buffer* y luego en una memoria SD. La comunicación con los procesadores de los sensores se realiza mediante el protocolo RS485, que permite una comunicación a distancias grandes y con varios esclavos multiplexando dispositivos RS232. Se realizaron pruebas con tres procesadores diferentes: PIC18F4620, Raspberry Pi model B y Arduino Mega 2560. Como resultado se reportó que la plataforma PIC es la de costo y consumo menores a pesar de ser la que más dificultad presenta en programación. Mientras se envían los datos se detiene el muestreo. Así, algunos datos son perdidos debido al tiempo que toma el envío. Estas pérdidas son grandes en Arduino y PIC, siendo menores en el segundo. El Raspberry Pi se desempeña mejor que los otros debido a que solo en esta plataforma se usaron hilos.

En este trabajo, a diferencia del de [23], los sensores son conectados directamente a la placa del *datalogger* y existe sólo un microprocesador que se encarga del procesamiento, almacenamiento y envío de los datos. De esta forma se elimina la necesidad de usar el protocolo RS485 y sus elementos complementarios. También se logra el uso de menos elementos, ya que los sensores cuentan con un conector específico adaptado a sus necesidades. También se implementan hilos

en el microprocesador PIC, así se logra realizar las tareas necesarias con un microcontrolador de menor capacidad. Debido a estas diferencias se logra un menor costo y un menor consumo de energía.

En [41] se desarrolló localmente un dispositivo que envía los datos tomados por un sensor de lluvia a través de mensajes SMS. Este sistema es limitado por la cobertura de la red *Global System for Mobile communications* (GSM). Los datos del sensor son obtenidos mediante un microprocesador. Este microprocesador se conecta con el sensor a través del puerto RS-232 y con un módulo GSM a través del puerto USB, el cual emula una terminal TTY y envía comandos AT. Este sistema se conecta a los *dataloggers* comerciales, de esta forma no se requirió desarrollar un dispositivo que tomara los datos del sensor, sino uno que analizara los datos que se enviaban.

El sistema de alimentación consiste en un panel solar que carga un banco de baterías conmutables. Es decir, el sistema es alimentado por una batería y al mismo tiempo el resto de baterías es cargado por los paneles. El microcontrolador se encarga de la conmutación entre las baterías.

En el trabajo [41] se depende de la red GSM. En el presente trabajo se usa la comunicación a través de una red propia diseñada con dispositivos XBee hasta llegar a un punto en donde exista dicha cobertura. De esta forma se elimina el problema de la falta de cobertura de la red GSM.

El prototipo diseñado e implementado en el presente trabajo permite que los datos sean recogidos todo el tiempo y que se envíen a un servidor central a través de una red implementada. Los datos también son respaldados localmente por el prototipo en la memoria SD. También, se logra un ahorro de energía al tener todo integrado haciendo que se use un microprocesador por estación. Además, se tiene la versatilidad de activar y configurar los sensores deseados hasta los siete disponibles. También se puede verificar el funcionamiento del sistema en la instalación. De esta forma se consigue una estación de bajo costo pero que reemplaza a una estación comercial y adicionalmente dispone de comunicación a distancia.

Capítulo 4

Diseño del Dispositivo

En el presente capítulo se describen los criterios considerados para el diseño del circuito electrónico que permite el acondicionamiento y adquisición de las señales de los sensores meteorológicos de la estación Davis Vantage Pro2 Plus y la interacción con otros componentes periféricos. Se explica la implementación de un RTOS en un microcontrolador para procesar, almacenar y transmitir los datos de los sensores. Además, se indican los parámetros considerados para el diseño del circuito impreso final y su contenedor.

4.1. Introducción

Para diseñar el dispositivo primero se analizaron los transductores de la estación para realizar el acondicionamiento y adquisición de las señales. Luego, se creó un programa que se ejecuta en un RTOS para procesar y gestionar los datos. Finalmente, se especificaron los criterios usados para la fabricación del contenedor. En la Figura 4.1 se muestra el diagrama de bloques en base al cual ha sido diseñado e implementado el prototipo.

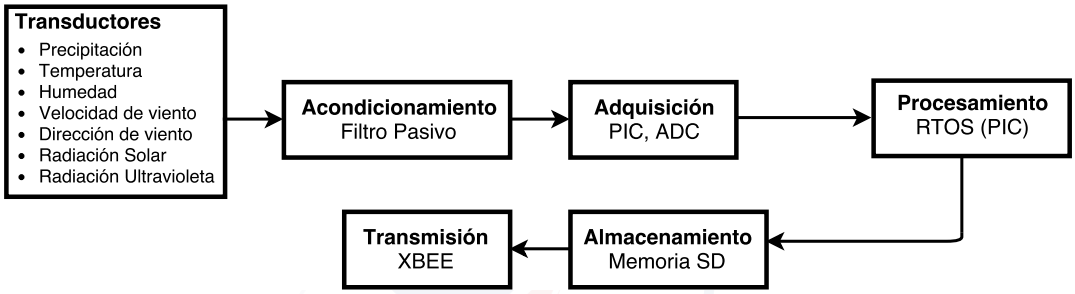


Figura 4.1: Diagrama de bloques de diseño del prototipo.

4.2. Acondicionamiento de señales

La estación meteorológica cuenta con sensores magnéticos *reed-switch* para las variables de precipitación y velocidad de viento, un sensor digital para la temperatura y humedad, y sensores analógicos para las variables de radiación solar, radiación UV y dirección del viento. Todos los sensores de la estación usan un conector RJ-11. A continuación se detalla cada sensor y el acondicionamiento realizado a sus señales.

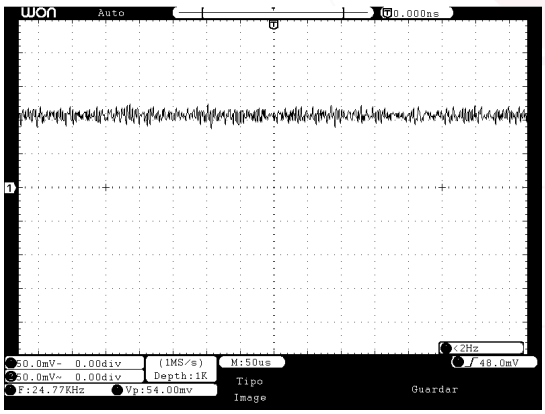
4.2.1. Señales analógicas

4.2.1.1. Radiación solar

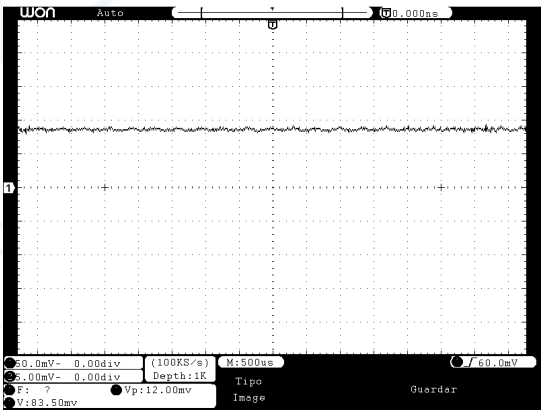
El sensor de radiación solar es un piranómetro analógico. El sensor utiliza un fotodiodo de silicio, el cual convierte la radiación incidente en corriente. Éste viene integrado con un amplificador que tiene una salida entre 0 y 3 Voltios proporcional a la corriente del fotodiodo. El sensor está protegido por una carcasa externa que además tiene un indicador de nivel para alinear el sensor de forma correcta. La Figura 4.2 muestra el sensor de radiación solar de la estación.

Figura 4.2: Sensor de radiación solar de la estación Davis 6162 Wireless Vantage Pro2 Plus.

La señal analógica a la salida del sensor de radiación solar se muestra en la Figura 4.3a. Esta señal presenta un ruido alrededor de los milivoltios, lo cual puede llegar a ser significativo ya que la señal presenta variaciones de 1,67 mV por W/m^2. Este ruido es externo al circuito, ya que al cambiar la fuente conmutada por una batería de 12 V, el ruido se ve reducido como se muestra en la Figura 4.3b. En la Figura 4.4 se muestra el ruido que genera la fuente conmutada y la batería.



(a) Señal usando una fuente conmutada

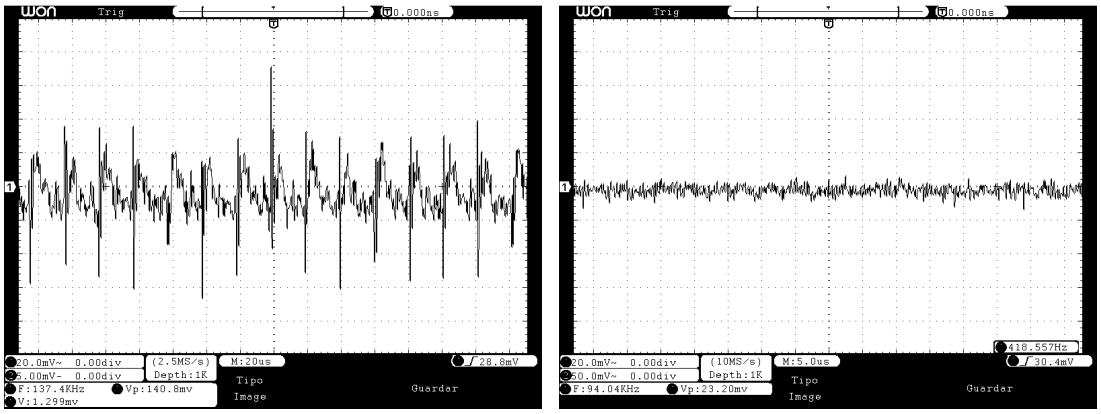


(b) Señal usando una batería

Figura 4.3: Señal de radiación solar, en escala de 50 mV por división.

Para observar como afectan componentes periféricos que se plantea utilizar tales como XBee, GPS y memoria SD y todos los sensores a las señales analógicas, se desarrolló un PCB de pruebas con estos componentes (Anexo F.4). La señal que se obtiene al conectar el sensor al circuito se muestra en la Figura 4.5a. El ruido aumenta debido a que los componentes periféricos utilizan comunicaciones seriales que distorsionan la señal analógica, además existen otras posibles fuentes de ruido como las señales de radiofrecuencia del XBee.

La radiación solar varia de forma lenta, es una señal que no presenta cambios bruscos. Por



(a) Fuente conmutada (b) Batería

Figura 4.4: Ruido de fuente, en escala de 20 mV por división.

lo tanto, la frecuencia que mide el osciloscopio en la Figura 4.5a de aproximadamente 1 kHz corresponde al ruido presente en la señal. Para eliminar el ruido se utiliza un filtro paso bajo RC de primer orden (Figura 4.6) con una frecuencia de corte por debajo de la frecuencia de la señal de ruido. La frecuencia de corte está dada por la Ecuación (4.1).

$$f_{corte} = \frac{1}{2\pi RC} \tag{4.1}$$

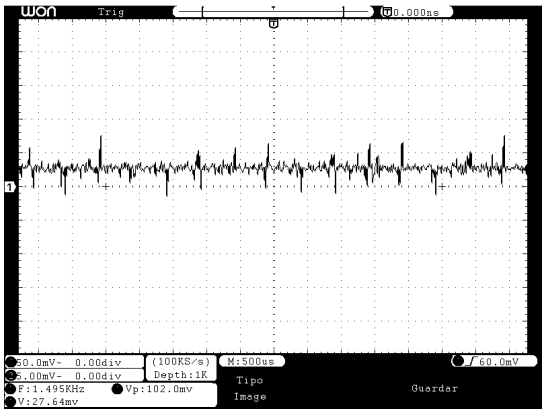
donde:

R: valor de la resistencia en Ohms (Ω)
C: valor del capacitor en Faradios (F)

Si se establece una frecuencia de corte de $f_{corte} = 15\,Hz$, un capacitor con un valor comercial común de $C = 100\,nF$, entonces despejando R de la Ecuación (4.1) se tiene que el valor de la resistencia es:

$$\begin{aligned} R &= \frac{1}{2\pi f_{corte} C} \\ &= \frac{1}{2\pi \cdot 15\,Hz \cdot 100\,nF} \\ &= 106103{,}3\,\Omega \approx 100\,k\Omega \end{aligned}$$

En la Figura 4.5b se muestra la señal filtrada, sin embargo, esta señal tiene características de ruido blanco. En la Sección 4.3.1 se muestra los resultados de realizar un promedio de muestras



(a) Señal analógica en circuito con todos los sensores conectados y componentes periféricos.

(b) Señal con filtro RC.

Figura 4.5: Radiación solar, en escala de 50 mV por división.

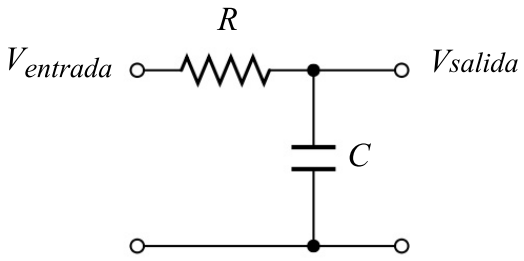


Figura 4.6: Filtro RC.

para filtrar este ruido. En la Figura 4.7 se presenta las conexiones realizadas al sensor para adquirir la señal.

4.2.1.2. Radiación Ultravioleta

El sensor de radiación UV funciona de manera similar al sensor de radiación solar, diferenciándose solamente por el tipo de fotodiodo. El sensor de radiación UV utiliza un fotodiodo semiconductor, el cual responde a la irradiancia UV. El armazón de protección del sensor es semejante al de radiación solar, como se muestra en la Figura 4.8.

Similar al caso de la radiación solar, el resto de sensores y componentes que forman parte del diseño del dispositivo distorsionan la señal analógica de UV en la misma proporción que la señal de radiación solar (Figura 4.9). Para filtrar el ruido se utiliza un filtro paso bajo con los mismos criterios seguidos para el sensor de radiación solar.

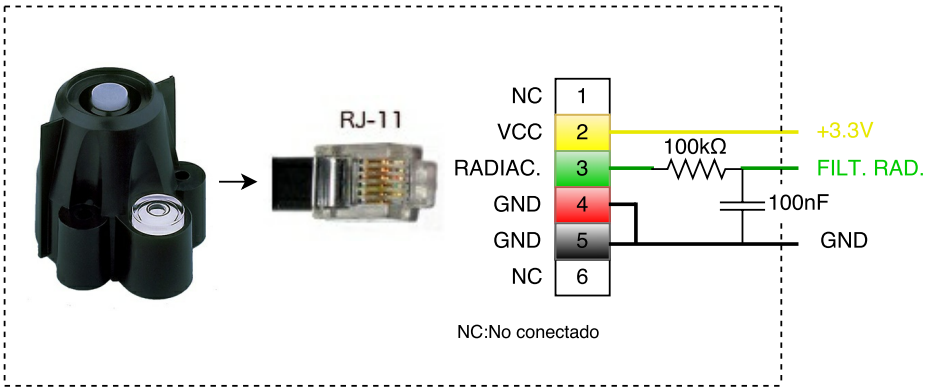


Figura 4.7: Conexión del sensor para adquirir la señal de radiación solar en el prototipo. Se muestra los pines de salidas y el filtro paso bajo implementado previo a la adquisición de la señal para su digitalización.

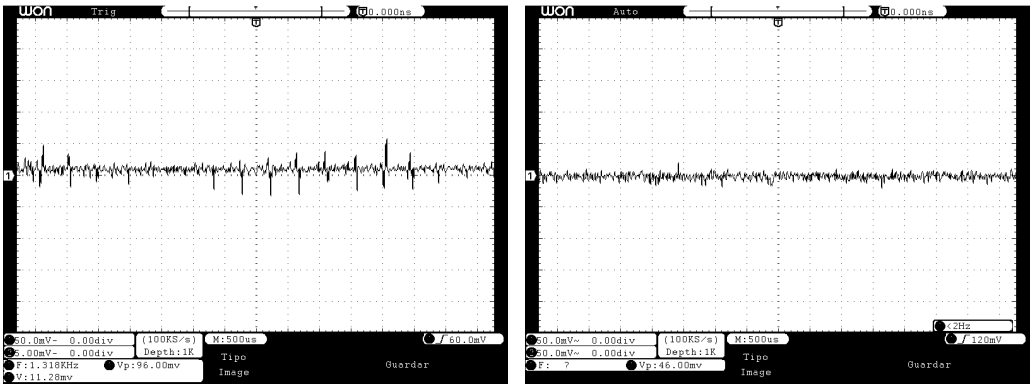
Figura 4.8: Sensor de radiación UV de la estación Davis 6162 Wireless Vantage Pro2 Plus.

En la Figura 4.9 se muestra la señal de radiación UV con ruido y la señal después del filtro. En la Figura 4.10 se muestra los pines de salidas del sensor y el filtro paso bajo implementado previo a la adquisición de la señal para su digitalización.

4.2.1.3. Dirección de viento

El anemómetro que posee la estación mide la velocidad y la dirección de viento y usa solo un conector. Para establecer la dirección del viento, el sensor utiliza una veleta como se muestra en la Figura 4.11. Esta veleta está internamente conectada a un potenciómetro lineal de 20 kΩ, el cual cambia su resistencia dependiendo de la dirección del viento. Así, el voltaje que se lee del sensor varía de acuerdo a la resistencia presente en el potenciómetro.

Para filtrar el ruido producido por el resto de sensores y otros componentes, se utiliza un filtro RC con las mismas características a los filtros implementados para radiación solar y UV.



(a) Señal analógica en circuito con todos los sensores conectados y componentes periféricos.

(b) Señal con filtro RC.

Figura 4.9: Radiación UV, en escala de 50 mV por división.

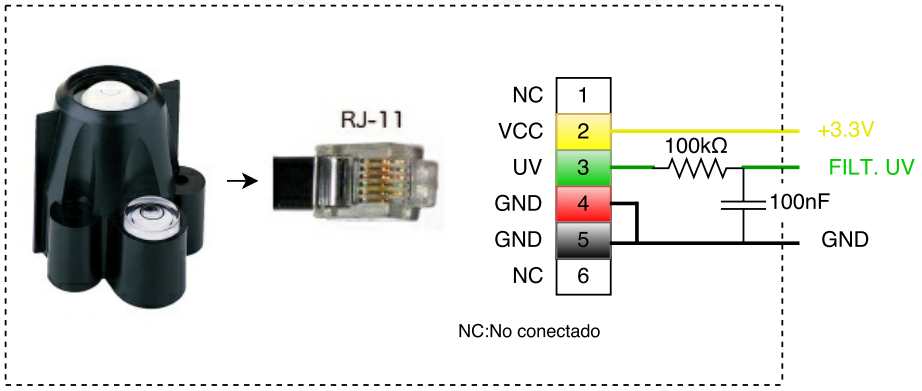


Figura 4.10: Conexión del sensor para adquirir la señal de radiación UV en el prototipo. Se muestra los pines de salidas y el filtro paso bajo implementado previo a la adquisición de la señal para su digitalización.

En la Figura 4.12 se presenta el acondicionamiento realizado al sensor de dirección de viento.

4.2.2. Señales digitales

4.2.2.1. Pluviómetro

El pluviómetro con el que cuenta la estación utiliza un sistema de balancín para medir la intensidad de lluvia. A través del colector (Figura 4.13a), el agua pasa a los depósitos del balancín (Figura 4.13b), en donde al llenarse un depósito, éste cae y se vacía, mientras el otro extremo pasa a captar el agua de lluvia, y así sucesivamente. El balancín posee un imán y en la

Figura 4.11: Veleta de anemómetro de la estación Davis 6162 Wireless Vantage Pro2 Plus.

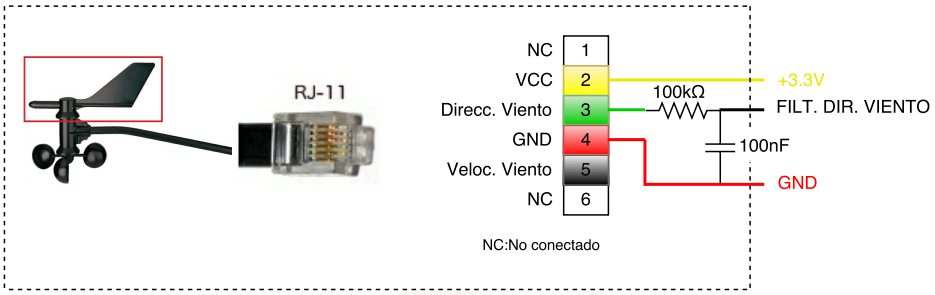


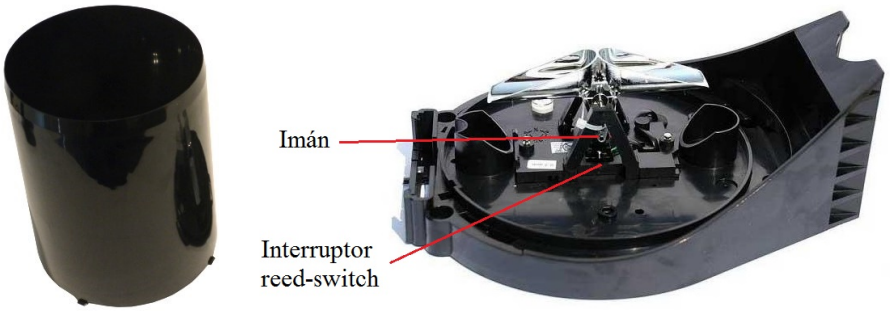
Figura 4.12: Conexión del anemómetro para dirección de viento. Se muestra los pines de salidas y el filtro paso bajo implementado previo a la adquisición de la señal para su digitalización.

base se encuentra un interruptor *reed-switch*[1] que normalmente está abierto. En cada cambio de posición del balancín, el imán pasa sobre el interruptor, lo cual provoca que éste se cierre y abra nuevamente. De ésta manera, cada pulso generado por el interruptor equivale a 0,2 mm de agua [25].

Los pulsos generados por el interruptor son detectados por el microcontrolador. En cada conmutación del interruptor se producen rebotes de contacto o ruidos de tensión eléctrica que el microcontrolador puede detectarlos como falsos pulsos de conmutación. En la Figura 4.14 se muestra el ruido presente en los cambios de estado del interruptor del balancín que dura aproximadamente 28 μs en flanco de bajada y 740 ns en flanco de subida. Para eliminar éste ruido se hace uso de un filtro RC antirebote, el cual utiliza la naturaleza del capacitor para oponerse a cambios bruscos de tensión, suavizando así el ruido producido por las cambios de estado del interruptor. El tiempo de carga y descarga de un capacitor (el cual debe ser mayor o igual a la duración del ruido por rebotes) esta dado por la Ecuación (4.2).

$$\tau = RC \tag{4.2}$$

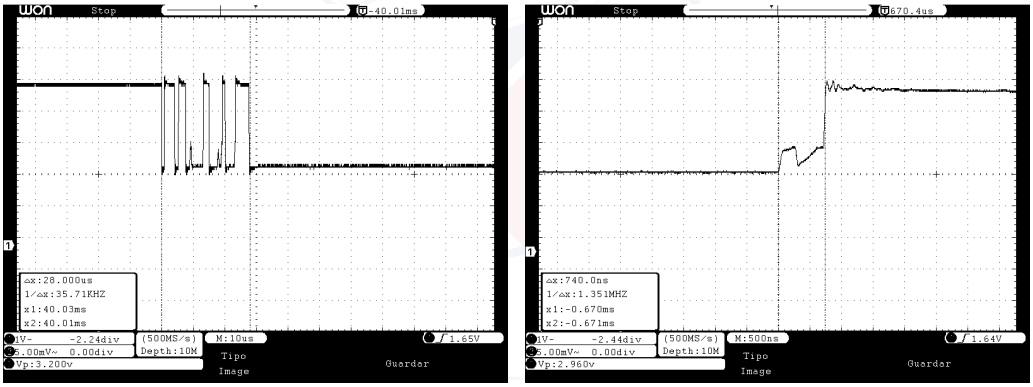
[1] Interruptor eléctrico activado por un campo magnético.



(a) Colector de agua de lluvia

(b) Balancín del pluviómetro, Fuente: [42]

Figura 4.13: Pluviómetro de la estación Davis 6162 Wireless Vantage Pro2 Plus.



(a) Flanco de bajada

(b) Flanco de subida

Figura 4.14: Rebotes de conmutación del interruptor *reed switch*

Si en la Ecuación (4.2) se despeja la capacitancia C, y se reemplaza el tiempo de carga y descarga del capacitor por $\tau \geq 1\ ms$, y la resistencia con un valor comercial de $R = 10k\Omega$, se encuentra $C = 0{,}1\ \mu F$. En la Figura 4.15 se muestra la implementación del filtro y las conexiones realizadas para el pluviómetro.

4.2.2.2. Velocidad de viento

Para medir la velocidad de viento se utiliza un anemómetro de cazoletas, que consiste en tres semiesferas huecas posicionadas en el mismo sentido y acopladas, como se muestra en la Figura 4.16. Independientemente de la dirección del viento, la rotación de las semiesferas es siempre en la misma dirección.

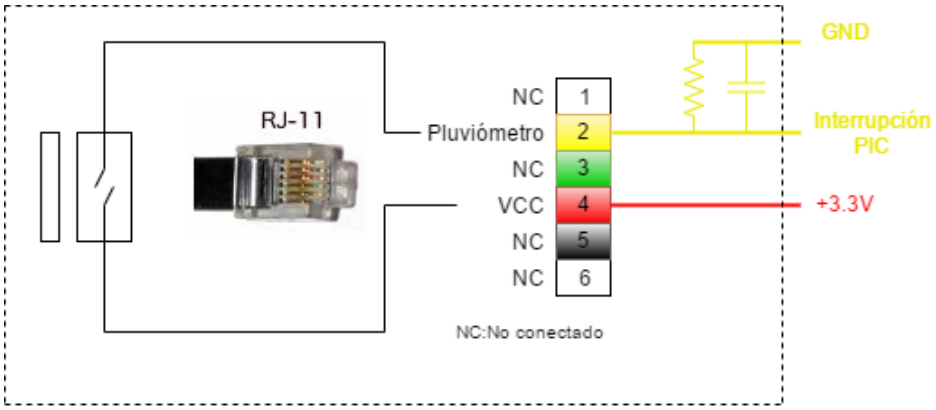


Figura 4.15: Conexión del pluviómetro en el prototipo. Se muestra los pines de salidas y el filtro implementado previo a la adquisición de la señal.

(a) Sensor de velocidad de viento

(b) Configuración de semiesferas, Fuente: [43]

Figura 4.16: Cazoletas del anemómetro de la estación Davis 6162 Wireless Vantage Pro2 Plus.

El conjunto de semiesferas está acoplado a un interruptor magnético *reed-switch*. El interruptor se cierra y abre cada vuelta completa, generando pulsos a la salida del sensor. Similar al acondicionamiento realizado al pluviómetro, se utiliza el mismo filtro RC para eliminar los rebotes de conmutación. Los pulsos generados por el interruptor son detectados por el microcontrolador. En la Figura 4.17 se muestran las conexiones realizadas al anemómetro para medir la velocidad de viento, y el filtro implementado previo a la adquisición de la señal para su digitalización.

4.2.2.3. Sensor de temperatura y humedad

La temperatura y humedad relativa del aire es medida mediante el sensor integrado SHT11. Este integrado está cubierto por un protector con una red interna que evita el ingreso de polvo

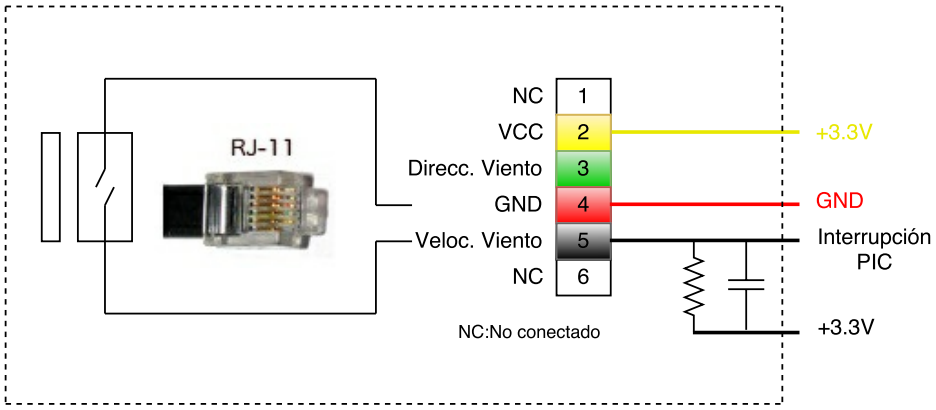


Figura 4.17: Conexión del sensor de dirección de viento. Se muestra los pines de salidas y el filtro paso bajo implementado previo a la adquisición de la señal para su digitalización.

e insectos al sensor (Figura 4.18b). Este conjunto a su vez está blindado por cinco platos que protegen al sensor de la intemperie; los dos primeros encima del sensor tienen una abertura circular y están agujereados por el borde como se muestra en la Figura 4.18c, posibilitando de está manera una ventilación que permite al sensor realizar mediciones correctas. Todo el conjunto de blindaje ya ensamblado se ve en la Figura 4.18a.

El sensor se comunica con un microcontrolador mediante un protocolo bus serie *2-wire* propio del fabricante del sensor. Se transmite dos señales, una por cada línea:

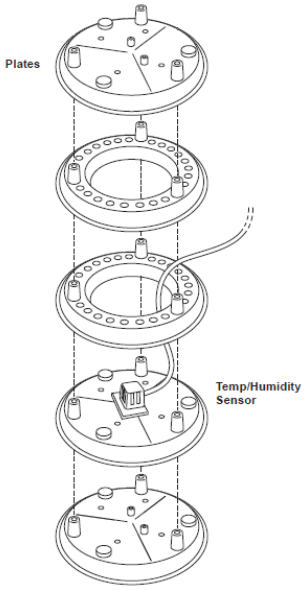
1. SCK (Serial Clock). Señal de reloj generada por el microcontrolador para sincronización de datos, y
2. DATA (Serial Data). Línea para transferencia de datos.

La línea DATA es bidireccional y debe estar conectada a tensión positiva a través de una resistencia de *pull-up* (Rp), generalmente de 10 kΩ, para llevar la señal a estado alto [5]. El sensor se alimenta con una tensión de 3.3 V de acuerdo a la recomendación del fabricante, para una mayor exactitud en la mediciones, esto debido a la calibración del sensor. En la Figura 4.19 se muestra el esquema de conexión del sensor SHT11 con un microcontrolador.

El protocolo de comunicación utilizado por el sensor puede convivir con el protocolo I2C, es decir, puede utilizar las mismas líneas de comunicación sin interferir con I2C [5]. Esta ventaja es aprovechada para comunicar el microcontrolador con el sensor y otros dispositivos I2C en un mismo bus. El bus I2C requiere de resistencias de *pull up* en las dos líneas de comunicación. Las conexiones para el sensor de temperatura y humedad relativa se muestra en la Figura 4.20.

(a) Blindaje de protección.

(b) Protector con membrana interna.



(c) Configuración de platos, fuente: [26].

Figura 4.18: Sensor de temperatura y humedad relativa de la estación Davis 6162 Wireless Vantage Pro2 Plus.

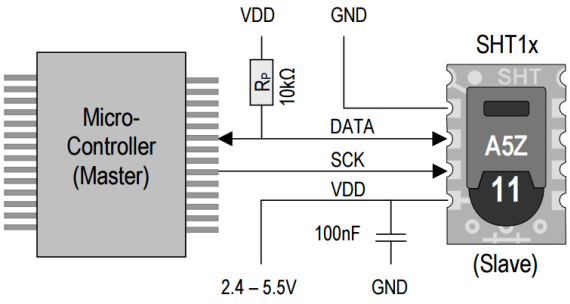


Figura 4.19: Conexión típica del sensor SHT11, fuente: [5].

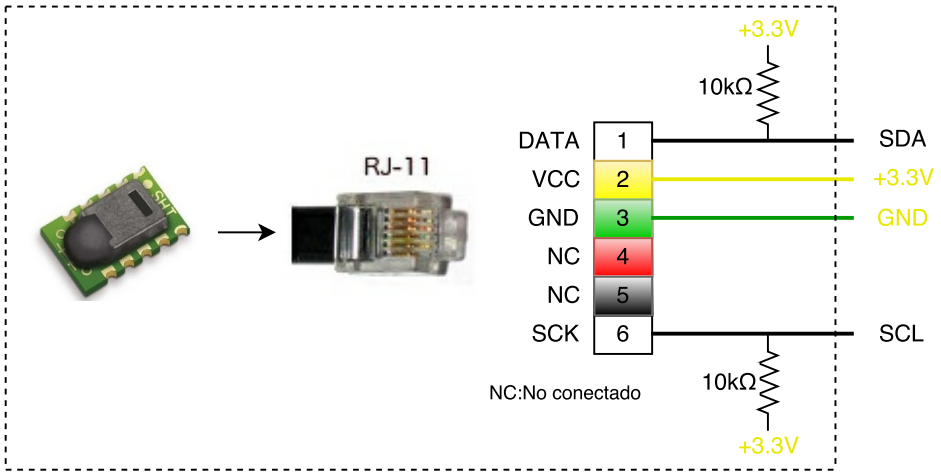


Figura 4.20: Conector de temperatura y humedad en el prototipo.

4.3. Adquisición de datos

La adquisición de las señales analógicas se realiza mediante un ADC ADS1115, mientras que para las señales digitales se utiliza el PIC 18F26K20. Si bien el PIC cuenta con un módulo ADC, se utiliza uno externo, debido a que las señales de radiación necesitan una resolución mayor a la del ADC del PIC. Esta situación se justifica en la Sección 4.3.1.

El PIC seleccionado posee varias entradas para interrupciones externas que son utilizadas para la detección de pulsos de los sensores de lluvia y de velocidad de viento. También cuenta con varios módulos para comunicaciones seriales como I2C, SPI y USART. Cada uno de estos módulos es empleado para comunicarse con dispositivos que intervienen en el diseño del sistema. Otra característica importante del microcontrolador es la gran cantidad de memoria RAM y ROM que posee respecto a otros de la misma gama, las cuales son necesarias para soportar la implementación de un RTOS que se detalla en la Sección 4.4. En el Anexo C se muestra las principales características del microcontrolador.

El ADS1115 es un ADC delta-sigma ($\Delta\Sigma$) con una resolución de 16 bits a 860 muestras/segundo. El ADC se comunica mediante protocolo I2C y puede ser configurado como 4 canales de entrada para señales no diferenciales o 2 canales para señales diferenciales. Entre sus principales características están:

- Rango de alimentación: 2.0 a 5.5 V,
- Bajo consumo de corriente:
  Modo Continuo: 150 μA,
  Modo Single-Shot: Auto Shut-Down.

- Tasa de datos programable: 8 a 860 muestras/segundo,
- Estabilización de un ciclo,
- Referencia de voltaje interna de baja desviación,
- Oscilador interno,
- Interfaz I2C con dirección seleccionable,
- 4 canales para entradas no diferencial o 2 para diferenciales,
- Comparador programable.

El esquema de conexión típico del ADC con un microcontrolador se muestra en la Figura 4.21. Para convertir las lecturas del ADC al valor de tensión medido, se utiliza la Ecuación (4.3).

$$V = cnt * \frac{FS}{(2^{15} - 1)} \tag{4.3}$$

donde:

V: tensión en voltios (V)
cnt: lectura digital del ADC
FS: Rango de escala completa dado por la configuración del amplificador de ganancia (PGA).

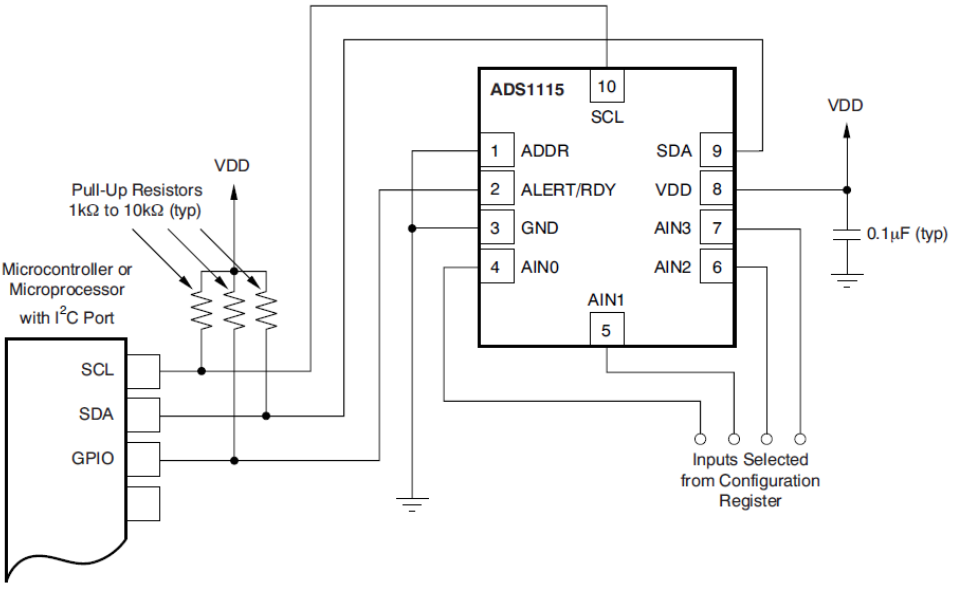


Figura 4.21: Conexión típica para el ADS1115, fuente: [6].

4.3.1. Radiación solar

La salida analógica del sensor de radiación solar es una señal no diferencial que varía entre 0 V y 3 V. De acuerdo a la hoja de datos del sensor [44], la señal de salida varía 1.67 mV por W/m^2. Por lo tanto, la resolución del ADC debe ser capaz de detectar estas variaciones para cumplir con las recomendaciones realizadas por la OMM [45], la cual sugiere que el menor cambio detectable debe ser 1 W/m^2.

La resolución de un ADC, la cual depende del voltaje de referencia (V_{ref}) y el número de bits (n), se obtiene mediante la Ecuación (4.4).

$$Resolución = V_{ref}/2^n \tag{4.4}$$

Si se despeja de la Ecuación (4.4) el numero de bits (n) que debe tener un ADC para detectar variaciones de 1,67 mV se tiene que:

$$\begin{aligned} n &= \frac{ln(V_{ref}) - ln(Resolución)}{ln(2)} \\ &= \frac{ln(3{,}3V) - ln(1{,}67mV)}{ln(2)} \\ &= 10{,}948 \end{aligned}$$

Puesto que el número de bits debe ser entero, si se utiliza 11 bits se tiene una resolución de 1,61 mV, mientras que con 10 bits se tiene una resolución de 3,22 mV. Dado que en ambos casos la resolución no es exactamente 1,67 mV, el ADC no puede detectar los cambios de 1 W/m^2. Por lo tanto se recalcula el número de bits para lograr una resolución más fina. Si se impone una resolución de 0,01 mV, de tal manera que el ADC pueda detectar las variaciones de 1,67 mV, se tiene que el número de bits necesario es:

$$\begin{aligned} n &= \frac{ln(V_{ref}) - ln(Resolución)}{ln(2)} \\ &= \frac{ln(3{,}3V) - ln(0{,}01mV)}{ln(2)} \\ &= 18{,}33 \\ &\approx 18 \end{aligned}$$

Con 18 bits se obtiene una resolución de 0,013 mV y con 19 bits de 0,006 mV. Por lo tanto, el ADC debe contar con 18 bits de resolución para detectar cambios exactos más cercanos a 1.67 mV. Entonces, tanto el almacenamiento como la transmisión de los datos muestreados se lo realiza en bytes, es decir que para un ADC de 18 bits se utilizaría 3 bytes (24 bits), en donde los últimos 6 bits serían de relleno, y por lo tanto desperdiciados. Para evitar esta situación y con el objetivo de optimizar la información a almacenar y transmitir, se utilizaron los 16 bits del ADC para ocupar 2 bytes. Para evitar sobredimensionar el número de bits de resolución, se usó un ADC de 16 bits. Entre los ADCs disponibles esta el ADS1115 de 16 bits con la que se obtiene una resolución de 0,05 mV, con la cual se puede controlar variaciones de 1,65 mV.

Para filtrar el ruido blanco presente en la señal, se realiza un promedio de las muestras obtenidas por el ADC. En la Figura 4.22 se realiza una comparación de la señal analógicas de radiación solar con el filtro RC, luego realizando un promedio de 4, 16 y 64 muestras. Mientras mayor es el numero de muestras, se filtra mejor el ruido.

No obstante, la adquisición de un mayor numero de muestras se ve reflejado en un mayor consumo de recursos para procesar los datos, de tal manera que para filtrar el ruido en la señal de radiación solar sin consumir una gran cantidad de recursos en el microcontrolador, en cada adquisición se registra el promedio de 16 muestras.

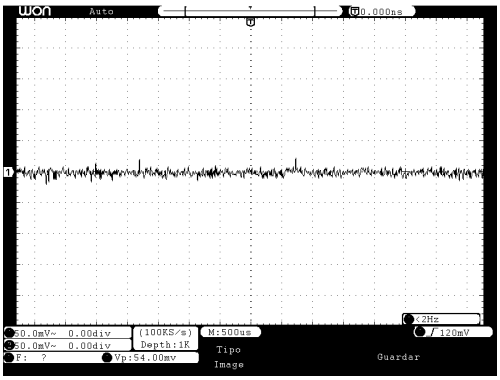
4.3.2. Radiación ultravioleta

La señal analógica del sensor UV es no diferencial y varia de 0 a 2,5 V. De acuerdo a la hoja de datos [28] el sensor presenta variaciones de 150 mV por índice UV o 364 mV por MED/hora [2]. Por lo tanto, si se establece la resolución del ADC igual a 1 mV para detectar estas variaciones, el número de bits del ADC debe ser:

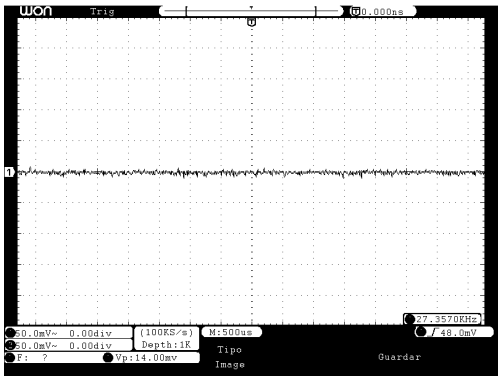
$$\begin{aligned} n &\geq \frac{ln(V_{ref}) - ln(Resolución)}{ln(2)} \\ &\geq \frac{ln(3{,}3V) - ln(1mV)}{ln(2)} \\ &\geq 11{,}68 \\ &= 12 \end{aligned}$$

Así, el número de bits necesario para detectar las variaciones del sensor es menor al número de bits que posee el ADC ADS1115. Por lo que la adquisición de esta señal es adecuada con

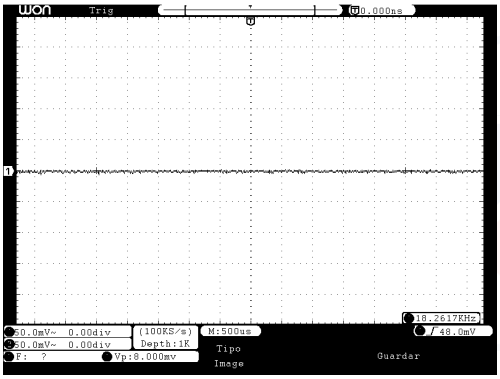
[2] MED es el tiempo mínimo necesario de radiación para producir una quemadura (eritema) en la piel [46].



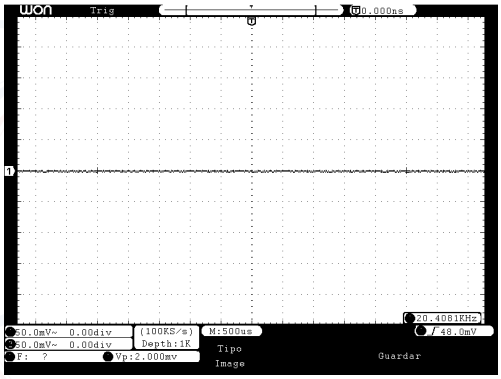
(a) Señal con filtro RC.



