UNIVERSITAT POLITÈCNICA DE VALÈNCIA

SISTEMA DISTRIBUIDO DE SENSORES PARA LA MONITORIZACIÓN DEL ESTRÉS TÉRMICO EN EL BARRIO DE BENICALAP

Trabajo Final de Máster

Alejandro Sánchez Aduna

Tutores:

Antonio Correcher Salvador

Carlos Vargas Salgado

Máster en Automática e Informática Industrial

Instituto Universitario de Automática e Informática Industrial

Universitat Politècnica de València

Curso 2019-2020

Valencia, diciembre de 2020

Resumen

En este trabajo se pretende diseñar, implementar y poner en funcionamiento un sistema distribuido de sensores, compuesto por estaciones de medida de variables medioambientales. El sistema debe tener las siguientes características: debe ser de bajo coste, replicable, tener una alimentación independiente de la red, y debe ser capaz de medir periódicamente las siguientes variables: temperatura, humedad, velocidad del viento, presión atmosférica, radiación solar, etc. Además, deberá comunicar de manera inalámbrica los datos de las variables a un sistema que los almacene en una base de datos en internet.

Este trabajo se enmarca en el proyecto GrowGreen, un proyecto integrado en el programa Horizonte2020 de la Unión Europea. El rango de actuación de este trabajo se limita a las acciones que el proyecto GrowGreen realiza en la ciudad de Valencia, concretamente en el barrio de Benicalap-Ciutat Fallera, con el objetivo de reducir el estrés térmico mediante soluciones inspiradas en la naturaleza.

Así pues, el objetivo de este trabajo consiste en desarrollar un sistema capaz de monitorizar indicadores del estrés térmico, como la temperatura de globo y bulbo húmedo y la temperatura radiante media, en diferentes localizaciones de la zona. Finalmente, se realizará una evaluación del desempeño del sistema y se analizarán los datos obtenidos. Desde el inicio de 2019, se han instalado ya un total de 11 dispositivos en distintas ubicaciones de Benicalap y, a día de hoy, siguen en funcionamiento y proporcionando datos que han sido utilizados en distintas publicaciones científicas.

Palabras clave: sensores, distribuido, estrés, térmico, datos, temperatura, medioambiente, GrowGreen, Arduino, Raspberry, impresión, 3D, Plesk, SQL.

Resum

En aquest treball es pretén dissenyar, implementar i posar en funcionament un sistema distribuït de sensors, compost per estacions de mesura de variables mediambientals. El sistema ha de tenir les següents característiques: ha de ser de baix cost, replicable, tenir una alimentació independent de la xarxa, i ser capaç de mesurar periòdicament les següents variables: temperatura, humitat, velocitat del vent, pressió atmosfèrica, radiació solar, etc. A més a més, haurà de comunicar sense fil les dades de les variables a un sistema que les emmagatzeme en una base de dades en internet.

El treball s'emmarca en el projecte GrowGreen, un projecte integrat en el programa Horitzó2020 de la Unió Europea. El rang d'actuació d'aquest treball es limita a les accions que el projecte GrowGreen realitza en la ciutat de València, concretament en el barri de Benicalap-Ciutat Fallera, amb l'objectiu de reduir l'estrès tèrmic mitjançant solucions basades en la natura.

Així doncs, l'objectiu d'este treball consisteix en desenvolupar un sistema capaç de monitorar indicadors de l'estrès tèrmic, com la temperatura de globus i bulb humit i la temperatura radiant mitjana, en diferents localitzacions de la zona. Finalment, es realitzarà una avaluació del funcionament del sistema i s'analitzaran les dades obtingudes. Des de l'inici de 2019, s'han instal·lat ja un total d'onze dispositius en distintes ubicacions de Benicalap i, a dia de hui, segueixen operatives i proporcionant dades que han sigut utilitzades en diverses publicacions científiques.

Paraules clau: sensors, distribuït, estrès, tèrmic, dades, temperatura, medi, ambient, GrowGreen, Arduino, Raspberry, impressió, 3D, Plesk, SQL.

Abstract

This paper deals with the design, assembly and operation of a distributed sensor system, consisting of measurement stations for climate variables. The system must comply with the following specifications: it must be low cost, reproducible, must have a stand-alone power supply and must be able to periodically measure the following variables: temperature, humidity, wind speed, atmospheric pressure, solar radiation, etc. In addition, it must communicate wirelessly the variable data to a system that stores it in a database on the Internet.

This work is part of the GrowGreen project, a European project integrated into the Horizon2020 program of the European Union. The scope of this project is limited to the actions the GrowGreen project carries out in the city of Valencia, specifically in the Benicalap-Ciutat Fallera district, with the aim of reducing thermal stress through nature-based solutions.

Thus, the aim of this work is to develop a system capable of monitoring thermal stress indicators, such as wet bulb globe temperature and mean radiant temperature, in different locations in the area. Finally, an evaluation of the performance of the system will be carried out and the data obtained will be analysed. Since the beginning of 2019, up to 11 devices have already been installed in different locations in Benicalap and, to this day, they're still on operation and providing data that has been used in several scientific papers.

Keywords: sensors, distributed, thermal, stress, data, temperature, climate, GrowGreen, Arduino, Raspberry, 3D, printing, Plesk, SQL.

Agradecimientos

Gracias a Jose Ignacio Marqués y a Joan Vicent Castillo por su inestimable trabajo y dedicación, ya que sin su ayuda este proyecto no habría podido salir adelante.

Índice

Índice de figuras

Nota: Todas las figuras de este documento son de fuente propia, a no ser que se indique lo contrario.

1. Introducción, motivación y objetivos

En los últimos años, el cambio climático se ha convertido en una realidad cada vez más cercana en el mundo, cuyos efectos son cada vez más visibles en el medio ambiente. Desde 1950, muchos de los cambios observados, no tienen precedentes en décadas, o incluso milenios. La atmósfera y los océanos han visto aumentada su temperatura, las cantidades de hielo y nieve han disminuido drásticamente aumentando el nivel del mar y las concentraciones de gases de efecto invernadero no han parado de ascender. [6]

Es más que probable que en las próximas décadas se sigan produciendo cambios profundos en el clima mundial: las temperaturas continuarán aumentando, habrá cambios en los patrones de precipitaciones, acrecentándose la incidencia de las sequías y las olas de calor, se espera un aumento de los eventos climáticos extremos, como los huracanes, y se producirán alteraciones importantes en el ciclo del agua. [6]

La influencia de la actividad humana ha sido un claro detonante de todos estos procesos y, desgraciadamente, también será quién más sufra sus consecuencias. Pero los efectos del cambio climático y la actividad humana en general no sólo se sienten a nivel global, sino también a nivel local. El reciente crecimiento progresivo de las ciudades está suponiendo importantes retos en materia medioambiental, como inundaciones, sequías, mala calidad del aire y estrés térmico, que a largo plazo afecta negativamente a la calidad de vida de sus habitantes. En este contexto, nace el proyecto GrowGreen.

1.1. El proyecto GrowGreen

El objetivo de GrowGreen es crear ciudades saludables y habitables, resistentes al clima y la escasez de agua, mediante soluciones inspiradas en la naturaleza. Pretende hacer de la naturaleza parte de las ciudades para mejorar la vida de los ciudadanos y ayudar a los negocios a prosperar. Con la creación de zonas verdes y vías de agua, se proponen soluciones innovadoras e inspiradoras en la planificación urbana a largo plazo para abordar los principales retos a los que se enfrentan las ciudades. [4]

Figura 1. Logo del proyecto GrowGreen. Fuente: growgreenproject.eu

GrowGreen es un proyecto financiado por el programa de investigación e innovación Horizonte2020 de la Unión Europea. Es un proyecto de 5 años de duración, que se inició en junio de 2017 y tiene prevista su finalización en mayo de 2022. Sus objetivos principales son los siguientes: [4]

1. Contribuir a la evidencia científica de las soluciones inspiradas en la naturaleza (*nature-based solutions,* NBS) en las ciudades, como medios asequibles y replicables de incrementar la resiliencia al agua y al clima, y para obtener beneficios tanto sociales y ambientales como económicos, con el objetivo de respaldar el desarrollo de políticas de NBS y el mercado global de NBS.

2. Desarrollar un enfoque replicable y fácil de usar para apoyar el desarrollo e implementación de estrategias NBS en las ciudades compatibles con sus prioridades actuales.

3. Apoyar la creación de las condiciones necesarias para el apoyo, impulso y activación de estrategias NBS mediante campañas de sensibilización y desarrollo de capacidades en ciudades de todo el mundo, apoyando el desarrollo del marco normativo necesario, de modelos de negocio de inversión en NBS y del mercado global de NBS.

Así, para cumplir estos objetivos, GrowGreen ha impulsado y diseñado proyectos de demostración de soluciones inspiradas en la naturaleza mediante asociaciones con un grupo de ciudades a través de Europa y China. En Manchester (Reino Unido), Valencia (España) y Breslavia (Polonia), se están llevando a cabo proyectos de demostración; en Brest (Francia), Zadar (Croacia) y Módena (Italia), se está trabajando con proyectos piloto para entender cómo las NBS pueden funcionar en sus ciudades. Además, la ciudad de Wuhan (China), también trabaja con NBS como parte de la fase piloto del programa *Chinese Sponge Cities*. [4]

1.2. GrowGreen Valencia

El clima de Valencia se caracteriza habitualmente por veranos calurosos y pocas precipitaciones. Las predicciones del cambio climático sugieren que en los próximos años se esperan temperaturas más altas y, probablemente, un aumento en eventos climáticos extremos. Estos últimos años hemos sido testigos de cada vez más fenómenos meteorológicos graves, como temporales marítimos, o tormentas tropicales que llegan hasta la península, así como olas de calor y sequías. Además, se espera que los veranos sean aún más calurosos, con los peligros para la salud que eso conlleva.

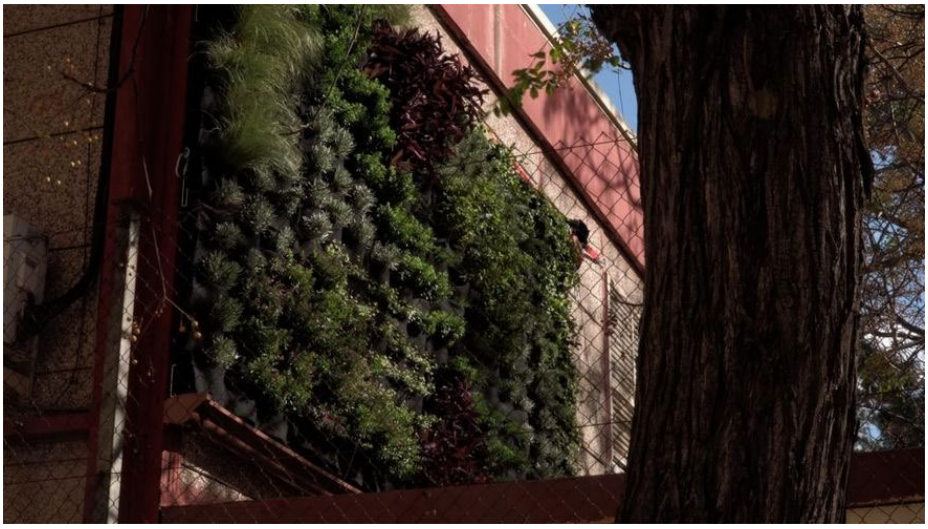


Figura 2. Jardín vertical instalado en el colegio de Benicalap. Fuente: levante-emv.com/valencia/2019/11/20/colegio-verde-benicalap-jardin-vertical-13725635.html

Así pues, el proyecto de demostración de GrowGreen en Valencia tiene por objetivo principal hacer frente al estrés térmico. Está situado en Benicalap-Ciutat Fallera, un barrio que presenta altos niveles de desempleo, una población envejecida y una infraestructura deteriorada. [5]

Una de las soluciones propuestas que se están probando actualmente se trata de un jardín vertical instalado en uno de los muros del Colegio Público Ciutat Artista Fallera, que proporciona regulación de la temperatura y aislamiento acústico. Además, filtra las aguas residuales provenientes de los lavabos y duchas, que luego es usada para regar el huerto escolar. [8]

Figura 3. Cubierta ajardinada instalada en el centro de mayores de Benicalap. Fuente: growgreenproject.eu/city-actions/valencia/

Por otro lado, el centro municipal de actividades para mayores se ha equipado con una cubierta ajardinada que permite reducir el calor en el interior del edificio, reduciendo así el consumo eléctrico derivado del sistema de aire acondicionado. [3] Además, está proyectada una nueva zona verde que supone una ampliación del parque de Benicalap y se convertiría en un punto de acumulación de agua de lluvia para evitar inundaciones y la sobrecarga del sistema de drenaje público. [7]

Figura 4. Emplazamiento de la nueva zona verde proyectada junto al parque de Benicalap. Fuente: lasprovincias.es/valencia-ciudad/nuevo-bosque-valencia-20200906001732-ntvo.html

Todas estas acciones se han diseñado e implementado en colaboración con los residentes del barrio. Además, se han llevado a cabo diversas campañas de concienciación y sensibilización entre los estudiantes y ciudadanos. En definitiva, los objetivos del proyecto GrowGreen en Valencia han sido los siguientes: [5]

- Demostrar el potencial de las soluciones inspiradas en la naturaleza para reducir el estrés térmico, mejorar la gestión del agua y otros beneficios adicionales.
- Involucrar a la población local en el diseño de estas soluciones que satisfagan sus necesidades, de forma que se consigan calles y ciudades más frescas, y un escenario favorable para la biodiversidad y la vegetación de la zona.
- Plantar las bases de una estrategia efectiva para desarrollar planes de este tipo para toda la ciudad.

1.3. Conceptos teóricos

Este fenómeno que se ha mencionado en varias ocasiones, el estrés térmico, resulta de suma importancia en este proyecto. Se define como la sensación de malestar que se experimenta cuando la permanencia en un ambiente determinado exige esfuerzos importantes a los mecanismos de que dispone el organismo para regular la temperatura corporal. Es decir, la exposición a situaciones de calor excesivo, puede provocar incomodidad o incluso generar riesgos para la salud de las personas. Escenarios como una exposición prolongada y factores personales como el sobrepeso o el estado de salud, pueden agravar los efectos que se producen. [14]

Puesto que el estrés térmico es un fenómeno difícil de cuantificar, habitualmente se usan los siguientes indicadores: la temperatura radiante media (RMT), la temperatura de globo y bulbo húmedo (WBGT) y la temperatura fisiológica equivalente (PET). Según NTP 322, el índice WBGT en exteriores, es decir, en ambientes con radiación solar, se calcula utilizando la Ecuación 1, donde t_{nwb} es la temperatura húmeda natural, t_g es la temperatura de globo negro y t_a es la temperatura ambiente: [11]

$$WBGT_{outdoor} = 0{,}7t_{nwb} + 0{,}2t_g + 0{,}1t_a \quad (1)$$

Esta temperatura húmeda natural se mide utilizando un termómetro con su bulbo cubierto con algodón humedecido sin protecciones frente al viento ni la radiación, y se puede calcular usando la Ecuación 2, como se muestra en (Alfonso-Solar, D., Bastida-Molina, P., Montuori, L., Vargas-Salgado, C., 2019), donde t_w es la temperatura psicométrica de bulbo húmedo, S la radiación solar y w_{speed} la velocidad del viento: [12]

$$t_{nwb} = t_w + 0{,}0021S - 0{,}42w_{speed} + 1{,}93 \quad (2)$$

A su vez, t_w se puede calcular usando las medidas de temperatura ambiente (t_a) y humedad relativa (RH%) al nivel del mar, usando la Ecuación 3: [12]

$$t_w = t_a \cdot \tan^{-1}\left[0{,}151977(RH\% + 8{,}131659)^{1/2}\right] + \tan^{-1}(t_a + RH\%) - \tan^{-1}(RH\% - 1{,}676331) + 0{,}00391838(RH\%)^{\frac{3}{2}} \cdot \tan^{-1}(0{,}023101RH\%) - 4{,}686035 \quad (3)$$

Por su parte, la temperatura radiante media (MRT) puede medirse, según (UNE ISO 7726, 2002), usando distintas ecuaciones. Para el caso de transmisión de calor por convección forzada usando un sensor de temperatura de globo negro con un diámetro de 15 cm, se calcula usando la Ecuación 4, donde w_{speed} es la velocidad del viento: [13]

$$\bar{t}_r = \left[\left(t_g + 273\right)^4 + 2{,}5 \cdot 10^8 \cdot \left(w_{speed}\right)^{0.6} \cdot \left(t_g - t_a\right)\right]^{1/4} - 273 \quad (4)$$

Finalmente, el índice PET puede modelarse utilizando la aplicación software de modelado RAYMAN. Sin embargo, según (Alfonso-Solar, D., Bastida-Molina, P., Montuori, L., Vargas-Salgado, C., 2019) puede existir una relación lineal entre el PET y la temperatura de globo negro (t_g), según la Ecuación 5, con un R^2 del 95%: [1]

$$PET = 1{,}113577 \cdot t_g - 5{,}814273 \quad (5)$$

1.4. Objetivos

Ahora que se ha descrito el proyecto GrowGreen, el contexto climático en el que se enmarca y se ha dado una pequeña introducción teórica a algunos aspectos clave, queda saber cómo se enmarcará este trabajo dentro del proyecto. Los objetivos principales que pretende alcanzar este proyecto son los siguientes:

- Este trabajo pretende servir de apoyo teórico y científico a las acciones que se realizan en el barrio de Benicalap (en línea con el primer objetivo del proyecto GrowGreen), proporcionando una estructura para recabar datos sobre el terreno.

