Universidad de La Laguna
Escuela Superior de Ingeniería y Tecnología
Departamento de Ingeniería Industrial

Trabajo de Fin de Grado

Diseño de una estación AIS portátil

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Estudiante:

Jesús Galván Santos

Tutor: Fernando Luis Rosa González

13 de septiembre de 2022

Universidad de La Laguna

Escuela Superior de Ingeniería y Tecnología
Departamento de Ingeniería Industrial

IMPRESO DE AUTORIZACIÓN DEL TRABAJO DE FIN DE GRADO POR EL TUTOR

Curso 2021/2022

El Dr. D. **Fernando Luis Rosa González**, con D.N.I. 43611314-W, como tutor del estudiante D. **Jesús Galván Santos** en el Trabajo de Fin de Grado titulado

Diseño de una estación AIS portátil

da su autorización, acreditada por la firma electrónica de este documento, para la presentación y defensa de dicho proyecto, a la vez que confirma que el estudiante ha cumplido con los objetivos generales y particulares que lleva consigo la realización del mismo.

La Laguna, 13 de septiembre de 2022.

“Para los que creen, para los que no creen. Para los humildes y para los orgullosos de serlo. Reza consciente de que todo viene de Dios. Trabaja como si todo dependiese de ti. Que tu trabajo te cueste oración y que tu oración te cueste trabajo.”

- Anónimo.

Agradecimientos

En primer lugar deseo expresar mi agradecimiento a mi familia, cuya preocupación y ánimos siempre estuvieron en cada encuentro, en especial, a mis padres por darme el apoyo, el ejemplo y lo más importante, el amor. Por enseñarme a perseverar ante las dificultades. Por guiarme cuando la noche era más oscura. Gracias una vez más.
A mis hermanos, por ser el lugar que más me da fuerza. Durante esta etapa, por muchas dificultades que hubieran, con un momento juntos era suficiente como para hacer salir el Sol de nuevo.
A mi tutor, el doctor Fernando Luis Rosa González, por confiar en mí, por ayudarme siempre que lo necesitaba, por saber disipar mis preocupaciones en cada tutoría.
A mis compañeros de clase, por todo el apoyo. Ellos son el mejor regalo de la Universidad. Ha sido un honor compartir este camino.
A mi amigo Daniel Dóniz, por toda la ayuda, por estar siempre disponible, por sus buenos consejos. Gracias de todo corazón.
Y por último, a todas las pruebas y dificultades que me encontré en el camino, ya que sin ellas, este momento no sería tan dulce.

Resumen

En el presente Trabajo de Final de Grado, se ha diseñado una estación AIS portátil, capaz de recibir los mensajes AIS emitidos por los barcos que se encuentren dentro del radio de recepción de la estación. Además, la estación recogerá datos climatológicos y su propia ubicación.
En el primer tomo se detallarán los fundamentos en los que está basado el proyecto, su diseño, su montaje e implementación y un capítulo final con los resultados obtenidos. Este proyecto se ha basado en la tecnología SDR (Software Defined Radio) para la recepeción, sintonización y demodulación de las señales recibidas. A partir de los datos recogidos, el dispositivo elabora un registro y muestra por pantalla los mismos a tiempo real.
El segundo tomo recoge las mediciones y el presupuesto de la elaboración de la estación AIS portátil. Finalmente, en el tercer tomo se muestran las hojas de datos de los componentes de la estación.

Abstract

For this Final Degree Project, a portable AIS station has been designed, capable of receiving AIS messages emitted by ships that are within the reception radius of the station. In addition, the station will collect weather data, and its own location. The project's first volume will detail the fundamentals on which the project is based, with chapters on its design, assembly and implementation, and a final chapter with the results obtained. This project has been based on SDR (Software Defined Radio) technology for the reception, tuning and demodulation of the received signals. Based on the data collected, the device creates a record and displays it on the screen in real time.
The second volume collects the measurements and the budget for the development of the portable AIS station. Finally, in the third volume the data sheets of the station components are shown.

Universidad de La Laguna
Escuela Superior de Ingeniería y Tecnología
Departamento de Ingeniería Industrial

Trabajo de Fin de Grado

Diseño de una estación AIS portátil

TOMO I

Memoria

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Estudiante:

Jesús Galván Santos

Tutor: Fernando Luis Rosa González

13 de septiembre de 2022

Índice general

Índice de figuras

Capítulo 1

Introducción

El objetivo del presente Trabajo de Fin de Grado es el diseño y desarrollo de una estación receptora AIS portátil. Este dispositivo será capaz de recibir los mensajes AIS que emiten los buques para así obtener información útil sobre los mismos (ubicación, velocidad, mmsi, etc.). Para ello, el equipo necesitará una antena capaz de recibir emisiones en VHF (Very High Frequency), ya que el standard AIS emite a una frecuencia de 162 MHz, un dispositivo SDR (Software Defined Radio) con el que a través de software, poder procesar radio. Un programa para demodular la señal y leer los mensajes AIS y una batería que permita al sistema ser portátil.

La motivación de este trabajo es aportar a la preservación de las zonas protegidas de canarias. Más en particular a la zona especial de conservación (ZEC) de la "Franja Teno-Rasca". Ya que esta zona es un habitat de cetaceos, es necesario medir la presencia de embarcaciones tanto las que se usan para su avistamiento como las que introducen energía sonora en el medio. Esta actividad provoca alteraciones en las rutinas de estos mamíferos, ya sea por la misma presencia humana o por el sonido que las embarcaciones puedan causar. Aparte de estas razones, que aunque esten dentro de un marco legal puede causar daño para la fauna, existen otras que, estando prohibidas, también pueden darse. Efectuar cualquier tipo de vertido en el mar, fondear sobre praderas de fanerógamas marinas, la captura de especies o la alimentación de las mismas y cualquier comportamiento que pueda causar un daño a los cetáceos son ejemplos de actividades prohibidas que también se realizan desde embarcaciones. En esta zona especial de conservación se encuentran hábitats naturales como bancos de arena, arrecifes, cuevas marinas y especies de interés comunitario como la tortuga boba (*Caretta caretta*), la tortuga verde (*Chelonia mydas*) o el delfín mular (*Tursiops truncatus*). Esta espacio ZEC está situado al oeste de la isla de Tenerife y cubre una superficie de 69.489,68 hectáreas. Se encuentra declarado desde septiembre de 2011 como una zona de

especial conservación (ZEC) y figura contenida en la Red Natura 2000 con la finalidad de asegurar la supervivencia de las especies y los hábitats naturales. Las especiales condiciones de aguas cálidas junto con las grandes profundidades cercanas a la costa, ofrecen unas características inigualables para la aparición de cetáceos grandes y medianos, registrándose hasta 22 especies distintas.

Es por ello que vemos adecuado el uso del sistema AIS, ya que nos puede dar mucha información sobre los buques que llevan a cabo este tipo de actividades. Dado que nuestro sistema será portátil, podremos movernos en busqueda de la recepción de señales AIS no solo en esta zona sino también en otras localizaciones.

1.1. Estructura de la memoria

El documento se estructura en tres tomos, el Tomo I recoje la Memoria, el segundo Tomo II es el presupuesto y por último hemos agrupado todo aquel contenido importante pero prescindible en la explicación de la memoria, en un Tomo III de Anexos.

El Tomo de la memoria, tras este primer capítulo, dónde se ha descrito la motivación y el propósito del proyecto, se encuentra el Capítulo 2 con los fundamentos del sistema AIS y de las técnicas de radio definida por software, básicos para el desarrollo de la estación diseñada. Los dos siguientes capítulos son el centro del documento. En el Capítulo 3 se expone el diseño de la estación portatil y en el Capítulo 4 la fabricación y el código de software utilizado. Terminamos el Tomo de la memoria con algunas pruebas de campo reales del instrumento y los resultados obtenidos que se recojen en el Capítulo 5. El Tomo de presupuesto recoje una evaluación económica del desarrollo del proyecto y en los Anexos del último Tomo se recojen DataSheets y otra información interesante para la reproducción del proyecto por el ingeniero interesado.

En el siguiente capítulo se abordan los fundamentos del proyecto.

Capítulo 2

Fundamentos

Para diseñar, montar y fabricar una estación AIS portátil es necesario conocer que los Sistemas de Identificación Automática (AIS) tienen como objetivo fundamental la comunicación entre buques y antenas terrestres para prevenir posibles accidentes. Esta comunicación es por medio de radiofrecuencia de modo que comenzaremos revisando el espectro radioeléctrico para luego hacer un poco de la historia que acompaña el desarrollo de este sistema. A continuación describiremos el sistema AIS tal como se usará en este proyecto y en la última sección se realizará una definición del sistema de radio definida por software (SDR). La radio definida por software constituye un cambio de paradigma en el diseño de este tipo de instrumentos.

El espectro radioeléctrico [2] o de radiofrecuencia es la porción del espectro electromagnético que se utiliza para las radiocomunicaciones. A su vez, se divide en 13 bandas de radiofrecuencia, siendo la frecuencia de la onda (y por consiguiente también su longitud de onda) la que se utiliza para definir a cada una de ellas. El espectro completo comprende desde los 3 kHz hasta los 300 GHz donde cada una de las bandas tiene una utilidad determinada. Como se ve en la Figura 2.1 la radiofrecuencia empieza en frecuencias bajas, pasando por Radio AM de 535 kHz a 1700 kHz, Radio de banda ciudadana, Radio FM, televisión VHF, Bluetooth, hornos de microondas, sistemas de GPS o comunicación satelital. El sistema AIS utiliza las frecuencias 161.975 MHz y 162.025 MHz, por lo que se encuentra en la banda VHF (Very High Frecuency), que ocupa el rango de frecuencias de 30 MHz hasta 300 MHz, y que corresponden a los canales 87B y 88B respectivamente, de la banda de VHF marina.

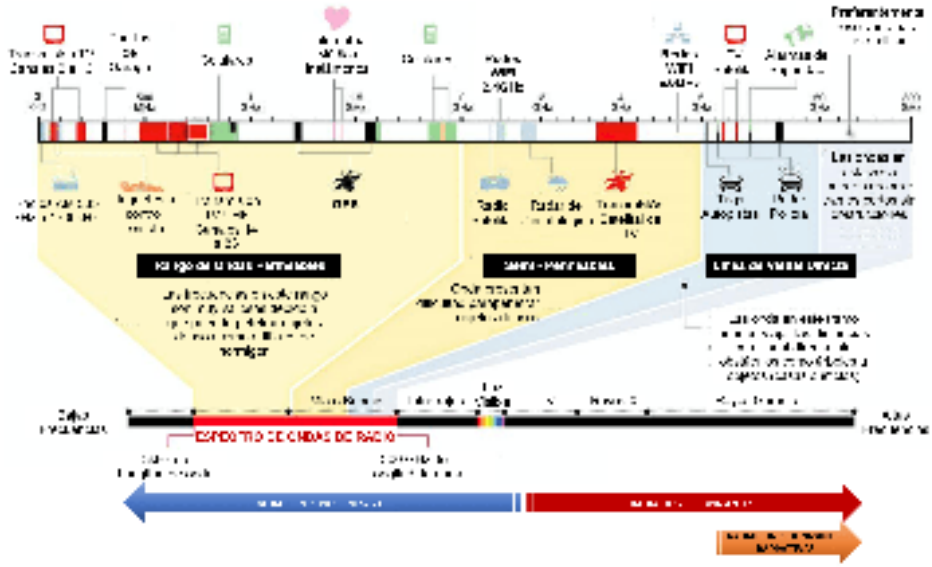


Figura 2.1: División en bandas del espectro radioeléctrico dentro del espectro electromagnético.

2.1. Historia y normativa para AIS

A partir de la catástrofe del Titanic en 1912, los gobiernos miraron con mayor preocupación la seguridad marítima, y como respuesta crearon el Convenio SOLAS (Safety of Life at Sea) cuya primera versión tuvo lugar en 1914, naciendo así el convenio más importante de todos los tratados internacionales sobre la seguridad de los buques [6]. En 1948, la ONU creó un organismo especializado llamado OMI (Organización Marítima Internacional) con la intención de mejorar la seguridad marítima, prevenir la contaminación, regular las normas del Convenio SOLAS y establecer un marco para que todos los países miembros cumplan con esta normativa. Posteriormente hubo cuatro versiones más, la última de ellas y actualmente en vigor, elaborada en 1974. El objetivo principal de este convenio es especificar las normas de construcción y equipamiento de los busques, promoviendo así la seguridad de los mismos y de los tripulantes.

Pero no fue hasta 1991 cuando se presentó el primero de los documentos sobre las características técnicas propias del estándar AIS. Posteriormente, en 2002 la OMI comenzó a exigir el uso obligatorio de este sistema para las embarcaciones cuyo tonalaje sea superior a 500 toneladas, 300 toneladas para los que realicen viajes internacionales y para todos los buques de pasajeros. Este sistema se ha extendido internacionalmente, ya que es de acceso libre, y mejora los anteriores sistemas de identificación. Gracias a la información (posición, velocidad, tamaño, etc.) que

este sistema proporciona a los buques, estos cuentan con mayor seguridad en sus actividades, evitando así posibles colisiones.

El estándar AIS presenta una serie de ventajas tales como:

- Facilidad de implementar, mantener y actualizar.
- Bajo coste operativo.
- Estandarización internacional.
- Cobertura entre 20 y 100 millas náuticas dependiendo de obstáculos y de la altura de las antenas.
- Ayuda a la seguridad de los barcos.

Los sistemas AIS se rigen por las recomendaciones de la UIT (Unión Internacional de Telecomunicaciones), que expone las características técnicas de un sistema de identificación automático (AIS) mediante acceso múltiple por división de tiempo (AMDT) en la banda de ondas métricas del servicio móvil marítimo.

La Asamblea de Radiocomunicaciones de la UIT recomienda que el AIS se defina de conformidad con las características de funcionamiento que figuran en los distintos anexos de la recomendación UIT-R M.1371-5 [8] considerando que:

1. la OMI (Organización Marítima Internacional) mantiene la formulación del requisito de un sistema de AIS universal a bordo de barcos;

2. el empleo universal del mismo permitiría el intercambio eficaz de datos de navegación entre barcos y estaciones, mejorando así la seguridad de la navegación;

3. un sistema AMDTA (acceso múltiple por división en el tiempo autoorganizado) satisfaría a todos los usuarios y cumpliría los probables requisitos futuros de utilización eficaz del espectro;

4. dicho sistema se podría utilizar para otras cuestiones distintas a su próposito principal;

5. el sistema funcionaría de manera autónoma, automática y continua;

6. el sistema sería capaz de ampliarse, para responder así a la futura expansión de usuarios;

El sistema AIS fue aprobado por la OMI en el 2002. Desde 2007 el estándar AIS es obligatorio para los buques adheridos al Convenio SOLAS ("Safety of Life at

Sea") que tengan alguna de las características que se detallan a continuación: Buques con arqueo bruto superior a 500 TRB, buques en viajes internacionales con arqueo bruto superior a 300 TRB y todos los buques de pasajeros, independientemente de su tamaño. Posteriormente, el 23 de Abril de 2009 se publicó la Directiva 2009/17/CE [3], con el motivo de crear un sistema comunitario de seguimiento e información sobre el tráfico marítimo, estableciendo el siguiente calendario de implementación:

- Buques pesqueros de eslora total superior o igual a 24 metros e inferior a 45 metros, desde 2012.
- Buques pesqueros de eslora total superior o igual a 18 metros e inferior a 24 metros, desde 2013.
- Buques pesqueros de eslora total superior o igual a 15 metros e inferior a 18 metros, desde 2014.

Para las embarcaciones que no están sujetas al Convenio SOLAS y que no sean embarcaciones de pesca, la Comisión Europea recomienda la implantación del sistema AIS para incrementar la seguridad de la navegación.

2.2. AIS

Los Sistemas de Identificación Automática (AIS) tienen como objetivo fundamental la comunicación entre buques y antenas terrestres para prevenir posibles accidentes. En esta comunicación se envían una serie de datos que aportan información referente a la identificación, posición, y otras cuestiones relevantes. Esta comunicación facilita la prevención de colisiones entre buques y el seguimiento de los mismos. Los sistemas AIS transmiten mediante un emisor VHF, en las frecuencias 161.975 MHz y 162.025 MHz de la banda marina en una modulación GMSK (Modulación de Desplazamiento Mínimo Gaussiano) posteriormente detallada. Este sistema utiliza un protocolo de control de enlace de datos de alto nivel (HDLC). En la Figura 2.2 se muestra un esquema del intercambio de señales radioeléctricas AIS entre barcos y estaciones.

Existen dos tipos de sistemas AIS, de clase A y de clase B. La clase A es obligatoria para los buques que estén bajo las normativas IMO y SOLAS. Trabaja en una gama de frecuencias desde 156,025 MHz a 162,025 MHz. Su protocolo de datos es el SOTDMA y transmiten en un intervalo de 2 a 10 segundos. Su alcance ronda las 50 millas y está establecido que su emisor tenga una potencia máxima de 12,5 W. La clase A tiene preferencia a la hora de emitir ante la clase B. El tipo B es

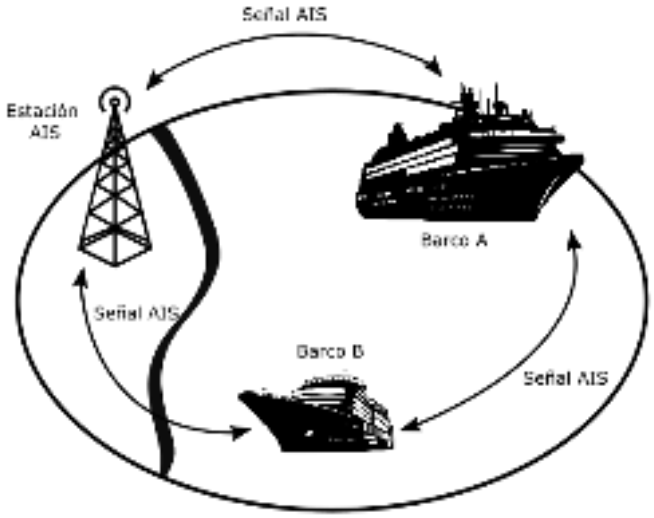


Figura 2.2: Flujo de información entre barcos y estaciones.

más económico, tiene una potencia máxima establecida de 2 W y su protocolo de datos es el CSTDMA. Transmiten cada 30 segundos en una gama de frecuencias de 161,975 MHz y 162,025 MHz. Tiene un alcance aproximado de 12 millas. Este tipo de sistema AIS, esta pensado para las embarcaciones menores y de recreo, ya que por su carga y su tamaño, el riesgo de un accidente grave es menor y no están obligadas a incorporar el sistema AIS, sin embargo es recomendable.

2.2.1. Arquitectura del estándar AIS

La arquitectura del estándar AIS se definide en la recomendación UIT-R M.1371 [8] en la que se describen 4 capas: física, de enlace, de red y de transporte como se muestra en el equema de la Figura 2.3.

La capa física convierte el paquete de datos codificado en NRZI (Non Return Zero Inverted) en una señal digital GMSK analógica.

La capa de enlace está dividida en 3 subcapas, la entidad de gestión de enlace (link management entity o LME), los servicios de enlace de datos (data link service o DLS) y el control de acceso al medio (medium access control o MAC).La subcapa LME ensambla los bits de los mensajes AIS, posteriormente la subcapa DLS calcula la secuencia de verificación de la trama para los bits del mensaje AIS y por último la subcapa MAC proporciona un medio para la transferencia de datos,

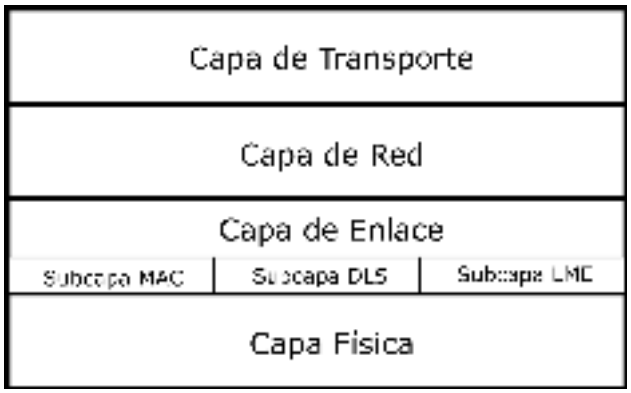


Figura 2.3: Arquitectura en capas del estándar AIS.

utilizando un esquema de acceso múltiple por división de tiempo (AMDT).

La capa de red organiza los tiempos y las prioridades de transmisión entre los canales y atiende a su saturación.

La capa de transporte se encarga de empaquetar en un tamaño adecuado y secuenciar los datos provenientes de los buques.

Capa Física

Como se ha mencionado anteriormente, la capa física se encarga de transferir una línea de bits generados por el micropocesador a la capa de enlace. En este apartado se describirán los parámetros del sistema AIS para que esto se produzca.

Las frecuencias de emisión y recepción de los sistemas AIS se encuentran encuadradas en la banda de frecuencias atribuidas al servicio móvil marítimo de ondas métricas según el Apéndice 18 del Reglamento de Radiocomunicaciones de la UIT. Dicha banda alberga un rango desde los 156.025 MHz hasta los 162,025 MHz, donde los sistemas AIS se encuentran a la cola de dicho rango, concretamente, los sistemas AIS clase 1 trabajan en la frecuencia 161.975 MHz y los sistemas de AIS clase 2 en la frecuencia 162.025 MHz, siendo los canales 87B y 88B de la banda marina respectivamente. Todos los canales que se encuentran dentro de la banda de frecuencias atribuidas al servicio móvil marítimo de ondas métricas tienen una separación entre sí de 25 KHz por lo que este también es el ancho de banda de cada canal.

El estándar AIS funciona a una velocidad binaria de 9600 bits/s, teniendo una secuencia de acondicionamiento de 24 bits. Otras características a incluir, y que posteriormente se describirán con mayor detalle son la modulación y la codifica-

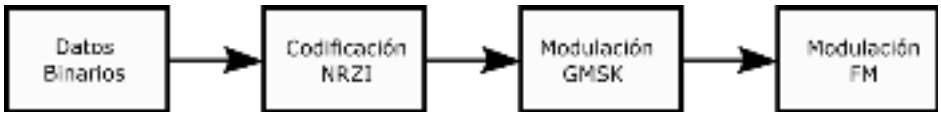


Figura 2.4: Etapas de procesamiento en la capa física.

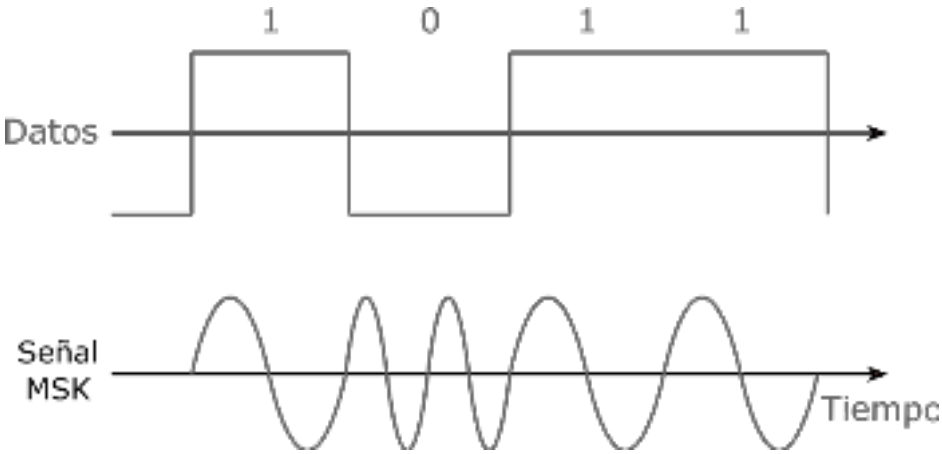


Figura 2.5: Modulación de una secuencia de unos y ceros lógicos en MSK.

ción de datos, siendo estas GMSK/FM y NRZI respectivamente. La información binaria proveniente del microprocesador debe ser codificada en NRZI y posteriormente codificada en GMSK antes de efectuar la modulación de frecuencia del transmisor, en las etapas que se muestran en la Figura 2.4.

Modulación GMSK Los sistemas AIS utilizan la modulación GMSK [1]. Su nombre completo es Modulación por Desplazamiento Mínimo Gaussiano (Gaussian Minimun Shift Keying) y es una forma de modulación de frecuencia que se utiliza en los sistemas de comunicaciones por radio. Esta modulación se basa en MSK, que es en sí misma una forma de modulación por desplazamiento de frecuencia de fase continua. La GMSK y la MSK son modulaciones que se conocen como esquemas de fase continua. No hay discontinuidades de fase, ya que los cambios de frecuencia ocurren en los puntos de cruce por el cero de la portadora. Esto surge a raíz del factor único de MSK, que dicta que la diferencia mínima de frecuencia que debe haber entre el 1 y el 0 lógico es de un índice de modulación de 0,5. La señal MSK se muestra para una señal binaria en la Figura 2.5 donde se ve la continuidad de la fase en las transiciones de los datos [11].

Para reducir el ancho de banda de una señal MSK, se puede utilizar un filtro paso bajo a la señal antes de aplicarla a la portadora. Este filtro debe tener una serie de requisitos tales como tener un corte agudo, un ancho de banda estrecho y su

respuesta al impulso no debe mostrar sobreimpulso. Uno de los filtros para este tipo de cometido es el gaussiano. De esta forma, una señal MSK se convierte en una GMSK.

Matemáticamente, la respuesta temporal del filtro ante un impulso viene dada por la ecuación 2.1 en la que B_b es el ancho de banda del filtro gaussiano, T es el tiempo de bit y B_t es el ancho de banda normalizado. La función $Q(x)$ tiene forma integral y se muestra en la ecuacion 2.2.

$$g(t) = \frac{1}{2T}\left[Q\left(2\pi B_t \frac{t-\frac{t}{2}}{\sqrt{\ln 2}}\right) - Q\left(2\pi B_t \frac{t+\frac{t}{2}}{\sqrt{\ln 2}}\right)\right] \forall 0 \leq B_b T \leq \infty \tag{2.1}$$

$$Q(x) = \int \frac{1}{\sqrt{2}} \exp\left(-\frac{x^2}{2}\right) \mathrm{d}x \tag{2.2}$$

La ecuación 2.3 representa la señal modulada en MSK, $s(t)$, donde $a_I(t)$ es la señal del canal de muestras I, $a_Q(t)$ en la señal del canal de muestas Q y f_c es la frecuencia de la portadora. Si sobre esa ecuación aplicamos identidades trigonométricas podemos representar la señal modulada como muestra la ecuación 2.4. Donde $b_k(t) = +1$ si $a_I(t) = a_Q(t)$ y $b_k(t) = -1$ si $a_I(t) = -a_Q(t)$, mientras que $\phi_k = 0$ si $a_I(t) = 1$ y $\phi_k = \pi$ en cualquier otro caso.

$$s(t) = a_I(t)\left(\frac{\pi t}{2T}\right)\cos(2\pi f_c t) - a_Q(t)\left(\frac{\pi t}{2T}\right)\sin(2\pi f_c t) \tag{2.3}$$

$$s(t) = \cos\left[2\pi f_c t + b_k(t)\frac{\pi t}{2T} + \phi_k\right] \tag{2.4}$$

Hay dos maneras de generar una modulación GMSK. La forma más directa es aplicar un filtro gaussiano a la señal binaria y luego aplicarle un modulador de frecuencia cuyo índice de modulación sea 0,5, como se muestra en la Figura 2.6. Para conseguir que la señal sea continua y que no haya saltos de fase entre un estado de la línea (por ejemplo para la señal 0) y el otro (la señal del 1), se hace uso de un VCO (Voltage-Controlled Oscillator) cuya función es la de cambiar la frecuencia de la señal generada según el voltaje de entrada. De esta manera se consigue que la señal no tenga saltos de fase entre cambios de estado y que este cambio sea suave. En la práctica se ha visto que este método no es el más adecuado, ya que al tener que establecer un índice de modulación exactamente igual a 0,5, este método analógico hace que se desvíen las tolerancias de los componentes y que este índice no se establezca con exactitud.

Un segundo método y más ampliamente utilizado es el modulador en cuadratura, que se muestra en la Figura 2.7. Este término dicta que la fase de la señal está

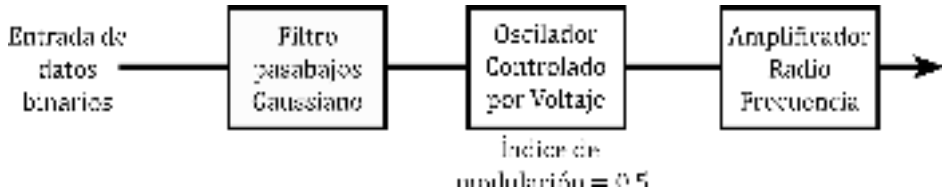


Figura 2.6: Diagrama de la modulación GMSK utilizando un VCO.

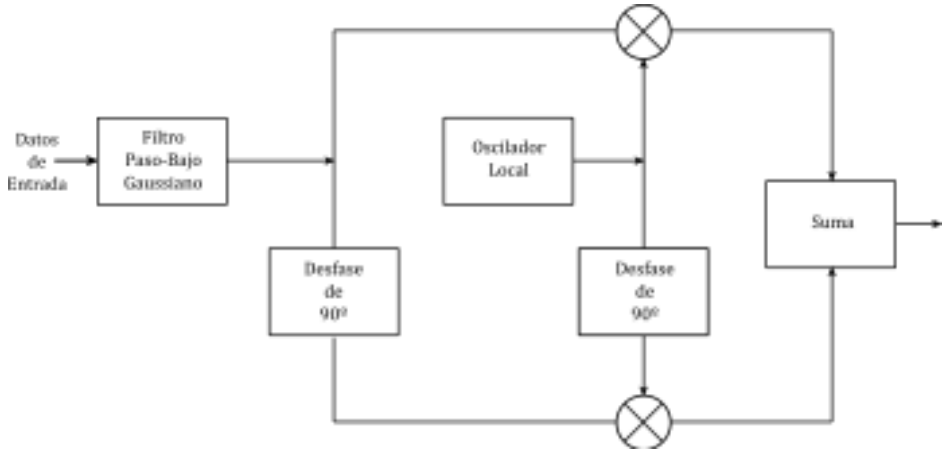


Figura 2.7: Diagrama de la modulación GMSK utilizando un modulador IQ.

en cuadratura o a 90 grados con respecto a otra señal. A menudo este modulador se conoce como modulador IQ. De esta manera, el índice de modulación se puede mantener exactamente en 0,5 sin necesidad de realizar ningún ajuste. Para la demodulación, la misma técnica se puede utilizar a la inversa.

La modulación GMSK presenta varias ventajas con respecto a otras formas de modulación. Una es la alta eficiencia espectral en comparación a otros modos de cambio de frecuencia, ya que requiere poco ancho de banda. Otra ventaja a tener en cuenta es que no hay elementos de la señal que transporten variaciones de amplitud, por lo que puede amplificarse con un amplificador no lineal y así permanecer sin distorsiones. A su vez, al no haber variaciones de amplitud, es más resistente al ruido.

NZRI (Non Return Zero Inverted) es el tipo de esquema de codificación de línea que se emplea en los transpondedores AIS. La comunicación digital convencional suele estar compuesta por una línea de datos y una línea de reloj. Esto, en comunicaciones de alta velocidad, puede traer problemas, ya que cualquier desincronización entre estas dos líneas puede hacer que el receptor no cumpla con el tiempo de muestreo de datos y que ocurran errores en la comunicación. Por ello,

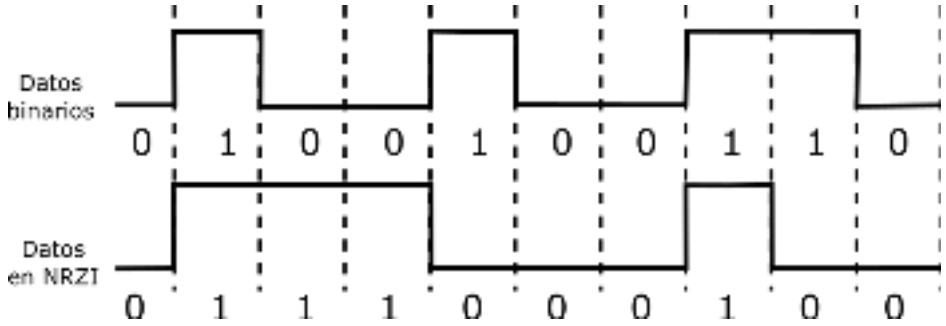


Figura 2.8: Ejemplo de codificación NRZI.

los esquemas de codificación como NRZ, Manchester o NRZI, fusionan la línea de reloj y los datos, creando una única línea. Como se puede ver en la Figura 2.8 la codificación en NRZI de cualquier línea de datos binarios se consigue simplemente creando una transición cuando hay un uno lógico.

Capa de Enlace

La capa de enlace se encarga del empaquetamiento así como de la detección y corrección de los errores de transferencia de los datos. Como se ha mencionado anteriormente, se divide en 3 subcapas, MAC, DLS y LME que se explican a continuación.

La subcapa MAC (Medium Access Control) brinda al sistema AIS una manera de tener acceso a la transferencia de datos. El método utilizado es el Acceso Múltiple por División de Tiempo (AMDT) usando el Tiempo Universal Coordinado (UTC). El UTC es el principal estándar de tiempo por el que se regulan los relojes a nivel internacional y tener acceso al mismo es importante ya que el sistema de multiplexación permite o restringe la transferencia de datos a los transpondedores AIS según un intervalo de tiempo determinado. Por ello, todos los transpondedores AIS que se encuentren dentro de una determinada área deberán estar sincronizados con el UTC. El sistema AIS cuenta con un protocolo específico para poder resincronizar las estaciones al UTC. Por orden de prioridad, si no se tiene acceso al UTC directamente, se tomarán las siguientes opciones como referencia: una estación con hora UTC, una estación de base designada como semáforo, otras estaciones sincronizadas con la estación base o una estación móvil designada como semáforo.

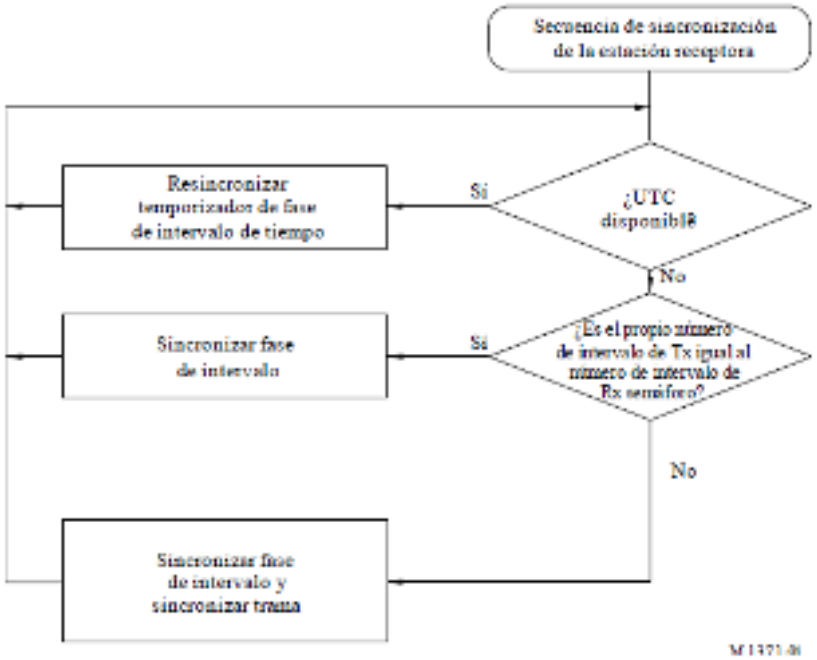


Figura 2.9: Esquema de sincronización del UTC de una estación receptora.

Dado que el objeto de este proyecto es el diseño y desarrollo de una estación receptora AIS, no se desarrollará el protocolo para las estaciones transmisoras. El correspondiente a las estaciones receptoras se encuentra en la Figura 2.9

La subcapa DLS aporta al sistema AIS los procedimientos de activación del enlace de datos y de transferencia, y desarrolla su formato. Para la activación del enlace de datos, la subcapa DLS, en base a la subcapa MAC (y por consiguiente a la disponibilidad de UTC), se estará a la escucha de un intervalo de tiempo disponible (esto se explicará con mayor detalle en la subcapa LME).

Para la transferencia de datos, la subcapa DLS, utiliza un protocolo orientado a bits basado en el control de alto nivel del enlace de datos (HDLC) según se especifica en La Norma 3309:1993 de la Organización Internacional de Normalización/Comisión Electrotécnica Internacional (ISO/CEI). Formado por 256 bits, la trama de datos esta formada con 6 partes: Secuencia de acondicionamiento, bandera de inicio, los datos, secuencia de verificación de la trama (FCS), bandera de fin y almacenamiento temporal, tal como se muestra en la Figura 2.10.

La secuencia de acondicionamiento sirve de preámbulo para la bandera de inicio y consta de 24 bits, 0 y 1 alternos. La bandera de inicio se emplea para la detección de un nuevo paquete de transmisión y tiene una longitud de 8 bits dispuestos de la

Figura 2.10: Partes de la trama de datos.

siguiente manera: 01111110. La carga útil, que es la información que los buques desean transmitir, tiene una longitud de 168 bits y es un mensaje NMEA como se explicará en el siguiente subapartado.