(b) Promedio con 4 muestras.



(c) Promedio con 16 muestras.



(d) Promedio con 64 muestras.

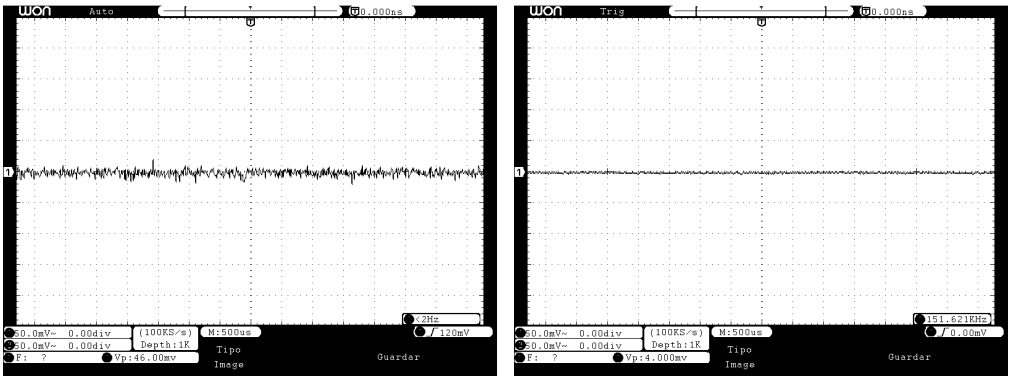
Figura 4.22: Filtrado de señal de radiación solar mediante filtro RC y promediado de muestras.

este ADC, utilizando 16 bits.

De igual forma al proceso realizado en la señal de radiación solar para filtrar el ruido blanco, se adquiere varias muestras de la señal de radiación UV para promediarlas. En la Figura 4.23 se muestra la señal de radiación UV con el filtro RC (Figura 4.23a) y luego realizando promedios con 16 muestras (Figura 4.23b.

4.3.3. Dirección de viento

Para establecer la dirección de viento se lee la diferencia de potencial analógica de una resistencia variable de 20 kΩ. Para determinar si el potenciómetro es lineal o logarítmico se realizaron mediciones de resistencia y potencial. La Figura 4.24 muestra los resultados de las mediciones y se puede concluir que el potenciómetro es lineal, i.e. la relación de grados con potencial es directamente proporcional. El valor de un grado se calcula mediante la Ecuación



(a) Señal con filtro RC. (b) Promedio con 16 muestras.

Figura 4.23: Filtrado de señal de radiación UV mediante filtro RC y promediado de muestras.

(4.5).

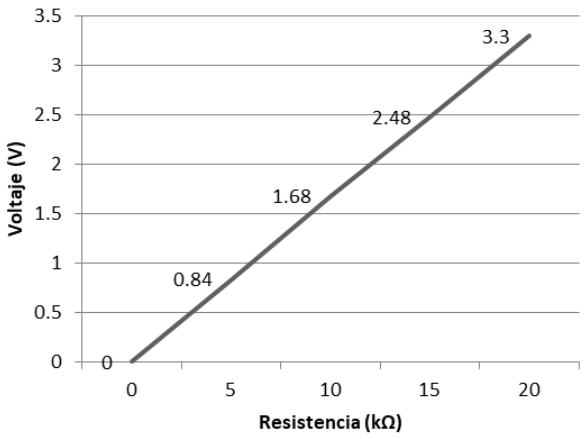
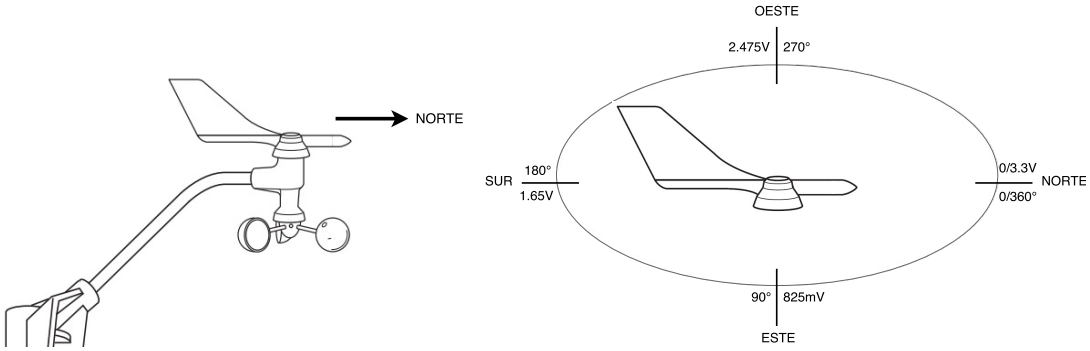


Figura 4.24: Variación de resistencia y potencial en potenciómetro

$$\frac{V_{total}}{360\,°} = \frac{3{,}3\,V}{360\,°} = 9{,}16\,mV/° \tag{4.5}$$

De la Ecuación (4.5) se obtiene que 9.16 mV equivale a 1°; esta señal se adquiere mediante el ADC ADS1115, el cual permite detectar variaciones de 0,05 mV. Por defecto, como referencia se recomienda que el brazo de soporte del anemómetro apunten hacia el norte para obtener una lectura correcta de la dirección de viento (Figura 4.25a).

Los esquemas de conexión de todas las señales analógicas con el ADC y la lectura de la tensión de una batería mediante un divisor de tensión se muestran en la Figura 4.26.



(a) Orientación de cero grados por defecto

(b) Relación de grados con voltaje

Figura 4.25: Orientación de la veleta con respecto al brazo en la estación Davis 6162 Wireless Vantage Pro2 Plus

4.3.4. Pluviómetro

Para detectar los pulsos generados por el balancín del pluviómetro se utiliza el módulo de interrupción externa `INT0` del PIC. Para hacer uso de este módulo, la señal del pluviómetro se conecta al PIN `RB0` del PIC. Esta interrupción es activada al detectar flancos de subida o bajada, dependiendo de cómo se configure el bit `INTEDG0` del registro `INTCON2`.

De acuerdo a las conexiones realizadas (Sección 4.2.2.1), la señal a la salida del interruptor normalmente está en bajo y al haber un cambio de posición en el balancín, se produce un flanco de subida y bajada (pulso) en la señal. Debido a que cada pulso indica un cambio en el balancín, el módulo de interrupción está configurado para detectar los flancos de subida. Se utiliza una variable que se incrementa en uno al producirse una interrupción. Esta variable se reinicia en cada muestreo.

4.3.5. Velocidad de viento

La señal de salida de este sensor es similar a la del pluviómetro. Sin embargo, para establecer la velocidad del viento se calcula la frecuencia de los pulsos. Para ello se usa el módulo `CCP1` del microcontrolador. Este módulo *Captura/Comparación/PWM* (CCP) en modo captura permite acceder al estado actual de un temporizador (`Timer 1`).

Para calcular la frecuencia de la señal se usa el módulo `CCP1` en modo *captura*, el cual adquiere la señal a procesar mediante el pin `RC2`. Este módulo genera una interrupción y guarda el registro del temporizador (`Timer1`) cada vez que detecta un flanco ascendente, configurado mediante el registro de control (`CCP1CON`). De esta manera se calcula la diferencia de tiempo

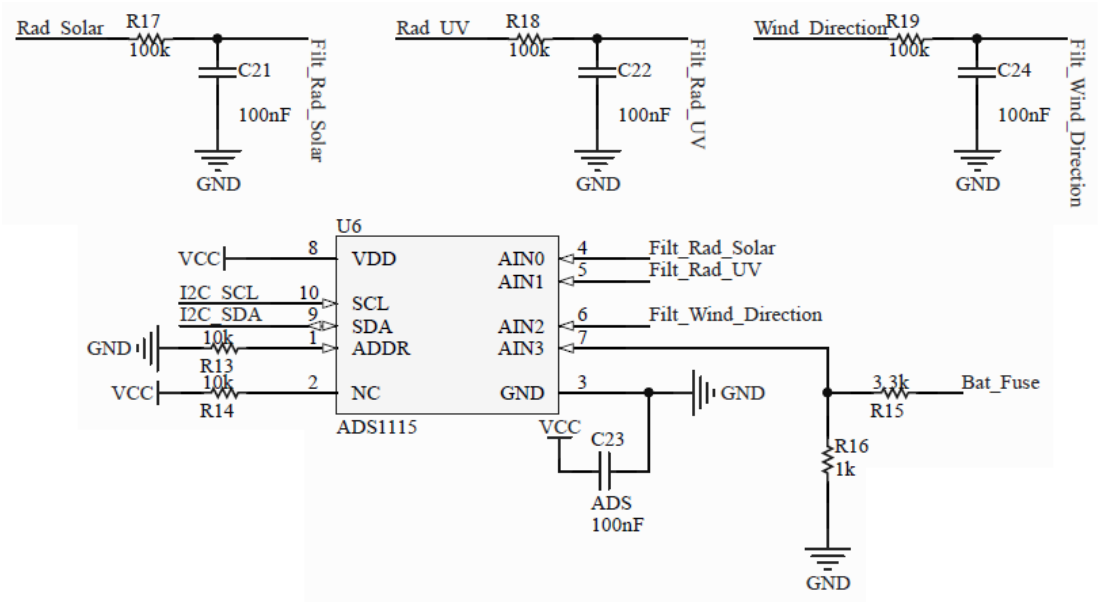


Figura 4.26: Esquema de conexión de señales analógicas con el ADC ADS1115.

entre flancos ascendentes, lo cual corresponde al periodo de la señal (Figura 4.27).

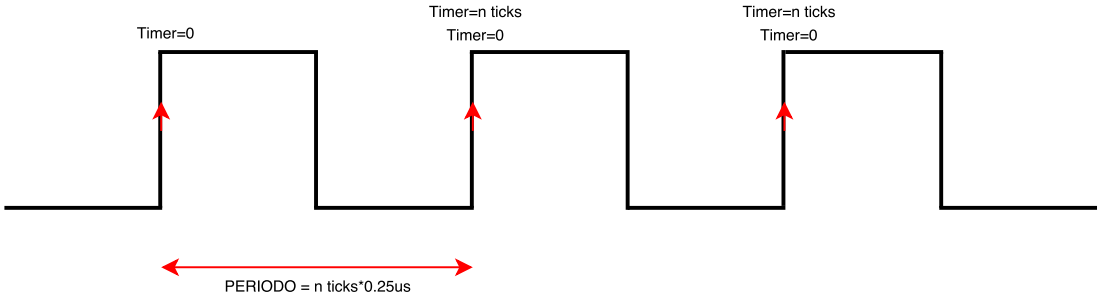


Figura 4.27: Cálculo del periodo de la señal digital.

El temporizador `Timer1` es un contador de 16 bits (0 a 65535) y se incrementa a una frecuencia determinada por el pre-escalador (prescaler)[3]. Para determinar el tiempo que equivale a cada incremento (tick) del temporizador, se utiliza la Ecuación 4.6.

$$t_{tick} = \frac{4 \cdot prescaler}{f_{clk}} \tag{4.6}$$

donde f_{clk} es la frecuencia de trabajo del PIC.

[3]Pre-escalador: escalador para Timer1 que permite dividir la frecuencia de entrada del reloj (f_{clk}) por 1, 2, 4 u 8 [47].

El microcontrolador está configurado para usar el oscilador interno a una frecuencia de 16 MHz, misma que no se escala para el temporizador. Entonces reemplazando los valores correspondientes se encuentra que cada incremento del *timer* es igual a

$$\begin{aligned} t_{tick} &= \frac{4 \cdot 1}{16\,MHz} \\ &= 0{,}25\,\mu s \end{aligned}$$

Por lo tanto, el *timer* funciona a una frecuencia de $1/0{,}25\,\mu s = 4\,Mhz$, y puede medir un rango de tiempo de $0{,}25\,\mu s \leq t \leq 65535 \times 0{,}25\,\mu s = 16{,}38375\,ms$. Sin embargo, el periodo de la señal digital a medir puede ser mayor al tiempo que puede medir el *timer* antes de desbordarse (reiniciarse). Para poder medir mayores periodos de tiempo se realiza un registro del numero de veces en que el temporizador se reinicia mediante una variable de 16 bits, lo cual permite aumentar el rango de medición a $65535 \times 16{,}38375\,ms \approx 1073{,}72s = 17,9\,min$, periodo más difícil de superar por la señal digital de velocidad de viento. Finalmente la frecuencia de la señal es obtenida mediante el valor inverso del periodo.

4.3.6. Temperatura y humedad

El microcontrolador se comunica con el sensor de temperatura y humedad través de un bus serie síncrono capaz de acoplarse al bus de datos del protocolo I2C sin interferir con el mismo, pero que usa su propio protocolo. Aunque el microcontrolador posee un módulo específico para comunicaciones seriales, este puede funcionar en modo I2C o SPI pero no los dos a la vez. Puesto que se hace uso del protocolo SPI para comunicar el microcontrolador con una memoria externa (Sección 4.7), se uso una librería del compilador MikroC que implementa el protocolo I2C por software.

El microcontrolador usa el protocolo I2C para comunicarse con el ADC y el RTC (DS1307) y conmuta entre los protocolos para comunicarse con cada dispositivo según corresponda. En la Figura 4.28 se muestra el esquema de conexión del sensor acoplado con el resto de dispositivos que utilizan el bus I2C.

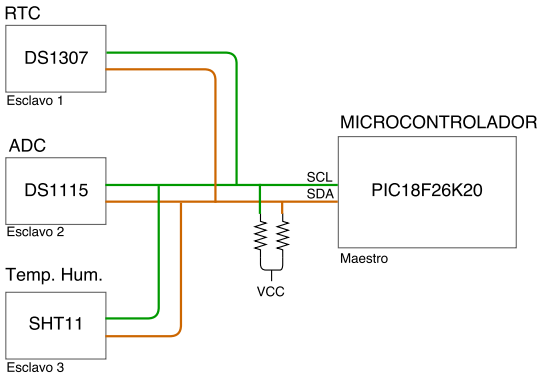


Figura 4.28: Conexión de dispositivos en bus I2C. El RTC y el ADC usan el protocolo I2C y el sensor de temperatura usa su propio protocolo.

4.4. Sistema Operativo de Tiempo Real

Debido a la cantidad de tareas que debe manejar el microcontrolador y la necesidad de que se ejecuten de forma independiente, se utiliza un RTOS. Dentro de los RTOSs que se investigaron (Anexo B), se eligió el sistema operativo OSA, por ser gratuito y distribuido con licencia BSD [4] [48], lo que permite aprovechar todas las capacidades del sistema operativo sin ningún tipo de restricción. OSA está diseñado para dispositivos con poca memoria RAM, soporta varios compiladores en lenguaje C y cuenta con una variedad de librerías que ahorran esfuerzo y tiempo. En cuanto al compilador, se utilizó MikroC Pro for PIC.

Las funciones principales del microcontrolador son adquirir los datos de todas las variables meteorológicas, agregar la referencia de tiempo mediante el RTC sincronizado con el GPS, almacenar los datos en una memoria externa SD como respaldo y transmitirlos de forma inalámbrica mediante un módulo XBee. Varias de estas funciones necesitan ejecutarse en diferentes tiempos, como es el caso de la adquisición de datos que dependen del tiempo de muestreo de cada sensor.

4.4.1. Creación de tareas

Un RTOS es un programa que gestiona recursos y planifica la ejecución de varias tareas simultáneas, creando así un sistema multitarea. Cada hilo se ejecuta independientemente. Por lo tanto se han creado hilos para ejecutar las diferentes funciones mencionadas en el párrafo anterior. En el anexo E se muestra la configuración del RTOS OSA. En resumen, el sistema consta de los siguientes hilos de ejecución:

[4] Se puede modificar y utilizar libremente

- **HiloSincGPS**: Sincronización de RTC con GPS.
- **HiloTransmitir**: Almacenamiento de datos en memoria SD y transmisión de los mismos por XBee.
- **HiloPluviometro**: Adquisición y procesamiento del sensor de precipitación.
- **HiloTempHumExt**: Adquisición y procesamiento del sensor de temperatura y humedad relativa.
- **HiloVelocViento**: Adquisición y procesamiento del sensor de velocidad de viento.
- **HiloRadiacion**: Adquisición y procesamiento del sensor de radiación solar y UV.
- **HiloDirecViento**: Adquisición y procesamiento del sensor de dirección de viento.

Al iniciar el sistema, antes de crear hilos, se lee un archivo de texto desde la memoria SD que tiene la configuración de cada sensor y la dirección destino que se va a utilizar en el módulo XBee para la transmisión. En este archivo se establece qué sensores se activan para adquirir datos, y de acuerdo a esta configuración el sistema crea las tareas para cada sensor, ya que, si la estación no cuenta con todos los sensores, no es necesario ejecutar todos los hilos. En caso de no encontrarse este archivo, el sistema carga una configuración por defecto (Tabla 4.4). El diagrama de flujo del funcionamiento del sistema se muestra en la Figura 4.29.

4.4.2. Asignación de prioridades a tareas

OSA usa el algoritmo de planificación de tareas *Co-operative Scheduling* [48], lo cual implica que si el programa se queda estancado en un bucle dentro de una tarea el planificador no puede forzar un cambio de contexto[5] para permitir la ejecución del resto de tareas. Para forzar un cambio de contexto, OSA dispone de las funciones `OS_Yield()` y `OS_Delay()`. Esta ultima genera un tiempo de retardo que es aprovechado atendiendo otras tareas.

OSA dispone de 8 niveles de prioridad, siendo 0 la prioridad más alta y 7 la prioridad más baja. El planificador examina todas las tareas activas y busca aquellas que están listas para ejecutarse. Luego el planificador cede el control a la tarea con la prioridad mas alta. No obstante, si una tarea que tiene el control no termina su ejecución y las funciones para forzar un cambio de contexto no han sido implementadas en la tarea, el planificador omite la ejecución del resto de tareas hasta completar la ejecución de la tarea actual [49]. De esta manera, las prioridades han sido asignadas tomando en cuenta el tiempo que necesita cada una para adquirir la muestra del sensor.

La tarea `HiloSincGPS` ha sido asignada con la prioridad más alta, ya que es la tarea que más tiempo necesita para sincronizar el RTC con el GPS. Después, sigue la tarea `HiloTransmitir`, la cual se encarga de almacenar los búferes con los datos registrados de cada sensor en la

[5] Al pausar una tarea para atender otra, el sistema operativo ejecutará ésta tarea de nuevo más tarde.

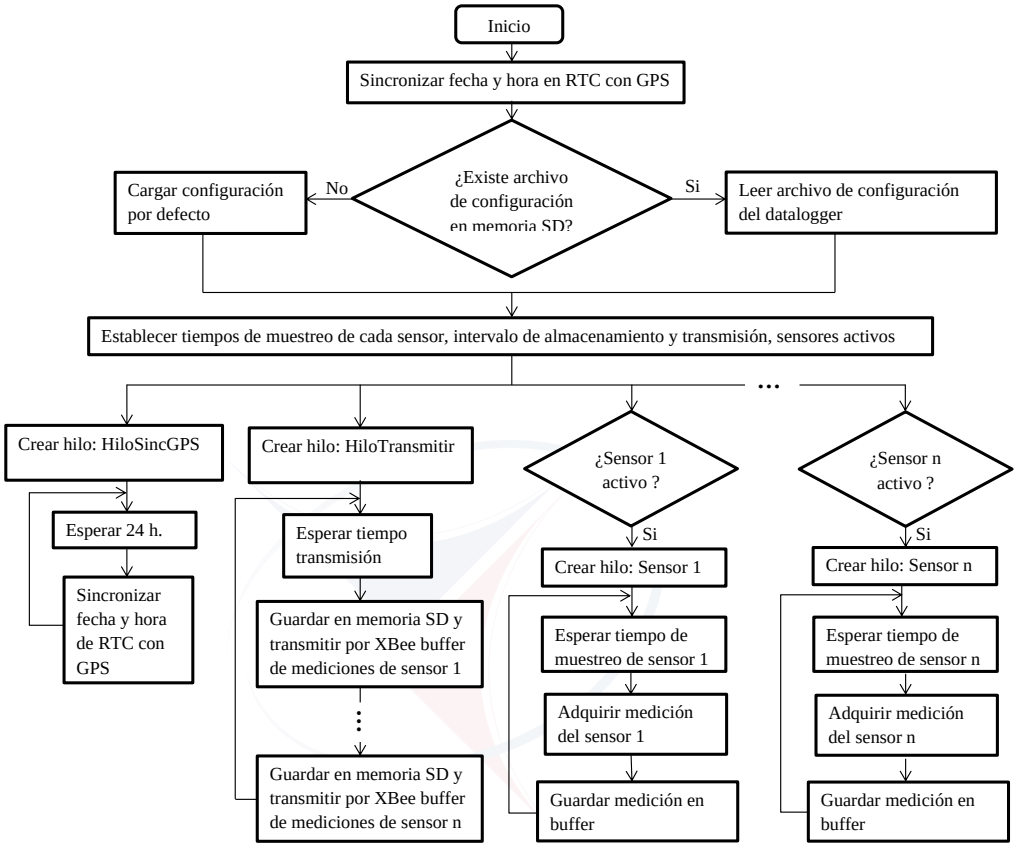


Figura 4.29: Diagrama de flujo del funcionamiento del prototipo.

memoria SD y transmitir por el puerto serial al módulo XBee. Luego, las tareas que siguen en prioridad son las que utilizan el ADC, ya que este necesita, además de ser configurado en cada adquisición, un intervalo de tiempo para completar la conversión. Aunque existen dos tareas que hacen uso del ADC, la tarea `HiloRadiacion` tiene mayor prioridad con respecto a `HiloDirecViento`, debido a que utiliza dos canales del ADC, para la señal de radiación solar y UV, y para cada señal procesa 16 muestras para promediar y filtrar ruido blanco. Mientras que `HiloDirecViento` ocupa sólo un canal para la señal de dirección de viento y no realiza promedios de la señal. La tarea que sigue en prioridad es `HiloTempHumExt`, la cual necesita comunicarse con el sensor de temperatura y humedad para obtener las lecturas de las variables correspondientes. Las tareas con menor prioridad son `HiloPluviometro` e `HiloVelocViento` debido a que estas actualizan sus lecturas mediante interrupciones. La Tabla 4.1 muestra la asignación de prioridades de cada tarea.

Tarea	Prioridad
HiloSincGPS	0
HiloTransmitir	1
HiloRadiacion	2
HiloDirecViento	3
HiloTempHumExt	4
HiloPluviometro	5
HiloVelocViento	6

Tabla 4.1: Prioridades de tareas

4.4.3. Temporizador

Todo RTOS requiere de una referencia de reloj para generar retardos y programar llamadas a tareas. La frecuencia de esta señal de reloj depende de la aplicación y la resolución deseada, una mayor frecuencia significa una mayor carga de procesamiento para el sistema. Para generar esta señal se configuro el Timer 0 para generar interrupciones cada milisegundo, lo cual significa que el RTOS no será capaz de medir tiempos o generar retardos menores a 1 ms.

Para determinar el intervalo de conteo del temporizador, se configura el tiempo de duración de cada incremento del contador mediante la Ecuación 4.6. La frecuencia de trabajo del microcontrolador es $f_{clk} = 16\,MHz$, y si se establece el preescalador a $prescaler = 16$ cada incremento del módulo Timer 0 es igual a:

$$\begin{aligned} t_{tick} &= \frac{4 \cdot 16}{16\,MHz} \\ &= 4\,\mu s \end{aligned}$$

Por lo tanto, el temporizador ha sido configurado para realizar $1\,ms/4\,\mu s = 250$ conteos para generar una interrupción cada milisegundo.

La frecuencia de ejecución de cada hilo está determinada por los intervalos de tiempo establecidos en el archivo de configuración, excepto para el hilo de sincronización del RTC con el GPS con un tiempo fijo de 24 horas. Para controlar la exactitud de estos intervalos de tiempo, cada hilo depende de una bandera controlada por interrupciones de un temporizador (`Timer 0`).

El procedimiento que emplean los hilos para la adquisición y procesamiento de las señales de los respectivos sensores se detalló en la Sección 4.3. El procedimiento para sincronizar el RTC con el GPS se presenta en la Sección 4.6 y en las Secciones 4.7 y 4.8 el proceso de almacenamiento y transmisión de la muestras de cada sensor.

4.5. Configuración del prototipo

4.5.1. Modos de funcionamiento

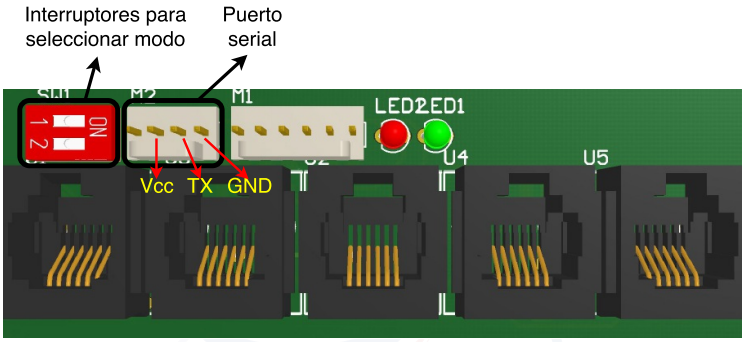


Figura 4.30: Tapa frontal de dispositivo. Los interruptores que permiten seleccionar los modos de funcionamiento se encuentran en la parte superior izquierda. Junto a los interruptores se encuentra el puerto serie para leer los mensajes del dispositivo en modo depurador.

El prototipo tiene 4 modos de funcionamiento. Estos modos se configuran mediante dos interruptores en la tapa frontal del dispositivo (Figura 4.30). Las combinaciones de los interruptores y el modo al que corresponden se muestran en la Tabla 4.2. A continuación se describe cada modo:

- **Transmisión con XBee:**
  El dispositivo transmite los datos adquiridos de los sensores a través del módulo XBee de acuerdo al intervalo de tiempo establecido en el archivo de configuración.
- **Transmisión con XBee y depurador:**
  Este modo cumple la misma función del modo anterior, solo que además permite usar el depurador y conocer el funcionamiento del dispositivo (Sección 4.9.).
- **Transmisión por puerto serial:**
  En este modo no se utiliza el módulo XBee para la transmisión de los datos, sino más bien la información se transmite por puerto serial, en caso de que el usuario desee observar directamente los datos en un monitor serial.
- **Transmisión con XBee en tiempo real:**
  Los datos son transmitidos de forma individual justo después de cada muestreo.

Interruptor 1	Interruptor 2	N° modo	Modo de funcionamiento
OFF	OFF	1	Transmisión con XBee
OFF	ON	2	Transmisión con XBee y depurador
ON	OFF	3	Transmisión por puerto serial
ON	ON	4	Transmisión con XBee en tiempo real

Tabla 4.2: Combinación de interruptores y modo de funcionamiento al que corresponden.

4.5.2. Archivo de configuración

El prototipo permite configurar el identificador de la estación en una red, la dirección del módulo XBee receptor, los periodos de muestreo de cada sensor y el periodo de transmisión de datos mediante un archivo de texto en el directorio raíz de la memoria SD.

Para ser reconocido, el archivo de configuración debe llamarse `CONFIG.TXT`. En este archivo se debe escribir los valores de los parámetros en el orden estricto que se muestra en la Tabla 4.3. Cada parámetro debe ser separado por un punto y coma (;) y en el caso de parámetros que tienen varios valores, éstos deben ser separados por una coma (,). Por ejemplo, la dirección del módulo XBee receptor (8 bytes para la dirección 1 y 2 bytes para la dirección 2) y los sensores (un valor para habilitar el sensor y un valor para establecer el tiempo de muestreo). Todos los valores deben estar en formato decimal.

No.	Parámetro	Descripción
1	Id. estación	Identificador de la estación en la transmisión de datos
2	Dir. 1 XBee RX	Dirección de 8 bytes del módulo Xbee receptor
3	Dir. 2 XBee RX	Dirección de 2 bytes del módulo XBee receptor
4	Pluviómetro	Habilitar sensor (1) o deshabilitar sensor (0), periodo de muestreo
5	Temp. y humedad	
6	Veloc. de viento	
7	Dirección de viento	
8	Radiación solar y UV	
9	Transmisión	Periodo de almacenamiento y transmisión de datos

Tabla 4.3: Parámetros de configuración en archivo de texto. El número indica el orden.

El archivo de configuración es opcional ya que el dispositivo cuenta con una configuración por defecto, sin embargo, al utilizar el archivo de configuración, éste debe contener siempre todos los parámetros con sus valores respectivos y en el orden correcto para evitar un funcionamiento inesperado. En la Tabla 4.4 se muestran, como ejemplo, los valores por defecto que utiliza el dispositivo y en la Figura 4.31 se muestra su implementación en el archivo de texto.

Al encender el dispositivo, éste lee el archivo de configuración desde la memoria SD. Todos los caracteres en el archivo de texto están en código ASCII y los valores numéricos en formato

Parámetro	Descripción de valor	Valor por defecto
Identificador estación	Número entre 0 y 4095	1
Dirección 1 XBee RX	Dirección destino de 8 bytes	0,0,0,0,0,0,0,0
Dirección 2 XBee RX	Dirección destino de 2 bytes	255,254
Pluviómetro	Habilitar (1)/Deshabilitar (0)	1
	Periodo de muestreo en segundos	20
Temp. hum. rel.	Habilitar (1)/Deshabilitar (0)	1
	Periodo de muestreo en segundos	10
Veloc. de viento	Habilitar (1)/Deshabilitar (0)	1
	Periodo de muestreo en segundos	10
Direc. de viento	Habilitar (1)/Deshabilitar (0)	1
	Periodo de muestreo en segundos	10
Radiación solar y UV	Habilitar (1)/Deshabilitar (0)	1
	Periodo de muestreo en segundos	50
Transmisión de datos	Periodo en minutos	5
Código de configuración final para archivo de texto: 1;0,0,0,0,0,0,0,0;255,254;1,20;1,10;1,10;1,10;1,50;5		

Tabla 4.4: Configuración por defecto de los parámetros de funcionamiento del prototipo.

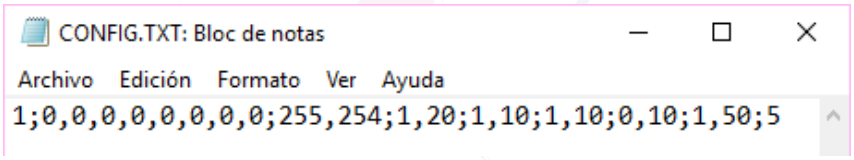


Figura 4.31: Archivo de texto con configuración por defecto de los parámetros de funcionamiento del prototipo.

decimal. Para la transmisión de datos se requiere conocer la estación de donde provienen, de ahí la necesidad de establecer el identificador de la estación en el archivo de configuración. Este identificador es de 12 bits, y se lo debe escribir en formato decimal al igual que el resto de valores. El XBee está programado para funcionar en modo API, por lo que, para enviar los datos, es necesario generar una trama que incluya la dirección destino del XBee receptor. Existen dos tipos de dirección en ZigBee, una de 64 bits (802.15.4) y otra de 16 bits (*network layer*). La dirección de 64 bits es comparable a la dirección MAC en Ethernet o redes Wi-Fi, está definida por el fabricante y es única para cada dispositivo. La dirección de 16 bits es similar a una dirección IP, es una dirección temporal para identificar un dispositivo en una red, es asignada por un coordinador y es única para la red actual. Si no se tiene esta dirección de 16 bits, aún se puede direccionar al dispositivo mediante la dirección de 64 bits si está asociado a la red [50].

Luego se tiene la configuración de los sensores de la estación. Para cada sensor se definen dos valores. El primero indica si se desea activar la adquisición de datos del sensor, dependiendo

si es 1 o 0. Puesto que para cada sensor existe un hilo específico, el RTOS determina los hilos a ser ejecutados. El segundo valor corresponde al tiempo de muestreo del sensor.

Si no se detecta una memoria SD o si no se encuentra un archivo de configuración, el RTOS carga una configuración por defecto (Tabla 4.4).

4.6. Referenciación de tiempo

La adquisición de datos meteorológicos se realiza con una referencia de fecha y hora. Para obtener el tiempo se utiliza un RTC, en este caso específico, el integrado DS1307, el cual provee información de segundos, minutos, horas, días, meses y años. El RTC depende de un oscilador de cristal externo de 32.768kHz. El microcontrolador (maestro) se comunica con el RTC (esclavo) a través del protocolo I2C. Para acceder a los datos del RTC se genera una condición inicio, seguido por el código identificador del dispositivo en el bus I2C, y la dirección del registro al que se quiere acceder.

El RTC transmite los datos (fecha y hora) en formato BCD y el microcontrolador los transforma a formato *epoch*[6] (*UNIX timestamp*). De esta manera cada vez que se toma una muestra de una variable meteorológica se registra también la fecha y la hora de la adquisición.

El reloj necesita ser igualado con la fecha y hora cada vez que inicia el sistema. Para la sincronización de tiempo con el RTC se utiliza un GPS. Por lo tanto, al iniciar el sistema, después de leer el archivo de configuración, se ejecuta un método que espera hasta que el módulo GPS obtenga el primer posicionamiento para poder sincronizar la hora y fecha con el RTC, y finalmente se ejecutan todos los hilos del sistema.

Se utiliza la tarjeta GPS Adafruit Ultimate, la cual usa el *chipset* MTK3339, un módulo GPS de alta calidad que puede seguir hasta 22 satélites en 66 canales, y con un receptor de alta sensibilidad (-165 dB tracking) [51]. Mediante la recepción de señales emitidas por varios satélites, el GPS puede determinar su posición actual (estado FIX), tiempo o velocidad. El GPS se comunica con el microcontrolador a través del puerto serial, y para la transmisión de datos usa el protocolo NMEA.

El protocolo NMEA usa caracteres en formato ASCII, en donde cada mensaje inicia con el signo $ y termina con los caracteres retorno de carro (CR- ASCII 13) y nueva línea (LF- ASCII 10). El significado de cada mensaje depende de la primera palabra. Así por ejemplo, el mensaje que inicia con $GPGLL provee información sobre latitud, longitud, hora exacta, validez de dato y un *checksum* para comprobar que los datos han sido recibidos correctamente. Cada campo

[6] El formato *epoch* codifica información de fecha y hora en 4 bytes

está separado por una coma (,), como se muestra en la Figura 4.32. Cada segundo el GPS envía un grupo de mensajes NMEA al microcontrolador. Los principales mensajes del estándar NMEA se muestran en la Tabla 4.5.

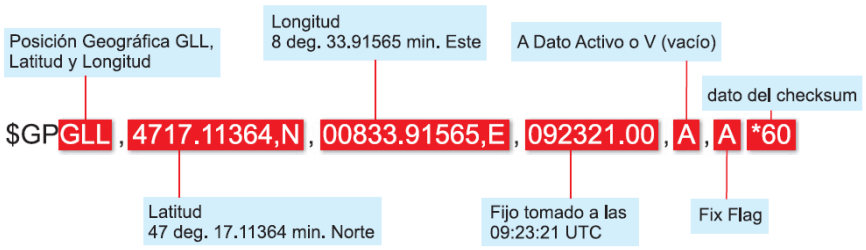


Figura 4.32: Cadena NMEA GPGLL (posición geográfica), datos de latitud y longitud del GPS, fuente: [7].

Registro NMEA	**Descripción**
GPGGA	Datos de localización
GPGLL	Datos de latitud y longitud
GPGSA	Datos de los satélites en general
GPGSV	Datos de un satélite detallado
GPRMC	Datos mínimos de posicionamiento, tiempo y velocidad
GPVTG	Datos de rastreo (*"track"*) y velocidad de movimiento

Tabla 4.5: Mensajes estándar NMEA.

Cuando el GPS no puede determinar los datos de posicionamiento, genera mensajes incompletos o con datos erróneos. Para determinar que el GPS tiene los datos de posicionamiento correctos, el microcontrolador analiza el mensaje GPGGA que tiene, entre otros campos, un indicador de posicionamiento. Si este indicador (*FIX*) es igual a 1 significa que el GPS ha podido establecer los datos de posicionamiento. Luego, para obtener la hora y fecha, se analiza el mensaje GPRMC, ya que GPGGA, aunque tiene un campo de hora, no provee información de la fecha. El formato del mensaje GPRMC se muestra a continuación:

$$\$GPRMC,\overbrace{HHMMSS}^{\text{Tiempo UTC en horas, minutos y segundos}}.SS,A,DDMM.MMM,N,DDDMM.MMM,W,Z.Z,$$
$$Y.Y,\underbrace{DDMMYY}_{\text{Fecha UTC de posicionamiento FIX en días, meses y años}},D.D,V,M,NS*CC<CR><LF \quad (4.7)$$

Cuando el RTC es sincronizado con la hora y la fecha del GPS, el RTOS lee el archivo de configuración en la memoria SD[7]. Finalmente crea y ejecuta los hilos del sistema de acuerdo a la configuración. Para mantener su precisión, el RTC es sincronizado con el GPS cada 24 horas a través del hilo `"HiloSincGPS"`. El hilo activa un PIN del microcontrolador para energizar el GPS momentáneamente hasta obtener la fecha y hora e igualar el RTC, luego lo apaga nuevamente. La ejecución del hilo se ejecuta paralelamente y no interfiere con el resto de tareas del RTOS.

4.7. Almacenamiento

Los datos obtenidos de los sensores son guardados en archivos en formato hexadecimal en una memoria SD. Los datos se guardan mediante *buffers* en la memoria RAM del microcontrolador periódicamente mediante el hilo `HiloTransmitir`. En la Tabla 4.6 se muestra el identificador y el nombre de archivo que corresponde a cada sensor.

Los archivos de todos los sensores se guardan en directorios diarios que llevan de nombre la fecha del día en que se ha realizado la adquisición de los datos. El formato de fecha que se utiliza para nombrar estos directorios es DDMMAA (día, mes y año). Estos directorios a su vez se guardan en uno con el nombre del identificador de la estación. La estructura de almacenamiento se muestra en la Figura 4.33.

El contenido de cada archivo inicia con el identificador del sensor, fecha y hora en formato *epoch* en la que inicia el registro, seguido de los datos de cada sensor; este formato se repite a lo largo del archivo para cada periodo de muestreo. A continuación se muestra este formato de registro:

[7]Si no existe el archivo, se utiliza la configuración por defecto

Sensor	**Identificador**	**Nombre archivo**
Pluviómetro	1	PRC
Temperatura	2	TMP
Humedad relativa	3	HMD
Velocidad de viento	4	ANM
Dirección de viento	5	DRV
Radiación Solar	6	SLR
Radiación UV	7	UV

Tabla 4.6: Identificador y nombre de archivo de datos de los sensores.

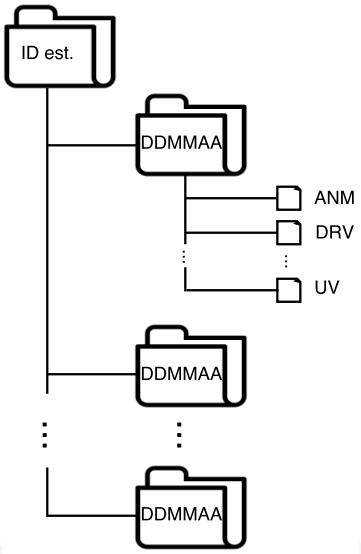


Figura 4.33: Estructura de directorios y archivos con la que se almacenan los datos en la memoria SD.

$$\underbrace{\text{ID sensor}}_{\text{1 byte}},\underbrace{\text{Fecha y hora (formato epoch)}}_{\text{4 bytes}},\text{muestra 1, muestra 2, ... , muestra n}$$

El tamaño en bytes de cada muestra depende del sensor (Tabla 4.7). Todos estos valores están en formato hexadecimal.

Sensor	N° de bytes
Pluviómetro	1
Temperatura	2
Humedad relativa	2
Velocidad de viento	1
Dirección de viento	1
Radiación solar	2
Radiación UV	2

Tabla 4.7: Número de bytes por muestra de cada sensor.

Se utiliza la memoria SD en formato FAT32, ya que éste permite direccionar mayor información (>2GB) que FAT16 y desperdicia menos espacio debido a *clusters* más pequeños. Para manipular este tipo de memoria se utiliza la librería FAT32 versión 2.5.0.0 para el compilador Mikroc For PIC, disponible en la comunidad de librerías de MikroElectronika[8]. Esta librería

[8] `https://libstock.mikroe.com/projects/view/108/fat32-library`

soporta memorias SD/MMC, operaciones de archivos y carpetas, y lectura/escritura de múltiples archivos a la vez. El esquema de conexión de la memoria SD con el microcontrolador se muestra en la Figura 4.34, y la distribución de los terminales de la memoria SD se detallan en el Anexo A.

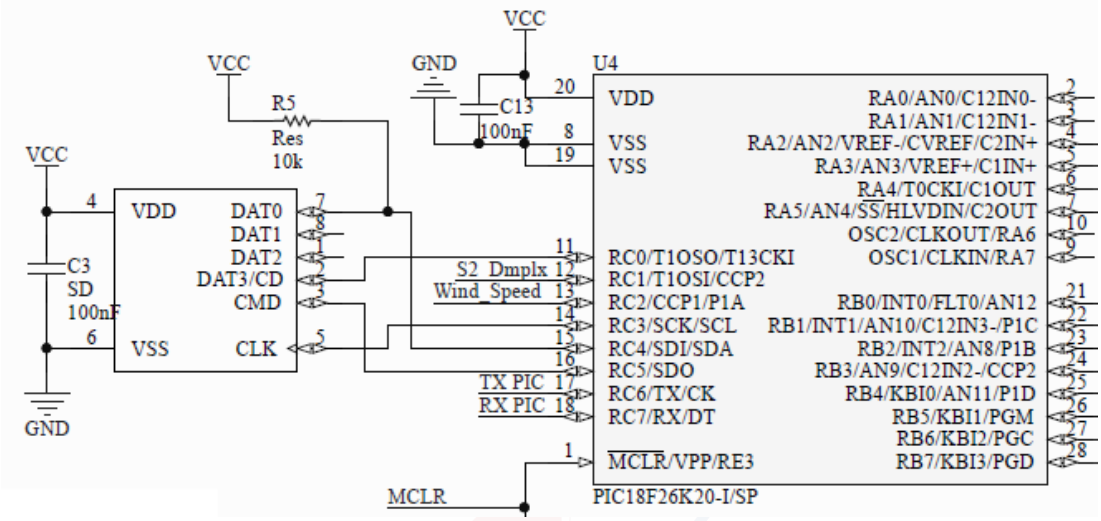


Figura 4.34: Esquema de conexión de memoria SD con el microcontrolador PIC 18F26K20.

4.8. Transmisión

La transmisión de los datos se realiza periódicamente mediante un hilo en el sistema (`HiloTransmitir`), el cual se ejecuta de acuerdo al tiempo de transmisión establecido en el archivo de configuración. Este hilo se encarga de leer los datos de cada sensor almacenados y con ellos generar una trama para transmitir los datos utilizando el estándar ZigBee.

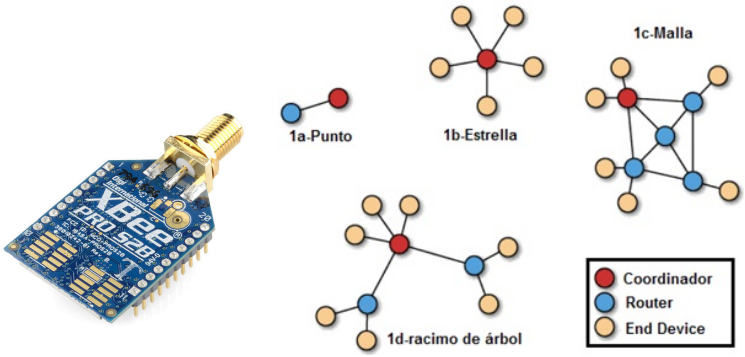
La transmisión inalámbrica se realiza mediante el módulo de radiofrecuencia XBee-PRO S2B ZB de Digi International Inc [9]. Los módulos XBee son utilizados para establecer comunicaciones inalámbricas a través del protocolo IEEE 802.15.4[10]. Estos dispositivos pueden operar dentro de diferentes topologías de red [52] (Figura 4.35b).

La principales características de este módulo son [53]:

- 3,3 V @ 295 mA.