- Se debe diseñar un sistema que permita monitorizar indicadores del estrés térmico en ubicaciones determinadas del barrio de Benicalap, de forma que se pueda evaluar el impacto de las acciones realizadas en la zona.

Para alcanzar estos objetivos, a continuación, se describen los objetivos específicos y los requisitos que este sistema deba cumplir:

- Se debe diseñar un sistema de bajo coste y replicable, con alimentación independiente de la red eléctrica, que permita recolectar y acceder a datos relativos a indicadores de estrés térmico.

- El dispositivo que se diseñe deberá estar preparado para trabajar tanto a la intemperie como en interiores y se deberán diseñar sujeciones que permitan su instalación en farolas, muros u otras localizaciones.

- Se instalará en ubicaciones clave del barrio de Benicalap que permitan obtener datos relevantes relativos a las acciones que se han realizado en el CEIP Ciudad Artista Fallero y el centro municipal de mayores, entre otras.

- El sistema deberá constar de un conjunto de sensores que permitan medir periódicamente (por defecto, 15 minutos) las siguientes variables ambientales: temperatura ambiente, temperatura de globo negro, humedad relativa, velocidad del viento, presión atmosférica, radiación solar y cantidad de lluvia.

- El sistema deberá ser capaz de transmitir esos datos de manera inalámbrica a una base de datos alojada en Internet donde pueda ser consultada para su posterior análisis.

2. Diseño del sistema: Descripción general

2.1. Introducción y metodología

A continuación, se procederá a describir el diseño del sistema propuesto. En este apartado se aportará una visión general de la arquitectura y una justificación para cada uno de sus dispositivos y nodos de comunicación. En sucesivos apartados se explicarán con detalle cada uno de los dispositivos, incluyendo una descripción de su hardware, circuito electrónico, construcción y programación, así como un análisis de las posibles alternativas que se plantearon.

Primeramente, cabe decir que este trabajo se ha desarrollado en el marco de unas prácticas en empresa en el Instituto de Ingeniería Energética (IIE) en la Ciudad Politécnica de la Innovación, de la UPV, iniciadas en diciembre de 2017. Al inicio de estas prácticas, el proyecto GrowGreen ya se encontraba en una fase inicial de desarrollo, por lo que a partir de esos primeros conceptos se desarrolló el sistema actual. Desde entonces, el proyecto ha pasado por diversas fases que se describirán a continuación. No se deben ver estas fases como periodos fijos y cerrados, sino como un proceso iterativo en que cada uno de las distintas etapas se superpone con las siguientes, de forma que se pretende obtener una evolución constante:

- **Fase de diseño (desde diciembre 2017):** Etapa de desarrollo en el que se realizan los principales trabajos de investigación del estado del arte y análisis de alternativas. En esta etapa se llevan a cabo las primeras pruebas de concepto, con pruebas de uso de sensores, alimentación y comunicaciones. Se construyen los primeros prototipos funcionales, con sus distintas versiones de hardware y software, así como la creación de la base de datos. Esta es una etapa sin final concreto, puesto que en cada momento se producen iteraciones de diseño a partir de los datos y la experiencia adquirida en etapas sucesivas, produciéndose mejoras y actualizaciones continuamente.

Figura 5. Primeras pruebas de prototipos en la UPV.

- **Fase de implementación y pruebas en la UPV (desde mayo de 2018):** Tras la construcción de los primeros prototipos y las pruebas de impresión de la estructura, se realizaron las primeras pruebas de operación en la UPV, colgando una serie de transmisores en farolas de la universidad. En esta etapa, se realizaron las pruebas de rango de comunicaciones, autonomía, integridad estructural, fiabilidad de medidas, durabilidad, etc. Gracias a los datos y la experiencia obtenida en estas pruebas, se aplicaron una serie de mejoras en las siguientes versiones de hardware y software que mejoraron sustancialmente el rendimiento del sistema. Se tratará con más detalle estas mejoras en el *Apartado 6: Resultados*.

- **Fase de pruebas en Benicalap (desde octubre de 2018):** En esta fase, de probó por primera vez el receptor de Benicalap, que se instaló en las oficinas del Parque Benicalap. Para testear su funcionamiento, se dejó un transmisor en el interior de la oficina y, posteriormente, se instaló un transmisor instalado en una farola en el interior del parque. Permitió probar la fiabilidad del sistema antes de la instalación final en Benicalap.

- **Fase de operación (desde enero de 2019):** Tras haber validado el diseño, se procedió a la instalación final de los transmisores en sus ubicaciones de Benicalap para la toma de datos. Desde entonces, se han realizado diversas tareas de mantenimiento, como sustitución de sensores o dispositivos. Los datos obtenidos también han permitido desarrollar nuevas mejoras, que se aplicarán próximamente en dispositivos de recambio que se instalarán en 2021.

2.2. Arquitectura del sistema

Seguidamente, procederemos a describir la arquitectura final del sistema y cada uno de los elementos que lo componen. Durante la fase de diseño, se barajaron distintas opciones, que se discutirán en sucesivos apartados, pero finalmente se consideró la que se describirá a continuación como la más adecuada para la aplicación requerida. Este sistema se ha denominado *Heat Stress Monitoring System* (HSM), y se compone de los siguientes elementos:

- **Transmisores:** Dispositivos electrónicos con microcontrolador ATMega y software basado en Arduino con sensores para las variables requeridas y alimentación a partir de panel solar y baterías de litio. Cada transmisor estará instalado en una ubicación concreta y transmitirá los datos medidos periódicamente a través de una emisión de radiofrecuencia de 433 MHz a un receptor instalado en una ubicación cercana.

- **Receptor:** Dispositivo constituido por un Arduino UNO conectado por USB a un computador Raspberry Pi con software basado en Debian. El Arduino UNO recibe los datos por radiofrecuencia y los envía por el puerto serie a la Raspberry Pi, que los procesa mediante scripts Python y los envía mediante un módem USB 3G con una tarjeta SIM a una base de datos en internet.

- **Base de datos:** Sistema de almacenamiento de datos mediante el servicio de alojamiento web Plesk. El receptor Raspberry Pi accede a un archivo PHP alojado en sus servidores que realiza una serie de peticiones SQL con las que guarda los datos en una base de datos MySQL. Así, los usuarios podrán acceder mediante un portal web a todos los datos recolectados por el sistema.

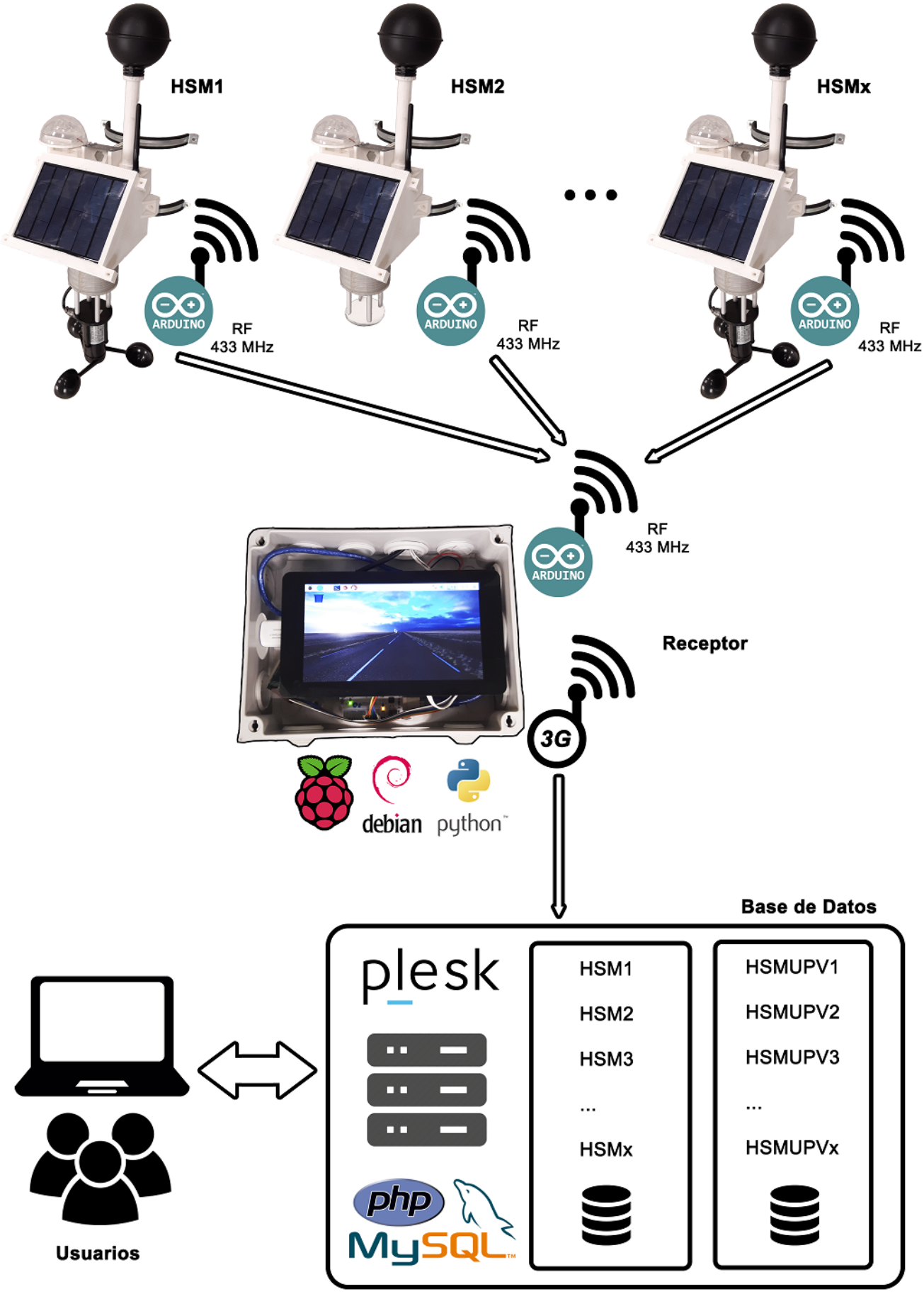


Figura 6. Esquema general de la arquitectura del sistema.

2.3. Análisis de alternativas

Hasta llegar al diseño final tanto del transmisor como del receptor, se barajaron una serie de alternativas para diferentes partes, que fueron finalmente descartadas por distintos motivos. Estas son algunas de las principales alternativas que se consideraron:

- Usar un único dispositivo para transmitir datos directamente a la nube sin un receptor: Esta fue la idea inicial para el diseño del transmisor. La idea era que la placa Arduino Mega contara con un *shield* GSM con una tarjeta SIM capaz de conectarse a Internet. Esto permitiría que cada transmisor enviara sus datos autónomamente desde cualquier lugar sin preocuparse por la cobertura de la antena de radiofrecuencia.

  Por desgracia, esta opción tenía una serie de inconvenientes. El primero y más importante, se comprobó tras una gran cantidad de pruebas que el microcontrolador ATMega no soporta la ejecución de los comandos necesarios para la petición que envía los datos a Plesk. Esto hubiera obligado a usar una versión diferente de microcontrolador, lo que hubiera incrementado el coste de componentes y de desarrollo, ya que supondría un cambio de arquitectura software, con lo que se perdería parte de la facilidad de uso de Arduino. Por otro lado, el uso de un transmisor con módulo GSM incorporado, hubiera requerido la adquisición de una tarjeta SIM para cada uno de los transmisores, lo que hubiera aumentado el coste.

  Cabe decir que, tiempo después de haber terminado la instalación en Benicalap, los conocimientos y la experiencia adquirida han permitido valorar otras alternativas con diferentes arquitecturas de hardware y software, como el uso de módulos basados en ESP32, que podrían haber resultado muy interesantes para su aplicación en el proyecto. Aun así, quedan anotadas para futuros proyectos.

- Sustituir el uso de módulos prefabricados de alimentación y sensores por un circuito personalizado con componentes SMD: Actualmente se usan módulos de conversión de tensión, carga de baterías, entre otros, que vienen ya fabricados y permiten su conexión directa en el circuito. Esto tiene una serie de ventajas, ya que simplifica mucho la electrónica y facilita el trabajo de desarrollo y fabricación sin necesidad de conocimientos muy avanzados. A cambio, aumenta el tamaño del circuito y disminuye la eficiencia energética, debido a la presencia de componentes innecesarios en cada módulo.

  La alternativa consiste en diseñar un circuito electrónico personalizado desde 0, eligiendo cada uno de los componentes para obtener las máximas prestaciones y el mínimo espacio mediante el uso de componentes SMD (Dispositivos de Montaje en Superficie) frente a los THT (Tecnología de Agujero Pasante). Como hemos dicho, esto requiere mayores conocimientos de electrónica y soldadura y dificulta las reparaciones. Y aunque en algunos casos resultan más asequibles, en otros la compra de componentes concretos al por menor puede ser más costosa que con el uso de módulos comerciales mucho más extendidos.

- Usar dos paneles solares en lugar de uno para la recolección de energía: Esta alternativa llegó a probarse tanto en la UPV como en Benicalap y, aunque prácticamente duplicaba la capacidad de carga de las baterías, actualmente está en desuso. Como se detalla en el *Apartado 6: Resultados*, la autonomía no supone un factor limitante, ya que los transmisores son capaces de aguantar más de 30 días con una sola carga de baterías, mientras que un panel solar puede cargarlas al máximo en unas 60 horas de insolación (unos 12 días).

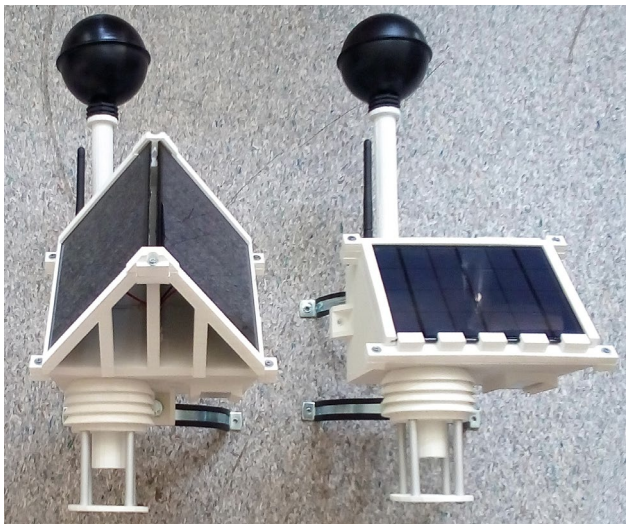


Figura 7. Versiones de transmisor con uno y dos paneles solares.

- Diferentes diseños estructurales: Por supuesto, se valoraron muchas opciones en cuanto a estructura del dispositivo. Las primeras versiones consistían en aparatos modulares basados en cajas de conexiones estancas, muy habituales en conexiones eléctricas. Por un lado, son asequibles, ofrecen buena protección y están hechos para trabajar en exteriores. Por desgracia, el dispositivo tenía la necesidad de muchas conexiones para sensores exteriores, para el panel solar, acoples para su instalación, etc. Eso suponía taladrar oberturas y comprometer su integridad y aislamiento, además de que hubiera necesitado de varias cajas interconectadas para satisfacer todos los requerimientos de diseño.

- Guardar los datos localmente sin base de datos en la nube: Una opción que se barajó fue la de utilizar un módulo de Arduino para tarjetas SD junto a un módulo de reloj de tiempo real (RTC), de forma que los datos que recolectara el transmisor, junto con la hora de la medida, se guardaran localmente en la tarjeta SD. De esa forma, se evitarían los problemas asociados a la transmisión de datos, incluyendo la necesidad de un receptor, y la infraestructura necesaria para el alojamiento de datos en la nube.

  El problema de esta propuesta es que se pierde el acceso en tiempo real a los datos, y supondría necesitar acceder físicamente a los transmisores para descargar los datos de las tarjetas SD cada cierta cantidad de tiempo. Además, no habría forma de detectar un fallo en el funcionamiento de los dispositivos, y eso podría suponer semanas o meses de datos perdidos.