Para la secuencia de verificación de la trama se utiliza un polinomio de dieciseis bits de verificación por redundancia cíclica que calcula la suma de control marcada por la Norma 3309:1993 de la ISO/CEI. Estos deben ponerse a 1 al comienzo de la verificación. La bandera de fin es la misma que la bandera de inicio: 01111110. El almacenamiento temporal consta generalmente de 24 bits y son utilizados para relleno de bits (mensajes), retardo por distancia y fluctuación de sincronización.

Los mensajes NMEA 0183 [5] que utiliza AIS forman parte de una especificación para la comunicación entre aparatos electrónicos marinos tales como anemómetros, profundímetros, receptores GPS entre otros. Su definición la realiza la organización estadounidense National Marine Electronics Association (NMEA) y será revisado y sustituido por el protocolo NMEA 2000 aunque de momento sigue siendo la norma más utilizada. Utiliza un protocolo de comunicación en serie simple con un código ASCII de 6 bits formando sentencias. En la norma se recoge con la siguiente frase: "Los datos son transmitidos en una sentencia desde un emisor simultáneamente a varios receptores".

El protocolo NMEA 0183 utiliza el tipo de mensaje al principio de la sentencia. Para el caso de las señales AIS, el tipo de mensaje es **!AIVDM**. A continuación en la sentencia se indica el número de líneas, el número de fragmento, el canal de radio utilizado, la carga útil, la marca de final del mensaje y el checksum.

En la carga útil, se encuentran los datos del buque que el mismo desea transmitir y los posibles campos se muestran en la Tabla 2.1.

Un ejemplo de sentencia NMEA es,

!AIVDM,1,1,,A,13EqPr0P00Nm”j@BJGP0?wB0D0B,0*78.

y en este mensaje podemos distinguir los campos de la sentencia descritos anteriormente:

Tabla 2.1: Campos en la carga útil de un mensaje NMEA 0183 AIS.

Campos en la carga útil	
Fecha y hora	MMSI
Nombre	Latitud
Longitud	Velocidad
Rumbo	Estado
Tipo de embarcación	Destino
Eslora	Tonelaje
Puerto actual	Puerto anterior
Fecha de llegada	-

- Tipo de mensaje NMEA (AIS): **!AIVDM**
- Número de líneas del mensaje: **1**
- Número del fragmento: **1**
- Canal de radio utilizado: **A**
- Carga útil: **13EqPr0P00Nm"j@BJGP0?wB0D0B**
- Fin de datos: **0***
- CheckSum NMEA: **78**

La subcapa LME es la encargada de controlar el funcionamiento de las subcapas DLS y MAC y de la capa física. Existen 4 sistemas distintos para controlar el acceso al medio para la transferencia de datos, que son subclases del sistema de Acceso Múltiple por División de Tiempo (AMDT). Este sistema se basa en proporcionar a cada dispositivo, un intervalo de tiempo en el cual puedan transmitir, evitando así que los transpondedores transmitan al mismo tiempo, ya que puede ser fuente de interferencias y pérdida de información. En el sistema AMDT, los transpondedores, de manera automática, saben cómo reclamar y reservar la ranura de tiempo para transmitir y se organizan en caso de haber una disputa con otro dispositivo que desee emitir, reservándole a este último una ranura disponible. Este sistema se ha diseñado para dar prioridad a las embarcaciones que estén más cerca entre sí, dado el mayor riesgo que esto supone.

Cada uno de estos subsistemas tiene una aplicación que depende del momento y del modo en que se utilice el sistema AIS. Los subsistemas AMDT son: Incremental (AMDTI), Autoorganizado (AMDTA), Acceso Fijo (AMDTAF) y Acceso Aleatorio (AMDTAA).

Para transmitir en un determinado intervalo, previamente se han de escoger una serie de intervalos candidatos. Debe haber siempre 4 intervalos candidatos para

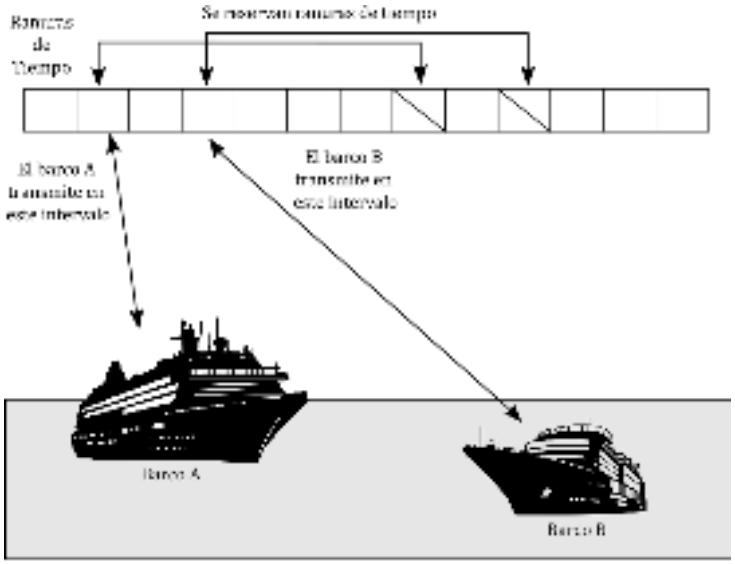


Figura 2.11: Funcionamiento de un sistema AMDT.

elegir 1 entre ellos, y si la información que se desea transmitir no cabe en un intervalo de tiempo, este tendrá que ser el primero de una serie de intervalos consecutivos libres. El sistema AMDT recoge 2250 intervalos en 1 minuto, por lo que cada transpondedor AIS tiene que transmitir su mensaje en 25 ms (un intervalo). Por ello, como se muestra en la Figura 2.12, existen subdivisiones de tiempo dentro de un intervalo para enviar cada una de las partes de la trama de datos.

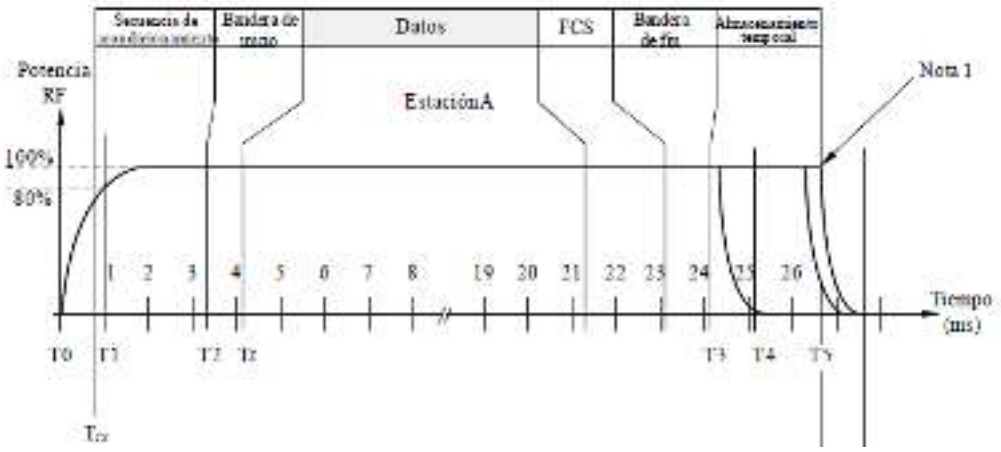


Figura 2.12: Tiempos de envío de los datos.

El subsistema AMDTI permite avisar sobre los intervalos de tiempo no repetitivos, con la excepción de que durante la entrada a la red de enlace de datos, los intervalos AMDTI deben de marcarse con el fin de que queden reservados para una trama extra. Este sistema se utiliza a la entrada de la red de enlace de datos, para modificaciones temporales y para anunciar mensajes de seguridad.

El subsistema AMDTAA se utiliza cuando una estación necesita utilizar un intervalo de tiempo que no se ha reservado. Estos mensajes se almacenan por orden de prioridad mediante un método FIFO. Se suele utilizar para el primer intervalo de transmisión o para mensajes no repetitivos.

El subsistema AMDTAF está reservado para las estaciones base y mensajes repetitivos. En estos mensajes la estación base informa al resto de estaciones sobre la asignación de intervalos e impide utilizar los mismos a otras estaciones cercanas.

El subsistema AMDTA está destinado a las estaciones móviles que trabajan en modo autónomo y tiene como objetivo resolver rápidamente las eventuales disputas con respecto a la asignación de intervalos de tiempo. Los mensajes con los que se utiliza este sistema es de carácter repetitivo y permite brindar al resto de estaciones un cuadro de vigilancia actualizado. Suele ser el modo más utilizado entre los transpondedores AIS.

Capa de Red

La capa Capa de Red es la encargada de establecer y mantener las conexiones en los canales, dar o quitar prioridad a los mensajes, distribuir los paquetes de datos entre los canales y resolver situaciones de congestión dentro del enlace de datos. El modo por defecto normal de funcionamiento constará de dos canales de recepción y 4 canales de emisión para las estaciones AIS móviles, mediante el cual el sistema AIS recibirá al mismo tiempo por el canal AIS 1 y por el canal AIS 2 en paralelo. Las áreas de funcionamiento se designan mediante un rectángulo con dos puntos de referencia. El primero de estos puntos será la dirección de coordenadas geográficas del ángulo noroeste y el segundo del ángulo suroeste.

Los informes de posición de otros buques se distribuirán a la interfaz de presentación, junto con las correcciones del sistema mundial de navegación por satélite (GNSS) recibidas y la posición propia, la cual deberá transmitirse por el enlace de datos. A la hora de establecer una jerarquía de prioridad de mensajes, el sistema

AIS establece 4 niveles: Máxima prioridad, donde se encuentran los mensajes críticos para la gestión del enlace de datos; Máxima prioridad de servicio, relativos a la seguridad; un nivel tercero designado para mensajes de asignación, interrogación y respuestas y en cuarto lugar el resto de mensajes.

En caso de haber una congestión en la red de enlace debido al gran número de transpondedores AIS que acceden a la misma, poniendo así en peligro la transmisión de información de seguridad, se deberá adoptar uno de los dos métodos que a continuación se desarrollarán.

Reutilización del intervalo de tiempo por la estación: A la hora de seleccionar nuevos intervalos de tiempo candidatos, en caso de haber menos de 4 intervalos disponibles, se reutilizará intencionalmente uno de los mismos con el fin de que el conjunto esté formado por 4 intervalos. Estos deben tomarse de las estaciones más distantes dentro del intervalo de selección.

Utilización de la asignación: Las estaciones base pueden usar asignaciones de intervalo para dirigir intervalos de tiempo usados por estaciones AIS móviles de buques de clase A hacia intervalos reservados para el sistema AMDTAF.

Las estaciones base se encargan de proporcionar la sincronización a las estaciones que no se hayan sincronizado previamente. Proporciona a los barcos la asignación de intervalos para transmitir y asigna periodicidades de informaciones a las estaciones móviles. Las estaciones base transmiten los mensajes de gestión del canal de comunicación y provee las correcciones GNSS.

Capa de Transporte

Esta capa se encarga de convertir los datos a paquetes de transmisión con un tamaño correcto, establece las secuencias de los mismos y marca la interfaz del protocolo con las capas superiores.

2.3. Tecnología SDR

La tecnología SDR (Software Defined Radio) [4] permite procesar señales de radio mediante el uso de *software* [9]. Tradicionalmente, la recepción de señales se conseguía mediante bobinas, condensadores, filtros, etc, sin embargo, hoy en día, es posible lograr lo mismo a través de los programas y el *hardware* adecuados

gracias a la tecnología SDR. Dependiendo del terminal, los dispositivos SDR nos brindan un ancho de banda bastante considerable, por regla general, alrededor de 500 KHz a 2 GHz, ancho más que suficiente para poder recibir señales AIS, ya que como se ha mencionado anteriormente, esta señal se emite a 162 MHz. Estos dispositivos son capaces de recibir cualquier tipo de onda que se encuentre en el espectro de radiofrecuencia, AM, FM o para el caso de las señales AIS, GMSK. Van conectados directamente a una antena, y sintonizan, procesan y demodulan la señal para posteriormente enviarla digitalizada a un ordenador.

2.3.1. Recepción IQ

Los sistemas SDR funcionan con las muestras IQ recibidas por un receptor. Estos receptores trabajan en 3 etapas: RF (Radiofrecuencia), IF (Frecuencias Intermedias) y banda base como se muestra en la Figura 2.13. La etapa de RF se encarga de la recepción y transmisión de señales. Recoge las señales recibidas por la antena, sintoniza la señal esperada y discrimina el resto de frecuencias. El proceso para la transmisión sería a la inversa. Para pasar a la etapa de frecuencias intermedias se hace uso de un mezclador y un oscilador local. El mezclador combina la señal recibida por la antena y la señal generada por el oscilador, una señal que es pura, sin modular y muy estable en frecuencia. A la salida del mezclador obtendremos una señal que puede ser la suma, la resta o partes de las señales originales, por lo que es necesario un filtro paso bajo para obtener la frecuencia intermedia. Una vez que la señal captada se encuentra a una frecuencia intermedia se digitaliza mediante conversores Analógico-Digitales para posteriormente, y una vez en banda base, realizar los procesos de demodulación, llevados a cabo mediante el uso de software.

Para demodular la señal ahora digital, se utiliza un método llamado mezclador en cuadratura, a través del cual se obtendrán las muestras IQ. Este método, como se muestra en la Figura 2.14 consiste en llevar la señal digital a dos mezcladores idénticos en paralelo. A su vez, a estos mezcladores se les inyecta una señal generada por el oscilador local, aunque distintas en fase. A un mezclador se le inyecta una señal en fase (seno) y al otro se le inyecta la misma desfasada noventa grados (coseno), obteniendo a su salida una señal en cuadratura. Estas dos señales I y Q, pasan individualmente por filtros pasobajo para eliminar las frecuencias no deseadas y a continuación son enviadas a la tarjeta de desarrollo.

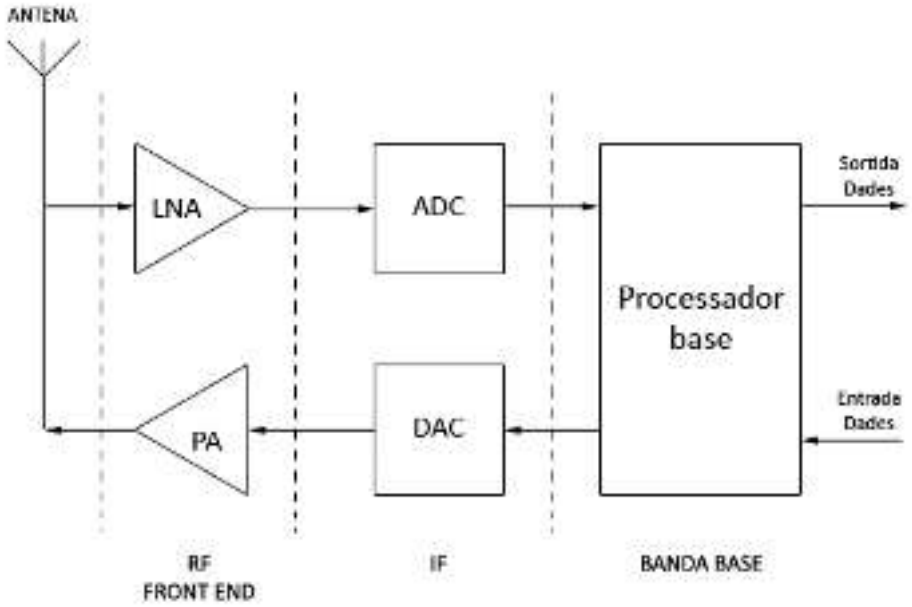


Figura 2.13: Etapas de un receptor–emisor IQ-SDR.

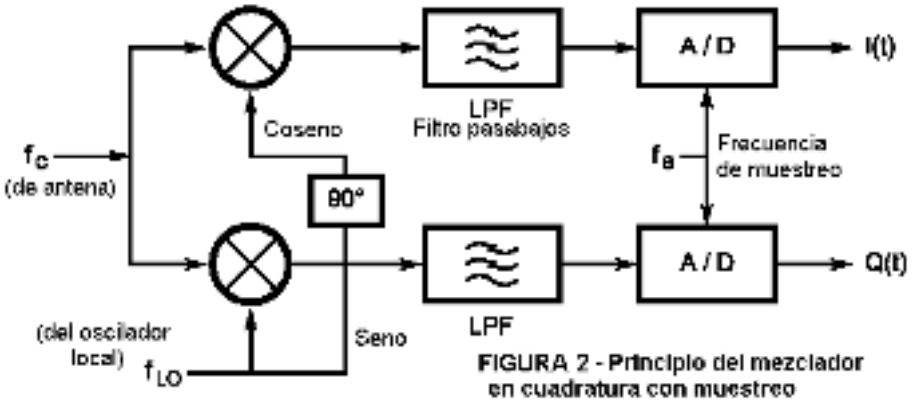


Figura 2.14: Diagrama de un mezclador en cuadratura.

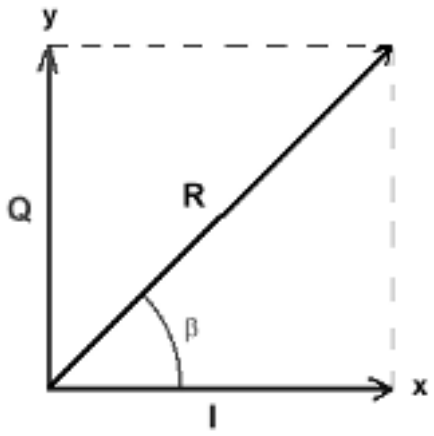


Figura 2.15: Componentes IQ representadas en ejes coordenados.

2.3.2. Radio definida por software

Una vez que hemos obtenido los componentes I y Q de una señal, debemos demodular la misma. Este proceso se hace mediante operaciones matemáticas en la tarjeta de desarrollo usando el software adecuado. Estas tarjetas pueden ser programadas para demodular, eliminar señales no deseadas, reducir el ruido, etc.

A modo de ejemplo se expondrá la demodulación de las señales AM Y FM. En un receptor que procese señales I y Q, al estar estas desfasadas 90º entre sí, si se representarán vectorialmente los componentes en dos ejes, los ejes I y Q, se representaría la señal compleja IQ como se muestra en la Figura 2.15.

Mediante este tipo de diagramas se pueden representar las componentes I y Q a modo de ejes, el vector A como la amplitud resultante y beta como ángulo de fase. Por tanto, para calcular la amplitud instantánea, basta con emplear el Teorema de Pitágoras, $A = \sqrt{I^2 + Q^2}$.

Para las modulaciones de fase, como la FM (modulación de frecuencia) o la PM (modulación de fase), el ángulo de fase de la portadora varía según la amplitud de la señal moduladora, por ello, la demodulación depende de las variaciones de fase de la señal de radiofrecuencia. Por ello, las amplitudes de las componentes IQ, no varían, pero si lo hace el ángulo de fase beta que podemos calcular como, $\beta = \arctan\left(\frac{Q}{I}\right)$

Capítulo 3

Diseño

En este capítulo, se describirán los pasos que se han seguido para diseñar una estación AIS portátil. Para diseñar una estación AIS, debemos conocer cuales son las etapas por las que pasa el mensaje AIS. Primero, existe una etapa de emisión, en la cual, los barcos mandan un mensaje, que como se ha mencionado anteriormente, se emite a una frecuencia de 162 MHz, con una modulación GMSK y cuyo protocolo de datos es el ATDMAA. Una vez que este mensaje se encuentra en el espectro de radiofrecuencia, otros barcos o estaciones con receptores AIS son capaces de recibirlo. En este punto entramos en la etapa de recepción, en la cual, los barcos o estaciones reciben las señales AIS. Este proyecto está basado en la tecnología SDR para recibir señales de radiofrecuencia, por que lo que se utilizará un receptor IQ, mediante el cual, se procesa la señal de radio a través de software para finalmente obtener las muestras IQ. Por último, entramos en la etapa de demodulación, en la que la unidad de control demodula las muestras IQ y muestra y guarda la información, ahora leíble para el usuario, enviada por los barcos.

En la Figura 3.1 se recoje un esquema de las partes del prototipo diseñado. Para aprovechar al máximo el potencial que nos brinda la tarjeta de desarrollo, a la estación AIS portátil se le añadirán sensores que nos aporten información sobre el momento exacto en el recibimos los mensajes AIS, como son la temperatura, la humedad, la presión y la posición de la estación. Toda esta información y los datos de los barcos de alrededor, será visible para el usuario gracias a una interfaz que saldrá por pantalla y a su vez, quedará registrada en una serie de ficheros, para el caso de que posteriormente a la recepción de mensajes, se puedan estudiar los datos recogidos. Por último, ya que una carácterística importante de esta estación es su portabilidad, se hará uso de una batería recargable que alimentará el dispositivo.

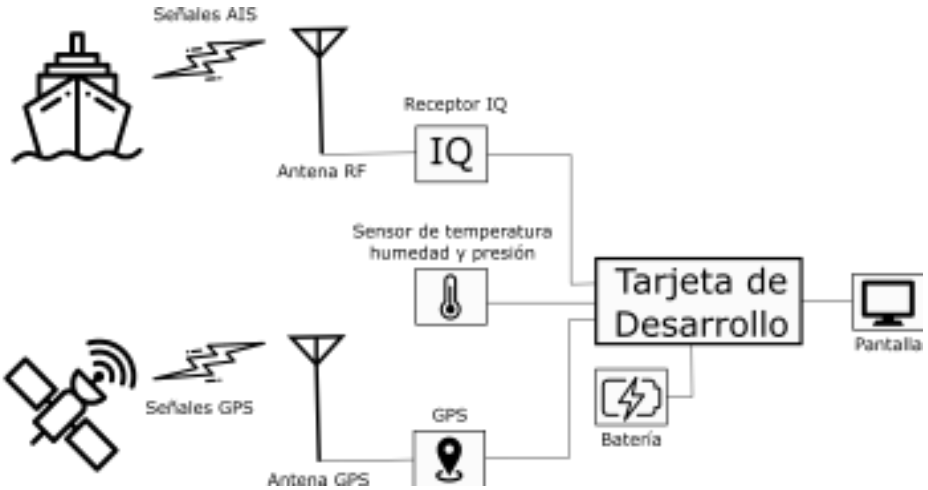


Figura 3.1: Esquema de la estación AIS portátil.

3.1. Recepción IQ

Los receptores IQ son una herramienta fundamental de la tecnología SDR. Estos dispositivos, mediante una antena, captan las señales de radiofrecuencia, filtran la banda de interés, la demodulan a la frecuencia intermedia para luego hacer el muestreo IQ. El receptor debe ser configurado con los parámetros de interés para lo que se necesita una unidad de control con un *software* adecuado.

3.1.1. Antena

Para recibir cualquier tipo de onda presente en el espectro de radiofrecuencia, es necesario una antena. En esta, por las variaciones que tenga la señal electromagnética recibida, aparecerán una serie de variaciones del orden de microvoltios, que posteriormente serán leídos por el receptor, el cual se explicará en el siguiente apartado.

Para dimensionar una antena es preciso conocer la onda electromagnética que se desea captar, cuyos parámetros más relevantes son la frecuencia y la longitud de onda. Estos, tienen una relación inversa entre sí, a mayor frecuencia menor longitud de onda, como se muestra en la ecuacion 3.1.

$$f = \frac{v}{\lambda}, \tag{3.1}$$

donde f es la frecuencia de la onda, λ la longitud de onda y v la velocidad de la onda. La velocidad de propagación de las ondas electromagnéticas en la atmósfera es la velocidad de la luz en el medio, $v = 299708km/s$. Dado que las señales AIS se emiten a una frecuencia de $f = 162MHz$, la longitud de onda de esta señal, aplicando la expresión anterior, es de $\lambda = 1{,}85m$. La longitud de una antena lambda cuartos se calcula como $l = \frac{\lambda}{4}$. Por lo que la longitud óptima para recibir las señales AIS es de $l = 0{,}46m$.

Para la estación AIS portátil se ha escogido una de menor tamaño, aunque se pierda ganancia, dado que el receptor IQ permite sintonizar la frecuencia de los mensajes AIS. Finalmente, la antena elegida es de la marca Lacucino, mide 14 cm, tiene una impedancia de 75 ohmios y viene incorporada con un cable de antena de 2 metros con conector SMA macho.

Para la elección de la antena, aparte de conocer la onda que se desea recibir, es necesario tener en cuenta una serie de características de la misma, entre ellas:

- Ancho de banda: El es intervalo de frecuencias sobre el que una antena puede trabajar. Esto básicamente depende de la longitud del dipolo en este tipo de antenas.
- Ancho de haz: Ángulo formado por dos ejes que unen el diagrama de radiación con los puntos donde la ganancia ha caído 3dB respecto al punto de máxima radiación.
- Ganancia: Es la relación que existe entre la densidad de potencia que radia en una dirección, a una distancia y la densidad de potencia que radiaría con la misma potencia, a la misma distancia, una antena isotrópica.
- Directividad: Capacidad de recibir una señal en una determinada dirección del espacio.
- Impedancia: Es la relación entre la tensión y la corriente en los terminales de entrada.
- Relación delante-atrás: Relación entre la ganancia de la dirección de máxima radiación y cualquier otra dirección.

3.1.2. Receptor IQ

Se ha escogido como receptor IQ un dispositivo RTL-SDR, que aunque tiene solo funciones de recepción, cumple con las características necesarias para recibir los mensajes AIS. Estos dispositivos son *dongles* USB de bajo coste y albergan en su interior el hardware necesario para sintonizar una frecuencia, demodularla y muestrearla para posteriormente enviar la información de la señal a la unidad de control. Originalmente estos dispositivos se utilizaban como receptores de televi-

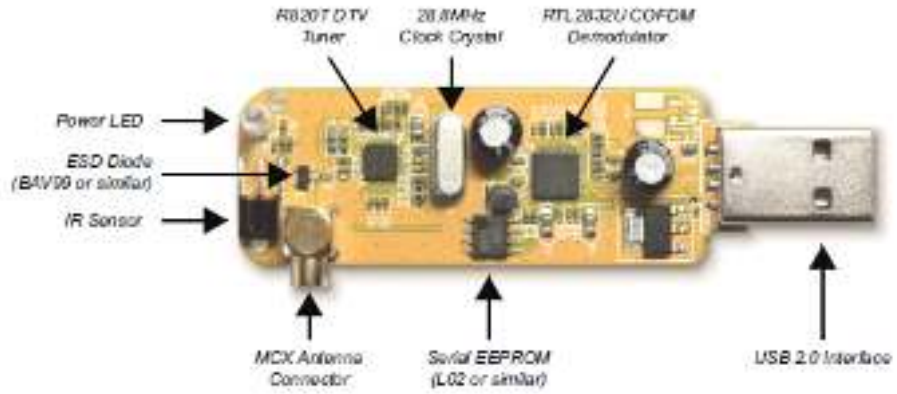


Figura 3.2: Elementos de un dispositivo RTL-SDR.

sión DVB-T (Digital Video Broadcast-Terrestrial) pero gracias a su gran rango de recepción, entre 25 MHz y 1.75 GHz, también podemos utilizarlo para demodular otras señales dentro del espectro de radiofrecuencia.

Como se puede ver en la Figura 3.2 las partes más relevantes del dispositivo RTL-SDR son sus dos microchips, uno de sintonización y otro de demodulación. En este proyecto, solamente se explicarán los dos microchips que alberga el RTL-SDR escogido, el Rafael Micro R820T (sintonizador) y el Realtek RTL2832U (demodulador).

En esta combinación [7], el sintonizador convierte una banda de las señales de radiofrecuencia a frecuencias intermedias (FI) con un valor de frecuencia de 3,57 MHz. La señal resultante se envía a el chip demodulador, que sintoniza la frecuencia central de la FI y la convierte a banda base. Después, el RTL2832U muestrea mediante su ADC a una velocidad de 28.8 MHz y ejecuta una demodulación en cuadratura para producir las muestras IQ, las cuales van a ser pasadas por un filtro de decimación para reducir la tasa de muestreo a 2,8 MHz. Finalmente, las muestras IQ se envían a la unidad de control mediante su interfaz USB.

R820T

El microchip R820T fabricado por la marca Rafael Micro, como se ha mencionado anteriormente, tiene la función de sintonizar la frecuencia de radio deseada y preparar la misma para su posterior demodulación, esto es, pasándola a frecuencias intermedias.

Como se puede ver en la Figura 3.4, una vez dentro del chip, la señal recibida de radiofrecuencia es pasada por un amplificador de bajo ruido (LNA). Este se encar-

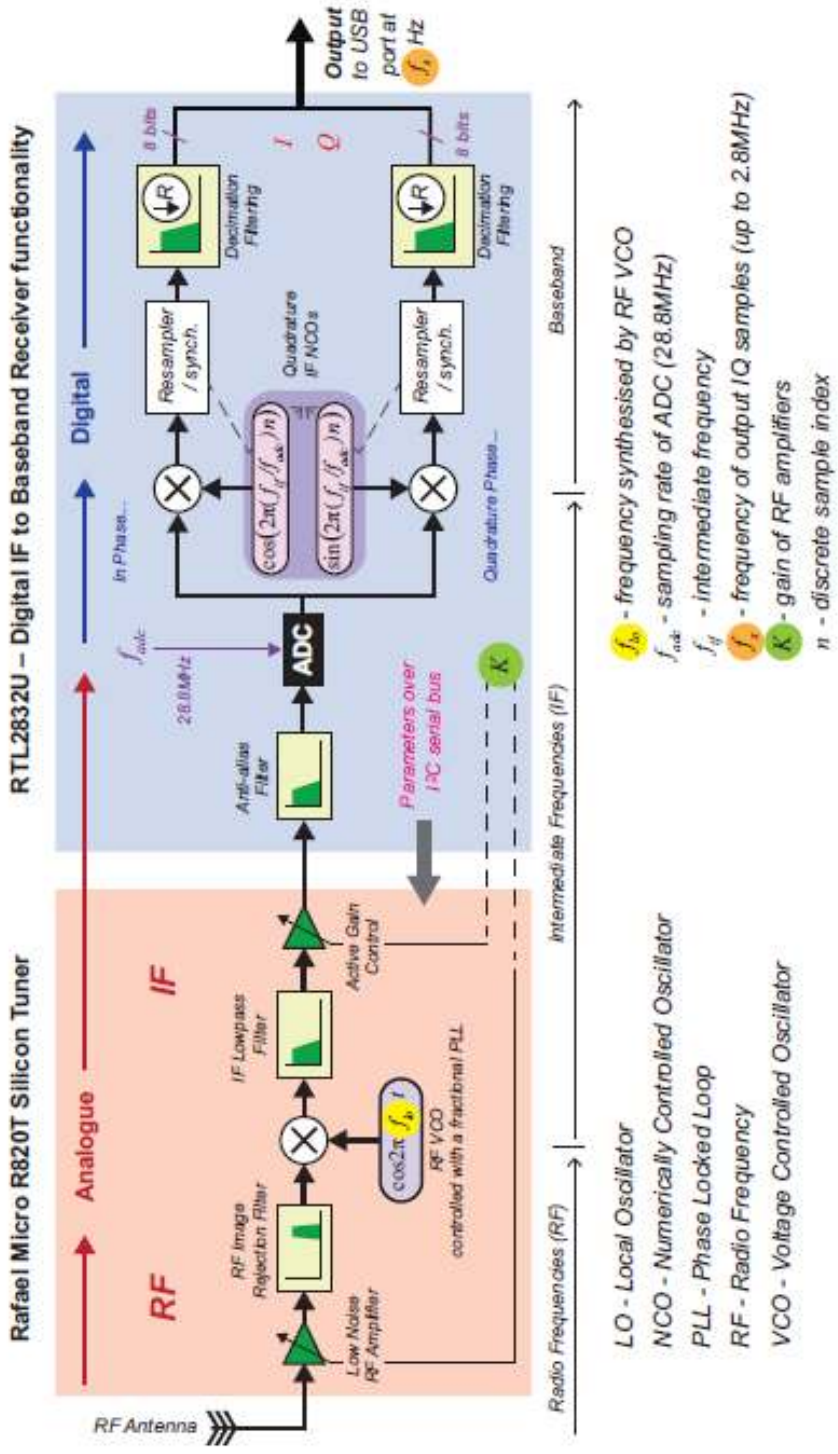


Figura 3.3: Diagrama de bloques de la combinación R820T y RTL2832U.

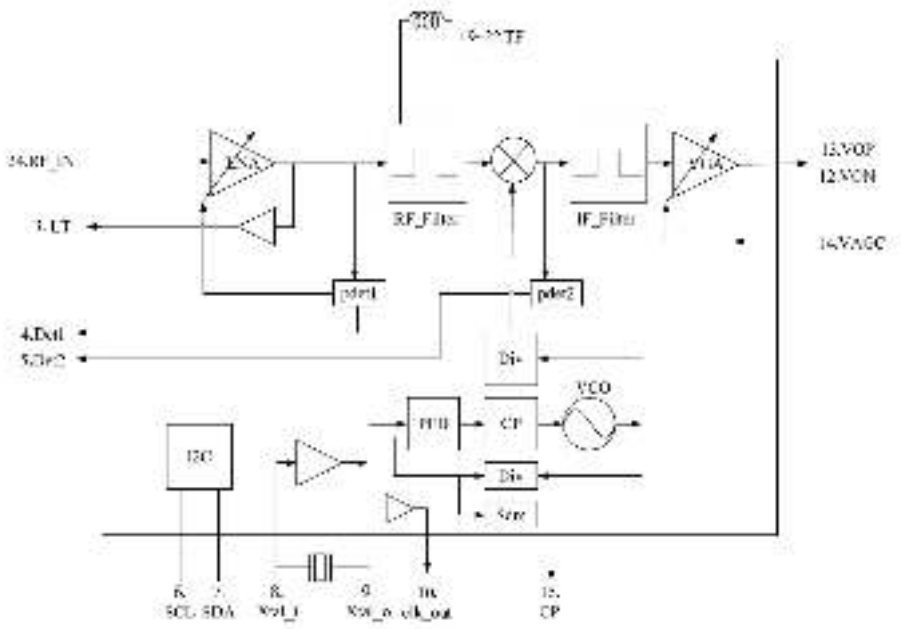


Figura 3.4: Diagrama de bloques del chip R820T.

ga de amplificar la señal proveniente de la antena, que previsiblemente será débil, y de reducir al mínimo el ruido adicional que el amplificador pueda añadir. Posteriormente, la señal es pasada por un filtro de radiofrecuencia, donde a través de software, se le dictará al chip, cuál será la frecuencia de corte, para así sintonizar y adquirir la señal buscada. Una vez se obtiene la señal, ahora amplificada y sintonizada, se traslada a frecuencias intermedias. Esto se consigue gracias a un receptor superheterodino, donde entran la señal de radiofrecuencia y otra originada en un oscilador local. Mediante procesos multiplicativos de combinación y utilizando las dos señales, el receptor superheterodino o mixer, permite generar una única señal a su salida, ahora en frecuencias intermedias, exactamente a 3,57 MHz.

RTL2832U

El microchip RTL2832U está fabricado por la marca Realtek y su principal función es la de muestrear y demodular la señal que ha sido anteriormente enviada por el chip sintonizador. Acepta una entrada a frecuencias intermedias, y posteriormente muestrea la señal con su ADC interno. El flujo de datos muestreados se procesa mediante demodulación OFDM (Multiplexación por división de frecuencias ortogonales). Esta, es una técnica de transmisión que consiste en la multiplexación de varias portadoras con distintas frecuencias, cada una de ellas con

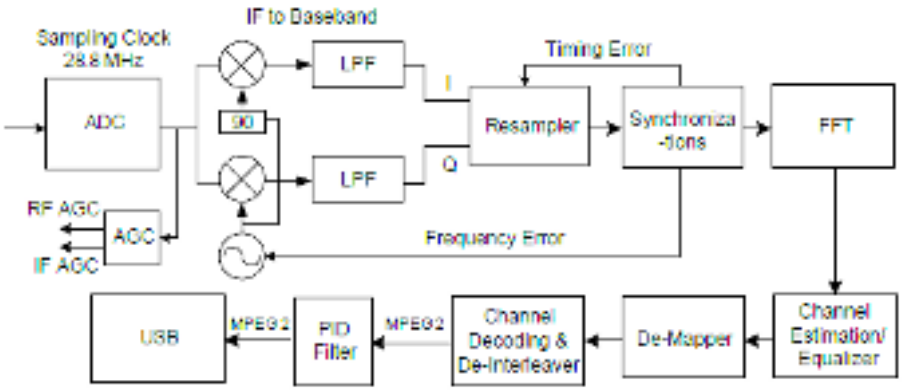


Figura 3.5: Diagrama de bloques del chip RTL2832U.

su flujo de información, y que posteriormente son moduladas en QAM o PSK. Después de la decodificación mediante un FEC (Decodificador Viterbi y Reed-Solomon) incorporado en el chip, la interfaz USB genera una serie de paquetes de datos que son enviados a la unidad de control.

Como se puede apreciar en la Figura 3.5, el primer bloque que se encuentra la señal proveniente del chip sintonizador, es el conversor analógico-digital, que utilizando un reloj de muestreo generado por el PLL (Lazo de seguimiento de fase) del chip, muestrea la señal a 28,8 MHz. A partir de este punto, la señal se encuentra en banda base.