[9] `https://www.digi.com/`

[10] Estándar que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con tasas bajas de transmisión de datos.



(a) XBee S2B (b) Topologías ZigBee: par, estrella, malla y de árboles de racimo, fuente:[52]

Figura 4.35: Topologías que XBee S2B soporta.

- Máxima velocidad de datos de 250 kbps.
- Salida 63mW (+17dBm).
- Rango de 1 milla (1600m).
- Conector RPSMA.
- Totalmente certificado por la FCC.
- 6 pines de entrada ADC 10-bit.
- 8 pines E/S digital.
- Encriptación 128-bit.
- Configuración local o inalámbrica.
- Set de comandos AT o API.
- Requiere antena externa.

El módulo XBee que se utiliza ha sido programado como *router* y modo API utilizando el software X-CTU[11]. El modo API permite, mediante la generación de tramas con datos predefinidos, armar redes con varios dispositivos remotos, identificando origen y destino de la información dentro de la misma trama. Las tramas están compuestas por valores en formato hexadecimal, las cuales son enviadas o recibidas de forma serial por el XBee. En la Tabla 4.8 se muestra los campos que componen la estructura de la trama para transmitir un paquete.

El RTOS, mediante el hilo `HiloTransmitir`, genera la trama XBee con los datos correspondientes a cada sensor y transmite por el puerto serial al módulo XBee. La transmisión se realiza de forma individual por cada sensor, es decir, una estación con n sensores realiza n transmisiones en cada ejecución del `HiloTransmitir`. La carga útil de cada transmisión tiene

[11]Sotware gratuito desarrollado por el fabricante de XBee, Digi, para configurar y gestionar módulos XBee, y probar redes XBee
[12]Acuse de recibo para confirmar la recepción de un mensaje

Campo	Descripción
Delimitador de inicio	Indica el inicio de la trama, este valor siempre es 7E
Longitud	Indica el número total de bytes que tiene la trama
Tipo de trama	Indica el tipo de trama, código para transmisión es 10
ID trama API	Permite reconocer la trama y relacionarla con un ACK[12]
Direcc. dest. 64-bit	Dirección física del dispositivo destino
Direcc. dest. 16 -bit	Dirección temporal asignada al dispositivo destino en la red
Opción	Permite deshabilitar el ACK o enviar la trama a todas las PAN
Paquete de datos	Información que se desea enviar, hasta 100 bytes por paquete
Checksum	Suma de verificación para validar los datos de la trama

Tabla 4.8: Trama para transmitir un mensaje desde un XBee en modo API.

el siguiente formato:

$$\underbrace{\text{ID estación}}_{\text{12 bits}}, \underbrace{\text{ID sensor}}_{\text{4 bits}}, \underbrace{\text{Fecha-hora (epoch)}}_{\text{3 bytes}}, \text{muestra 1, muestra 2, ... , muestra n}$$

El número de muestras de cada sensor depende del tiempo de muestreo establecido en el archivo de configuración. Aunque la fecha y hora en formato *epoch* está compuesto de 4 bytes, sólo se transmite los 3 bytes menos significativos y en la recepción se debe completar con byte faltante con el correspondiente de la fecha y hora en ese instante. El byte más significativo cambia cada 3.2 días, y por lo tanto es el mismo que se obtiene en la transmisión y recepción. El número de bytes que ocupa cada muestra, mismo que depende del sensor, se presenta en la Tabla 4.7.

El microcontrolador utiliza el puerto serial para comunicarse con el módulo XBee, el módulo GPS, el computador para el depurador y un dispositivo adicional dedicado al control de la batería de alimentación (Sección 4.10). Todos los dispositivos están configurados para comunicarse una velocidad de 9600 baudios. Aunque existen microcontroladores con varios puertos seriales, éstos disponen de funcionalidades adicionales y pines extra que no serian utilizados, además su costo es mayor y no están disponibles en el mercado local para realizar pruebas. Aunque el PIC 18F26K20 sólo dispone de un puerto serial, las líneas de comunicación serial (TX, RX) se multiplexan y demultiplexan para permitir la conexión con varios dispositivos a la vez. Para este fin se utilizó el integrado FSA3357, el cual es un multiplexor/demultiplexor 3:1. El diagrama de conexiones internas y asignación de pines de este chip se presenta en la Figura 4.36.

La conexión del pin A con B0, B1, B2 se controla mediante los pines S1 y S2, de acuerdo

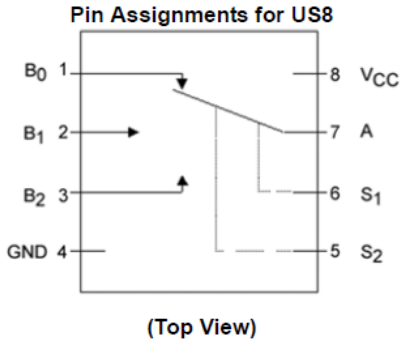


Figura 4.36: Asignación de pines multiplexor/demultiplexor del integrado FSA3357, Fuente: [8].

a la Tabla 4.9. De esta manera, el pin TX del puerto serial del microcontrolador se multiplexa entre el XBee, el computador y el espacio para la conexión del dispositivo adicional, y la señal proveniente del XBee, GPS y un dispositivo adicional se demultiplexa al pin RX del PIC. El esquema de conexión de los pines TX y RX con el multiplexor/demultiplexor se muestra en la Figura 4.37.

S1	S2	Función
0	0	Sin conexión
1	0	B0 conectado a A
0	1	B1 conectado a A
1	1	B2 conectado a A

Tabla 4.9: Tabla de función de los pines S1 y S2 del chip FSA3357.

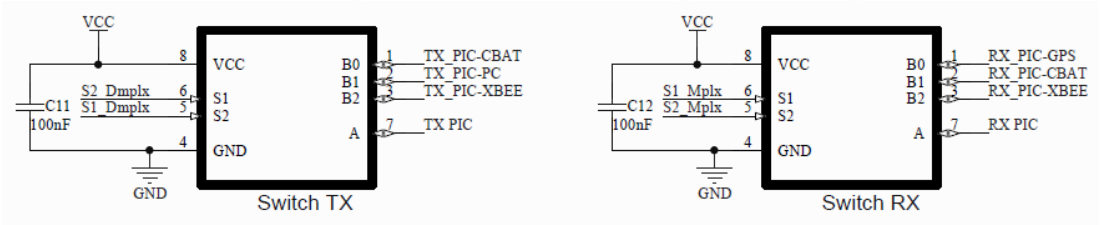


Figura 4.37: Esquema de conexión de los pines TX, RX del microcontrolador con los dispositivos Multiplexor/Demultiplexor FSA3357.

4.9. Depurador

El microcontrolador ha sido programado para permitir conocer si el dispositivo realiza las tareas correctamente. Este modo ha sido llamado modo depurador. El modo depurador se activa de forma física mediante dos interruptores en la tapa frontal del dispositivo, los cuales deben

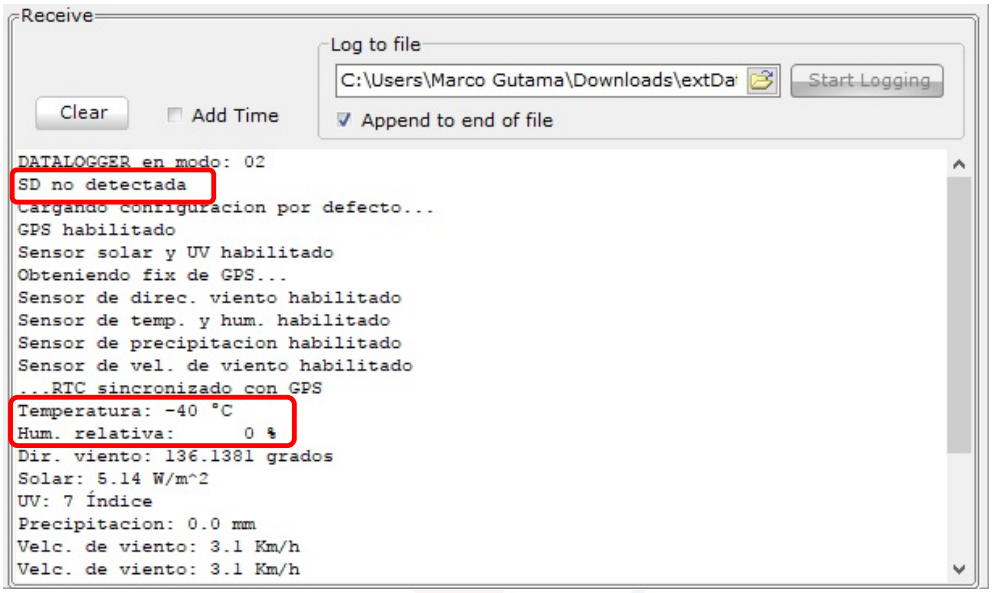
ser establecidos antes de encender el dispositivo, caso contrario el dispositivo no cambiará de modo. La configuración de estos dos interruptores para activar el modo depurador (modo 2) se muestra en la Tabla 4.2.

Este modo permite que el microcontrolador transmita, usando el protocolo UART, los diferentes eventos que suceden en el sistema. Los pines de conexión del puerto serial se encuentran en la tapa frontal del dispositivo (Figura 4.30). El modo depurador permite conocer el resultado de los siguientes procesos:

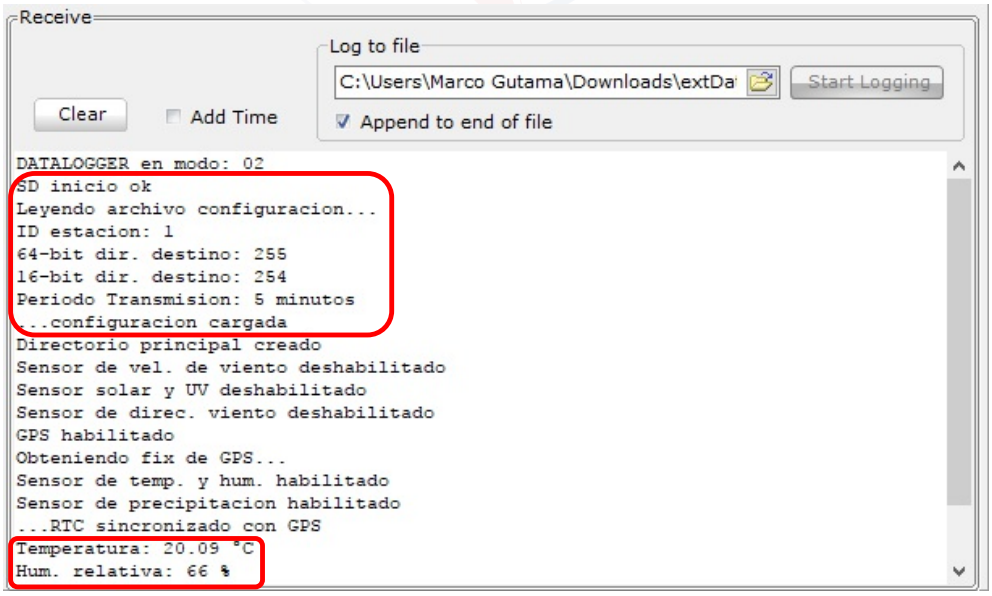
- Detección de una memoria SD.
- Comprobación del archivo de configuración.
- Carga de valores de configuración al RTOS (archivo de configuración o configuración por defecto).
- Sincronización de GPS con RTC.
- Habilitación de sensores para adquisición.
- Lecturas de cada sensor.
- Transmisión y almacenamiento de los datos.

El resultado de estos procesos se muestra en la Figura 4.38 mediante un monitor serial que se comunica con el dispositivo en modo depurador mediante el protocolo UART. En la Figura 4.38a el dispositivo indica que no ha sido posible detectar una memoria SD, y que pasa a usar la configuración por defecto para su funcionamiento. En esta configuración todos los sensores son activados, y la lectura de cada uno de ellos son presentados en el monitor, los cuales poseen valores de acuerdo a la realidad en ese instante, a excepción del sensor de temperatura y humedad. Las lecturas que obtiene de este sensor no son correctas, lo cual puede deberse a diversos factores, pero que en este caso ha sido desconectado a propósito para determinar las lecturas que se obtiene en este situación. Luego, al insertar la memoria SD en el dispositivo, se puede apreciar en la Figura 4.38b que el dispositivo ha detectado y leído la memoria correctamente. También, al conectar el sensor de temperatura y humedad, se puede notar que los valores leídos corresponden a la realidad.

La implementación del modo depurador en el sistema tiene una carga considerable en la memoria RAM y ROM del microcontrolador. En la Figura 4.39 se muestra el consumo de la memoria RAM y ROM al compilar el microcontrolador con y sin depurador. Aunque esta situación no es crítica para el normal funcionamiento del sistema, el modo depurador está implementado mediante directivas que permiten señalar al compilador de MikroC si el programa debe ser compilado con o sin este modo, de tal manera que el sistema puede ser recompilado si se cuenta con el código fuente.

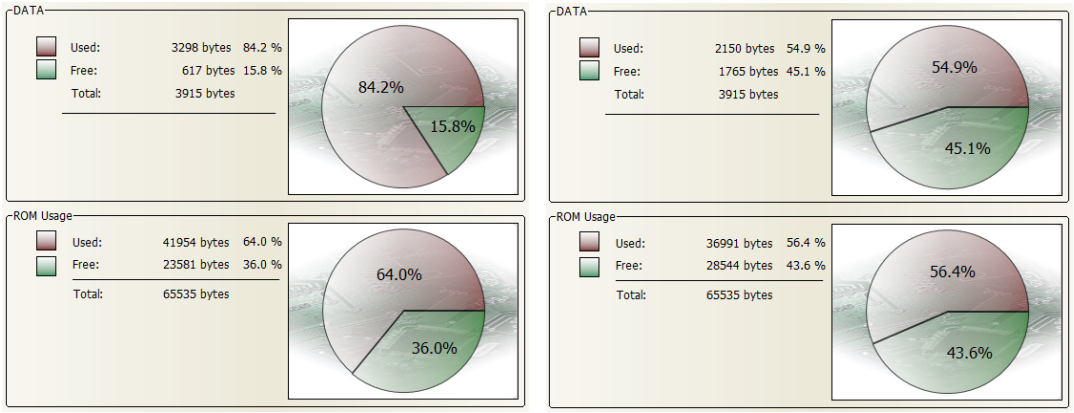


(a) Lectura de valores en monitor serial del prototipo en modo depurador sin memoria SD.



(b) Lectura de valores en monitor serial del prototipo en modo depurador con memoria SD.

Figura 4.38: Verificación de funcionamiento del prototipo mediante modo depurador.



(a) Compilación con el modo depurador activo (b) Compilación sin activar el modo depurador

Figura 4.39: Consumo de RAM y ROM en el microcontrolador.

4.10. Alimentación

El dispositivo ha sido diseñado para funcionar en lugares remotos utilizando una batería de 12 V. Se emplea un regulador de voltaje para obtener una tensión de 3,3 V para alimentar a todos los elementos del circuito (Figura 4.40).

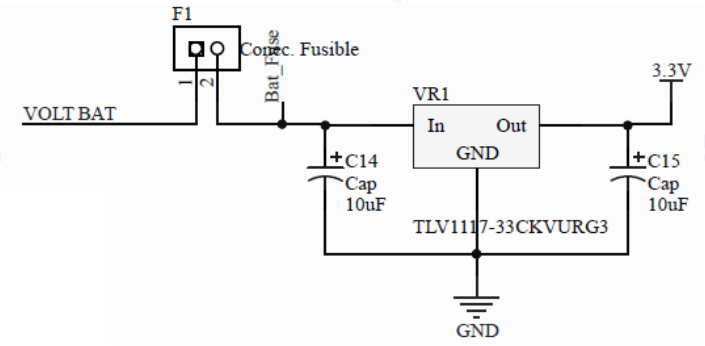


Figura 4.40: Regulador de voltaje a 3,3V.

Para conocer el nivel de la batería y establecer alertas, se adquiere la señal de tensión de la batería mediante un divisor de tensión pasivo en el ADC. El esquema del divisor se muestra en la Figura 4.41, y los valores de sus componentes están determinados por la Ecuación 4.8.

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in} \tag{4.8}$$

Si se establece la tensión de entrada $V_{in} = 12\,V$, la tensión de salida $V_{out} = 3{,}3\,V$ y se

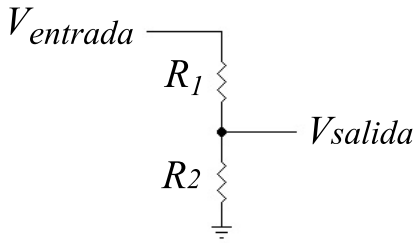


Figura 4.41: Divisor resistivo de tensión.

impone la resistencia R_2 con un valor comercial de $R_2 = 1\,k\Omega$, entonces la resistencia R_1 de la Ecuación 4.8 resulta $R_1 = 3{,}3\,k\Omega$. La conexión realizada al ADC en conjunto con el resto de señales analógicas de los sensores se muestra en la Figura 4.26. El nivel de tensión en la batería se determina mediante la Ecuación (4.9).

$$V_{bateria} = \frac{12\,V}{3{,}3\,V} * V_{ADC} \tag{4.9}$$

Con el objetivo de ofrecer una escalabilidad para trabajo futuros, se ha dispuesto de un espacio adicional en la multiplexación del puerto serial para comunicar el microcontrolador con un regulador de carga.

4.11. Placa de circuito impreso (PCB)

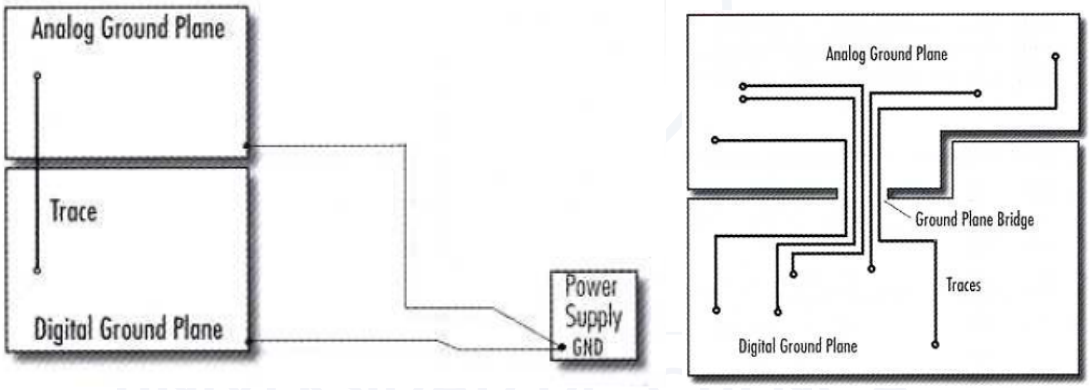
El diseño de la placa PCB se realizó considerando la naturaleza de las señales, el contenedor de la placa y la *Compatibilidad Electromagnética* (EMC). Se han diseñado dos placas, la primera contiene todo el circuito electrónico y la segunda todos los conectores de entrada y salida que se ubican en las tapas del contenedor. Las señales analógicas pueden ser muy sensibles al ruido generado por señales digitales y radiación electromagnética provenientes de dispositivos de radiofrecuencia. Con el objetivo de reducir éstas interferencias y tener compatibilidad electromagnética en el circuito se han seguido varias técnicas que se detallan a continuación.

4.11.1. Ruido de masa

Los planos de tierra o masa poseen reactancias parásitas que generan ruido en el circuito. Además, el ruido generado puede propagarse entre los diferentes componentes de la placa mediante la tierra, por lo que se debe evitar mezclar circuitos sensibles al ruido.

La EMC tiene dos principios básicos: 1) las corrientes deben retornar a la fuente de alimentación de forma local y lo más compactas posibles (a través del bucle de menor área posible), caso contrario se crea una antena de cuadro y 2) el circuito debe tener sólo un plano de referencia, ya que si se crean dos referencias en el circuito, se crea una antena de dipolo. En ambos casos, los efectos de no seguir éstos principios son indeseables [9].

Aunque en principio se puede dividir los planos de tierra en digital y analógico unidos a través de un solo punto a la fuente de alimentación como se muestra en la Figura 4.42a, el flujo de la corriente de retorno estaría forzada a recorrer un largo trayecto hasta la tierra de la fuente, formando así un gran bucle, lo cual es indeseable. Además se estaría creando una antena de dipolo a través de los planos analógico y digital, conectados a través de pistas largas. Sin embargo, si ambos planos están unidos a través de un puente entre ambos planos, como se muestra en la Figura 4.42b, el puente provee un camino para la corriente de retorno de las señales, lo cual produce un bucle muy pequeño [9].



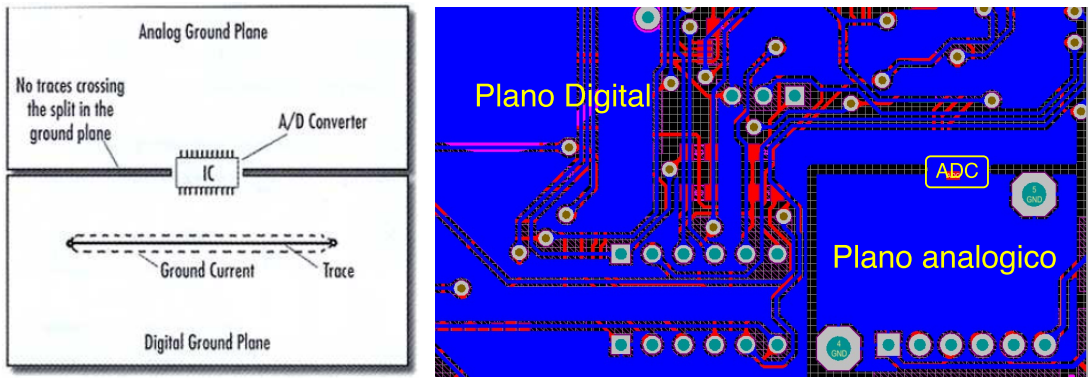
(a) Conexión directa a fuente de alimentación (b) Conexión mediante puente

Figura 4.42: Conexión de plano analógico y digital, fuente: [9].

En caso de utilizar un ADC, se recomienda conectar los circuitos analógicos y digitales mediante un puente justo debajo del ADC como se presenta en la Figura 4.43a. El puente entre los dos planos debe ser del ancho del ADC y las pistas a trazar no deben cruzar la parte dividida entre los planos [9]. En la Figura 4.43b se muestra el diseño de la placa con la separación de los planos digital y analógico mediante el ADC.

4.11.2. Masa flotante

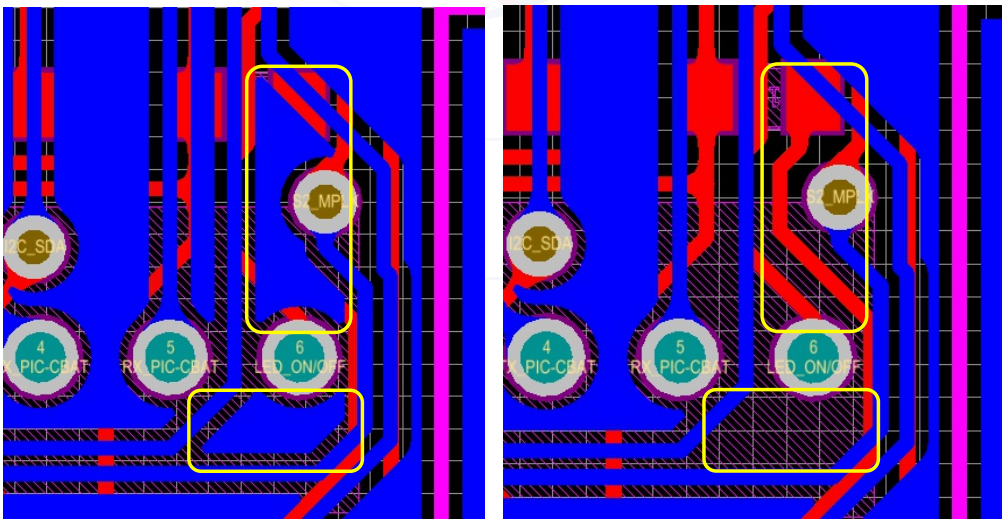
La masa flotante hace referencia a islas de cobre en la placa que no están conectadas a ningún plano de masa (Figura 4.44). Estas porciones de cobre se producen al crear planos de masa en el diseño. En estas regiones de masa flotante el ruido puede ser acoplado capacitivamente y el



(a) Esquema recomendado, fuente: [9] (b) Esquema realizado

Figura 4.43: Separación de planos con ADC.

efecto *crosstalk* o diafonía[13] entre las pistas cercanas a la región de cobre empeoraría. Por lo tanto, estas islas de cobre deben ser eliminadas o, en caso de ser posible, conectadas mediante vías al plano de masa [55].



(a) Masa flotante (b) Masa flotante eliminada

Figura 4.44: Plano de masa.

[13]Forma no intencionada de acoplo entre señales, debido a la interacción entre pistas, cables y componentes [54].

4.11.3. Pistas

Las pistas en un circuito impreso presentan características resistivas e inductivas, mismas que dependen del largo, ancho y grosor de la pista. Como se mencionó en la sección anterior, la cercanía entre pistas puede generar un acople indeseado entre ellas (*crosstalk*). Para evitar el *crosstalk*, se ha reducido las pistas que corren paralelas, y en las que no ha sido posible, se ha separado la distancia entre ellas para evitar el acople (Figura 4.45). Los componentes están agrupados de acuerdo a la naturaleza de las señales, minimizando de ésta manera la longitud de las pistas de conexión, lo cual a su vez reduce los efectos parásitos resistivos y capacitivos de las pistas.

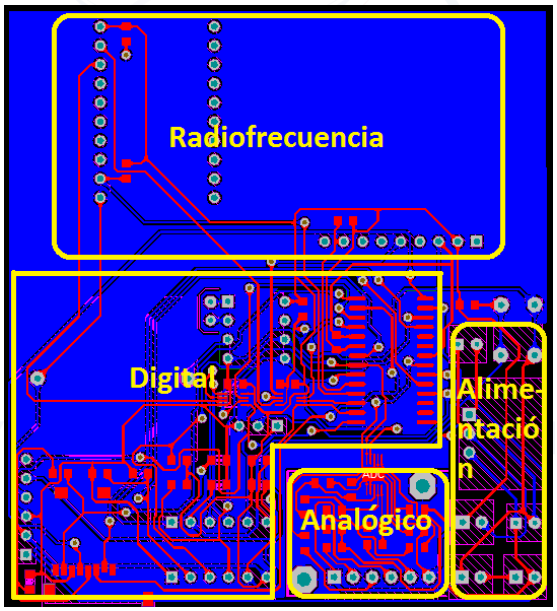
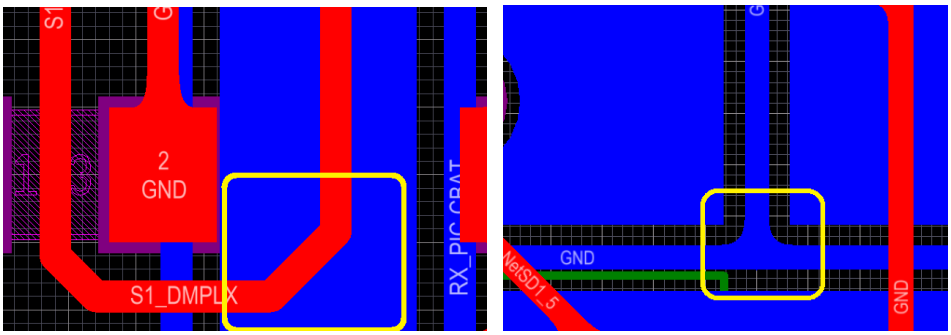


Figura 4.45: Bloques del circuito impreso.

El trazado de pistas con ángulos de 90 grados tienen un efecto insignificante en la integridad de la señal, incluso en frecuencias de hasta 100 GHz [56]. Un estudio realizado en [57] muestra que no hay diferencia medible de interferencia electromagnética radiada en este tipo de conexión. A pesar de esto, se recomienda evitar conexiones con ángulos de 90 grados suavizando el ángulo a 45 grados como se muestra en la Figura 4.46, lo cual ayuda a reducir bucles de corriente y mejorar la EMC [58].



(a) Giro de pista (b) Unión en T de pistas

Figura 4.46: Trazado de pistas con ángulos de 45 grados.

4.11.4. Vías

Las vías son caminos que permiten la conexión eléctrica entre pistas de diferentes capas a través de una perforación. Las vías poseen valores parásitos de capacitancia e inductancia que pueden degradar una señal al incrementar los tiempos de conmutación. Cuando el tamaño de las vías es menor, se reduce su capacidad parásita, pero se incrementa su inductancia y resistencia [59]. Estos efectos son más críticos en circuitos digitales de alta frecuencia (MHz), ya que las capacitancias parásitas cambian la impedancia de la señal y generan reflexiones. El circuito realizado no utiliza comunicaciones de alta velocidad, sin embargo, se ha evitado en lo posible las vías en pistas con señales rápidas y la generación de barreras con vías que bloqueen el paso del plano de masa, lo cual dificulta la circulación de los retornos de corriente.

4.11.5. Cristales

Los cristales de cuarzo pueden ser fuentes de ruido para las señales adyacentes. Por lo que es recomendable ubicar al oscilador o cristal lo más cerca posible del componente que lo utiliza. Además, es conveniente realizar un corte en el plano de masa alrededor del cristal y sus capacitores. De esta manera se evita que el ruido del cristal interfiera con las pistas cercanas. En la Figura 4.47 se muestra el corte realizado alrededor del cristal del RTC.

4.11.6. Capacitores de desacoplo

Las señales digitales de alta velocidad exigen a la fuente de alimentación cambios bruscos de corriente que no pueden ser suministrados inmediatamente debido a las características parásitas y distancia de las pistas a la fuente y, por lo tanto, producen cambios y ruido en la tensión

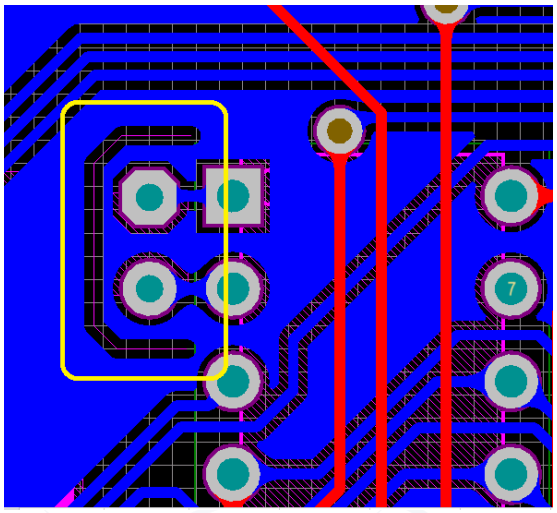


Figura 4.47: Corte en plano de masa alrededor de cristal de cuarzo.

de alimentación. Para evitar estos efectos es conveniente colocar un condensador de desacoplo los más cerca de los pines de alimentación del componente, ya que el condensador se opone a cambios bruscos de tensión y proporciona corriente acumulada en los momentos iniciales de demanda del componente. Bajo estos criterios se ha colocado en todos los componentes capacitores de desacoplo de 100 nF, valor típico recomendado para este propósito.

El diseño final del PCB se muestra en el Anexo F.

4.12. Contenedor

El diseño inicial y la construcción del contenedor fueron realizados por los ingenieros de la Red Sísmica del Austro.

Agentes externos y medioambintales, tales como la lluvia, el polvo o la humedad, pueden interferir negativamente con el funcionamiento del circuito del prototipo. Por esta razón, y, puesto que el dispositivo está destinado a ser usado a la intemperie, se consideraron criterios de tropicalización. La tropicalización consiste en diseñar los dispositivos de forma que no permitan que agentes que pueden provocar un mal funcionamiento o reducir el tiempo de vida de los circuitos afecten al sistema.

Tomando en cuenta criterios relacionados a la tropicalización, se diseñó un contenedor capaz de aislar a la parte principal del circuito del ambiente externo. De esta forma, los únicos componentes expuestos al ambiente son los conectores. Para diseñar el contenedor del dispositivo

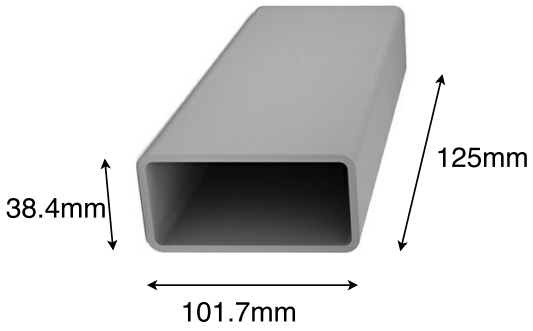


Figura 4.48: Dimensiones del tubo de aluminio usado en la construcción del contenedor.

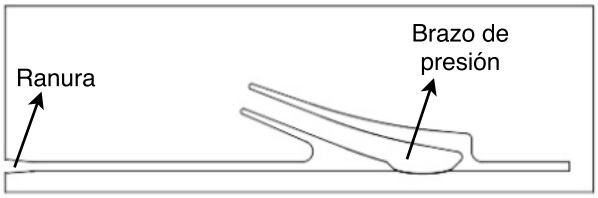
se consideraron materiales de bajo costo y fáciles de manipular. Se utilizó un tubo rectangular de aluminio, ya que este material es resistente, liviano y de bajo costo. El tubo rectangular fue cortado en los extremos de tal manera que cubra el largo de la placa. La placa principal fue diseñada teniendo en cuenta el ancho del tubo para que quepa exactamente en éste. Las dimensiones del tubo rectangular se presentan en la Figura 4.48.

Para sujetar la placa al interior del tubo y cubrir los extremos abiertos del mismo se utilizaron láminas de acrílico. El material acrílico es una variante del plástico, es liviano, resistente a la intemperie y radiación UV, de gran resistencia a impactos y de una alta durabilidad [60]. Además, la mecanización del acrílico es más fácil que la del aluminio, también se pueden realizar cortes más finos y precisos a través de un láser.

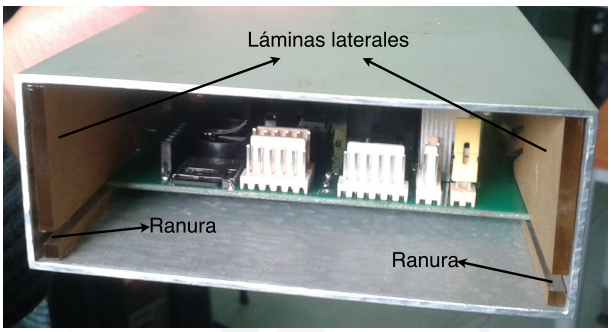
Para fijar la placa dentro del contenedor se utilizaron dos láminas de acrílico de 3mm en cada pared lateral al interior del tubo. Estas láminas fueron diseñadas con ranuras para que la placa PCB sea insertada en éstas, y luego existe la resistencia de un brazo, el cual se eleva al ser insertada la placa y ejerce una presión en esta para fijarla al contenedor. El diseño de las láminas, realizado en un programa CAD (Diseño Asistido por Computador) , se muestra en la Figura 4.49a. Las láminas en las paredes laterales del contenedor se muestran en la Figura 4.49b.

Para tapar los extremos del tubo y a su vez contener los diferentes conectores de entrada y salida a la placa principal se utilizaron láminas de acrílico de 5mm. Para contener las tapas mediante tornillos, se crearon marcos que van dentro del tubo. El diseño de los marcos y su colocación dentro del tubo se muestran en la Figura 4.50. Las láminas están agujereadas en las esquinas para sujetar las tapas del contenedor mediante tornillos.

La lámina frontal se diseñó con un espacio para ubicar la placa PCB de conectores como se muestra en la Figura 4.51. Para ubicar en la tapa posterior los conectores para las antenas de GPS, XBee y el interruptor de encendido se crearon agujeros mediante una máquina de *Control*

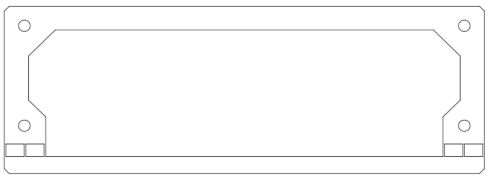


(a) Diseño

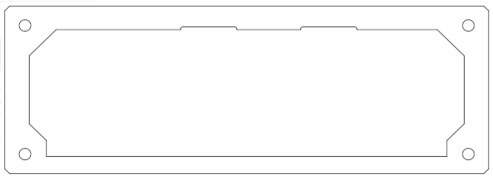


(b) Implementación

Figura 4.49: Láminas laterales para fijar la placa PCB.



(a) Diseño marco frontal



(b) Diseño marco posterior

(c) Colocación marco frontal

(d) Colocación marco posterior

Figura 4.50: Marcos para sujetar las tapas del contenedor.

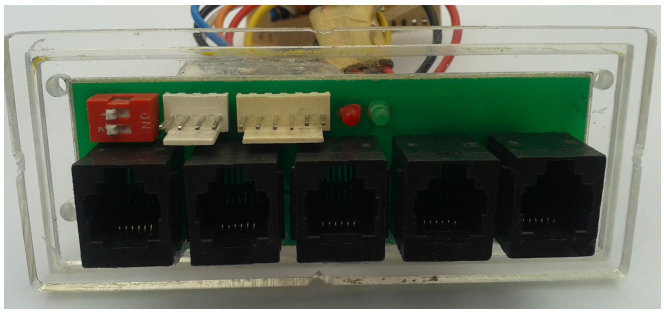


Figura 4.51: Tapa frontal del contenedor.

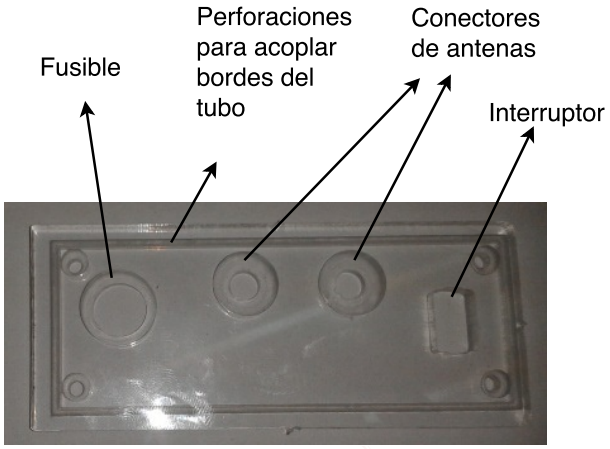
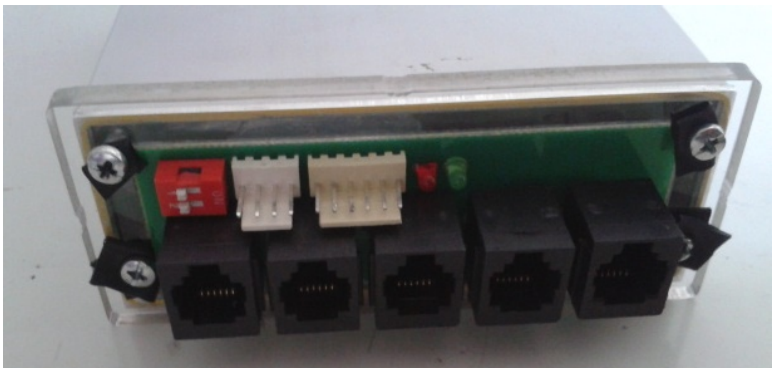


Figura 4.52: Dimensiones del tubo de aluminio del contenedor.

Numérico Computarizado (CNC) ya que estos necesitan acabados como se muestra en la Figura 4.52 para sostener los conectores de las antenas.

Para permitir que las tapas se acoplen a los bordes del tubo de aluminio se realizaron perforaciones hasta la mitad del grosor de la lámina mediante la CNC (Figura 4.52), de tal manera que los bordes del tubo quepan dentro de esas perforaciones.

Para evitar que por estas perforaciones ingrese polvo, agua, etc., se ha colocado un O-ring en cada conector de la tapa posterior y pedazos de caucho como arandelas en los tornillos que sujetan las tapas al dispositivo. El diseño final del contenedor se muestra en la Figura 4.53.



(a) Parte frontal

(b) Parte posterior

(c) Cuerpo

Figura 4.53: Contenedor construido.

Capítulo 5

Resultados y Discusión

En este capítulo se describen los resultados obtenidos durante el diseño, construcción y prueba del prototipo. Se tratan el funcionamiento final y los resultados obtenidos en varias pruebas en cuanto a almacenamiento y transmisión. El modo depurador , un modo que permite verificar el funcionamiento del prototipo en el lugar de instalación, también es tratado. Además, se presentan datos del consumo energético del sistema con sus diferentes componentes. Luego, se evalúa el funcionamiento del prototipo desarrollado comparando el error obtenido entre éste y una estación de referencia vs el error obtenido entre una segunda estación comercial con la misma estación de referencia. Finalmente, se realiza una explicación de la utilidad del *RTOS*.

5.1. Configuración del dispositivo

El dispositivo tiene 4 modos de funcionamiento, los cuales son fácilmente configurados mediante dos interruptores en la tapa frontal del dispositivo. Los diferentes modos permiten al usuario controlar el funcionamiento del sistema. Sin embargo, recordar la combinación de los interruptores para establecer el modo deseado puede resultar confuso, por lo que se ha colocado una etiqueta en el dispositivo mostrando los modos de funcionamiento y la combinación correspondiente en los interruptores.

La configuración del identificador de la estación, dirección del módulo XBee receptor, transductores disponibles, periodo de muestreo de cada transductor y la transmisión mediante un archivo de texto en la memoria SD permiten al usuario establecer y modificar los mismos de acuerdo a sus requerimientos para el funcionamiento del dispositivo.

En caso de no detectarse la memoria SD o el archivo de configuración, el dispositivo utiliza una configuración base almacenada en el sistema. Por lo tanto, el dispositivo puede prescindir de la memoria SD. Esta situación brinda una mayor autonomía al dispositivo.

5.2. Almacenamiento

Un ejemplo de datos guardados por el dispositivo en la memoria SD se muestra en la Figura 5.1. El nombre del directorio principal (`001_EST`) corresponde al identificador de la estación establecido en el archivo de configuración. Los nombres de los subdirectorios corresponden a la fecha de registro de datos, misma que corresponden a las fechas en las que el dispositivo estuvo en funcionamiento.

Cada subdirectorio contiene archivos con las muestras registradas de cada transductor en ese día. Los transductores registrados corresponden a los habilitados mediante el archivo de configuración. Los archivos llevan nombres abreviados de su transductor de acuerdo a la Tabla 4.6. La Figura 5.2 muestra, mediante el programa *WinHex*[1], parte de los valores registrados en formato hexadecimal del transductor de velocidad de viento (ANM) del segundo día de registro. De los dos primeros valores, 12 bits corresponden al identificador de la estación (igual a 1), y los 4 bits restantes identifican el transductor (igual a 4), los siguientes tres valores corresponden a la fecha y hora en formato epoch (3 bytes menos significativos), y los valores que siguen corresponden a las muestras del transductor, en donde cada muestra puede ocupar 1 byte o dos, dependiendo del tipo de transductor de acuerdo a la Tabla 4.7.