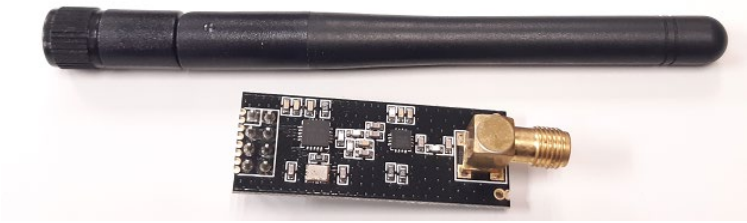


Figura 8. Módulo radiofrecuencia de 2,4 GHz nRF24L01.

- Usar un módulo de radiofrecuencia de 2,4 GHz nRF24L01: Esta fue la primera opción que se desarrolló y testeó antes de usar el módulo de 433 MHz. Los motivos de esta elección se explican con más detalle en el *Apartado 6.4: Alcance de comunicaciones*.

3. Diseño del sistema: Dispositivo transmisor

3.1. Descripción general del dispositivo

El dispositivo transmisor consiste en un sistema de sensorización de variables ambientales. En su versión más completa, el aparato incluye 4 sensores de temperatura, un sensor de presión, un sensor de humedad, un sensor de lluvia, dos de radiación solar (uno superior y otro inferior) y dos sensores de viento, además de otros de tensión y corriente eléctrica para supervisar el funcionamiento interno del sistema. Todo ello está controlado mediante un microcontrolador de tipo Arduino MEGA.

Cada cierto tiempo determinado y programable (por defecto, 15 minutos), el dispositivo se enciende y mide las variables deseadas, extrae una media, y envía los datos a través de una antena de radiofrecuencia a 433 MHz a otro aparato receptor situado en un lugar cercano, que posteriormente los enviará a una base de datos en Internet. Una vez concluida la transmisión, el aparato se queda en un estado de reposo hasta el siguiente ciclo.

El sistema se alimenta a través de 2 o 4 baterías de litio de 3,7 V formato 18650 situadas en el interior de la caja y, al mismo tiempo, dispone de un panel solar de 6 V y 3,5 W que carga las baterías durante los periodos de sol.

El sistema se encuentra montado dentro de una caja de plástico PLA diseñada a medida e impresa en una impresora 3D. Generalmente, las cajas están pintadas con pintura blanca y barniz transparente e identificadas cada una por un número.

La caja tiene una forma semejante a un prisma triangular para aprovechar la inclinación y mejorar la recepción de luz del panel solar. En un rellano horizontal en la parte superior, se encuentran la antena de radiofrecuencia, los sensores de radiación solar, lluvia y un sensor de temperatura ubicado dentro de un globo metálico negro, en el extremo de un pequeño poste de plástico. En la parte inferior, dentro de una pantalla protectora de radiación para sensores impresa en 3D, se encuentran otros dos sensores de temperatura, uno de humedad, uno de presión y uno de viento. Además, en la versión completa, montan en la parte inferior un anemómetro negro de copas.

Finalmente, todo el conjunto se podrá colgar en su ubicación final con dos tornillos metálicos largos y dos abrazaderas metálicas circulares de diferentes tamaños.

Figura 9. Transmisor completo listo para instalar.

3.2. Tecnologías utilizadas: Hardware y software

A continuación, se ofrece una lista detallada de los principales componentes y sensores utilizados en el diseño del dispositivo transmisor y una breve descripción de cada uno. Para una lista completa de todos los componentes y materiales utilizados en la fabricación, se puede consultar la lista que aparece en el *Apartado 7: Presupuesto* o el manual de montaje del *Anexo VIII*:

- **Placa Arduino Mega 2560:** Placa de desarrollo basada en el microcontrolador de 8 bits ATmega2560, con capacidad para 54 GPIO, 16 entradas analógicas, 4 puertos serie, conector USB y compatibilidad con protocolos de comunicación serie como SPI e I2C.

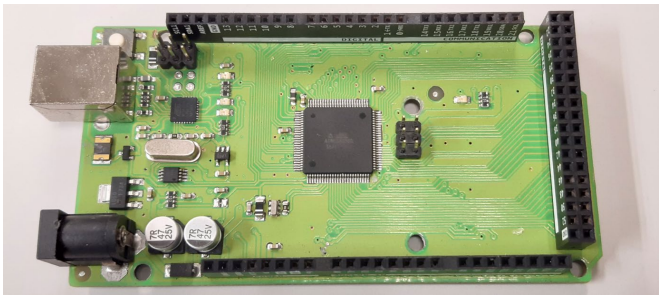


Figura 10. Placa Arduino Mega.

- **Módulo de radiofrecuencia E32-TTL-1W:** Módulo de comunicaciones por radiofrecuencia a 433 MHz E32-TTL-1W. Con interfaz de comunicación por UART y antena de largo alcance (alrededor de 7500 m).

Figura 11. Módulo de radiofrecuencia E32-TTL-1W

- **Sensor DTH22:** También llamado AM2302, es un sensor de humedad y temperatura. Usa un protocolo propio de comunicación serie a través de un pin digital y alimentación de 5V.

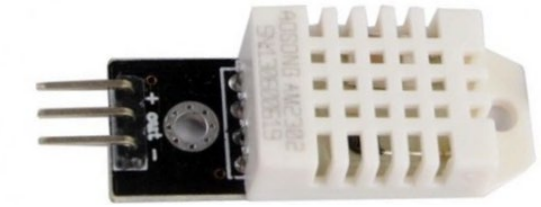


Figura 12. Módulo de sensor DHT22.

- **Sensor INA3221:** Módulo de sensor de tensión y corriente, con tres canales para medidas de corriente y una medida de tensión, con comunicación por bus I2C.

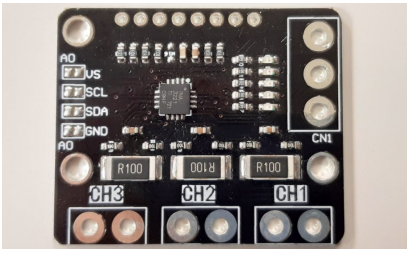


Figura 13. Módulo de sensor INA3221.

- **Módulo de carga VMA-321:** Módulo cargador de baterías de litio basado en el chip TP4056, con tensión de entrada de 4,5 a 5,5 V, tensión de carga de 4,2 V y corriente de carga máxima de 1 A.

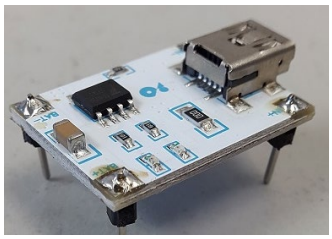


Figura 14. Módulo de carga VMA-321.

- **Convertidor de tensión VMA-403:** Módulo boost DC-DC. Convertidor de tensión de 2,5 a 5 V con salida a 5 V y 600 mA por USB.

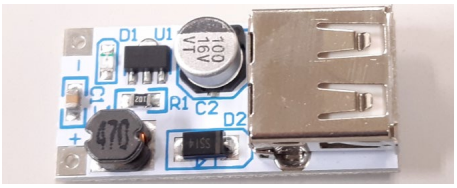


Figura 15. Módulo convertidor de tensión DC-DC VMA-403.

- **Sensor DS18B20 encapsulado:** Sensor de temperatura digital DS18B20 con encapsulado metálico. Utiliza un protocolo de comunicaciones serie digital llamado *OneWire*. El formato tipo sonda resulta ideal para trabajos en exterior. También está disponible en formato chip THT.

Figura 16. Sonda de temperatura DS18B20.

- **Sensor BME280:** Módulo de sensor de presión, humedad y temperatura BME280 con comunicación por bus I2C y alimentación de 3,3 V.

Figura 17. Módulo de sensor de presión BME280.

- **Módulo sensor de lluvia:** Sensor de gotas de lluvia (*Raindrops module*) MH-RD. Se trata de una placa que detecta las variaciones de resistencia eléctrica en su superficie por la presencia de gotas de lluvia. Aunque no permite establecer una equivalencia con L/m^2, ofrece una idea aproximada de la cantidad de lluvia en un momento determinado. Se comunica a través de una entrada analógica.

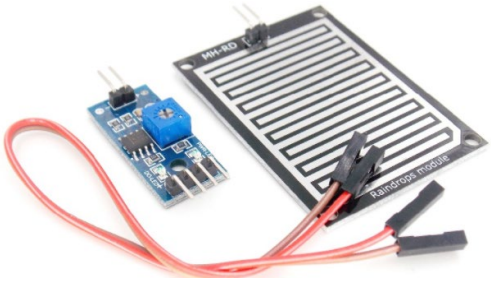


Figura 18. Módulo de sensor de lluvia MH-RD.

- **Sensor INA219:** Módulo de sensor de corriente de un canal con comunicación por bus I2C y dirección programable, para hasta 3,2 A.

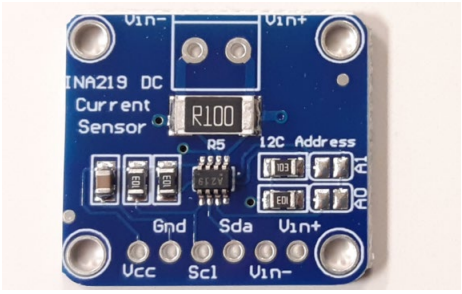


Figura 19. Módulo sensor de corriente INA219.

- **Temporizador TPL5110:** Módulo de interruptor electrónico con temporizador de bajo consumo, e intervalo regulable entre 100 ms y 2 horas.

Figura 20. Módulo temporizador TPL5110.

- **Sensor de radiación solar Cebekit C-0121:** Panel solar en miniatura que entrega una corriente eléctrica directamente proporcional a la radiación solar que recibe mediante una relación lineal calibrada por el fabricante.

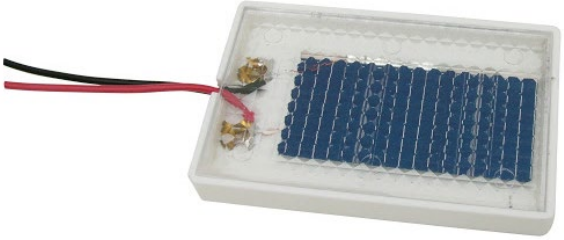


Figura 21. Sensor de radiación solar Cebekit C-0121.

- **Módulo convertidor de tensión de 5 a 12 V:** Módulo de convertidor Boost de 5 a 12 V y 5 W con entrada de USB.

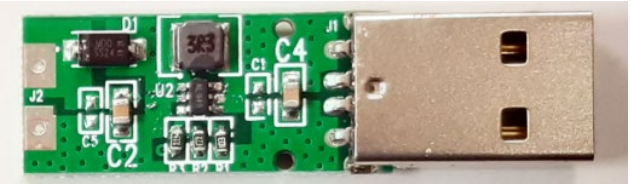


Figura 22. Módulo convertidor de tensión de 5 a 12 V.

- **Anemómetro de copas:** Sensor de velocidad del viento de 360 grados con copas de plástico. Ofrece un rango de hasta 32,4 m/s con una precisión de ±1 m/s y una velocidad de arranque de hasta 0,4 m/s. Tiene una señal de salida analógica de entre 0 y 5 V.

Figura 23. Anemómetro de tres copas.

Y en cuanto a software, ahora se ofrece una lista de las principales plataformas de desarrollo, lenguajes, protocolos de comunicación, etc., utilizados en el diseño del dispositivo transmisor junto a una breve descripción de cada uno:

- **Arduino (lenguaje C):** Plataforma de desarrollo de código abierto para microcontroladores de la familia AVR, como ATMega, aunque es compatible con una gran variedad de hardware. En ese caso, la programación se ha realizado utilizando el lenguaje C.

- **Comunicación serie UART:** Protocolo de comunicación serie denominado "Transmisor-Receptor Asíncrono Universal" para la comunicación entre dispositivos utilizado, por ejemplo, en la interfaz USB.

- **Bus de datos I2C:** Bus de datos síncrono para comunicación serie denominado "Circuito Inter-Integrado". Permite la comunicación en modo maestro-esclavo entre un gran número de dispositivos. Resulta especialmente útil para su uso entre dispositivos que se encuentren en la misma placa o a distancias cortas.

- **Bus de datos SPI:** Bus de datos síncrono para comunicación serie denominado "*Serial Peripheral Interface*". Presenta una línea para envío de datos, otro para recepción, una de reloj y otra de selección de periférico.

- **Bus de datos 1-Wire:** Protocolo de comunicación asíncrono diseñado por Dallas Semiconductor (fabricantes de los sensores DS18B20), que permite la comunicación en modo maestro-esclavo, con una sola línea de datos.

3.3. Descripción del circuito electrónico

Ahora que se han descrito los principales componentes, sensores y tecnologías utilizadas en este diseño, se procederá a describir el circuito electrónico que controla el dispositivo transmisor, así como las conexiones de cada uno de los sensores y elementos que lo componen. Así pues, empezamos con una visión general del esquema del circuito (véase *Anexo II* para ver el esquema a tamaño completo).

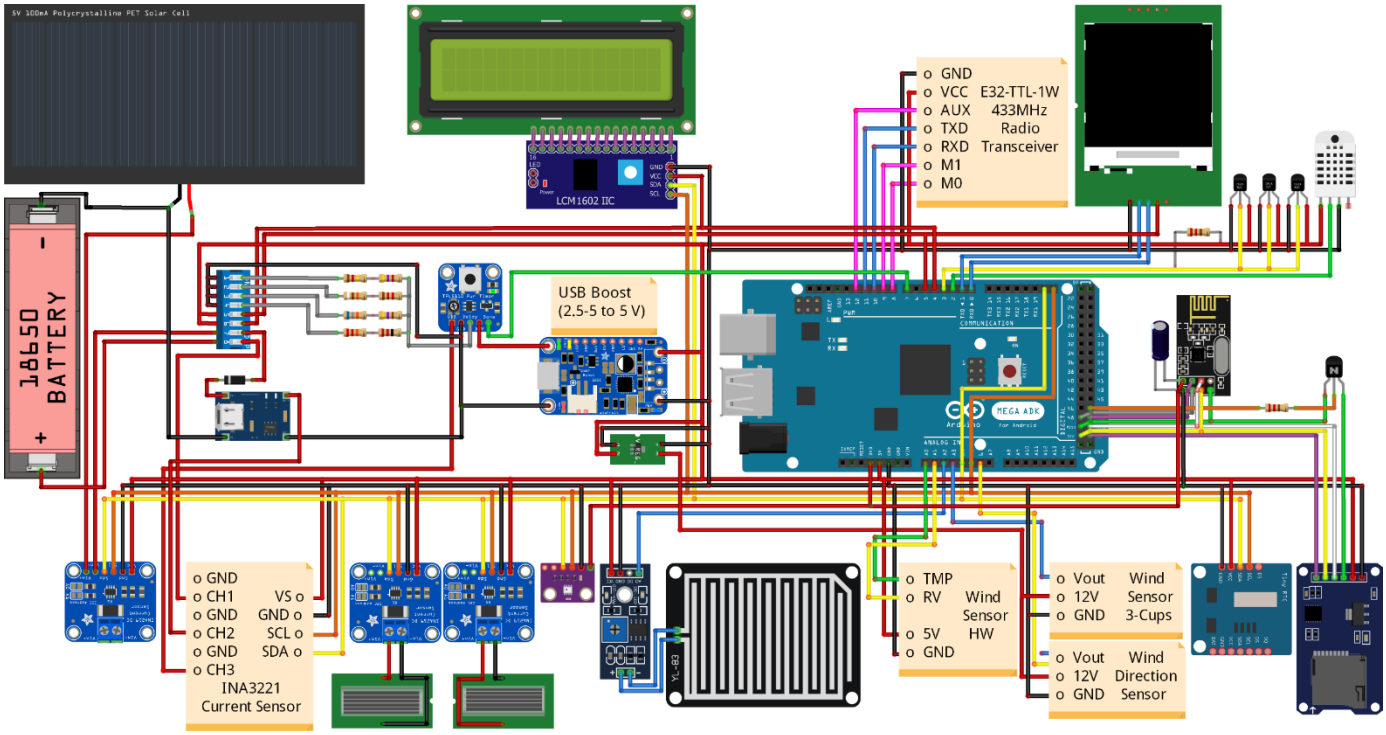


Figura 24. Esquema completo del dispositivo transmisor.

3.3.1. Etapa de alimentación

Empezando de izquierda a derecha, comenzamos primero con la **etapa de alimentación**. El dispositivo cuenta con un **panel solar** de 165x135 mm de 6 V i 3,5 W. La corriente que genera el panel pasa a través de un sensor de corriente INA219 que registra la tensión y la corriente del panel solar (**variables I_PV y V_PV**). Luego, pasa a través de uno de los canales de un interruptor tipo DIP-Switch de 8 canales que permite conectar y desconectar la alimentación por panel solar, y luego a un diodo tipo Schottky. Este diodo evita que la corriente de retorno de las baterías dañe el panel solar, a costa de producir una caída de unos 0,3 V por el valor de tensión directa del diodo. Finalmente, la corriente del panel solar llega al módulo de carga de baterías VMA-321 que, si la tensión del panel solar es superior a 4,5 V, la reduce hasta una tensión de 4,2 V con corriente constante y adecuada para la carga de las baterías.