Después del muestreo, la señal pasa por una demodulación en cuadratura, obteniendo así las componentes IQ, que a su vez son pasadas por un filtro paso bajo. Luego, la señal entra al circuito remuestreador, donde pasa de una frecuencia de muestreo fija, a una frecuencia de muestreo de multiplexación por división de frecuencia ortogonal de acuerdo con el ancho de banda de la señal. Posteriormente, a la señal se le aplica la transformada rápida de Fourier, donde pasa del dominio del tiempo al de la frecuencia. Después de la etapa de ecualización, donde se compensa la degradación que pueda tener la señal, el chip utiliza un desintercalador para reordenar el bit de datos de decisión en la secuencia correcta, un decodificador de corrección de errores (FEC) para corregir los bits de error en la secuencia y un último decodificador (llamado *Descrambler*) para recuperar la salida y formar una secuencia de flujo de transporte estándar. Por último, la señal pasa por un filtro PID y se procesa para poder ser enviada mediante la interfaz USB.

Figura 3.6: Sensor BME–280 antes de soldar sus conectores.

3.2. Otros sensores

3.2.1. Sensor ambiental

Para aprovechar al máximo la unidad de control que se va a utilizar (y que se desarrollará más adelante), se implementará un sensor ambiental para poder medir temperatura, humedad y presión atmosférica. Estos datos pueden ser de interés en relación con el estudio del comportamiento de los barcos que se quieren estudiar.

El sensor que se va a emplear es el BME–280, de la marca Bosch, debido a las características que lo rodean. Su tamaño y peso reducidos son carácterísticas interesantes para un dispositivo portátil. También sus buenas reseñas, su alta disponibilidad, su bajo precio y la gran cantidad de material didáctico que se puede encontrar en internet. Estas son algunas de las ventajas que nos presenta este sensor y por las cuales se ha decidido utilizar.

El BME–280 se muestra en la Figura 3.6, sin contar con la pequeña placa que lo contiene, está alojado en una tapa metálica de solo 2,5 x 2,5 mm2, y con una altura de 0,93 mm. Proporciona interfaces SPI e I2C, y tiene un rango de alimentación de 1,71 V a 3,6 V.

3.2.2. GPS

Debido a que el objetivo de este proyecto es la creación de una estación AIS portátil y que la misma va a poder funcionar de manera offline, es de especial relevancia saber la ubicación de la misma y posteriormente a la recogida de datos,

Figura 3.7: GPS VK-162 G-Mouse.

tener un registro, para su estudio, de dónde ha estado localizada geográficamente. Por esto, se ha utilizado un sensor GPS en la estación. Se ha escogido el modelo VK–162 G–Mouse de la marca DIYmalls que se muestra en la Figura 3.7. Algunas de las características más interesantes que brinda este dispositivo son su facilidad de conexión con una interfaz USB, su alta fiabilidad, su tamaño reducido y su rapidez a la hora de recibir señales de los satélites.

3.3. Unidad de control

La estación AIS debe tener una unidad de control con la cual interactuar, enviando así, órdenes a los periféricos como puede ser el RTL-SDR (órdenes de sintonización y demodulación), el GPS o el sensor ambiental. Debe tener capacidad de almacenamiento para la recogida de datos y un tamaño, peso, y consumo reducidos debido a que esta estación será portátil. Esta unidad de control, como se ha mencionado anteriormente, tendrá una serie de periféricos, por lo que es necesario que haya una compatibilidad con ellos y también debe ser capaz de procesar todos los datos que estos recogen. Por estos motivos, la unidad de control que se ha seleccionado para la estación AIS portátil es una Raspberry Pi modelo B+. En la Figura 3.8 se muestra la vista superior de la placa utilizada.

Esta es una placa SBC (Single Board Computer) que nos brinda unas características idóneas para el terminal, dada su alta versatilidad, accesibilidad y fácil manejo. Al ser un ordenador de placa única, sus dimensiones son muy pequeñas, por lo que es fácilmente portable. Tiene un bajo consumo de electricidad, aspecto

Figura 3.8: Raspberry Pi modelo B+.

importante dado que esta estación es portátil y necesita baterías externas. Son de fácil acceso ya que tienen un costo reducido y están bastante extendidas. Por defecto, su sistema operativo es Raspbian, una versión adaptada de Debian, basado en Linux, por lo que tendrá fácil acceso a todos los repositorios de Debian.

La Raspberry Pi 3 Modelo B+ tiene un procesador ARM Cortex-A53 con una capacidad de procesamiento de 1,2GHz de 64 bits, 1 GB de RAM, 4 puertos USB 2.0 (1 de ellos estará reservado para el RTL-SDR y otro para el GPS) y 1 puerto Ethernet. También y aunque no se utilizarán en este proyecto, tiene un puerto HDMI y uno de audio. Se le ha añadido una tarjeta microSD de 64 GB, espacio más que suficiente para almacenar las librerías necesarias para hacer funcionar el terminal AIS y guardar los registros generados por el mismo. El tamaño de la placa es de 85 x 56 mm y 17 mm de alto. Su alimentación es a 5V y 2,5A, cuestión a tener en cuenta a la hora de dimensionar las baterías.

3.4. Visualización y batería

3.4.1. Visualización

Esta estación ha sido diseñada para, a parte de recoger los datos anteriormente mencionados, que los mismos puedan ser visualizados por el usuario a tiempo

Figura 3.9: Raspberry Pi3 7"Touchscreen Display.

real. Por lo que se incorporará una pantalla táctil de 7 pulgadas para que se pueda visualizar e interactuar con los datos recogidos. Esta interfaz se desarrollará con más detalle en el capítulo 4.

La pantalla escogida es la Raspberry Pi3 7"Touchscreen Display, diseñada y fabricada por la marca Raspberry Pi que se muestra en la Figura 3.9.

3.4.2. Batería

Dado el carácter portátil del prototipo es preciso el uso de una batería que pueda alimentar todo el sistema. Para el dimensionamiento de la misma, se ha tenido como referencia de tiempo de uso, una jornada de trabajo. Es decir, que la estación AIS pueda ser utilizada durante 8 horas sin necesidad de volver a cargarla. Por lo que, teniendo el sistema una batería completamente cargada, se pueda disponer de 8 horas para la recopilación de datos. La batería debe tener una salida de 5 V y como mínimo 2,5 A para poder alimentar todo el resto de la estación, ya que la Raspberry Pi tiene estos parámetros de alimentación.

Como resultado, se ha escogido una *powerbank* de la marca XO modelo XO-PR144, cuyos principales características son: una capacidad de 20000mAh y una salida de 22,5W (5V - 4,5A), parámetros suficientes como para alimentar el resto del dispositivo durante un periodo de tiempo extendido. Cabe mencionar que esta *powerbank* tiene un pequeño visualizador del porcentaje de carga, como se ve en la Figura 3.10.

Figura 3.10: *Powerbank* XO-PR144.

3.5. Carcasa

Como se puede ver en la Figura 3.11, la envolvente del prototipo de estación AIS portátil tiene un diseño cuadrado de 22 x 22 cm, con unas esquinas achaflanadas reforzadas en el interior con un grosor de 5 cm para aumentar su resistencia ante los posibles impactos, adecuado para poder ser sostenido entre las manos y poder ver los datos que se muestran en la pantalla. Se ha utilizado una impresora 3D para su creación, con plástico PLA. El diseño de la carcasa se ha concebido teniendo en cuenta 4 piezas distintas. La primera es la parte frontal, donde irá el montaje de la estación AIS, las dos piezas siguientes son las tapas de los cables para las antenas de radiofrecuencia y de GPS, y la última parte es la tapa trasera de la estación. En los siguientes apartados se desarrollarán todos los puntos relevantes del diseño de la carcasa.

Como se puede ver en la Figura 3.12, esta pieza es la parte frontal de la carcasa y también incluye las paredes de la misma. Para explicar la funcionalidad de cada parte se ha utilizado un número dentro de un círculo en la figura. En el punto ①, se puede observar como en todas las esquinas achaflanadas hay un bloque de PLA macizo que se utilizará para atornillar la parte trasera de la carcasa y poder tapar la estación AIS. En el punto ② se puede ver un pequeño saliente con un hueco en su interior que será la parte fija de las bisagras para tapar los receptáculos de los cables de las antenas que iran en los puntos ③ y ④. Estos receptáculos tienen una profundidad de 2 cm, donde irán alojados los cables que pueden ser extraídos fácilmente para poder mover las antenas a mejores ubicaciones.

El punto ⑤ es un saliente con un hueco en su interior donde mediante una argolla, irán colgadas las llaves para encender y apagar la estación AIS. El punto ⑥ es

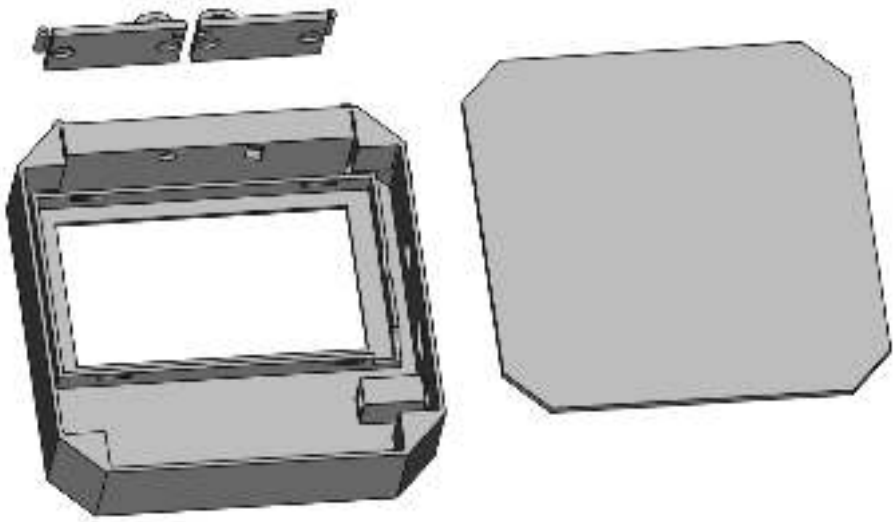


Figura 3.11: Diseño de la carcasa en 3D.

la parte de la carcasa donde reposará la pantalla. Se impedirá el movimiento de la misma en el plano XY mediante un marco, punto ⑩ y en el eje Z gracias a unos huecos dentro del marco, punto ⑨ donde irán unos pasadores de aluminio de extremo a extremo. El montaje se explicará en más detalle en el capítulo 4.

Los puntos ⑦ y ⑧ son huecos que se le han hecho al marco, punto ⑩, teniendo en cuenta el montaje del interruptor y de los puertos USB, como se muestra en la Figura 3.12. El punto ⑪ hace referencia al espacio donde irá alojada la *power-bank* y aprovechando que esta tiene un pequeño visor que indica su porcentaje de carga, se ha diseñado un pequeño túnel, punto ⑫, para poder ver desde el exterior este porcentaje. En el extremo de la pared, esta estructura tiene un cuadrado de metacrilato por el que a su través se pueda observar el porcentaje de carga.

En la Figura 3.13 se puede ver la vista superior de la pieza frontal de la carcasa. En el punto ① se pueden observar las bisagras anteriormente mencionadas, los puntos ② y ⑤ hacen referencia a los huecos hechos en la parte inferior del receptáculo de cables por donde pasarán el conector SMA de la antena de radiofrecuencia, punto ② y el conector USB de la antena GPS, punto ⑤. El punto ③ es un saliente del receptáculo donde irá alojado un imán circular para facilitar el cierre de la tapa, que se posará en un marco, punto ④ de 2 mm de grosor.

En la Figura 3.14 se muestra el lateral de la pieza frontal de la carcasa. En ella, podemos observar el saliente donde se colgarán las llaves para el encendido y apagado de la estación, punto ②. El punto ③ hace referencia al hueco donde

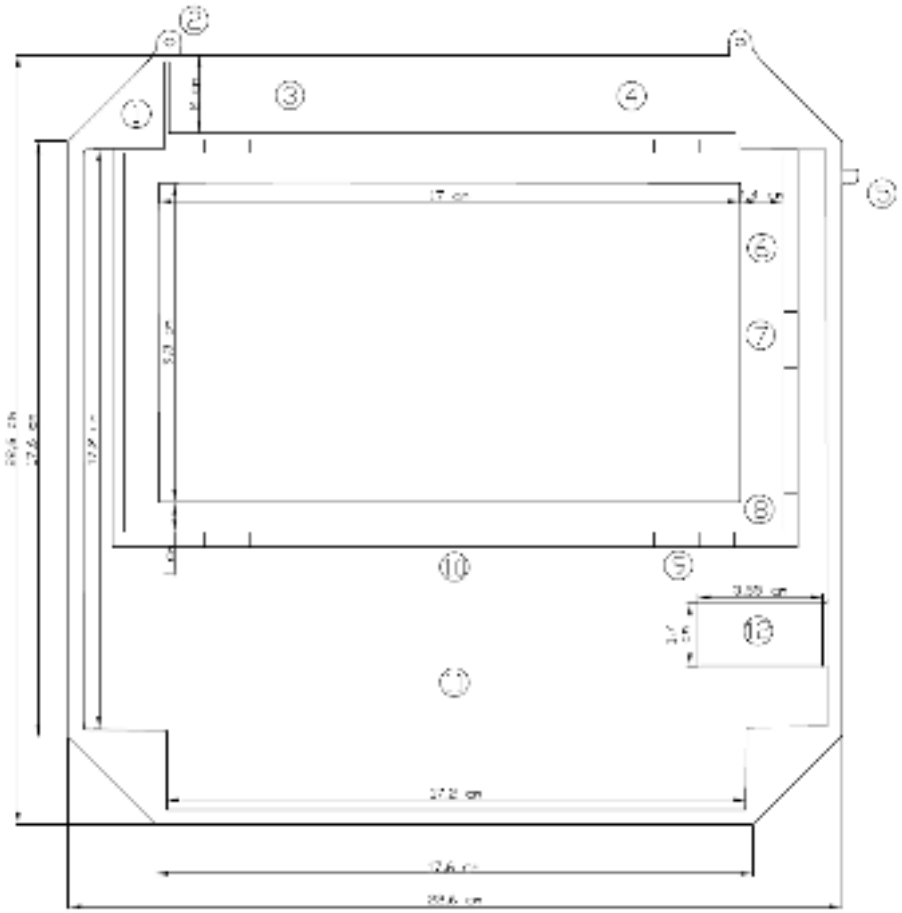


Figura 3.12: Interior de la pieza frontal.

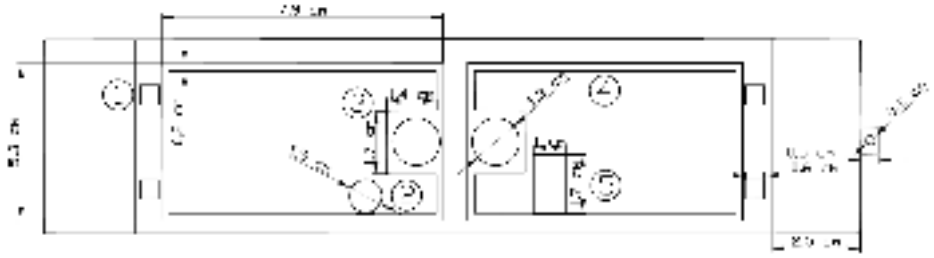


Figura 3.13: Vista superior de la pieza frontal.

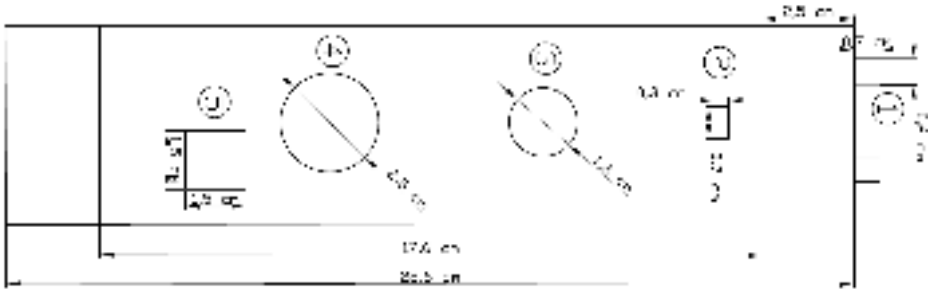


Figura 3.14: Vista lateral de la pieza frontal.

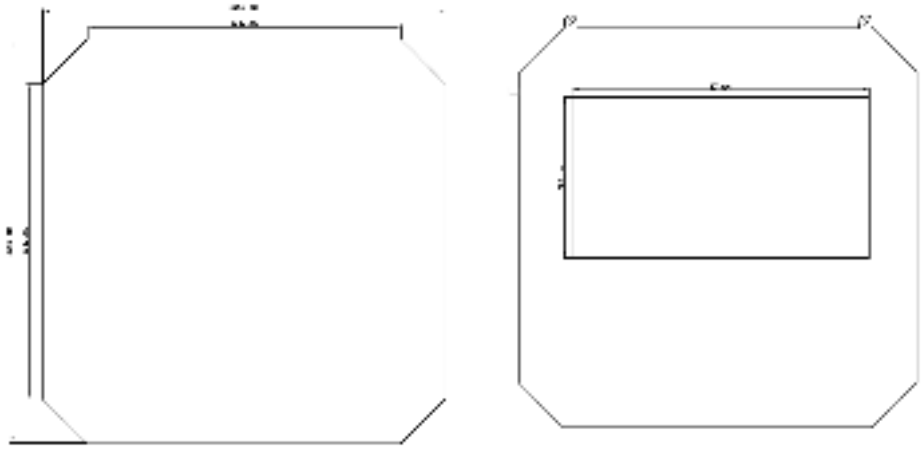


Figura 3.15: Tapa trasera y pieza frontal de la carcasa.

irá alojado el interruptor accionado mediante las llaves. Este interruptor tiene una forma cilíndrica y se sujeta a la pared mediante una tuerca, de ahí el hueco circular hecho a la pared de la carcasa. El punto ④ es un hueco donde irán los puertos USB de la estación, uno de ellos se utilizará para la carga de la batería y otro para poder interactuar con la Raspberry Pi. Finalmente, el hueco de forma cuadrada, punto ⑤ es por medio del cual se hará visible el porcentaje de carga de la batería.

En la Figura 3.15, se puede ver la tapa trasera y pieza frontal de la estación AIS, que como se ha mencionado con anterioridad, son dos piezas distintas. En la figura podemos ver el hueco donde posteriormente irá alojada la pantalla para la visualización de los datos en la frontal.

En la Figura 3.16 se recogen tres vistas. Se pueden ver todas las partes de las tapas de los receptáculos, que aunque son dos, al ser iguales, con detallar las características de una de ellas será suficiente. En el punto ① se puede observar un hueco circular donde irá el imán para el cierre de la tapa. El punto ② hace

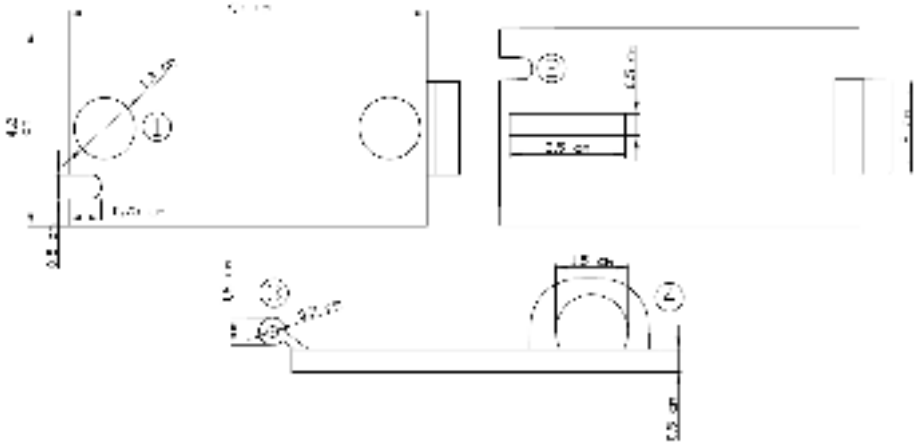


Figura 3.16: Vistas de las tapas.

referencia al hueco de la tapa por el cual irá el cable que saldrá hasta la antena. El punto ③ es la parte de la bisagra que es móvil y el punto ④ hace referencia a la anilla para abrir y cerrar la tapa.

Capítulo 4

Montaje e Implementación

4.1. Montaje

Como se ha mencionado anteriormente, todos los dispositivos que se van a utilizar en la estación AIS portátil, están alojados en la pieza principal de la carcasa. En esta sección se describen las partes del montaje del prototipo final empezando por las antenas, su alojamiento en el exterior en la parte superior de la carcasa, el lateral y la parte trasera. A continuación se explicará el contenido de la pieza principal de la carcasa.

4.1.1. Antenas

Como se puede apreciar en la Figura 4.1 las dos antenas, radiofrecuencia a la derecha y GPS a la izquierda, están ubicadas en la parte superior de la estación AIS. Estas se sitúan sobre las tapas de los receptáculos para los cables de las antenas y van sujetas a las mismas mediante sujeción magnética. Las antenas tienen unos imanes en su base. Por tanto, se ha pegado una pequeña placa de acero dentro de la tapa con un pegamento para plásticos.

4.1.2. Parte superior

Como se puede ver en la Figura 4.2, los puntos ① y ② hacen referencia a los imanes de las tapas, que están sujetos mediante pegamento para plásticos. En el punto ③ se puede observar el pasador utilizado para las bisagras, que en este caso han sido 2 clavos de cabeza plana por bisagra. El punto ④ hace alusión a

Figura 4.1: Imagen de la estación AIS portátil terminada.

Figura 4.2: Vista superior de la estación AIS portátil.

Figura 4.3: Vista lateral de la estación AIS portátil.

los huecos dejados en el interior de los receptáculos por los que se introducen los conectores para su conexión en el interior de la estación, USB para el caso de la antena GPS y SMA para la antena de radiofrecuencia.

4.1.3. Parte lateral

Desde la vista lateral que se puede ver en la Figura 4.3 de la estación AIS, se pueden observar: el colgador para las llaves de arranque en la parte superior, el interruptor accionado mediante estas y los dos puertos USB, el de la izquierda sirve para acceder a la Raspberry Pi y el de la derecha para cargar la batería. En la parte inferior de la Figura se puede ver el porcentaje de carga de la batería.

4.1.4. Parte Trasera

Desde la vista trasera de la estación que se muestra en la Figura 4.4, se ve claramente la forma de cierre de la tapa, haciendo uso de arandelas y tuercas ciegas. Se

Figura 4.4: Vista trasera de la estación AIS portátil.

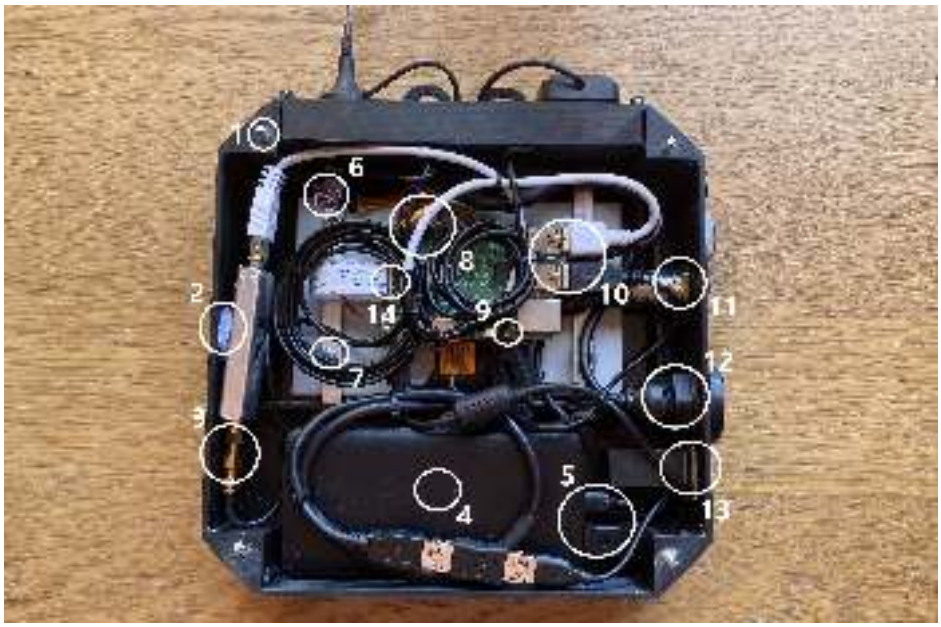


Figura 4.5: Interior de la estación AIS portátil.

han incorporado unas varillas roscadas en el interior de los refuerzos triangulares, atravesando la parte trasera como se explica mejor en la subsección 4.1.5.

4.1.5. Interior de la carcasa

Como se puede ver en la Figura 4.5, esta es la vista trasera de la estación, ahora sin la tapa. Cabe mencionar que todos los cables, menos los de las antenas se han cortado y soldado a medida, debido al escaso espacio de la carcasa.

En el punto ① se pueden ver las varillas roscadas de métrica 4. Para su montaje se han perforado las esquinas macizas de plástico PLA y se ha introducido resina en su interior. Posteriormente se ha insertado las varillas en los huecos.

El punto ② muestra la fijación del dispositivo RTL-SDR, que se ha conseguido mediante velcro y en el punto ③ se puede ver la conexión entre el receptor IQ y la antena de radiofrecuencia, para la cual, ha hecho falta un adaptador SMA macho-macho.

En los puntos ④ y ⑤ se ve el montaje de la *powerbank*, el primero hace alusión a su fijación, que se ha conseguido mediante velcro, y el segundo a la conexión con el resto de la estación, siendo el conector USB el que alimenta la Raspberry Pi y el conector USB tipo C la alimentación de la batería.

En el punto ⑥ se muestra el sensor BME-280, para el cual, se tuvo que soldar con estaño los 4 pines de conexión. Estos se conectan a la Raspberry Pi de la siguiente manera: El pin VIN del sensor (alimentación) va conectado a el pin 1 de la Raspberry Pi, siendo este el de 3,3 V. El pin GND del sensor (tierra), va conectado a cualquier pin GND de la Raspberry, en este caso, se ha escogido el pin 20. Los pines SDA y SCL (datos), van conectados a los pines 3 y 5 de la Raspberry Pi, siendo estos los pines con el protocolo I2C, comúnmente utilizados para leer datos de sensores.

El punto número ⑦ sirve para la fijación de la pantalla, específicamente a la sujeción que impide el movimiento en el eje Z. Esto se ha conseguido mediante unas tiras de aluminio perforadas, que insertadas en los huecos del marco de la carcasa reservado para impedir el movimiento de la pantalla en el plano XY, restringen el movimiento en el eje Z. Además, se ha aprovechado los huecos para tornillos de métrica 3 que trae la pantalla, consiguiendo así una mejor sujeción.

En los puntos ⑧ , ⑨ y ⑭ , se detallan la interconexión y la fijación entre la pantalla y la Raspberry Pi. En el punto ⑧ podemos ver la conexión de pines, que se disponen de la siguiente manera: El pin de 5V de la pantalla (alimentación), va conectado al pin 4 de la Raspberry, obteniendo así los 5V necesarios. El pin GND (tierra), como en el caso del sensor BME-280, se puede conectar a cualquier pin de la Raspberry reservado para este cometido, sin embargo se ha utilizado el pin 14. El punto número ⑭ , hace alusión al conector DSA, que se encarga de la transferencia de datos entre la Raspberry y la pantalla, y el número ⑨ a la fijación. Esta se consigue gracias a unos tornillos de métrica 2,5 que van atornillados entre la Raspberry y unos cilindros roscados que salen desde la placa de la pantalla, donde se apoya la Raspberry.

En el punto ⑩ se detallan las conexiones USB de la Raspberry Pi. Un conector está reservado para el RTL-SDR, otro para la antena GPS, y por último, otro para poder acceder a la Raspberry desde el exterior.

El ⑪ indica al interruptor de encendido y apagado de la estación AIS. Como se ha mencionado anteriormente, se acciona con una llave, y se fija a la pared de la carcasa mediante una tuerca. Este interruptor está en serie con el cable que sale desde la batería con el conector USB y termina alimentando la Raspberry Pi con el conector microUSB. Para mecanizarlo, solo ha hecho falta el cable rojo del interior del cable USB, ya que este porta los 5 V.

El punto número ⑫ hace referencia al montaje empotrado de los puertos USB. Estos van sujetos a la pared de la carcasa mediante una tuerca y como se ha mencionado, están reservados para la alimentación de la batería y para el acceso a la estación AIS desde el exterior.

Finalmente, el punto número (13) hace alusión a la pequeña placa de metacrilato que va ubicada al final del túnel de visualización de porcentaje de carga de la batería. Al igual que con los imanes de las tapas, se ha utilizado pegamento para plásticos para su sujeción.

4.2. Implementación de software

En esta sección se detallará como se ha obtenido los datos, su registro y su visualización. Se empezará detallando los programas que se han utilizado (RTL-AIS, GNUAIS, GPSD, etc.), posteriormente, los códigos para su implementación (gnuais.py, temperatura.py, gps.py) y finalmente, el *script* que creará la página donde se podrán ver los resultados (index.php).

4.2.1. Implementación del Receptor IQ

Para poder utilizar el dispositivo RTL-SDR, es necesario instalar una serie de paquetes que se podrán encontrar en los repositorios de Linux. Por lo que se instalaron las herramientas necesarias para gestionar y compilar el código. Para ello en rasbian, debian y ubuntu se utilizan los siguientes comandos:

- *sudo apt-get install git* # Para control de versiones
- *sudo apt-get install cmake* # Para compilación automática
- *sudo apt-get install build-essential* # Compiladores

Ya que el dispositivo RTL-SDR es un dongle USB, debemos instalar la librería libusb-1.0-0-dev, que proporcionará acceso genérico a dispositivos USB. Esto se consigue mediante el comando: *sudo apt-get install libusb-1.0-0-dev*.

Para poder interactuar con los microchips RTL2832U y R820T del RTL-SDR, es necesario instalar los controladores adecuados. Estos, convierten al dongle DVB-T en un receptor SDR, ya que el chip permite transferir muestras I/Q. Para ello, se descargan estos drivers mediante el comando: *git clone git://git.osmocom.org/rtl-sdr.git*. Posteriormente, se compilarán e instalarán. Para finalizar, debemos incluir en la "*Blacklist*" de la Raspberry Pi, el controlador predeterminado que se carga automáticamente para usar el *dongle*, ya que lo utiliza como un dispositivo de TV y no para propósitos SDR, chocando así con los nuevos controladores Osmocom.

4.2.2. Recepción de mensajes AIS

Como se ha mencionado en capítulos anteriores, para recibir mensajes AIS es necesario sintonizar, demodular y procesar las señales AIS que los barcos emiten. Por lo que debemos tener un software capaz de interactuar con el dispositivo RTL-SDR, y enviarle así, las órdenes adecuadas para, como resultado, obtener el mensaje AIS entendible para el usuario de la estación. Esto se conseguirá mediante los programas "rtl-ais" y "gnuais", que se detallarán en las siguientes secciones.

RTL-AIS

Este software es capaz de sintonizar y decodificar los datos AIS recibidos por el RTL-SDR y generar las sentencias NMEA explicadas en el capítulo 2 de Fundamentos. Se puede tener acceso al mismo utilizando el comando *sudo git clone https://github.com/dgiardini/rtl-ais* para posteriormente compilarlo.

Para su ejecución, se utilizará el comando,

`"./rtl_ais -n -g 60 -p -3 -A /home/jesus/rtl-ais/audioais"`,

donde la opción "-n" hace que se muestren por consola las sentencias NMEA, la opción "-g" modifica la ganancia y la opción "-p" modifica la desviación de frecuencia en partes por millón de la frecuencia de sintonización deseada y la frecuencia sintonizada real debido a la desviación del oscilador de cristal. El valor de estos dos últimos parámetros se ha determinado probando el comando y observando con que valor se reciben más mensajes, debido a que dependen del dispositivo receptor que se utilice.

En este programa, el valor de sintonización está por defecto centrado en los 162 MHz, por lo que no hace falta modificar este parámetro. Por último, la opción "-A" sirve para enviar al fichero indicado el audio recibido. En este proyecto, se ha optado por la creación de un fichero de tubería (*pipe*), en el que a medida que se va escribiendo, las anteriores líneas se borran una vez leídas. Este fichero se crea mediante el comando *mkfifo audioais*, donde audioais es el nombre del *pipe*.

GNUAIS

Una vez se ha obtenido un fichero de audio con las sentencias NMEA codificadas, es necesario un software con el que decodificarlas y crear las sentencias leíbles para el usuario. Gnuais, es una herramienta para demodular y decodificar mensajes AIS utilizando la entrada de línea de la tarjeta de sonido, de ahí, el fichero de audio

escrito por el anterior software. Esta herramienta se puede obtener utilizando el comando *sudo apt-get install gnuais*.

Para el caso de este proyecto, se utilizará ejecutando el comando

"`gnuais -c gnuais.conf -l /home/jesus/rtl-ais/audioais`",

enviando así la orden "-l" de leer en el pipe audioais.

4.2.3. GPSD

Para la recopilación de los datos de la ubicación de la estación AIS portátil, se ha hecho uso del software GPSD. Gpsd es un *daemon* de servicio que monitoriza a uno o más GPS haciendo que todos los datos de ubicación, rumbo, velocidad, etc. estén disponibles en la computadora en la que se utilice. Para descargar este programa en la Raspberry, se ha utilizado el comando: "sudo aptitude install gpsd gpsd-clients python-gps", y el comando utilizado para que los mensajes con la posición GPS se muestren en la consola es "gpspipe -w".

Los mensajes recibidos tienen la siguiente forma:


```
{
"class":"TPV", "device":"/dev/ttyACM0", "mode":3,
"time":"2022-08-15T17:45:44.000Z", "leapseconds":18,
"ept":0.005, "lat":28.454416700, "lon":-16.315155400,
"altHAE":555.8230, "altMSL":520.1260,
"alt":520.1260, "epx":32.959, "epy":32.278, "epv":7.328,
"track":0.0000, "magtrack":355.2123, "magvar":-4.8,"speed":0.043,
"climb":0.000,"eps":13.28,"epc":14.66,"ecefx":5386363.94,
"ecefy":-1576637.16,"ecefz":3021110.35,"ecefvx":10.99,
"ecefvy":-0.80, "ecefvz":6.18,"ecefpAcc":121.73,
"ecefvAcc":13.28, "geoidSep":35.700,"eph":4.140,"sep":58.710
}
```


Donde los únicos datos que se utilizarán en este proyecto serán la latitud, la longitud y tiempo, para poder trazar un recorrido.

4.2.4. Temperatura, humedad y presión

Para interactuar con el sensor BME-280, primeramente hay que activar la interfaz I2C desde la Raspberry y posteriormente se deberán instalar las librerías que per-

mitan el uso del lenguaje Python. Para ello, dentro de una terminal, se ingresará el comando “sudo pip install pimoroni-bme280 smbus”.

Como se ve en el siguiente fragmento de código, se deben importar las librerías necesarias. El módulo “csv” para que los datos puedan ser guardados en una base de datos con ese formato, el módulo “datetime” para conocer el momento exacto en el cual se ha tomado una muestra y los módulos “SMBus” y “BME280” para poder interactuar con el sensor.

```
import csv
import datetime

from smbus import SMBus
from bme280 import BME280
```

En el siguiente código se puede ver como se abre un fichero llamado “temperatura.csv” para añadir datos y se comienza le añaden las cabeceras de los datos.

```
bus = SMBus(1)
bme280 = BME280(i2c_dev=bus)

f = open('/home/jesus/www/html/datos/temperatura.csv', 'a')
writer = csv.writer(f)
header = ['Temperatura', 'Presion', 'Humedad']
writer.writerow(header)
```

Primero se declaran las variables “dc” (para los decimales) y “tiempo” (el instante en el que el programa se ejecuta). Después, se crea un bucle infinito en el cual se comprueba el instante actual. Si ese momento es superior al anterior en 2 segundos, se ordena abrir y leer un fichero. En el caso de que en el mismo se encuentre la cadena “encendido”, se guarda en las respectivas variables los valores retornados a través de las funciones descritas en la librería del sensor BME-280. Por último, se redondean a dos decimales los valores recogidos para luego escribirlos en el fichero csv. Para finalizar, se actualiza el fichero y el tiempo para volver al bucle.

```
dc = 2
tiempo = datetime.datetime.now()
while True:
  if datetime.datetime.now() > tiempo + datetime.timedelta(
   seconds=2):
    ficheroBoton = open('/home/jesus/www/html/py/boton', 'r')
    contenidoBoton = ficheroBoton.read()
    if contenidoBoton == 'Encendido':
      temperatura = bme280.get_temperature()
      presion = bme280.get_pressure()
      humedad = bme280.get_humidity()
```

```
11        datos = [round(temperatura, dc), round(presion, dc), round
    (humedad, dc)]
12        writer.writerow(datos)
13        f.flush()
14     tiempo = datetime.datetime.now()
```

4.2.5. Ficheros de datos

Una vez obtenidos los datos AIS y de GPS, se escribirán en un fichero para posteriormente discriminar los datos que no hacen falta. Esto se ha conseguido utilizando los shell-scripts gps.sh y gnuais.sh, plasmados respectivamente en los siguientes códigos.

```
1 for f in *.fastq; do
2   gpspipe -w
3 done > /home/jesus/www/html/datos/gps.temp 2>&1
```

```
1 for f in *.fastq; do
2   gnuais -c gnuais.conf -l /home/jesus/rtl-ais/audioais
3 done > /home/jesus/www/html/datos/gnuais.temp 2>&1
```

Como se puede ver, en los códigos se ordena ejecutar un comando para que lo que devuelvan se almacene en los ficheros gps.temp y gnuais.temp. En este punto, se deben discriminar los datos no necesarios. Esto se consigue ejecutando los programas: gps.py y gnuais.py.