[1] `http://www.winhex.com/winhex/hex-editor.html`

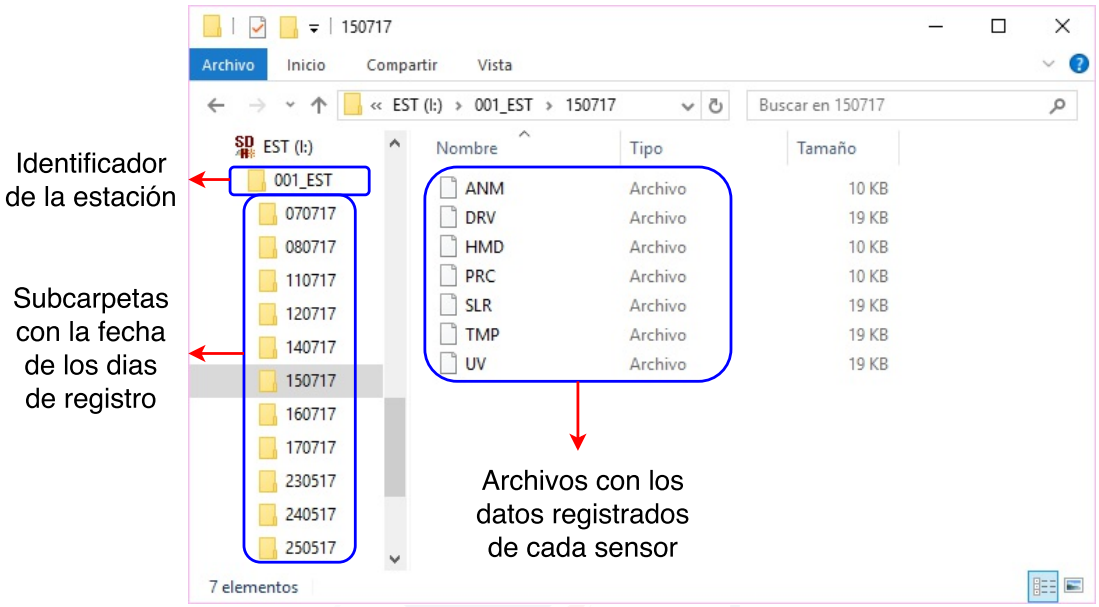


Figura 5.1: Datos registrados en la memoria SD del dispositivo.

Teniendo un registro de todos los transductores a una tasa de muestreo de acuerdo a la Tabla 4.4, el dispositivo escribe aproximadamente cada día en la memoria SD 101 Kbytes. De esta manera, con una memoria SD de 1 GB de capacidad, y el dispositivo funcionando de forma continua, el periodo de registro de datos es de $1048576kB/101kB \simeq 10382$ días, o $10382/365 \simeq 28$ años, aunque esto dependerá también del tamaño de clusters de la memoria.

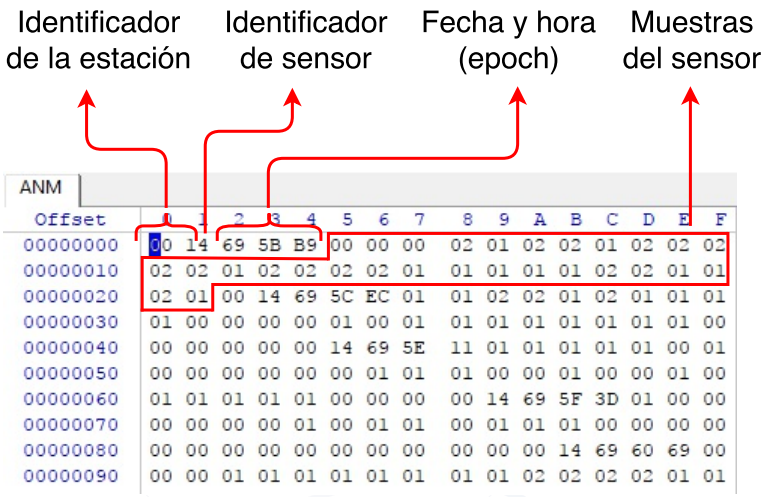


Figura 5.2: Valores registrados del transductor de velocidad de viento visto mediante el programa WinHex.

5.3. Transmisión

Para validar la transmisión de datos del prototipo se utilizaron dos módulos XBee-PRO S2B ZB, uno como transmisor y otro como receptor, en conjunto con la plataforma Arduino[2] Mega y la distancia entre el transmisor y receptor fue aproximadamente de dos metros. La plataforma Arduino había sido programada para procesar y separar la carga útil de la transmisión, misma que era retransmitida a un computador mediante comunicación serial, y a un módulo GSM para transmitir los datos a un servidor web.

Esta prueba se realizó por un periodo de 1 hora con tres sensores, en donde los datos recibidos fueron visualizados a través de un monitor serial en el computador. Se pudo notar que las transmisiones intermedias se perdían, es decir, la primera y la tercera transmisión se receptaba correctamente, mientras que la segunda se perdía. Esto pudo deberse a que en cada intervalo de transmisión al módulo XBee, el dispositivo genera una trama de datos por cada transductor y los envía sucesivamente, sin ningún espacio de tiempo entre trama y trama. Además, en la recepción la plataforma Arduino procesaba cada trama de datos recibidos, tiempo en el cual se podría haber estado ignorando la trama siguiente hasta terminar de procesar la trama anterior. Para solucionar esta situación se agregaron retardos de 100 ms después de cada transmisión, con el fin de proveer tiempo de procesar los datos correctamente tanto al módulo transmisor como receptor.

Con el objetivo de facilitar la visualización de datos recibidos se desarrolló un programa en

[2] https://www.arduino.cc/

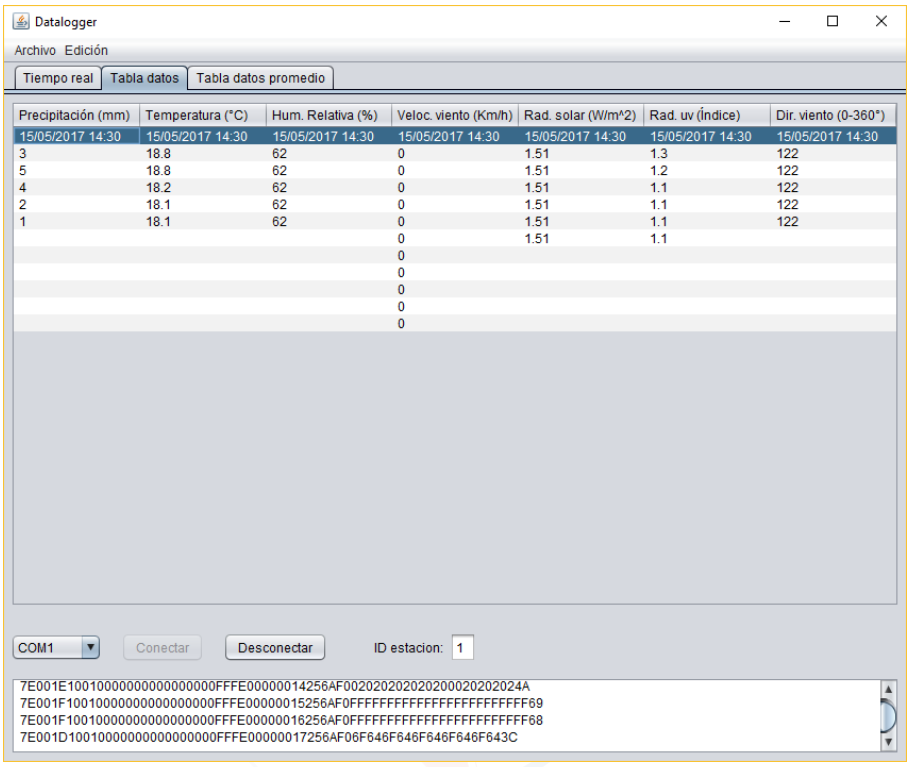


Figura 5.3: Últimas series de datos recibidos. Se utiliza el programa desarrollado en JAVA.

JAVA. El programa decodifica los datos y los presenta en tablas. En la Figura 5.3 se muestra los datos recibidos mediante el programa desarrollado en cada transmisión del dispositivo, y en la Figura 5.4 los promedios de los datos por transmisión. Los registros de cada tabla se pueden exportar a archivos en formato Excel mediante un menú emergente (pop-up) al dar click derecho sobre la tabla. Además, el programa detecta si el dispositivo está en modo de transmisión de tiempo real y muestra el estado actual de cada variable de forma gráfica que se recibe en tiempo real (Figura 5.5).

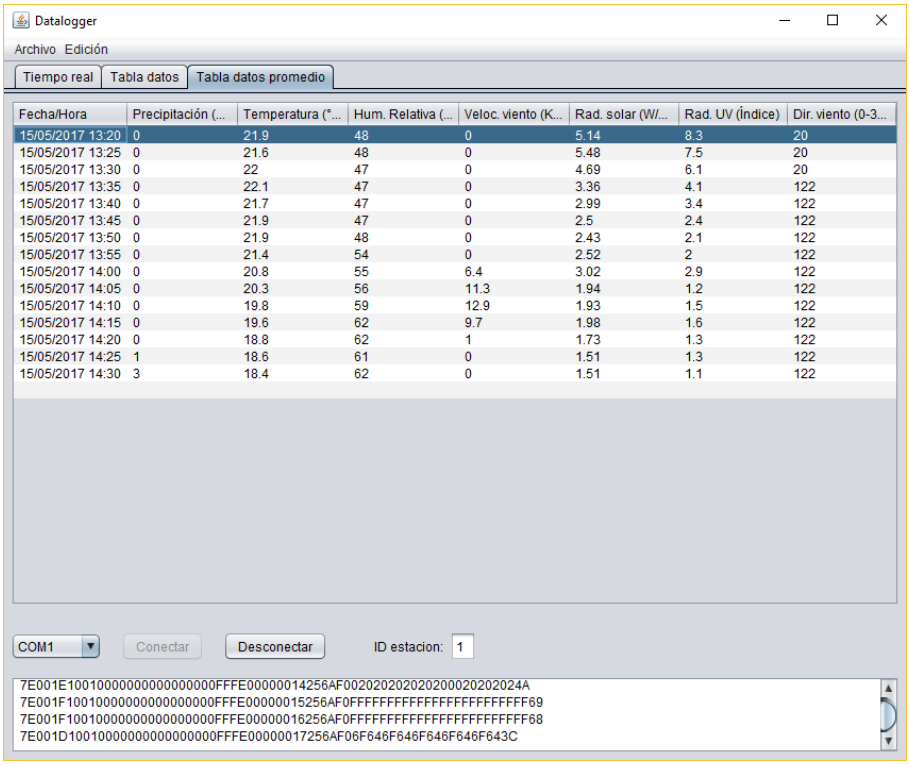


Figura 5.4: Promedios de las series de datos recibidos. Se utiliza el programa desarrollado en JAVA.

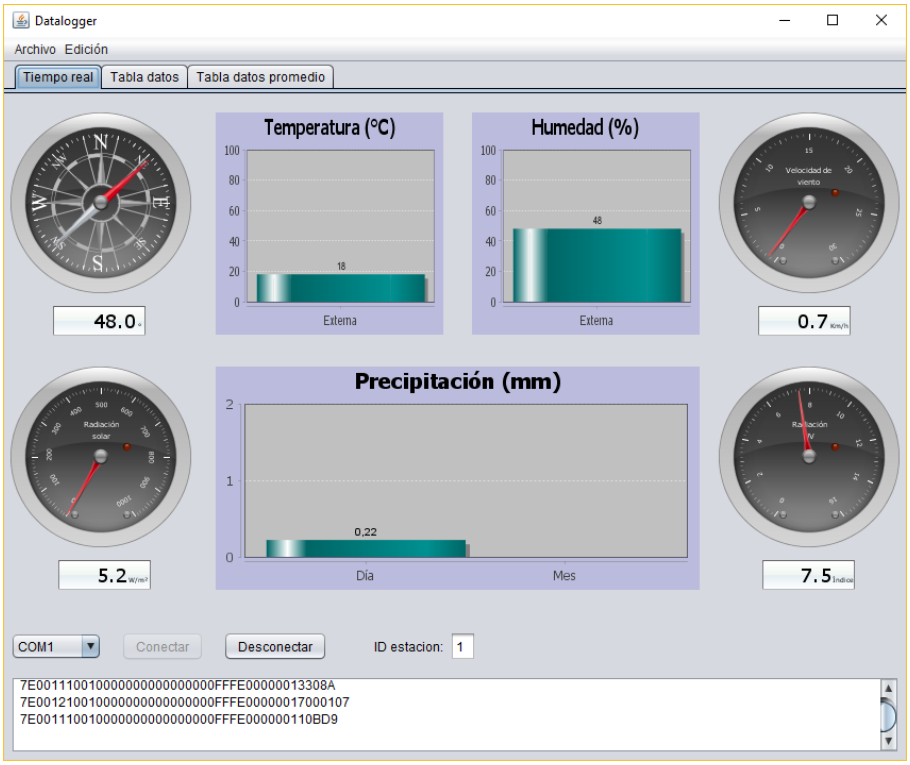


Figura 5.5: Visualización en forma gráfica de datos en tiempo real. Se utiliza el programa desarrollado en JAVA.

5.4. Contenedor

El contenedor debe proteger al dispositivo de las condiciones medioambientales adversas a las que estará expuesto como lluvia y humedad. Por lo tanto, debe ser impermeable. Para probar esta cualidad, se utilizó el contenedor únicamente con los conectores de las tapas frontal y posterior, y se lo sumergió completamente en una tina con agua como lo muestra la Figura 5.6a.

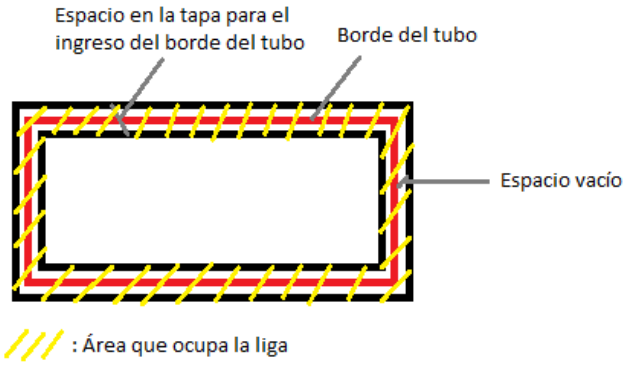
(a) Contenedor sumergido en agua (b) Evacuación del agua ingresada al contenedor

Figura 5.6: Prueba de impermeabilidad del contenedor.

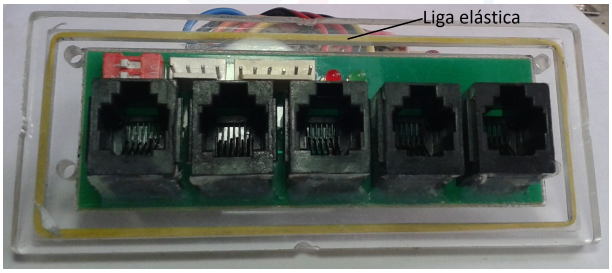
Después de un periodo de 5 minutos, al retirar el contenedor, se comprobó que una pequeña cantidad de agua había ingresado (Figura 5.6b). Para identificar las secciones por donde se filtraba el agua, se sumergió nuevamente el contenedor de forma vertical, primero con la tapa frontal colocada y sumergida y la otra descubierta como se muestra en la Figura 5.7 y luego de forma inversa.

Figura 5.7: Contenedor con la tapa frontal colocada y sumergida, y la posterior descubierta.

Se pudo observar que el agua ingresaba por la unión entre el tubo y la tapa. Para mejorar esta situación, se colocó una liga elástica entre la tapa y los bordes del tubo. Esta liga llena el espacio que existía entre el tubo y la tapa permitiendo todavía que los bordes del tubo quepan en la tapa (Figura 5.8).



(a) Área que ocupa la liga elástica



(b) Colocación de liga elástica

Figura 5.8: Liga elástica en los espacios donde se ajustan los bordes del tubo.

Aunque esta prueba fue un caso extremo, sirvió para identificar por donde ingresaba el agua al contenedor. En pruebas de funcionamiento, se dejó el dispositivo a la intemperie más de tres días, que resultaron poco lluviosos. En la inspección se pudo notar que existía poca humedad dentro del dispositivo (Figura 5.9).

Figura 5.9: Prototipo dejado a la intemperie en la estación.

5.5. Consumo energético

Conocer el consumo de energía del dispositivo desarrollado es importante debido a que está dirigido a funcionar en ubicaciones donde generalmente no se cuenta con alimentación de la red eléctrica. Por lo tanto, conocer este parámetro permite dimensionar de forma adecuada la fuente de energía.

Con el fin de realizar una aproximación del consumo de energía se usó una fuente alimentación continua de 12 Voltios, una tensión de alimentación comúnmente usada en este tipo de dispositivos. En la Tabla 5.1 se puede ver la potencia y la corriente requeridas por el dispositivo con todos sus elementos conectados y también con solo algunos de ellos. Además, se especifica si el GPS está activo sincronizándose o si el prototipo está adquiriendo los datos de los transductores.

GPS	XBee	SD	transductores	Lluvia	Modo depurador	Corriente(mA)	Potencia (mW)
No	No	No	No	Sí	No	17,3	207,6
No	No	Sí	No	No	No	17,8	213,6
Sí	No	Sí	No	No	No	18,1	217,2
No	No	Sí	No	Sí	No	18,2	218,4
No	Sí	Sí	No	No	No	74,2	890,4
Sí	Sí	Sí	Adq	Adq	No	74,8	897,6
Sí	Sí	Sí	No	No	No	75,1	901,2
Sí	Sí	Sí	Sí	Sí	Sí	77,8	933,6
Sí	Sí	Sí	Adq	Adq	Sí	78,7	944,4
Sí	Sí	Sí	Sí	Sí	Sí	81	972
Sinc	Sí	Sí	No	No	No	103	1236
Sinc	Sí	Sí	Sí	Sí	Sí	105,5	1266

Tabla 5.1: Corriente y potencia consumidas por el dispositivo cuando están conectados todos sus elementos y solo algunos de ellos ordenados ascendentemente por consumo. Sinc: el GPS está sincronizándose; Adq: los datos de los transductores están siendo obtenidos.

5.6. Verificación de medidas

Para realizar las pruebas se contaron con dos estaciones Davis Vantage Pro2 Plus completas. Una de estas estaciones se usó como referencia puesto que estaba instalada y siendo usada por el PROMAS (estación Referencia). Esta estación se encontraba en la cubierta del edificio de la facultad de arquitectura y urbanismo, edifico A del campus central de la Universidad de Cuenca. La segunda estación (estación Davis 2) fue facilitada también por el PROMAS. Los transductores que se usan en el dispositivo de prueba (Prototipo) pertenecen a la estación Davis 2. La segunda estación fue instalada junto a la primera como se muestra en la Figura 5.10. Ambas estaciones son de iguales características, pero, debido a su diferente tiempo de uso, sus transductores presentan diferente sensibilidad.

Los datos de las estaciones comerciales fueron descargados a través de su consola y los del prototipo se tomaron de la memoria SD de respaldo y fueron procesados para lograr tener promedios que coincidan en tiempo con los datos de la estación de referencia. También los datos fueron escalados por un factor calculado para corresponder con los valores reales.

(a) Vista delantera de las estaciones instaladas. La estación de referencia se encuentra a la derecha

(b) Vista posterior de las estaciones instaladas

Figura 5.10: Estaciones ubicadas sobre la cubierta del edificio de la Facultad de Arquitectura y Urbanismo, edificio A del campus central de la Universidad de Cuenca.

5.7. Análisis de datos

Con el fin de conocer si los datos tomados por el prototipo son comparables a los tomados por una estación comercial se realizaron dos comparaciones: entre dos estaciones comerciales (Referencia vs Davis 2) y entre una estación comercial y el prototipo (Referencia vs Prototipo) como se detalló en la sección 5.6.

Las estaciones comerciales (Referencia y Davis 2) registran los datos cada 5 minutos coincidiendo con múltiplos de 5 minutos (i.e. 00:00, 00:05, etc.). Sin embargo los datos son tomados a un intervalo de tiempo menor, luego promediados y enviados al final del periodo (exactamente cada cinco minutos); además se registra el mínimo y el máximo. Por otro lado, el prototipo

toma medidas cada 10 segundos y las reporta todas al final de los cinco minutos. Debido a esta diferencia, se hizo necesario procesar los datos para conseguir que sean comparables. La comparación se realizó entre los promedios.

Se tomaron datos a partir de la sincronización con el GPS, a las 13:12:23 del un día hasta las 11:06:41 tres días después. Sin embargo, los datos a comparar son desde las 21:50 del primer día hasta las 11:05 del último día debido a que la estación Davis 2 tardó en conectarse con la consola e iniciar el registro de datos (inicio de comparación) y el prototipo se desconectó primero (fin de comparación). Por lo tanto, el resto de datos no se tomaron en cuenta en el análisis.

El GPS se sincroniza cuando se inicia el prototipo y cada 24 horas, es decir, todos los días a la misma hora. Durante la sincronización el sistema deja de tomar datos de los transductores. Debido al tiempo que se usó en la sincronización, el segundo día se perdieron 4 promedios (un tiempo aproximado a 20 minutos) y el tercero se perdieron 11 (un tiempo aproximado a 55 minutos). En total, la pérdida representa 1.82 % de los datos. La sincronización es más rápida (menos de 5 minutos) cuando se conecta la antena, así no se pierden datos. Sin embargo, debido a que se dejó de contar con el conector adecuado, se usó solo un conector sin la antena.

Los datos comparados de Referencia vs Davis 2 fueron 268 muestras, mientras que los de Referencia vs Prototipo fueron 721, excepto para la variable de radiación UV donde la estación de referencia tenía muestras faltantes al final, por lo que se obtuvieron 691 muestras.

Los datos guardados por la estación son acondicionados previamente, pero requieren ser multiplicados por un factor para llegar a representar las medidas reales. Este factor es conocido por las características de cada transductor. Sin embargo, para el caso de los transductores de radiación solar, de radiación ultravioleta y de velocidad de viento, se calcularon factores de conversión que dan resultados más aproximados a los datos de referencia debido a las diferencias entre los transductores de las dos estaciones que se han producido por el uso y la falta de calibración; estos factores reemplazaron a los calculados a partir de las características teóricas de los transductores. Para los datos del transductor de velocidad de viento se usó el factor de 1,706 en lugar de 0,724. Para los de radiación solar se usó 0.0693 en lugar de 0,0749 y para los de radiación UV se usó 0,000704 en lugar de 0,000833.

Para el caso de la lluvia, la mayoría de datos eran cero debido a que los días que se realizaron las pruebas hubo poca precipitación, pero se verificó el funcionamiento del transductor y el correcto registro de los datos de forma manual accionando directamente el balancín.

En el análisis de datos se calcularon las siguientes métricas de error para comparar las observaciones de las estaciones: *Suma Acumulada de Errores de Pronóstico* (CFE) , *Desviación Media Absoluta* (MAD)) y *Error Cuadrático Medio* (MSE) Las distintas métricas de error

realzan diferentes características de las comparaciones.

La CFE es la suma de todas las diferencias entre cada muestra de las dos series de datos y permite ver el sesgo de los datos, es decir, si se encuentran subestimados o sobrestimados. Un valor de CFE negativo significa que el valor comparado generalmente es menor al de referencia, es decir, que la mayoría de veces se subestima el valor. Por otro lado, si el error es positivo los datos han sido sobrestimados. Si el valor de CFE se acerca a cero quiere decir que el error es producido por ruido y que no proviene del sistema. Hay que tener en cuenta que la cantidad de datos de la comparación entre la estación de referencia y el prototipo es un poco mayor a 2,5 veces la de la comparación entre la estación referencia y estación Davis 2. Su fórmula se presenta en la Ecuación (5.1).

$$CFE = \sum_{i=1}^{n} (\hat{Y}_i - Y_i) \tag{5.1}$$

donde:

n: es la cantidad de muestras.
$\hat{Y}_i$: el valor de la muestra de prueba.
Y_i: el valor de la muestra de referencia.

La métrica MAD mide la cantidad de error que existe en las muestras. Se calcula sumando el valor absoluto de las diferencias entre las muestras y dividiendo para el número de muestras. El valor absoluto permite conocer la magnitud del error e ignorar el signo. Su fórmula se muestra en la Ecuación (5.2)

$$MAD = \frac{1}{n} \sum_{i=1}^{n} \left| \hat{Y}_i - Y_i \right| \tag{5.2}$$

donde:

n: es la cantidad de muestras.
$\hat{Y}_i$: el valor de la muestra de prueba.
Y_i: el valor de la muestra de referencia.

El MSE se parece al MAD, pero da mayor importancia a los errores grandes. Se calcula sumando el cuadrado de las diferencias entre las muestras y dividiendo para el número de muestras, como se indica en la e la Ecuación (5.3).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{5.3}$$

donde:

n: es la cantidad de muestras.
$\hat{Y}_i$: el valor de la muestra de prueba.
Y_i: el valor de la muestra de referencia.

Los errores entre la estación de referencia y la estación Davis 2 se muestran en la Tabla 5.2. Las métricas de error entre las observaciones de la estación de referencia y el prototipo se muestran en la Tabla 5.3. En la Tabla 5.4 se muestran las métricas de los errores de ambas comparaciones.

Variable	CFE	MAD	MSE
Temperatura	59,80	0,29	0,12
Humedad	-474,00	2,15	6,76
Velocidad del viento	-520,00	1,95	6,80
Lluvia	0,00	0,00	0,00
Radiación solar	-3489,00	21,94	7401,56
Índice UV	-107,70	0,43	0,67
Dirección de viento	-1440,00	19,89	1267,72

Tabla 5.2: Errores obtenidos al comparar las observaciones de la estación de referencia con las observaciones de la estación Davis 2.

Variable	CFE	MAD	MSE
Temperatura	-406,95	0,56	0,37
Humedad	564,88	1,70	3,97
Velocidad del viento	-495,86	0,94	2,19
Lluvia	-0,31	0,00	0,00
Radiación Solar	7760,34	28,78	3775,42
Radiación solar -–255	-1126,26	18,24	3605,91
Índice UV	377,79	0,97	1,85
Índice UV -–255	475,90	0,69	1,82
Índice UV -–511	231,80	0,34	1,51
Índice UV -–1023	101,70	0,15	1,39
Dirección de viento	5298,21	91,29	10464,57

Tabla 5.3: Errores obtenidos al comparar las observaciones de la estación de referencia con las observaciones del prototipo.

En las siguientes subsecciones se presentan los gráficos que comparan las series temporales y

Variable	Estaciones	CFE	MAD	MSE
Temperatura	Ref-Dav2	59,80	0,29	0,12
	Ref-Pro	-406,95	0,56	0,37
Humedad	Ref-Dav2	-474,00	2,15	6,76
	Ref-Pro	564,88	1,70	3,97
Velocidad de viento	Ref-Dav2	-520,00	1,95	6,80
	Ref-Pro	-495,86	0,94	2,19
Lluvia	Ref-Dav2	0,00	0,00	0,00
	Ref-Pro	-0,31	0,00	0,00
Radiación solar	Ref-Dav2	-3489,00	21,94	7401,56
	Ref-Pro	7760,34	28,78	3775,42
	Ref-Pro-255	-1126,26	18,24	3605,91
Índice UV	Ref-Dav2	-107,70	0,43	0,67
	Ref-Pro	377,79	0,97	1,85
	Ref-Pro-255	475,90	0,69	1,82
	Ref-Pro-511	231,80	0,34	1,51
	Ref-Pro-1023	101,70	0,15	0,56
Dirección de viento	Ref-Dav2	-1440,00	19,89	1267,72
	Ref-Pro	5298,21	91,29	10464,57

Tabla 5.4: Comparación de errores de ambas estaciones. Ref-Dav2 corresponde a la comparación entre la estación de referencia y la estación Davis 2 y Ref-Pro corresponde a la comparación entre la estación de referencia y el prototipo. Ref-Pro-255, -511 y -1023 indican que se restó al valor del prototipo 255, 511 y 1023, respectivamente.

muestran el error, el gráfico de dispersión de las series de tiempo y el histograma de los errores (diferencia entre observaciones) para cada variable estudiada.

5.7.1. Temperatura

La Figura 5.11 muestra la comparación de los datos obtenidos para la variable de temperatura. La estación Davis 2, en general, sobrestima el valor de la temperatura, mientras que el prototipo lo subestima. Además, ambas estaciones presentan errores bastante bajos. El error del prototipo resulta mayor según las métricas de error (Tabla 5.4), sin embargo, según el histograma del error, el error está en el orden de -0.5 °C con mayor frecuencia (Figura 5.11b).

Los datos con valores mayores se ajustan mejor en ambos casos, mientras que los datos con valores bajos resultan sobrestimados con la estación Davis 2. Para el prototipo los valores bajos son subestimados pero muy cercanos al valor de referencia según el gráfico de dispersión de la figura 5.11b. El coeficiente de correlación es r=1, lo que evidencia que se podría mejorar la

aproximación con una transformación lineal de las muestras del prototipo. Por ejemplo, teniendo en cuenta que la frecuencia de los errores en aproximadamente -0.5 °C es mayor (por mucho) y que el gráfico de dispersión (Figura 5.11b) muestra una subestimación igualmente distribuida, se podría sumar 0.5 °C al valor entregado por el prototipo para mover la media de los errores a cero y mantener el mismo coeficiente de correlación r=1.

5.7.2. Humedad

La Figura 5.12 muestra la comparación de los datos obtenidos para la variable de humedad. Según el gráfico de dispersión de la Figura 5.12b, el prototipo sobreestima los valores altos y subestima los valores bajos que es lo contrario cuando se compara la estación Davis 2 (Figura 5.12a). Esta sobrestimación y subestimación del prototipo hace que se presenten dos valores máximos en el histograma de los errores de la Figura 5.12b: en aproximadamente 1 % y -2 %.

Al igual que en la temperatura, el coeficiente de correlación es r=1. Esto indica que se puede hacer cierta transformación lineal a los datos presentados por el prototipo para disminuir el error. Una alternativa podría ser multiplicar por un factor a los valores altos y por otro factor a los valores bajos para disminuir la frecuencia relativa de los errores que se presentaran en el histograma (Figura 5.12b) y hacer que el error se acerca a cero.

5.7.3. Velocidad de viento

Para esta variable, ambas estaciones tiene un error negativo. El prototipo muestra menor error que la segunda estación comercial. La unidad en la que se mide la variable es kilómetros por hora (km/h). En este caso, se calculó un nuevo factor, diferente del teórico, que se ajustaba mejor a las características del transductor.

En la Figura 5.13 se ve el comportamiento de la velocidad de viento. La Figura 5.13a y la Figura 5.13b corresponden a las comparaciones entre la estación de referencia y la segunda estación comercial y entre la estación de referencia y el prototipo respectivamente. Los gráficos de dispersión tienen una forma parecida, pero con el prototipo se aproximan más los valores, por lo tanto existe menos error y se presenta con mayor frecuencia en los valores cercanos a cero (ver histograma de la Figura 5.13b). Se debe a que se calculó un factor más ajustado a las características reales del transductor.

5.7.4. Lluvia

La Figura 5.14 muestra la comparación de los datos obtenidos para la variable de lluvia. En este transductor el error es bajo, e incluso cero, pero muy probablemente esto sea debido a que existen pocas muestras diferentes de cero porque las pruebas se hicieron en un tiempo con poca precipitación. Por lo tanto, se requieren más datos con lluvia mayor a cero para que estos valores tengan un significado. Sin embargo, el funcionamiento del transductor puede ser comprobado de forma manual. La distribución de los puntos es similar en las Figuras 5.14a y 5.14b).

5.7.5. Radiación solar

La radiación solar es medida en W/m^2. El transductor presenta un rango activo bastante amplio: desde 0 hasta 1800, por lo que las diferencias presentan valores mayores. La CFE es menor con la segunda estación comercial y de signo contrario que con el prototipo. El MSE muestra que, aunque el error total es menor, la segunda estación comercial presenta más valores con error grande que el prototipo. Debido a un problema que se presentó en la placa del prototipo, su ADC toma erróneamente como 255 a todos los valores menores a 255. Este error no pudo ser corregido físicamente, por lo que se restaron 255 a los datos y se calculó un nuevo factor. Los resultados fueron mejores, ya que la correlación no se ve afectada, pero el error disminuye.

En la Figura 5.15 se pueden ver gráficos que muestran el comportamiento de la radiación solar tanto en series de tiempo como en graficos de dispersión y su error. En la Figura 5.15a se ve la comparación que usa la segunda estación Davis y en la Figura 5.15b se ve la comparación que usa el prototipo. Se puede ver que los datos se aproximan a una línea recta en el gráfico de dispersión, especialmente en los valores menores, pero que existen muchos datos que salen de esta tendencia provocando que existan errores grandes. Sin embargo, esa tendencia se en ve en ambas comparaciones. Estos errores se pueden deber a que la radiación es parcialmente por nubes que podrían cubrir solo una de las estaciones. Se realizaron dos comparaciones con sus respectivos gráficos para el caso del prototipo. Por eso se hace una comparación segunda comparación para el prototipo. Se puede ver que el comportamiento es bastante similar y que la transformación que se usa corrige en parte el problema (ver Figura 5.15c).

5.7.6. Índice UV

La Figura 5.16 muestra la comparación de los datos obtenidos para el índice UV. En esta caso el rango activo va de 0 a 16. La comparación con la segunda estación de prueba presentó valores en su mayoría subestimados. Pasa lo contrario al usar el prototipo, son sobrestimados. Al igual que en el caso del transductor de radiación solar, en este transductor también se restó un valor a los datos y se calculó un nuevo factor. Esta operación se realizó con varios valores.

Para el índice de radiación UV se realizaron varias comparaciones con los datos de prueba por la misma razón que para el transductor de radiación solar. En la Figura 5.16b se pueden ver los diferentes resultados obtenidos, serie de tiempo y error, gráfico de dispersión y distribución del error al restar 255 (Figura 5.16c), 511 (Figuras 5.16d) y 1023 (Figuras 5.16e) a los valores del prototipo. En la Figura 5.16b se ve la comparación con la segunda estación comercial (Davis 2). Con este transductor, en ambos casos se ajustan mejor los datos con valores intermedios. La comparación con el prototipo lleva signos contrarios entre los valores bajos y los altos. La mejor comparación con el prototipo ocurre al restar 1023 a los datos (ver Figuras 5.16e).

5.7.7. Dirección de viento

Las estaciones comerciales del modelo usado entregan los datos en 16 categorías que corresponden a los puntos cardinales, los rumbos laterales y los rumbos colaterales. Cada categoría cubre 22,5 grados. Los datos de las estaciones comerciales fueron transformados de su dirección a un valor en grados. El norte corresponde a 0°, el Nornoreste a 22,5° y así sucesivamente. Los datos del prototipo fueron procesados de forma que cada categoría tenga su valor en el centro del rango. Es decir, se tomaron como pertenecientes a la categoría todos los valores que discrepan hasta 11,25 grados. Por ejemplo, el norte (N) corresponde a cero grados, pero se tomó como norte a los ángulos entre -11,25° (348,75°) y 11,25°, si son mayores a 11,25 y hasta 33,75° corresponde a Nornoreste (NNE) y así sucesivamente. El error se tomó como cero si el dato se encuentra en este rango. Es decir, si la referencia es el norte, -12° tiene un error de -0,75° y 12° tiene un error de 0,75. Para este caso los errores presentan tendencias opuestas y en el caso en que se usa el prototipo la MAD es mucho mayor y el MSE es mayor.

La Figura 5.17 muestra las comparaciones de la dirección de viento. La Figura 5.17a muestra la comparación entre la estación de referencia y la segunda estación comercial. Se ve que los datos tienen una baja correlación y se agrupan mayormente en valores cercanos al norte. La Figura 5.17b corresponde a la comparación entre el dispositivo de referencia y el prototipo. En este caso la correlación es menor al de la comparación con la segunda estación comercial. Los datos en la estación de referencia se encuentran mayormente entre valores de 50 y 100 grados, pero en el prototipo están más dispersos. Se pudo comprobar que este error se debe al conector

Dirección de viento	Grados medidos
Norte	359,5667
Este	90,75422
Sur	101,522
Oeste	272,2899
Noreste	45,37029
Sureste	136,1381
Suroeste	226,906
Noroeste	321,1649

Tabla 5.5: Grados medidos por el dispositivo para cada dirección de viento.

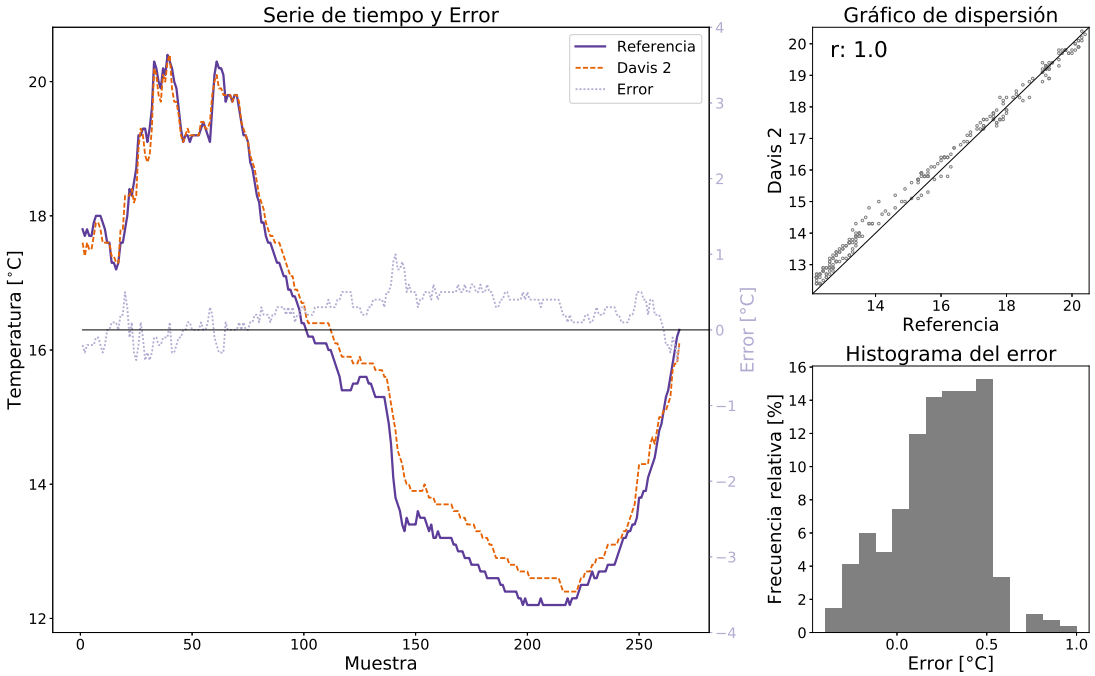
del transductor.

El error en los datos se produjo por un problema en el mecanismo de sujeción del conector RJ11 que impidió que se fije adecuadamente y provocó evito el contacto correcto. Sin embargo, se realizaron pruebas manuales para confirmar el funcionamiento adecuado del sensor. Se usó comprobó el valor de los puntos cardinales y las direcciones laterales ubicando el sensor aproximadamente en esas direcciones. En la Figura 5.18 se pueden ver dos conectores. El conector mostrado en la Figura 5.18a está completo y funciona adecuadamente, mientras que el conector de la Figura 5.18b es el del transductor del anemómetro.

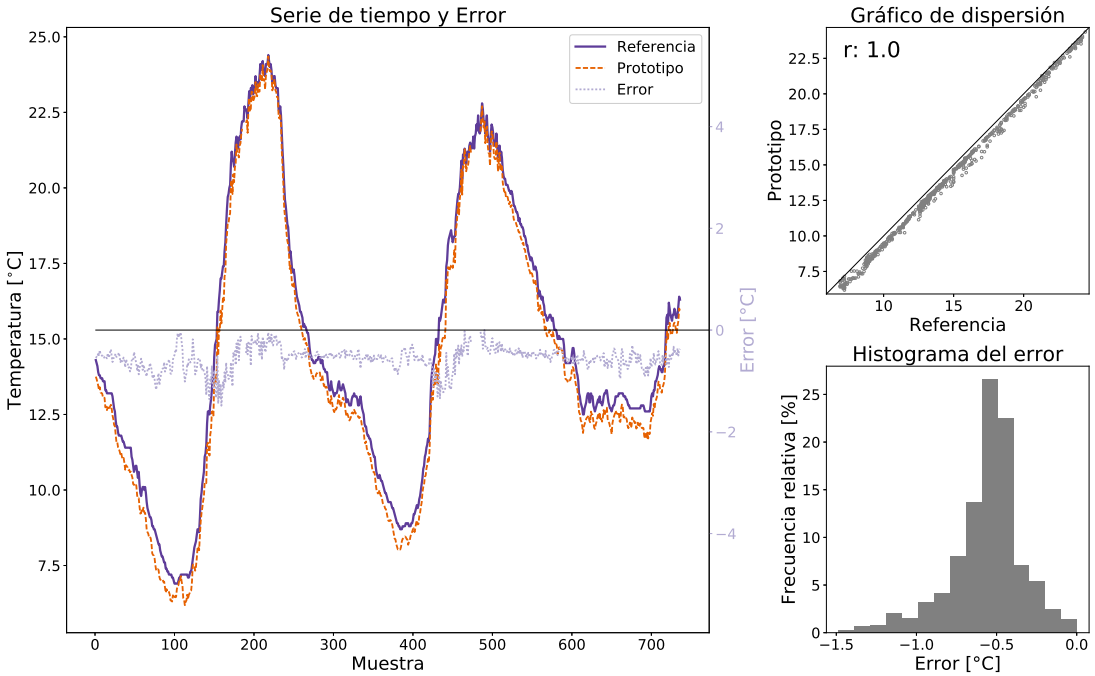
Se realizaron nuevas pruebas con el fin de demostrar el correcto funcionamiento del prototipo respecto a la medición de esta variable. Se conectó el transductor de dirección de viento y se usó el modo depurador para obtener los datos en la pantalla. Las fotografías tomadas del transductor junto al monitor donde se registran los datos se encuentran en el Anexo G. Se tomaron fotografías con la veleta dirigida hacia los puntos cardinales y los puntos colaterales y se muestran en la Figura 5.19. La Tabla 5.5 muestra los grados medidos con mayor frecuencia en cada dirección. Se encuentran primero los puntos cardinales y luego los rumbos laterales.

Las pruebas realizadas mostraron que el prototipo mide la dirección correctamente, por lo tanto, el error se debía al conector.

Se puede concluir que los datos recolectados muestran que el prototipo funciona de forma aceptable y comparable a la estación comercial. En el caso del transductor de dirección de viento se presentó una discrepancia. Se pudo comprobar que se debía a una falla en el conector de dicho transductor, y se realizaron otras pruebas que demostraron que la adquisición de datos es correcta.

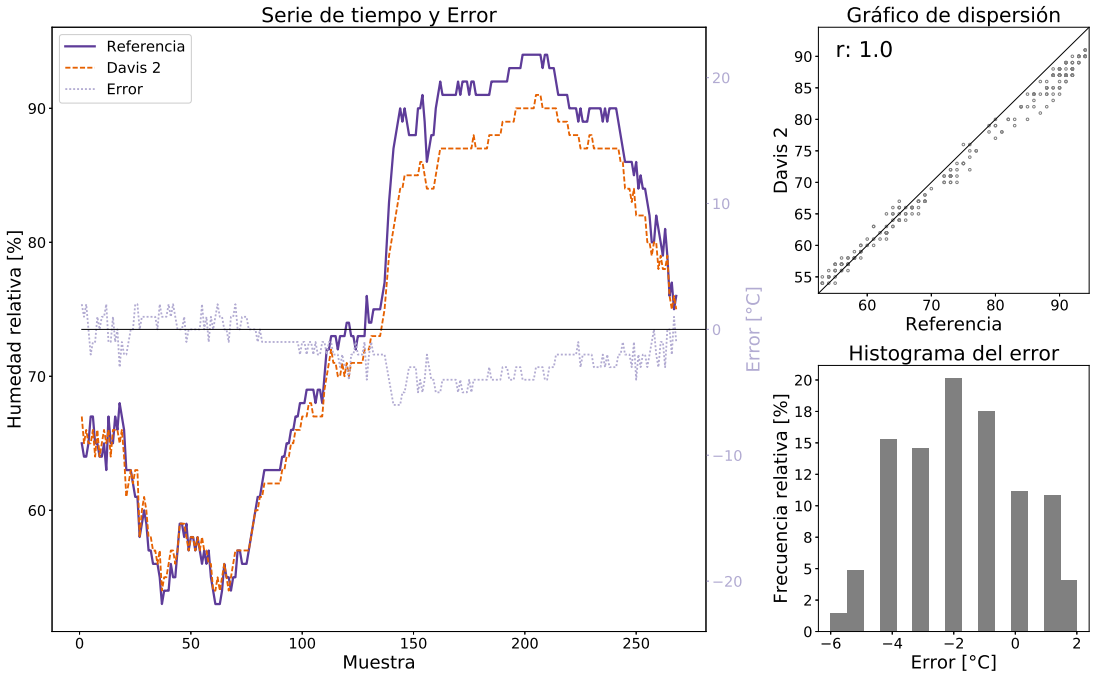


(a) Comparación entre los datos de la estación de referencia y los de la segunda estación comercial (Davis 2).