Así pues, la salida del cargador se conecta al bus de alimentación. Este bus se conecta a diversos lugares: por un lado, a la salida del cargador, se llega a uno de los canales del sensor INA3221, que medirá esta tensión de alimentación (**variable V_BUS**) y la corriente de carga del panel solar (**variable I_CH**). Desde ahí (todos los canales del INA3221 están conectado entre ellos), un segundo canal se dirige a las **baterías**, al mismo tiempo que mide la corriente que entra y sale de ellas, dependiendo de si están cargándose o alimentando al circuito (**variable I_BAT**). Antes de llegar, la línea pasa por otro de los canales del interruptor para conectar y desconectar la alimentación. Las baterías consisten en 2 o 4 celdas de litio recargables en formato 18650 de 3,7 V de diferentes capacidades (entre 2000 y 3000 mAh cada una) y conectadas en paralelo.

De vuelta al bus de alimentación, el tercer canal del INA3221 se dirige al circuito del microcontrolador, con lo que mide la corriente de entrada al circuito, es decir, la corriente final que consume el dispositivo (**variable I_IN**). Pero antes de eso, pasa por el **temporizador** TPL5110, que actuará como interruptor, cortando el paso de esta corriente cuando el ciclo de medida haya terminado. Este temporizador permite programar el intervalo de tiempo que el dispositivo permanece en reposo mediante un potenciómetro o mediante un valor de resistencia conectado a uno de sus pines. El diseño del transmisor permite elegir entre 4 periodos de medida: 1 min, 15 min, 30 min y 1 hora (véase *Anexo VIII - Apartado 3: Soldadura del circuito electrónico* para más información). Este periodo se puede seleccionar mediante 4 de los canales del interruptor principal.

3.3.2. Etapa del microcontrolador

Tras pasar el temporizador, nos dirigimos al convertidor Boost DC-DC VMA-403, que nos transforma esta tensión variable proveniente de la etapa de alimentación a una tensión constante de 5 V. Esta será la tensión que, salvo ciertas excepciones, alimentará a todo el resto del circuito. Así pues, los 5 V entran por el pin 5V de la **placa Arduino MEGA**. Aunque este pin no está pensado con este propósito, la única diferencia con la alimentación por USB es que así se puentean los reguladores internos de la placa, lo cual no es un problema, ya que el convertidor Boost actúa como regulador.

A partir de aquí, la placa Arduino sirve como nodo principal para la conexión de todos los sensores y demás periféricos del circuito. La primera de las conexiones que se van a tratar es el **bus I2C**. A este bus se conectan una buena parte de componentes:

- El sensor de corriente INA3221 y tres sensores INA219: uno para el panel solar, y los otros dos para los sensores de radiación. Para estos dos, ambos cables de las células solares calibradas se conectan en cortocircuito a los pines de medida V+ y V- de cada INA219, de forma que pueden medir la corriente que generan (**variables IRR_UP e IRR_DOWN**).

- El sensor de presión BME280 (**variable PATM**), alimentado a partir de la salida de 3,3 V que ofrece el Arduino MEGA.

- Una pantalla LCD de 20x4 caracteres para la visualización de los datos medidos. Este elemento no se instala siempre, por lo que su pin de alimentación está conectado al bus de 5 V a través de uno de los canales del interruptor, por lo que se puede desactivar para ahorrar energía.

- Un módulo de reloj en tiempo real (RTC) para medir la fecha y la hora de las medidas en caso de que se guarden en la tarjeta SD.

Seguidamente, tenemos los **pines analógicos** del Arduino, a los que están conectados los siguientes elementos:

- El sensor de gotas de lluvia (**variable RAIN**), a través de su convertidor analógico-digital.

- Un sensor de viento de hilo caliente Rev. C (actualmente en desuso), que usa dos pines analógicos (**variable V_WIND_HW**).

- Por último, alimentados con 12 V a través del conversor DC-DC de 5 a 12 V, tenemos el anemómetro de copas (**variable V_WIND_CUP**), y un sensor de dirección de viento, una especie de veleta metálica que también ofrece una

salida analógica de 0 a 5 V (**variable DIR_WIND**). Aunque a día de hoy aún no se ha utilizado en Benicalap, el transmisor es compatible con este sensor de dirección en caso de que sea necesario.

Finalmente, veremos los distintos elementos conectados a los **pines digitales** de la placa Arduino:

- Pines 0 y 1: Aquí tenemos el puerto serie UART, que reservamos para la conexión de una pantalla tipo Nextion. Aunque a día de hoy no se ha añadido al desarrollo del programa, se ha preferido mantener la compatibilidad con este dispositivo. Al igual que en el caso de la pantalla LCD I2C, el pin de alimentación de esta pantalla también pasa a través del último de los canales del interruptor principal, por lo que puede desconectarse para ahorrar energía.

- Pin 2: Pin de comunicación del sensor de humedad y temperatura DHT22 (**variables HR_DHT y TEX_DHT**).

- Pin 3: Pin de Comunicaciones del bus 1-Wire para los tres sensores DS18B20. Cuenta con una resistencia de pull-up a 5 V para su correcto funcionamiento (**variables WBGT_DS y TEX_DS**, con un tercer conector disponible para la temperatura interior de la caja **TBOX_DS**, o para temperaturas de superficie, **TWALL1 y TWALL2**).

- Pin 4: Entrada digital conectada a la alimentación de la pantalla LCD para detectar cuándo está activado su interruptor de encendido y así ahorrar energía.

- Pin 5: Entrada digital conectada a la alimentación de la pantalla Nextion para detectar cuándo está activado su interruptor de encendido y así ahorrar energía.

- Pin 7: Salida digital para activar el temporizador TPL5110 y poner el dispositivo en estado de reposo.

- Pines 8-12: Pines de comunicación del módulo de radiofrecuencia 433 MHz E32-TTL-1W. Dos pines para puerto serie, dos para selección de modo y uno auxiliar.

- Pin 47: Salida digital para controlar un transistor NPN que active y desactive la línea CS del bus SPI, por motivos derivados de un funcionamiento anómalo del módulo de tarjeta SD.

- Pines 48 y 49: Pines digitales para comunicación del módulo de radiofrecuencia nRF24L01 de 2,4 GHz, alimentado por la salida de 3,3 V de la placa Arduino, y con un condensador electrolítico de 10 uF para regular la tensión a la entrada.

- Pines 50-53: Pines de comunicación del bus SPI para el módulo de tarjeta SD para el guardado local de datos.

3.3.3. Placa de circuito impreso

Una vez decidido el circuito electrónico que controlará el dispositivo, se diseñó mediante la herramienta online EasyEDA una placa de circuito impreso (PCB) que incluya todos los elementos anteriormente mencionados y/o sus conectores en un espacio compacto. Se llegaron a diseñar un total de dos versiones. Puesto que se trata de un proyecto en constante evolución, a partir de que se desarrollaran mejoras y actualizaciones, esta PCB requiere de ciertas modificaciones durante el montaje, pero en esencia sigue siendo válida (véase *Anexo VIII - Apartado 3: Soldadura del circuito electrónico* para más información).

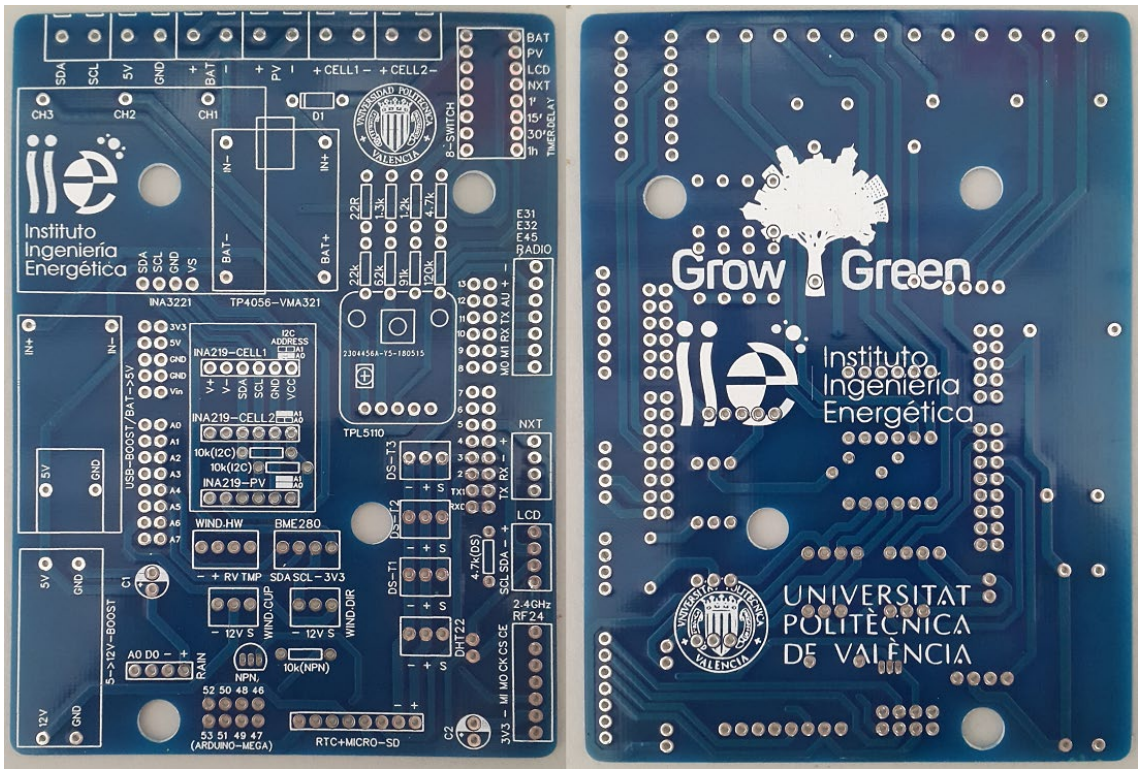


Figura 25. Placa de circuito impreso del transmisor.

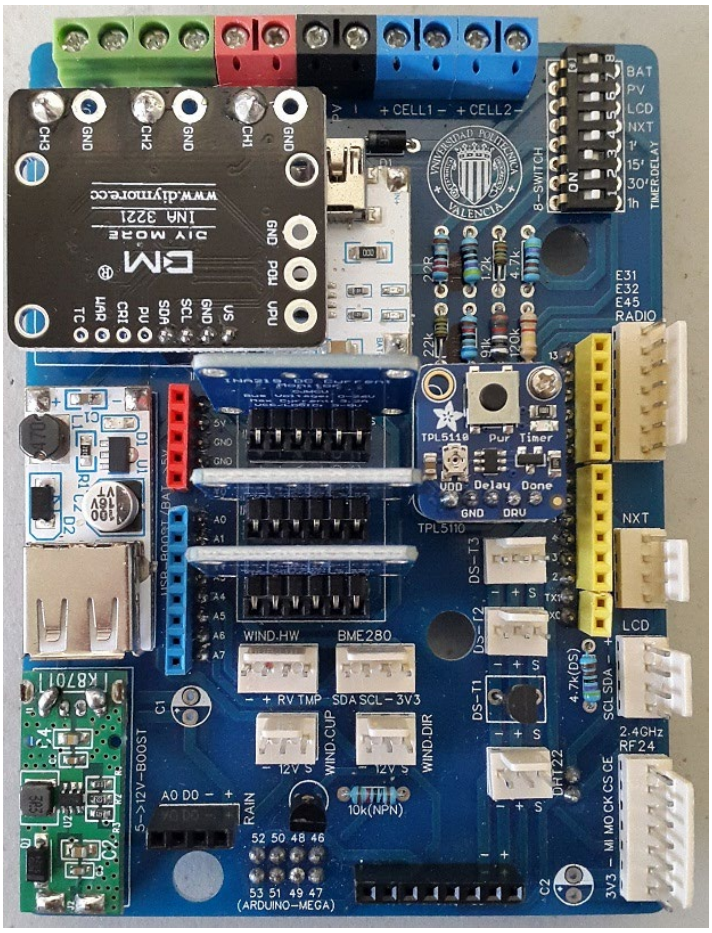


Figura 26. Placa de circuito impreso del transmisor soldada.

3.3.4. Circuito simplificado

Como se habrá podido comprobar, el circuito electrónico que se ha diseñado es compatible con una gran cantidad y variedad de periféricos y sensores. Sin embargo, no todos son necesarios para el funcionamiento habitual del transmisor, y algunos quedan reservados para tareas de desarrollo o como opcionales. En la *Figura 27*, se puede comprobar cómo quedaría el esquema del transmisor conectando tan sólo los elementos imprescindibles (véase *Anexo III* para ver el esquema a tamaño completo).

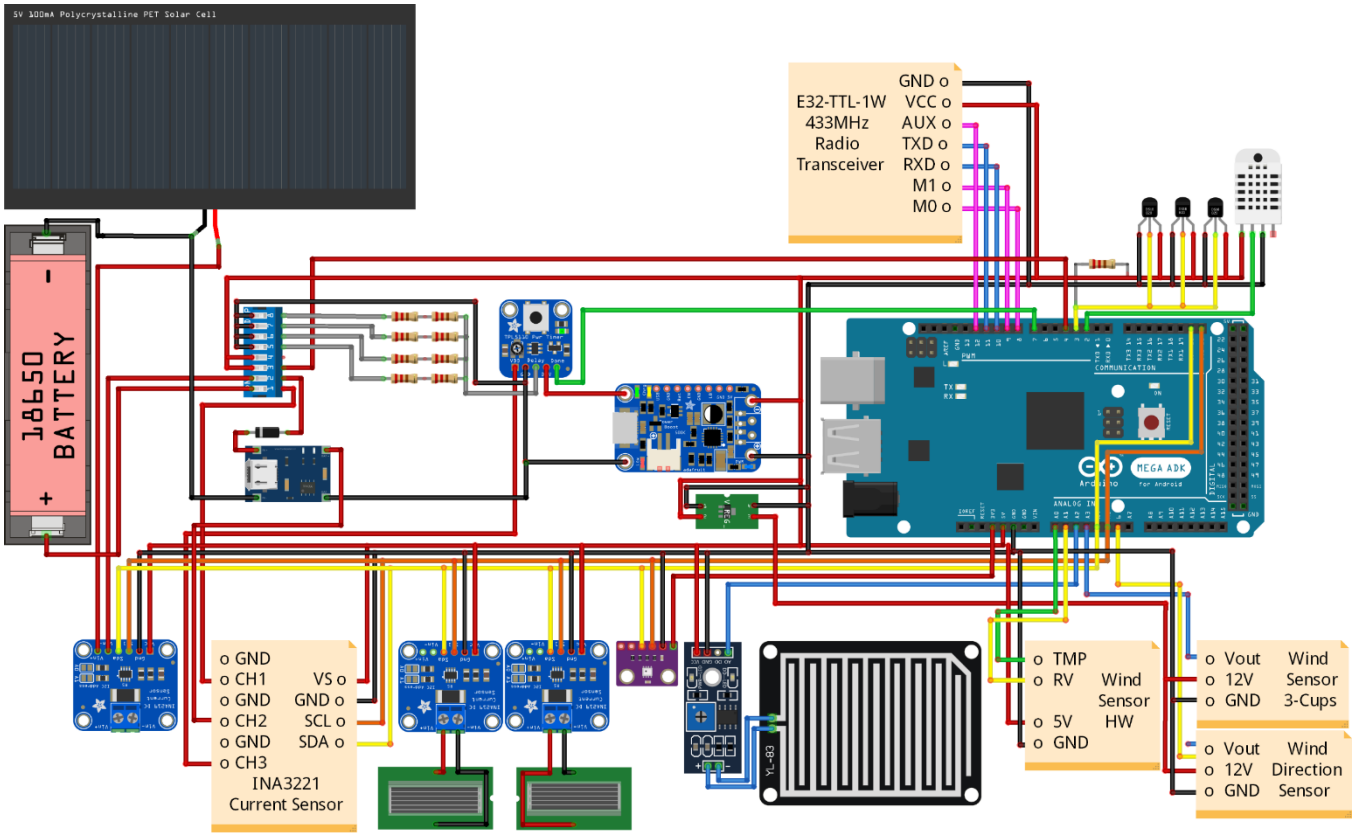


Figura 27. Esquema simplificado de dispositivo transmisor.

Algunos de los elementos de los que se prescinde son: las dos pantallas LCD y Nextion, ya que la visualización de datos directamente en la nube es más sencilla en inmediata; el módulo de radiofrecuencia nRF24L01, ya que finalmente se decidió usar la tecnología de 433 MHz; y los módulos RTC y SD, que no son necesarios con el sistema de guardado en la base de datos.

Además, algunos sensores, como el sensor de dirección de viento, el INA219 que mide el panel solar, el sensor de viento de hilo caliente o el tercer sensor DS18B20 no se usan habitualmente, o han caído en desuso. Además, en cada transmisor se montarán los sensores necesarios para cada ubicación, por lo que, en algunos casos, habrá otros que tampoco se conectarán (véase *Apartado 5: Instalación del sistema GrowGreen Valencia* para más información).