Empezando por el fichero gps.py, como se puede ver en el siguiente fragmento de código, el mismo comienza igual que el fichero temperatura.py, abriendo con orden de agregar el fichero gpsCoord.csv que será el fichero de datos de la posición de la estación AIS. A este, cuando se crea por primera vez el fichero se le añaden las cabeceras de las columnas de la base de datos.

```
1 fout = open('/home/jesus/www/html/datos/gpsCoord.csv', 'a')
2 writer = csv.writer(fout)
3 if fout.tell() == 0:
4   header = ['hora', 'Lat', 'Lon']
5   writer.writerow(header)
6   fout.flush()
```

Primero, se recoge en la variable tiempo el instante en el que comienza a ejecutarse el código, se fijan las coordenadas para que el fichero mapa.py (del que posteriormente se hablará) tenga una ubicación donde centrar, y se utilizan las funciones del mismo. Esto es posible de ejecutar gracias a que anteriormente se ha importado dicho código y sus funciones. Como se puede ver, utiliza un zoom de nivel 13 y se centra en las coordenadas marcadas por el código.

```
tiempo = datetime.datetime.now()
coords = [0, 0]
coordsString = "center: [0,0],\n"
map = Map(center = coords, zoom_start = 13)
map.showMap(coordsString)
numeroLineaAnterior = 0
```

Al igual que con el fichero temperatura.py, las muestras se recogerán cada 2 segundos y se inicializará el bucle una vez que el botón esté en modo encendido, por lo que el bucle inicial funciona de la misma manera. Luego, sobre la variable "ficheroGPSTemp" se abre en modo de lectura el fichero gps.temp y en la variable "lineaPendiente", gracias a la función ".readlines" pasa a haber un listado de las líneas del fichero gps.temp. En cada iteración se comprueba si se han añadido más líneas al gps.temp y si es así pasa a la etapa de filtrado. Posteriormente, viene la elección de la línea, ya que el software GPSD devuelve dos tipos de mensaje, el SKY y el TPV, siendo este último el que tiene las coordenadas. Para finalizar, se han incorporado una serie de discriminadores marcados por las comillas del mensaje para poder guardar el tiempo, la latitud y la longitud.

```
while True:
  if datetime.datetime.now() > tiempo + datetime.timedelta(
   seconds=2):
    ficheroBoton = open('/home/jesus/www/html/py/boton', 'r')
    contenidoBoton = ficheroBoton.read()
    ficheroBoton.close()
    if contenidoBoton == 'Encendido':
      ficheroGPSTemp = open("/home/jesus/www/html/datos/gps.temp
   ", "r")
      lineaPendiente = ficheroGPSTemp.readlines()
      if (numeroLineaAnterior < len(lineaPendiente)):
        line = lineaPendiente[numeroLineaAnterior]
        if line.split('"')[10][1:-1] == "2" or line.split('"')
   [10][1:-1] == "3":
          writer.writerow([
            line.split('"')[13],
            line.split('"')[20][1:-1],
            line.split('"')[22][1:-1],
          ])
        numeroLineaAnterior = len(lineaPendiente)
      ficheroGPSTemp.close()
      ficheroGPS.flush()
    tiempo = datetime.datetime.now()
```

Siguiendo con el código gnuais.py, en este, primeramente se importan las librerías necesarias y el código mapa (explicado en el siguiente apartado) para poder utilizar sus funciones.

```
import csv
```

```
import datetime
from mapa import *
import time
```

La siguiente parte del código trata sobre la geolocalización de la estación AIS. Como se puede ver, se abre con orden de lectura el fichero que contiene la base de datos de las coordenadas del gps. De ahí se obtienen, con la función ".split", la latitud y la longitud para posteriormente pasarle los datos recogidos a la función "addCircle" del objeto map.

```
def marcadorPersona(map):
  fopen3 = open("/home/jesus/www/html/datos/gpsCoord.csv", "r")
  last_line = fopen3.readlines()[-2]
  fopen3.close()
  lat = last_line.split(',')[1]
  lon = last_line.split(',')[2]
  print(lat, lon)
  map.addCircle(lat, lon)
```

Como en el código gps.py, se abre en modo de agregar las bases de datos donde irá la información enviada por los barcos y se añaden las cabeceras de las columnas si el fichero está vacío. Hay muchos tipos de mensajes AIS, sin embargo, esta estación escribirá en las bases de datos los dos más comunes. El primero y más frecuente es el que tiene en su interior la ubicación, el rumbo, la velocidad, etc. Y el segundo muestra el nombre del barco, su destino, etc. Estos dos mensajes irán en bases de datos separadas.

```
fout = open('/home/jesus/www/html/datos/gnuaisCoord.csv', 'a')
writer = csv.writer(fout)
if fout.tell() == 0:
  header = ['hora', 'mmsi', 'Lat', 'Lon', 'course', 'speed', '
    rateofturn', 'navstat', 'heading']
  writer.writerow(header)
  fout.flush()

fout2 = open('/home/jesus/www/html/datos/gnuaisDest.csv', 'a')
writer2 = csv.writer(fout2)
if fout2.tell() == 0:
  header2 = ['hora', 'mmsi', 'name', 'destination', 'type', '
    length', 'width', 'draught']
  writer2.writerow(header2)
  fout2.flush()
```

En esta parte del código gnuais.py, nos aseguramos de que el botón Encendido/Apagado y del que posteriormente se hablará, empiece siempre apagado, ya que el usuario se puede haber olvidado de apagarlo en el último uso de la estación AIS.

```
fout4 = open('/home/jesus/www/html/py/boton', 'w')
```

```
writer4 = csv.writer(fout4)
header4 = ['Apagado']
writer4.writerow(header4)
fout4.flush()
```

En esta parte, el código recoge las coordenadas de la base de datos creada por el gps. Esto lo hace almacenando en una variable la última línea de la base de datos. A continuación, se declara el objeto map de la clase Map inicializándolo con las variables requeridas, en esta ocasión las coordenadas almacenadas en una lista y el número del zoom. Posteriormente se llama a la función showMap del objeto map pasándole las coordenadas almacenadas en una cadena.

```
fopen3 = open("/home/jesus/www/html/datos/gpsCoord.csv", "r")
line = fopen3.readlines()[-1]
fopen3.close()
lat = line.split(',')[1]
lon = line.split(',')[2]
tiempo = datetime.datetime.now()
coords = [lat, lon]
coordsString = "center: [" + lat + "," + lon[:-1]+ "],\n"
map = Map(center = coords, zoom_start = 13)
map.showMap(coordsString)
```

Como se puede ver en la siguiente parte del código, este empieza el bucle While de la misma manera que en el código gps.py. Comprobando el estado del botón de encendido y apagado, y verificando si se han escrito nuevas líneas en el fichero gnuais.temp. Una diferencia a remarcar es la comprobación de la línea dónde se escribe la información de los barcos, ya que en este caso, el software gnuais la escribe empezando por “ch”.

```
numerolineas = 0
while True:
  try:
    if datetime.datetime.now() > tiempo + datetime.timedelta(
    seconds=2):
      ficheroBoton = open('/home/jesus/www/html/py/boton', 'r')
      contenidoBoton = ficheroBoton.read()
      ficheroBoton.close()
      if contenidoBoton == 'Encendido':
        marcadorPersona(map)
        map.showMap(coordsString)
        fopen = open("/home/jesus/www/html/datos/gnuais.temp", "
    r")
        lines = fopen.readlines()
        if (numerolineas < len(lines)):
          line = lines[numerolineas]
          if line.startswith("ch"):
            print(line)
```

```
17              hora = str(datetime.datetime.now())
```

Como se puede observar, esta es la parte del código donde se recogen los datos de las líneas escritas en el gnuais.temp. Una vez, que entra a la línea que empieza por "ch", el código comprueba que estén todos los campos de información a rellenar, y a continuación contando por espacios y por comillas, recoge la información escrita para guardarla en las bases de datos gnuaisCoord.csv y gnuaisDest.csv. Luego, y con la función addMarker del objeto map, se le envía el mmsi, la latitud y la longitud de los barcos recibidos para que este añada un marcador con la posición del mismo.

```
1  if line.split(" ")[6] == 'name' and line.split('"')[2] == '
       destination ' and line.split('"')[4].split(" ")[1] == 'type'
       and line.split('"')[4].split(" ")[3] == 'length' and line.
       split('"')[4].split(" ")[5] == 'width' and line.split('"')
       [4].split(" ")[7] == 'draught':
2                 print("entra")
3                 writer2.writerow([
4                   hora.split(" ")[1],
5                   line.split(" ")[5][:-1],
6                   line.split('"')[1],
7                   line.split('"')[3],
8                   line.split('"')[4].split(" ")[2],
9                   line.split('"')[4].split(" ")[4],
10                  line.split('"')[4].split(" ")[6],
11                  line.split('"')[4].split(" ")[8]
12                  ])
13              if line.split(" ")[6] == 'lat'and line.split(" ")[8]
       == 'lon'and line.split(" ")[10] == 'course'and line.split("
       ")[12] == 'speed'and line.split(" ")[14] == 'rateofturn'and
       line.split(" ")[16] == 'navstat'and line.split(" ")[18] == '
       heading':
14                print("entra")
15                writer.writerow([
16                  hora.split(" ")[1],
17                  line.split(" ")[5][:-1],
18                  line.split(" ")[7],
19                  line.split(" ")[9],
20                  line.split(" ")[11],
21                  line.split(" ")[13],
22                  line.split(" ")[15],
23                  line.split(" ")[17],
24                  line.split(" ")[19]
25                  ])
26                map.addMarker(line.split(" ")[5][:-1], line.split(
       " ")[7], line.split(" ")[9])
27                map.showMap(coordsString)
28            numerolineas = len(lines)
```

```
29          fopen.close()
30          fout.flush()
31          fout2.flush()
32        tiempo = datetime.datetime.now()
```

En esta última parte del código, se ha implementado un control de errores, para que cuando se reciba un mensaje no esperado, se registre la hora del mismo y así poder verlo en el fichero gnuais.temp.

```
1 except:
2     fopen = open("/home/jesus/www/html/py/logs", "a")
3     writer = csv.writer(fopen)
4     header = [str(datetime.datetime.now()), 'Error']
5     writer.writerow(header)
6     fopen.flush()
```

4.2.6. Autostart

La estación AIS deberá funcionar con la menor interacción posible con el usuario, por ello, los comandos anteriormente mencionados deberán ejecutarse automáticamente solo con encender la estación. Una vez se pulse el botón ON, y la palabra "Encendido" esté escrita en el fichero "boton", los pogramas gnuais.py, gps.py y temperatura.py rellenarán los registros.

Para conseguir esto, se han creado una serie de ficheros con la extensión ".desktop" con los cuales es posible automatizar la ejecución de estos comandos. Entrando en el directorio "/etc/xdg/autostart" que permite esta función, se han creado estos ficheros ".desktop", y a modo de ejemplo, en el siguiente fragmento se puede ver el código rtl-ais.desktop.

```
1 [Desktop Entry]
2 Name=rtl-ais
3 Exec=/home/jesus/rtl-ais/rtl_ais -n -g 60 -p -3 -A /home/jesus/
      rtl-ais/audioais
```

Este mismo procedimiento se ha seguido para los programas, gnuais.sh, gps.sh, temperatura.py, gnuais.py y gps.py.

4.2.7. Mapa

Como se ha mencionado en el capítulo 3 de Diseño, la estación AIS tendrá una pantalla de visualización en la que aparecerán los datos recogidos y también un mapa donde se podrá ver la ubicación de la estación y la de los barcos cuyos

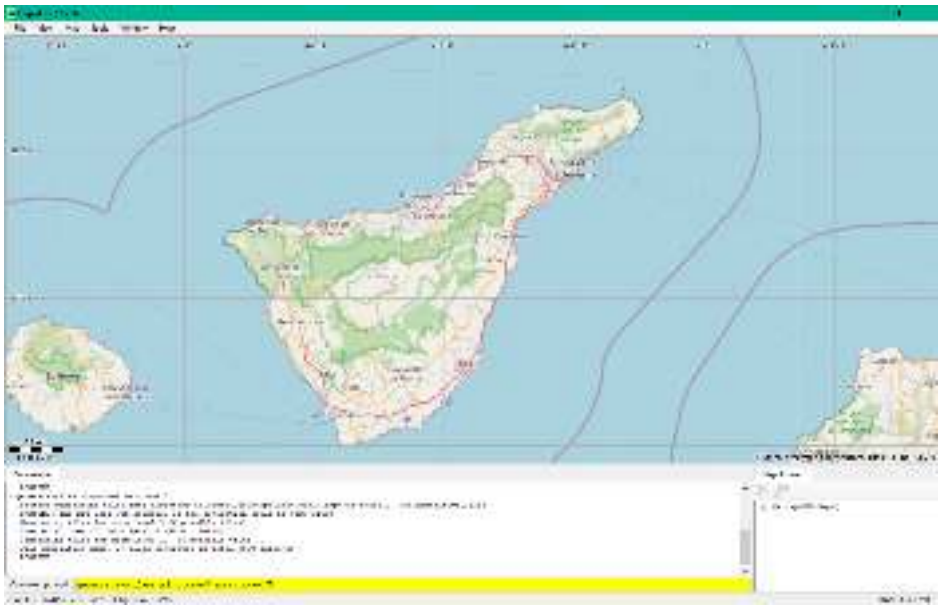


Figura 4.6: Ventana del programa Maperitive.

mensajes AIS sean captados. Por lo que en este apartado se abarcarán los códigos y los pasos a seguir para conseguir que se vea por pantalla dicho mapa.

Este proyecto está diseñado para que funcione de manera completamente autónoma, es decir, sin conexión a internet. Por lo que es preciso descargar el mapa en el cual se ubicarán la posición de otros barcos y la de la estación AIS. Esto se ha conseguido mediante un programa llamado Maperitive, una aplicación de escritorio gratuita para dibujar mapas basados en datos de OpenStreetMap. Mediante este software se descargarán los *tiles*. Estos, son imágenes en formato png que organizados por niveles de zoom, abarcan todo el mapa en su plenitud, con el nivel de detalle marcado por el nivel de zoom que se requiera. La aplicación se ve en ejecución en la Figura 4.6.

Una vez en la aplicación, debemos centrar la ventana del mapa en el punto en el que se requiere generar los *tiles*. Luego, se debe introducir el comando "generate-tiles minzoom=x maxzoom=y", siendo x el nivel más alejado de zoom e y el nivel de zoom con más detalle. En el caso de este proyecto, se ha escogido un nivel de zoom mínimo de 9 y máximo de 17.

Una vez introducido el comando, se empezarán a generar las imágenes en el directorio de destino de menor zoom a mayor. Este tiempo dependerá del nivel de zoom escogido. Es importante no manipular los nombres que el software le pone a las carpetas y a las imágenes, ya que esta es su manera de organizar por nivel de zoom y por latitud y longitud. Para el caso de que la estación AIS se mueva a una

ubicación distinta a la de los *tiles*, se deberá con anterioridad generar los *tiles* de la nueva zona donde estará la estación AIS.

Para visualizar el mapa, ha hecho falta utilizar la librería Folium. Esta facilita la visualización de datos que han sido manipulados en Python para un mapa interactivo, por ejemplo, añadir marcadores, centrar el mapa, etc. Sin embargo, esta librería por defecto utiliza internet para este cometido. Por lo que se han seguido una serie de pasos detallados a continuación.

Como se puede ver en el siguiente fragmento del código de la librería Folium, esta utiliza direcciones web para poder funcionar.

```
_default_js = [
    ('leaflet',
     'https://cdn.jsdelivr.net/npm/leaflet@1.6.0/dist/leaflet.js'),
    ('jquery',
     'https://code.jquery.com/jquery-1.12.4.min.js'),
    ('bootstrap',
     'https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js'),
    ('awesome_markers',
     'https://cdnjs.cloudflare.com/ajax/libs/Leaflet.awesome-markers/2.0.2/leaflet.awesome-markers.js'),  # noqa
    ]

_default_css = [
    ('leaflet_css',
     'https://cdn.jsdelivr.net/npm/leaflet@1.6.0/dist/leaflet.css'),
    ('bootstrap_css',
     'https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css'),
    ('bootstrap_theme_css',
     'https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-theme.min.css'),  # noqa
    ('awesome_markers_font_css',
     'https://maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-awesome.min.css'),  # noqa
    ('awesome_markers_css',
     'https://cdnjs.cloudflare.com/ajax/libs/Leaflet.awesome-markers/2.0.2/leaflet.awesome-markers.css'),  # noqa
    ('awesome_rotate_css',
     'https://cdn.jsdelivr.net/gh/python-visualization/folium/folium/templates/leaflet.awesome.rotate.min.css'),  # noqa
    ]
```

Por lo que se ha manipulado la librería Folium de manera que donde antes habían direcciones web, ahora hay rutas que conducen a los mismos códigos pero

guardados en local, quedando de la siguiente manera:

```
_default_js = [
    ('leaflet',
     '/py/offline/leaflet.js'),
    ('jquery',
     '/py/offline/jquery-1.12.4.min.js'),
    ('bootstrap',
     '/py/offline/bootstrap.min.js'),
    ('awesome_markers',
     '/py/offline/leaflet.awesome-markers.js'),  # noqa
    ]

_default_css = [
    ('leaflet_css',
     '/py/offline/leaflet.css'),
    ('bootstrap_css',
     '/py/offline/bootstrap.min.css'),
    ('bootstrap_theme_css',
     '/py/offline/bootstrap-theme.min.css'),  # noqa
    ('awesome_markers_font_css',
     '/py/offline/font-awesome.min.css'),  # noqa
    ('awesome_markers_css',
     '/py/offline/leaflet.awesome-markers.css'),  # noqa
    ('awesome_rotate_css',
     '/py/offline/leaflet.awesome.rotate.min.css'),  # noqa
    ]
```

Mapa.py

Como se ha mencionado anteriormente, hace falta un programa que reciba las coordenadas de la estación AIS y las de los barcos para poder dibujar encima del mapa estas ubicaciones. Este código es el mapa.py, responsable también y a través de la librería folium, de crear el mapa.html, el cual se plasmará en la pantalla.

Como se puede ver en el siguiente fragmento del código mapa.py, primeramente se crea la clase mapa y luego comienza a definir las funciones. En la función “init”, el código primeramente recibe las coordenadas de centrado y el valor del zoom inicial del código gps.py, y después las recibe del gnuais.py. También se define cual es la ruta a seguir para encontrar los tiles que se requieran y se establece un zoom máximo de nivel 17 y un mínimo de nivel 9 ya que estos niveles de zoom fueron los se introdujeron en el comando de descarga de los tiles y son los que tiene el dispositivo.

```
class Map:
  def __init__(self, center, zoom_start):
```

```
3       self.center = center
4       self.zoom_start = zoom_start
5       self.my_map = folium.Map(location = self.center,
6                                zoom_start = self.zoom_start,
7                                tiles = '/datos/mapa/{z}/{x}/{y}.
    png',
8                                maxZoom = 17,
9                                minZoom = 9)
```

En el siguiente fragmento de código, la función showMap se encarga de actualizar el mapa y escribir en el fichero mapa.html las coordenadas de centrado. Esto lo consigue creando una lista a partir del mismo script y escribiendo en la línea 45 las coordenadas.

```
1  def showMap(self, coordsString):
2      ruta = "/home/jesus/www/html/mapa.html"
3      self.my_map.save(ruta)
4      with open(ruta, 'r+') as fp:
5        lineas1 = fp.readlines()
6        fp.seek(0)
7        fp.truncate()
8        for number, line in enumerate(lineas1):
9            if number in [45]:
10               fp.write(coordsString)
11           if number not in [45]:
12               fp.write(line)
```

En esta parte de la clase, se utiliza la función de la librería Folium addMarker, con la cual se añaden marcadores al mapa. Como se puede ver, se le envían el mmsi de los barcos y su ubicación. A los marcadores se le han añadido una serie de características como son: Si el usuario desliza el dedo por la estación AIS y pasa por encima de un marcador, este indicará el mmsi del barco señalado (tooltip), y si se le pulsa encima, aparecerá una pequeña ventana con el mmsi, la latitud y la longitud (popup).

```
1  def addMarker(self, mmsi, lat, lon):
2      popupstr = '<b>Mmsi:</b> ' + str(mmsi) + '<br>' + '<b>Lat:</
    b> ' + str(lat) + '<br>' + '<b>Lon:</b> ' + str(lon)
3      folium.Marker([lat,lon], popup=popupstr, tooltip= mmsi).
    add_to(self.my_map)
```

La función addCircle se ha utilizado para geolocalizar en el mapa a la estación AIS, representada con un círculo. Por esto, se le envían la latitud y la longitud de la misma, y como en el caso de los marcadores, si se le pulsa encima, aparecerá una ventana con la latitud y la longitud de la estación AIS. También se han tenido que especificar las características de este marcador, como son su radio, la localización y el color.

Figura 4.7: Imagen del marcador utilizado para los barcos.

```
def addCircle(self, lat, lon):
    popupstr = '<b>Lat:</b> ' + str(lat) + '<br>' + '<b>Lon:</b>
    ' + str(lon)
    folium.CircleMarker(
    radius = 3,
    location = [lat,lon],
    popup = popupstr,
    color = "#3186cc",
    fill = True,
    fill_color = "#3186cc",
    ).add_to(self.my_map)
```

Marcador

En este proyecto se ha cambiado el marcador que viene por defecto en la librería Folium. Esto se ha conseguido accediendo a el fichero offline "leaflet.css" y cambiando la ruta a una con la imagen nueva.

```
/* Default icon URLs */
.leaflet-default-icon-path {
  background-image: url(images/marker-icon.png);
  }
```

Siendo la nueva imagen la que aparece en la Figura 4.7:

4.2.8. Visualización del contenido

En este apartado se explicarán los pasos que se han seguido para conseguir visualizar los datos obtenidos y las maneras por las que el usuario de la estación AIS interactúa con la misma. Para ello, se utilizará el archivo index.php como índice y guía de la explicación. Como herramienta a la hora de diseñar la página donde se mostrarán los datos, se ha hecho uso de la herramienta de Google "DevTools" y de Visual Studio Code, que facilitará los trabajos de diseño y los cambios en el mismo ocurrirán instantáneamente.

La página de visualización tendrá 4 ventanas distintas. La parte superior e inferior será la misma para las 4 ventanas, ambas serán explicadas en esta misma sección.

El usuario de la estación AIS podrá ver como primera pantalla, un mapa centrado en su localización actual, y a medida que la estación recibe los mensajes que los barcos emiten, estos serán dibujados en dicho mapa. A la izquierda del mapa, habrá una tabla con el mmsi, la latitud y la longitud de los 10 últimos mensajes recibidos. Si el usuario pulsa el botón para pasar a la siguiente ventana, se encontrará un registro con la temperatura, la presión y la humedad actuales, mostrándose los últimos 10 valores recogidos. En la tercera ventana el usuario verá un registro con los datos del mensaje AIS tipo 1 y se mostrarán solamente los últimos 10 mensajes. En la cuarta y última ventana, el usuario verá un registro de los datos de los últimos 10 mensajes tipo 2.

Servidor

Para visualizar el contenido que se quiere mostrar al usuario, se ha optado por la creación de una página web desde un servidor web local. Estos son aquellos servidores web que residen en una red local al equipo. Se ha elegido esta opción ya que muchas de las APIs modernas del navegador solamente están disponibles si se utiliza un servidor web para mostrar nuestra página. Una API es una interfaz de procesamiento de aplicaciones entre un servidor web y un navegador web.

El servidor web elegido ha sido el Apache2. Actualmente es el servidor más utilizado, está bien documentado y su instalación es bastante sencilla, es suficiente con ejecutar el comando “sudo apt-get install apache2”.

Si en la barra de búsqueda del navegador predeterminado se escribe “localhost” esto nos llevará a la página web de el servidor local instalado. De primeras, aparecerá la página creada por defecto por el servidor web Apache2, donde se recogen las instrucciones a seguir para que la página actual (ubicada en el directorio /var/www/html/ cuyo nombre es index.html) sea reemplazada por la que se quiere mostrar.

Autostart

Para automatizar la visualización de la página, se ha accedido a el directorio /home/pi/.config/lxsession/LXDE-pi/autostart y se ha añadido la siguiente línea.

```
@chromium-browser --incognito --start-fullscreen --start-
    maximized http://localhost/
```

Con esto, una vez que el usuario de la estación AIS encienda la misma, se abrirá esta página. Como se puede ver, se ha escogido el navegador Chromium, instalado

por defecto en la Raspberry Pi, en incógnito para evitar problemas de caché y en pantalla completa.

Índice

El fichero index.php se apoya en dos ficheros, boton.css y style.css. Estos últimos contienen el estilo de la página que se creará y le marcará al fichero index.php como mostrar la información. Por lo que como se ve a continuación, estos dos ficheros tienen que ser referenciados en la cabecera del archivo index.

```
<header>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="style/css/style.css">
  <link rel="stylesheet" href="style/css/boton.css">
</header>
```

Superior

A continuación se empieza a detallar el estilo de la página. Como se puede ver, se cita a una clase llamada “Superior”. Esta, está descrita en el fichero style.css y abarca la parte superior de la página, donde se encuentran dos de los botones de la interfaz. Esta parte de la página será la misma para todas las ventanas, por este motivo se ubica a parte del resto del cuerpo de la interfaz.

```
<body>
  <div class="Superior">
```

Como se puede ver en el siguiente fragmento del código style.css, este recoge las características para el bloque “Superior”, donde se detallan las proporciones del bloque, su color y su posicionamiento en la página.

```
.Superior {
  width: 100%;
  height: 15%;
  background: #aad3df;
  float: right;
}
```

Justo debajo, y siguiendo con el fichero index.php, nos encontramos con las citas a las clases de los botones.

```
<div class="btn-bg Ocean">
      <div class="btn-group" style ="float: right;">
```

La clase llamada btn-group contiene las características de todos los botones como grupo, estableciendo sus márgenes, su disposición en la pantalla, etc. Como se mostrará a continuación, dentro de la clase btn-group, existen dos subgrupos de botones, sin embargo, al estar por debajo de sus respectivos contenedores (como en el caso actual el bloque "Superior"), no hace falta hacer mayor distinción entre ellos, ya que como se ve en el fichero index, al subgrupo de botones ubicados en el contenedor "Superior", lo desplaza independientemente del resto a la derecha.

```
.btn-group {
  margin-top: 7px;
  display: flex;
  flex-wrap: nowrap;
  flex-direction: row;
  align-content: flex-end;
  align-items: stretch;
  justify-content: space-evenly;
}
```

En este punto, empieza la división entre los dos botones de la parte superior. El botón al que se hará referencia a continuación, llamado "ON", será el responsable de activar los programas gnuais.py, gps.py y temperatura.py, por lo que a su vez es responsable de rellenar las bases de datos con la información devuelta por estos códigos.

Este botón se ha diseñado para activar o desactivar los códigos anteriormente mencionados por lo que está enlazado a un código llamado boton.php. Este actualiza el contenido del fichero boton dependiendo del estado del mismo.

```
<?php
  $fichero = '../py/boton';
  $actual = file_get_contents($fichero);
  if ($actual == "Encendido") {
    $actual = "Apagado";
  } else {
    $actual = "Encendido";
  }
  file_put_contents($fichero, $actual);
?>
```

Por lo que si en el fichero boton, está escrita la palabra "Encendido", el resto de códigos (gnuais.py, gps.py y temperatura.py) lo leerán y se activarán. De lo contrario, si en el fichero boton se encuentra la palabra "Apagado", estos permanecerán a la espera de que se pulse el mismo.

Siguendo con el contenido del fichero index, la disposición de este botón en la página se ve descrita en el siguiente fragmento de código. Como se mencionó anteriormente, la clase btn-group de la parte superior tiene dos botones, sin embargo

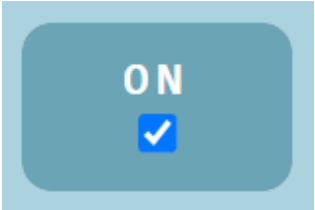


Figura 4.8: Botón ON.

en el fragmento se hace referencia solamente al botón ON, describiendo su porción dentro del subgrupo btn-group, su separación, el radio del borde, su nombre y el tipo de botón que será.

```
<form method="post" action="" style="width: 100px;margin-right:
    20px;">
          <table><th style="border-radius: 10px;">
            <span>ON</span><br/>
            <input type="checkbox" id="AovivoOno" name="
    onoffswitch" class="onoffswitch-checkbox" id="myonoffswitch">
          </th></table>
        </form>
```

Finalmente, el botón ON se muestra en la Figura 4.8:

A continuación, en el fichero index se describe el botón de apagar.

```
<div class="btn Debris">
          <button onclick="location.href='php/shutdown.php'">
    Apagar<span class="one"></span></button>
        </div>
```

Este, es el primer botón incluido en el método “button” y el único dentro de la clase “btn Debris” del fichero boton.css. Este elemento describe las características de todos los botones de la interfaz, incluídos los de la parte inferior de cada página y que se escribirán más adelante, como por ejemplo, sus dimensiones, la manera de actuar con el cursor, el tamaño de la letra, el color, etc.

```
button {
  cursor: pointer;
  overflow: visible;
  outline: none;
  color: #fff;
  position: relative;
  letter-spacing: 0.1em;
  font-weight: 400;
  padding: 10px 20px 10px 20px;
  text-transform: uppercase;
  font-family: Monospace;
```

```
  font-size: 20px;
}
```

La clase btn Debris descrita en el fichero boton.css, engloba un conjunto de etiquetas. La primera de ellas, el button, describe el color de la letra, la posición dentro de la clase btn-group, el radio del marco, etc.

```
body .btn-bg.Ocean .btn-group .Debris button {
  color: #fff;
  overflow: hidden;
  position: relative;
  border-radius: 5px;
  border: 5px solid #fff;
  background: transparent;
  transition: all 2s ease;
  font-family: Monospace;
  font-weight: bold;
}
```

La segunda etiqueta, span, hace referencia a la animación que se encuentra dentro del botón. Se puede modificar dicha animación, el color de la misma, el tiempo de repetición, etc.

```
body .btn-bg.Ocean .btn-group .Debris button span {
  position: absolute;
  content: '';
  top: 1.5em;
  left: 50%;
  width: 20em;
  height: 20em;
  opacity: 0.5;
  background: #fff;
  margin-left: -10em;
  border-radius: 42.5%;
  transform-origin: 50% 50%;
  animation: wave 5s infinite linear;
  transition: all 2s ease, top 1.5s ease;
}
```

La tercera, es la que hace referencia a los cambios que ocurren en el botón cuando se posiciona el cursor encima del mismo. Para el caso de esta interfaz, se ha optado por el cambio de color de las letras y del marco.

```
body .btn-bg.Ocean .btn-group .Debris button:hover {
  color: #50514f;
  border-color: #50514f;
  transition: all 2s ease;
}
```

Figura 4.9: Botón APAGAR.

Figura 4.10: Parte Superior.

Y la cuarta y última, hace referencia a los cambios que ocurren en la animación cuando el cursor se encuentra encima del botón. En este caso también se ha optado por el cambio de color y de tamaño de la misma.

```
body .btn-bg.Ocean .btn-group .Debris button:hover span {
  opacity: 1;
  top: 0.5em;
  background-color: #50514f;
  transition: all 2s ease, top 1.5s ease;
}
```

Volviendo con el fichero index.php, como se puede ver, una vez activado este botón, se da la señal a el código shutdown.php para que se ejecute, apagando así la estación AIS.

```
<style>
  body {background-color: black}
</style>
<?php system('sudo /sbin/shutdown -h now'); ?>
```

Mostrándose el botón de apagar como aparece en la Figura 4.9:

Finalmente, la parte superior de la interfaz, queda como se muestra en la Figura 4.10 :

Mapa

Siguiendo con el fichero index.php, a continuación se detallarán las páginas por las que el usuario podrá moverse para visualizar los distintos tipos de datos. En la ventana llamada “Mapa” se pueden encontrar 1 contendor y dentro de él, 3 divisiones.

```
<div id="Mapa">
```

```
<div class="Contenedor0">
  <div class="Izquierda">
  <div class="Derecha">
  <div class="Inferior">
```

El contenedor principal llamado “Contenedor0” es el que alberga en su interior todos los bloques que se encuentran en la página. Este bloque está detallado en el fichero “style.css”, es el mismo para todas las ventanas y tiene las siguientes características.

```
.Contenedor0 {
  width: 100%;
  height: 100%;
  background: #464646;
}
```

A su vez, este bloque en la página “Mapa” se subdivide en 3 bloques distintos. El primero de ellos es el bloque “Izquierda”.

```
<div class="Izquierda">
  <div style="margin: 10px;">
    <table id="mapaTabla"> </table>
  </div>
</div>
```

Cuyas características como el tamaño y el color vienen recogidas en el fichero “style.css”.

```
.Izquierda {
  width: 30%;
  height: 70%;
  background: #aad3df;
}
```

Este bloque se ha reservado para la ubicación de una tabla con los resultados obtenidos del código gnuais.py. En la misma, se encuentran el mmsi, la latitud y la longitud de los últimos barcos cuyas señales han sido captadas. El funcionamiento de la tabla y la descripción de los scripts necesarios serán detallados en la última parte de este capítulo dado que es común para todas las páginas.

Quedando finalmente el contenedor “Izquierda” como se ve en la Figura 4.11.

A su derecha, se ha ubicado el contenedor “Derecha”, cuyas características se encuentran recogidas en el fichero “style.css”. Como se puede ver en el siguiente fragmento de código, en él se detallan las dimensiones de este contenedor, su color, una imagen de fondo, etc.

```
.Derecha {
  width: 70%;
```

MMSI	LAT	LON
224781000	28.469747	-16.244275
224131000	28.469887	-16.246148
224781000	28.469743	-16.244277
224185000	28.464813	-16.245358
224474000	28.477863	-16.236975
224522000	28.476967	-16.241533
224185000	28.46481	-16.245357
351481000	28.450497	-16.246248
227457190	28.465953	-16.244493
224781000	28.469747	-16.244277

Figura 4.11: Tabla de la página Mapa.

Figura 4.12: Mapa de la página Mapa.

```
  height: 70%;
  background: #aad3df;
  background-image: url(../img/jesus.png) ;
  background-position: center;
  background-repeat: no-repeat;
  background-size: 68%;
}
```

Como se puede ver en el siguiente fragmento de código, en su interior irá ubicado el mapa creado por el código mapa.py.

```
<div class="Derecha">
   <iframe id="myFrame" src="mapa.html"
     style="width: 100%; height: 100%;border: 0px;"></iframe>
 </div>
```

El resultado final del contenedor “Derecha” se muestra como aparece en la Figura 4.12.

Finalmente, esta página termina de formarse con el contenedor llamado “Inferior”, cuyas características vienen descritas en el fichero “style.css”.

```
.Inferior {
  width: 100%;
  height: 15%;
  background: #aad3df;
}
```

Como se puede ver en el siguiente fragmento del código index.php, esta es la parte de la página donde van ubicados el resto de los botones, los cuales son los responsables de que el usuario se pueda mover entre las distintas páginas y así visualizar los diferentes registros. Empieza, por específicar las clases de los botones, que como se ha mencionado con anterioridad, es la misma clase que los botones ubicados en el contenedor "Superior", con la única diferencia de que en vez de la clase Debris como en el caso de el botón apagar, la clase de estos botones pasa a ser llamada Coral.

```
<div class="Inferior">
   <div class="btn-bg Ocean">
     <div class="btn-group">
       <div class="btn Coral">
```

Al igual que con la clase Debris, la Coral engloba un conjunto de etiquetas. En la primera se muestran una serie de características como puede ser el color, el radio del borde, el fondo, etc.

```
body .btn-bg.Ocean .btn-group .Coral button {
  color: #fff;
  overflow: hidden;
  position: relative;
  border-radius: 5px;
  border: 5px solid #fff;
  background: transparent;
  transition: all 1s ease;
  font-family: Monospace;
  font-weight: bold;
}
```

La segunda etiqueta hace referencia a la animaciones de su interior, el tamaño de la misma, la posición, etc.

```
body .btn-bg.Ocean .btn-group .Coral button span {
  position: absolute;
  content: '';
  top: 1.75em;
  left: 50%;
  width: 20em;
  height: 20em;
  margin-left: -10em;
  border-radius: 42.5%;
  transform-origin: 50% 50%;
  transition: all 1s ease, top 1.5s ease;
}
```

La tercera, se encarga de los cambios que se producen en el botón si el cursor se ubica encima del mismo.

```
body .btn-bg.Ocean .btn-group .Coral button:hover {
  transition: all 1s ease;
  font-family: Monospace;
}
```

Y la cuarta y última se encarga de los cambios que se producen en la animación cuando el cursor se desplaza y se posiciona encima del botón.

```
body .btn-bg.Ocean .btn-group .Coral button:hover span {
  top: 0.5em;
  transition: all 1s ease, top 1.5s ease;
}
```

Volviendo al index.php, como se puede ver en el código, se crean los botones para la navegación de la página. El primero con el que nos encontramos es el botón llamado “Mapa”. Este, en esta página no tiene utilidad alguna ya que nos dirigiría al mismo lugar del que partimos. A continuación, vienen los siguientes 3 botones, llamados “Climatología” (que moverá al usuario a la página llamada “Temp” y donde se encuentra el registro creado por el código “temperatura.py”), “Registro 1” (que moverá al usuario a la página llamada “Coord” y donde se encuentra el registro creado por el código “gnuais.py” con los mensajes del tipo 1) y “Registro 2” (que moverá al usuario a la página llamada “Dest”, donde se encuentra el registro creado por el código “gnuais.py” con los mensajes del tipo 2).

```
<button href="#">Mapa<span class="Coralwave1"></span><span class
    ="Coralwave2"></span><span class="Coralwave3"></span></button
    >
        </div>
        <div class="btn Coral">
          <button href="#"  onclick="return show('Temp','Mapa
    ');">Climatolo
            <span class="Coralwave1"></span>
            <span class="Coralwave2"></span>
            <span class="Coralwave3"></span>
          </button>
        </div>
        <div class="btn Coral">
          <button href="#"  onclick="return show('Coord','Mapa
    ');">Registro 1<span class="Coralwave1"></span><span class="
    Coralwave2"></span><span class="Coralwave3"></span></button>
        </div>
        <div class="btn Coral">
          <button href="#"  onclick="return show('Dest','Mapa
    ');">Registro 2<span class="Coralwave1"></span><span class="
    Coralwave2"></span><span class="Coralwave3"></span></button>
        </div>
```

Figura 4.13: Parte Inferior.