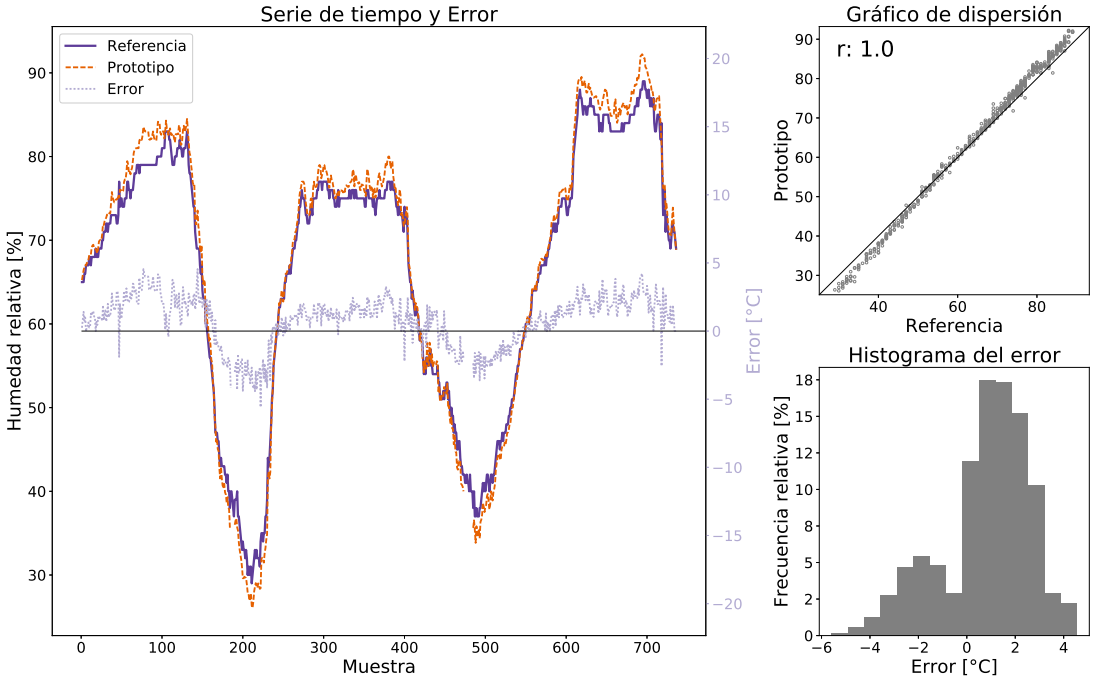


(b) Comparación entre los datos de la estación de referencia y los del prototipo.

Figura 5.11: Comparaciones de datos recolectados para la variable de temperatura. En (a) y (b): en el gráfico de serie de tiempo y error (izquierda) el eje de las ordenadas izquierda indica la temperatura y el eje derecho indica el error, el gráfico de dispersión (superior derecha) muestra el coeficiente de correlación de Pearson (medida de la relación lineal) y el histograma de los errores (inferior derecha) muestra las frecuencias relativas.

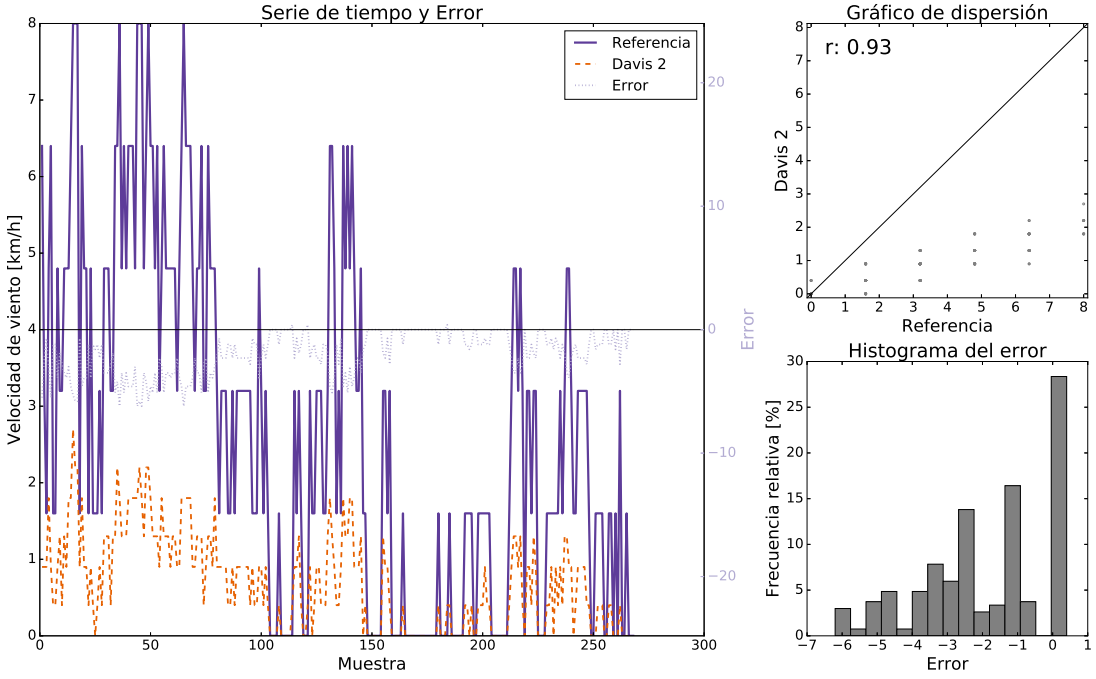


(a) Comparación entre los datos de la estación de referencia y los de la segunda estación comercial (Davis 2)

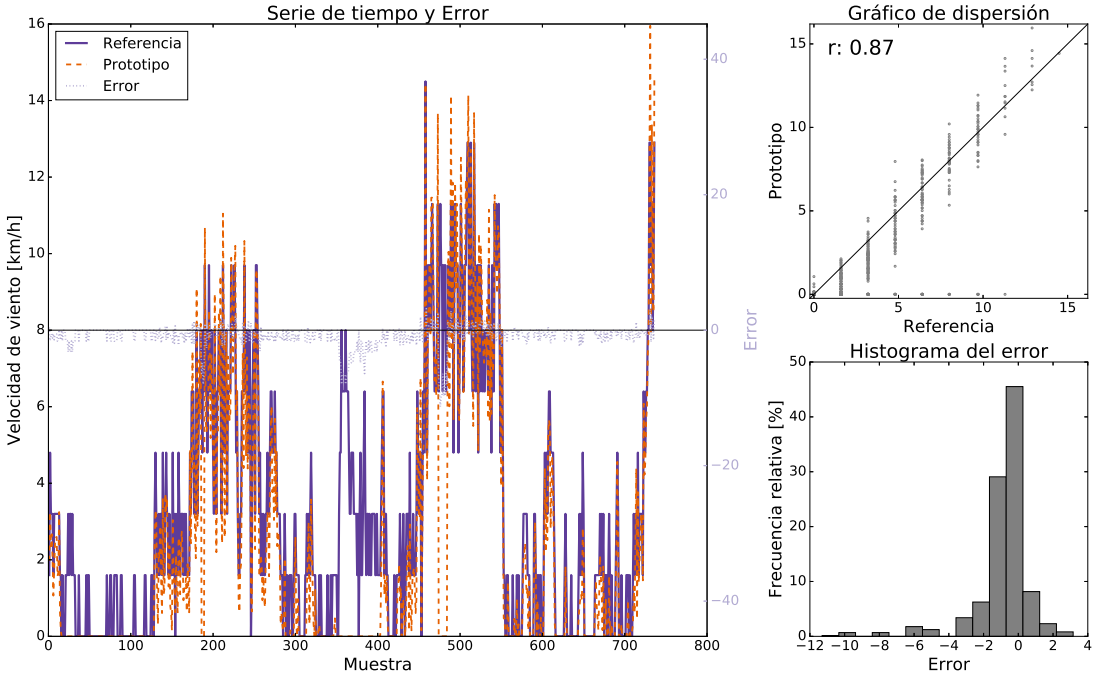


(b) Comparación entre los datos de la estación de referencia y los del prototipo

Figura 5.12: Comparaciones de datos recolectados para la variable de humedad relativa. En (a) y (b): en el gráfico de serie de tiempo y error (izquierda) el eje de las ordenadas izquierda indica la humedad relativa y el eje derecho indica el error, el gráfico de dispersión (superior derecha) muestra el coeficiente de correlación de Pearson (medida de la relación lineal) y el histograma de los errores (inferior derecha) muestra las frecuencias relativas.

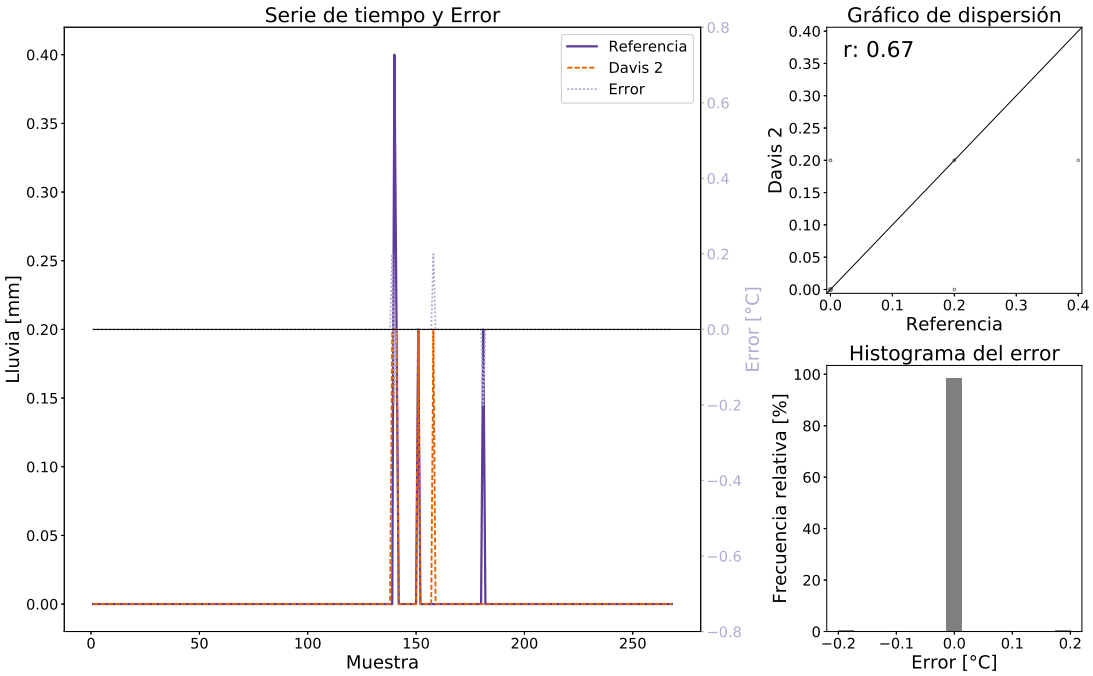


(a) Comparación entre los datos de la estación de referencia y los de la segunda estación comercial (Davis 2)

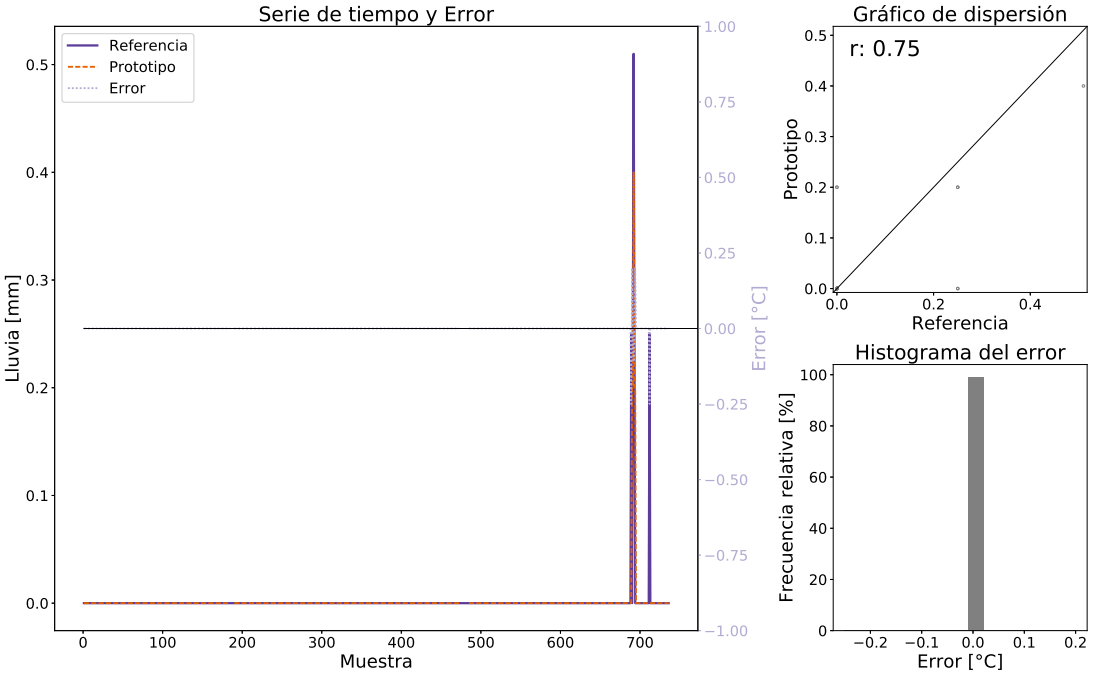


(b) Comparación entre los datos de la estación de referencia y los del prototipo

Figura 5.13: Comparaciones de datos recolectados para la variable de velocidad de viento. En (a) y (b): en el gráfico de serie de tiempo y error (izquierda) el eje de las ordenadas izquierda indica la velocidad de viento y el eje derecho indica el error, el gráfico de dispersión (superior derecha) muestra el coeficiente de correlación de Pearson (medida de la relación lineal) y el histograma de los errores (inferior derecha) muestra las frecuencias relativas.

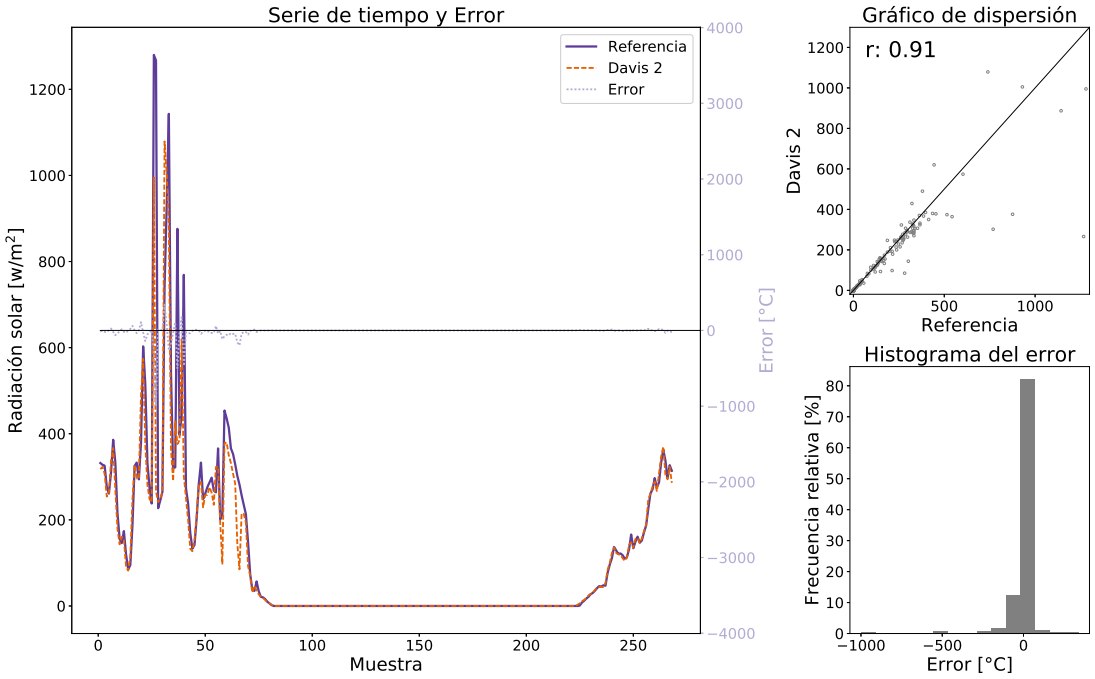


(a) Comparación entre los datos de la estación de referencia y los de la segunda estación comercial (Davis 2)

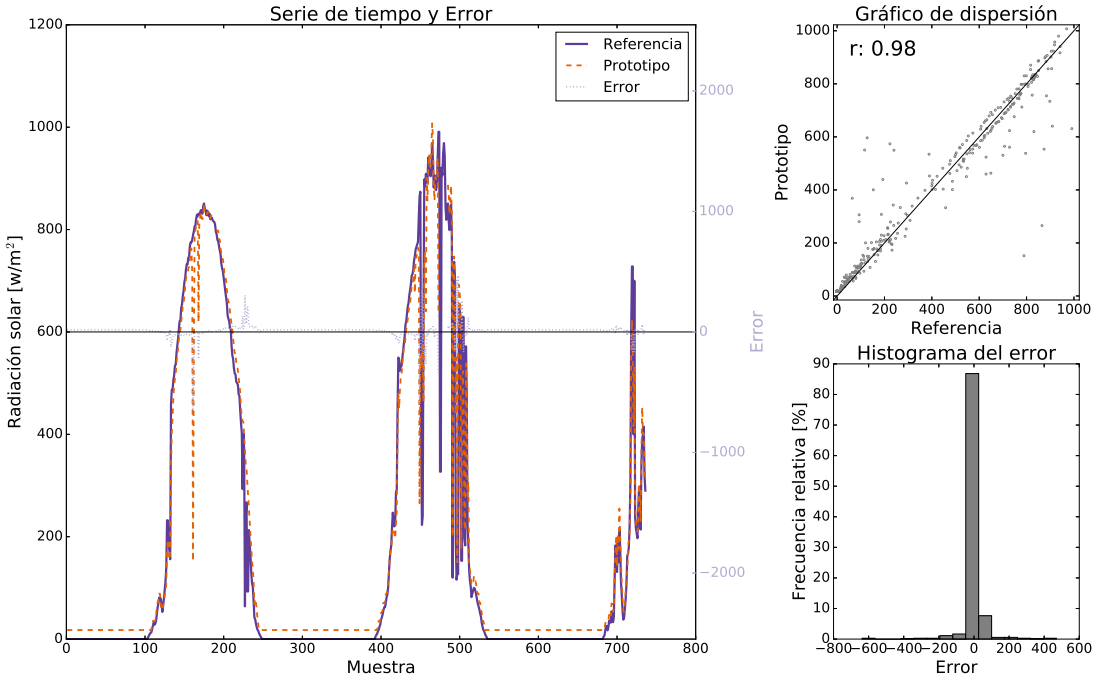


(b) Comparación entre los datos de la estación de referencia y los del prototipo

Figura 5.14: Comparaciones de datos recolectados para la variable de cantidad de lluvia. En (a) y (b): en el gráfico de serie de tiempo y error (izquierda) el eje de las ordenadas izquierda indica la cantidad de lluvia y el eje derecho indica el error, el gráfico de dispersión (superior derecha) muestra el coeficiente de correlación de Pearson (medida de la relación lineal) y el histograma de los errores (inferior derecha) muestra las frecuencias relativas.

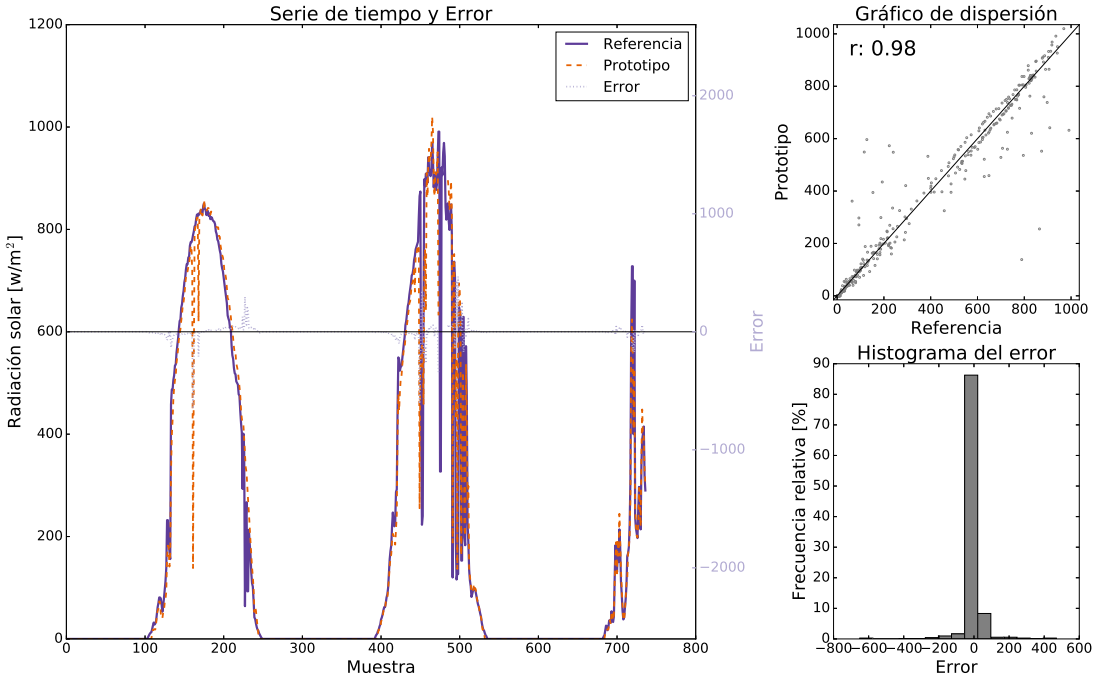


(a) Comparación entre los datos de la estación de referencia y los de la segunda estación comercial (Davis 2)



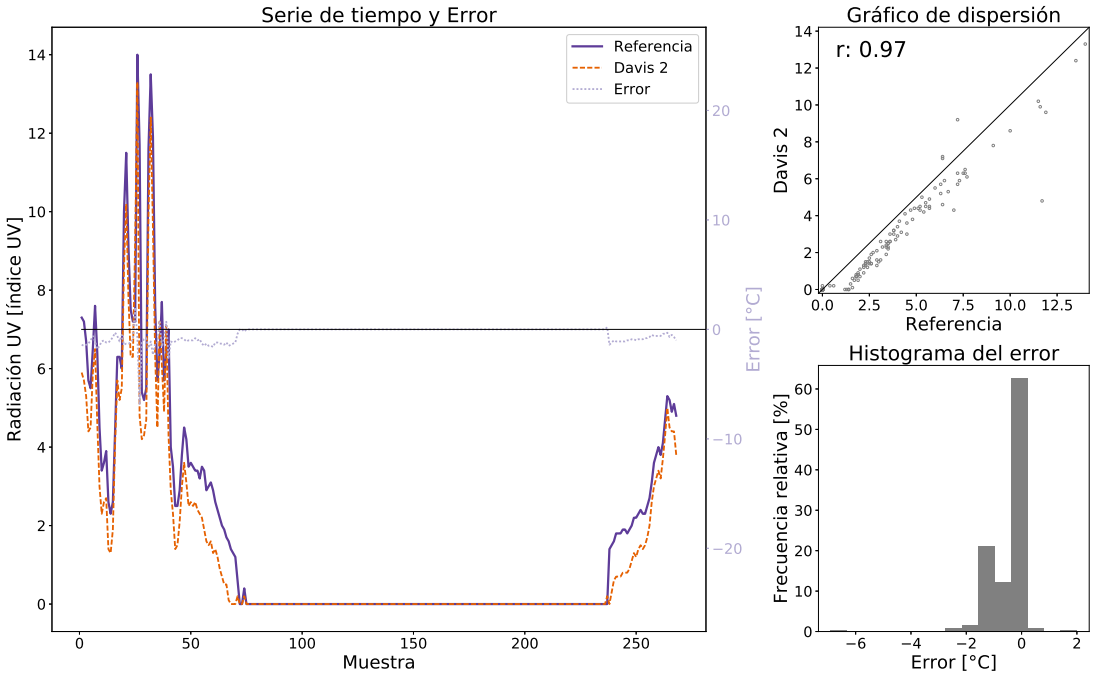
(b) Comparación entre los datos de la estación de referencia y los del prototipo

Figura 5.15: Comparaciones de datos recolectados para la variable de radiación solar. En (a), (b) y (c): en el gráfico de serie de tiempo y error (izquierda) el eje de las ordenadas izquierda indica la radiación solar y el eje derecho indica el error, el gráfico de dispersión (superior derecha) muestra el coeficiente de correlación de Pearson (medida de la relación lineal) y el histograma de los errores (inferior derecha) muestra las frecuencias relativas.

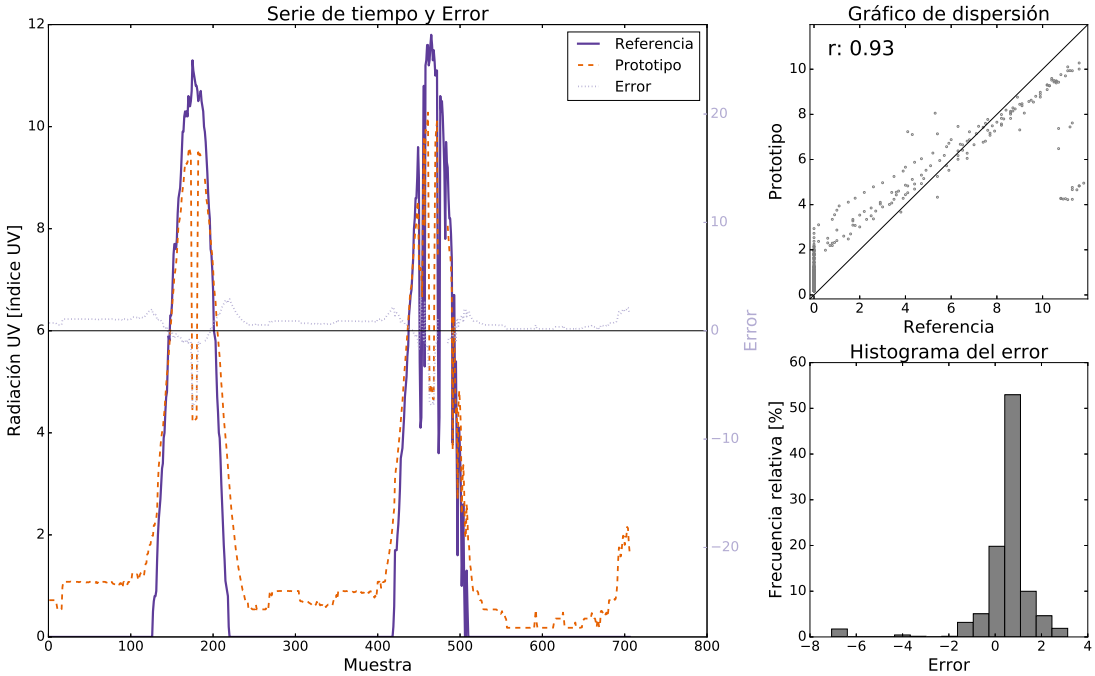


(c) Comparación entre los datos de la estación de referencia y los del prototipo restado 255 w/m^2

Figura 5.15: Comparaciones de datos recolectados para la variable de radiación solar. En (a), (b) y (c): en el gráfico de serie de tiempo y error (izquierda) el eje de las ordenadas izquierda indica la radiación solar y el eje derecho indica el error, el gráfico de dispersión (superior derecha) muestra el coeficiente de correlación de Pearson (medida de la relación lineal) y el histograma de los errores (inferior derecha) muestra las frecuencias relativas.

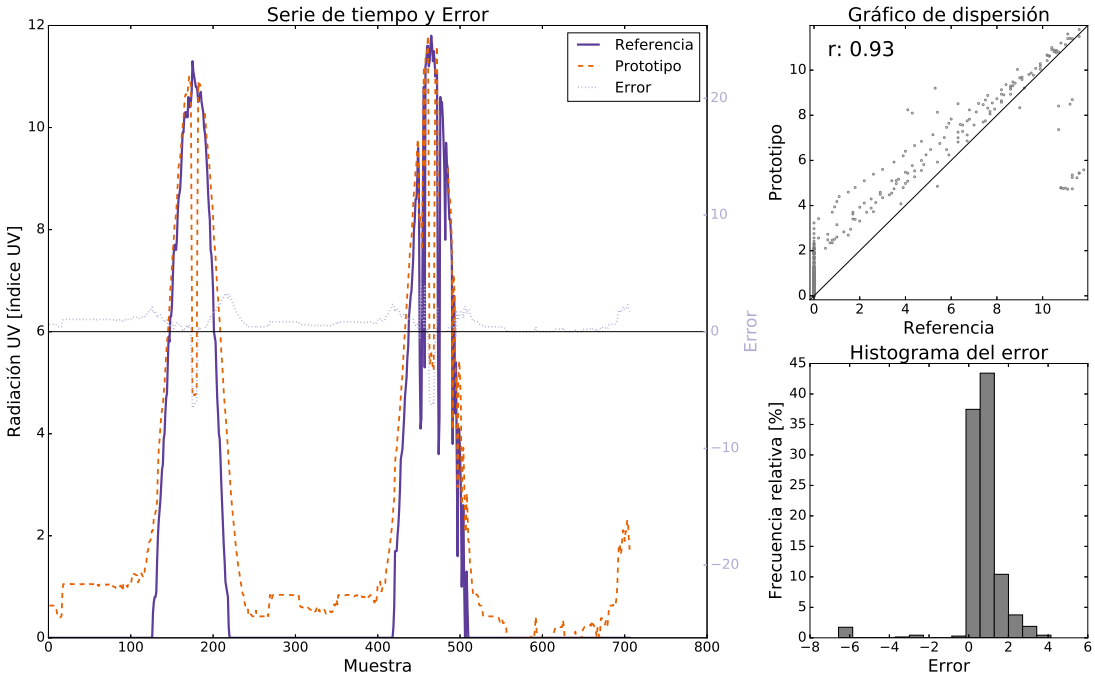


(a) Comparación entre los datos de la estación de referencia y los de la segunda estación comercial (Davis 2)

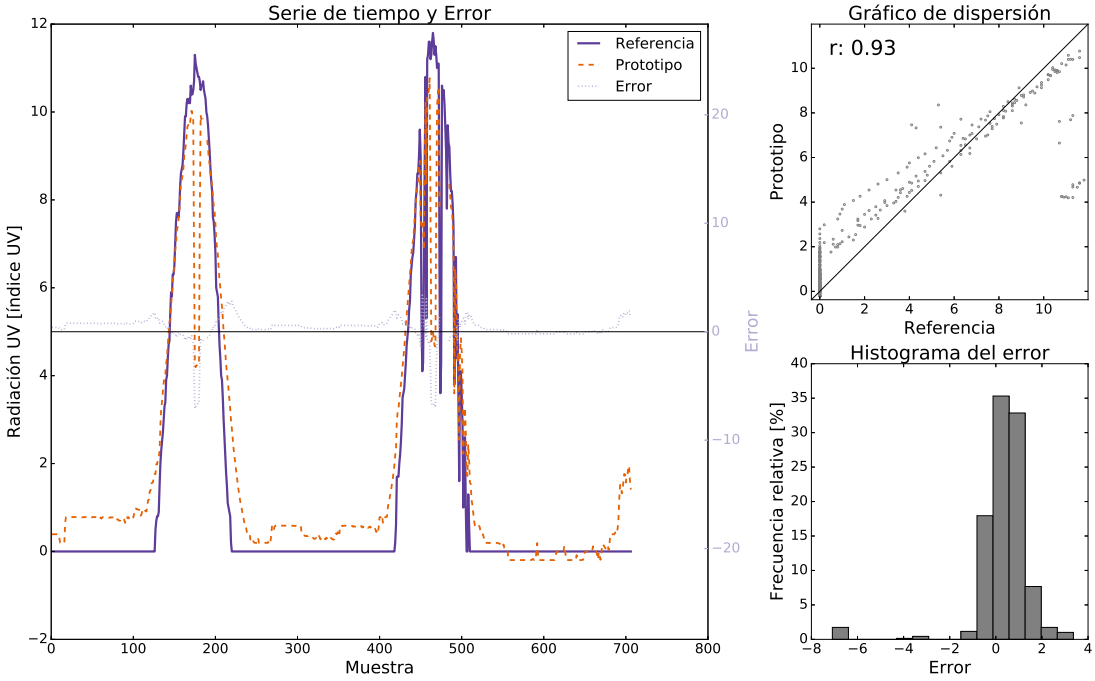


(b) Comparación entre los datos de la estación de referencia y los del prototipo

Figura 5.16: Comparaciones de datos recolectados para la variable de radiación UV. En (a)-(e): en el gráfico de serie de tiempo y error (izquierda) el eje de las ordenadas izquierda indica la radiación UV y el eje derecho indica el error, el gráfico de dispersión (superior derecha) muestra el coeficiente de correlación de Pearson (medida de la relación lineal) y el histograma de los errores (inferior derecha) muestra las frecuencias relativas.

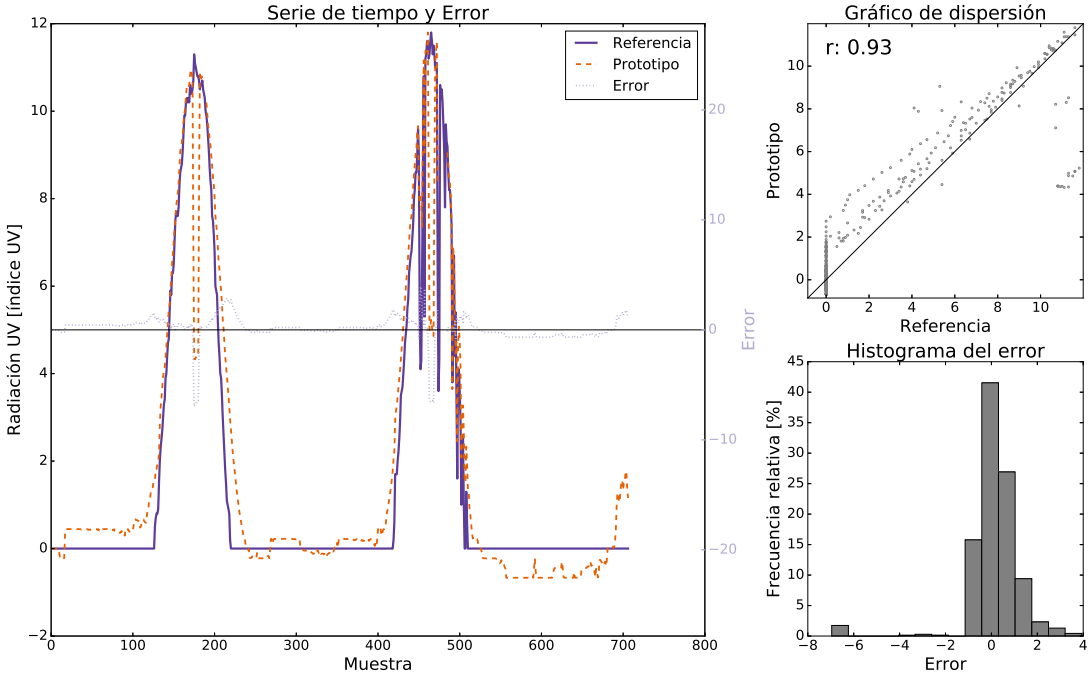


(c) Comparación entre los datos de la estación de referencia y los del prototipo restado 255



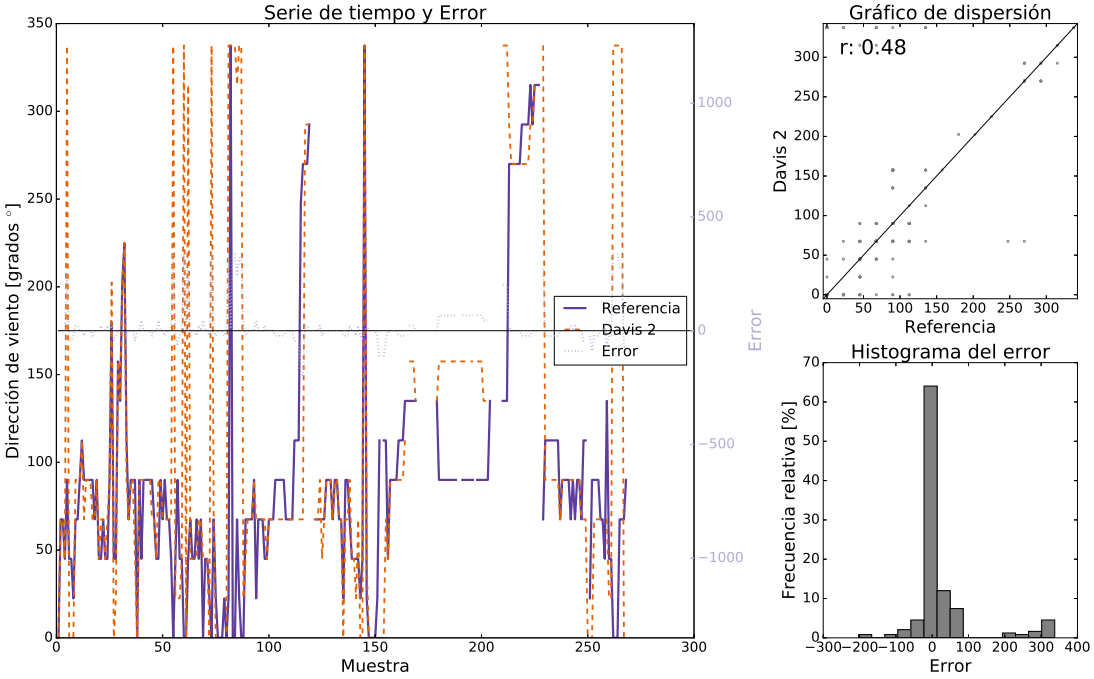
(d) Comparación entre los datos de la estación de referencia y los del prototipo restado 511

Figura 5.16: Comparaciones de datos recolectados para la variable de radiación UV. En (a)-(e): en el gráfico de serie de tiempo y error (izquierda) el eje de las ordenadas izquierda indica la radiación UV y el eje derecho indica el error, el gráfico de dispersión (superior derecha) muestra el coeficiente de correlación de Pearson (medida de la relación lineal) y el histograma de los errores (inferior derecha) muestra las frecuencias relativas.

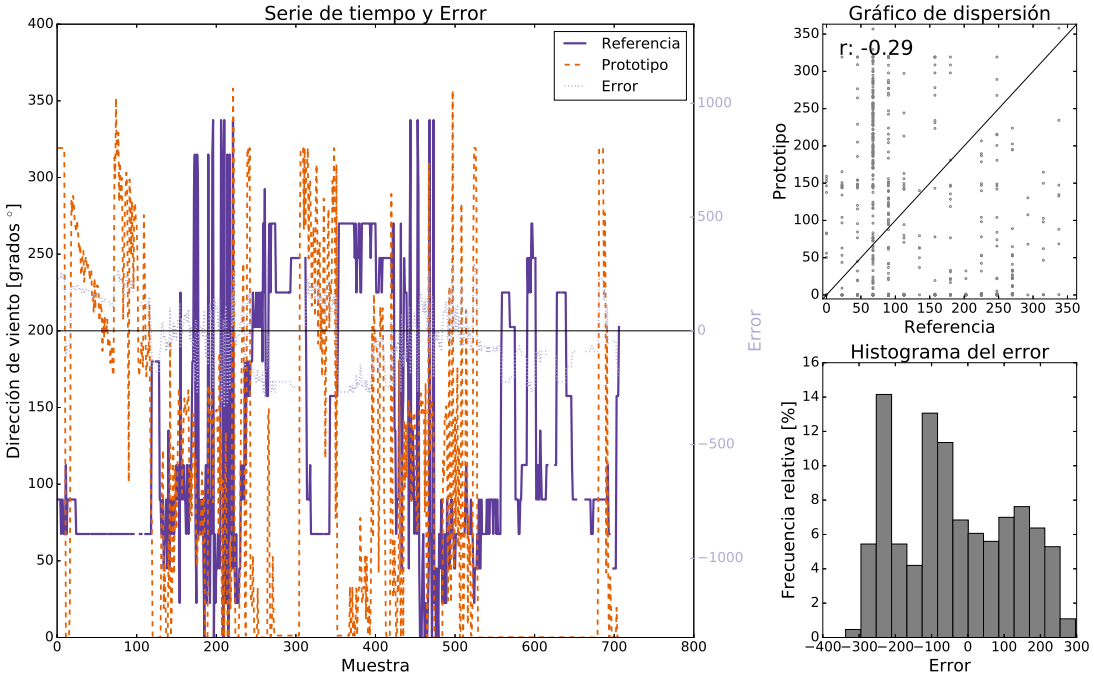


(e) Comparación entre los datos de la estación de referencia y los del prototipo restado 1023

Figura 5.16: Comparaciones de datos recolectados para la variable de radiación UV. En (a)-(e): en el gráfico de serie de tiempo y error (izquierda) el eje de las ordenadas izquierda indica la radiación UV y el eje derecho indica el error, el gráfico de dispersión (superior derecha) muestra el coeficiente de correlación de Pearson (medida de la relación lineal) y el histograma de los errores (inferior derecha) muestra las frecuencias relativas.

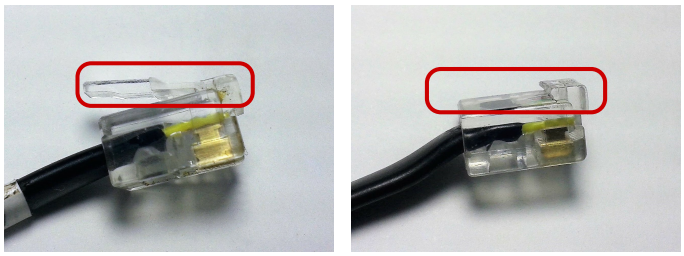


(a) Comparación entre los datos de la estación de referencia y los de la segunda estación comercial (Davis 2)



(b) Comparación entre los datos de la estación de referencia y los del prototipo

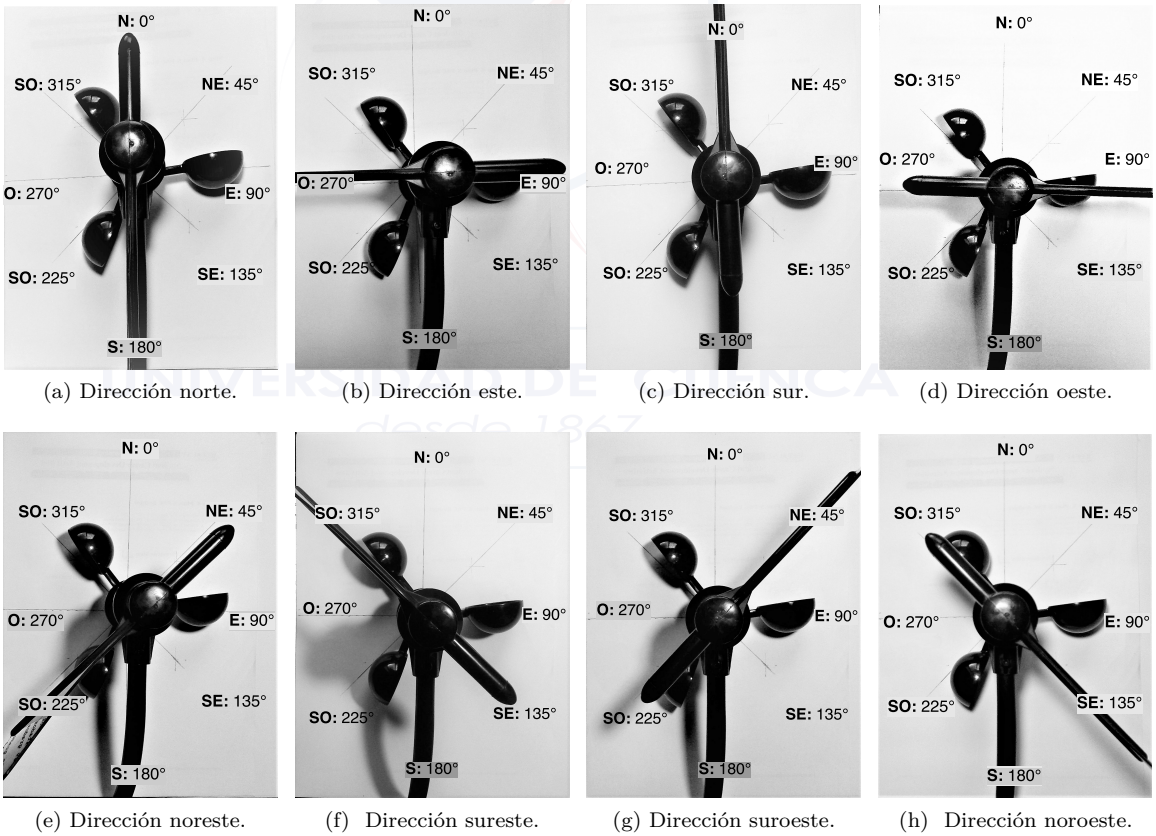
Figura 5.17: Comparaciones de datos recolectados para la variable de dirección de viento. En (a) y (b): en el gráfico de serie de tiempo y error (izquierda) el eje de las ordenadas izquierda indica la dirección de viento y el eje derecho indica el error, el gráfico de dispersión (superior derecha) muestra el coeficiente de correlación de Pearson (medida de la relación lineal) y el histograma de los errores (inferior derecha) muestra las frecuencias relativas.