Esta versión simplificada del transmisor tiene una característica interesante: puesto que usa menos pines y recursos que la versión completa, es completamente compatible con un Arduino UNO, una placa más compacta y económica que la MEGA.

3.4. Diseño estructural

Tras haber diseñado toda la electrónica, es necesario desarrollar una estructura que le dé soporte y protección adecuada para cumplir las especificaciones. El formato que se ha elegido es el de una caja impresa en 3D en plástico PLA y diseñada a medida para encajar con todos los sensores. Además, generalmente, la parte exterior de las cajas estarán pintadas con pintura blanca y barniz transparente e identificadas cada una por un número. Para más detalles sobre la estructura impresa en 3D y el resto de elementos mecánicos, véase *Anexo VIII – Apartado 2: Ensamblaje de la estructura.*

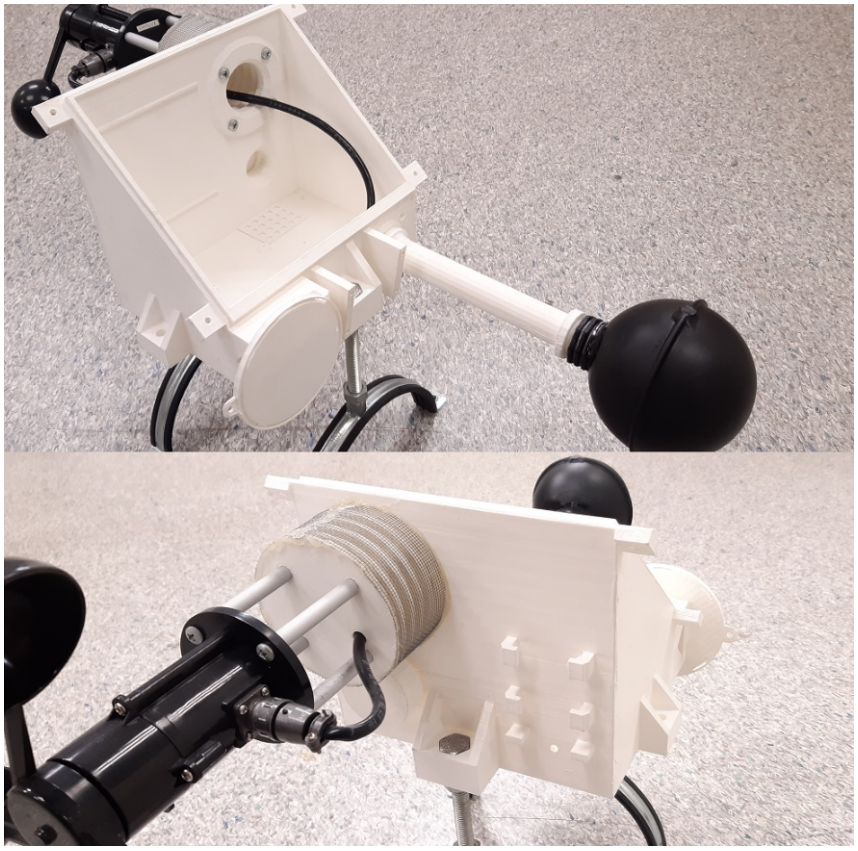


Figura 28. Caja del transmisor montada sin electrónica.

La caja tiene una forma semejante a un prisma triangular para aprovechar la inclinación y mejorar la recepción de luz del panel solar. En un rellano horizontal en la parte superior, se encuentran la antena de radiofrecuencia, los sensores de radiación solar, lluvia y una sonda de temperatura DS18B20 ubicada dentro de un globo metálico negro, en el extremo de un pequeño poste de plástico, que se encargará de medir la temperatura de globo negro. En la parte inferior, dentro de una pantalla protectora de radiación para sensores impresa en 3D, se encuentran el sensor DHT22, junto a otra sonda DS18B20 y el sensor de presión BME280. Además, en la versión completa, montan en la parte inferior el anemómetro de copas y un segundo sensor de radiación.

Por otro lado, las baterías estarán alojadas en un portapilas pegado a las paredes interiores de la caja, mientras que el panel solar estará encajado en la tapa. Todos los sensores exteriores están cableados de tal forma que las oberturas para llegar al interior se encuentran en puntos que no ponen en riesgo la integridad de la estructura. Además, se sellarán todas estas oberturas para mejorar la estanqueidad del conjunto (véase *Apartado 6.1: Integridad estructural y durabilidad* para más información). Finalmente, todo el conjunto se podrá colgar en su ubicación final con dos tornillos metálicos largos y dos abrazaderas metálicas circulares de diferentes tamaños.

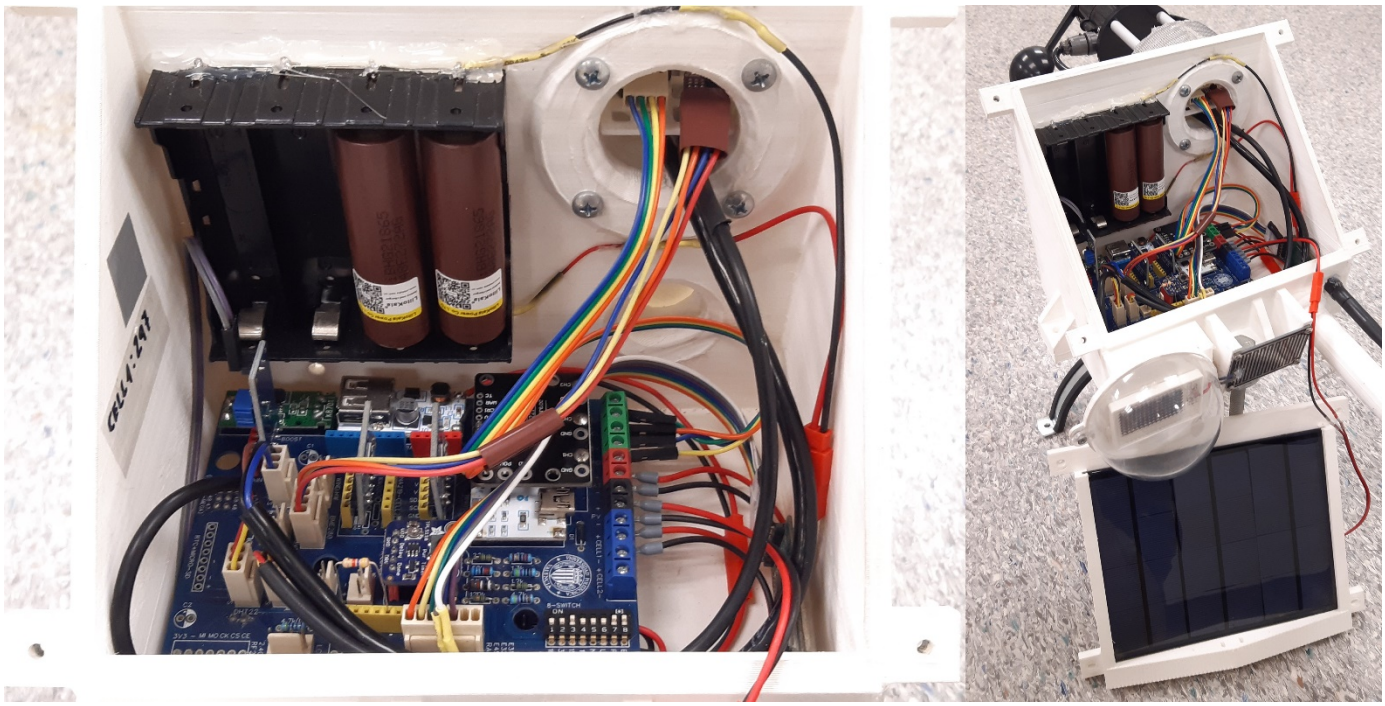


Figura 29. Interior de la caja de un transmisor con electrónica montada.

3.5. Programación del transmisor

Finalmente, tras haber detallado todo el hardware del transmisor, es el momento de hablar de su programación. El transmisor está programado usando la plataforma de Arduino, y consta de un único programa escrito en lenguaje C y contenido en el archivo "*Transmitter_4.2_MEGA_E32.ino"*, además del archivo de variables de programa *"defines.h"*. Para obtener más información acerca del proceso concreto de configuración y carga del programa, véase *Anexo VIII – Apartado 4: Programación del dispositivo*. En esta memoria, se va a explicar el contenido del programa, detallando cada una de las partes mediante el diagrama de flujo de la *Figura 31*. Así pues, se compone de las siguientes partes (véase *Anexo IV* para acceder al código completo):

- **Declaración de librerías:** A continuación, se muestra una lista de las librerías utilizadas en el programa y los dispositivos que las necesitan:

Librería	Dispositivo asociado
DHT.h	DHT22
OneWire.h	DS18B20
DallasTemperature.h	BME280
Adafruit_Sensor.h	BME280
Wire.h	LCD, INA219, INA3221
LiquidCrystal_I2C.h	LCD
Adafruit_INA219.h	INA219
SDL_Arduino_INA3221.h	INA3221
SoftwareSerial.h	E32-TTL-1W
RTClib.h	RTC
SD.h	microSD
SPI.h	microSD

Figura 30. Tabla de librerías utilizadas en el programa del transmisor.

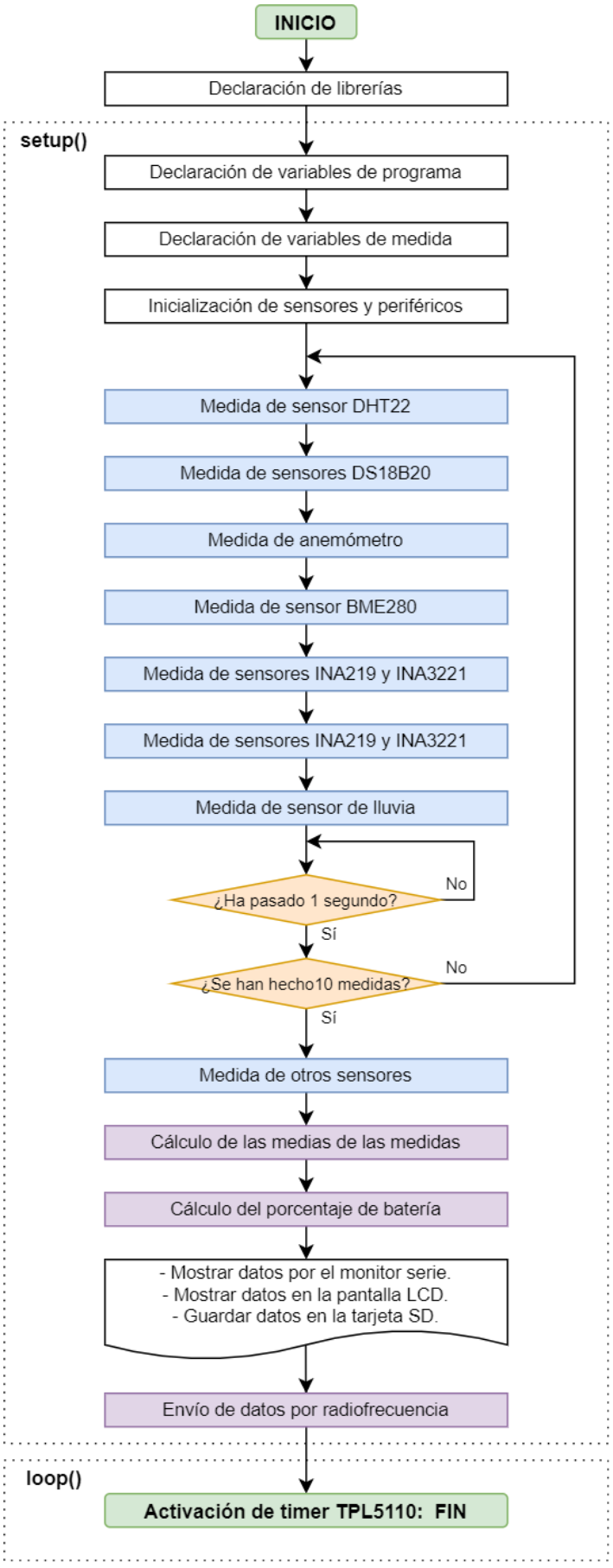


Figura 31. Diagrama de flujo del programa del transmisor.

- **Declaración de variables:** Creación de las variables e instancias necesarias para el funcionamiento de las librerías. Las variables de medida son vectores de diferentes tipos para cada una de las variables y del mismo tamaño que el número de iteraciones de medida que se van a realizar (por defecto, 10). Además, otra variable para guardar la media de las medidas.

- **Inicialización de sensores y periféricos:** Se llaman a las funciones de las librerías para configurar y preparar los sensores para las medidas. Se crea el archivo para guardar los datos en la SD en caso de que sea necesario.

- **Bucle de medida:** A partir de aquí se inicia un bucle de medida en el que se consultan cada uno de los sensores que estén habilitados (indicándolo en archivo de variables *"defines.h"*) cada segundo durante 10 iteraciones y se guardan esos valores en los vectores que hemos creado anteriormente. En caso de que algún sensor dé error en la medida o no esté conectado, se guarda el valor de error "32767".

- **Medidas de otros sensores:** Otros sensores, como el sensor de viento de hilo caliente, el sensor de dirección de viento, u otros con los que pudiera ser compatible el transmisor, necesitan tomar las medidas una sola vez o tras haber esperado un tiempo determinado de preparación.

- **Cálculo de las medias de las medidas:** Usando la función propia CalcularMedia(), calculamos la media aritmética de las medidas que hemos guardado en los vectores para eliminar la posible variabilidad que puedan presentar los sensores. Y luego, las multiplicamos por 100 para convertir los valores decimales en valores enteros, lo que permite transmitir las variables usando valores enteros de 8, 16 o 32 bits.

- **Cálculo del porcentaje de batería:** Mediante una fórmula que hemos obtenido experimentalmente, calculamos el porcentaje de carga de las baterías a partir de su tensión (véase *Apartado 6.5: Autonomía* para más información).

- **Mostrar datos:** Si está habilitado, se muestran los datos en el monitor serie y en la pantalla LCD, mostrando los datos de 4 en 4 y esperando unos segundos entre cada tanda. Además, se guardarían los datos en el archivo de la tarjeta SD.

- **Envío de datos por radiofrecuencia:** Finalmente, se envían los datos al módulo de radiofrecuencia. Para ello, se crea un vector de bytes con un formato concreto en que se envía cada variable una detrás de la otra, y cada variable ocupando un número concreto de bytes. Después, se envía este vector por puerto serie al módulo de radiofrecuencia. Se envían todos los datos juntos para ahorrar tiempo de ejecución y para evitar la fragmentación de los datos por interferencias. El receptor se programará de tal forma que reconocerá el formato de este vector para reconstruir las variables que ha medido el transmisor.

  En la *Figura 32*, se muestra una tabla con la composición de este vector, con cada variable, su tipo, su tamaño y las diferentes conversiones que se le aplican a cada una tanto en el transmisor como en el receptor para enviar los datos correctamente y mantener el mismo formato de decimales. Obsérvese que al final del vector hay dos variables, IR_T y GAS, que corresponden a un sensor de temperatura de infrarrojos y un sensor de monóxido de carbono. Estas variables se incluyeron en la fase de diseño y, aunque finalmente estos sensores no llegaron a usarse, se mantuvieron las variables para usar el mismo formato de vector en caso de necesitar añadir variables adicionales más adelante.

Nombre	Tipo en TX	Bytes	Conversión TX	Tipo en BD	Conversión RX
address	byte[2]	2	.	.	.
channel	byte	1	.	.	.
TX_ID	uint8_t	1	.	int	.
BAT_LV	uint8_t	1	.	int	.
DHT_HR	uint16_t	2	.	float	/100
DHT_TEX	int16_t	2	.	float	/100
DS_T1	int16_t	2	.	float	/100
DS_T2	int16_t	2	.	float	/100
DS_T3	int16_t	2	.	float	/100
WIND_HW	uint16_t	2	.	float	/100
WIND_CUP	uint16_t	2	.	float	/100
WIND_DIR	uint8_t	1	.	String	int a String
PATM	int32_t	4	/100	int	.
V_BUS	int16_t	2	.	float	/100
V_PV	int16_t	2	.	float	/100
I_BAT	int32_t	4	.	float	/100
I_CH	int32_t	4	.	float	/100
I_IN	int32_t	4	.	float	/100
I_PV	int32_t	4	.	float	/100
IRR_UP	int32_t	4	.	float	/100
IRR_DOWN	int32_t	4	.	float	/100
RAIN	uint8_t	1	/100	int	.
IR_T	uint16_t	2	.	float	/100
GAS	uint16_t	2	.	int	.
	Total	57			

Figura 32. Tabla de formato de variables enviadas por el transmisor.