Figura 4.14: Página Mapa.

Como se puede ver en el fragmento, cada botón alberga dentro de si 3 subclases. Estas solamente modifican los colores y el tiempo de la animación que se encuentra dentro de los botones.

```
body .btn-bg.Ocean .btn-group .Coral button .Coralwave1 {
  background: #ffb733;
  animation: smallwave 3s infinite linear;
}
body .btn-bg.Ocean .btn-group .Coral button .Coralwave2 {
  background: #f97890;
  animation: smallwave 4s infinite linear;
}
body .btn-bg.Ocean .btn-group .Coral button .Coralwave3 {
  background: #4ab3e1;
  animation: smallwave 5s infinite linear;
  font-family: Monospace;
}
```

Finalmente, el bloque “Inferior” se ve tal y como aparece en la Figura 4.13.

El resultado de la página “Mapa” al completo, se muestra en la Figura 4.14:

Figura 4.15: Página Climatología.

Temp, Coord y Dest

Estas 3 páginas están creadas de la misma manera que la página Mapa, la única diferencia es la división de las mismas. En este caso, la división es más sencilla, ya que en su interior, las 3 páginas albergarán un único contenedor llamado "Completo" donde se encontrará la tabla de los datos extraídos de los registros "csv". Como se ha comentado en la sección anterior, la descripción de los scripts necesarios para la creación, actualización y visualización de las tablas que se encuentran en las distintas páginas, será detallada en la siguiente sección.

Las características del bloque "Completo", vienen dadas en el fichero "style.csv" y como en el caso de los anteriores contenedores, hacen referencia al tamaño, el color, la imagen de fondo, etc.

```
.Completo {
  width: 100%;
  height: 70%;
  background: #aad3df;
  background-image: url(../img/jesus.png) ;
  background-position: center;
  background-repeat: no-repeat;
  background-size: 68%;
}
```

Finalmente, el resultado de las páginas se muestran en las Figuras 4.15, 4.16, 4.17:

Figura 4.16: Página Registro 1.

Figura 4.17: Página Registro 2.

Tablas

En este apartado, se detallarán los pasos que se han seguido para la creación y actualización a tiempo real de las tablas. Como se puede ver en el siguiente fragmento de código, a cada una de las tablas, marcadas como un "id" en el fichero index, se les referencia y se crea la variable "data" con todos los campos a rellenar. A continuación, se llama a la función "generateTableHead" y se le pasan estas dos variables.

```
let table = document.getElementById("mapaTabla");
let data = ['mmsi', 'lat', 'lon'];
generateTableHead(table, data);

table = document.getElementById("tempTabla");
data = ['Temperatura', 'Presión', 'Humedad'];
generateTableHead(table, data);

table = document.getElementById("coordTabla");
data = ['hora', 'mmsi', 'Lat', 'Lon', 'course', 'speed', '
    rateofturn', 'navstat', 'heading'];
generateTableHead(table, data);

table = document.getElementById("destTabla");
data = ['hora', 'mmsi', 'name', 'destination', 'type', 'length',
     'width', 'draught'];
generateTableHead(table, data);
```

Esta función sirve para crear una cabecera para las tablas. Como se puede ver, en el siguiente fragmento, donde aparece en su plenitud dicha función, primeramente declara la variable "thead" creando una cabecera para las tablas. Luego, a dicha variable se le inserta una nueva fila. Para cada una de las palabras que se encuentran dentro de la variable data, se crea un elemento HTML llamado "th" cuyas características serán posteriormente explicadas. Se escriben a través del método "createTextNode" las palabras de la variable data (que son los distintos campos dentro de las bases de datos) y finalmente, con el método "appendChild" se enlazan las distintas variables, siendo el texto un nuevo nodo del elemento "th" y este un nodo de la fila.

```
function generateTableHead(table, data) {
  let thead = table.createTHead();
  let row = thead.insertRow();
  for (let key of data) {
    let th = document.createElement("th");
    let text = document.createTextNode(key);
    th.appendChild(text);
    row.appendChild(th);
  }
```

```
}
```

Como se ha mencionado anteriormente, el ahora elemento HTML “th” tiene unas características que están incorporadas en el fichero “style.css” donde se recogen el color de las celdas, el tamaño del texto. etc.

```
th {
  background-color: #6aa4b5;
  text-align: center;
  cursor: pointer;
  overflow: visible;
  outline: none;
  color: #fff;
  position: relative;
  letter-spacing: 0.2em;
  padding: 10px;
  text-transform: uppercase;
  font-family: Monospace;
  font-size: 15px;
  font-weight: bold;
}
```

Siguiendo con el fichero index, continúa con el método “setInterval”, el cual ejecuta una función cada determinado tiempo, en este caso y como se puede ver en el siguiente fragmento de código, 30 segundos.

```
setInterval(function() {
    leerFichero ('datos/temperatura.csv', 'tempTabla',
    numeroLineasTemp);
    leerFichero1('datos/gnuaisCoord.csv', 'mapaTabla',
    numeroLineasMapa);
    leerFichero2('datos/gnuaisDest.csv' , 'destTabla',
    numeroLineasDest);
    document.getElementById('myFrame').setAttribute('src', 'mapa
    .html');
  }, 30000);
```

Como se puede ver, este ejecuta 3 funciones, sin embargo, en este proyecto se detallará el funcionamiento de la función “leerFichero” ya que el resto son iguales salvo por 1 línea, la cuál se verá a continuación. A estas funciones se le envía el fichero csv (la base de datos), el elemento tabla y el número de líneas.

```
const leerFichero = async (file, tabla, numeroLineas) => {
    let lineasTotales;
    const response = await fetch(file)
    const text = await response.text()
    var splitArray = text.split('\n');
    lineasTotales = Number(splitArray.length - 1);
    if (lineasTotales - numeroLineas > 10) {
```

```
8       envia = splitArray.slice(lineasTotales - 10, lineasTotales
    );
9     } else {
10      envia = splitArray.slice(numeroLineas, lineasTotales);
11    }
12    if (lineasTotales > 1) {
13      actualizarTabla(envia, tabla);
14    }
15    numeroLineasTemp = lineasTotales;
16  }
```

Como se puede apreciar, la función abre el fichero csv y lo lee, para luego separarlo por saltos de línea. A la variable lineasTotales se le pasa el valor del número de líneas del fichero, y dependiendo del valor de la resta, se recogen las últimas 10 líneas del fichero o las últimas recibidas. A continuación y con el resultado adquirido, se llama a la función "actualizarTabla". Finalmente se actualiza el número de líneas para en el siguiente bucle, compararlo de nuevo con el número de líneas del fichero csv.

```
1  function actualizarTabla(lineas, tabla) {
2    var table = document.getElementById(tabla);
3    if (tabla === "mapaTabla") {
4      if (Array.isArray(lineas)) {
5        lineas = lineas.reverse();
6        for(let i = 0; i < lineas.length; i++) {
7          var lineaSplit = lineas[i].split(',');
8          table.innerHTML = "<tr><td>" + Number(lineaSplit[1]) + "
     </td><td>"
9                                      + Number(lineaSplit[2]) + "
     </td><td>"
10                                     + Number(lineaSplit[3]) + "
     </td></tr>" + table.innerHTML;
11       }
12     }
```

Como se puede ver, esta función está preparada para identificar a la variable tabla según el id del html, en este caso "mapaTabla". A continuación, el bucle for recorre las líneas que se han enviado desde la función "leerFichero", separando dentro de cada una los datos mediante comas. Finalmente la propiedad "innerHTML" da la sintaxis HTML para cada uno de los datos recogidos de las líneas, actualizando la tabla con los últimos valores recogidos.

Capítulo 5

Resultados

En este capítulo se detallarán las pruebas efectuadas el día 31 de Agosto de 2022 en el puerto de Los Cristianos y en el puerto de Santa Cruz. Se aportarán imágenes de la pantalla de la estación AIS portátil, parte del resultado de los registros, algunas fotografías de los barcos cuyos mensajes AIS han sido captados e imágenes de la página "Vesselfinder"[10] a modo de comprobación de los mmsi captados.

5.1. Prueba 1: Puerto de Los Cristianos

Como se puede ver en la Figura 5.1, esta es la pantalla principal de la estación AIS portátil, donde se encuentra un registro del MMSI, la latitud y la longitud de los barcos de alrededor y el mapa de la costa oeste de Tenerife. En el registro, solo aparecen dos mmsi distintos, ya que solo habían dos barcos emitiendo señales AIS en ese momento. En el mapa, aparece el recorrido de un barco que estaba atracando justo en el momento de la prueba. Este, cuyo mmsi es el número 224836000, es el barco "Volcán de Tirajana"de la compañía Naviera Armas. Como se puede ver en la Figura 5.2, esta es una foto tomada en el momento de la recepción de señales a dicho barco y en la Figura 5.3 sacada de la página Vesselfinder aparece una tabla con los datos del mismo a modo de comprobación del MMSI.

A continuación, en las Figuras 5.4, 5.5, 5.6, 5.7, 5.8, 5.9 se mostrarán las imágenes de las distintas páginas de la estación AIS y los extractos de los registros obtenidos.

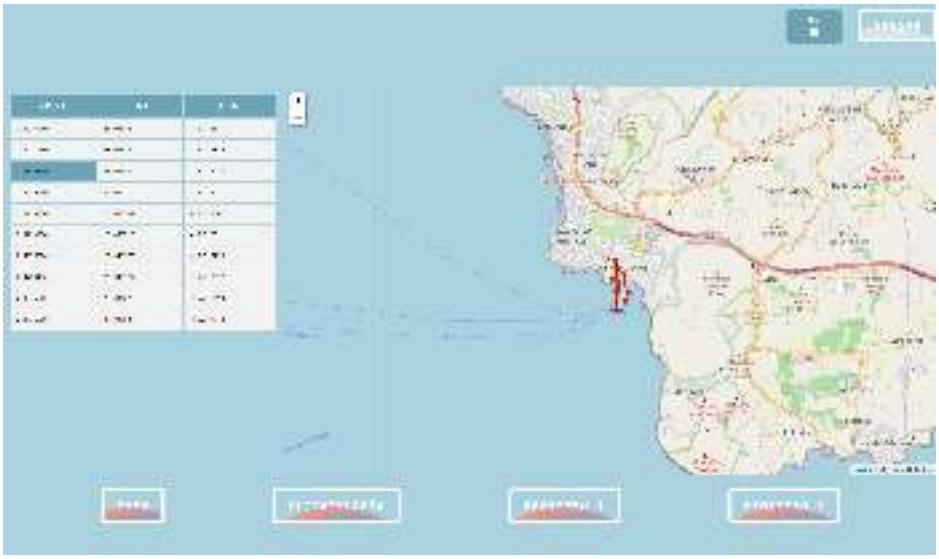


Figura 5.1: Página del Mapa de la prueba 1.

Figura 5.2: Foto tomada al barco Volcán de Tirajana.

Figura 5.3: Imagen de la página Vesselfinder del barco Volcán de Tirajana.

Figura 5.4: Página de Climatología de la prueba 1.

```
 3    Temperatura,Presión,Humedad
 4    22.64,680.45,78.05
 5    33.04,1016.6,55.81
 6    33.07,1016.6,55.76
 7    33.09,1016.61,55.73
 8    33.11,1016.61,55.68
 9    33.13,1016.61,55.64
10    33.16,1016.62,55.58
11    33.18,1016.61,55.55
```

Figura 5.5: Registro de climatología de la prueba 1.

Figura 5.6: Página del Registro 1 de la prueba 1.

```
1   hora,mmsi,Lat,Lon,course,speed,rateofturn,navstat,heading
2   15:41:14.553370,224223430,28.048750,-16.718495,360,0.0,224,1,511
3   15:41:25.089703,224223430,28.048750,-16.718493,360,0.0,224,1,511
4   15:41:55.823637,224223430,28.048753,-16.718498,360,0.0,224,1,511
5   15:48:17.235718,224223430,28.048740,-16.718508,360,0.1,224,1,511
6   15:51:50.555015,224836000,28.041027,-16.718433,22,17.8,32,0,21
7   15:52:31.246036,224836000,28.043187,-16.717530,21,10.0,0,0,21
8   15:52:40.203478,224836000,28.043440,-16.717422,21,9.8,31,0,22
9   15:53:17.910006,224223430,28.048730,-16.718515,360,0.0,224,1,511
10  15:53:56.957392,224836000,28.045440,-16.716192,43,3.6,31,0,44
11  15:57:04.986539,224836000,28.046607,-16.717048,302,1.8,32,0,125
12  15:57:35.951403,224836000,28.046620,-16.717198,236,0.6,31,0,126
13  15:58:09.044409,224223430,28.048727,-16.718512,360,0.0,224,1,511
14  15:59:41.814947,224836000,28.046573,-16.717238,336,0.1,0,0,127
15  16:43:40.661867,224223430,28.048723,-16.718500,360,0.1,224,1,511
16  16:46:10.632676,224223430,28.048707,-16.718507,360,0.0,224,1,511
17  16:47:04.688124,224836000,28.046573,-16.717237,336,0.0,0,0,126
18  16:47:56.442570,224836000,28.046580,-16.717233,336,0.0,0,0,126
19  16:48:40.838027,224223430,28.048720,-16.718512,360,0.0,224,1,511
20  16:50:24.009983,224223430,28.048720,-16.718515,360,0.2,224,1,511
21  16:57:15.309537,224223430,28.048713,-16.718527,360,0.0,224,1,511
22  16:57:44.352460,224223430,28.048720,-16.718503,360,0.0,224,1,511
23  16:58:35.223624,224836000,28.046573,-16.717232,336,0.0,0,0,126
24  16:58:55.489395,224836000,28.046573,-16.717232,336,0.0,0,0,126
```

Figura 5.7: Registro de los mensajes de tipo 1 de la prueba 1.

Figura 5.8: Página del Registro 2 de la prueba 1.

```
1   hora,Lat,Lon
2   2022-08-31T14:21:59.000Z,28.049202200,-16.719849200
3   2022-08-31T14:22:01.000Z,28.049202200,-16.719849200
4   2022-08-31T14:22:03.000Z,28.049202200,-16.719849200
5   2022-08-31T14:22:05.000Z,28.049202200,-16.719849200
6   2022-08-31T14:22:07.000Z,28.049202200,-16.719849200
7   2022-08-31T14:22:09.000Z,28.049202200,-16.719849200
8   2022-08-31T14:22:11.000Z,28.049202200,-16.719849200
9   2022-08-31T14:22:13.000Z,28.049202200,-16.719849200
10  2022-08-31T14:22:15.000Z,28.049202200,-16.719849200
11  2022-08-31T14:22:17.000Z,28.049202200,-16.719849200
12  2022-08-31T14:22:19.000Z,28.049202200,-16.719849200
13  2022-08-31T14:22:21.000Z,28.049202200,-16.719849200
14  2022-08-31T14:22:23.000Z,28.049202200,-16.719849200
15  2022-08-31T14:22:25.000Z,28.049202200,-16.719849200
16  2022-08-31T14:22:27.000Z,28.049202200,-16.719849200
17  2022-08-31T14:22:29.000Z,28.049202200,-16.719849200
```

Figura 5.9: Registro del GPS de la prueba 1.

5.2. Prueba 2: Puerto de Santa Cruz

Esta prueba se realizó en el puerto de Santa Cruz a las 18:00 de la tarde. Como se puede ver en la Figura 5.10, al igual que en la prueba 1, aparece un registro con el MMSI, la latitud y la longitud de los barcos cuyos mensajes AIS se han recibido. En el mapa, aparecen muchos más barcos que en la prueba anterior, ya que el muelle de Santa Cruz es mucho más concurrido que el de Los Cristianos. A modo de ejemplo, en este apartado se mostrán algunos de ellos.

Si se introducen los mmsi que se encuentran en el registro creado por la estación AIS en la página Vesselfinder, se puede comprobar qué barcos fueron los encontrados. A continuación, en la Figura 5.11 se muestra el registro de mensajes tipo 1 obtenido y en las Figuras 5.12, 5.13, 5.14 se puede ver la comprobación a través de las hojas de datos de los barcos en la web Vesselfinder.

Por último, en las Figuras 5.15, 5.16, 5.17, 5.18 se muestran el resto de las pantallas de la estación AIS portátil, junto con los registros creados por la misma.

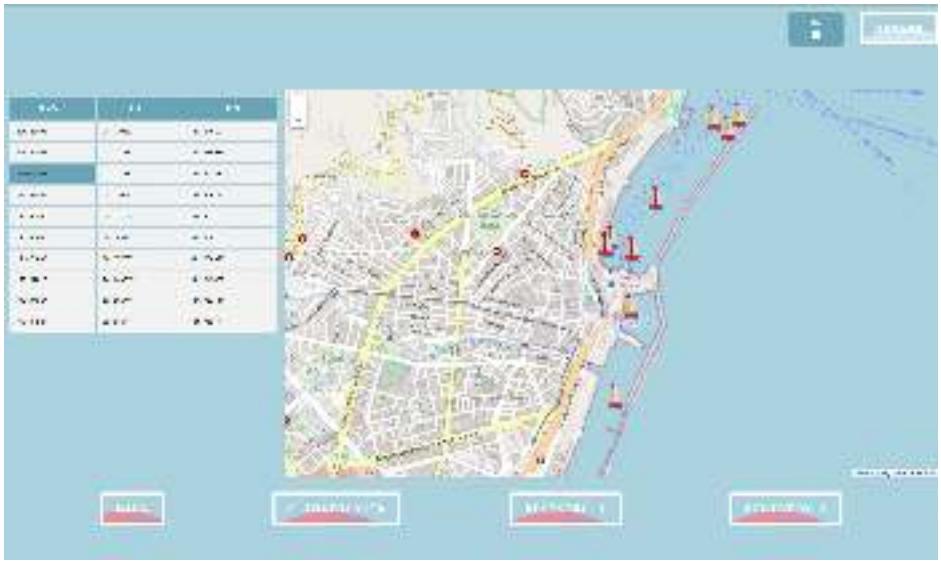


Figura 5.10: Página del Mapa de la prueba 2.

```
25  hora,mmsi,Lat,Lon,course,speed,rateofturn,navstat,heading
26  17:22:26.030534,224984000,28.469787,-16.244262,207,0.0,64,2,16
27  17:22:45.882525,224984000,28.469798,-16.244262,207,0.0,64,2,16
28  17:25:11.678142,211856260,28.465717,-16.243873,360,0.0,0,15,511
29  17:27:40.543745,224984000,28.469787,-16.244262,207,0.0,64,2,16
30  17:29:43.433211,224984000,28.469783,-16.244260,207,0.0,64,2,16
31  17:30:56.993698,224185000,28.477087,-16.237087,140,0.1,0,2,214
32  17:37:18.642350,224388000,28.470500,-16.245718,209,4.1,0,0,15
33  17:38:10.910685,224575460,28.466330,-16.244407,360,0.0,0,15,511
34  17:38:28.608738,224388000,28.470043,-16.246083,251,0.5,0,0,7
35  17:38:46.357268,224984000,28.469783,-16.244270,207,0.0,64,2,15
36  17:39:48.275486,224575460,28.466340,-16.244403,360,0.0,0,15,511
37  17:40:40.772505,224388000,28.470033,-16.246120,322,0.0,0,0,3
38  17:41:10.058607,224388000,28.470033,-16.246118,69,0.0,0,0,3
39  17:41:57.416829,227232820,28.468867,-16.245467,35,2.2,0,15,511
40  17:43:28.932882,224474000,28.477887,-16.236977,246,0.0,32,0,511
41  17:44:39.029146,224984000,28.469700,-16.244265,207,0.0,64,2,15
42  17:48:20.118557,224185000,28.477000,-16.237700,140,0.1,0,2,214
43  17:49:22.809918,224388000,28.470033,-16.246122,287,0.0,0,0,3
44  17:52:32.339217,224984000,28.472880,-16.242657,26,7.0,64,2,28
45  17:55:44.166110,224474000,28.477800,-16.236960,72,0.0,32,0,511
46  18:00:54.483247,224388000,28.470033,-16.246120,308,0.0,0,0,3
47  19:46:21.406777,225430000,28.477720,-16.238093,218,7.2,0,0,218
48  19:47:48.952584,224388000,28.470033,-16.246125,162,0.0,0,0,3
49  19:48:56.750695,224388000,28.470033,-16.246123,128,0.0,0,0,2
```

Figura 5.11: Registro de los mensajes de tipo 1 de la prueba 2.

Figura 5.12: Imagen de la página Vesselfinder del barco Banaderos Express.

Figura 5.13: Imagen de la página Vesselfinder del barco Villa de Tazacorte.

Figura 5.14: Imagen de la página Vesselfinder del barco Volcán de Taidia.

Figura 5.15: Página de Climatología de la prueba 2.

```
3097    Temperatura,Presión,Humedad
3098    22.64,680.45,78.05
3099    25.74,962.42,58.06
3100    25.76,962.42,58.08
3101    25.77,962.43,58.1
3102    25.77,962.43,58.12
3103    25.78,962.42,58.15
3104    25.79,962.42,58.17
3105    25.79,962.43,58.2
3106    25.79,962.4,58.23
3107    25.8,962.41,58.26
```

Figura 5.16: Registro de climatología de la prueba 2.

Figura 5.17: Página del Registro 1 de la prueba 2.

```
1740  hora,Lat,Lon
1741  2022-08-31T17:58:04.000Z,28.468452700,-16.245684300
1742  2022-08-31T17:58:06.000Z,28.468452700,-16.245684300
1743  2022-08-31T17:58:08.000Z,28.468452700,-16.245684300
1744  2022-08-31T17:58:10.000Z,28.468452700,-16.245684300
1745  2022-08-31T17:58:12.000Z,28.468452700,-16.245684300
1746  2022-08-31T17:58:14.000Z,28.468452700,-16.245684300
1747  2022-08-31T17:58:16.000Z,28.468452700,-16.245684300
1748  2022-08-31T17:58:18.000Z,28.468452700,-16.245684300
1749  2022-08-31T17:58:20.000Z,28.468452700,-16.245684300
1750  2022-08-31T17:58:22.000Z,28.468452700,-16.245684300
1751  2022-08-31T17:58:24.000Z,28.468452700,-16.245684300
1752  2022-08-31T17:58:26.000Z,28.468452700,-16.245684300
1753  2022-08-31T17:58:28.000Z,28.468452700,-16.245684300
1754  2022-08-31T17:58:30.000Z,28.468452700,-16.245684300
```

Figura 5.18: Registro del GPS de la prueba 2.

Bibliografía

[1] Edwin Andrés Quintero. Andres Agudelo R. Pablo Cesar Bernal G. «Modulación GMSK para transmisión de información a través de líneas eléctricas.» En: (2010).

[2] *Espectro electromagnético*. URL: https://es.wikipedia.org/wiki/Espectro_electromagn%C3%A9tico.

[3] Parlamento Europeo. *Modificación de la Directiva 2002/59/CE relativa al establecimiento de un sistema comunitario de seguimiento y de información sobre el tráfico marítimo*. 2009.

[4] Fernando. *SDR : EQUIPOS DE RADIO DEFINIDOS POR SOFTWARE*. 2008. URL: https://www.ea1uro.com/sdr1/sdr.htm.

[5] *NMEA 0183*. URL: https://es.wikipedia.org/wiki/NMEA_0183.

[6] Nicolás Molina Padrón. *Diseño e Implementación de un prototipo hardware para un banco de pruebas del estándar AIS*. 2015.

[7] Bob Stewart y col. *Software Defined Radio using MATLAB and Simulink and the RTL-SDR*. 2017.

[8] Unión Internacional de Telecomunicaciones. *Características técnicas de un sistema de identificación automático mediante acceso múltiple por división en el tiempo en la banda de frecuencias de ondas métricas del servicio móvil marítimo*. 2014.

[9] v3l0c1r4pt0r. *RTL-SDR FM Radio Receiver With GNU Radio Companion*. 9 de septiembre de 2022. URL: https://www.instructables.com/RTL-SDR-FM-radio-receiver-with-GNU-Radio-Companion/.

[10] *VesselFinder*. URL: https://www.vesselfinder.com/es.

[11] *What is GMSK Modulation - Gaussian Minimum Shift Keying*. URL: https://www.electronics-notes.com/articles/radio/modulation/what-is-gmsk-gaussian-minimum-shift-keying.php.

Universidad de La Laguna
Escuela Superior de Ingeniería y Tecnología
Departamento de Ingeniería Industrial

Trabajo de Fin de Grado

Diseño de una estación AIS portátil

TOMO II

Presupuesto

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Estudiante:

Jesús Galván Santos

Tutor: Fernando Luis Rosa González

13 de septiembre de 2022

Capítulo 6

Presupuesto

6.1. Presupuesto

En esta sección se detallarán las mediciones realizadas para llevar a cabo el diseño y la fabricación de la estación AIS portátil. Para ello, se detallará el costo de las actividades en la Tabla 6.1, de los materiales necesarios para la creación del prototipo en la Tabla 6.2 y del costo total en la Tabla 6.1.

Mano de obra			
Concepto	**Horas**	**Precio por hora**	**Precio total**
Diseño	90	30.00	2700 €
Programación	60	30.00	1800 €
Fabricación y pruebas	40	25.00	1000 €
Redacción	100	20.00	2000 €
-	**290 horas**	-	**7500 €**
Costo Total			
Concepto		**Precio**	
Mano de Obra		7500 €	
Materiales		371,01 €	
Precio Total		**7871,01 €**	

Tabla 6.1: Costo de la mano de obra.

Materiales					
Componente	**Modelo**	**Precio(€/Ud)**	**Unidad/es**	**Precio Total**	**Precio por 10 Unidades (-10 %)**
Unidad de Control	Raspberry Pi 3 Model B	47,99 €	1	47,99 €	431,91 €
Pantalla	Raspberry Pi3 7"Touchscreen Display	79,00 €	1	79,00 €	711,00 €
Montaje USB Empotrado	CentBest FMCUB23	15,99 €	1	15,99 €	143,91 €
Antena GPS	DIYmalls VK-162	26,99 €	1	26,99 €	242,91 €
Carcasa	-	20 €	1	20 €	180 €
Powerbank	XO-PR144	28,50 €	1	28,50 €	256,50 €
Sensor temperatura, humedad y presión	BME-280	18,49 €	1	18,49 €	166,41 €
MicroSD	SanDisk Ultra PLUS 64GB	9,99 €	1	9,99 €	89,91 €
Antena RF	TengKo Small car sma-k	9,99 €	1	9,99 €	89,91 €
Receptor IQ	RTL-SDR V.3	80,83 €	1	80,83 €	727,47 €
Adaptador SMA	SMA Macho-Macho	8,99 €	1	8,99 €	80,91 €
Interruptor	Interruptor por llaves	8,40 €	1	8,40 €	75,60 €
Adaptador USB	USB Hembra-Hembra	4,88 €	1	4,88 €	43,92 €
Cable USB	USB - USB C	5,09 €	1	5,09 €	45,81 €
Cable USB	USB Macho-Macho	4,50 €	1	4,50 €	40,50 €
Cable USB	USB - MicroUSB	4,90 €	1	4,90 €	44,10 €
Tornillos tapa	Cabeza avellanada acero métrica 4 50mm	0,17 €	4	0,68 €	6,12 €
Tuerca ciega	Métrica 4	0,10 €	4	0,40 €	3,60 €
Clavos visagra	1,3 20mm	0,01 €	4	0,04 €	0,36 €
Barra Aluminio Pantalla	1m 10x2mm	0,30 €	2	0,6 €	5,40 €
Barra Acero Sujeción Magnética	1m 23,5x1,2mm	0,13 €	2	0,26 €	2,34 €
Tornillos Pantalla	Cabeza avellanada acero métrica 3 10mm	0,07 €	4	0,28 €	2,52 €
Velcro	Redondo	0,02 €	5	0,10 €	0,90 €
Imánes	Redondos 3mm	0,17 €	4	0,67 €	5,98 €
Precios Totales	-	-	-	**371,01 €**	**3357,72 €**

Tabla 6.2: Materiales.

Universidad de La Laguna
Escuela Superior de Ingeniería y Tecnología
Departamento de Ingeniería Industrial

Trabajo de Fin de Grado

Diseño de una estación AIS portátil

TOMO III

Anexos

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Estudiante:

Jesús Galván Santos

Tutor: Fernando Luis Rosa González

13 de septiembre de 2022

Capítulo 7

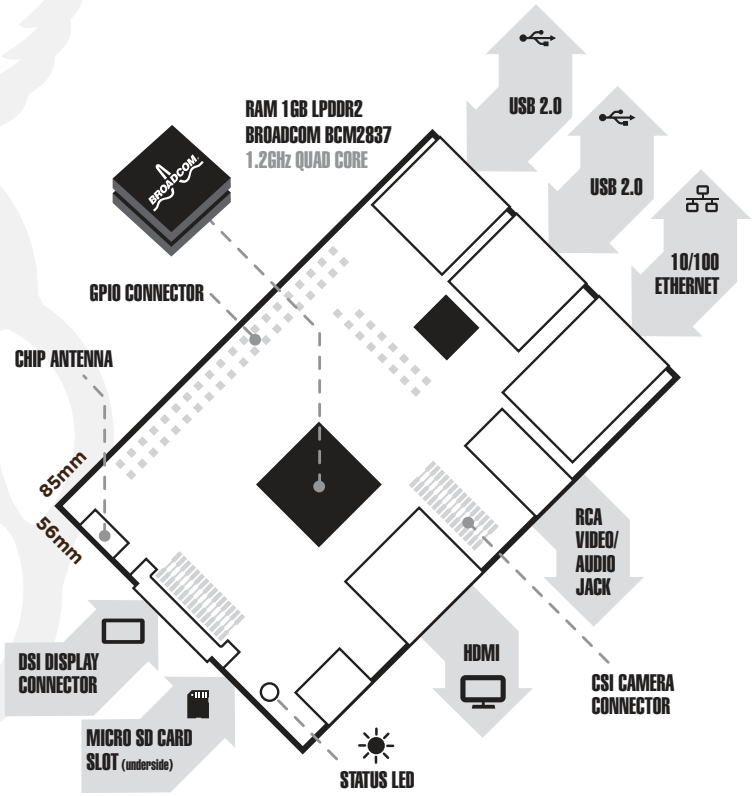
Anexos

7.1. Rasberry Datasheet

Raspberry Pi

Raspberry Pi 3 Model B

Product Name	**Raspberry Pi 3**
Product Description	The Raspberry Pi 3 Model B is the third generation Raspberry Pi. This powerful credit-card sized single board computer can be used for many applications and supersedes the original Raspberry Pi Model B+ and Raspberry Pi 2 Model B. Whilst maintaining the popular board format the Raspberry Pi 3 Model B brings you a more powerful processer, 10x faster than the first generation Raspberry Pi. Additionally it adds wireless LAN & Bluetooth connectivity making it the ideal solution for powerful connected designs.
RS Part Number	**896-8660**



RS

www.rs-components.com/raspberrypi

Raspberry Pi

Raspberry Pi 3 Model B

Specifications

Processor	Broadcom BCM2387 chipset. 1.2GHz Quad-Core ARM Cortex-A53 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
GPU	Dual Core VideoCore IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	1GB LPDDR2
Operating System	Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V1, 2.5A

Connectors:

Ethernet	10/100 BaseT Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4 Composite RCA (PAL and NTSC)
Audio Output	Audio Output 3.5mm jack, HDMI USB 4 x USB 2.0 Connector
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
Memory Card Slot	Push/pull Micro SDIO

Key Benefits

- Low cost
- Consistent board format
- 10x faster processing
- Added connectivity

Key Applications

- Low cost PC/tablet/laptop
- IoT applications
- Media centre
- Robotics
- Industrial/Home automation
- Server/cloud server
- Print server
- Security monitoring
- Web camera
- Gaming
- Wireless access point
- Environmental sensing/monitoring (e.g. weather station)

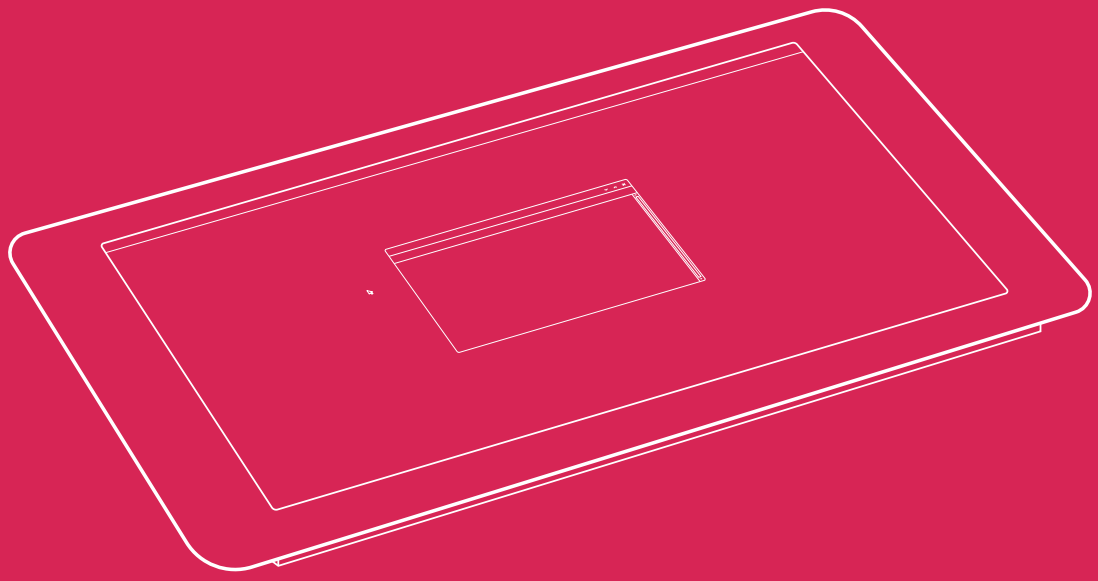
RS

www.rs-components.com/raspberrypi

7.2. Rasberry Pi Touch Display Datasheet

Raspberry Pi Touch Display

Published November 2021



Raspberry Pi Ltd

Overview

This 7" touchscreen display for Raspberry Pi lets you create interactive projects such as tablets, entertainment systems, and information dashboards.

Raspberry Pi OS provides touchscreen drivers with support for ten-finger touch and an on-screen keyboard, giving you full functionality without the need to connect a keyboard or mouse.

The 800 x 480 display connects to Raspberry Pi via an adapter board that handles power and signal conversion. Only two connections to your Raspberry Pi are required: power from the GPIO port, and a ribbon cable that connects to the DSI port on all Raspberry Pi computers except for the Raspberry Pi Zero line.

Specification

Assembly module size:	192.96mm × 110.76mm
Display size (diagonal):	7 inches
Display format:	800 (RGB) × 480 pixels
Active area:	154.08mm × 85.92mm
LCD type:	TFT, normally white, transmissive
Touch panel:	True multi-touch capacitive touch panel with up to 10 points of absolution
Surface treatment:	Anti-glare
Colour configuration:	RGB-stripe
Backlight type:	LED B/L
Compliance:	For a full list of local and regional product approvals, please visit pip.raspberrypi.com

For our whitepaper on designing with the Raspberry Pi Touch Display, email applications@raspberrypi.com

SAFETY INSTRUCTIONS

To avoid malfunction or damage to this product, please observe the following:

- **Before connecting the device, shut down your Raspberry Pi computer and disconnect it from external power.**
- If the cable becomes detached, pull the locking mechanism forward on the connector, insert the ribbon cable ensuring the metal contacts face towards the circuit board, then push the locking mechanism back into place.
- This device should be operated in a dry environment at 0–50°C.
- Do not expose it to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose it to excessive heat from any source.
- Care should be taken not to fold or strain the ribbon cable.
- Care should be taken when screwing in parts. A cross-thread can cause irreparable damage and void the warranty.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Avoid handling the printed circuit board whilst it is powered and only handle by the edges to minimise the risk of electrostatic discharge damage.
- Store in a cool, dry location.
- Avoid rapid changes of temperature, which can cause moisture build up in the device.

Raspberry Pi

Raspberry Pi is a trademark of Raspberry Pi Ltd

7.3. RTL-SD Blog V3 Datasheet

RTL-SDR Blog V3 Datasheet

The RTL-SDR Blog V3 is an improved RTL-SDR dongle. RTL-SDR dongles were originally designed for DVB-T HDTV reception, but they were found by hardware hackers to be useful as a general purpose SDR. The standard dongles are okay for DVB-T reception, but are just barely suitable for SDR users/experimenters. The RTL-SDR Blog V3 was redesigned with SDR user needs in mind, instead of DVB-T HDTV users who typically have more relaxed requirements.