(a) Conector RJ11 con su mecanismo de sujeción.

(b) Conector RJ11 que ha perdido su mecanismo de sujeción. Este conector corresponde al transductor del anemómetro

Figura 5.18: Conectores RJ11. En (a): Conector completo, con su mecanismo de sujeción. En (b): Conector del transductor del anemómetro que ha perdido su mecanismo de sujeción.



(a) Dirección norte. (b) Dirección este. (c) Dirección sur. (d) Dirección oeste.

(e) Dirección noreste. (f) Dirección sureste. (g) Dirección suroeste. (h) Dirección noroeste.

Figura 5.19: Orientación de la veleta del transductor de dirección de viento apuntando a diferentes orientaciones. (a): Norte. (b): Este. (c): sur. (d): Oeste. (e): Noreste. (f): Sureste. (g): Suroeste. (h): Noroeste.

5.8. Sistema Operativo de Tiempo Real

En la revisión de datos registrados en la memoria SD (Sección 5.7), se descubrió que cada 24 horas existía perdidas de datos por varios minutos, periodo que coincide con la ejecución del hilo dedicado a la actualización del RTC mediante el GPS (*HiloSincGPS*). Este hilo bloqueaba la ejecución del resto de hilos hasta obtener una señal fija del GPS y sincronizar el RTC, lo cual es correcto al iniciar el sistema, ya que antes de almacenar las mediciones de los transductores, es necesario obtener una referencia de tiempo para su registro, sin embargo, las posteriores ejecuciones de este hilo debían realizarse de forma paralela, sin bloquear el resto de hilos, puesto que ya se tiene una referencia de tiempo, y solo se desea actualizar el RTC para asegurar la precisión del mismo.

Se pudo identificar que en el hilo *HiloSincGPS*, la función textitOS_Delay() para evitar bloquear el resto de tareas era llamada a través de una segunda función. Esta llamada indirecta no tenía efecto. Para resolver esta situación se programó la función dentro de la tarea. Para corroborar el funcionamiento se realizaron pruebas comprobando que la tarea *HiloSincGPS* ya no bloquea al resto de tareas.

Para controlar los periodos de ejecución de cada hilo, se utiliza como referencia el temporizador del RTOS. Sin embargo, mediante el modo depurador se pudo notar que éstos periodos no eran exactos, y que cada 40 o 50 segundos se generaba un retraso de 1 segundo. Para validar la precisión del temporizador del RTOS de 1ms establecido mediante interrupción del Timer 0, se generó una señal digital mediante la negación de un pin en cada interrupción. El calculo de los valores establecidos en los registros de control del Timer 0 se muestran en la Sección 4.4.3.

La precisión del módulo Timer 0 depende de su señal de reloj, la cual puede ser el oscilador del mismo procesador, o una fuente externa mediante el pin `RA4`. El temporizador hace uso del oscilador interno del microcontrolador configurado a 16 MHz. De acuerdo a la Tabla C.2 el oscilador a esta frecuencia presenta un error del $\pm 2\,\%$. Para corregir la base de tiempo del RTOS se modificó la interrupción del Timer 0 en base a prueba y error hasta conseguir un marcado exacto de 1ms, en donde el temporizador ha sido configurado sin preescalador, obteniendo así un incremento en cada conteo de $t_{ick} = 0,25\,\mu s$ (Ecuación 4.6), realizando 3960 conteos, lo cual equivale a $3960 \cdot 0,25\ \mu s = 0,99\ ms$. De esta manera también se corrigió la precisión de los periodos de ejecución de cada hilo en el RTOS.

Capítulo 6

Conclusiones y Recomendaciones

En este capítulo se encuentran varias conclusiones a las que se llegaron luego de diseñar, construir y probar el prototipo, y elaborar el trabajo en general. También se dan recomendaciones para desarrollar trabajos similares y se indican tareas a futuro que se puede realizar para mejorar el presente trabajo.

6.1. Conclusiones

Con el fin de conocer el mecanismo de funcionamiento de cada sensor de la estación Davis 6162 Wireless Vantage Pro2 Plus, se consultaron sus manuales y se analizaron y probaron las terminales hasta llegar a determinar la forma adecuada de conexión. Del análisis realizado se pudo observar que las señales de los transductores están acondicionadas de forma que se requieren pocos elementos adicionales para realizar mediciones.

El diseño se probó en una placa inicial con componentes de inserción (*Through hole*), más fáciles de conseguir en el medio local. Luego de probar y validar satisfactoriamente el circuito, se procedió a enviar un diseño con componentes en su mayoría de montaje superficial para su manufactura en una empresa especializada local. Este diseño también fue probado posteriormente con resultados similares. Se logró que el dispositivo sea de bajo consumo y de un tamaño reducido, debido a que se usaron componentes de montaje superficial en la mayoría de los elementos.

Los componentes del circuito, especialmente los que funcionan a alta frecuencia, como el GPS y el XBee, introducen ruido en las señales debido a los cambios rápidos de corriente que exigen, por lo que fue necesario identificar el ruido y diseñar un filtro. También se realizó un promedio entre las muestras con el fin de disminuir el ruido aleatorio en la señal.

El uso de un RTOS en el microcontrolador facilitó la programación y ejecución de las tareas paralelas y redujo el tiempo de ejecución de las mismas. El RTOS permitió la ejecución de varios hilos al mismo tiempo de forma independiente. De esta forma fue posible programar hilos dedicados a diferentes funciones del dispositivo y a controlar los tiempos de ejecución para que los intervalos correspondan a los periodos de muestreo que se han configurado para cada sensor.

El contenedor diseñado estuvo expuesto a la intemperie por hasta 3 días, en donde se pudo comprobar su resistencia al ingreso de agentes externos que pueden afectar el normal funcionamiento del prototipo, tales como polvo, humedad y lluvia. Sin embargo, para garantizar su tropicalización, se debe realizar pruebas con periodos mas largos. Con respecto a los conectores en las tapas del contenedor, y los que se encuentran en la placa, estos deben ser reubicados para acoplarse de mejor manera, y así permitir un acceso mas fácil a los componentes internos para su revisión, mantenimiento, etcétera.

Con respecto a la memoria SD, el proceso de extraer o introducir la misma en el contenedor resulta incomodo debido a la necesidad de abrirlo para acceder a la memoria.

Los datos recolectados por la estación tenían variaciones pequeñas que no alteran la tenden-

cia de la señal; pueden ser causadas por ruido aleatorio introducido por el circuito o la fuente. El promedio que se realiza en el muestreo redujo en parte estas variaciones y permitió obtener la señal más limpia. También el promedio que se realizó en el envío, cada cinco minutos, redujo las variaciones que se pudieron haber producido debido a que se toman muestras cada 10 segundos.

Las pruebas realizadas muestran que:

Las observaciones reportadas por el prototipo son comparables a las de la estación comercial de referencia. La correlación entre los datos de referencia y la estación de prueba es alta, incluso llega a ser uno en las variables temperatura y humedad relativa con un error muy bajo. En el caso de la temperatura existe una subestimación de aproximadamente 0.5 °C que se mantiene aproximadamente constante; una posible solución es realizar una calibración que sume 0.5 °C a las observaciones para disminuir el error y mantener la correlación lineal (r=1). En relación a la humedad relativa, la Figura 5.12 da evidencia de una sobrestimación en los valores altos y una subestimación en los valores bajos. Una posible solución podría ser dar un tratamiento diferenciado a los valores altos y bajos para mantener la correlación lineal (r=1) pero disminuyendo el error total.

Para el caso del sensor de velocidad de viento, el error es menor en la comparación que usa el prototipo respecto a la comparación entre estaciones comerciales, pero la correlación es un poco menor. Para el caso de la radiación solar, ambas comparaciones tienen una correlación similar, e incluso la frecuencia del error es mayor cerca a cero. En el caso de la radiación UV, se puede notar que las mediciones son parecidas y el error es mayormente cero. El pluviómetro fue probado manualmente y se pudo comprobar su funcionamiento correcto. En relación al sensor de dirección de viento, su correcto funcionamiento se comprobó de forma manual, debido a un daño en el mecanismo de sujeción del sensor que no permitió la correcta adquisición de datos de su señal en el prototipo.

6.2. Recomendaciones

En este trabajo se usó el oscilador interno del microprocesador, sin embargo, es preferible el uso de un oscilador externo de mayor calidad para obtener una frecuencia más estable y exacta.

El diseño del contenedor actual deja expuestos los conectores que con el tiempo requerirán mantenimiento. Con el fin de aumentar el periodo de mantenimiento se puede rediseñar el contenedor de forma que los conectores queden aislados del ambiente.

Puesto que, en la mayoría de los sensores, entre las muestras consecutivas hay una variación mínima o ninguna, se puede armar la trama de transmisión con las diferencias, que son menores

al valor total. De esta forma se requieren menos bits para representar los números, lo que cual reducirá la carga de transmisión en el dispositivo XBee.

Por economizar energía, los datos son almacenados en un buffer temporal en la memoria RAM del microcontrolador reduciendo el número de veces que se requiere escribir en la SD . Sin embargo, esto limita la cantidad de datos que se pueden manejar y, por lo tanto el periodo máximo de transmisión. Se recomienda el uso de una memoria RAM externa para poder configurar periodos de transmisión mayores.

6.3. Trabajos futuros

No enviar todos los datos, solo el máximo, el mínimo y el promedio como en el caso de la estación Davis Vantage Pro2, pero guardar todos los datos en la memoria SD.

El sistema debería realizar las transmisiones coincidiendo con un inicio a las cero horas y se debería validar que el periodo de envío sea mayor al de muestreo.

Diseñar el colector y la placa de forma que la memoria SD quede más fácilmente accesible y no sea necesario desarmar el contenedor para sacarla.

Bibliografía

[1] F. E. Valdés Perez y R. Areny Pallas, *Microcontroladores: fundamentos y aplicaciones con PIC*, ser. Alfaomega. Marcombo. Marcombo, 2007. [En línea]. Disponible: https://books.google.com.ec/books?id=ODenKGOHMRkC

[2] D. Gislason, *Zigbee Wireless Networking*. Elsevier Science, 2008. [En línea]. Disponible: https://books.google.com.ec/books?id=up8Oa7456I8C

[3] S. Kuo, B. Lee, y W. Tian, *Real-Time Digital Signal Processing: Implementations and Applications*. Wiley, 2006. [En línea]. Disponible: https://books.google.com.ec/books?id=QIj9Pthp_T8C

[4] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, y others, "Luster: wireless sensor network for environmental research," in *Proceedings of the 5th international conference on Embedded networked sensor systems*. ACM, 2007, pp. 103–116. [En línea]. Disponible: http://dl.acm.org/citation.cfm?id=1322274

[5] Sensirion, "Datasheet SHT1x (SHT10, SHT11, SHT15) Humidity and Temperature Sensor IC," 2010. [En línea]. Disponible: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/2_Humidity_Sensors/Sensirion_Humidity_Sensors_SHT1x_Datasheet_V5.pdf

[6] T. Instruments, "ADS111x-Q1 Automotive, Low-Power, I2c-Compatible, 860-SPS, 16-Bit ADCs With Internal Reference, Oscillator, and Programmable Comparator," 2016. [En línea]. Disponible: http://www.ti.com/lit/ds/symlink/ads1115-q1.pdf

[7] D. Mihajlovic, "Muy bien. Lo que necesitamos ahora es un ... Sistema GPS," Abr. 2009. [En línea]. Disponible: http://download.mikroe.com/documents/articles/spa/featured-articles/gps/pic/elektor-es-article-mikrobasic-pic-04-09.pdf

[8] F. Semiconductors, *FSA3357 Low Voltage SP3T Analog Swithc (3:1 Multiplexer/Demultiplexer*.

[9] H. W. Ott, "Partitioning and Layout of a Mixed-Signal PCB," Jun. 2001. [En línea]. Disponible: http://www.hottconsultants.com/pdf_files/june2001pcd_mixedsignal.pdf

[10] MICROCHIP, "PIC18f23k20/24k20/25k20/26k20/43k20/44k20/45k20/46k20 Data Sheet," 2010. [En línea]. Disponible: http://ww1.microchip.com/downloads/en/DeviceDoc/41303G.pdf

[11] A. S. Tanenbaum, *Redes de computadoras*. Pearson Educación, 2003.

[12] "Misión & Visión." [En línea]. Disponible: http://promas.ucuenca.edu.ec/Promas/index.php/mision-vision.html

[13] "Proyecto: Aplicación de Tecnologías Inalámbricas al Monitoreo Climatológico en la Cuenca del Río Paute - Universidad de Cuenca." [En línea]. Disponible: http://www.ucuenca.edu.ec/la-oferta-academica/oferta-de-grado/facultad-de-ingenieria/dptos/dcc/proyectos-dcc/2282-proyecto-aplicacio%CC%81n-de-tecnologi%CC%81as-inala%CC%81mbricas-al-monitoreo-climatolo%CC%81gico-en-la-cuenca-del-ri%CC%81o-paute/

[14] "Predicción meteorológica | WMO for Youth." [En línea]. Disponible: http://www.wmo.int/youth/es/met-subpages/predicci%C3%B3n-meteorol%C3%B3gica

[15] O. M. Mundial, "Vocabulario meteorológico internacional mmo- no-182."

[16] ——, "Gía de prácticas climatológicas (omm-n100)."

[17] ——, "Guía de instrumentos y métodos de observación meteorológicos (omm-n° 8."

[18] "Los organismos especializados." [En línea]. Disponible: http://www.un.org/es/aboutun/uninbrief/institutions.shtml

[19] O. M. Mundial, *Reglamento Técnico Documentos Fundamentales N°2 OMM-N°49*.

[20] E. C. Fonseca, "Manual de procedimientos para las estaciones meteorológicas."

[21] O. M. Mundial, "International glossary of hydrology - glossaire international d'hydrologie-glosario hidrologico internacional."

[22] J. W. Gilmore, "medición y predicción de la radiación solar global UV-B bajo cielos claros y sin nubes," *Uniciencia*, vol. 24, num. 1, pp. 111–120, 2016. [En línea]. Disponible: http://www.revistas.una.ac.cr/index.php/uniciencia/article/view/378

[23] R. D. Mejía Garcés, "Diseño e implementación de uh dispositivo para la adquisición de datos meteorológicos y conectividad IP que se integre con la red de datos de Senagua en la cuenca del río Santa Bárbara," B.S. thesis, Universidad de Cuenca, 2016. [En línea]. Disponible: http://dspace.ucuenca.edu.ec/handle/123456789/25961

[24] D. Instruments, *User Manual*.

[25] ——, *Rain Collector II. Installation manual*, 2012. [En línea]. Disponible: http://www.davisnet.com/product_documents/weather/manuals/07395-275_IM_07852.pdf

[26] D. Intruments, *User Manual Temperature Humidity sensor with 24- Hour Solar-Powered Fan-Aspirated Radiation Shield.*

[27] D. Instruments, *Solar Radiation Sensor.*

[28] ——, *UV Sensor 6490*, 2014. [En línea]. Disponible: http://www.davisnet.com/product_documents/weather/spec_sheets/6490_SS.pdf

[29] R. Kamal, *Embedded Systems: Architecture, Programming and Design.* McGraw-Hill Education (India) Pvt Limited, 2011. [En línea]. Disponible: https://books.google.com.ec/books?id=pWlbvW0H3IAC

[30] D.-R. UK, *DHT11 Humidity & Temperature Sensor.*

[31] ATMEL, *Industry-standard 2-wire Protocol "Bit-banged" C Rountines for the AVR®Microcontroller/ISP Code for the AT17LVXXX FPGA Configuration Memories.*

[32] R. Faludi, *Building wireless sensor networks: [a practical guide to the ZigBee Mesh networking protocol]*, 1ra ed. Beijing: O'Reilly, 2011, oCLC: 706971541.

[33] F. Eady, *Hands-On ZigBee: Implementing 802.15.4 with Microcontrollers*, ser. Embedded Technology. Elsevier Science, 2010. [En línea]. Disponible: https://books.google.com.ec/books?id=_KcH0wrcavgC

[34] R. Tocci, N. Widmer, y J. Cárdenas, *Sistemas digitales: principios y aplicaciones.* Prentice-Hall Hispanoamericana, 2003. [En línea]. Disponible: https://books.google.com.ec/books?id=bmLuH0CsIh0C

[35] J. P. A. Moya, *Procesamiento Digital de Señales.* Wiley, 2011. [En línea]. Disponible: http://www.ie.itcr.ac.cr/palvarado/PDS/pds.pdf

[36] B. Baker, "How delta-sigma ADCs work, Part," *Analog Applications*, vol. 7, p. 0000000, 2011. [En línea]. Disponible: http://www.deyisupport.com/cfs-file.ashx/___key/telligent-evolution-components-attachments/13-112-00-00-00-00-34-89/_216ADF62945E28751F670A52_-2011-_2C7B094E635BA65E_.pdf#page=13

[37] "Bienvenidos a GPS.gov." [En línea]. Disponible: http://www.gps.gov/spanish.php

[38] B. Hofmann-Wellenhof, H. Lichtenegger, y E. Wasle, *GNSS – Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more.* Springer Vienna, 2007. [En línea]. Disponible: https://books.google.com.ec/books?id=Np7y43HU_m8C

[39] D. Ibrahim, *Advanced PIC Microcontroller Projects in C: From USB to RTOS with the PIC 18F Series*, ser. PIC Bundle Series. Elsevier Science, 2011. [En línea]. Disponible: https://books.google.com.ec/books?id=NPSQShtCQaUC

[40] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, y M. Welsh, "Deploying a wireless sensor network on an active volcano," *IEEE internet computing*, vol. 10, num. 2, pp. 18–25, 2006. [En línea]. Disponible: http://ieeexplore.ieee.org/abstract/document/1607983/

[41] C. E. F. Veintimilla R. Jaime, Capelo U. Patricio, "Diseño e implementación de sensores remotos de lluvia por medio de una red gsm o gprs," *Revista Tecnológica ESPOL-RTE*, 2014.

[42] "Base et augets pour pluviomètre Vantage Pro2 - 7852.804 - Davis Instruments." [En línea]. Disponible: http://www.meteo-shopping.fr/Station-meteo/Base-et-augets-pour-pluviometre-Vantage-Pro2-7852804-Davis-Instruments

[43] "Davis Weather 7342178 Wind Cups Vantage Vue ISS | ($23.85)RYDA Weather." [En línea]. Disponible: https://www.ryda.com.au/davis-weather-7342178-wind-cups-vantage-vue-iss

[44] D. Instruments, "Solar Radiation Sensor 6450," 2014. [En línea]. Disponible: http://www.davisnet.com/product_documents/weather/spec_sheets/6450_SS.pdf

[45] World Meteorological Organization, World Meteorological Organization, y Organisation Météorologique Mondiale, *Guide to meteorological instruments and methods of observation*. Geneva: WMO, 2008, oCLC: 785715260.

[46] T. Engel y P. J. Reid, *Introducción a la fisicoquímica: termodinámica*. Pearson Educación, 2007, google-Books-ID: uGC0DiP3g30C.

[47] M. Verle, "3.5 Temporizador TIMER1 | Microcontroladores PIC – Programación en C con ejemplos." [En línea]. Disponible: https://learn.mikroe.com/ebooks/microcontroladorespicc/chapter/temporizador-timer1/

[48] D. Ibrahim, "μRTOS: Simple Multitasking with Microcontrollers," *in "Electronics World*, 2010. [En línea]. Disponible: http://www.mikroe.com/downloads/get/1040/u%EE%80%80rtos%EE%80%81_mikroc_ew_06_2010.pdf

[49] "OSA : Introduction [PIC24]." [En línea]. Disponible: http://www.pic24.ru/doku.php/en/osa/ref/intro

[50] O. Abdelmalek., "DESIGN AND IMPLEMENTATION OF SMART WIRELESS SENSOR NETWORK," Ph.D. dissertation, Université de Setif, 2009.

[51] L. Ada, "Adafruit Ultimate GPS," Feb. 2017. [En línea]. Disponible: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps.pdf

[52] C. A. V. Romero, J. E. B. Jaimes, y D. C. P. González, "Parámetros de configuración en módulos XBEE-PRO® S2b ZB para medición de variables ambientales," *Revista Tecnura*, vol. 19, num. 45, pp. 141–158, 2015. [En línea]. Disponible: http://revistas.udistrital.edu.co/ojs/index.php/Tecnura/article/view/9022

[53] "XBee Pro 63mw RPSMA Series 2b (ZigBee Mesh)," Oct. 2014. [En línea]. Disponible: http://xbee.cl/xbee-pro-63mw-rpsma/

[54] J. P. L. Veraguas, *Compatibilidad electromagnética: diseño de módulos electrónicos.* Marcombo, 2006, google-Books-ID: TLTUH4sE4tMC.

[55] D. L. Terrell y R. K. Keenan, *Digital Design for Interference Specifications.* Newnes, 1997.

[56] H. Johnson y M. Graham, *High Speed Digital Design: A Handbook of Black Magic*, 1ra ed. Englewood Cliffs, N.J: Prentice Hall, Abr. 1993.

[57] M. Montrose, "Right angle corners on printed circuit board traces, time and frequency domain analysis," pp. 638–641, 1999. [En línea]. Disponible: http://ieeexplore.ieee.org/document/801409/

[58] T. Instruments, "System design guidelines for the tm4c129x family of tiva™ c series microcontrollers." [En línea]. Disponible: http://www.ti.com/lit/an/spma056/spma056.pdf

[59] J. P. L. Veraguas, *Compatibilidad electromagnética y seguridad funcional en sistemas electrónicos.* Marcombo, Mar. 2010, google-Books-ID: mOiwM97Zw5QC.

[60] M. García, "Estudio de la influencia de la concentración de nanotubos de carbono de pared simple sobre la resistencia al rayado en polímeros de polimetilmetacrilato," Ingeniero Técnico Industrial, Universidad Politécnica de Cartagena, Cartagena, España, Sep. 2012. [En línea]. Disponible: http://repositorio.upct.es/bitstream/handle/10317/2879/pfc4424.pdf;jsessionid=3DBAA0850C18431D8B66FA8C71F35C04?sequence=1

[61] "CMX Systems - Small and Fast Real-Time Software." [En línea]. Disponible: http://cmx.com/

[62] "CMX Systems, Inc." [En línea]. Disponible: http://www.cmx.com/spec-tiny.htm

[63] C. Y. Wong, M. F. Zakaria, y H. Mhd Poad, "Real time operating system for mobile robot using PICos18," 2012. [En línea]. Disponible: http://eprints.uthm.edu.my/6136/

[64] Pragmatec, *PICos18 RealTime. Kernel for PIC18. Kernel Interface API Version 2.xx.*

[65] "Real-Time Kernels | Micrium." [En línea]. Disponible: https://www.micrium.com

[66] E. Karahanna, D. Straub, y N. Chervany, "Information technology adoption across time: A cross-sectional comparison of pre-adoption and post-adoption beliefs," *MIS Quarterly: Management Information Systems*, vol. 23, num. 2, pp. 183–213, 1999.

Anexos

UNIVERSIDAD DE CUENCA

desde 1867

Anexo A

Distribución de terminales en las tarjetas SD

La distribución de terminales de cada versión y su conexión en modo SPI se pueden ver en las Tablas A.1 (SD), A.2 (miniSD) y A.3 (microSD).

Terminal	Nombre	Modo SD	Modo SPI
1	DAT3/CS	Línea de datos 3	SS
2	CMD/DI	Línea de comando	MOSI
3	VSS1	Tierra	Tierra
4	Vdd	Fuente de alimentación	Fuente de alimentación
5	Clock	Reloj	SCK
6	Vss2	Tierra	Tierra
7	DAT0/D0	Línea de datos 0	MISO
8	DAT1/IRQ	Línea de datos 1	No usado o IRQ
9	DAT2/NC	Línea de datos 2	No usado

Tabla A.1: Distribución de pines de una tarjeta SD estándar.

Terminal	Nombre	Modo SD	Modo SPI
1	CD/DAT3	Chip Select/Línea de datos 3	SS
2	CMD	Línea de comando	MOSI
3	VSS1	Tierra	Tierra
4	Vdd	Fuente de alimentación	Fuente de alimentación
5	Clock	Reloj	SCK
6	Vss2	Tierra	Tierra
7	DAT0	Línea de datos 0	MISO
8	DAT1	Línea de datos 1	No usado
9	DAT2	Línea de datos 2	No usado
10	NC	Para uso futuro	Para uso futuro
11	NC	Para uso futuro	Para uso futuro

Tabla A.2: Distribución de pines de una tarjeta miniSD.

Terminal	Nombre	Modo SD	Modo SPI
1	DAT2/NC	Línea de datos 2	No conectado
2	CD/DAT3/CS	card detect/ Línea de datos 3	Chip Select
3	CMD/DI	Línea de Comando	MOSI
4	Vdd	Fuente de alimentación	Fuente de alimentación
5	CLK	Reloj	SCK
6	Vss	Tierra	Tierra
7	DAT0/D0	Línea de datos 0	MISO
8	DAT1/RSV	Línea de datos 1	Reservado

Tabla A.3: Distribución de pines de una tarjeta microSD.

Anexo B

Sistemas operativos de tiempo real

A continuación se encuentra una lista de RTOSs conocidos con algunas de sus características.

Salvo:

Salvo es un RTOS cooperativo, multitarea, basado en prioridades y eventos, diseñado para microcontroladores de bajo costo y memoria limitada. Salvo tiene soporte para muchos microcontroladores, entre ellos: Familia 8051, ARM, Atmel AVR, M68HC11, MS430 y familia PIC.

Salvo esta escrito en lenguaje C, tiene soporte para muchos compiladores, entre ellos: Keil C51, Hi-Tech 8051, Hi-Tech PICC-18 y Microchip MPLAB C18.

Existen tres versiones de Salvo: la versión *demo* (*Salvo Lite*) y dos versiones de pago. La *demo* contiene un subconjunto de las funcionalidades de Salvo, y es usada solo para fines de evaluación. Por otro lado, las versiones de pago son LE y PRO. La versión LE es para sistemas que requieren un menor numero de tareas y menos características, mientras que la versión PRO cuenta con todas la características y es altamente configurable [48].

CCS:

CCS es un compilador en lenguaje C para microcontroladores PIC. Tiene varias versiones que dependen de las familias a las que están dirigidas. CCS ofrece un RTOS cooperativo que no requiere de interrupciones. Cuando una tarea pasa a ser ejecutada, adquiere el control del

procesador; cuando ésta termina o no necesita del procesador, devuelve el control al despachador de tareas (*dispatch*) para que se asigne a la siguiente tarea. Por ello, el RTOS no necesita de interrupciones, sin embargo no es con derecho preferente (*pre-emptive*), depende del usuario asegurarse de que una tarea no se ejecute infinitamente [39].

CMX-Tiny+:

CMX-Tiny+ es un RTOS multitarea con derecho preferente, diseñado para plataformas con poca memoria RAM (mínimo 512 bytes). Soporta una gran cantidad de microcontroladores, incluyendo la familia PIC24 y dsPIC. Entre sus servicios: control de tareas, eventos, mensajes, recursos, temporizadores cíclicos y de semáforos, y permite interrupción de tareas [61].

Ademas, CMX-Tiny+ admite la funcionalidad de trabajar como cooperativo en caso de ser necesario. Aunque este RTOS ofrece alto rendimiento, su desventaja es que tiene un precio relativamente alto, además que no tiene soporte para microcontroladores PIC de gamas inferiores (para Microchip, soporta las familias PIC24, dsPIC30, dsPIC33) [62].

PICos18:

PICos18 es un RTOS multitarea con derecho preferente, diseñado para la familia de microcontroladores PIC18 y esta desarrollado bajo la licencia *General Public License* (GPL) [63]. Soporta inicialización y programación, alarmas, salto entre rutinas, gestor de tareas, eventos e interrupciones.

Además, PICos18 permite que el acceso a los periféricos del microcontrolador funcione como una tarea independiente del *kernel*. Toda la documentación esta disponible y libre sin ninguna limitación [64].

MicroC/OS-II:

MicroC/OS-II es un RTOS portable, escalable, con derecho preferente y multitarea. Tiene soporte para muchos microcontroladores, incluyendo la familia PIC. El código fuente esta escrito en ANSI C, es libre, excepto para usos comerciales.

Entre sus características tiene: un código fuente limpio, consistente, bien documentado y organizado, es portable y escalable además robusto, confiable y eficiente [65]. Provee servicios de tareas, banderas de eventos, paso de mensajes, gestión de memoria, semáforos y gestión de tiempos.

FreeRTOS:

FreeRTOS es un sistema operativo de tiempo real de código abierto. Soporta diferentes

arquitecturas y compiladores, y esta diseñado para ser simple y ocupar poca memoria (ROM, RAM).

FreeRTOS es distribuido bajo los términos de la Licencia Publica General (GPL). Sin embargo, si se utiliza solamente el API de FreeRTOS en conjunto con otros módulos independientes, es posible distribuir el código con licencias diferentes a GPL [66].

Debido a que FreeRTOS esta escrito en lenguaje C, su código es altamente portable a muchas plataformas: ARM, Microchip PIC, Atmel AVER y Motorola/Freescale HCS12; cada una provista con un ejemplo [66].

OSA:

OSA es un RTOS cooperativo multitarea escrito como una función en C. Funciona en microcontroladores PIC de las familias PIC10, PIC12, PIC16, PIC18, PIC24 y dsPIC; en Atmel AVR 8-bit y en STMicroelectronics. Las tarea que realiza son: conmutación entre procesos paralelos, chequeo de tiempos fuera (*timeouts*), conteo de retrasos (*counting delays*), búsqueda y ejecución de tareas con mayor prioridad e intercambio de datos usando semáforos, mensajes, colas y otra señalización [49].

En el caso de los microprocesadores PIC, existen varios compiladores que funcionan para diferentes familias de microcontroladores. A continuación se puede ver la lista completa de compiladores y la familia para la que está disponible cada uno:

PICC STD: PIC10/12, PIC16/12, PIC18.
PICC PRO: PIC16F1xxx.
MPLAB C: PIC18, PIC24/dsPIC.
mikroC Pro: PIC16/12, PIC18 y PIC24/dsPIC en construcción.
CCS: PIC16/12, PIC18.

El calendarizador monitorea todas las tareas, compara su prioridad y entrega el control a la que tenga la mayor. Si encuentra una con la mayor posible, pasa a esta tarea. Si se está ejecutando una tarea crítica, este procedimiento no se cumple y se le da prioridad a esta tarea. Las tareas son sincronizadas a través de eventos. Una tarea puede estar en uno de los cinco estados siguientes:

No activa (*not active*): la tarea no ha sido creada o ha sido eliminada;
En espera (*waiting*): la tarea está esperando un evento;
lista (*ready*): la tarea ya no espera eventos, pero aún no ha recibido el control;
En ejecución (*running*): la tarea esta ejecutándose;

en pausa (*paused*): la tarea está activa, pero aún no puede recibir el control.

Para que una tarea tome el control del CPU debe cumplir con dos condiciones: que su estado sea lista (ready) y su prioridad mayor a otras tareas con el mismo estado.

Los eventos en este RTOS son:

Semáforos (binarios y de conteo): El binario puede tomar solo dos valores, mientras que el de conteo puede tomar cualquier valor. Las tareas que dependen de un semáforo se encuentran en espera hasta que este alcance el valor requerido.

Mensajes : sirven para que una tarea envíe información a otra.

Combinación de banderas se parecen a los semáforos binarios. Indican si ha sucedido o no un evento.

Tiempos fuera (*timeouts*): se dan cuando un temporizados se ha desbordado. Sirven para controlar tareas que requieren de tiempos específicos.

Expresiones boolianas: Se usan para eventos independientes del sistema operativo.

La prioridad que puede tener una tarea va de 0 a 7, siendo 0 la de mayor prioridad. Se cuenta con tres configuraciones de prioridad:

Prioridades deshabilitadas: se ignoran las prioridades.

Prioridades normales: cada tarea tiene una prioridad y las que tiene mayor se ejecutan primero. Al existir varias tareas con la prioridad mayor, se ejecutan en modo *round-robin*. Tiene dos desventajas: las de mayor prioridad pueden no dejar ejecutarse a las de menor prioridad y si dos tareas con la misma prioridad esperan un mismo evento, pudiera ejecutarse siempre la misma.

Prioridad extendida: las tareas obtienen el control del CPU por un tiempo acorde a su prioridad. Las tareas que están listas y no obtienen el control incrementan su prioridad.

El sistema provee el servicio de secciones criticas, las cuales son recursos compartidos que no pueden ser usados por dos tareas a la vez.

Anexo C

Características principales del microcontrolador PIC18F26K20

Parámetro	Valor
Tipo de memoria de programa	*Flash*
Memoria de programa (kB)	64
Velocidad de CPU (MIPS)	16
Bytes de RAM	3936
EEPROM de datos (bytes)	1024
Periféricos de comunicación digital	1 UART, 1 SPI, 1 I2C, 1 MSSP
Periféricos CCP	1 CCP, 1 ECCP
Temporizadores	1 8-bit, 3 16-bit
ADC	10 canales, 10-bit
Comparadores	2
Rango de temperatura (°C)	-40 a 125
Rango de voltaje de operación (V)	1,8 a 3,6
Número de pines	28
XLP (extreme low power)	Sí
Canales Cap Touch	10

Tabla C.1: Características principales del microcontrolador PIC18F26K20, fuente: [10].

Condiciones de funcionamiento estándar					
Temperatura de funcionamiento -40°C≤ T_A ≤125°C					
Min	**Typ**	**Max**	**Unidad**	**Condiciones**	
HFINTOSC Precisión @ Freq = 16 MHz, 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz					
-2	0	+2	%	+0° C a +70° C	V_{DD} = 1,8 - 3, 6 V
-3	-	+2	%	+70° C a +85° C	V_{DD} = 1,8 - 3,6V
-5	-	+5	%	-40° C a 0° C y +85° C a 125° C	V_{DD} = 1,8 - 3,6 V
LFINTOSC Precisión @ Freq = 31,25 kHz					
-15	-	+15	%	-40° C a +125° C	V_{DD} = 1,8-3,6 V

Tabla C.2: Características de precisión del oscilador interno PIC18F2XK20/4XK20, fuente: [10].