- **Activación de timer TPL5110:** Tras un tiempo de espera para completar el envío de los datos, se activa una salida digital que se dirige a uno de los pines del timer, que corta la alimentación del circuito, por lo que entra en estado de reposo durante el tiempo que se haya marcado en el selector hasta el siguiente ciclo de medida.

4. Diseño del sistema: Receptor y Base de Datos

4.1. Descripción general del dispositivo

El dispositivo receptor se encarga de recibir los paquetes de datos procedentes de los transmisores. Consiste en un módulo de radiofrecuencia conectado a un microcontrolador Arduino UNO que traduce la trama de bytes recibidos por radio en una cadena de caracteres que se enviará por USB a un computador Raspberry Pi.

La Raspberry Pi cuenta con una distribución Linux basada en Debian y un módem 3G con una tarjeta SIM conectado por USB. A través de una serie de scripts Python, el receptor se conecta a la red y envía una petición al servidor de Plesk para escribir en una base de datos MySQL alojada en la nube. Además, el receptor cuenta con una pantalla táctil para configurar el dispositivo cómodamente. También supervisa automáticamente su correcto funcionamiento en todo momento, monitorizando la red y reiniciándose periódicamente para asegurar que se pierden el menor número de medidas posible.

El receptor se encuentra instalado en una caja de conexiones eléctricas anclada en la pared con tornillos. Está alimentado con una fuente de alimentación de 5 V y 2 A con conector micro-USB, y cuenta con una antena externa de largo alcance preparada para funcionar en exteriores.

Figura 33. Receptor montado y funcionando en su caja.

4.2. Tecnologías utilizadas: Hardware y software

A continuación, se ofrece una lista detallada de los principales componentes y sensores utilizados en el diseño del dispositivo receptor junto a una breve descripción:

- **Placa Arduino UNO:** Placa de desarrollo basada en el microcontrolador de 8 bits ATMega328P, con capacidad para 14 GPIO, 6 entradas analógicas, puertos serie, conector USB y compatibilidad con protocolos de comunicación serie como SPI e I2C.

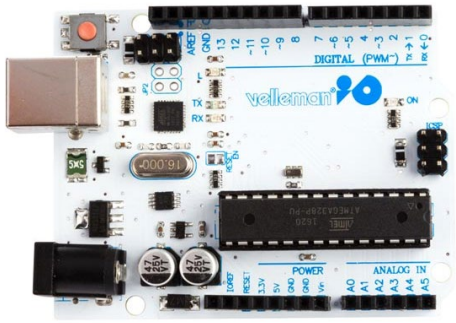


Figura 34. Placa de desarrollo Arduino UNO.

- **Módulo de radiofrecuencia E32-TTL-1W:** Módulo de comunicaciones por radiofrecuencia a 433 MHz E32-TTL-1W. Con interfaz de comunicación por UART y antena de largo alcance (alrededor de 7500 m).

Figura 35. Módulo de radiofrecuencia E32-TTL-1W

- **Raspberry Pi 3 Modelo B:** Computador compacto basado en un procesador de 64 bits de 1,2 GHz, 1 GB de memoria RAM, compatible con Wifi y Bluetooth, con conexiones USB, Ethernet, HDMI, entre otras, y con sistema operativo Linux.

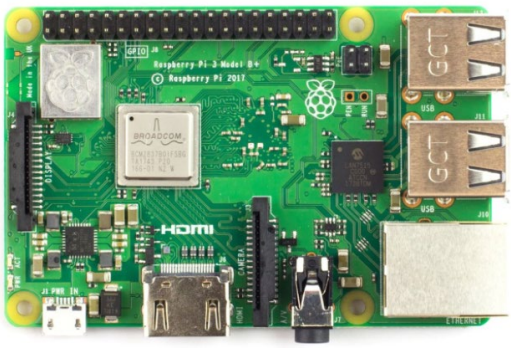


Figura 36. Raspberry Pi 3 Modelo B.

- **Módem USB Huawei E1750:** Módem USB 3G compatible con redes GSM y GPRS para acceso a llamadas, SMS y datos a través de internet, y con un slot para tarjeta SIM.

Figura 37. Módem USB Huawei E1750.

- **Pantalla táctil LCD de 7":** Pantalla táctil LCD compatible con Raspberry Pi con placa de conversión y conector FDC.

Figura 38. Pantalla táctil LCD para Raspberry Pi.

Y en cuanto a software, ahora se ofrece una lista de las principales plataformas de desarrollo, lenguajes, protocolos de comunicación, etc., utilizados en el diseño del dispositivo transmisor junto a una breve descripción de cada uno:

- **Arduino (lenguaje C):** Plataforma de desarrollo de código abierto para microcontroladores de la familia AVR, como ATMega, aunque es compatible con una gran variedad de hardware. En ese caso, la programación se ha realizado utilizando el lenguaje C.

- **Comunicación serie UART:** Protocolo de comunicación serie denominado "Transmisor-Receptor Asíncrono Universal" para la comunicación entre dispositivos utilizado, por ejemplo, en la interfaz USB.

- **NOOBS:** Distribución de software personalizada para Raspberry Pi que permite la instalación de sistemas operativos Linux basados en Debian, entre otros, de forma sencilla y cómoda.

- **Python:** Lenguaje de programación interpretado multiplataforma, especialmente eficaz en el tratamiento de cadenas de caracteres.

- **Plesk:** Software de alojamiento web escrito en lenguaje PHP, apto para la gestión de bases de datos de tipo MySQL en servidores web.

- **PHP:** "PHP: Hypertext Preprocessor". Es un lenguaje de programación utilizado principalmente en desarrollo web.

- **SQL: "**Structured Query Language". Es un lenguaje de programación ideado para realizar operaciones sobre bases de datos. Se utiliza para realizar peticiones, escribir, modificar, exportar y, en general, gestionar las bases. MySQL es el conjunto de aplicaciones que usa el lenguaje SQL para crear y manejar las bases de datos.

4.3. Descripción del circuito electrónico

Ahora que se han descrito los principales componentes y tecnologías utilizadas en este diseño, se procederá a describir el circuito electrónico que controla el dispositivo receptor y sus distintas conexiones. Así pues, empezamos con una visión general del esquema del circuito.

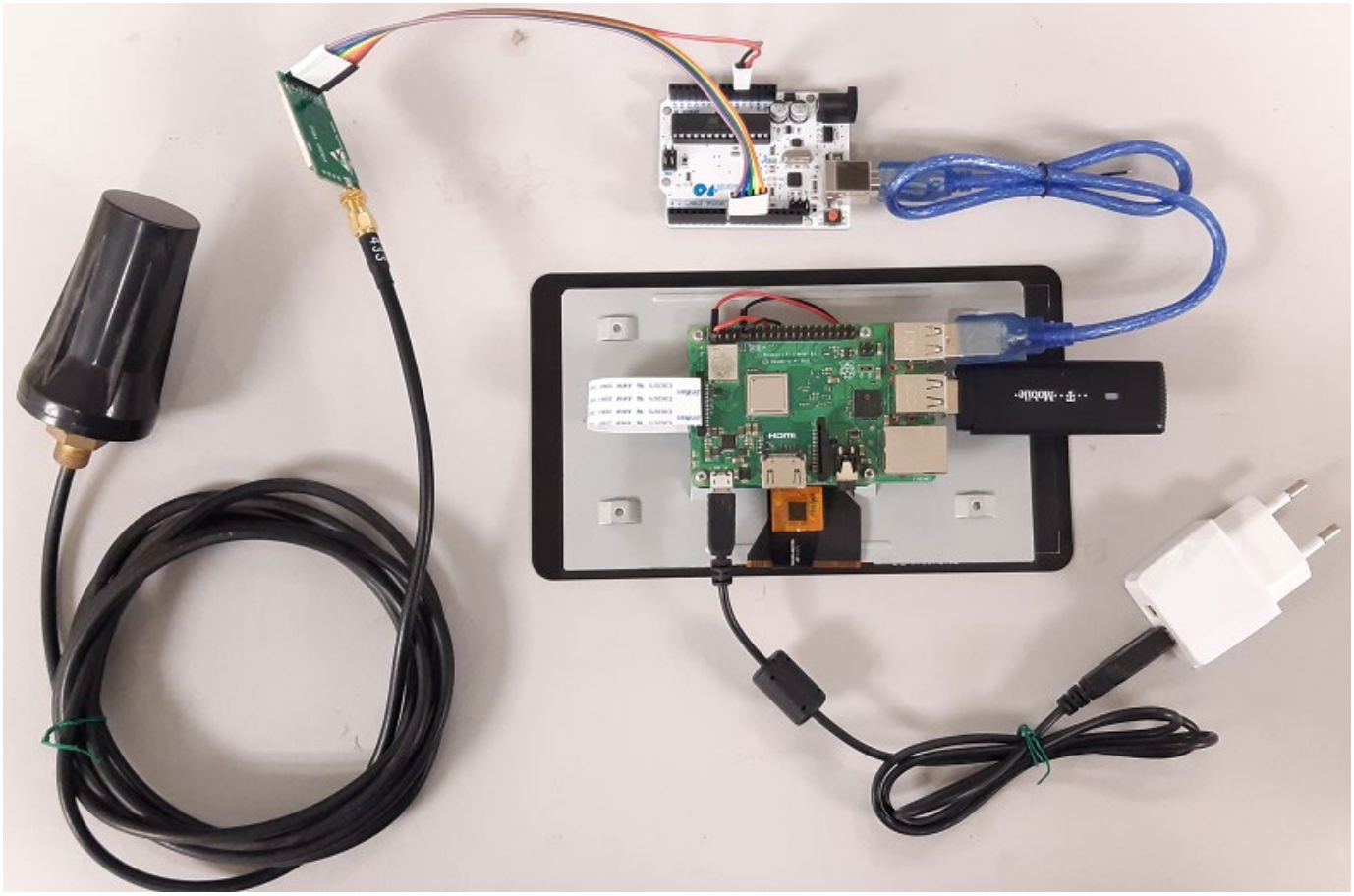


Figura 39. Conexionado del dispositivo receptor.

Primero, nos encontramos con el módulo de radiofrecuencia E32.TTL-1W, utilizado también en el transmisor. Este módulo se encuentra conectado a la placa Arduino UNO siguiendo el esquema de la *Figura 40*. Por otro lado, tiene conectada una antena de radio de 433 MHz de larga distancia con un encapsulado adecuado para trabajar en exteriores.

Seguidamente, la placa Arduino UNO está conectada a la Raspberry Pi a través de un cable USB-A a USB-B a uno de sus puertos USB. A su vez, la Raspberry Pi se acoplará a la pantalla táctil LCD siguiendo las instrucciones del fabricante. A continuación, tenemos el módem 3G, que está conectado a otro de los puertos USB de la Raspberry Pi. Para configurar la Raspberry Pi, se puede usar la pantalla táctil, aunque también es recomendable usar un teclado inalámbrico, cuyo receptor se conectaría a uno de los puertos USB que queden libres.

Finalmente, todo el conjunto se conectaría a la red eléctrica usando una fuente de alimentación de 5 V y, al menos, 2 A, con terminación micro-USB, para conectarla al puerto micro-USB de alimentación de la Raspberry Pi.

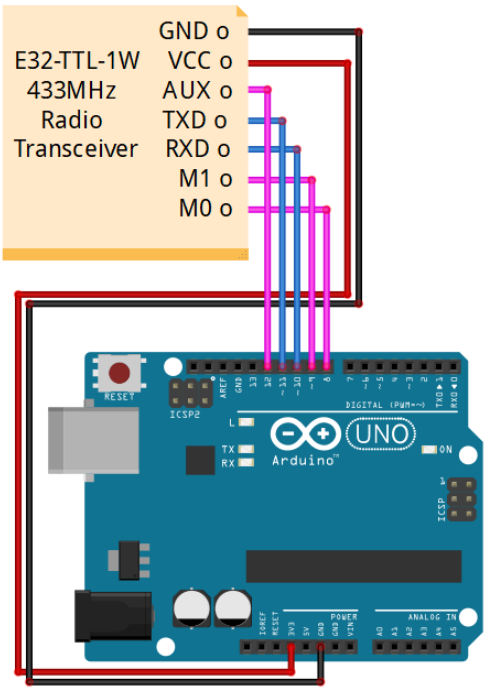


Figura 40. Esquema de conexionado del módulo E32-TTL-1W en Arduino UNO.

4.4. Programación del receptor

4.4.1. Programación Arduino

Finalmente, tras haber detallado todo el hardware del receptor, es el momento de hablar de su programación. El receptor necesita distintos tipos de configuración y programación. Por un lado, tenemos la placa Arduino UNO, que está programada usando la plataforma de Arduino, y consta de un único programa escrito en lenguaje C y contenido en el archivo "*Receiver_Rasp_4.2_E32.ino*". Ahora procedemos a explicar el contenido del programa, detallando cada una de las partes mediante el diagrama de flujo de la *Figura 41*. Así pues, se compone de las siguientes partes (véase *Anexo V* para acceder al código completo):

- **Declaración de librerías:** El receptor sólo necesita la librería "SoftwareSerial.h" para manejar la comunicación por puerto serie del módulo de radiofrecuencia.

- **Declaración de variables:** Creación de las variables e instancias necesarias para el funcionamiento del módulo de radiofrecuencia. Se declaran e inicializan a 0 las variables de medida. Serán variables enteras de diferentes tamaños para almacenar los valores de los sensores provenientes del transmisor, además del String para guardar los datos.

- **Configuración del módulo de radiofrecuencia:** Configuración del módulo de radio y del puerto serie.

- **Lectura de datos:** Si el buffer de la antena está disponible, se empiezan a leer secuencialmente los bytes de la trama recibida y se convierten en variables según el formato descrito en la *Figura 32*, siguiendo el camino inverso que en el transmisor.

- **Montaje de la cadena de caracteres:** Se crea una cadena de caracteres concatenando las variables guardadas anteriormente y realizando las conversiones adecuadas descritas en la *Figura 32*. Así, se recuperan los formatos originales de cada variable, ya sea números enteros, decimales o caracteres, dándole el nombre adecuado a cada variable para que sea entendido en las siguientes etapas del proceso.

- **Envío de datos por puerto serie:** Se envía la cadena de caracteres por el puerto serie, que pasará a ser leída por la Raspberry Pi. Después, se vacía la cadena y se espera durante 10 segundos para luego comprobar si se ha recibido por radio el próximo paquete de datos.

  Este es un ejemplo de la cadena de caracteres que crea este programa:

```
ID=1&BAT=100&DHTH=60.82&DHTT=18.87&T1=24.84&T2=19.40&T3=32.44&WHW=0.00&WCUP=0.
54&WDIR=0&PATM=101018&VBUS=4.04&VPV=4.69&IBAT=208.68&ICH=40.15&IIN=250.27&IPV=
39.23&IRRU=344.92&IRRD=40.62&RAIN=0&IRT=32767&GAS=32767
```

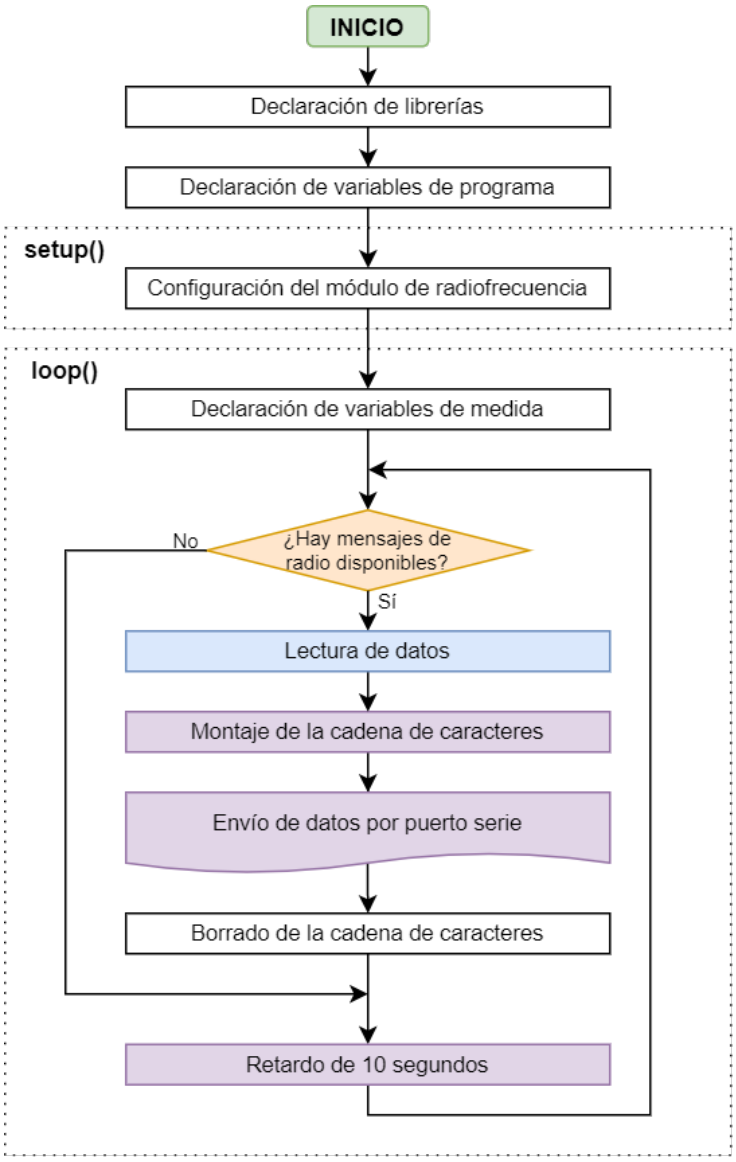


Figura 41. Diagrama de flujo del programa de Arduino del receptor.