Purchase at: www.rtl-sdr.com/store

Quickstart setup guide available at: www.rtl-sdr.com/qsg

Basic Information

- **Bandwidth:** Up to 2.4 MHz stable.
- **ADC:** RTL2832U 8-bits
- **Frequency Range:** 500 kHz – 1766 MHz (500 kHz – 24 MHz in direct sampling mode)
- **Typical Input Impedance:** 50 Ohms
- **Typical Current Draw:** 270 – 280 mA

Required Computing Hardware

Same requirements as a regular RTL-SDR. Compatible with Windows XP and above (SDR# requires Win 7 or newer), Linux, MacOS and Android. A dual core machine is recommended.

Single board PCs like the Raspberry Pi, Odroid, C.H.I.P are also supported with most command line apps.

RTL-SDR V3 Improvements over generic models

TCXO

The V3 uses a 1PPM TCXO for excellent frequency stability. The temperature drift is around 0.5 – 1 PPM, and the initial offset is 0 – 2 PPM. This means that the signal will not drift on the spectrum as the dongle or ambient temperature changes. Also, the frequency offset will be close to zero. Standard dongles have a PPM offset of up to 100PPM, and tend to drift a lot. Using a TCXO solves these problems.

SMA Connector

Typical RTL-SDR dongles use a relatively obscure MCX RF connector. The V3 uses commonly used SMA connectors, so it is easy to obtain adapters, connectors and antennas for the unit. SMA connectors also last longer.

Aluminium Enclosure

Unlike standard RTL-SDR's, the V3 comes standard with an aluminium enclosure. The enclosure has two purposes. The first is to help block any RF interference from entering through the PCB. The second is to act as a heatsink to the PCB.

Improved Heat Dissipation

Typical R820T/2 RTL-SDR dongles tend to lose PLL lock in L-band at around 1.5 GHz and above, causing a loss of reception to those frequencies. The reason is due to the high heat generated by the R820T2 chip. The V3 uses a thin thermal pad to thermally bond the PCB and metal enclosure together. This allows the metal case to work as a heat sink, which solves the PLL lock problem. Ideally the thermal pad should be as thin as possible to enhance maximum heat transfer, and we have designed the enclosure so that the thermal pad only needs to be 3mm thick.

The V3 also uses a larger ground plane on the middle layers of the PCB which also helps with heat dissipation.

R820T2 Chip

Older RTL-SDR units used the R820T chip. There is a newer R820T2 which has slightly better manufacturing tolerances. The R820T2 is produced in a factory with higher quality silicon which allows for more reliable chips. A side effect of the better silicon is overall slightly better and more stable sensitivity across manufacturing runs compared to the R820T, and less PLL lock problems at L-band frequencies.

Improved ESD protection on the RF front end

The BAV99 diode which is used on most RTL-SDR dongles is not a true ESD rated diode. We have added a real ESD rated diode for better protection. The BAV99 remains in the circuit as it works a strong signal clipper, which prevents damage to the R820T2 from overly strong signals. Please remember that not even this will save your radio from a lightning strike or huge ESD impulse, and any permanently outdoor mounted antenna system must have its own lightning and ESD protection. To help avoid lightning damage unplug your antenna during a storm and when the dongle is not in use.

Improved front end circuit

The standard matching circuit on the RTL-SDR was designed for DVB-T use, and tends to attenuate signals above ~1 GHz. The new matching circuit has less attenuation above 1 GHz and similar performance below. We have used high quality, high SRF, high Q inductors in this circuit.

Software switchable 4.5v bias tee.

The V3 makes use of a low noise LDO and one of the GPIO pins on the RTL2832U to provide a 4.5V bias tee that can be activated in software. The bias tee can pull about 180 mA continuously so is suitable for the majority of 3-5V powered LNAs that are popular with RTL-SDR devices. The bias tee is protected against accidental short circuits at the LDO level, and with a thermal auto-resetting PTC fuse. See 'Activating the Bias Tee' for more information on software for activating the bias tee.

This bias tee is great for powering a remote LNA (like Adams PSA5043+ based LNA4ALL) or something like the SpyVerter upconverter.

Bias Tee Warning: The bias tee thermal fuse or LDO could be damaged if you short circuit the bias tee for long periods of time. Before turning on the bias tee, ensure the circuit to be powered is not shorted, or that the RTL-SDR is not connected to a DC shorted antenna!

Lower Voltage Operation

The V3 uses an LDO that has a much lower 'dropout' voltage compared to the typical AMS1117 LDO used on most dongles. Hence the V3 should run better on long USB extension cables.

Long USB cables tend to drop the 5V USB voltage down to lower levels. Below about 4V the AMS1117 stops working. The LDO used in the V3 works almost down to 3.3V.

Of course, with low voltages from long USB cable, the bias tee will be unable to put out 4.5V. At low voltages the bias tee LDO will revert to a non-filtered voltage slightly under the supply.

Reduced noise with a modified PCB design

Typical RTL-SDR dongles use 2-layer PCB designs and route signal lines improperly. The V3 uses a modified 4-layer PCB design which helps to significantly reduces clock spurs and noise pickup.

The V3 also adds a USB common mode choke on the USB data lines to reduce USB noise, adds SMD ferrite chokes on the PCB power lines, and uses a lower noise LDO.

HF direct sampling circuit, diplexed out from the SMA connector

The idea behind direct sampling mode is that an antenna can be connected directly to the ADC pins of the RTL2832U, and this can enable HF reception. This is useful because the R820T/2 tuner can only tune down to about 24 MHz at the lowest. On typical R820T RTL-SDR dongles one can enable direct sampling mode by soldering a wire to the Q-branch pins of the RTL2832U. The RTL2832U samples at 28.8 MHz, so 0 – 14.4 MHz, and 14.4 MHz – 28.8 MHz can be listened to.

The V3 has direct sampling mode implemented in hardware already, so no hardware mods are required to listen to HF via direct sampling.

To split the HF signal out at the SMA connector, a diplexer tuned to 25 MHz is used. A 10dB buffer preamp sits after the diplexer which helps to boost the signal and overcome losses in the subsequent filter and impedance transformer. After the preamp is a 24 MHz low pass filter and then an impedance matching and single to double ended transformer. The addition of the preamp, filter and transformer ensures good direct sampling performance.

The result is that 500 kHz to about 24 MHz can be received in direct sampling mode.

Direct sampling could be more sensitive than using an upconverter, but dynamic won't be as good as with an upconverter. It can overload easily if you have strong signals since there is no gain control. And you will see aliasing of signals mirrored around 14.4 MHz due to the Nyquist theorm. But direct sampling mode should at least give the majority of users a decent taste of what's on HF. If you then find HF interesting, then you can consider upgrading to an upconverter like the SpyVerter (the SpyVerter is the only upconverter we know of that is compatible with our bias tee for easy operation, other upconverters require external power).

If you search on YouTube for "RTL-SDR V3", you will find several videos showing what you can get in direct sampling mode. Most people are surprised at how good it can be, but also many users will need a broadcast AM filter to reduce overloading. We sell a suitable broadcast AM filter on our store www.rtl-sdr.com/store.

Expansion pads on the PCB

Access pads for the unused GPIO pins, CLK in/out, 3.3V, GND and I2C pins have been added. The CLK input/output is disconnected by default. Access pads for the I branch have also been added as some users and industrial customers are using these in special projects. These pads are only for advanced users who need them for special projects. Take care as these pins are not ESD protected.

Clock selector jumper

By soldering in a 4 pin 1.27mm pitch jumper header and removing the default 0 Ohm resistor, one can now easily select between the onboard clock, an external clock, or having the on board clock be the output for another dongle. This is for advanced users only who want to experiment with things like passive radar, and coherent receivers.

Corner mounting holes for those who want to stack PCBs.

Some customers have been building devices that require multiple RTL-SDR dongles, and these standoff holes should aid in stacking.

Feature Information

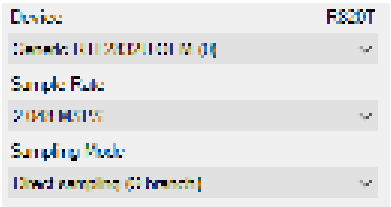
Feature 1: Direct Sampling HF Mode

This feature allows you to listen to HF signals between about 500 kHz to 28.8 MHz.

To use direct sampling mode first connect an appropriate HF antenna to the SMA antenna port (this is the same port where you connect your VHF/UHF antenna).

In SDR# select the Q-branch in the configure menu (the cog icon next to the play button). (If it is greyed out make sure you stop the SDR first, by clicking the stop button in SDR#)

Press Play and tune to 500 kHz – 28.8 MHz.



VHF antennas like small discones or short whip antennas will probably not pick up HF signals very well, if at all. If you have no such antenna you *might* get something with the large telescopic antenna extended to its maximum length of 1.5m, but really this is still not long enough for HF. You can instead use the screw nut provided with the antenna base to clamp on a long wire antenna that is 5 meters or more in length. Ideally you should use a 9:1 unun with the long wire antenna for optimal reception but it is not totally necessary. Even more ideally you'd use an antenna tuner, though this is expensive.

Other software like HDSDR and GQRX can also support direct sampling. It may entail setting a device string, and for the Q-branch, the value should be 2. In GQRX the device string would be "rtl=0,direct_samp=2" (without the quotes). Make sure that there is no space after the comma.

To go back to listening to frequencies above 28.8 MHz remember to change the sampling mode back to "Quadrature Sampling".

Note that this feature makes use of *direct sampling* and so aliasing will occur. The RTL-SDR samples at 28.8 MHz, thus you may see mirrors of strong signals from 0 – 14.4 MHz while tuning to 14.4 – 28.8 MHz and the other way around as well. If these images cause problems, then to remove them you will need to use a low pass filter for 0 – 14.4 MHz, and a high pass filter for 14.4 – 28.8 MHz. Either that or you can simply filter your exact band of interest.

Feature 2: Software Selectable Bias Tee

The V3 RTL-SDR introduces a bias tee which can be enabled easily in software.

WARNING: Before using the bias tee please ensure that you understand that you should not use this option when the dongle is connected *directly* to a DC short circuited antenna. Although the bias tee circuit is dual protected against accidental shorts with a PTC automatically resetting fuse and overcurrent protection on the LDO, short circuiting the bias tee for an extended period (hours) could damage the LDO or fuse permanently. Only use it while connected to an actual powered device, like an LNA, active antenna or the SpyVerter.

To make things clearer: DC Short Antenna -> LNA -> Coax -> V3(bias tee on) is fine. What's not good and makes no sense anyway is DC Short Antenna -> Coax -> V3(bias tee on). DC Short Antenna -> Coax -> V3(bias tee off) is fine.

To enable the bias tee in Windows:

1. Download and extract all the files in the zip file downloadable at https://github.com/rtlsdrblog/rtl-sdr/releases/tag/v1.1 into a folder on your PC. It contains two batch files that can be run.

2. Next make sure that all SDR software like SDR#/HDSDR/SDR-Console etc is fully closed. If there is another program accessing the RTL-SDR the bias tee software will not run.
3. Run the biastee_on.bat file to turn the bias tee on. It will run and open a CMD prompt that will briefly say "Found Rafael Micro R820T Tuner". The CMD prompt will close soon after upon success.

The bias tee is now on. To turn it off repeat steps 2 & 3, but instead run the biastee_off.bat batch file. Alternatively, simply disconnect and then reconnect the SDR to turn the bias tee off.

If you have multiple dongles connected you'll need to edit the batch file to specify what dongle's bias tee you want to activate. Open the bat file with any text editor, like Notepad, and add the dongle selector "-d" flag. For example, to activate the bias tee on the dongle that was plugged in second you'd need to change it to "rtl_biast -b 1 -d 1".

If you get a Smart Screen message, click on More Info, and then on Run Anyway. Also note that some versions of Windows may fail to run batch files due to misconfiguration or aggressive antivirus software. If you cannot fix these problems with Windows or your antivirus, run the command manually on the CMD line.

To run it manually on the CMD line first browse to the directory where the bias tee software is stored using "cd" (e.g. cd C:\SDR\bias_tee_folder), and then run:

ON: rtl_biast -b 1

OFF: rtl_biast -b 0

If needed select a particular RTL-SDR device with the -d flag.

In Linux or MacOS download the source from git, compile it the same way you do the regular RTL-SDR drivers, and then run ./rtl_biast -b 1 to turn the bias tee on and ./rtl_biast -b 0 to turn the bias tee off. The procedure is:

```
git clone https://github.com/rtlsdrblog/rtl_biast
cd rtl_biast
mkdir build
cd build
cmake ..
make
cd src
./rtl_biast -b 1
```

If you want to be able to run the bias tee program from anywhere on the command line you can also run "sudo make install".

If you have trouble running the bias tee use a multimeter to check if there is 4.5V at the SMA port, and that your powered device is actually capable of receiving power. Remember that not all LNA's can accept bias tee power. We recommend Adam 9A4QV's LNA4ALL, as you can order this from his store with the bias tee power option enabled.

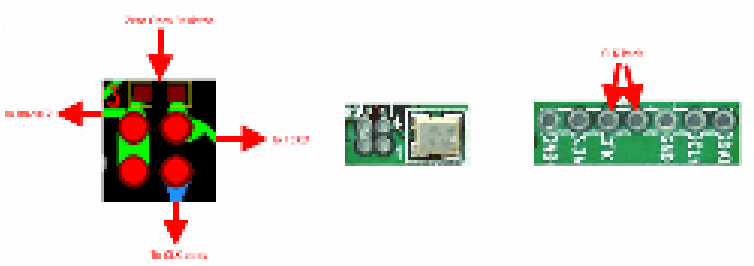
Feature 3: Selectable Clock & Expansion Headers

This is for advanced users who need to daisy chain clocks together for coherent experiments, or need to access other ports. You can either bridge the clock selector the directly with a solder bridge, or solder on a 1.27mm 2×2 header pin jumper.

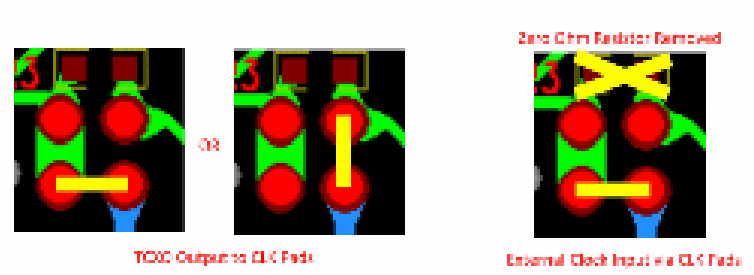
1. To add a jumper to the CLK selector header.
2. Carefully remove the 0 Ohm resistor.
3. Very carefully solder a 1.27mm 2×2 header onto the clock selector pads.

You can now select your clock input.

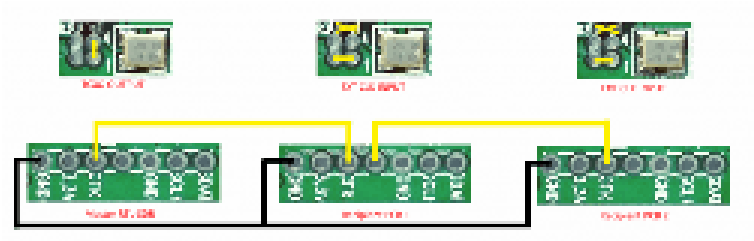
How to connect the CLK jumpers:



The first position allows you to output the dongles clock to the CLK pads. The second position allows you to input an external clock.



An example of CLK daisy chaining is shown below. One dongles TCXO is connected to two other dongles who have disconnected clocks.



LF Improvement / Bias Tee Disable Mod

If you want to improve the performance at LF/MW and do not require the bias tee, then you can remove the bias tee inductor at L13. Of course, remember that if you are really interested in VLF/LF, then it might be a better idea to use a VLF/LF compatible upconverter like the SpyVerter, which can be powered by the bias tee on the dongle. Obviously if you remove the bias tee inductor, the bias tee will no longer function, and so you'd have to power the SpyVerter externally via a USB cable.

7.4. R820T Datasheet

R820T
High Performance Low Power Advanced Digital TV Silicon Tuner Datasheet

Suite 808, Building 53, No.195, Sec.4, Chung Hsing Road, Chutung, HsinChu 310, Taiwan , R.O.C.
310 台灣新竹縣竹東鎮中興路四段 195 號 53 館 808 室
Tel: 886-3-5820868 Fax: 886-3-5820968 www.rafaelmicro.com

7.5. RTL2832U Datasheet

RTL2832U

DVB-T COFDM DEMODULATOR+USB 2.0

DATASHEET

(CONFIDENTIAL: Development Partners Only)

Rev. 1.4
01 November 2010
Track ID: JATR-2265-11

Realtek Semiconductor Corp.
No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan
Tel.: +886-3-578-0211. Fax: +886-3-577-6047
www.realtek.com

7.6. BME-280 Datasheet

BOSCH

BME280

Combined humidity and pressure sensor

BME280 – Data sheet

Document revision	1.6
Document release date	September 2018
Document number	BST-BME280-DS002-15
Technical reference code	0 273 141 185
Notes	Data and descriptions in this document are subject to change without notice. Product photos and pictures are for illustration purposes only and may differ from the real product appearance

BME280
Digital humidity, pressure and temperature sensor

Key features

- Package — 2.5 mm x 2.5 mm x 0.93 mm metal lid LGA
- Digital interface — I²C (up to 3.4 MHz) and SPI (3 and 4 wire, up to 10 MHz)
- Supply voltage — V_{DD} main supply voltage range: 1.71 V to 3.6 V; V_{DDIO} interface voltage range: 1.2 V to 3.6 V
- Current consumption — 1.8 µA @ 1 Hz humidity and temperature; 2.8 µA @ 1 Hz pressure and temperature; 3.6 µA @ 1 Hz humidity, pressure and temperature; 0.1 µA in sleep mode
- Operating range — -40…+85 °C, 0…100 % rel. humidity, 300…1100 hPa
- Humidity sensor and pressure sensor can be independently enabled / disabled
- Register and performance compatible to Bosch Sensortec BMP280 digital pressure sensor
- RoHS compliant, halogen-free, MSL1

Key parameters for humidity sensor

- Response time ($\tau_{63\%}$) — 1 s
- Accuracy tolerance — ±3 % relative humidity
- Hysteresis — ±1% relative humidity

Key parameters for pressure sensor

- RMS Noise — 0.2 Pa, equiv. to 1.7 cm
- Offset temperature coefficient — ±1.5 Pa/K, equiv. to ±12.6 cm at 1 °C temperature change

Typical application

- Context awareness, e.g. skin detection, room change detection
- Fitness monitoring / well-being
  - Warning regarding dryness or high temperatures
  - Measurement of volume and air flow
- Home automation control
  - control heating, venting, air conditioning (HVAC)
- Internet of things
- GPS enhancement (e.g. time-to-first-fix improvement, dead reckoning, slope detection)
- Indoor navigation (change of floor detection, elevator detection)
- Outdoor navigation, leisure and sports applications
- Weather forecast
- Vertical velocity indication (rise/sink speed)

Target devices

- Handsets such as mobile phones, tablet PCs, GPS devices
- Navigation systems
- Gaming, e.g flying toys
- Camera (DSC, video)
- Home weather stations
- Flying toys
- Watches

General Description

The BME280 is as combined digital humidity, pressure and temperature sensor based on proven sensing principles. The sensor module is housed in an extremely compact metal-lid LGA package with a footprint of only 2.5 × 2.5 mm² with a height of 0.93 mm. Its small dimensions and its low power consumption allow the implementation in battery driven devices such as handsets, GPS modules or watches. The BME280 is register and performance compatible to the Bosch Sensortec BMP280 digital pressure sensor (see chapter 5.2 for details).

The BME280 achieves high performance in all applications requiring humidity and pressure measurement. These emerging applications of home automation control, in-door navigation, fitness as well as GPS refinement require a high accuracy and a low TCO at the same time.

The humidity sensor provides an extremely fast response time for fast context awareness applications and high overall accuracy over a wide temperature range.

The pressure sensor is an absolute barometric pressure sensor with extremely high accuracy and resolution and drastically lower noise than the Bosch Sensortec BMP180.

The integrated temperature sensor has been optimized for lowest noise and highest resolution. Its output is used for temperature compensation of the pressure and humidity sensors and can also be used for estimation of the ambient temperature.

The sensor provides both SPI and I²C interfaces and can be supplied using 1.71 to 3.6 V for the sensor supply V_{DD} and 1.2 to 3.6 V for the interface supply V_{DDIO}. Measurements can be triggered by the host or performed in regular intervals. When the sensor is disabled, current consumption drops to 0.1 µA.

BME280 can be operated in three power modes (see chapter 3.3):

- sleep mode
- normal mode
- forced mode

In order to tailor data rate, noise, response time and current consumption to the needs of the user, a variety of oversampling modes, filter modes and data rates can be selected.

Please contact your regional Bosch Sensortec partner for more information about software packages.

Index of Contents

1. Specification

If not stated otherwise,

- All values are valid over the full voltage range
- All minimum/maximum values are given for the full accuracy temperature range
- Minimum/maximum values of drifts, offsets and temperature coefficients are ±3σ values over lifetime
- Typical values of currents and state machine timings are determined at 25 °C
- Minimum/maximum values of currents are determined using corner lots over complete temperature range
- Minimum/maximum values of state machine timings are determined using corner lots over 0...+65 °C temperature range

The specification tables are split into humidity, pressure, and temperature part of BME280.

1.1 General electrical specification

Table 1: Electrical parameter specification

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Supply Voltage Internal Domains	V_{DD}	ripple max. 50 mVpp	1.71	1.8	3.6	V
Supply Voltage I/O Domain	V_{DDIO}		1.2	1.8	3.6	V
Sleep current	I_{DDSL}			0.1	0.3	µA
Standby current (inactive period of normal mode)	I_{DDSB}			0.2	0.5	µA
Current during humidity measurement	I_{DDH}	Max value at 85 °C		340		µA
Current during pressure measurement	I_{DDP}	Max value at -40 °C		714		µA
Current during temperature measurement	I_{DDT}	Max value at 85 °C		350		µA
Start-up time	$t_{startup}$	Time to first communication after both V_{DD} > 1.58 V and V_{DDIO} > 0.65 V			2	ms
Power supply rejection ratio (DC)	PSRR	full V_{DD} range			±0.01 ±5	%RH/V Pa/V
Standby time accuracy	$\Delta t_{standby}$			±5	±25	%

1.2 Humidity parameter specification

Table 2: Humidity parameter specification

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Operating range[1]	R_H	For temperatures < 0 °C and > 60 °C see Figure 1	-40	25	85	°C
			0		100	%RH
Supply current	$I_{DD,H}$	1 Hz forced mode, humidity and temperature		1.8	2.8	µA
Absolute accuracy tolerance	A_H	20...80 %RH, 25 °C, including hysteresis		±3		%RH
Hysteresis[2]	H_H	10→90→10 %RH, 25 °C		±1		%RH
Nonlinearity[3]	NL_H	10→90 %RH, 25 °C		1		%RH
Response time to complete 63% of step[4]	$\tau_{63\%}$	90→0 or 0→90 %RH, 25°C		1		s
Resolution	R_H			0.008		%RH
Noise in humidity (RMS)	N_H	Highest oversampling, see chapter 3.6		0.02		%RH
Long term stability	ΔH_{stab}	10...90 %RH, 25 °C		0.5		%RH/ year

[1] When exceeding the operating range (e.g. for soldering), humidity sensing performance is temporarily degraded and reconditioning is recommended as described in section 7.8. Operating range only for non-condensing environment.
[2] For hysteresis measurement the sequence 10→30→50→70→90→70→50→30→10 %RH is used. The hysteresis is defined as the difference between measurements of the humidity up / down branch and the averaged curve of both branches
[3] Non-linear contributions to the sensor data are corrected during the calculation of the relative humidity by the compensation formulas described in section 4.2.3.
[4] The air-flow in direction to the vent-hole of the device has to be dimensioned in a way that a sufficient air exchange inside to outside will be possible. To observe effects on the response time-scale of the device an air-flow velocity of approx. 1 m/s is needed.

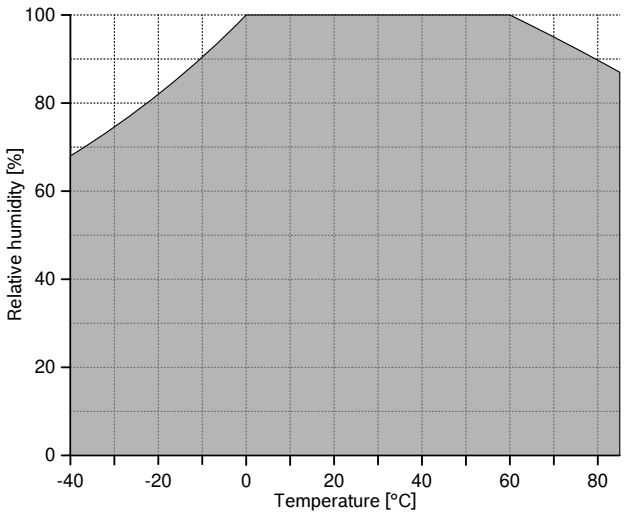


Figure 1: humidity sensor operating range

1.3 Pressure sensor specification

Table 3: Pressure parameter specification

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Operating temperature range	T_A	operational	-40	25	+85	°C
		full accuracy	0		+65	
Operating pressure range	P	full accuracy	300		1100	hPa
Supply current	$I_{DD,LP}$	1 Hz forced mode, pressure and temperature, lowest power		2.8	4.2	µA

Temperature coefficient of offset[5]	TCO_P	25...65 °C, 900 hPa		±1.5		Pa/K
				±12.6		cm/K
Absolute accuracy pressure	A^P_{ext}	300. . 1100 hPa -20 . . . 0 °C		±1.7		hPa
	$A_{P,full}$	300 . . . 1100 hPa 0 . . . 65 °C		±1.0		hPa
	A^P	1100 . . . 1250 hPa 25 . . . 40 °C		±1.5		hPa
Relative accuracy pressure V_{DD} = 3.3V	A_{rel}	700 ... 900hPa 25 . . . 40 °C		±0.12		hPa
Resolution of pressure output data	R_P	Highest oversampling		0.18		Pa
Noise in pressure	$N_{P,fullBW}$	Full bandwidth, highest oversampling See chapter 3.6		1.3		Pa
				11		cm
	$N_{P,filtered}$	Reduced bandwidth, highest oversampling See chapter 3.6		0.2		Pa
				1.7		cm
Solder drift		Minimum solder height 50µm	-0.5		+2.0	hPa
Long term stability[6]	ΔP_{stab}	per year		±1.0		hPa
Possible sampling rate	f_{sample_P}	Lowest oversampling, see chapter 9.2	157	182		Hz

1.4 Temperature sensor specification

Table 4: Temperature parameter specification

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Operating range	T	Operational	-40	25	85	°C
		Full accuracy	0		65	°C
Supply current	$I_{DD,T}$	1 Hz forced mode, temperature measurement only		1.0		µA
	$A_{T,25}$	25 °C		±0.5		°C

[5] When changing temperature by e.g. 10 °C at constant pressure / altitude, the measured pressure / altitude will change by (10 × TCO_P).
[6] Long term stability is specified in the full accuracy operating pressure range 0 ... 65 °C

Absolute accuracy temperature[7]	$A_{T,full}$	0...65 °C		±1.0		°C
	$A_{T,ext}$[8]	-20 .... 0 °C		±1.25		°C
	$A_{T,ext}$[9]	-40 ... -20 °C		±1.5		°C
Output resolution	R_T	API output resolution		0.01		°C
RMS noise	N_T	Lowest oversampling		0.005		°C

[7] Temperature measured by the internal temperature sensor. This temperature value depends on the PCB temperature, sensor element self-heating and ambient temperature and is typically above ambient temperature.
[8] Target values & not guaranteed
[9] Target values & not guaranteed

2. Absolute maximum ratings

The absolute maximum ratings are determined over complete temperature range using corner lots. The values are provided in Table 5.

Table 5: Absolute maximum ratings

Parameter	Condition	Min	Max	Unit
Voltage at any supply pin	V_{DD} and V_{DDIO} pin	-0.3	4.25	V
Voltage at any interface pin		-0.3	V_{DDIO} + 0.3	V
Storage temperature	≤ 65% RH	-45	+85	°C
Pressure		0	20 000	hPa
ESD	HBM, at any pin		±2	kV
	CDM		±500	V
	Machine model		±200	V
Condensation	No power supplied	Allowed		

3. Functional description

3.1 Block diagram

Figure 2 shows a simplified block diagram of the BME280:

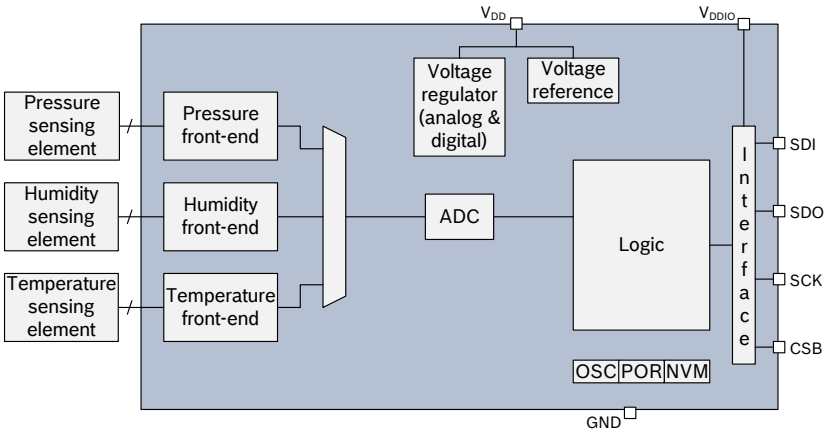


Figure 2: Block diagram of BME280

3.2 Power management

The BME280 has two distinct power supply pins

- V_{DD} is the main power supply for all internal analog and digital functional blocks
- V_{DDIO} is a separate power supply pin used for the supply of the digital interface

A power-on reset (POR) generator is built in; it resets the logic part and the register values after both V_{DD} and V_{DDIO} reach their minimum levels. There are no limitations on slope and sequence of raising the V_{DD} and V_{DDIO} levels. After powering up, the sensor settles in sleep mode (described in chapter 3.3.2).

It is prohibited to keep any interface pin (SDI, SDO, SCK or CSB) at a logical high level when V_{DDIO} is switched off. Such a configuration can permanently damage the device due an excessive current flow through the ESD protection diodes.

If V_{DDIO} is supplied, but V_{DD} is not, the interface pins are kept at a high-Z level. The bus can therefore already be used freely before the BME280 V_{DD} supply is established.

Resetting the sensor is possible by cycling V_{DD} level or by writing a soft reset command. Cycling the V_{DDIO} level will not cause a reset.

3.3 Sensor modes

The BME280 offers three sensor modes: sleep mode, forced mode and normal mode. These can be selected using the *mode*[1:0] setting (see chapter 5.4.5). The available modes are:

- Sleep mode: no operation, all registers accessible, lowest power, selected after startup
- Forced mode: perform one measurement, store results and return to sleep mode
- Normal mode: perpetual cycling of measurements and inactive periods.

The modes will be explained in detail in chapters 3.3.2 (sleep mode), 3.3.3 (forced mode) and 3.3.4 (normal mode).

3.3.1 Sensor mode transitions

The supported mode transitions are shown in Figure 3. If the device is currently performing a measurement, execution of mode switching commands is delayed until the end of the currently running measurement period. Further mode change commands or other write commands to the register *ctrl_hum* are ignored until the mode change command has been executed. Mode transitions other than the ones shown below are tested for stability but do not represent recommended use of the device.

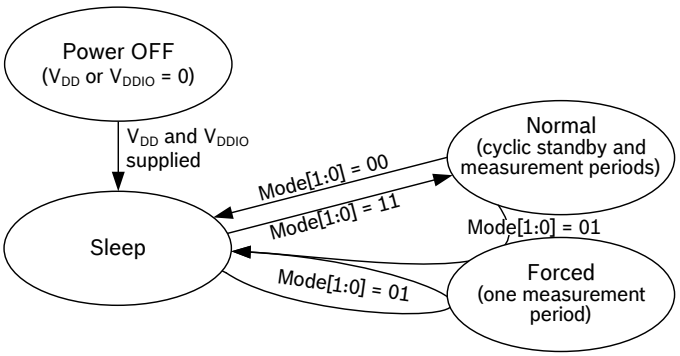


Figure 3: Sensor mode transition diagram

3.3.2 Sleep mode

Sleep mode is entered by default after power on reset. In sleep mode, no measurements are performed and power consumption (I_{DDSM}) is at a minimum. All registers are accessible; Chip-ID and compensation coefficients can be read. There are no special restrictions on interface timings.

3.3.3 Forced mode

In forced mode, a single measurement is performed in accordance to the selected measurement and filter options. When the measurement is finished, the sensor returns to sleep mode and the measurement results can be obtained from the data registers. For a next measurement, forced mode needs to be selected again. This is similar to BMP180 operation. Using forced mode is recommended for applications which require low sampling rate or host-based synchronization. The timing diagram is shown below.

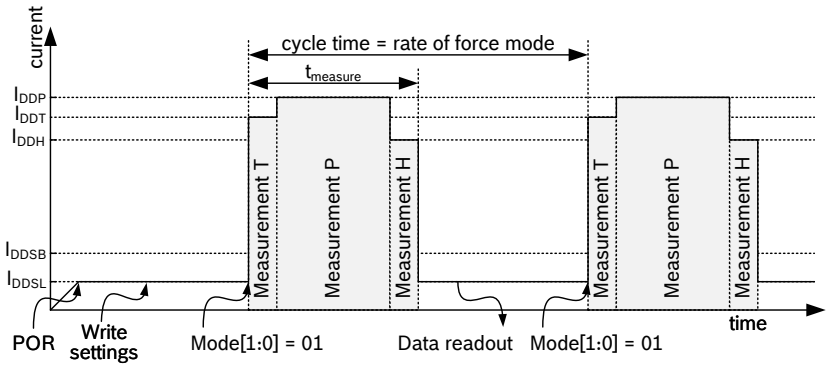


Figure 4: Forced mode timing diagram

3.3.4 Normal mode

Normal mode comprises an automated perpetual cycling between an (active) measurement period and an (inactive) standby period.

The measurements are performed in accordance to the selected measurement and filter options. The standby time is determined by the setting *t_sb*[2:0] and can be set to between 0.5 and 1000 ms according to Table 27.

The total cycle time depends on the sum of the active time (see chapter 9) and standby time $t_{standby}$. The current in the standby period (I_{DDSB}) is slightly higher than in sleep mode. After setting the measurement and filter options and enabling normal mode, the last measurement results can always be obtained at the data registers without the need of further write accesses.

Using normal mode is recommended when using the IIR filter. This is useful for applications in which short-term disturbances (e.g. blowing into the sensor) should be filtered. The timing diagram is shown below:

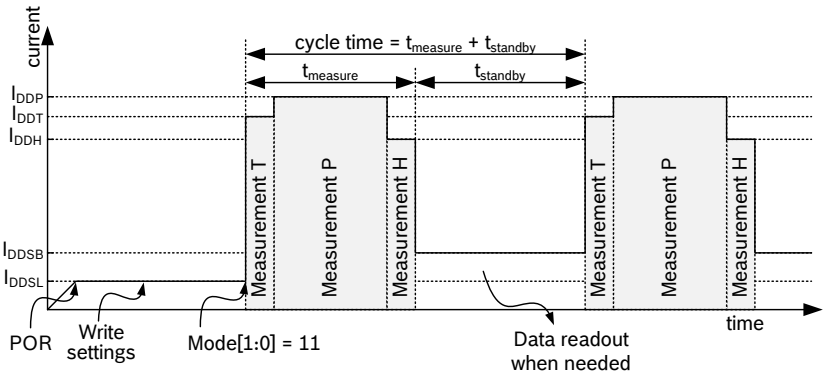


Figure 5: Normal mode timing diagram

3.4 Measurement flow

The BME280 measurement period consists of a temperature, pressure and humidity measurement with selectable oversampling. After the measurement period, the pressure and temperature data can be passed through an optional IIR filter, which removes short-term fluctuations in pressure (e.g. caused by slamming a door). For humidity, such a filter is not needed and has not been implemented. The flow is depicted in the diagram below.

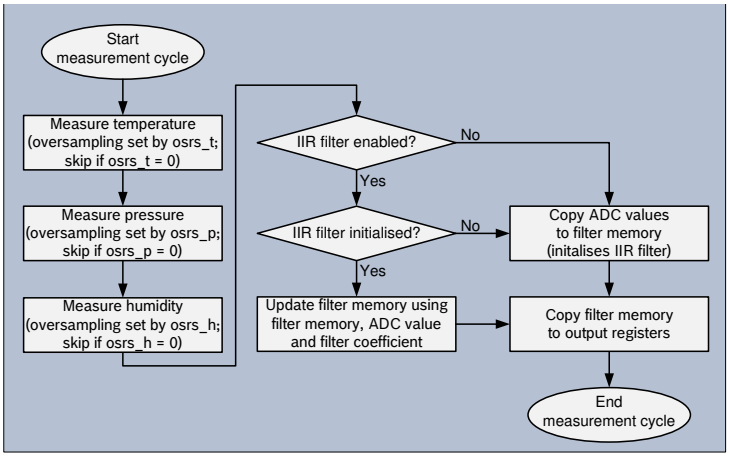


Figure 6: BME280 measurement cycle

The individual blocks of the diagram above will be detailed in the following subchapters.

3.4.1 Humidity measurement

The humidity measurement can be enabled or skipped. When enabled, several oversampling options exist. The humidity measurement is controlled by the *osrs_h*[2:0] setting, which is detailed in chapter 5.4.3. For the humidity measurement, oversampling is possible to reduce the noise. The resolution of the humidity measurement is fixed at 16 bit ADC output.