Anexo D

Programación del microcontrolador

D.1. Programa principal

```
#include <osa.h>
#include "ds1307_i2c_virtual.c"
#include "timelib.h"
#include "ADS1115.c"

// Pines para I2C por software
sbit Soft_I2C_Scl at RB1_bit;
sbit Soft_I2C_Sda at RB2_bit;
sbit Soft_I2C_Scl_Direction at TRISB1_bit;
sbit Soft_I2C_Sda_Direction at TRISB2_bit;

//Pines para sensor SHT11 (Temp. y humedad)
sbit SDA_pin at RB2_bit; // Serial data pin
sbit SCL_pin at RB1_bit; // Serial clock pin
sbit SDA_dir at TRISB2_bit; // Serial data direction pin
sbit SCL_dir at TRISB1_bit; // Serial clock direction pin

//Pines para Multiplexor
sbit S1_Mltplx at RB5_bit;
sbit S2_Mltplx at RB6_bit;
sbit S1_Mltplx_Direction at TRISB5_bit;
sbit S2_Mltplx_Direction at TRISB6_bit;

//Pines para Demultiplexor
```

```
sbit S2_Dmltplx at RB7_bit;
sbit S1_Dmltplx at RC1_bit;
sbit S2_Dmltplx_Direction at TRISB7_bit;
sbit S1_Dmltplx_Direction at TRISC1_bit;

//Pines para establecer modos funcionamiento
sbit switch1_modo at PORTA.B3;   //Interruptor 1
sbit switch2_modo at PORTA.B4;   //Interruptor 2
sbit led_verde at LATA.B5;                  //Led indicador verde
sbit c_switch1_modo at TRISA.B3;
sbit c_switch2_modo at TRISA.B4;
sbit c_led_verde at TRISA.B5;
char modo; //Numero de modo

//Configuracion de paramatros para XBee
#define START_DELIMITER 0x7E // Delimitador inicio
#define TX_REQUEST 0x10 //ZIGBEE Transmit Request
#define FRAME_ID 0x01 //
#define TX_REQUEST_OPTIONS 0x00 //Broadcast Radius
unsigned char TX_REQUEST_DESTINATION_ADDRESS_64[8]; //Dir. Xbee dest. de 64 bits
unsigned char TX_REQUEST_DESTINATION_ADDRESS_16[2]; //Dir. Xbee dest. de 16 bits
//
char TX_REQUEST_DESTINATION_ADDRESS_64_txt[24];
char TX_REQUEST_DESTINATION_ADDRESS_16_txt[6];

#define DEBUGGER 1    ////Directiva para compilar (1) con o sin debugger (0)

//Banderas y contadores
int cont_24h; //Contador de 24 horas
int cont_ms;  //Contador de milisegundos
int cont_seg; //Contador de segundos
int cont_min; //Contador de minutos

//Banderas para habilitar hilos
bit bandAnem;
bit bandPluv;
bit bandTempHumExt;
bit bandUV_Solar;
bit bandDRV;
bit bandTransmision;
bit band_GPS_SYNC;

bit primerGPS; //Bandera aviso de primera sincronizacion GPS=>RTC

int seg_prec;
int seg_th_out;
int seg_anem;
int seg_rad;
int seg_drv;
```

```
//Vraiables de configuracion estacion
char confEst[73];
char idEstTxt[] = "1_EST"; //Valor identificador estacion por defecto
char idSensor;
char confSensorTxt[5][6]; //Num. sensores x num. caracteres (habilitar/periodo)
char confSensor[5][2];
char txtTemporal[7]; //Guarda conversion de valores a string
char periodoTX; //Define el intervalo de transmision de los vectores de datos
//Variables contadores para cada sensor
unsigned char k; //Contador vector Precipitacion
unsigned char m; //Contador vector Temperatura
unsigned char h; //Contador vector Humedad relativa
unsigned char x; //Contador vector Anemometro
unsigned char y; //Contador vector Radiacion solar y uv
unsigned char g; //Contador vector Velocidad de viento
unsigned char bandEnvDat;

//Variables para anemometro
char * puntero8, * punteroDist; //punteros de 1 byte para contador Timer 1
long int contTimer1, distancia; //distancia= diferencia de contador timer1 entre
                                                          //dos pulsos
char frecAnem;
float periodo;
int * puntero16; //puntero de 2 bytes para contador Timer 1

//Variables para sensor de temperatura y humedad relativa
float temperatura, humedad_rel;
int temp16;
char hum8;

void Read_SHT11(float * fT, float * fRH);

bit STATE_I2C; //Controla el acceso a I2C
bit STATE_UART; //Controla el acceso a UART
bit STATE_SD; //Estado de memoria sd: disponible/ no disponible
char buffPRC[50];  //Buffer de pluviomtero
char buffTMP[100]; //Buffer de temperatura
char buffHMD[50];  //Buffer humedad relativa
char buffANM[50];  //Buffer veloc. de viento
char buffSLR[100]; //Buffer radiacion solar
char buffUV[100];  //Buffer radiacion UV
char buffDRV[100]; //Buffer direccion de viento

char * puntEpoch; //Puntero a variable epoch (time stamp)
char * puntTmp; //Puntero a variables de temperatura y humedad relativa
unsigned int idEst; //Identificadr de la estacion
char * puntID;

//Variables para estructura del tiempo en formato epoch
TimeStruct ts1; //Formato variable epoch
```

```
long epoch;
char txtEpoch[17]; //Almacenar conversion a string de Epoch

Hora HrF;
Fecha FchF;

unsigned char contPrcp; //Contador pulsos balancin pluviometro

//Variables para GPS
sbit NMEA_IntFlag at RCIF_bit; // NMEA Library RX Interrupt flag
char buffer[128]; // Buffer GPS
char rcvd = 0; // Bandera de sentencia NMEA recibida
char horaGPS[12];
char fechaGPS[10];
char fixGPS[2];
bit bandFix; //Bandera de fix gps
char horas[3];
unsigned int convInt;
bit bandGPS;
Hora sH;
Fecha sF;

//Variable temporales para conversion de ADC
float convADC;   //Conversion de radiacion solar y UV
float convVeloc; //Conversion de ADC

//Variables para memoria SD
typedef unsigned short uint8;
typedef signed short int8;
static const uint8
FILE_READ = 0x01, FILE_WRITE = 0x02, FILE_APPEND = 0x04;

// Mmc Library requirements
sbit Mmc_Chip_Select at RC0_bit;
sbit Mmc_Chip_Select_Direction at TRISC0_bit;

// SPI initialization routines required by Mmc Library
void initSPI(void) {
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV64,
     _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH);
}
void initFastSPI(void) {
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV4,
     _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH);
}

int8 err; //Codigo de error al utilizar memoria SD
short fhandle;  // file handle variable should be signed
                                // 4 = max files can be open at the same time
unsigned long fsize; //Guarda tamaño de archivo en bytes
```

```
// Método para guardar la fecha en cada vector de datos
void getTimeSensor() {
    //while (STATE_I2C);
    //STATE_I2C = 1;
    HrF = DS1307GetHora();
    FchF = DS1307GetFecha();
    //STATE_I2C = 0;
    ts1.ss = HrF.Seg;
    ts1.mn = HrF.Min;
    ts1.hh = HrF.Hor;
    ts1.md = FchF.Fec;
    ts1.mo = FchF.Mes;
    ts1.yy = FchF.Ano + 2000;
    epoch = Time_dateToEpoch( & ts1);

}

// Establece la fecha y hora en buffers de almacenamiento
void setTimeSensor(char BUFF[]) {
    BUFF[2] = * (puntEpoch + 2);
    BUFF[3] = * (puntEpoch + 1);
    BUFF[4] = * puntEpoch;
}

// Método para escribir en la memoria sd
void WRITE_SD(char BUFF[], char folder[], char nBytes) {
        txtEpoch[0]='\0';
        ByteToStr( FchF.Fec, txtEpoch );

        ByteToStr( FchF.Mes, txtTemporal );
        strcat(txtEpoch, txtTemporal);

        ByteToStr( FchF.Ano, txtTemporal );
        strcat(txtEpoch, txtTemporal);

        txtEpoch[0]=txtEpoch[1];
        txtEpoch[1]=txtEpoch[2];

        txtEpoch[2]=txtEpoch[4];
        txtEpoch[3]=txtEpoch[5];

        txtEpoch[4]=txtEpoch[7];
        txtEpoch[5]=txtEpoch[8];

        txtEpoch[6]='\0';

                // Se reemplaza los espacios " " por "0"
        for(idSensor=0; idSensor<6; idSensor++){
                        // Espacio ascii=0x20
```

```
            if(txtEpoch[idSensor]==0x20) txtEpoch[idSensor]='0';
        }

        FAT32_MakeDir(txtEpoch);    // Crea una nueva carpeta txtEpoch
        FAT32_ChangeDir(txtEpoch); // Ingresa a carpeta txtEpoch

    fhandle = FAT32_Open(folder, FILE_APPEND); //Abrir/crear TEXT.TX
    if (fhandle >= 0) {
        FAT32_Write(fhandle, BUFF, nBytes);
        delay_ms(100);
        //Close files
        err = FAT32_Close(fhandle); // Close  TEXT.TXT file
    }

    FAT32_ChangeDir(".."); // Regresa a directorio padre

}

// Metodo para transmitir por UART trama xbee
void TX_XBEE(char TX_REQUEST_RF_DATA[], char nDATA) {
    unsigned char i;
    unsigned char Packet_checksum;
    S1_Dmltplx = 1; // Switch to XBEE

        // Se calcula checksum de trama
    Packet_checksum = 0xFF;
    Packet_checksum -= FRAME_ID;
    Packet_checksum -= TX_REQUEST;
    for (i = 0; i < 8; i++)
    Packet_checksum -= TX_REQUEST_DESTINATION_ADDRESS_64[i];

    for (i = 0; i < 2; i++)
    Packet_checksum -= TX_REQUEST_DESTINATION_ADDRESS_16[i];

    for (i = 0; i < nDATA; i++)
    Packet_checksum -= TX_REQUEST_RF_DATA[i];

    // Paquete de datos a enviar
    UART1_Write(START_DELIMITER);
    UART1_Write(0x00);
    UART1_Write(nDATA + 14);
    UART1_Write(TX_REQUEST);
    UART1_Write(FRAME_ID);

    for (i = 0; i < 8; i++)
    UART1_Write(TX_REQUEST_DESTINATION_ADDRESS_64[i]);

    for (i = 0; i < 2; i++)
    UART1_Write(TX_REQUEST_DESTINATION_ADDRESS_16[i]);
```

```
    UART1_Write(0x00);
    UART1_Write(0x00);
    for (i = 0; i < nDATA; i++)
    UART1_Write(TX_REQUEST_RF_DATA[i]);

    UART1_Write(Packet_checksum);

    delay_ms(500);

    if(modo==1) S1_Dmltplx = 0;

}

// Método para sincronizar hora GPS con RTC
void sincGPS_Time() {
    strncpy(horas, horaGPS, 2); // Divide la trama hora en h:mm:ss
    convInt = atoi(horas); // Conversion a valores enteros
    sH.Hor = convInt;

    strncpy(horas, horaGPS + 2, 2);
    convInt = atoi(horas);
    sH.Min = convInt;

    strncpy(horas, horaGPS + 4, 2);
    convInt = atoi(horas);
    sH.Seg = convInt;

    strncpy(horas, fechaGPS, 2); // Divide la trama fecha en dd:mm:yy
    convInt = atoi(horas);
    sF.Dia = convInt;
    sF.Fec = convInt;

    strncpy(horas, fechaGPS + 2, 2);
    convInt = atoi(horas);
    sF.Mes = convInt;

    strncpy(horas, fechaGPS + 4, 2);
    convInt = atoi(horas);
    sF.Ano = convInt;

    DS1307SetHora(sH);
    DS1307SetFecha(sF);
    Delay_ms(100);
}

// Ticks de 1ms para RTOS
void InitTimer0() {
    T0CON = 0x88;
    TMR0H = 0xF0;
    TMR0L = 0x88;
```

```
    GIE_bit = 1;
    TMR0IE_bit = 1;
}

void Interrupt() {

    // Interrupcion timer 0 (ticks de 1ms para RTOS)
    if (TMR0IF_bit) {
        TMR0IF_bit = 0;
        TMR0H = 0xF0;
        TMR0L = 0x88;
        // Enter your code here
        OS_Timer(); // incrementing by one tick
        cont_ms++;
        if (cont_ms >= 1000 && primerGPS) {
           cont_ms=0;

           if(confSensor[0][0]==1){ // Pluviometro activo?
                 seg_prec++;
                 if(seg_prec==confSensor[0][1]){
                     seg_prec=0;
                     bandPluv = 1;
                 }
           }

           if(confSensor[1][0]==1){  // Tem y hum activo?
                 seg_th_out++;
                 if(seg_th_out==confSensor[1][1]){
                     seg_th_out=0;
                     bandTempHumExt = 1;
                 }
           }

           if(confSensor[2][0]==1){ // Velocidad de viento activo?
                 seg_anem++;
                 if(seg_anem==confSensor[2][1]){
                     seg_anem=0;
                     bandAnem = 1;
                 }
           }

           if(confSensor[3][0]==1){  // UV, Solar activo?
                 seg_rad++;
                 if(seg_rad==confSensor[3][1]){
                     seg_rad=0;
                     bandUV_Solar = 1;
                 }
           }

           if(confSensor[4][0]==1){  // Direccion de viento activo?
```

```
                seg_drv++;
                if(seg_drv==confSensor[4][1]){
                    seg_drv=0;
                    bandDRV = 1;
                }
            }

            cont_seg++; // Contador de minutos
            if(cont_seg==60){
                cont_seg=0;
                cont_min++;
                if(cont_min==periodoTX) {
                        cont_min=0;
                        bandTransmision=1;
                }
                 cont_24h++;
                 if(cont_24h==1440){ // 1440min=24h
                        cont_24h=0;
                        band_GPS_SYNC=1;
                }
            }

        }
    }

    // Interrupcion modulo CCCP, contador de pulsos pulviometro
    if (CCP1IF_bit) {
        CCP1IF_bit = 0; //Borra la interrupcion externa 0
        TMR1L = 21;
        TMR1H = 0X00;
        * (puntero8) = CCPR1L;
        * (puntero8 + 1) = CCPR1H;
        distancia = contTimer1;
        contTimer1 = 0;
    }

    // Interrupcion Timer 1, para calcular periodo entre pulsos
    if (TMR1IF_bit) {
        TMR1IF_bit = 0; //Borra la interrupcion externa 0
        * (puntero16 + 1) = * (puntero16 + 1) + 1;
    }
    if (INT0IF_bit) {
        contPrcp++;
        INT0IF_bit = 0;
    }

    // Interrupcion en recepcion UART
    if (RCIF_bit) {
        if (TRISB.B3) NMEA_IntHandler();
        RCIF_bit = 0;
```

```
    }
}
// Callback function called on reception of NMEA sentence
void NMEA_DataReceived() {
    rcvd = 1;
}

#pragma funcall main HiloSincGPS // Hilo para sincronizar gps
void HiloSincGPS(void) {
     primerGPS=0; // bandera para avisar al resto de hilos que ya
                  // se ha sincronizado el GPS por primera vez
     #if DEBUGGER==1
    if (modo==1) UART1_Write_Text("GPS habilitado\r\n");
    #endif

    while (1) {
    OS_Cond_Wait(band_GPS_SYNC || ~primerGPS);
      band_GPS_SYNC=0;
      //---------------
      #if DEBUGGER==1
       if (modo==1) UART1_Write_Text("Obteniendo fix de GPS...\r\n");
       #endif
        TRISB.B3 = 1;
        LATB.B3 = 1; //Despierta gps (energizar)
        S1_Mltplx = 1; //Switch to GPS
        S2_Mltplx = 0;
        RCIE_bit = 1; // Habilita interrupción UART1
        bandGPS = 1;
        while (bandGPS) {
        //iDebug=1;
            if (rcvd) {
               led_verde=~led_verde;
                   // Si se tiene fix y si sentencia es GGA
                   if (bandFix == 0) {
                       if (NMEA_IsValid(buffer) && strstr(buffer, "$GPGGA")) {
                           NMEA_Split(fixGPS, buffer, ',', 6); // get fix
                           if (fixGPS[0] == '1') bandFix = 1;
                       }
                   } else {
                       if (NMEA_IsValid(buffer) && strstr(buffer, "$GPRMC")) {
                           NMEA_Split(horaGPS, buffer, ',', 1); // Obtener hora
                           NMEA_Split(fechaGPS, buffer, ',', 9); // Obtnener fecha
                           bandGPS = 0; //Condicion para salir de bucle
                           bandFix = 0; //Se setea valor de fix
                           sincGPS_Time();
                       }
                   }
                   rcvd = 0;
               }
               OS_Delay(100);
```

```
        }
        primerGPS=1;
        RCIE_bit = 0; // Desactivada Interrupcion UART1
        TRISB.B3 = 0;
        LATB.B3 = 0; //Hasta mañana GPS (desenergizar)

        S1_Mltplx = 0; //Switch to XBEE
        S2_Mltplx = 1;
        led_verde=0;
        #if DEBUGGER==1
        if (modo==1) {UART1_Write_Text("...RTC sincronizado con GPS\r\n");
        led_verde=1;  }
        #endif
        //-----------------
      getTimeSensor();

    }
}

#pragma funcall main HiloTransmitir // Hilo para enviar datos por xbee
void HiloTransmitir(void) {
    while (1) {
        OS_Cond_Wait(bandTransmision);

        TRISB.B4 = 0;
        LATB.B4 = 0; // Despierta XBEE (hibernacion)
        delay_ms(20); // Cuento tiempo xbee necesitas para ponerte pilas?

        if (confSensor[0][0] == 1) { // Pluviometro activo?
            bandEnvDat.B0 = 1; // Se activa bandera para enviar datos
            setTimeSensor(buffPRC);
            TX_XBEE(buffPRC, 5 + k);
            #if DEBUGGER==1
            if (modo==1) UART1_Write_Text("Precip enviado xbee\r\n");
            #endif

            if (STATE_SD){
              WRITE_SD(buffPRC, "PRC", 5 + k);
              #if DEBUGGER==1
              if (modo==1) UART1_Write_Text("Precip guardado sd\r\n");
              #endif
            }
            k = 0;
            bandEnvDat.B0 = 0;
        }

        if (confSensor[1][0] == 1) { // Temperatura y humedad activo?
            bandEnvDat.B1 = 1;
            setTimeSensor(buffTMP);
            setTimeSensor(buffHMD);
```

```
        TX_XBEE(buffTMP, 5 + m);
        TX_XBEE(buffHMD, 5 + h);
        #if DEBUGGER==1
        if (modo==1) UART1_Write_Text("TMP HMD enviado xbee\r\n");
        #endif

        if (STATE_SD){
          WRITE_SD(buffTMP, "TMP", 5 + m);
          WRITE_SD(buffHMD, "HMD", 5 + h);
          #if DEBUGGER==1
          if (modo==1) UART1_Write_Text("TMP HMD guardado sd\r\n");
          #endif
        }

        m = 0;
        h=0;
        bandEnvDat.B1 = 0;
    }

    if (confSensor[2][0] == 1) { // Velocidad de viento activo?
        bandEnvDat.B2 = 1;
        setTimeSensor(buffANM);
        TX_XBEE(buffANM, 5 + x);
        #if DEBUGGER==1
        if (modo==1) UART1_Write_Text("Anem enviado xbee\r\n");
        #endif

        if (STATE_SD){
          WRITE_SD(buffANM, "ANM", 5 + x);
          #if DEBUGGER==1
          if (modo==1) UART1_Write_Text("Anem guardado sd\r\n");
          #endif
         }

        x = 0;
        bandEnvDat.B2 = 0;
    }

    if (confSensor[3][0] == 1) { // UV, Solar activo?
        bandEnvDat.B3 = 1;
        setTimeSensor(buffSLR);
        setTimeSensor(buffUV);
        TX_XBEE(buffSLR, 5 + y);
        TX_XBEE(buffUV, 5 + y);
        #if DEBUGGER==1
        if (modo==1) UART1_Write_Text("SLR UV enviado xbee\r\n");
        #endif

        if (STATE_SD){
          WRITE_SD(buffSLR, "SLR", 5 + y);
```

```
                WRITE_SD(buffUV, "UV", 5 + y);
                #if DEBUGGER==1
                if (modo==1) UART1_Write_Text("SLR UV guardado sd\r\n");
                #endif
            }

            y = 0;
            bandEnvDat.B3 = 0;
        }

        if (confSensor[4][0] == 1) { // Direccion de viento activo?
            bandEnvDat.B4 = 1;
            setTimeSensor(buffDRV);
            TX_XBEE(buffDRV, 5 + g);
            #if DEBUGGER==1
            if (modo==1) UART1_Write_Text("DRV enviado xbee\r\n");
            #endif

            if (STATE_SD){
              WRITE_SD(buffDRV, "DRV", 5 + g);
              #if DEBUGGER==1
              if (modo==1) UART1_Write_Text("DRV guardado sd\r\n");
              #endif
            }

            g = 0;
            bandEnvDat.B4 = 0;
        }

        getTimeSensor();
        bandTransmision=0; // Desactiva bandera de transmision
        TRISB.B4=1;
        LATB.B4=1; // Hiberna XBEE
    }
}

#pragma funcall main HiloPluviometro // Hilo para sensor precipitacion
void HiloPluviometro(void) {
    unsigned char IDS[6];
    unsigned char TX_DATA[8];
    buffPRC[0] = * (puntID + 1);
    buffPRC[1] = * puntID | 0x01;
    k=0;
    #if DEBUGGER==1
    if (modo==1) UART1_Write_Text("Sensor de Precipitacion habilitado\r\n");
    #endif
    while (1) {
          OS_Cond_Wait(bandPluv);

          if(modo!=3){
```

```
            // Si el HiloTransmitir esta enviado los datos
            while (bandEnvDat.B0 == 1); // esperar q haya enviado
            buffPRC[5 + k] = contPrcp;
            k++;
            if(k>=40) k=0;
          } else{
            buffPRC[2] = contPrcp;
            while(bandEnvDat);
            bandEnvDat=1;
            TX_XBEE(buffPRC, 3);
            bandEnvDat=0;
          }

          #if DEBUGGER==1
          if (modo==1) {
              while(STATE_UART);
              STATE_UART=1;
              UART1_Write_Text("Precipitacion: ");
              ShortToStr(contPrcp, txtTemporal);
              UART1_Write_Text(txtTemporal);
              UART1_Write_Text(" mm");
              UART1_Write_Text("\r\n");
                          STATE_UART=0;
          }
          #endif
          contPrcp = 0;
          bandPluv = 0;

    }

}

#pragma funcall main HiloTempHumExt // Hilo para sensor de temp.y humedad
void HiloTempHumExt(void) {
    buffTMP[0] = * (puntID + 1);
    buffTMP[1] = * puntID | 0x02;
    buffHMD[0] = * (puntID + 1);
    buffHMD[1] = * puntID | 0x03;
    puntTmp = & temp16;
    h=0;
    m=0;
    #if DEBUGGER==1
    if (modo==1) UART1_Write_Text("Sensor de Temp. y hum. habilitado\r\n");
    #endif
    while (1)
    {
          OS_Cond_Wait(bandTempHumExt);

          while (STATE_I2C);
          STATE_I2C = 1;
```

```
SCL_dir = 0; // SCL is output
SDA_pin = 1; // SDA is high
Read_SHT11( & temperatura, & humedad_rel);
STATE_I2C = 0;

temp16 = (int) 100 * temperatura;
hum8 = (int) humedad_rel;

if(modo!=3){
  // Si el HiloTransmitir esta enviado los datos
  while (bandEnvDat.B1) // esperar q haya enviado
  m=m;
  buffTMP[5 + m] = * (puntTmp + 1);
  buffTMP[6 + m] = * puntTmp;

  buffHMD[5 + h] = hum8;
  h++;
  m+=2;
  if(m>=80) {m=0; h=0;}

}else{
  buffTMP[2] = * (puntTmp + 1);
  buffTMP[3] = * puntTmp;
  buffHMD[2] = hum8;
  while(bandEnvDat);
  bandEnvDat=1;
  TX_XBEE(buffTMP, 4);
  TX_XBEE(buffHMD, 3);
  bandEnvDat=0;
}
#if DEBUGGER==1
if (modo==1) {
    while(STATE_UART);
    STATE_UART=1;
    UART1_Write_Text("Temperatura: ");
    IntToStr(temp16, txtTemporal);
    UART1_Write_Text(txtTemporal);
    UART1_Write_Text(" °C");
    UART1_Write_Text("\r\n");

    UART1_Write_Text("Hum. relativa: ");
    IntToStr(hum8, txtTemporal);
    UART1_Write_Text(txtTemporal);
    UART1_Write_Text(" %");
    UART1_Write_Text("\r\n");
    STATE_UART=0;
}
#endif
bandTempHumExt = 0;
```

```
    }
}

#pragma funcall main HiloVelocViento // Hilo para sensor vel. de viento
void HiloVelocViento(void) {
    contTimer1 = 0;
    distancia = 0;
    frecAnem = 0;
    periodo = 0;
    puntero16 = & contTimer1; // Se apunta en puntero direccion de contTimer1
    puntero8 = & contTimer1;
    CCP1CON = 0b00000101; // Deteccion de evento en flanco ascendente
    TMR1ON_bit = 1; // Enables Timer1
    buffANM[0] = * (puntID + 1);
    buffANM[1] = * puntID | 0x04;
    punteroDist = & distancia;
    x=0;
    #if DEBUGGER==1
    if (modo==1) UART1_Write_Text("Sensor de Veloc. de viento habilitado\r\n");
    #endif
    while (1)
    {
          OS_Cond_Wait(bandAnem);

          distancia=distancia;
          if (distancia != 0) {
              periodo = (float) 4 / 16000000 * distancia;
              frecAnem = (unsigned int) 1 / periodo; //Redondea a int ej 16.66=16
              distancia = 0;
          } else {
              frecAnem = 0;
          }

          if(modo!=3){
            // Si el HiloTransmitir esta enviado los datos
            while (bandEnvDat.B2) // esperar q haya enviado
            x=x;
            buffANM[5 + x] = frecAnem;
            x++;
            if(x>=40) x=0;
          } else{
            buffANM[2] = frecAnem;
            while(bandEnvDat);
            bandEnvDat=1;
            TX_XBEE(buffANM, 3);
            bandEnvDat=0;
          }

          #if DEBUGGER==1
```

```
            if (modo==1) {
                while(STATE_UART);
                STATE_UART=1;
                UART1_Write_Text("Veloc. de viento: ");
                convVeloc= (float) frecAnem*3.6208;
                convVeloc=convVeloc/confSensor[2][1];
                FloatToStr(convVeloc, txtTemporal);
                UART1_Write_Text(txtTemporal);
                UART1_Write_Text(" Km/h");
                UART1_Write_Text("\r\n");
                            STATE_UART=0;
            }
            #endif

            bandAnem = 0;

    }
}

#pragma funcall main HiloRadiacion // Hilo para radiacion UV, Solar
void HiloRadiacion(void) {
    char cProm;
    signed long cntRad; // Guarda lectura de ADC
    char * puntLectRad;
    ADS1115_init();
    puntLectRad = & cntRad; // Se apunta en puntero direccion de cntRad
    buffSLR[0] = * (puntID + 1);
    buffSLR[1] = * puntID | 0x05;
    buffUV[0] = * (puntID + 1);
    buffUV[1] = * puntID | 0x06;
    y=0;
    #if DEBUGGER==1
    if (modo==1) UART1_Write_Text("Sensor Solar y UV habilitado\r\n");
    #endif
    while (1)
    {
          OS_Cond_Wait(bandUV_Solar);

          cntRad=cntRad;
          cntRad=0;
          for(cProm=0; cProm<16; cProm++){
            while (STATE_I2C);
            STATE_I2C = 1;
            ADS1115_configure(start_one_conversion | AINP_AIN0__AINN_GND |
            FS_4096mV | power_down_single_shot_mode | data_rate_860SPS |
            disable_comparator);
            delay_ms(1);
            cntRad += ADS1115_read(ADS1115_conversion_reg_pointer);
            STATE_I2C = 0;
            delay_ms(1);
```

```
}
cntRad=cntRad/16;

if(modo!=3){
  // Si el HiloTransmitir esta enviado los datos
  while (bandEnvDat.B3) // esperar q haya enviado
  y=y;
  buffSLR[5 + y] = * (puntLectRad + 1);
  buffSLR[6 + y] = * puntLectRad;
} else{
  buffSLR[2] = * (puntLectRad + 1);
  buffSLR[3] = * puntLectRad;
  while(bandEnvDat);
  bandEnvDat=1;
  TX_XBEE(buffSLR, 4);
  bandEnvDat=0;
}

#if DEBUGGER==1
if (modo==1) {
    while(STATE_UART);
    STATE_UART=1;
    UART1_Write_Text("Solar: ");
    convADC= (float) cntRad*0.074852;
    FloatToStr(convADC, txtTemporal);
    UART1_Write_Text(txtTemporal);
    UART1_Write_Text(" W/m^2");
    UART1_Write_Text("\r\n");
                STATE_UART=0;
}
#endif

cntRad=0;
for(cProm=0; cProm<16; cProm++){
  while (STATE_I2C);
  STATE_I2C = 1;
  ADS1115_configure(start_one_conversion | AINP_AIN1__AINN_GND |
  FS_4096mV | power_down_single_shot_mode | data_rate_860SPS |
  disable_comparator);
  delay_ms(1);
  cntRad += ADS1115_read(ADS1115_conversion_reg_pointer);
  STATE_I2C = 0;
  delay_ms(1);
}
cntRad=cntRad/16;
if(modo!=3){
  buffUV[5 + y] = * (puntLectRad + 1);
  buffUV[6 + y] = * puntLectRad;
  y+=2;
  if(y>=80) y=0;
```

```
            } else{
              buffUV[2] = * (puntLectRad + 1);
              buffUV[3] = * puntLectRad;
              while(bandEnvDat);
              bandEnvDat=1;
              TX_XBEE(buffUV, 4);
              bandEnvDat=0;
            }

            #if DEBUGGER==1
            if (modo==1) {
                while(STATE_UART);
                STATE_UART=1;
                UART1_Write_Text("UV: ");
                convADC= (float) cntRad*0.00083336;
                FloatToStr(convADC, txtTemporal);
                UART1_Write_Text(txtTemporal);
                UART1_Write_Text(" Índice");
                UART1_Write_Text("\r\n");
                            STATE_UART=0;
            }
            #endif

            bandUV_Solar = 0;

/*CONVERSION
        volt = cnt*v_max/(2^15-1);*/
    }
}

#pragma funcall main HiloDirecViento // Hilo para sensor Dir. viento
void HiloDirecViento(void) {
    signed int cntDRV; // Guarda lectura de ADC
    char * puntLectDRV;
    ADS1115_init();
    puntLectDRV = & cntDRV; // Se apunta en puntero direccion de cntDRV
    buffDRV[0] = * (puntID + 1);
    buffDRV[1] = * puntID | 0x07;
    g=0;
    #if DEBUGGER==1
    if (modo==1) UART1_Write_Text("Sensor de Direc. de viento habilitado\r\n");
    #endif
    while (1)
    {
          OS_Cond_Wait(bandDRV);

          while (STATE_I2C);
          STATE_I2C = 1;
          ADS1115_configure(start_one_conversion | AINP_AIN2__AINN_GND |
          FS_4096mV | power_down_single_shot_mode | data_rate_860SPS |
```

```
            disable_comparator);
            delay_ms(1);
            cntDRV = ADS1115_read(ADS1115_conversion_reg_pointer);
            STATE_I2C = 0;

            if(modo!=3){
              //Si el HiloTransmitir esta enviado los datos
              while (bandEnvDat.B4); //esperar q haya enviado
              buffDRV[5 + g] = * (puntLectDRV + 1);
              buffDRV[6 + g] = * puntLectDRV;
              g+=2;
              if(g>=80) g=0;
            } else{
              buffDRV[2] = * (puntLectDRV + 1);
              buffDRV[3] = * puntLectDRV;
              while(bandEnvDat);
              bandEnvDat=1;
              TX_XBEE(buffDRV, 4);
              bandEnvDat=0;
            }
            #if DEBUGGER==1
            if (modo==1) {
                while(STATE_UART);
                STATE_UART=1;
                UART1_Write_Text("Dir. viento: ");
                convADC= (float) cntDRV*0.013637;
                FloatToStr(convADC, txtTemporal);
                UART1_Write_Text(txtTemporal);
                UART1_Write_Text(" grados");
                UART1_Write_Text("\r\n");
                STATE_UART=0;
            }
            #endif

            bandDRV = 0;

/*CONVERSION
        volt = cnt*v_max/(2^15-1);*/
    }
}

void main() {
    IRCF0_bit = 1; /////////////////
    IRCF1_bit = 1; // Se configura registro INTOSC a 16MHz
    IRCF2_bit = 1; /////////////////

    ANSEL = 0; // Configura pines AN como digital I/O
    ANSELH = 0;
    C1ON_bit = 0; // Se desabilita comparadores
    C2ON_bit = 0;
```

```
// Config Multiplexor
S1_Mltplx_Direction = 0;
S2_Mltplx_Direction = 0;
S1_Mltplx = 0;
S2_Mltplx = 0;

// Config Demultiplexor
S1_Dmltplx_Direction = 0;
S2_Dmltplx_Direction = 0;
S1_Dmltplx = 0; // Conexion a serial
S2_Dmltplx = 1; // PC por defecto (B2)

// Pines para interruptores para configura modo de funcionamiento
c_switch1_modo = 1; // Pin como entrada
c_switch2_modo = 1; // Pin como entrada
c_led_verde = 0; // Pin como salida, led indicador de
                                    // sincronizacion de GPS y debugger
LATA = 0x00;
modo=0x00;

    // Se lee interuptores para configurar modo de datalogger
modo.B0=switch1_modo;
modo.B1=switch2_modo;
// modo=switch2_modo&switch1_modo; // modo=0&0=0 Transm. con xbee
                                   // modo=0&1=1 Transm. con xbbe con debugger
                                   // modo=1&0=2 Transm. sin xbee (solo serial)
                                   // modo=1&1=3 Transm. tiempo real
if (modo==1) {
led_verde = 1; //Led indicador de debugger activo
}

if(modo==3){
  TRISB.B4 = 0;
  LATB.B4 = 0; // Despierta XBEE
} else{
  TRISB.B4=1;
  LATB.B4=1; // Duerme XBEE
}

UART1_Init(9600); // Inicializa modulo UART en 9600 bps
Delay_ms(100); // Tiempo de espera para estabilizar UART
NMEA_Init( & UART1_Read, buffer, sizeof(buffer));
RCIE_bit = 0; // Interrupcion UART1 desactivada,
              // ésta se controla en el metodo que sincroniza el gps
PEIE_bit = 1; // Se habilita interrupcion de perifericos

//Configuracion interrupcion externa INT0 (para pluviometro)
TRISB.B0 = 1;
LATB.B0 = 0;
```

```
INT0IE_bit = 1; // Habilita interrupciones externas
INTEDG0_bit = 1; // Interrupcion en flanco ascendente
contPrcp = 0; // Variable para contar los flancos del sensor de precipitacion

// Configuracion Timer1 (para velocidad de viento)
T1CON = 0b00000000; // Stop Timer1,Internal clock (FOSC/4), 1:1 Prescale value
TMR1IE_bit = 1; // Habilita interrupcion por overflow timer1
// PEIE_bit=1; // Habilita interrupciones perfericas
TMR1H = 0;
TMR1L = 0;

// Configuracion modulo CCP (para velocidad de viento)
TRISC.B2 = 1; // BIT CCP1
CCP1IE_bit = 1; // Habilita interrupcion de CCP1
CCP1IP_bit = 1; // Interrupcion de alta prioridad para CCP1
GIE_bit = 1; // Interrupciones habilitadas.

DS1307Inicio(); // Inicia libreria de RTC

STATE_I2C = 0; // Bandera de estado para protocolo I2C
STATE_SD = 0;  // Bandera de estado para memoria SD
STATE_UART=0;  // Bandera de estado para UART
bandFix = 0;
bandGPS = 1;
puntID = & idEst;
bandEnvDat = 0x00;
puntEpoch = & epoch;

// Banderas y contadores
cont_ms = 0;
cont_seg = 0;
cont_min = 0;
bandAnem = 0;
bandPluv = 0;
bandTempHumExt = 0;
bandUV_Solar = 0;
bandDRV = 0;
bandTransmision=0;
band_GPS_SYNC=0;

seg_prec=0;
seg_th_out=0;
seg_anem=0;
seg_rad=0;
seg_drv=0;
cont_24h=0;

#if DEBUGGER==1
if (modo==1) {
  UART1_Write_Text("DATALOGGER en modo: ");
```

```
  ShortToHex(modo, txtTemporal);
  UART1_Write_Text(txtTemporal);
  UART1_Write_Text("\r\n");
  Delay_ms(100);
}
#endif

// Configuracion SD
initSPI();
err = FAT32_Init(); // Inicializa SD
if (err >= 0) {
    //  Si SD es detectada se reinicia
    //  SPI con mayor velocidad
    ////////////////////////////////////
    initFastSPI();
    STATE_SD=1; //Bandera SD activa
    #if DEBUGGER==1
    if (modo==1) UART1_Write_Text("SD inicio ok\r\n");
    #endif

    // Proceso de lectura de archivo de configuracion
    fhandle = FAT32_Open("CONFIG.TXT", FILE_READ);
    if (fhandle >= 0) { // ¿Se abrio archivo de texto?
       #if DEBUGGER==1
        if (modo==1) UART1_Write_Text("Leyendo archivo configuracion...\r\n");
        #endif
        FAT32_Size("CONFIG.TXT", &fsize);
        FAT32_Read(fhandle, confEst, fsize);
        confEst[fsize] = '\0';
        NMEA_Split(idEstTxt, confEst, ';', 0);
       #if DEBUGGER==1
        if (modo==1) {
            UART1_Write_Text("ID estacion: ");
            UART1_Write_Text(idEstTxt);
            UART1_Write_Text("\r\n");
        }
        #endif
        idEst = atoi(idEstTxt); // Convierte Id. estacion de texto a numero
        strcat(idEstTxt, "_EST"); // Se crea nombre de carpeta principal
                                                                    // con el identificador

                    // Se separa direccion 1 y 2 de XBee
        NMEA_Split(TX_REQUEST_DESTINATION_ADDRESS_64_txt, confEst, ';', 1);
        NMEA_Split(TX_REQUEST_DESTINATION_ADDRESS_16_txt, confEst, ';', 2);
        #if DEBUGGER==1
        if (modo==1) {
            UART1_Write_Text("64-bit dir. destino: ");
            UART1_Write_Text(TX_REQUEST_DESTINATION_ADDRESS_64_txt);
            UART1_Write_Text("\r\n");
            UART1_Write_Text("16-bit dir. destino: ");
```

```
    UART1_Write_Text(TX_REQUEST_DESTINATION_ADDRESS_16_txt);
    UART1_Write_Text("\r\n");
    Delay_ms(100);
}
#endif

            // Se separa valores de direccion 1 de XBee
for (x = 0; x < 8; x++) {
    NMEA_Split(txtTemporal, TX_REQUEST_DESTINATION_ADDRESS_64_txt, ',', x);
    TX_REQUEST_DESTINATION_ADDRESS_64[x] = atoi(txtTemporal);
}

            // Se separa valores de direccion 2 de XBee
for (x = 0; x < 2; x++) {
    NMEA_Split(txtTemporal, TX_REQUEST_DESTINATION_ADDRESS_16_txt, ',', x);
    TX_REQUEST_DESTINATION_ADDRESS_16[x] = atoi(txtTemporal);
}

            // Se divide parametros de cada sensor
for (x = 0; x < 5; x++) { //5 #sensors
    NMEA_Split(confSensorTxt[x], confEst, ';', x + 3);
}

            // Se separa valores de cada sensor
for (x = 0; x < 5; x++) {
    for (y = 0; y < 2; y++) {
        NMEA_Split(txtTemporal, confSensorTxt[x], ',', y);
        confSensor[x][y] = atoi(txtTemporal);
    }
}

            // Se separa intervalo de transmision de datos
NMEA_Split(txtTemporal, confEst, ';', 8);
periodoTX = atoi(txtTemporal);

#if DEBUGGER==1
if (modo==1) {
    UART1_Write_Text("Periodo Transmision: ");
    UART1_Write_Text(txtTemporal);
    UART1_Write_Text("\r\n");
    Delay_ms(100);
}
#endif

// Cierra archivo de texto
err = FAT32_Close(fhandle); //Cierra archivo de texto
#if DEBUGGER==1
if (modo==1) UART1_Write_Text("...configuracion cargada\r\n");
#endif
Delay_ms(100);
```

```
        }

        //Proceso para escribir en archivos de texto
        err = 0;
        if (err == FAT32_MakeDir(idEstTxt)) // Crea directorio principal
        {
        #if DEBUGGER==1
            if (modo==1) UART1_Write_Text("Directorio principal creado\r\n");
            #endif
            err = FAT32_ChangeDir(idEstTxt); // Ingresa en direcorio
        } else {
        #if DEBUGGER==1
            if (modo==1) UART1_Write_Text("Directorio principal no creado\r\n");
            #endif
            err = FAT32_ChangeDir(idEstTxt); // Ingresa en direcorio principal
        }


    } else { // Se carga en configuracion valores por defecto
    #if DEBUGGER==1
        if (modo==1) {
                        UART1_Write_Text("SD no detectada\r\n");
            UART1_Write_Text("Cargando configuracion por defecto...\r\n");
                        }
        #endif
        idEst = 1;
        for (x = 0; x < 8; x++) TX_REQUEST_DESTINATION_ADDRESS_64[x] = 0x00;
        TX_REQUEST_DESTINATION_ADDRESS_16[0] = 0xFF;
        TX_REQUEST_DESTINATION_ADDRESS_16[1] = 0xFE;

            confSensor[0][0] = 1;  // Pluviometro
            confSensor[0][1] = 20; // Tiempo de muestreo

            confSensor[1][0] = 1;  // Temperatura y humedad externa
            confSensor[1][1] = 10; // Tiempo de muestreo

            confSensor[2][0] = 1;  // Vel. de viento
            confSensor[2][1] = 10; // Tiempo de muestreo

            confSensor[3][0] = 1;  // Radiacion solar y uv
            confSensor[3][1] = 50; // Tiempo de muestreo

            confSensor[4][0] = 1;  // Direccion de viento
            confSensor[4][1] = 10; // Tiempo de muestreo

        periodoTX=5; //Periodo de transmision y almac. en minutos
    }

    idEst = IdEst << 4; // Se valida que Id. estacion sea de 12 bits
```

```
OS_Init(); // Incializa servicios de RTOS

//Creación de hilo para sincronizar GPS RTC
    OS_Task_Create(0, HiloSincGPS);

//Creación de hilo para transmitir datos de sensores
    if(modo!=3) OS_Task_Create(1, HiloTransmitir);

    //Creación de hilo para sensor de precipitación (pluviometro)
if (confSensor[0][0] == 1) {OS_Task_Create(5, HiloPluviometro);}
else{
#if DEBUGGER==1
   if (modo==1) UART1_Write_Text("Sensor de precipitacion deshabilitado\r\n");
   #endif
}

//Creación de hilo para sensor de temperatura y humedad
    if (confSensor[1][0] == 1) {OS_Task_Create(4, HiloTempHumExt);}
else{
#if DEBUGGER==1
   if (modo==1) UART1_Write_Text("Sensor de Temp. Hum. deshabilitado\r\n");
   #endif
}

    //Creación de hilo para sensor de velocidad de viento
if (confSensor[2][0] == 1) {OS_Task_Create(6, HiloVelocViento);}
else{
#if DEBUGGER==1
   if (modo==1) UART1_Write_Text("Sensor de Veloc. de viento deshabilitado\r\n");
   #endif
}

    //Creación de hilo para sensores de radiacion solar y UV
if (confSensor[3][0] == 1) {OS_Task_Create(2, HiloRadiacion);}
else{
#if DEBUGGER==1
   if (modo==1) UART1_Write_Text("Sensor Solar y UV deshabilitado\r\n");
   #endif
}

    //Creación de hilo para sensor de velocidad de viento
if (confSensor[4][0] == 1) {OS_Task_Create(3, HiloDirecViento);}
else{
#if DEBUGGER==1
   if (modo==1) UART1_Write_Text("Sensor de Direc. de viento deshabilitado\r\n");
   #endif
}

InitTimer0(); // Inicia temporizador de RTOS
OS_Run(); // Llama al planificador (scheduler)
```

```
}
```

D.2. Modificación de Librería D.3.5 a I2C por software

```
 //Declaración de estructuras
typedef struct
{
unsigned short Hor, Min, Seg;
}Hora;
typedef struct
{
unsigned short Dia, Fec, Mes, Ano;
}Fecha;
//Función para definir dirección de memoria.
void DS1307SetDir( unsigned short dir )
{
Soft_I2C_Start();
Soft_I2C_Write(0xD0);
Soft_I2C_Write(dir);
}
//Función para convertir de código Bcd a Entero.
unsigned short BcdToShort( unsigned short bcd )
{
unsigned short LV, HV;
LV = bcd&0x0F;
HV = (bcd>>4)&0x0F;
return LV + HV*10;
}
//Función para convertir de Entero a Bcd.
unsigned short ShortToBcd( unsigned short valor )
{
unsigned short HV, LV;
HV = valor/10;
LV = valor - HV*10;
return LV + HV*16;
}
//Función para inicializar el DS1307.
void DS1307Inicio( void )
{
unsigned short VAL[7], HV, LV, DATO;
Soft_I2C_Init(); //Inicio del bus I2C.
delay_ms(50); //Retardo.
//Lectura de las primeras 7 direcciones.
DS1307SetDir(0);
Soft_I2C_Start();
Soft_I2C_Write(0xD1);
VAL[0] = Soft_I2C_Read(1);
VAL[1] = Soft_I2C_Read(1);
```

```
VAL[2] = Soft_I2C_Read(1);
VAL[3] = Soft_I2C_Read(1);
VAL[4] = Soft_I2C_Read(1);
VAL[5] = Soft_I2C_Read(1);
VAL[6] = Soft_I2C_Read(0);
Soft_I2C_Stop();
delay_ms(50); //Retardo.
//Validación y corrección de información,
//como hora y fecha.
DATO = BcdToShort( VAL[0] );
if( DATO > 59 )VAL[0]=0;
DATO = BcdToShort( VAL[1] );
if( DATO>59 )VAL[1]=0;
DATO = BcdToShort( VAL[2] );
if( DATO>23 )VAL[2]=0;
DATO = BcdToShort( VAL[3] );
if( DATO>7 || DATO==0 )VAL[3]=1;
DATO = BcdToShort( VAL[4] );
if( DATO>31 || DATO==0 )VAL[4]=1;
DATO = BcdToShort( VAL[5] );
if( DATO>12 || DATO==0 )VAL[5]=1;
DATO = BcdToShort( VAL[6] );
if( DATO>99 )VAL[6]=0;
//Grabación de las primeras 7 direcciones.
DS1307SetDir(0);
Soft_I2C_Write(VAL[0]);
Soft_I2C_Write(VAL[1]);
Soft_I2C_Write(VAL[2]);
Soft_I2C_Write(VAL[3]);
Soft_I2C_Write(VAL[4]);
Soft_I2C_Write(VAL[5]);
Soft_I2C_Write(VAL[6]);
Soft_I2C_Write(0x10); //Se activa la salida oscilante 1Hz.
Soft_I2C_Stop();
delay_ms(50); //Retardo.
}
//Función para grabar la hora minutos y segundos.
void DS1307SetHora( Hora h )
{
DS1307SetDir(0);
Soft_I2C_Write( ShortToBcd(h.Seg) );
Soft_I2C_Write( ShortToBcd(h.Min) );
Soft_I2C_Write( ShortToBcd(h.Hor) );
Soft_I2C_Stop();
}
//Función para grabar el día, fecha, mes, y año.
void DS1307SetFecha( Fecha f )
{
DS1307SetDir(3);
Soft_I2C_Write( ShortToBcd(f.Dia) );
```

```
Soft_I2C_Write( ShortToBcd(f.Fec) );
Soft_I2C_Write( ShortToBcd(f.Mes) );
Soft_I2C_Write( ShortToBcd(f.Ano) );
Soft_I2C_Stop();
}
//Función para leer la hora minutos y segundos.
Hora DS1307GetHora( void )
{
Hora H;
unsigned short VAL[3];
DS1307SetDir(0);
Soft_I2C_Start();
Soft_I2C_Write(0xD1);
VAL[0] = Soft_I2C_Read(1);
VAL[1] = Soft_I2C_Read(1);
VAL[2] = Soft_I2C_Read(0);
Soft_I2C_Stop();
H.Seg = BcdToShort( VAL[0] );
H.Min = BcdToShort( VAL[1] );
H.Hor = BcdToShort( VAL[2] );
return H;
}
//Función para leer el día, fecha, mes, y año.
Fecha DS1307GetFecha( void )
{
Fecha F;
unsigned short VAL[4];
DS1307SetDir(3);
Soft_I2C_Start();
Soft_I2C_Write(0xD1);
VAL[0] = Soft_I2C_Read(1);
VAL[1] = Soft_I2C_Read(1);
VAL[2] = Soft_I2C_Read(1);
VAL[3] = Soft_I2C_Read(0);
Soft_I2C_Stop();
F.Dia = BcdToShort( VAL[0] );
F.Fec = BcdToShort( VAL[1] );
F.Mes = BcdToShort( VAL[2] );
F.Ano = BcdToShort( VAL[3] );
return F;
}
// Funciones para leer y grabar datos individuales //
//Función para leer las horas.
unsigned short DS1307GetHoras( void )
{
Hora h;
h=DS1307GetHora();
return h.Hor;
}
//Función para leer los minutos.
```

```
unsigned short DS1307GetMinutos( void )
{
Hora h;
h=DS1307GetHora();
return h.Min;
}
//Función para leer los segundos.
unsigned short DS1307GetSegundos( void )
{
Hora h;
h=DS1307GetHora();
return h.Seg;
}
//Función para grabar las horas.
void DS1307SetHoras( unsigned short ho )
{
Hora h;
h=DS1307GetHora();
h.Hor = ho;
DS1307SetHora( h );
}
//Función para grabar los minutos.
void DS1307SetMinutos( unsigned short mi )
{
Hora h;
h=DS1307GetHora();
h.Min = mi;
DS1307SetHora( h );
}
//Función para grabar los segundos.
void DS1307SetSegundos( unsigned short se )
{
Hora h;
h=DS1307GetHora();
h.Seg = se;
DS1307SetHora( h );
}
//Función para leer el día de la semana.
unsigned short DS1307GetDias( void )
{
Fecha f;
f=DS1307GetFecha();
return f.Dia;
}
//Función para leer la fecha del mes.
unsigned short DS1307GetFechas( void )
{
Fecha f;
f=DS1307GetFecha();
return f.Fec;
```

```
}
//Función para leer el mes del año.
unsigned short DS1307GetMeses( void )
{
Fecha f;
f=DS1307GetFecha();
return f.Mes;
}
//Función para leer el año.
unsigned short DS1307GetAnos( void )
{
Fecha f;
f=DS1307GetFecha();
return f.Ano;
}
//Función para grabar el dia de la semana.
void DS1307SetDias( unsigned short di )
{
Fecha f;
f=DS1307GetFecha();
f.Dia = di;
DS1307SetFecha(f);
}
//Función para grabar la fecha del mes.
void DS1307SetFechas( unsigned short fe )
{
Fecha f;
f=DS1307GetFecha();
f.Fec = fe;
DS1307SetFecha(f);
}
//Función para grabar el mes del año.
void DS1307SetMeses( unsigned short me )
{
Fecha f;
f=DS1307GetFecha();
f.Mes = me;
DS1307SetFecha(f);
}
//Función para grabar el año.
void DS1307SetAnos( unsigned short an )
{
Fecha f;
f=DS1307GetFecha();
f.Ano = an;
DS1307SetFecha(f);
}
```

D.3. Librerías

Para la programación del microcontrolador se ha utilizado varias librerías con las que cuenta el compilador mikroC PRO for PIC. Sin embargo, además de estas se ha utilizado las siguientes librerías:

D.3.1. Manejo de memoria SD con formato FAT32

- FAT32 Library
- Versión 2.5.0.0
- https://libstock.mikroe.com/projects/view/108/fat32-library

D.3.2. Decodificación de sentencias NMEA

- NMEA Library
- Version 1.0.0.0
- https://libstock.mikroe.com/projects/view/695/nmea-library

D.3.3. Comunicación con sensor de Temperatura y humedad SHT11

- SHT1x click - Example
- Version 1.0.0.0
- https://libstock.mikroe.com/projects/view/211/sht1x-click-example

D.3.4. Comunicación con ADC ADS1115

- ADS1115 I2C 16-bit ADC + PGA Library
- Version 1.0.0.0
- https://libstock.mikroe.com/projects/view/1896/ads1115-i2c-16-bit-adc-pga-library

D.3.5. Comunicación con RTC DS1307

- Librería DS1307, en libro:
- J. R. Clavijo Mendoza, Diseño y simulación de sistemas microcontrolados en lenguaje C. Programación con el compilador MikroC PRO y simulación en Proteus ISIS. Colombia, 2011.