4.4.2. Programación Raspberry Pi

La programación de la Raspberry Pi se realiza sobre el sistema operativo Linux, en una distribución basada en Debian, instalado a partir de la herramienta NOOBS. Una vez realizada la instalación del sistema operativo, se debe configurar el sistema para nuestro propósito. Así pues, la programación consta de distintos archivos y scripts, siendo los más importantes los que se describen a continuación:

- **Crontab:** Este es uno de los archivos más importantes. Se trata de un comando de Linux que nos permite ejecutar tareas periódicamente. En nuestro caso, se ha programado para que al inicio configure automáticamente el módem 3G para acceder a la red y que inicie el archivo "conexionrasp.py".

  Además, cada monitoriza la temperatura del equipo cada 5 minutos y la guarda en otro archivo. Mediante el archivo "controllog.py", también lee cada 32 minutos el registro de medidas, y si no se ha recibido ninguna medida en los últimos 30 minutos, reinicia el sistema. Finalmente, reinicia automáticamente el sistema cada día a las 0 horas.

- **conexionrasp.py:** Script de Python que lee los datos del puerto serie procedentes del Arduino UNO, guarda los datos en un archivo de registro "log.csv" y crea la petición para enviar los datos al servidor de Plesk. En el *Anexo VI*, se puede encontrar el código completo comentado con detalle. Sin embargo, hay un aspecto importante que vale la pena destacar de este archivo. En él, podemos encontrar la siguiente línea de código:

  ```
  peticion = 'https://growgreenvlc.webs.upv.es/conexionrasp.php?'
  ```

  Esta línea de código es la cabecera de la petición al servidor de Plesk, y hace referencia a un archivo PHP alojado en el servidor: "conexionrasp.php". Este archivo tiene una gran importancia y se comentará con más detalle en el *Apartado 4.5: Base de datos*.

4.5. Base de datos

4.5.1. Servidor Plesk. Creación de la Base de Datos

Tras todo el sistema de medición y transmisión de datos, el último elemento restante del sistema es la Base de Datos. Para ello se ha creado una página web alojada en servidores de Plesk, que nos permiten crear la infraestructura necesaria para gestionar los datos. En la *Figura 42*, se puede ver la interfaz de usuario del panel web de Plesk. La base de datos que permite gestionar Plesk es de tipo MySQL, por lo que usaremos peticiones SQL para la escritura y la gestión.

Los datos se organizarán en tablas, teniendo una tabla diferente para cada transmisor. Sin embargo, hay un factor a tener en cuenta. Los transmisores tienen programados unos números identificadores, habitualmente, del 1 al 15, que indican la ubicación en la que están instalados en Benicalap. El problema surge durante las pruebas de los dispositivos tras la fabricación en la UP, ya que éstos estarían generando datos que no se han medido en la ubicación esperada, por lo que se estarían mezclando datos no deseados. Para resolver este problema, se van a crear dos grupos de tablas: HSMx y HSMUPVx. Estos grupos mostrarían las medidas tomadas por cada transmisor (HSM1, HSM2…) en Benicalap y en la UPV por separado.

Figura 42. Interfaz de usuario del panel de Plesk.