3.4.2 Pressure measurement

Pressure measurement can be enabled or skipped. When enabled, several oversampling options exist. The pressure measurement is controlled by the *osrs_p*[2:0] setting which is detailed in chapter 5.4.5. For the pressure measurement, oversampling is possible to reduce the noise. The resolution of the pressure data depends on the IIR filter (see chapter 3.4.4) and the oversampling setting (see chapter 5.4.5):

- When the IIR filter is enabled, the pressure resolution is 20 bit.
- When the IIR filter is disabled, the pressure resolution is 16 + (*osrs_p* – 1) bit, e.g. 18 bit when *osrs_p* is set to '3'.

3.4.3 Temperature measurement

Temperature measurement can be enabled or skipped. Skipping the measurement could be useful to measure pressure extremely rapidly. When enabled, several oversampling options exist. The temperature measurement is controlled by the *osrs_t*[2:0] setting which is detailed in chapter 5.4.5. For the temperature measurement, oversampling is possible to reduce the noise.
The resolution of the temperature data depends on the IIR filter (see chapter 3.4.4) and the oversampling setting (see chapter 5.4.5):

- When the IIR filter is enabled, the temperature resolution is 20 bit.

- When the IIR filter is disabled, the temperature resolution is 16 + (*osrs_t* – 1) bit, e.g. 18 bit when *osrs_t* is set to '3'.

3.4.4 IIR filter

The humidity value inside the sensor does not fluctuate rapidly and does not require low pass filtering. However, the environmental pressure is subject to many short-term changes, caused e.g. by slamming of a door or window, or wind blowing into the sensor. To suppress these disturbances in the output data without causing additional interface traffic and processor work load, the BME280 features an internal IIR filter. It effectively reduces the bandwidth of the temperature and pressure output signals[10] and increases the resolution of the pressure and temperature output data to 20 bit. The output of a next measurement step is filtered using the following formula:

$$data\_filtered = \frac{data\_filtered\_old \cdot (filter\_coefficient - 1) + data\_ADC}{filter\_coefficient}$$

Data_filtered_old is the data coming from the current filter memory, and data_ADC is the data coming from current ADC acquisition. Data_filtered is the new value of filter memory and the value that will be sent to the output registers.

The IIR filter can be configured to different filter coefficients, which slows down the response to the sensor inputs. Note that the response time with enabled IIR filter depends on the number of samples generated, which means that the data output rate must be known to calculate the actual response time. For register configuration, please refer to Table 28. A sample response time calculation is shown in chapter 9.4.

Table 6: *filter* settings

Filter coefficient	Samples to reach ≥75 % of step response
Filter off	1
2	2
4	5
8	11
16	22

In order to find a suitable setting for *filter*, please consult chapter 3.5.

When writing to the register *filter*, the filter is reset. The next ADC values will the pass through the filter unchanged and become the initial memory values for the filter. If temperature or pressure measurements are skipped, the corresponding filter memory will be kept unchanged even though the output registers are set to 0x80000. When the previously skipped measurement is re-enabled, the output will be filtered using the filter memory from the last time when the measurement was not skipped. If this is not desired, please write to the *filter* register in order to re-initialize the filter.

[10] Since the BME280 does not sample continuously, filtering can suffer from signals with a frequency higher than the sampling rate of the sensor. E.g. environmental fluctuations caused by windows being opened and closed might have a frequency <5 Hz. Consequently, a sampling rate of ODR = 10 Hz is sufficient to obey the Nyquist theorem.

The step response (e.g. response to in sudden change in height) of the different filter settings is displayed in Figure 7.

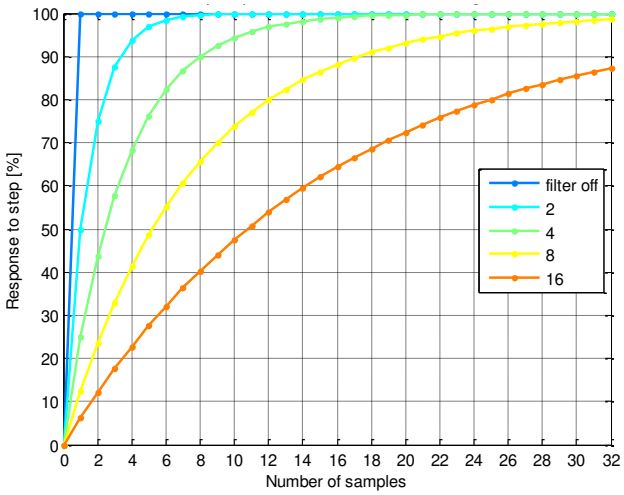


Figure 7: Step response at different IIR filter settings

3.5 Recommended modes of operation

The different oversampling options, filter settings and sensor modes result in a large number of possible settings. In this chapter, a number of settings recommended for various scenarios are presented.

3.5.1 Weather monitoring

Description: Only a very low data rate is needed. Power consumption is minimal. Noise of pressure values is of no concern. Humidity, pressure and temperature are monitored.

Table 7: Settings and performance for weather monitoring

Suggested settings for weather monitoring	
Sensor mode	forced mode, 1 sample / minute
Oversampling settings	pressure ×1, temperature ×1, humidity ×1
IIR filter settings	filter off
Performance for suggested settings	
Current consumption	0.16 µA
RMS Noise	3.3 Pa / 30 cm, 0.07 %RH
Data output rate	1/60 Hz

3.5.2 Humidity sensing

Description: A low data rate is needed. Power consumption is minimal. Forced mode is used to minimize power consumption and to synchronize readout, but using normal mode would also be possible.

Table 8: Settings and performance for humidity sensing

Suggested settings for weather monitoring	
Sensor mode	forced mode, 1 sample / second
Oversampling settings	pressure ×0, temperature ×1, humidity ×1
IIR filter settings	filter off
Performance for suggested settings	
Current consumption	2.9 µA
RMS Noise	0.07 %RH
Data output rate	1 Hz

3.5.3 Indoor navigation

Lowest possible altitude noise is needed. A very low bandwidth is preferred. Increased power consumption is tolerated. Humidity is measured to help detect room changes. This setting is suggested for the Android settings 'SENSOR_DELAY_NORMAL' and 'SENSOR_DELAY_UI'.

Table 9: Settings and performance for indoor navigation

Suggested settings for indoor navigation	
Sensor mode	normal mode, $t_{standby}$ = 0.5 ms
Oversampling settings	pressure ×16, temperature ×2, humidity ×1
IIR filter settings	filter coefficient 16
Performance for suggested settings	
Current consumption	633 µA
RMS Noise	0.2 Pa / 1.7 cm
Data output rate	25Hz
Filter bandwidth	0.53 Hz
Response time (75%)	0.9 s

3.5.4 Gaming

Low altitude noise is needed. The required bandwidth is ~2 Hz in order to respond quickly to altitude changes (e.g. be able to dodge a flying monster in a game). Increased power consumption is tolerated. Humidity sensor is disabled. This setting is suggested for the Android settings 'SENSOR_DELAY_GAMING' and 'SENSOR_DELAY_FASTEST'.

Table 10: Settings and performance for gaming

Suggested settings for gaming	
Sensor mode	normal mode, $t_{standby}$ = 0.5 ms
Oversampling settings	pressure ×4, temperature ×1, humidity ×0
IIR filter settings	filter coefficient 16
Performance for suggested settings	
Current consumption	581 µA
RMS Noise	0.3 Pa / 2.5 cm
Data output rate	83 Hz
Filter bandwidth	1.75 Hz
Response time (75%)	0.3 s

3.6 Noise

The noise depends on the oversampling and, for pressure and temperature, on the filter setting used. The stated values were determined in a controlled environment and are based on the average standard deviation of 32 consecutive measurement points taken at highest sampling speed. This is needed in order to exclude long term drifts from the noise measurement. The noise depends both on humidity/pressure oversampling and temperature oversampling, since the temperature value is used for humidity/pressure temperature compensation. The oversampling combinations use below results in an optimal power to noise ratio.

Table 11: Noise and current for humidity

Humidity / temperature oversampling setting	Typical RMS noise in humidity [%RH] at 25 °C	Typ. current [µA] at 1 Hz forced mode, 25 °C, humidity and temperature measurement, incl. I_{DDSM}
×1 / ×1	0.07	1.8
×2 / ×1	0.05	2.5
×4 / ×1	0.04	3.8
×8 / ×1	0.03	6.5
×16 / ×1	0.02	11.7

Table 12: Noise and current for pressure

Typical RMS noise in pressure [Pa] at 25 °C						Typ. current [µA] at 1 Hz forced mode, 25 °C, pressure and temperature measurement, incl. I_{DDSM}
Pressure / temperature oversampling setting	**IIR filter coefficient**					
	off	**2**	**4**	**8**	**16**	
×1 / ×1	3.3	1.9	1.2	0.9	0.4	2.8
×2 / ×1	2.6	1.5	1.0	0.6	0.4	4.2
×4 / ×1	2.1	1.2	0.8	0.5	0.3	7.1
×8 / ×1	1.6	1.0	0.6	0.4	0.2	12.8
×16 / ×2	1.3	0.8	0.5	0.4	0.2	24.9

Table 13: Temperature dependence of pressure noise

RMS noise at different temperatures	
Temperature	Typical change in noise compared to 25 °C
-10 °C	+25 %
25 °C	±0 %
75 °C	-5 %

Table 14: Noise in temperature

Temperature oversampling setting	**Typical RMS noise in temperature [°C] at 25 °C**
×1	0.005
×2	0.004
×4	0.003
×8	0.003
×16	0.002

4. Data readout

To read out data after a conversion, it is strongly recommended to use a burst read and not address every register individually. This will prevent a possible mix-up of bytes belonging to different measurements and reduce interface traffic. Note that in I²C mode, even when pressure was not measured, reading the unused registers is faster than reading temperature and humidity data separately.
Data readout is done by starting a burst read from 0xF7 to 0xFC (temperature and pressure) or from 0xF7 to 0xFE (temperature, pressure and humidity). The data are read out in an unsigned 20-bit format both for pressure and for temperature and in an unsigned 16-bit format for humidity. It is strongly recommended to use the BME280 API, available from Bosch Sensortec, for readout and compensation. For details on memory map and interfaces, please consult chapters 5 and 6 respectively.

After the uncompensated values for pressure, temperature and humidity 'ut', 'up' and 'uh' have been read, the actual humidity, pressure and temperature needs to be calculated using the compensation parameters stored in the device. The procedure is elaborated in chapter 4.2.

4.1 Data register shadowing

In normal mode, the timing of measurements is not necessarily synchronized to the readout by the user. This means that new measurement results may become available while the user is reading the results from the previous measurement. In this case, shadowing is performed in order to guarantee data consistency. Shadowing will only work if all data registers are read in a single burst read. Therefore, the user must use burst reads if he does not synchronize data readout with the measurement cycle. Using several independent read commands may result in inconsistent data.
If a new measurement is finished and the data registers are still being read, the new measurement results are transferred into shadow data registers. The content of shadow registers is transferred into data registers as soon as the user ends the burst read, even if not all data registers were read.
The end of the burst read is marked by the rising edge of CSB pin in SPI case or by the recognition of a stop condition in I²C case. After the end of the burst read, all user data registers are updated at once.

4.2 Output compensation

The BME280 output consists of the ADC output values. However, each sensing element behaves differently. Therefore, the actual pressure and temperature must be calculated using a set of calibration parameters. In this chapter, the method to read out the trimming values will be given. The recommended calculation uses fixed point arithmetic and is given in chapter 4.2.3.
In high-level languages like Matlab™ or LabVIEW™, fixed-point code may not be well supported. In this case the floating-point code in appendix 8.1 can be used as an alternative.
For 8-bit micro controllers, the variable size may be limited. In this case a simplified 32 bit integer code with reduced accuracy is given in appendix 8.2.

4.2.1 Computational requirements

In the table below an overview is given for the number of clock cycles needed for compensation on a 32 bit Cortex-M3 micro controller with GCC optimization level -O2. This controller does not feature a floating point unit, thus all floating-point calculations are emulated. Floating point is only recommended for PC application, where an FPU is present and these calculations are performed drastically faster.

Table 15: Computational requirements for compensation formulas

Compensation of	Number of clocks (ARM Cortex-M3)		
	32 bit integer	64 bit integer	Double precision
Humidity	~83	–	~2900 [11]
Temperature	~46	–	~2400 [11]
Pressure	~112 [12]	~1400	~5400 [11]

4.2.2 Trimming parameter readout

The trimming parameters are programmed into the devices' non-volatile memory (NVM) during production and cannot be altered by the customer. Each compensation word is a 16-bit signed or unsigned integer value stored in two's complement. As the memory is organized into 8-bit words, two words must always be combined in order to represent the compensation word. The 8-bit registers are named calib00...calib41 and are stored at memory addresses 0x88...0xA1 and 0xE1...0xE7. The corresponding compensation words are named dig_T# for temperature compensation related values, dig_P# for pressure related values and dig_H# for humidity related values. The mapping is seen in Table 16.

Table 16: Compensation parameter storage, naming and data type

Register Address	Register content	Data type
0x88 / 0x89	dig_T1 [7:0] / [15:8]	unsigned short
0x8A / 0x8B	dig_T2 [7:0] / [15:8]	signed short
0x8C / 0x8D	dig_T3 [7:0] / [15:8]	signed short
0x8E / 0x8F	dig_P1 [7:0] / [15:8]	unsigned short
0x90 / 0x91	dig_P2 [7:0] / [15:8]	signed short
0x92 / 0x93	dig_P3 [7:0] / [15:8]	signed short
0x94 / 0x95	dig_P4 [7:0] / [15:8]	signed short
0x96 / 0x97	dig_P5 [7:0] / [15:8]	signed short
0x98 / 0x99	dig_P6 [7:0] / [15:8]	signed short
0x9A / 0x9B	dig_P7 [7:0] / [15:8]	signed short
0x9C / 0x9D	dig_P8 [7:0] / [15:8]	signed short
0x9E / 0x9F	dig_P9 [7:0] / [15:8]	signed short
0xA1	dig_H1 [7:0]	unsigned char
0xE1 / 0xE2	dig_H2 [7:0] / [15:8]	signed short
0xE3	dig_H3 [7:0]	unsigned char
0xE4 / 0xE5[3:0]	dig_H4 [11:4] / [3:0]	signed short
0xE5[7:4] / 0xE6	dig_H5 [3:0] / [11:4]	signed short

[11] Use only recommended for high-level programming languages like Matlab™ or LabVIEW™
[12] Use only recommended for 8-bit micro controllers

0xE7	dig_H6	signed char

4.2.3 Compensation formulas

Please note that it is strongly advised to use the API available from Bosch Sensortec to perform readout and compensation. If this is not wanted, the code below can be applied at the user's risk. Both pressure and temperature values are expected to be received in 20 bit format, positive, stored in a 32 bit signed integer. Humidity is expected to be received in 16 bit format, positive, stored in a 32 bit signed integer.
The variable t_fine (signed 32 bit) carries a fine resolution temperature value over to the pressure and humidity compensation formula and could be implemented as a global variable.
The data type "BME280_S32_t" should define a 32 bit signed integer variable type and can usually be defined as "long signed int".
The data type "BME280_U32_t" should define a 32 bit unsigned integer variable type and can usually be defined as "long unsigned int".
For best possible calculation accuracy in pressure, 64 bit integer support is needed. If this is not possible on your platform, please see appendix 8.2 for a 32 bit alternative.
The data type "BME280_S64_t" should define a 64 bit signed integer variable type, which on most supporting platforms can be defined as "long long signed int". The revision of the code is rev.1.1.

```
// Returns temperature in DegC, resolution is 0.01 DegC. Output value of "5123" equals 51.23
DegC.
// t_fine carries fine temperature as global value
BME280_S32_t t_fine;
BME280_S32_t BME280_compensate_T_int32(BME280_S32_t adc_T)
{
   BME280_S32_t var1, var2, T;
   var1  = ((((adc_T>>3) - ((BME280_S32_t)dig_T1<<1))) * ((BME280_S32_t)dig_T2)) >> 11;
   var2  = (((((adc_T>>4) - ((BME280_S32_t)dig_T1)) * ((adc_T>>4) - ((BME280_S32_t)dig_T1)))
>> 12) *
      ((BME280_S32_t)dig_T3)) >> 14;
   t_fine = var1 + var2;
   T  = (t_fine * 5 + 128) >> 8;
   return T;
}

// Returns pressure in Pa as unsigned 32 bit integer in Q24.8 format (24 integer bits and 8
fractional bits).
// Output value of "24674867" represents 24674867/256 = 96386.2 Pa = 963.862 hPa
BME280_U32_t BME280_compensate_P_int64(BME280_S32_t adc_P)
{
   BME280_S64_t var1, var2, p;
   var1 = ((BME280_S64_t)t_fine) - 128000;
   var2 = var1 * var1 * (BME280_S64_t)dig_P6;
   var2 = var2 + ((var1*(BME280_S64_t)dig_P5)<<17);
   var2 = var2 + (((BME280_S64_t)dig_P4)<<35);
   var1 = ((var1 * var1 * (BME280_S64_t)dig_P3)>>8) + ((var1 * (BME280_S64_t)dig_P2)<<12);
   var1 = (((((BME280_S64_t)1)<<47)+var1))*((BME280_S64_t)dig_P1)>>33;
   if (var1 == 0)
   {
      return 0; // avoid exception caused by division by zero
   }
   p = 1048576-adc_P;
   p = (((p<<31)-var2)*3125)/var1;
   var1 = (((BME280_S64_t)dig_P9) * (p>>13) * (p>>13)) >> 25;
   var2 = (((BME280_S64_t)dig_P8) * p) >> 19;
   p = ((p + var1 + var2) >> 8) + (((BME280_S64_t)dig_P7)<<4);
   return (BME280_U32_t)p;
}

// Returns humidity in %RH as unsigned 32 bit integer in Q22.10 format (22 integer and 10
fractional bits).
// Output value of "47445" represents 47445/1024 = 46.333 %RH
BME280_U32_t bme280_compensate_H_int32(BME280_S32_t adc_H)
{
   BME280_S32_t v_x1_u32r;

   v_x1_u32r = (t_fine - ((BME280_S32_t)76800));
```

```
    v_x1_u32r = (((((adc_H << 14) - (((BME280_S32_t)dig_H4) << 20) - (((BME280_S32_t)dig_H5) *
       v_x1_u32r)) + ((BME280_S32_t)16384)) >> 15) * (((((((v_x1_u32r *
       ((BME280_S32_t)dig_H6)) >> 10) * (((v_x1_u32r * ((BME280_S32_t)dig_H3)) >> 11) +
       ((BME280_S32_t)32768))) >> 10) + ((BME280_S32_t)2097152)) * ((BME280_S32_t)dig_H2) +
       8192) >> 14));
    v_x1_u32r = (v_x1_u32r - (((((v_x1_u32r >> 15) * (v_x1_u32r >> 15)) >> 7) *
       ((BME280_S32_t)dig_H1)) >> 4));
    v_x1_u32r = (v_x1_u32r < 0 ? 0 : v_x1_u32r);
    v_x1_u32r = (v_x1_u32r > 419430400 ? 419430400 : v_x1_u32r);
    return (BME280_U32_t)(v_x1_u32r>>12);
}
```

5. Global memory map and register description

5.1 General remarks

The entire communication with the device is performed by reading from and writing to registers. Registers have a width of 8 bits. There are several registers which are reserved; they should not be written to and no specific value is guaranteed when they are read. For details on the interface, consult chapter 6.

5.2 Register compatibility to BMP280

The BME280 is downward register compatible to the BMP280, which means that the pressure and temperature control and readout is identical to BMP280. However, the following exceptions have to be considered:

Table 17: Register incompatibilities between BMP280 and BME280

Register	Bits	Content	BMP280	BME280
0xD0 “*id*”	7:0	*chip_id*	Read value is 0x56 / 0x57 (samples) 0x58 (mass production)	Read value is 0x60
0xF5 “*config*”	7:5	*t_sb*	‘110’: 2000 ms ‘111’: 4000 ms	‘110’: 10 ms ‘111’: 20 ms
0xF7...0xF9 “*press*”	19:0	*press*	Resolution (16...20 bit) depends only on *osrs_p*	Without filter, resolution depends on *osrs_p*; when using filter, resolution is always 20 bit
0xFA...0xFC “*temp*”	19:0	*temp*	Resolution (16...20 bit) only depends on *osrs_t*	Without filter, resolution depends on *osrs_t*; when using filter, resolution is always 20 bit

5.3 Memory map

The memory map is given in Table 18 below. Reserved registers are not shown.

Table 18: Memory map

Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
hum_lsb	0xFE	hum_lsb<7:0>								0x00
hum_msb	0xFD	hum_msb<7:0>								0x80
temp_xlsb	0xFC	temp_xlsb<7:4>				0	0	0	0	0x00
temp_lsb	0xFB	temp_lsb<7:0>								0x00
temp_msb	0xFA	temp_msb<7:0>								0x80
press_xlsb	0xF9	press_xlsb<7:4>				0	0	0	0	0x00
press_lsb	0xF8	press_lsb<7:0>								0x00
press_msb	0xF7	press_msb<7:0>								0x80
config	0xF5	t_sb[2:0]			filter[2:0]				spi3w_en[0]	0x00
ctrl_meas	0xF4	osrs_t[2:0]			osrs_p[2:0]			mode[1:0]		0x00
status	0xF3					measuring[0]			im_update[0]	0x00
ctrl_hum	0xF2						osrs_h[2:0]			0x00
calib26..calib41	0xE1…0xF0	calibration data								individual
reset	0xE0	reset[7:0]								0x00
id	0xD0	chip_id[7:0]								0x60
calib00..calib25	0x88…0xA1	calibration data								individual

Registers:	Reserved registers	Calibration data	Control registers	Data registers	Status registers	Chip ID	Reset
Type:	do not change	read only	read / write	read only	read only	read only	write only

5.4 Register description

5.4.1 Register 0xD0 *"id"*

The *"id"* register contains the chip identification number chip_id[7:0], which is 0x60. This number can be read as soon as the device finished the power-on-reset.

5.4.2 Register 0xE0 *"reset"*

The *"reset"* register contains the soft reset word reset[7:0]. If the value 0xB6 is written to the register, the device is reset using the complete power-on-reset procedure. Writing other values than 0xB6 has no effect. The readout value is always 0x00.

5.4.3 Register 0xF2 *"ctrl_hum"*

The *"ctrl_hum"* register sets the humidity data acquisition options of the device. **Changes to this register only become effective after a write operation to "*ctrl_meas*".**

Table 19: Register 0xF2 *"ctrl_hum"*

Register 0xF2 *"ctrl_hum"*	Name	Description
Bit 2, 1, 0	osrs_h[2:0]	Controls oversampling of humidity data. See Table 20 for settings and chapter 3.4.1 for details.

Table 20: register settings *osrs_h*

osrs_h[2:0]	Humidity oversampling
000	Skipped (output set to 0x8000)
001	oversampling ×1
010	oversampling ×2
011	oversampling ×4
100	oversampling ×8
101, others	oversampling ×16

5.4.4 Register 0xF3 *"status"*

The *"status"* register contains two bits which indicate the status of the device.

Table 21: Register 0xF3 *"status"*

Register 0xF3 *"status"*	Name	Description
Bit 3	measuring[0]	Automatically set to '1' whenever a conversion is running and back to '0' when the results have been transferred to the data registers.
Bit 0	im_update[0]	Automatically set to '1' when the NVM data are being copied to image registers and back to '0' when the copying is done. The data are copied at power-on-reset and before every conversion.

5.4.5 Register 0xF4 *"ctrl_meas"*

The *"ctrl_meas"* register sets the pressure and temperature data acquisition options of the device. The register needs to be written after changing *"ctrl_hum"* for the changes to become effective.

Table 22: Register 0xF4 *"ctrl_meas"*

Register 0xF4 *"ctrl_meas"*	Name	Description
Bit 7, 6, 5	osrs_t[2:0]	Controls oversampling of temperature data. See Table 24 for settings and chapter 3.4.3 for details.
Bit 4, 3, 2	osrs_p[2:0]	Controls oversampling of pressure data. See Table 23 for settings and chapter 3.4.2 for details.
Bit 1, 0	mode[1:0]	Controls the sensor mode of the device. See Table 25 for settings and chapter 3.3 for details.

Table 23: register settings *osrs_p*

osrs_p[2:0]	Pressure oversampling
000	Skipped (output set to 0x80000)
001	oversampling ×1
010	oversampling ×2
011	oversampling ×4
100	oversampling ×8
101, others	oversampling ×16

Table 24: register settings *osrs_t*

osrs_t[2:0]	Temperature oversampling
000	Skipped (output set to 0x80000)
001	oversampling ×1
010	oversampling ×2
011	oversampling ×4
100	oversampling ×8
101, others	oversampling ×16

Table 25: register settings *mode*

mode[1:0]	Mode
00	Sleep mode
01 and 10	Forced mode
11	Normal mode

5.4.6 Register 0xF5 *"config"*

The *"config"* register sets the rate, filter and interface options of the device. Writes to the "config" register in normal mode may be ignored. In sleep mode writes are not ignored.

Table 26: Register 0xF5 *"config"*

Register 0xF5 *"config"*	Name	Description
Bit 7, 6, 5	t_sb[2:0]	Controls inactive duration $t_{standby}$ in normal mode. See Table 27 for settings and chapter 3.3.4 for details.
Bit 4, 3, 2	filter[2:0]	Controls the time constant of the IIR filter. See Table 27 for settings and chapter 3.4.4 for details.
Bit 0	spi3w_en[0]	Enables 3-wire SPI interface when set to '1'. See chapter 6.3 for details.

Table 27: *t_sb* settings

t_sb[2:0]	$t_{standby}$ [ms]
000	0.5
001	62.5
010	125
011	250
100	500
101	1000
110	10
111	20

Table 28: *filter* settings

filter[2:0]	Filter coefficient
000	Filter off
001	2
010	4
011	8
100, others	16

5.4.7 Register 0xF7...0xF9 *"press" (_msb, _lsb, _xlsb)*

The *"press"* register contains the raw pressure measurement output data up[19:0]. For details on how to read out the pressure and temperature information from the device, please consult chapter 4.

Table 29: Register 0xF7 ... 0xF9 *"press"*

Register 0xF7...0xF9 *"press"*	Name	Description
0xF7	press_msb[7:0]	Contains the MSB part up[19:12] of the raw pressure measurement output data.
0xF8	press_lsb[7:0]	Contains the LSB part up[11:4] of the raw pressure measurement output data.
0xF9 (bit 7, 6, 5, 4)	press_xlsb[3:0]	Contains the XLSB part up[3:0] of the raw pressure measurement output data. Contents depend on temperature resolution.

5.4.8 Register 0xFA...0xFC *"temp" (_msb, _lsb, _xlsb)*

The *"temp"* register contains the raw temperature measurement output data ut[19:0]. For details on how to read out the pressure and temperature information from the device, please consult chapter 4.

Table 30: Register 0xFA ... 0xFC *"temp"*

Register 0xFA...0xFC *"temp"*	Name	Description
0xFA	temp_msb[7:0]	Contains the MSB part ut[19:12] of the raw temperature measurement output data.
0xFB	temp_lsb[7:0]	Contains the LSB part ut[11:4] of the raw temperature measurement output data.
0xFC (bit 7, 6, 5, 4)	temp_xlsb[3:0]	Contains the XLSB part ut[3:0] of the raw temperature measurement output data. Contents depend on pressure resolution.

5.4.9 Register 0xFD...0xFE *"hum" (_msb, _lsb)*

The *"temp"* register contains the raw temperature measurement output data ut[19:0]. For details on how to read out the pressure and temperature information from the device, please consult chapter 4.

Table 31: Register 0xFD ... 0xFE *"hum"*

Register 0xFD...0xFE *"hum"*	Name	Description
0xFD	hum_msb[7:0]	Contains the MSB part uh[15:8] of the raw humidity measurement output data.
0xFE	temp_lsb[7:0]	Contains the LSB part uh[7:0] of the raw humidity measurement output data.

6. Digital interfaces

The BME280 supports the I²C and SPI digital interfaces; it acts as a slave for both protocols. The I²C interface supports the Standard, Fast and High Speed modes. The SPI interface supports both SPI mode '00' (CPOL = CPHA = '0') and mode '11' (CPOL = CPHA = '1') in 4-wire and 3-wire configuration.

The following transactions are supported:

- Single byte write
- multiple byte write (using pairs of register addresses and register data)
- single byte read
- multiple byte read (using a single register address which is auto-incremented)

6.1 Interface selection

Interface selection is done automatically based on CSB (chip select) status. If CSB is connected to V_{DDIO}, the I²C interface is active. If CSB is pulled down, the SPI interface is activated. After CSB has been pulled down once (regardless of whether any clock cycle occurred), the I²C interface is disabled until the next power-on-reset. This is done in order to avoid inadvertently decoding SPI traffic to another slave as I²C data. Since the device startup is deferred until both V_{DD} and V_{DDIO} are established, there is no risk of incorrect protocol detection because of the power-up sequence used. However, if I²C is to be used and CSB is not directly connected to V_{DDIO} but is instead connected to a programmable pin, it must be ensured that this pin already outputs the V_{DDIO} level during power-on-reset of the device. If this is not the case, the device will be locked in SPI mode and not respond to I²C commands.

6.2 I²C Interface

The I²C slave interface is compatible with Philips I²C Specification version 2.1. For detailed timings, please review Table 33. All modes (standard, fast, high speed) are supported. SDA and SCL are not pure open-drain. Both pads contain ESD protection diodes to VDDIO and GND. As the devices does not perform clock stretching, the SCL structure is a high-Z input without drain capability.

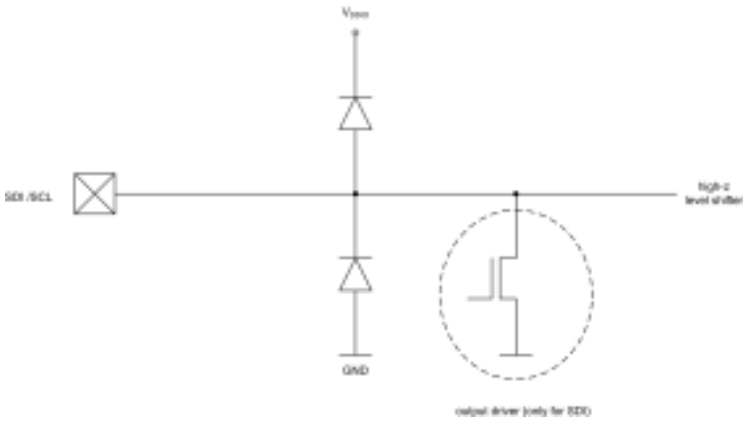


Figure 8: SDI/SCK ESD drawing

The 7-bit device address is 111011x. The 6 MSB bits are fixed. The last bit is changeable by SDO value and can be changed during operation. Connecting SDO to GND results in slave address 1110110 (0x76); connection it to V_{DDIO} results in slave address 1110111 (0x77), which is the same as

BMP280's I²C address. The SDO pin cannot be left floating; if left floating, the I²C address will be undefined.

The I²C interface uses the following pins:

- SCK: serial clock (SCL)
- SDI: data (SDA)
- SDO: Slave address LSB (GND = '0', V_{DDIO} = '1')

CSB must be connected to V_{DDIO} to select I²C interface. SDI is bi-directional with open drain to GND: it must be externally connected to V_{DDIO} via a pull up resistor. Refer to chapter 7 for connection instructions.
The following abbreviations will be used in the I²C protocol figures:

- S Start
- P Stop
- ACKS Acknowledge by slave
- ACKM Acknowledge by master
- NACKM Not acknowledge by master

6.2.1 I²C write

Writing is done by sending the slave address in write mode (RW = '0'), resulting in slave address 111011X0 ('X' is determined by state of SDO pin. Then the master sends pairs of register addresses and register data. The transaction is ended by a stop condition. This is depicted in Figure 9.

Figure 9: I²C multiple byte write (not auto-incremented)

6.2.2 I²C read

To be able to read registers, first the register address must be sent in write mode (slave address 111011X0). Then either a stop or a repeated start condition must be generated. After this the slave is addressed in read mode (RW = '1') at address 111011X1, after which the slave sends out data from auto-incremented register addresses until a NOACKM and stop condition occurs. This is depicted in Figure 10, where register 0xF6 and 0xF7 are read.

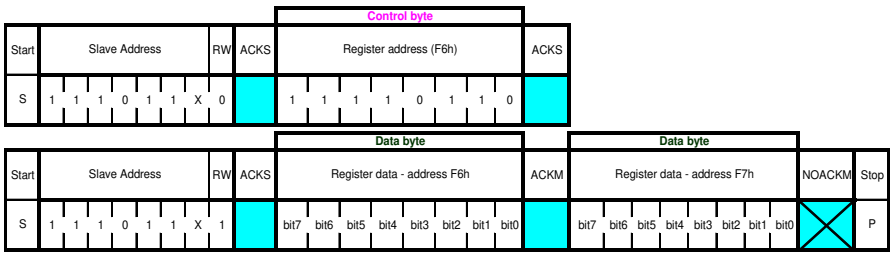


Figure 10: I²C multiple byte read

6.3 SPI interface

The SPI interface is compatible with SPI mode '00' (CPOL = CPHA = '0') and mode '11' (CPOL = CPHA = '1'). The automatic selection between mode '00' and '11' is determined by the value of SCK after the CSB falling edge.
The SPI interface has two modes: 4-wire and 3-wire. The protocol is the same for both. The 3-wire mode is selected by setting '1' to the register spi3w_en. The pad SDI is used as a data pad in 3-wire mode.

The SPI interface uses the following pins:

- CSB: chip select, active low
- SCK: serial clock
- SDI: serial data input; data input/output in 3-wire mode
- SDO: serial data output; hi-Z in 3-wire mode

Refer to chapter 7 for connection instructions.

CSB is active low and has an integrated pull-up resistor. Data on SDI is latched by the device at SCK rising edge and SDO is changed at SCK falling edge. Communication starts when CSB goes to low and stops when CSB goes to high; during these transitions on CSB, SCK must be stable. The SPI protocol is shown in Figure 11. For timing details, please review Table 34.

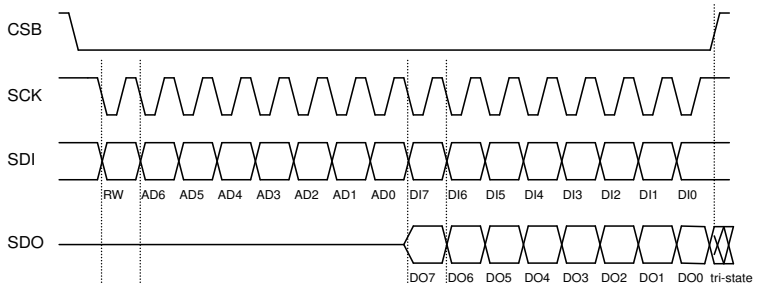


Figure 11: SPI protocol (shown for mode '11' in 4-wire configuration)

In SPI mode, only 7 bits of the register addresses are used; the MSB of register address is not used and replaced by a read/write bit (RW = '0' for write and RW = '1' for read).
Example: address 0xF7 is accessed by using SPI register address 0x77. For write access, the byte 0x77 is transferred, for read access, the byte 0xF7 is transferred.

6.3.1 SPI write

Writing is done by lowering CSB and sending pairs control bytes and register data. The control bytes consist of the SPI register address (= full register address without bit 7) and the write command (bit7 = RW = '0'). Several pairs can be written without raising CSB. The transaction is ended by a raising CSB. The SPI write protocol is depicted in Figure 12.

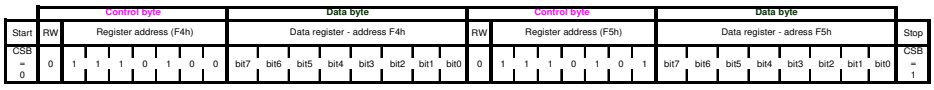


Figure 12: SPI multiple byte write (not auto-incremented)

6.3.2 SPI read

Reading is done by lowering CSB and first sending one control byte. The control bytes consist of the SPI register address (= full register address without bit 7) and the read command (bit 7 = RW = '1'). After writing the control byte, data is sent out of the SDO pin (SDI in 3-wire mode); the register address is automatically incremented. The SPI read protocol is depicted in Figure 13.

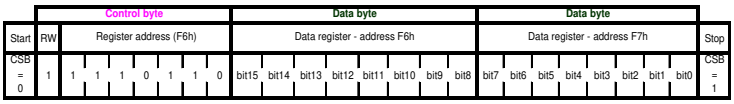


Figure 13: SPI multiple byte read

6.4 Interface parameter specification

6.4.1 General interface parameters

The general interface parameters are given in Table 32 below.