Anexo E

Configuración del RTOS OSA

El código fuente del RTOS OSA se puede descargar de la página oficial, el cual incluye un programa (*OSAcfg_Tool_Setup.exe*) para configurar diferentes parámetros de OSA. Este programa permite definir el compilador que se va a utilizar, el número de hilos, semáforos, temporizadores, etc., creando un archivo de configuración (OSAcfg.h) necesario para utilizar el kernel de OSA.

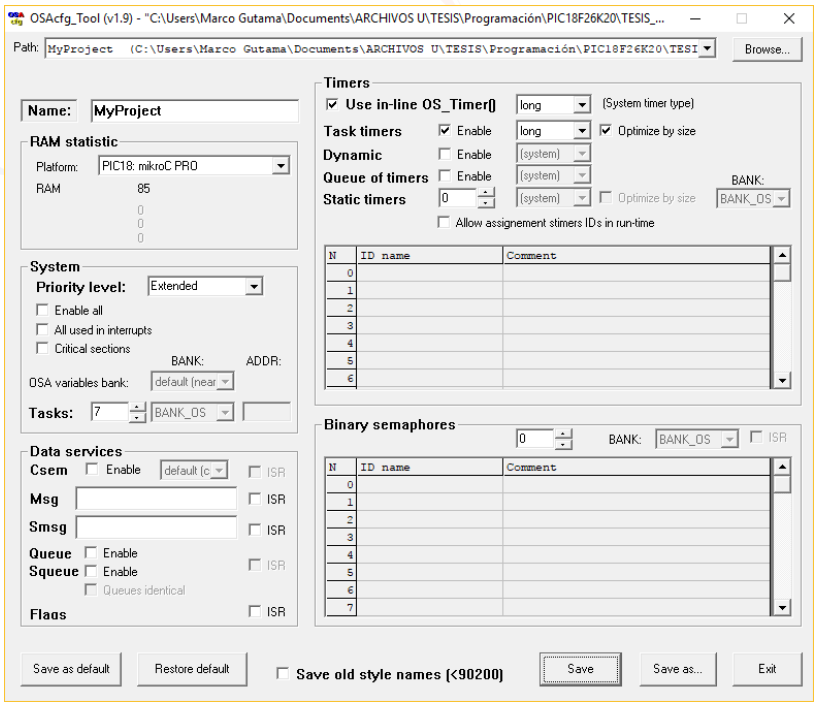


Figura E.1: Generador de archivo de configuración para OSA.

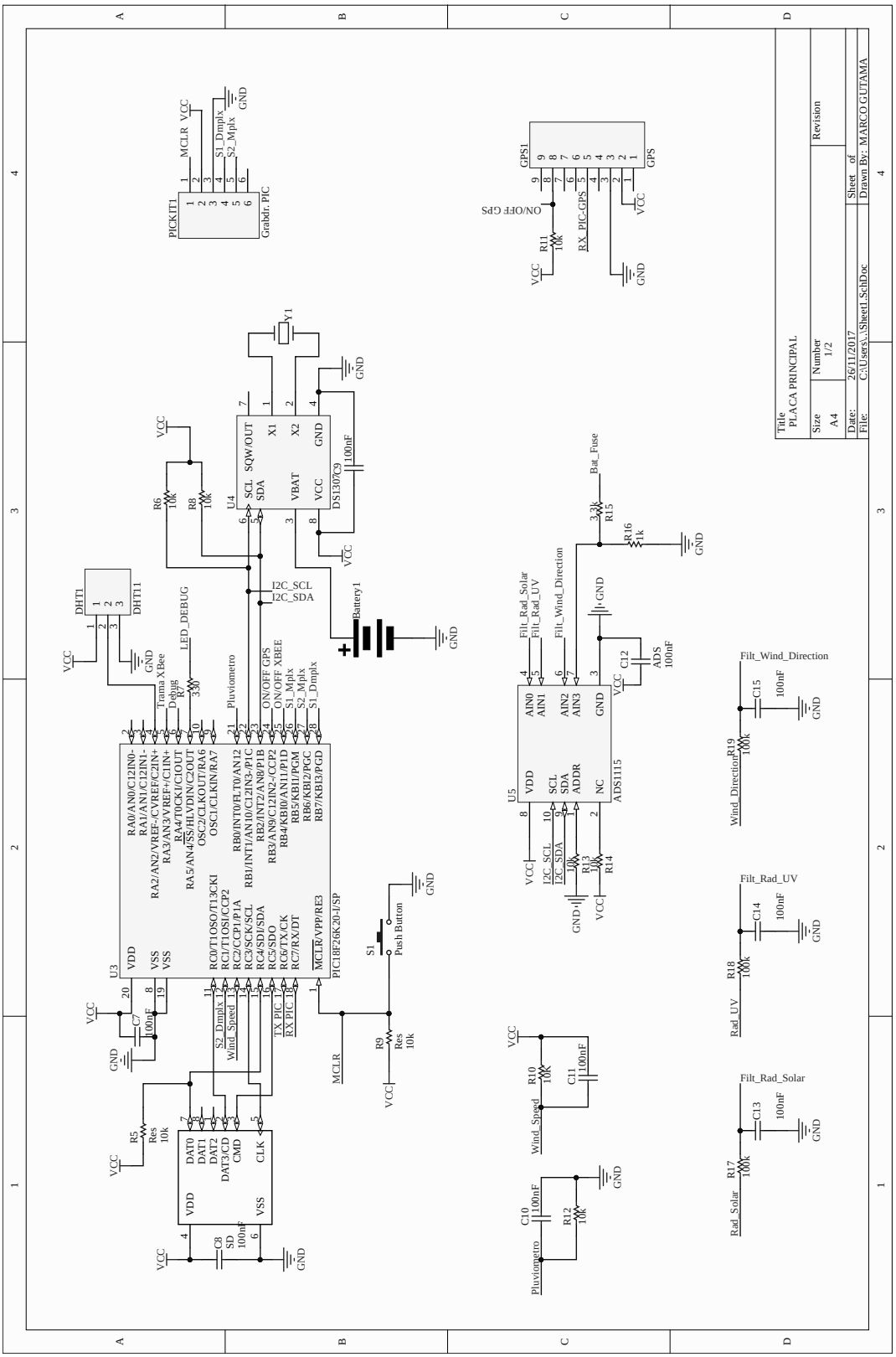
Para la aplicación desarrollada se configuro el compilar MikroC para la familia de PICs18, 7 hilos de ejecución, tareas temporizadas y uso del servicio OS_Timer() como función para establecer la base de tiempo del sistema. En la Figura E.1 se muestra éste programa y la configuración realizada. El archivo de configuración *OSAcfg.h* generado se muestra a continuación:

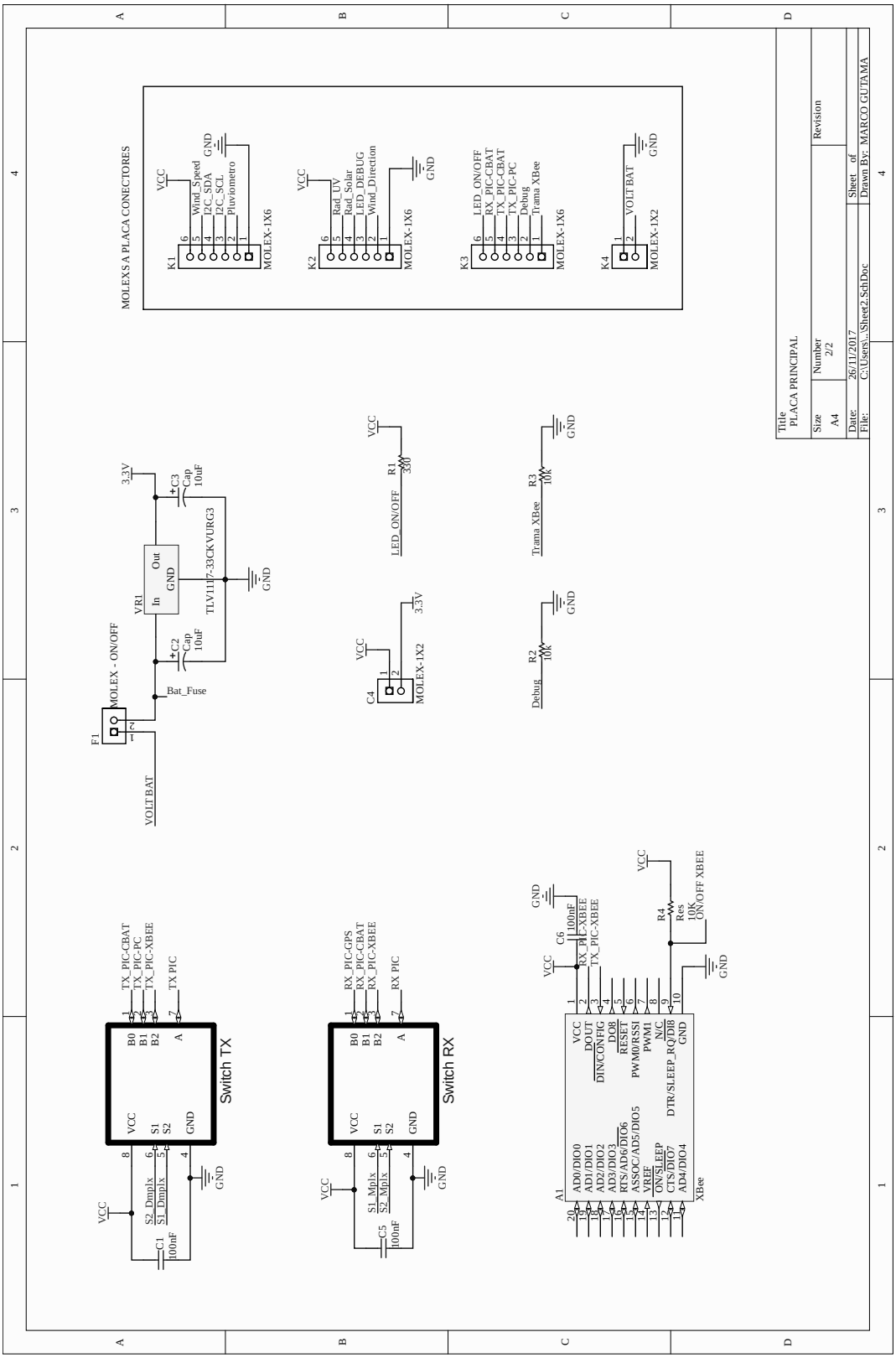
```
/*************************************************************************/
//
// This file was generated by OSAcfg_Tool utility.
// Do not modify it to prevent data loss on next editing.
//
// PROJECT NAME: MyProject
//
// PLATFORM:     mikroC for PIC18
//
/*************************************************************************/


#ifndef _OSACFG_H
#define _OSACFG_H

//-------------------------------------------------------------------------
// SYSTEM
//-------------------------------------------------------------------------

#define OS_TASKS               7   // Number of tasks that can be active at one time
#define OS_PRIORITY_LEVEL  OS_PRIORITY_EXTENDED

//-------------------------------------------------------------------------
// ENABLE CONSTANTS
//-------------------------------------------------------------------------

#define OS_ENABLE_TTIMERS          // Enable task timers (OS_Delay and OS_xxx_Wait_TO)
#define OS_USE_INLINE_TIMER        // Make OS_Timer service as in-line function
#define OS_TTIMERS_OPTIMIZE_SIZE   // Optimize OS_Timer code for task timers by size


//-------------------------------------------------------------------------
// TYPES
//-------------------------------------------------------------------------

#define OS_TIMER_SIZE          4   // Size of all system timers (1, 2 or 4)
#define OS_TTIMER_SIZE         4   // Size of task timers (1, 2 or 4)



#endif
```

Anexo F

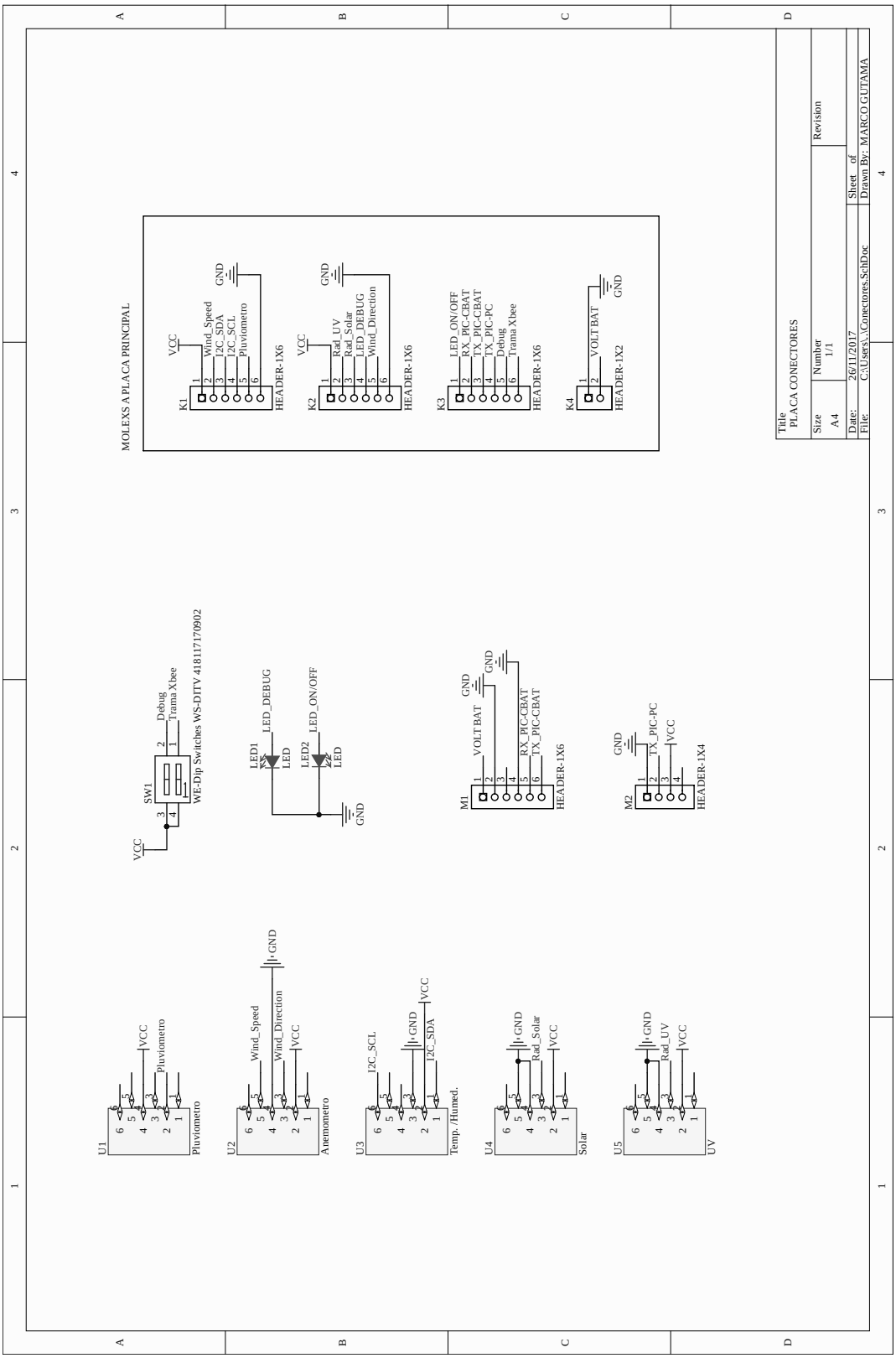
Diseños Esquemáticos y PCBs

F.1. Esquema placa principal

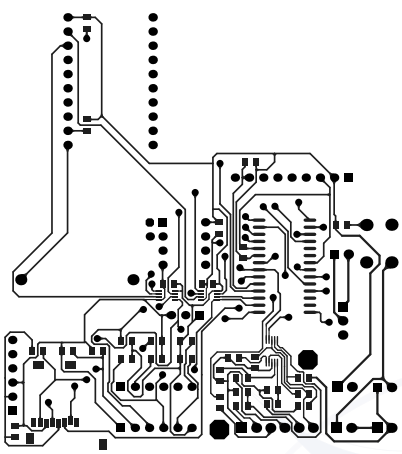



MOLEXS A PLACA CONECTORES
K1
MOLEX-1X6
Wind_Speed
I2C_SDA
I2C_SCL
Pluviometro
GND
VCC
K2
MOLEX-1X6
Rad_UV
Rad_Solar
LED_DEBUG
Wind_Direction
GND
VCC
K3
MOLEX-1X6
LED_ON/OFF
RX_PIC-CBAT
TX_PIC-CBAT
TX_PIC-PC
Debug
Trama XBee
K4
MOLEX-1X2
VOLT BAT
GND
Title
PLACA PRINCIPAL
Size
A4
Number
2/2
Revision
Date:
26/11/2017
Sheet of
File:
C:\Users\...\Sheet2.SchDoc
Drawn By:
MARCO GUTAMA
F1
MOLEX - ON/OFF
VOLT BAT
Bat_Fuse
VR1
TLV1117-33CKVURG3
3.3V
C3
Cap
10uF
C2
Cap
10uF
GND
R1
330
LED_ON/OFF
VCC
C4
MOLEX-1X2
R3
10k
Trama XBee
R2
10k
Debug
Switch TX
TX_PIC-CBAT
TX_PIC-PC
TX_PIC-XBEE
TX PIC
C1
100nF
Switch RX
RX_PIC-GPS
RX_PIC-CBAT
RX_PIC-XBEE
RX PIC
C5
100nF
A1
XBee
C6
100nF
RX_PIC-XBEE
TX_PIC-XBEE
R4
Res
10K
ON/OFF XBEE

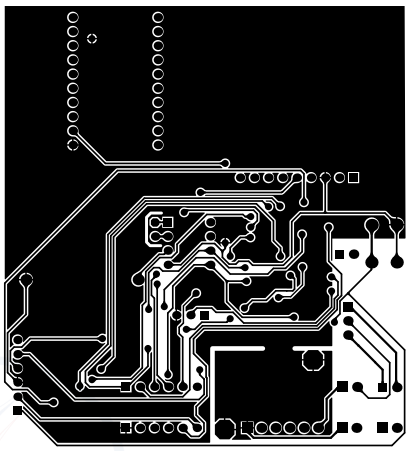
F.2. Esquema placa conectores



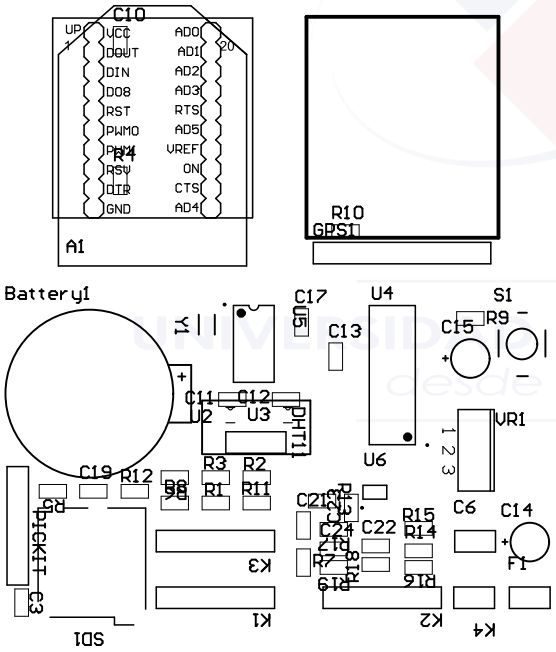
F.3. Diseño PCB principal



(a) Parte superior



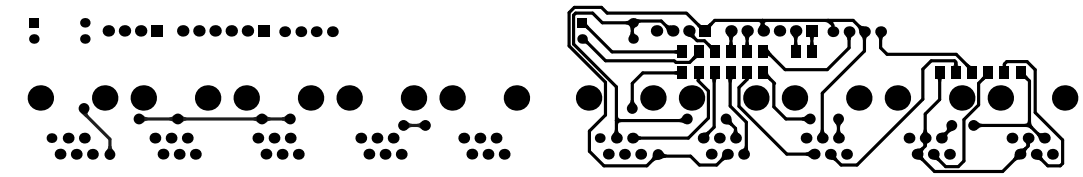
(b) Parte inferior



(c) Etiqueta de componentes parte superior

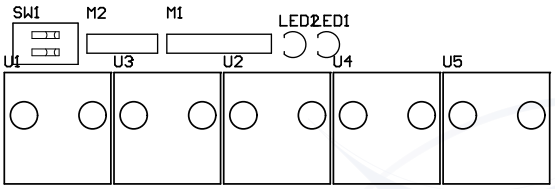
Figura F.1: PCB principal.

F.4. Diseño PCB conectores



(a) Parte superior

(b) Parte inferior

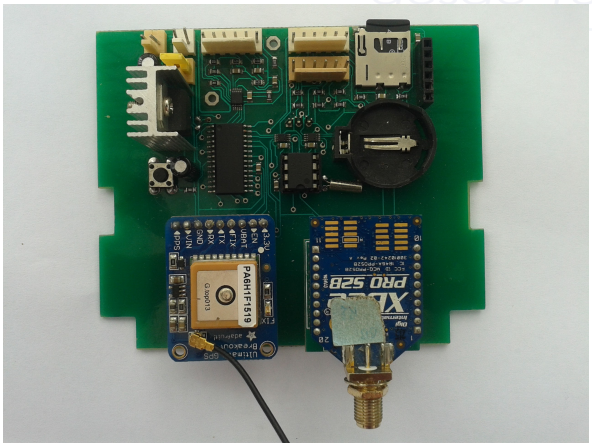


(c) Etiqueta de componentes parte superior

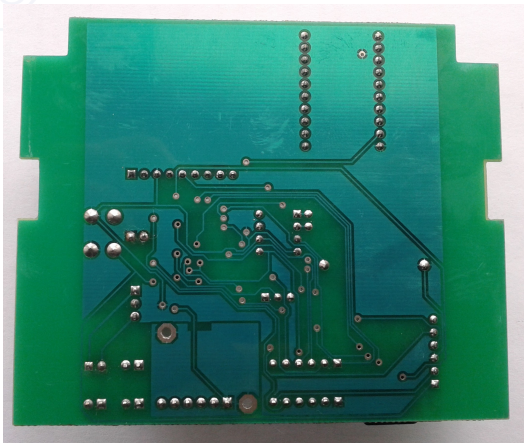
Figura F.2: PCB conectores.

F.5. Montaje de componentes en PCB principal

La placa fue diseñada en su mayoría con componentes de montaje superficial.



(a) Vista superior

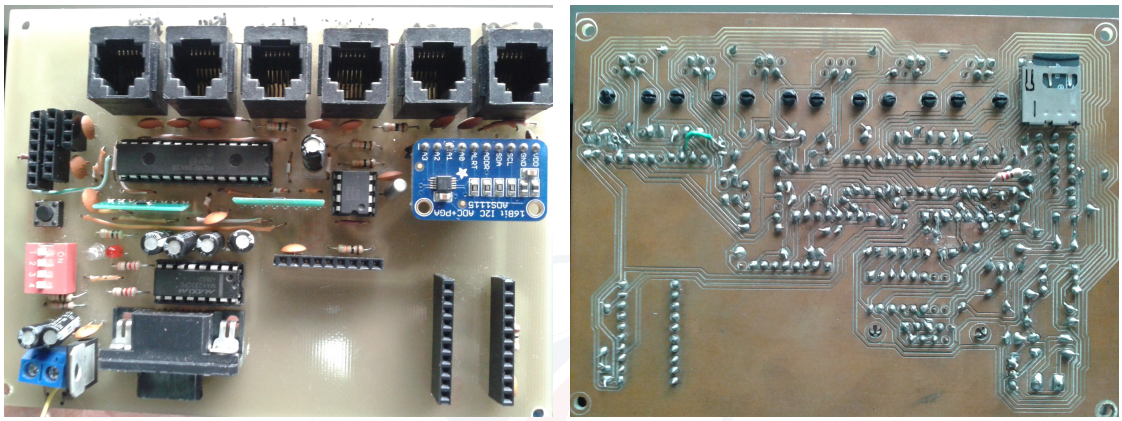


(b) Vista inferior

Figura F.3: PCB final.

F.6. PCB Inicial

Esta placa fue desarrollada inicialmente para probar el diseño antes de conseguir los materiales de montaje superficial.



(a) Vista superior (b) Vista inferior

Figura F.4: PCB inicial desarrollado para pruebas.

Anexo G

Imágenes de prueba realizada a sensor de dirección de viento

En las imágenes del presente anexo se observan la toma de datos de dirección de viento posicionando manualmente la veleta en 8 direcciones representativas. A la izquierda de estas imágenes se observan los datos enviados a la computadora a través del puerto serial, y a la derecha, la posición del transductor del anemómetro. En todas las imágenes se muestra a la izquierda el registro de datos que llega al computador y a la derecha la posición del sensor. Las medidas en la parte inferior del registro corresponden a la posición actual de cada fotografía.

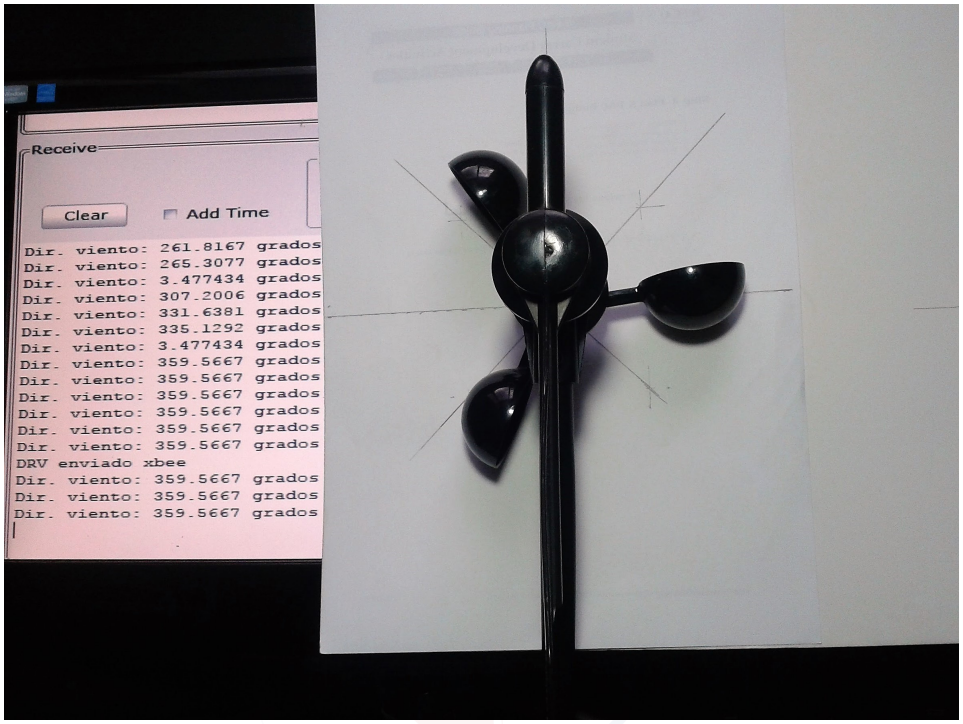


Figura G.1: Dirección norte. Izquierda: Registro de datos. Derecha: Posición del sensor.

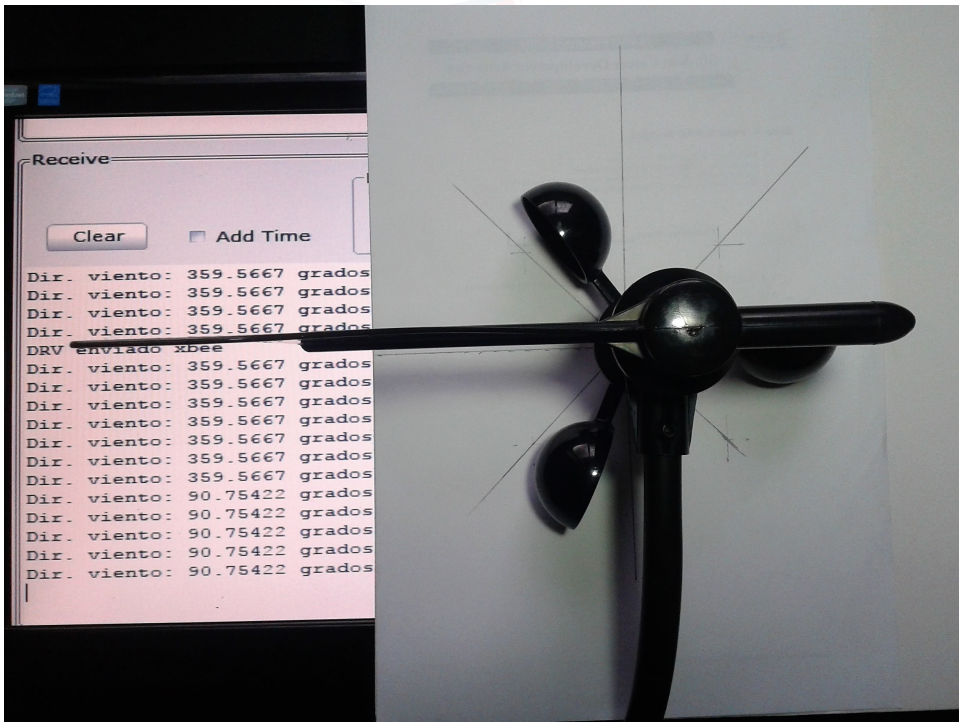


Figura G.2: Dirección este. Izquierda: Registro de datos. Derecha: Posición del sensor.

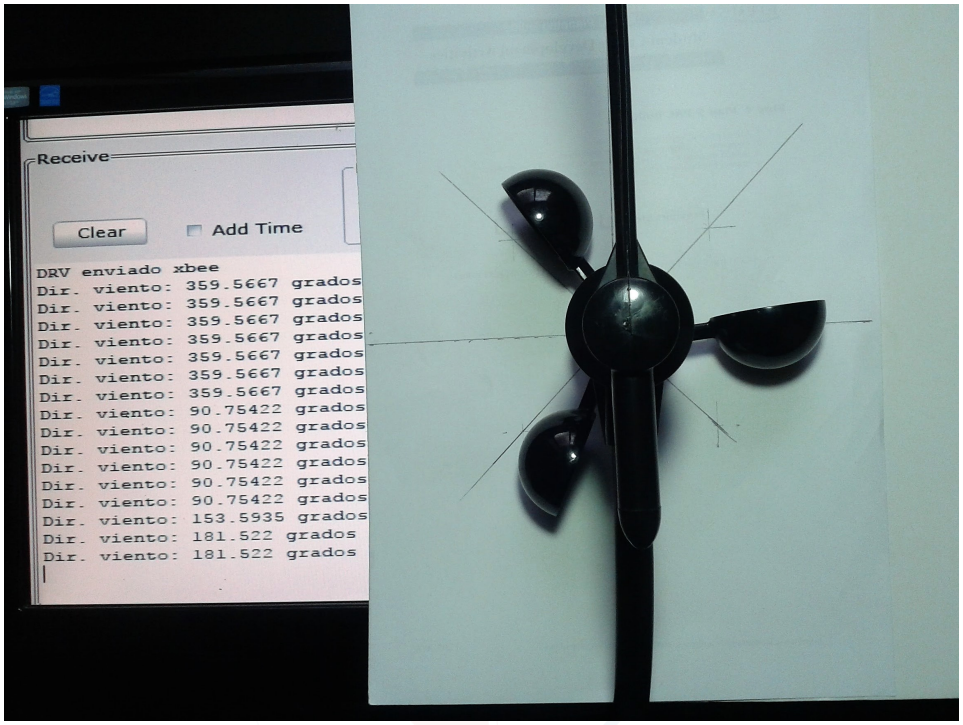


Figura G.3: Dirección sur. Izquierda: Registro de datos. Derecha: Posición del sensor.

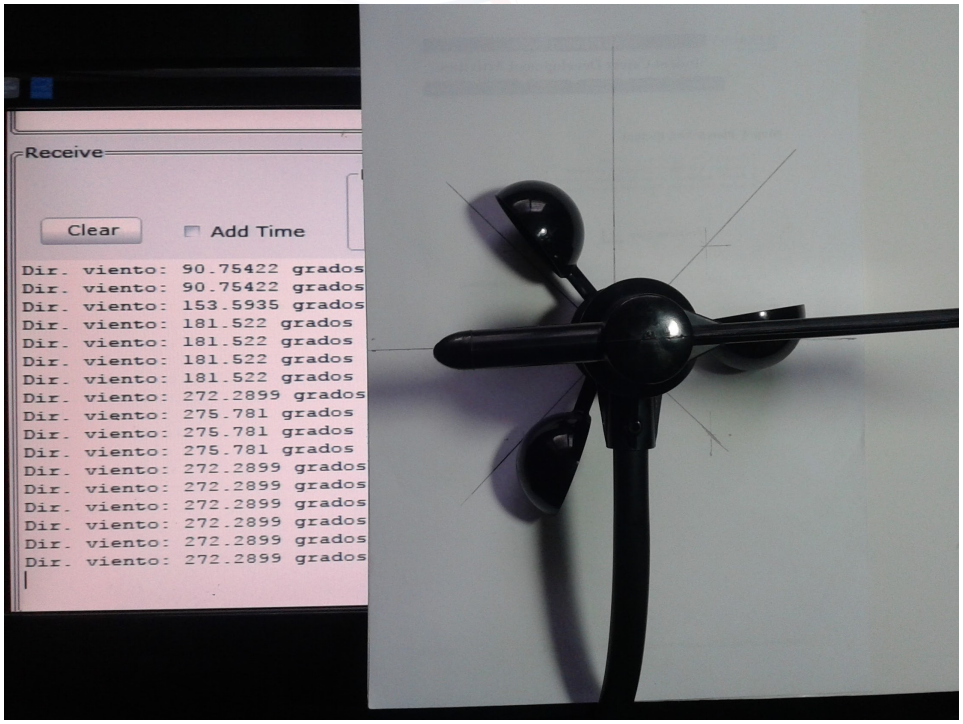


Figura G.4: Dirección oeste. Izquierda: Registro de datos. Derecha: Posición del sensor.

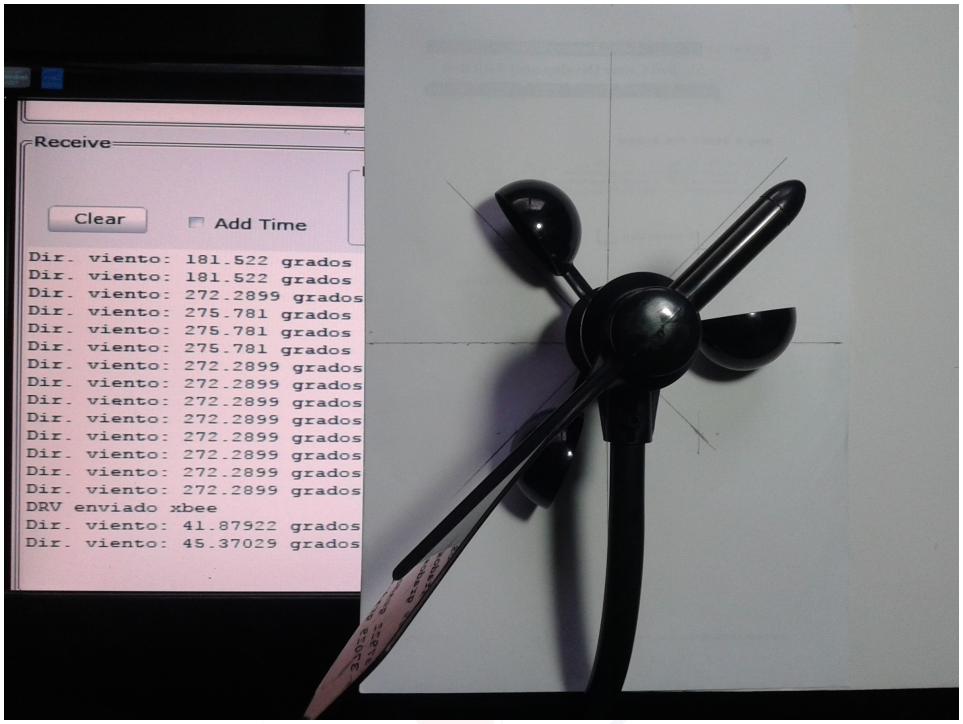


Figura G.5: Dirección noreste. Izquierda: Registro de datos. Derecha: Posición del sensor.

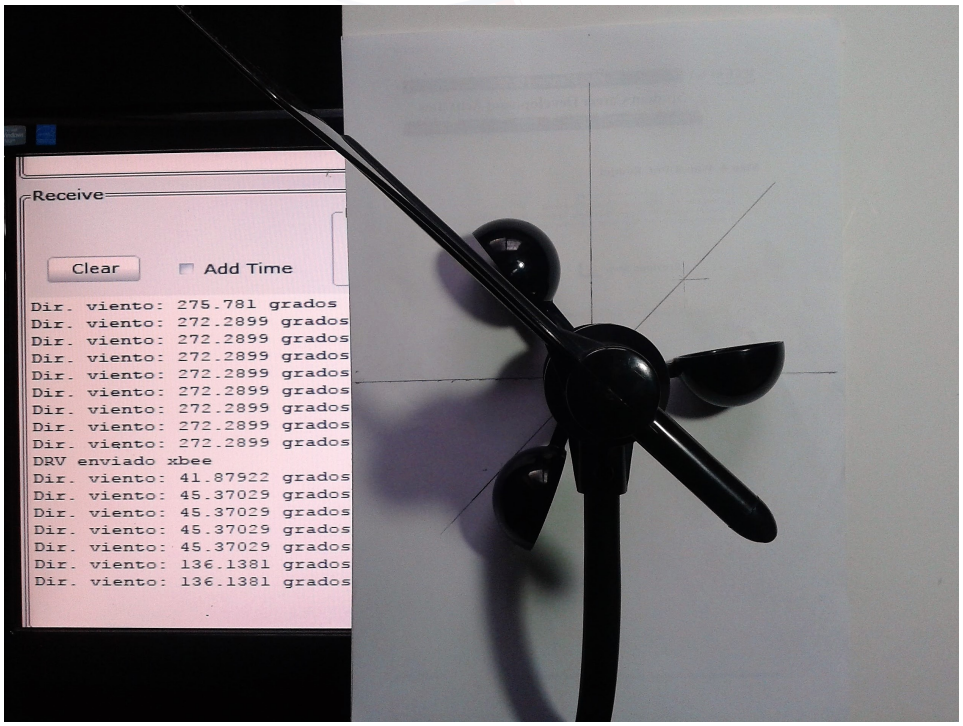


Figura G.6: Dirección sureste. Izquierda: Registro de datos. Derecha: Posición del sensor.

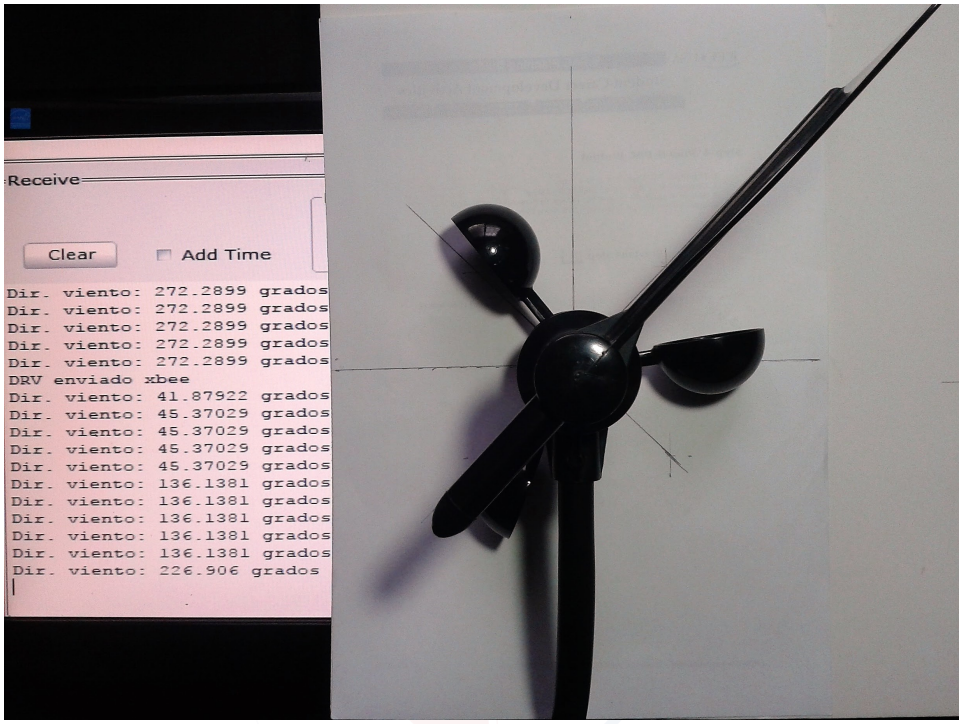


Figura G.7: Dirección suroeste. Izquierda: Registro de datos. Derecha: Posición del sensor.

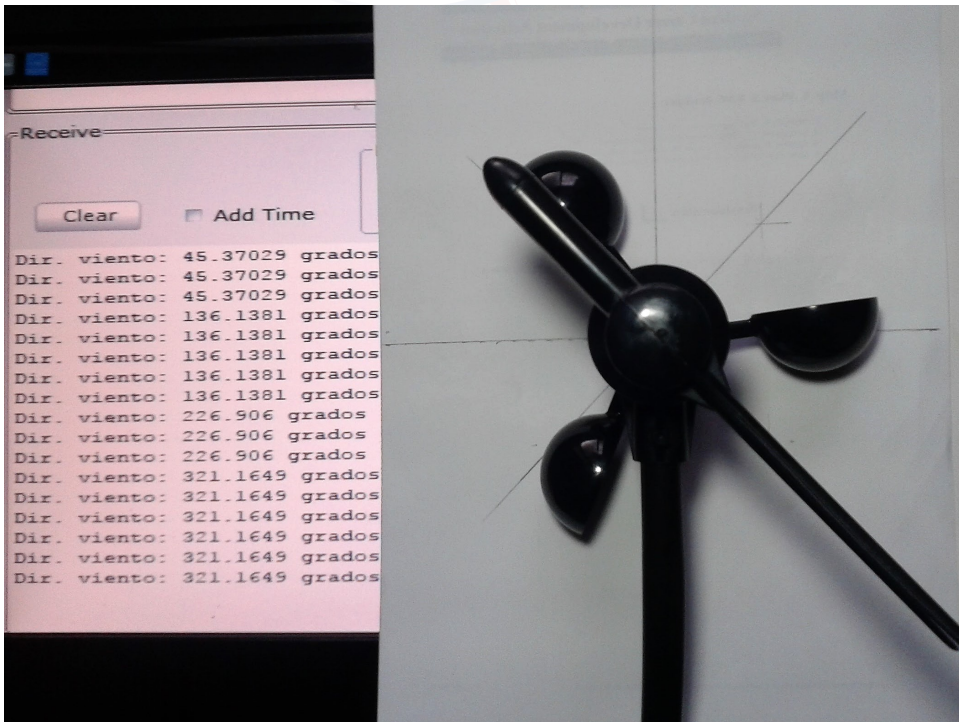


Figura G.8: Dirección noroeste. Izquierda: Registro de datos. Derecha: Posición del sensor.