Table 32: interface parameters

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Input low level	V_{il_si}	V_{DDIO}=1.2 V to 3. 6V			20	%V_{DDIO}
Input high level	V_{ih_si}	V_{DDIO}=1.2 V to 3.6 V	80			%V_{DDIO}
Output low level I²C	V_{ol_SDI}	V_{DDIO}=1.62 V, I_{ol}=3 mA			20	%V_{DDIO}
Output low level I²C	$V_{ol\_SDI\_1.2}$	V_{DDIO}=1.20 V, I_{ol}=3 mA			23	%V_{DDIO}
Output low level SPI	V_{ol_SDO}	V_{DDIO}=1.62 V, I_{ol}=1 mA			20	%V_{DDIO}
Output low level SPI	$V_{ol\_SDO\_1.2}$	V_{DDIO}=1.20 V, I_{ol}=1 mA			23	%V_{DDIO}
Output high level	V_{oh}	V_{DDIO}=1.62 V, I_{oh}=1 mA (SDO, SDI)	80			%V_{DDIO}
Output high level	$V_{oh\_1.2}$	V_{DDIO}=1.20 V, I_{oh}=1 mA (SDO, SDI)	60			%V_{DDIO}
Pull-up resistor	R_{pull}	Internal CSB pull-up resistance to V_{DDIO}	70	120	190	kΩ
I²C bus load capacitor	C_b	On SDI and SCK			400	pF

6.4.2 I²C timings

For I²C timings, the following abbreviations are used:

- "S&F mode" = standard and fast mode
- "HS mode" = high speed mode
- C_b = bus capacitance on SDA line

All other naming refers to I²C specification 2.1 (January 2000).

The I²C timing diagram is in Figure 14. The corresponding values are given in Table 33.

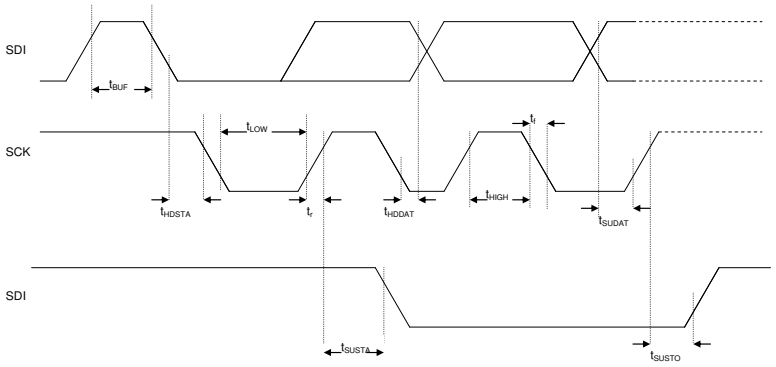


Figure 14: I²C timing diagram

Table 33: I²C timings

Parameter	Symbol	Condition	Min	Typ	Max	Unit
SDI setup time	$t_{SU;DAT}$	S&F Mode HS mode	160 30			ns ns
SDI hold time	$t_{HD;DAT}$	S&F Mode, C_b≤100 pF S&F Mode, C_b≤400 pF HS mode, C_b≤100 pF HS mode, C_b≤400 pF	80 90 18 24		 115 150	ns ns ns ns
SCK low pulse	t_{LOW}	HS mode, C_b≤100 pF V_{DDIO} = 1.62 V	160			ns
SCK low pulse	t_{LOW}	HS mode, C_b≤100 pF V_{DDIO} = 1.2 V	210			ns

The above-mentioned I²C specific timings correspond to the following internal added delays:

- Input delay between SDI and SCK inputs: SDI is more delayed than SCK by typically 100 ns in Standard and Fast Modes and by typically 20 ns in High Speed Mode.
- Output delay from SCK falling edge to SDI output propagation is typically 140 ns in Standard and Fast Modes and typically 70 ns in High Speed Mode.

6.4.3 SPI timings

The SPI timing diagram is in Figure 15, while the corresponding values are given in Table 34. All timings apply both to 4- and 3-wire SPI.

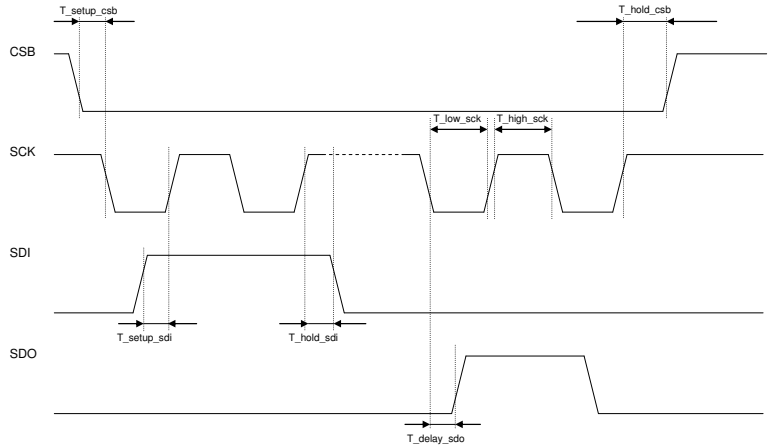


Figure 15: SPI timing diagram

Table 34: SPI timings

Parameter	Symbol	Condition	Min	Typ	Max	Unit
SPI clock input frequency	F_spi		0		10	MHz
SCK low pulse	T_low_sck		20			ns
SCK high pulse	T_high_sck		20			ns
SDI setup time	T_setup_sdi		20			ns
SDI hold time	T_hold_sdi		20			ns
SDO output delay	T_delay_sdo	25 pF load, V_{DDIO}=1.6 V min			30	ns
SDO output delay	T_delay_sdo	25 pF load, V_{DDIO}=1.2 V min			40	ns
CSB setup time	T_setup_csb		20			ns
CSB hold time	T_hold_csb		20			ns

7. Pin-out and connection diagram

7.1 Pin-out

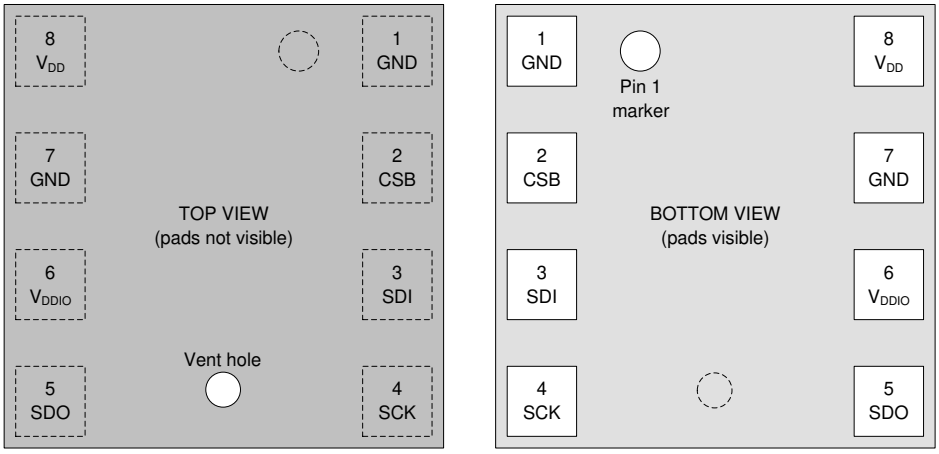


Figure 16: Pin-out top and bottom view

Note: The pin numbering of BME280 is performed in the untypical clockwise direction when seen in top view and counter-clockwise when seen in bottom view.

Table 35: Pin description

Pin	Name	I/O Type	Description	Connect to		
				SPI 4W	SPI 3W	I²C
1	GND	Supply	Ground	GND		
2	CSB	In	Chip select	CSB	CSB	V_{DDIO}
3	SDI	In/Out	Serial data input	SDI	SDI/SDO	SDA
4	SCK	In	Serial clock input	SCK	SCK	SCL
5	SDO	In/Out	Serial data output	SDO	DNC	GND for default address
6	V_{DDIO}	Supply	Digital / Interface supply	V_{DDIO}		
7	GND	Supply	Ground	GND		
8	V_{DD}	Supply	Analog supply	V_{DD}		

7.2 Connection diagram I²C

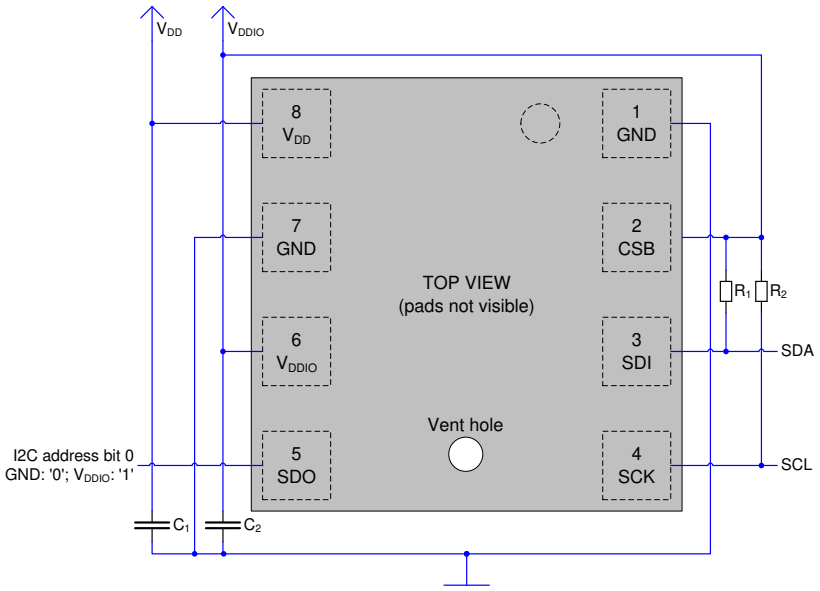


Figure 17: I²C connection diagram

Notes:

- The recommended value for C_1, C_2 is 100 nF
- The value for the pull-up resistors R_1, R_2 should be based on the interface timing and the bus load; a normal value is 4.7 kΩ
- A direct connection between CSB and V_{DDIO} is required

7.3 Connection diagram 4-wire SPI

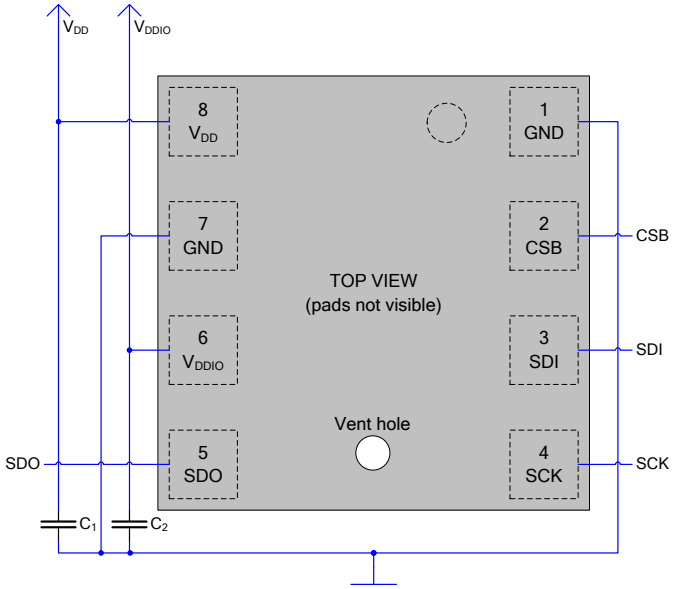


Figure 18: 4-wire SPI connection diagram

Note: The recommended value for C_1, C_2 is 100 nF

7.4 Connection diagram 3-wire SPI

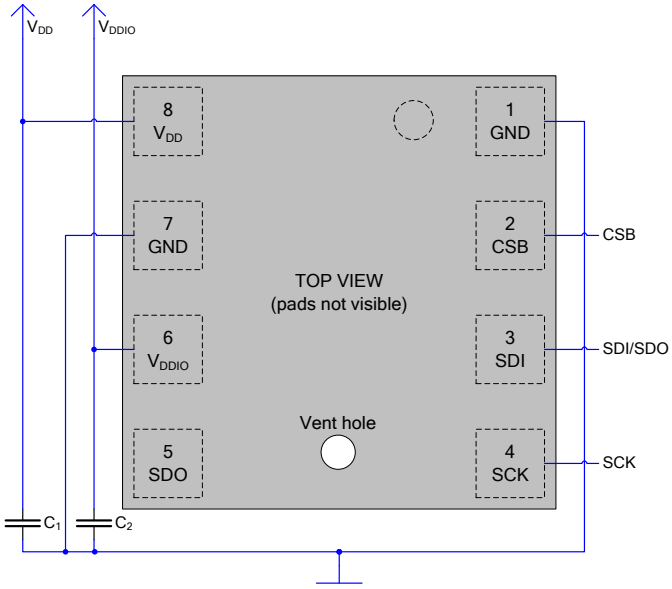


Figure 19: 3-wire SPI connection diagram

Note: The recommended value for C_1, C_2 is 100 nF

7.5 Package dimensions

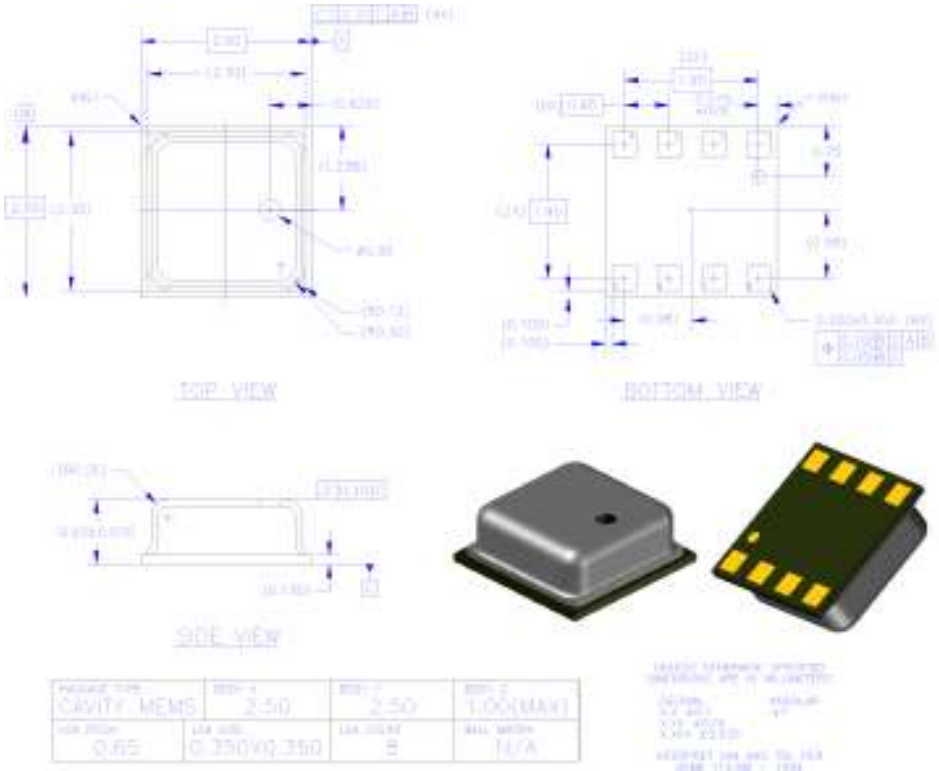


Figure 20: Package dimensions for top, bottom and side view

7.6 Landing pattern recommendation

For the design of the landing pattern, the following dimensioning is recommended:

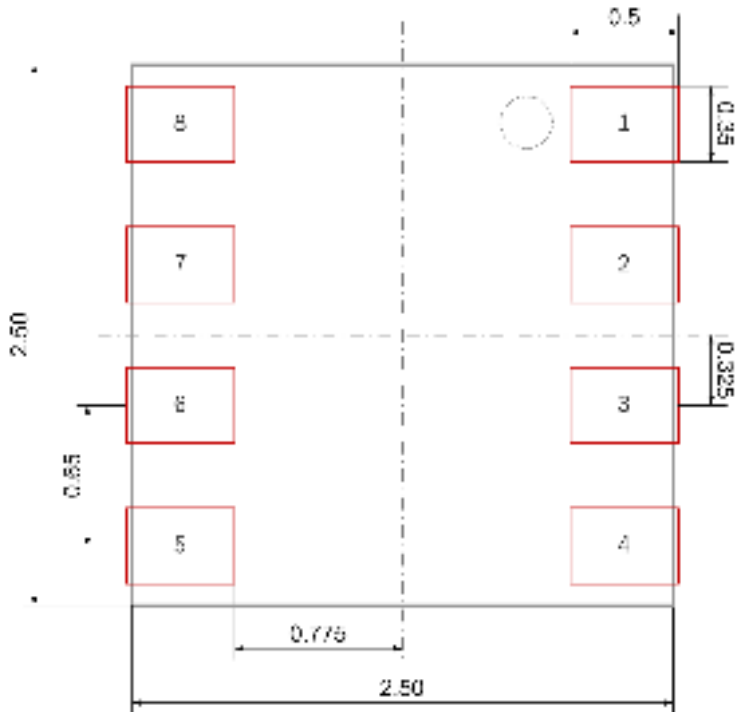


Figure 21: Recommended landing pattern (top view)

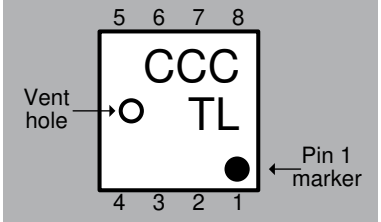
Note: red areas demark exposed PCB metal pads.

- In case of a solder mask defined (SMD) PCB process, the land dimensions should be defined by solder mask openings. The underlying metal pads are larger than these openings.
- In case of a non solder mask defined (NSMD) PCB process, the land dimensions should be defined in the metal layer. The mask openings are larger than these metal pads.

7.7 Marking

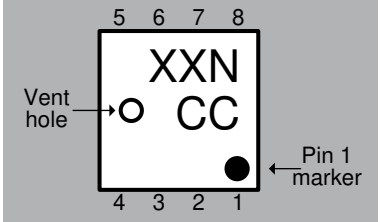
7.7.1 Mass production devices

Table 36: Marking of mass production parts

Marking	Symbol	Description
	CCC	Lot counter: 3 alphanumeric digits, variable to generate mass production trace-code
	T	Product number: 1 alphanumeric digit, fixed to identify product type, T = “U” “U” is associated with the product BME280 (part number 0 273 141 185)
	L	Sub-contractor ID: 1 alphanumeric digit, variable to identify sub-contractor (L = “P”)

7.7.2 Engineering samples

Table 37: Marking of engineering samples

Marking	Symbol	Description
	XX	Sample ID: 2 alphanumeric digits, variable to generate trace-code
	N	Eng. Sample ID: 1 alphanumeric digit, fixed to identify engineering sample, N = “ * ” or “e” or “E”
	CC	Counter ID: 2 alphanumeric digits, variable to generate trace-code

7.8 Soldering guidelines and reconditioning recommendations

The moisture sensitivity level of the BME280 sensors corresponds to JEDEC Level 1, see also:

- IPC/JEDEC J-STD-020C "Joint Industry Standard: Moisture/Reflow Sensitivity Classification for non-hermetic Solid State Surface Mount Devices"
- IPC/JEDEC J-STD-033A "Joint Industry Standard: Handling, Packing, Shipping and Use of Moisture/Reflow Sensitive Surface Mount Devices".

The sensor fulfils the lead-free soldering requirements of the above-mentioned IPC/JEDEC standard, i.e. reflow soldering with a peak temperature up to 260°C. The minimum height of the solder after reflow shall be at least 50µm. This is required for good mechanical decoupling between the sensor device and the printed circuit board (PCB).

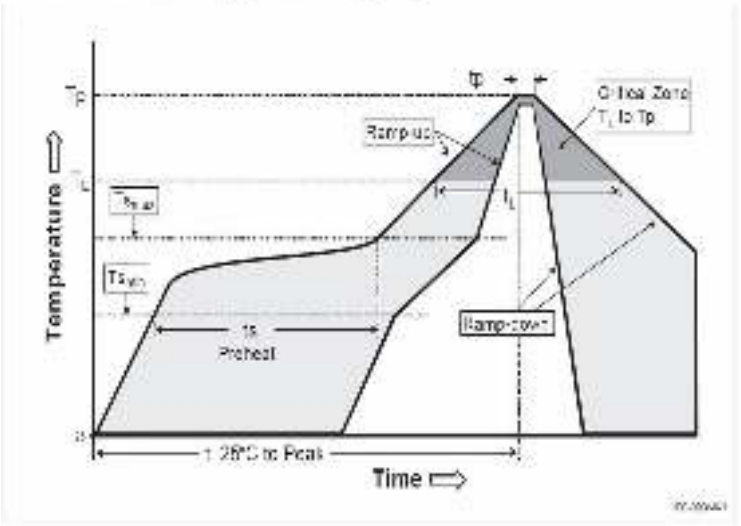


Figure 22: Soldering profile

7.9 Reconditioning Procedure

After exposing the device to operating conditions, which exceed the limits specified in section 1.2, e.g. after reflow, the humidity sensor may possess an additional offset. Therefore the following reconditioning procedure is mandatory to restore the calibration state:

1. Dry-Baking: 120 °C at <5% rH for 2 h
2. Re-Hydration: 70 °C at 75% rH for 6 h

or alternatively

1. Dry-Baking: 120 °C at <5% rH for 2 h
2. Re-Hydration: 25 °C at 75% rH for 24 h

or alternatively after solder reflow only

1. Do not perform Dry-Baking
2. Ambient Re-Hydration: ~25 °C at >40% rH for >5d

7.10 Tape and reel specification

7.10.1 Dimensions

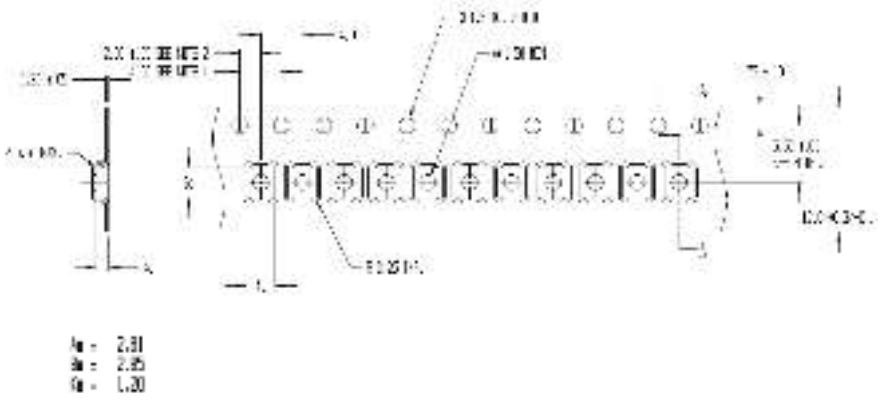


Figure 23: Tape and Reel dimensions

Quantity per reel: 10 kpcs.

7.10.2 Orientation within the reel

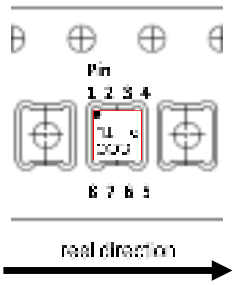


Figure 24: Orientation within tape

7.11 Mounting and assembly recommendations

In order to achieve the specified performance for you design, the following recommendations and the "Handling, soldering & mounting instructions BME280" should be taken into consideration when mounting a pressure sensor on a printed-circuit board (PCB):

- The clearance above the metal lid shall be 0.1mm at minimum.
- For the device housing appropriate venting needs to be provided in case the ambient pressure shall be measured.
- Liquids shall not come into direct contact with the device.
- During operation the sensor chip is sensitive to light, which can influence the accuracy of the measurement (photo-current of silicon). The position of the vent hole minimizes the light exposure of the sensor chip. Nevertheless, Bosch Sensortec recommends avoiding the exposure of BME280 to strong light sources.
- Soldering may not be done using vapor phase processes since the sensor will be damaged by the liquids used in these processes.

7.12 Environmental safety

7.12.1 RoHS

The BME280 sensor meets the requirements of the EC restriction of hazardous substances (RoHS) directive, see also:

Directive 2011/65/EU of the European Parliament and of the Council of 8 June 2011 on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

7.12.2 Halogen content

The BME280 is halogen-free. For more details on the analysis results please contact your Bosch Sensortec representative.

7.12.3 Internal package structure

Within the scope of Bosch Sensortec's ambition to improve its products and secure the mass product supply, Bosch Sensortec qualifies additional sources (e.g. 2nd source) for the package of the BME280.

While Bosch Sensortec took care that all of the technical packages parameters are described above are 100% identical for all sources, there can be differences in the chemical content and the internal structural between the different package sources.

However, as secured by the extensive product qualification process of Bosch Sensortec, this has no impact to the usage or to the quality of the BME280 product.

8. Appendix A: Alternative compensation formulas

8.1 Compensation formulas in double precision floating point

Please note that it is strongly advised to use the API available from Bosch Sensortec to perform readout and compensation. If this is not wanted, the code below can be applied at the user's risk. Both pressure and temperature values are expected to be received in 20 bit format, positive, stored in a 32 bit signed integer. Humidity is expected to be received in 16 bit format, positive, stored in a 32 bit signed integer.
The variable t_fine (signed 32 bit) carries a fine resolution temperature value over to the pressure compensation formula and could be implemented as a global variable.
The data type "BME280_S32_t" should define a 32 bit signed integer variable type and could usually be defined as "long signed int". The revision of the code is rev. 1.1 (pressure and temperature) and rev. 1.0 (humidity).
Compensating the measurement value with double precision gives the best possible accuracy but is only recommended for PC applications.

```
// Returns temperature in DegC, double precision. Output value of "51.23" equals 51.23 DegC.
// t_fine carries fine temperature as global value
BME280_S32_t t_fine;
double BME280_compensate_T_double(BME280_S32_t adc_T)
{
   double var1, var2, T;
   var1  = (((double)adc_T)/16384.0 - ((double)dig_T1)/1024.0) * ((double)dig_T2);
   var2  = ((((double)adc_T)/131072.0 - ((double)dig_T1)/8192.0) *
      (((double)adc_T)/131072.0 - ((double) dig_T1)/8192.0)) * ((double)dig_T3);
   t_fine = (BME280_S32_t)(var1 + var2);
   T  = (var1 + var2) / 5120.0;
   return T;
}
// Returns pressure in Pa as double. Output value of "96386.2" equals 96386.2 Pa = 963.862 hPa
double BME280_compensate_P_double(BME280_S32_t adc_P)
{
   double var1, var2, p;
   var1 = ((double)t_fine/2.0) - 64000.0;
   var2 = var1 * var1 * ((double)dig_P6) / 32768.0;
   var2 = var2 + var1 * ((double)dig_P5) * 2.0;
   var2 = (var2/4.0)+(((double)dig_P4) * 65536.0);
   var1 = (((double)dig_P3) * var1 * var1 / 524288.0 + ((double)dig_P2) * var1) / 524288.0;
   var1 = (1.0 + var1 / 32768.0)*((double)dig_P1);
   if (var1 == 0.0)
   {
      return 0; // avoid exception caused by division by zero
   }
   p = 1048576.0 - (double)adc_P;
   p = (p - (var2 / 4096.0)) * 6250.0 / var1;
   var1 = ((double)dig_P9) * p * p / 2147483648.0;
   var2 = p * ((double)dig_P8) / 32768.0;
   p = p + (var1 + var2 + ((double)dig_P7)) / 16.0;
   return p;
}
// Returns humidity in %rH as as double. Output value of "46.332" represents
      46.332 %rH
double bme280_compensate_H_double(BME280_S32_t adc_H);
{
   double var_H;

   var_H = (((double)t_fine) - 76800.0);
   var_H = (adc_H - (((double)dig_H4) * 64.0 + ((double)dig_H5) / 16384.0 *
      var_H)) * (((double)dig_H2) / 65536.0 * (1.0 + ((double)dig_H6) /
      67108864.0 * var_H *
      (1.0 + ((double)dig_H3) / 67108864.0 * var_H)));
   var_H = var_H * (1.0 - ((double)dig_H1) * var_H / 524288.0);

   if (var_H > 100.0)
      var_H = 100.0;
   else if (var_H < 0.0)
      var_H = 0.0;
return var_H;
}
```

8.2 Pressure compensation in 32 bit fixed point

Please note that it is strongly advised to use the API available from Bosch Sensortec to perform readout and compensation. If this is not wanted, the code below can be applied at the user's risk. Both pressure and temperature values are expected to be received in 20 bit format, positive, stored in a 32 bit signed integer.
The variable t_fine (signed 32 bit) carries a fine resolution temperature value over to the pressure compensation formula and could be implemented as a global variable.
The data type "BME280_S32_t" should define a 32 bit signed integer variable type and can usually be defined as "long signed int".
The data type "BME280_U32_t" should define a 32 bit unsigned integer variable type and can usually be defined as "long unsigned int".
Compensating the pressure value with 32 bit integer has an accuracy of typically 1 Pa (1-sigma). At high filter levels this adds a significant amount of noise to the output values and reduces their resolution.

```
// Returns temperature in DegC, resolution is 0.01 DegC. Output value of "5123" equals 51.23
DegC.
// t_fine carries fine temperature as global value
BME280_S32_t t_fine;
BME280_S32_t BME280_compensate_T_int32(BME280_S32_t adc_T)
{
   BME280_S32_t var1, var2, T;
   var1  = ((((adc_T>>3) - ((BME280_S32_t)dig_T1<<1))) * ((BME280_S32_t)dig_T2)) >> 11;
   var2  = (((((adc_T>>4) - ((BME280_S32_t)dig_T1)) * ((adc_T>>4) - ((BME280_S32_t)dig_T1)))
>> 12) *
      ((BME280_S32_t)dig_T3)) >> 14;
   t_fine = var1 + var2;
   T  = (t_fine * 5 + 128) >> 8;
   return T;
}

// Returns pressure in Pa as unsigned 32 bit integer. Output value of "96386" equals 96386 Pa
= 963.86 hPa
BME280_U32_t BME280_compensate_P_int32(BME280_S32_t adc_P)
{
   BME280_S32_t var1, var2;
   BME280_U32_t p;
   var1 = (((BME280_S32_t)t_fine)>>1) - (BME280_S32_t)64000;
   var2 = (((var1>>2) * (var1>>2)) >> 11 ) * ((BME280_S32_t)dig_P6);
   var2 = var2 + ((var1*((BME280_S32_t)dig_P5))<<1);
   var2 = (var2>>2)+(((BME280_S32_t)dig_P4)<<16);
   var1 = (((dig_P3 * (((var1>>2) * (var1>>2)) >> 13 )) >> 3) + ((((BME280_S32_t)dig_P2) *
var1)>>1))>>18;
   var1 =((((32768+var1))*((BME280_S32_t)dig_P1))>>15);
   if (var1 == 0)
   {
      return 0; // avoid exception caused by division by zero
   }
   p = (((BME280_U32_t)(((BME280_S32_t)1048576)-adc_P)-(var2>>12)))*3125;
   if (p < 0x80000000)
   {
      p = (p << 1) / ((BME280_U32_t)var1);
   }
   else
   {
      p = (p / (BME280_U32_t)var1) * 2;
   }
   var1 = (((BME280_S32_t)dig_P9) * ((BME280_S32_t)(((p>>3) * (p>>3))>>13)))>>12;
   var2 = (((BME280_S32_t)(p>>2)) * ((BME280_S32_t)dig_P8))>>13;
   p = (BME280_U32_t)((BME280_S32_t)p + ((var1 + var2 + dig_P7) >> 4));
   return p;
}
```

9. Appendix B: Measurement time and current calculation

In this chapter, formulas are given to calculate measurement rate, filter bandwidth and current consumption in different settings.

9.1 Measurement time

The active measurement time depends on the selected values for humidity, temperature and pressure oversampling and can be calculated in milliseconds using the formulas below.

$$t_{measure,typ} = 1 + [2 \cdot T\_oversampling]_{osrs\_t \neq 0} + [2 \cdot P\_oversampling + 0.5]_{osrs\_p \neq 0} + [2 \cdot H\_oversampling + 0.5]_{osrs\_h \neq 0}$$

$$t_{measure,max} = 1.25 + [2.3 \cdot T\_oversampling]_{osrs\_t \neq 0} + [2.3 \cdot P\_oversampling + 0.575]_{osrs\_p \neq 0} + [2.3 \cdot H\_oversampling + 0.575]_{osrs\_h \neq 0}$$

For example, using temperature oversampling ×1, pressure oversampling ×4 and no humidity measurement, the measurement time is:

$$t_{measure,typ} = 1 + [2 \cdot 1] + [2 \cdot 4 + 0.5] + [0] = 11.5 \text{ ms}$$

$$t_{measure,max} = 1.25 + [2.3 \cdot 1] + [2.3 \cdot 4 + 0.575] + [0] = 13.325 \text{ ms}$$

9.2 Measurement rate in forced mode

In forced mode, the measurement rate depends on the rate at which it is forced by the master. The highest possible frequency in Hz can be calculated as:

$$ODR_{max,forced} = \frac{1000}{t_{measure}}$$

If measurements are forced faster than they can be executed, the data rate saturates at the attainable data rate. For the example above with 11.5 ms measurement time, the typically achievable output data rate would be:

$$ODR_{max,forced} = \frac{1000}{11.5} = 87 \text{ Hz}$$

9.3 Measurement rate in normal mode

The measurement rate in normal mode depends on the measurement time and the standby time and can be calculated in Hz using the following formula:

$$ODR_{normal\_mode} = \frac{1000}{t_{measure} + t_{standby}}$$

The accuracy of $t_{standby}$ is described in the specification parameter $\Delta t_{standby}$. For the example above with 11.5 ms measurement time, setting normal mode with a standby time of 62.5 ms would result in a data rate of:

$$ODR_{normal\_mode} = \frac{1000}{11.5 + 62.5} = 13.51 \text{ Hz}$$

9.4 Response time using IIR filter

When using the IIR filter, the response time of the sensor depends on the selected filter coefficient and the data rate used. It can be calculated using the following formula:

$$t_{response,\,75\%} = \frac{1000 \cdot n_{samples,\,75\%}}{ODR}$$

For the example above with a data rate of 13.51 Hz, the user could select a filter coefficient of 8. According to Table 6, the number of samples needed to reach 75% of a step response using this filter setting is 11. The response time with filter is therefore:

$$t_{response,\,75\%} = \frac{1000 \cdot 11}{13.51} = 814\ \text{ms}$$

9.5 Current consumption

The current consumption depends on the selected oversampling settings, the measurement rate and the sensor mode, but not on the IIR filter setting. It can be calculated as:

$$I_{DD,forced} = I_{DDSL} \cdot (1 - t_{measure} \cdot ODR) + \frac{ODR}{1000} \cdot \left(205 + I_{DDT} \cdot [2 \cdot T\_oversampling]_{osrs\_t \neq 0} + I_{DDP} \cdot [2 \cdot P\_oversampling + 0.5]_{osrs\_p \neq 0} + I_{DDH} \cdot [2 \cdot H\_oversampling + 0.5]_{osrs\_h \neq 0}\right)$$

$$I_{DD,normal} = I_{DDSB} \cdot (1 - t_{measure} \cdot ODR) + \frac{ODR}{1000} \cdot \left(205 + I_{DDT} \cdot [2 \cdot T\_oversampling]_{osrs\_t \neq 0} + I_{DDP} \cdot [2 \cdot P\_oversampling + 0.5]_{osrs\_p \neq 0} + I_{DDH} \cdot [2 \cdot H\_oversampling + 0.5]_{osrs\_h \neq 0}\right)$$

Note that the only difference between forced and normal mode current consumption is that the current for the inactive time is either I_{DDSL} or I_{DDSB}. For the example above, the current would be

$$\begin{aligned} I_{DD,normal} &= 0.2 \cdot (1 - 0.0115 \cdot 13.51) + \frac{13.51}{1000}(205 + 350 \cdot [2 \cdot 1] + 714 \cdot [2 \cdot 4 + 0.5] + [0]) \\ &= 0.2 \cdot (0.845) + \frac{13.51}{1000}(205 + 700 + 6069 + 0) \\ &= 0.2 + 94.2 = 94.4\ \mu\text{A} \end{aligned}$$

10. Legal disclaimer

10.1 Engineering samples

Engineering Samples are marked with an asterisk (*) or (e). Samples may vary from the valid technical specifications of the product series contained in this data sheet. They are therefore not intended or fit for resale to third parties or for use in end products. Their sole purpose is internal client testing. The testing of an engineering sample may in no way replace the testing of a product series. Bosch Sensortec assumes no liability for the use of engineering samples. The Purchaser shall indemnify Bosch Sensortec from all claims arising from the use of engineering samples.

10.2 Product use

Bosch Sensortec products are developed for the consumer goods industry. They are not designed or approved for use in military applications, life-support appliances, safety-critical automotive applications and devices or systems where malfunctions of these products can reasonably be expected to result in personal injury. They may only be used within the parameters of this product data sheet.

The resale and/or use of products are at the Purchaser's own risk and the Purchaser's own responsibility.

The Purchaser shall indemnify Bosch Sensortec from all third party claims arising from any product use not covered by the parameters of this product data sheet or not approved by Bosch Sensortec and reimburse Bosch Sensortec for all costs in connection with such claims.

The Purchaser accepts the responsibility to monitor the market for the purchased products, particularly with regard to product safety, and inform Bosch Sensortec without delay of any security relevant incidents.

10.3 Application examples and hints

With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Bosch Sensortec hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights or copyrights of any third party. The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. They are provided for illustrative purposes only and no evaluation regarding infringement of intellectual property rights or copyrights or regarding functionality, performance or error has been made.

11. Document history and modification

Rev. No	Page	Description of modification/changes	Date
0.1		Document creation	2012-11-06
1.0		Final datasheet	2014-11-12
1.1	48	Updated RoHS directive to 2011/65/EU effective 8 June 2011	2015-05-07
1.2	2, 3	Adjusted target devices, applications	2015-10-15
1.4		Minor corrections	2018-01-17
1.5		Template update	2018-09-17

Bosch Sensortec GmbH
Gerhard-Kindler-Straße 9
72770 Reutlingen / Germany

contact@bosch-sensortec.com
www.bosch-sensortec.com

Document number: BST-BME280-DS002-15
Revision_1.6_092018

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Bosch:
BME280