Así pues, se van a crear 30 tablas (HSM1 a HSM15, y HSMUPV1 a HSMUPV15), usando la siguiente petición SQL, en que cada línea es una variable diferente:

```
CREATE TABLE `HSM1` (
  `Input` int(11) NOT NULL COMMENT 'Entradas',
  `Receptor` int(11) NOT NULL COMMENT 'Equipo receptor',
  `Tx_id` int(11) NOT NULL COMMENT 'Equipo de emision',
  `Date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT 'Fecha y Hora',
  `BAT_LV` float NOT NULL COMMENT '%',
  `HR_DHT` float NOT NULL COMMENT '%',
  `TEX_DHT_C` float NOT NULL COMMENT '° C',
  `WBGT_DS_C` float NOT NULL COMMENT '° C',
  `TX_DS_C` float NOT NULL COMMENT '° C',
  `TBOX_DS_C` float NOT NULL COMMENT '° C',
  `V_WIND_HW` float NOT NULL COMMENT 'ms',
  `V_WIND_CUP` float NOT NULL COMMENT 'ms',
  `DIR_WIND` varchar(3) NOT NULL COMMENT 'Direccion del viento',
  `PATM_Pa` float NOT NULL COMMENT 'Pa',
  `V_BUS_V` float NOT NULL COMMENT 'V',
  `V_PV_V` float NOT NULL COMMENT 'V',
  `I_BAT_mA` float NOT NULL COMMENT 'mA',
  `I_CH_mA` float NOT NULL COMMENT 'mA',
  `I_IN_mA` float NOT NULL COMMENT 'mA',
  `I_PV_mA` float NOT NULL COMMENT 'mA',
  `IRR_UP_Wm2` float NOT NULL COMMENT 'Wm2',
  `IRR_DOWN_Wm2` float NOT NULL COMMENT 'Wm2',
  `RAIN` float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

4.5.2. Escritura en la Base de Datos

Una vez hemos creado todas las tablas, sólo nos queda empezar a escribir los datos, y tendremos que ser capaces de indicar a qué grupo de tablas queremos enviarlos. Para ello, hemos creado dos scripts PHP: “conexionrasp.php” y “conexionbeni.php”. Estos archivos, que ahora se explicarán con detalle, realizan las peticiones SQL necesarias para escribir los datos, y cada uno de ellos los envía a un grupo de tablas diferente, entre HSMx y HSMUPVx.

Así pues, la encargada de ejecutar estos archivos será la Raspberry Pi, a través del script “conexionrasp.py”. Como ya hemos visto anteriormente, este archivo realiza una petición a los servidores de Plesk, y en la cabecera de esa petición se hace referencia a este archivo PHP. Para escoger en qué tablas queremos escribir los datos, basta con cambiar este nombre de archivo, dependiendo de qué receptor estemos configurando.

Así, un receptor que se encuentre en la UPV para realizar pruebas, enviará la petición al archivo “conexionrasp.php”, mientras que el receptor que se instale en Benicalap usará el archivo “conexionbeni.php”. Esto significa que un transmisor con ID 1 recién terminado de fabricar en la UPV, enviaría sus datos a la tabla HSMUPV1. Cuando ese transmisor se lleve a Benicalap y se instale, en el momento en que entre en el rango del receptor de Benicalap, sus medidas empezarán a registrarse automáticamente en la tabla HSM1, con lo que no habremos ensuciado la base de datos con medidas incorrectas.

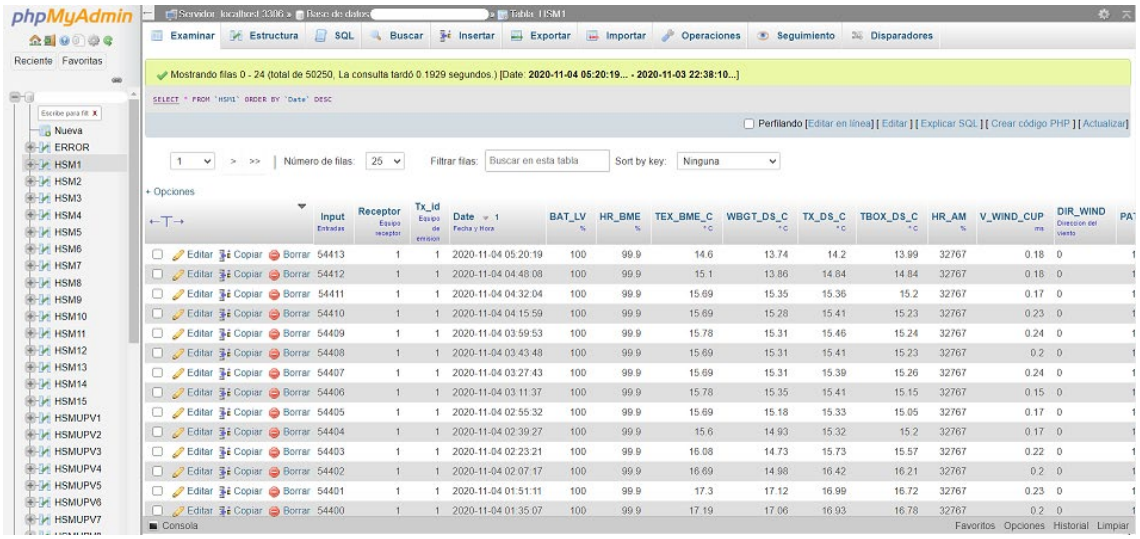


Figura 43. Lista de tablas en la base de datos.

Una vez aclarado este punto, sólo nos queda describir cómo funciona el archivo “conexionbeni.php”. En el *Anexo VII*, se puede encontrar el código completo y comentado. Básicamente, el script realiza las siguientes acciones.

- **Conexión a la Base de Datos:** Introduce las credenciales necesarias para acceder a la Base de Datos.

- **Lectura de variables:** A partir de la petición que ha llamado al archivo, extrae los valores de las medidas y las guarda en unas variables locales.

- **Filtrado de datos:** Comprobando ciertos valores, el archivo es capaz de filtrar paquetes de datos que no sean correctos y están cortados debido a interferencias en la transmisión por radio. Cuando detecta uno de estos paquetes, lo registra en una nueva tabla que hemos creado llamada “ERROR”. Esto nos permite tener un registro de medidas erróneas para saber cuándo y en qué transmisor se produjeron.

- **Escritura en la base de datos:** Finalmente, dependiendo del ID del transmisor, realiza la petición SQL para escribir los datos en la tabla correspondiente, ya sea HSMx o HSMUPVx, según el archivo al que se haya llamado.

5. Instalación del sistema GrowGreen Valencia

Una vez definido el diseño final del sistema, se ha procedido a su instalación en el barrio de Benicalap. Actualmente, el sistema consta de 11 transmisores y un receptor. Cada una de las ubicaciones se ha escogido en base a la utilidad de los datos recogidos, ya sea para medir datos concretos de las acciones del proyecto GrowGreen o para tomar datos de control y de referencia de la zona.

Por otro lado, se han buscado zonas aptas para la colocación y buen funcionamiento de los dispositivos, esto es, lugares que cuenten con farolas altas o muros en los que no puedan ser manipulados o dañados, pero lo suficientemente bajas como para que la toma de parámetros a nivel de peatón no se vea distorsionada por la altura. Además, se ha intentado que sean relativamente accesibles en caso de necesitar mantenimiento, que queden orientadas de tal forma que se maximice la radiación en el panel solar para la recarga de las baterías y que la distancia al receptor se mantenga dentro del rango de las antenas.

Así pues, a continuación, se van a describir las ubicaciones elegidas para la instalación de cada uno de los dispositivos fabricados, así como sus características y los diferentes sensores con los que cuenta cada uno:

ID	Ubicación	Coordenadas	Distancia	Instalación
Receptor	Parque Benicalap: Oficina	39°29'55.2"N 0°23'46.7"W	-	25-10-2018
HSM1	Parque Benicalap: Entrada	39°29'54.8"N 0°23'45.6"W	33 m	09-01-2019
HSM3	Plaza Regino Mas	39°29'57.6"N 0°23'38.3"W	214 m	09-01-2019
HSM4	Calle Luis Braille	39°29'48.4"N 0°23'40.2"W	260 m	09-01-2019
HSM5	Parque Benicalap: Gradas	39°29'55.5"N 0°23'48.0"W	34 m	11-12-2018
HSM6	Carrer del Ninot	39°30'00.0"N 0°23'33.6"W	345 m	09-01-2019
HSM7	Carrer del Foc	39°30'01.5"N 0°23'38.7"W	271 m	17-04-2019
HSM8	Colegio: Muro exterior	39°29'58.4"N 0°23'37.8"W	235 m	20-05-2019
HSM9	Colegio: Muro interior	39°29'58.4"N 0°23'37.8"W	235 m	01-06-2019
HSM10	Centro de Mayores: Azotea	39°29'46.7"N 0°23'43.9"W	263 m	10-05-2019
HSM11	Centro de Mayores: Interior	39°29'47.1"N 0°23'43.2"W	258 m	10-05-2019
HSM12	Centro de Mayores: Azotea	39°29'47.1"N 0°23'43.2"W	258 m	01-06-2019

Figura 44. Tabla resumen de dispositivos instalados en Benicalap.

ID	DHT22		3x DS18B20			Anemo.	BME280	2x Célula solar		Lluvia
HSM1	HR_DHT	TEX_DHT	WBGT_DS	TX_DS	-	V_WIND	PATM	IRR_UP	IRR_DOWN	RAIN
HSM3	HR_DHT	TEX_DHT	WBGT_DS	TX_DS	-	V_WIND	PATM	IRR_UP	IRR_DOWN	RAIN
HSM4	HR_DHT	TEX_DHT	WBGT_DS	TX_DS	-	V_WIND	PATM	IRR_UP	IRR_DOWN	RAIN
HSM5	HR_DHT	TEX_DHT	WBGT_DS	TX_DS	-	V_WIND	PATM	IRR_UP	-	RAIN
HSM6	HR_DHT	TEX_DHT	WBGT_DS	TX_DS	-	-	PATM	IRR_UP	-	RAIN
HSM7	HR_DHT	TEX_DHT	WBGT_DS	TX_DS	-	V_WIND	PATM	IRR_UP	IRR_DOWN	RAIN
HSM8	HR_DHT	TEX_DHT	WBGT_DS	TWALL1	TWALL2	V_WIND	PATM	IRR_UP	IRR_DOWN	RAIN
HSM9	HR_DHT	TEX_DHT	WBGT_DS	TWALL1	TWALL2	-	PATM	-	-	-
HSM10	HR_DHT	TEX_DHT	WBGT_DS	TWALL1	TWALL2	V_WIND	PATM	IRR_UP	IRR_DOWN	RAIN
HSM11	HR_DHT	TEX_DHT	WBGT_DS	TWALL1	TWALL2	-	PATM	-	-	-
HSM12	HR_DHT	TEX_DHT	TX_DS	TWALL1	TWALL2	-	-	-	-	-

Figura 45. Tabla resumen de sensores y variables de medida de los dispositivos.

Figura 46. Mapa de Benicalap con las ubicaciones de los sensores. Fuente: ICV.

- **Receptores:** En total se han instalado dos receptores con acceso a la base de datos. El primer receptor está situado en el interior de una oficina del parque Benicalap, en una caja atornillada a la pared. Está alimentada a la red eléctrica y cuenta con antena de radiofrecuencia externa, colocado en lo alto de un poste, y el módem 3G conectado a través de un cable alargador para maximizar la

cobertura de datos. Este receptor se encarga de dar cobertura a todos los transmisores instalados en el barrio de Benicalap, y envía todas las mediciones a las tablas “HSMx” de la base de datos.

El segundo receptor está situado en el laboratorio del Instituto de Ingeniería Energética (IIE), en el CPI de la UPV. Este receptor está pensado para las pruebas de los prototipos nuevos que se fabriquen y para dar cobertura a los transmisores que se han colgado en la UPV a lo largo del periodo de desarrollo del sistema. Envía todas las mediciones que recoge a las tablas “HSMUPVx” de la base de datos.

Figura 47. Receptor de Benicalap instalado.

- **HSM1:** Transmisor situado en la entrada del Parque Benicalap. Se encuentra colgado en una farola junto a un bosquete de árboles altos, que le dan cierta cantidad de sombra. Cuenta con sensor de radiación solar superior e inferior y anemómetro. Su objetivo es proporcionar datos ambientales de referencia de los alrededores del parque Benicalap, como ejemplo de zona arbolada.

Figura 48. Transmisor HSM1 instalado.

- **HSM2:** Actualmente, este receptor no se encuentra instalado. Está proyectado que se ubique próximamente en un poste en la esquina norte del Parque Benicalap en línea con los trabajos de ampliación del parque en esa zona.

- **HSM3:** Transmisor situado en la Plaza Regino Mas. Está colgado mediante abrazaderas anchas de una palmera en el centro de la plaza, junto al CEIP Ciutat Artista Faller. Cuenta con sensor de radiación superior en inferior y anemómetro. Se espera que proporcione datos de referencia de los alrededores del Colegio.

Figura 49. Transmisor HSM3 instalado.

- **HSM4:** Transmisor situado en la calle Luis Braille. Se encuentra colgado en una farola junto a un lavadero de coches, en una zona muy despejada con una gran superficie de asfalto alrededor. Cuenta con sensor de radiación superior e inferior y anemómetro. Su objetivo es obtener datos de referencia de los alrededores del centro de mayores y, además, servir como modelo de zona sin cobertura ni zonas verdes, donde se espera que el nivel de insolación sea muy superior.

Figura 50. Transmisor HSM4 instalado.

- **HSM5:** Transmisor situado en el interior del Parque Benicalap, junto al anfiteatro. Está colgado en una farola alta rodeada de césped y arbustos. Cuenta con sensor de radiación superior y anemómetro. Fue el primer transmisor que se instaló. Su objetivo era servir como prueba de concepto del sistema y para testear la toma de datos, transmisión, autonomía y durabilidad del prototipo.

Figura 51. Transmisor HSM5 instalado.

- **HSM6:** Transmisor instalado en el Carrer del Ninot, junto al Museu de l'Artista Faller y el CEIP Ciutat Artista Faller, colgado en una farola. Cuenta con sensor de radiación superior, y su objetivo es obtener más datos de referencia de los alrededores del colegio y las calles del barrio.

Figura 52. Transmisor HSM6 instalado. Fuente: Google Street View.

- **HSM7:** Transmisor colocado en el Carrer del Foc, colgado de una farola. Cuenta con sensor de radiación superior e inferior y anemómetro. Su objetivo es tomar datos de esta calle residencial, en la que los edificios ofrecen cierta cobertura, y se está rodeado árboles de hoja caduca, por lo que el nivel de insolación debería variar considerablemente en diferentes épocas del año.

Figura 53. Transmisor HSM7 instalado. Fuente: Google Street View.

- **HSM8:** Transmisor situado en el muro exterior del CEIP Ciutat Artista Faller, en el edificio de aulas de educación infantil. Cuenta con sensor de radiación superior e inferior y anemómetro. En este caso los sensores de temperatura DS18B20 tienen una configuración diferente: uno mide la temperatura de globo negro, mientras que los otros dos están conectados a través de un cable largo de unos 5 metros, cada uno fijado con adhesivo en un extremo del muro, cubiertos con pasta conductora térmica y cinta aislante reflectora de aluminio. El objetivo de este transmisor es medir las variables ambientales en el exterior del edificio y, además, medir las temperaturas superficiales del exterior del muro. Mientras una de las dos sondas sirve de control, la otra medirá la temperatura superficial del muro antes y después de la instalación del jardín vertical.

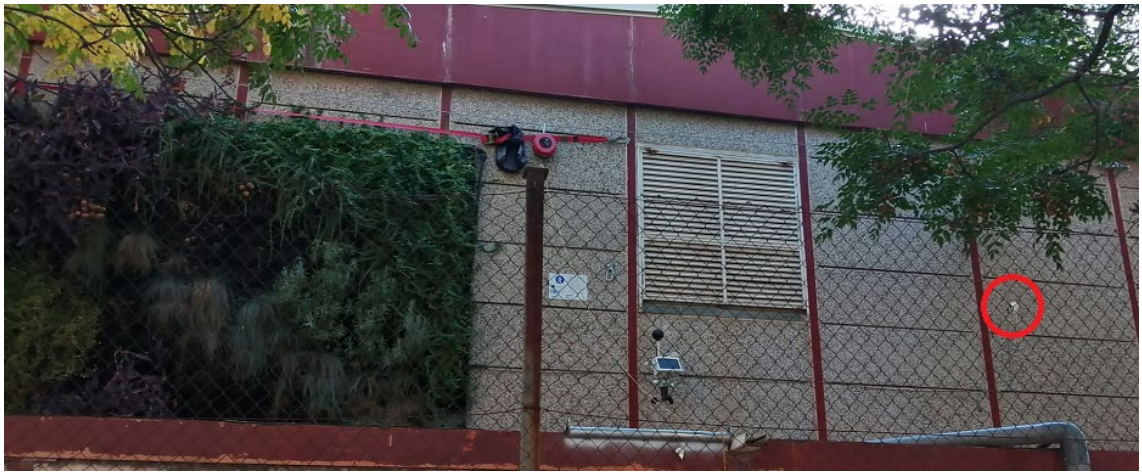


Figura 54. Transmisor HSM8 instalado.

- **HSM9:** Transmisor instalado en una de las aulas del CEIP Ciutat Artista Faller, en la parte interior del muro destinado a la instalación del jardín vertical. Al estar situada en interiores, se alimenta a través de la red eléctrica, y cuenta tan solo con sensor de temperatura, humedad y presión, además de temperatura de globo negro y dos sondas que, al igual que en la parte exterior del muro, miden las temperaturas superficiales de la pared. Se espera que, tras la instalación del jardín vertical, puedan apreciarse variaciones relevantes en los datos de temperaturas recogidos.

Figura 55. Transmisor HSM9 instalado.

- **HSM10 y HSM11:** Transmisores situados en la azotea del centro municipal para mayores. Toman medidas en la parte superior del techo del edificio y, con dos sondas de temperatura conectadas con cables largos, mide las temperaturas superficiales del techo. HSM10 cuenta con temperatura de globo negro, anemómetro y sensor de radiación superior e inferior, y HSM11 con sensor de temperatura de globo negro. Entre los dos tienen 4 sondas para temperatura superficial, acopladas con cinta adhesiva y cubiertas con pasta térmica conductora y Cintia de aluminio reflectante. Cuando se instale la cubierta ajardinada en la azotea del edificio se podrán tener medidas anteriores y posteriores a ello para comparar.

Figura 56. Transmisor HSM10 instalado.

- **HSM12:** Transmisor instalado en el interior del centro municipal para mayores, en la sala multiusos. Se trata de una sala amplia usada habitualmente para actividad física, y en verano alcanza temperaturas muy altas. Se espera que, tras la instalación de la cubierta ajardinada en la azotea, la temperatura en el interior de la sala mejore notablemente. Por lo tanto, este transmisor se encarga de tomar los datos relativos al interior de la sala. Puesto que se encuentra en interiores, se alimenta a través de la red eléctrica, y sólo cuenta con sensores de temperatura y humedad. Además, tiene dos sondas de cable largo para medir la temperatura superficial de la parte interior del techo, con las que se pretende ver el efecto que produce la cubierta ajardinada.

Figura 57. Instalación de sondas de temperatura en el falso techo en HSM12.

- **HSMUPV8 y HSMUPV9:** Transmisores de pruebas situados en la UPV, colgados ambos en una misma farola cerca del CPI. HSMUPV9 estaba situado más abajo, con sensor de radiación inferior, anemómetro y temperatura de globo negro de 90 mm (usado habitualmente en este proyecto). HSMUPV8 estaba situado encima y contaba con sensor de radiación superior, anemómetro y un sensor de temperatura dentro de un globo negro de 15 cm de diámetro. Su objetivo principal era la prueba de concepto del transmisor, así como para testear su durabilidad y autonomía.

  Además, se pretendía comparar las temperaturas de los sensores de globo negro en cada uno de los globos de diferente diámetro. El objetivo era determinar las variaciones que podían encontrarse a la hora de efectuar el cálculo de la temperatura radiante media según (UNE ISO 7726, 2002),[9] que fija en 15 cm el diámetro de referencia para el globo negro. Los resultados de estas pruebas se pueden consultar en (Vargas-Salgado, Chiñas-Palacios, Aguila-León and Alfonso-Solar, 2019).[10]

Figura 58. Transmisores HSMUPV8 y HSMUPV9 instalados.

6. Resultados

Así pues, el sistema ha quedado instalado en sus ubicaciones finales, y actualmente sigue en funcionamiento. A lo largo de los meses que ha durado por el momento el tiempo de operación, han sido necesarias tareas de mantenimiento, reparación y actualización de los dispositivos. Puesto que los periodos de diseño y desarrollo se han solapado con los periodos de instalación, resulta inevitable que se hayan sucedido diferentes iteraciones en el diseño y que algunos transmisores presenten diferencias entre ellos a medida que se van sustituyendo.

Sin embargo, los prototipos que se retiran nos ofrecen información muy valiosa acerca de su desempeño, y a lo largo de los meses se han realizado diferentes pruebas que nos dan datos acerca de su funcionamiento.

A continuación, se ofrece una evaluación del desempeño del sistema en diferentes aspectos clave de su funcionamiento, todo ello en base a los datos recogidos o a las pruebas realizadas durante el desarrollo. Esto nos permitirá comprobar si se han cumplido los objetivos de diseño que se habían marcado y evaluar la idoneidad del sistema o determinar si son necesarias nuevas mejoras.

6.1. Integridad estructural y durabilidad

Uno de los aspectos más críticos del diseño es su estructura. El transmisor debía cumplir con muchas especificaciones: trabajar a la intemperie, siendo resistente a lluvias y radiación solar intensa, con alimentación independiente de la red y equipado con una serie de sensores que requerían de unas ciertas condiciones para medir sin distorsiones. Todos estos requisitos se ven reflejados en el diseño de la caja, diseñada a medida e impresa en 3D. Este es un elemento que ha pasado por diversas iteraciones de diseño, pero básicamente se pueden destacar dos versiones principales.

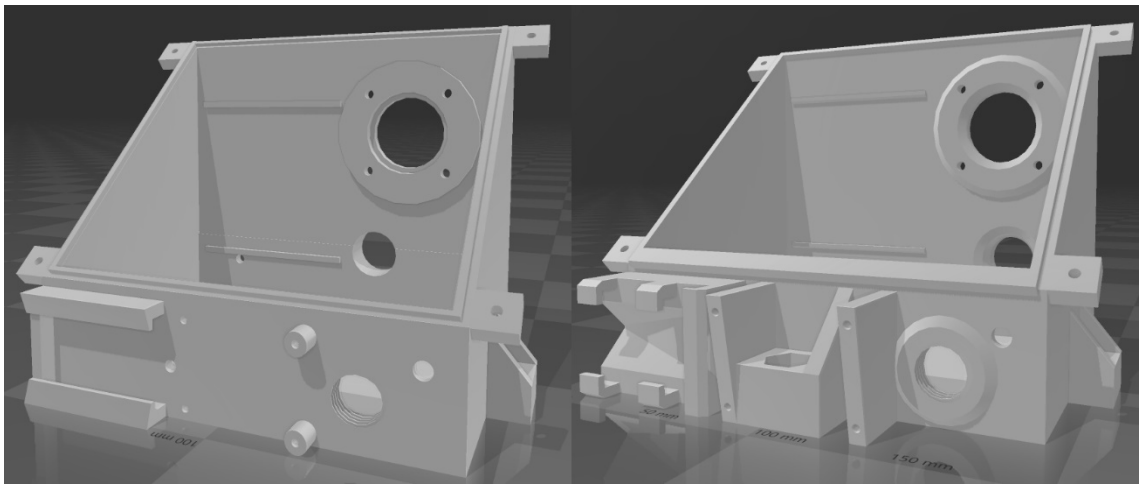


Figura 59. Comparativa entre la caja vieja (izquierda) y la caja nueva (derecha).

A simple vista, se pueden apreciar diferencias importantes en el diseño. Algunos de estos cambios se deben a mejoras de cara a la impresión 3D, ahorro de plástico, optimización de soportes, etc. Pero el principal motivo de las modificaciones realizadas surge en vista a mejorar la estanqueidad del conjunto.

Los primeros prototipos que se probaron en la UPV sufrían habitualmente de filtraciones de agua, agravadas por deformaciones en el plástico, que producían oxidación en los componentes, cortocircuitos y apagones recurrentes. Además, era común la entrada de insectos en el interior de la caja, que provocaban más problemas.

Figura 60. Deformaciones y daños observados en los primeros prototipos.

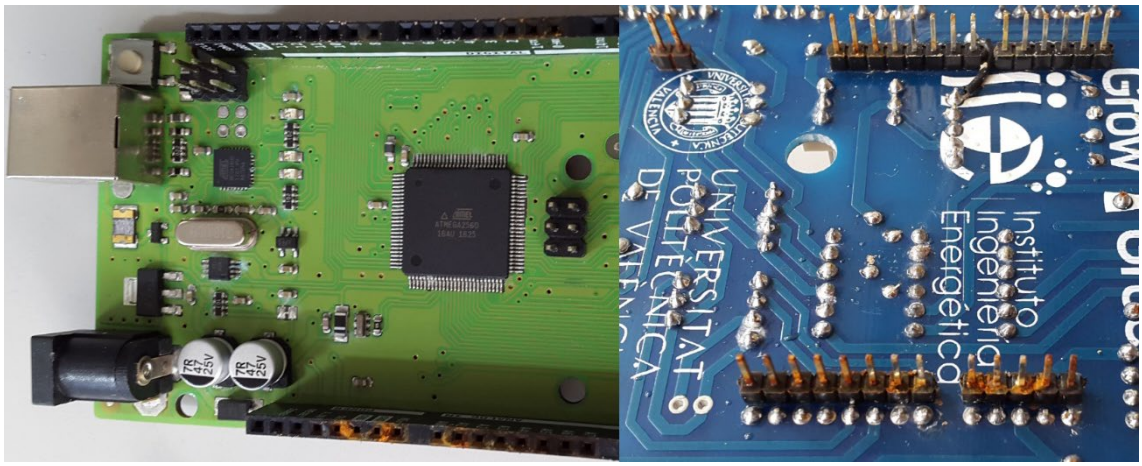


Figura 61. Daños por óxido observados en los primeros prototipos.

Una de las soluciones propuestas fue la utilización de otro material para la impresión 3D. El PLA es el más extendido, más económico y más sencillo de imprimir, sin embargo, no está pensado para trabajar a la intemperie. Se realizaron pruebas con plástico ABS y ASA, pero estos materiales suelen imprimirse a más altas temperaturas y suelen tener aparejados una serie de complicaciones a la hora de trabajar con ellos. Por lo que la solución adoptada finalmente consistía, por un lado, cambiar la configuración de la impresora para mejorar la robustez y el resultado general de la impresión y, por otro lado, recubrir las cajas con pintura blanca de exterior y barniz transparente. Esto sellaba las posibles imperfecciones en las paredes impresas y protegía al conjunto de deformaciones frente a la radiación solar. Además, se efectuaron una serie de modificaciones importantes en el diseño 3D del conjunto del transmisor:

- Forma de juntas y oberturas mejorada para favorecer la evacuación del agua.
- Reducción y recolocación de oberturas para minimizar las filtraciones.
- Sellado de juntas con silicona y/o con cinta de teflón para fontanería.
- Sistema de sujeción de sensores de radiación mejorado para facilitar el montaje.

- Nuevo diseño de tapa más resistente a deformaciones y pensada para evacuar mejor el agua.

- Añadida cúpula de plástico transparente para evitar filtraciones a los sensores de radiación superiores.

- Añadida tela mosquitera en la pantalla de sensores inferior para evitar la entrada de insectos al interior.

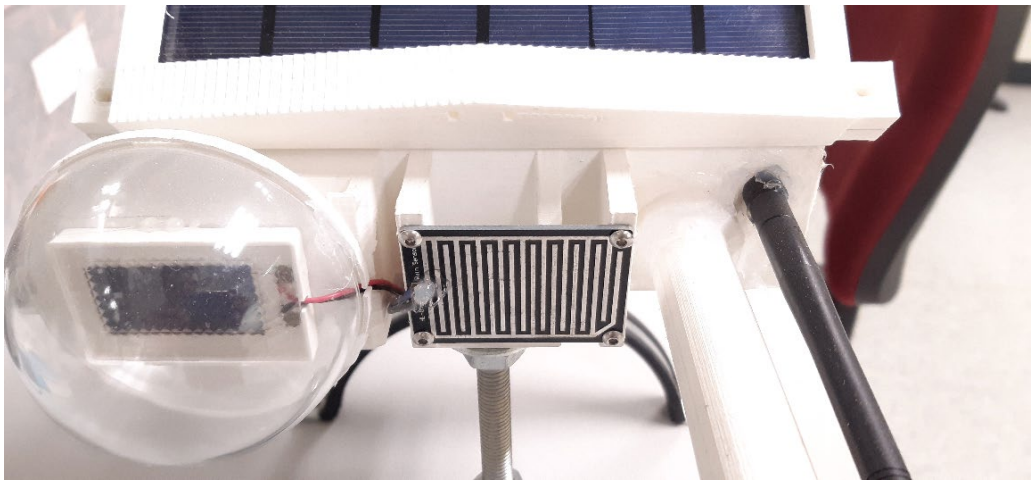


Figura 62. Vista superior de la caja nueva montada con sus modificaciones.

Con todas estas modificaciones, el prototipo fue sometido a varias pruebas de estanqueidad, como situarla debajo de un grifo abierto rociando cada una de sus partes. Las pruebas fueron exitosas y no se detectaron fugas de agua. Un ejemplo es el transmisor HSM1, el primero que contó con estas modificaciones, y desde su sustitución en febrero de 2019 ha resistido temporadas de lluvias intensas (como las de marzo y abril de 2019 y 2020, por ejemplo), y a día de hoy sigue funcionando sin interrupciones.

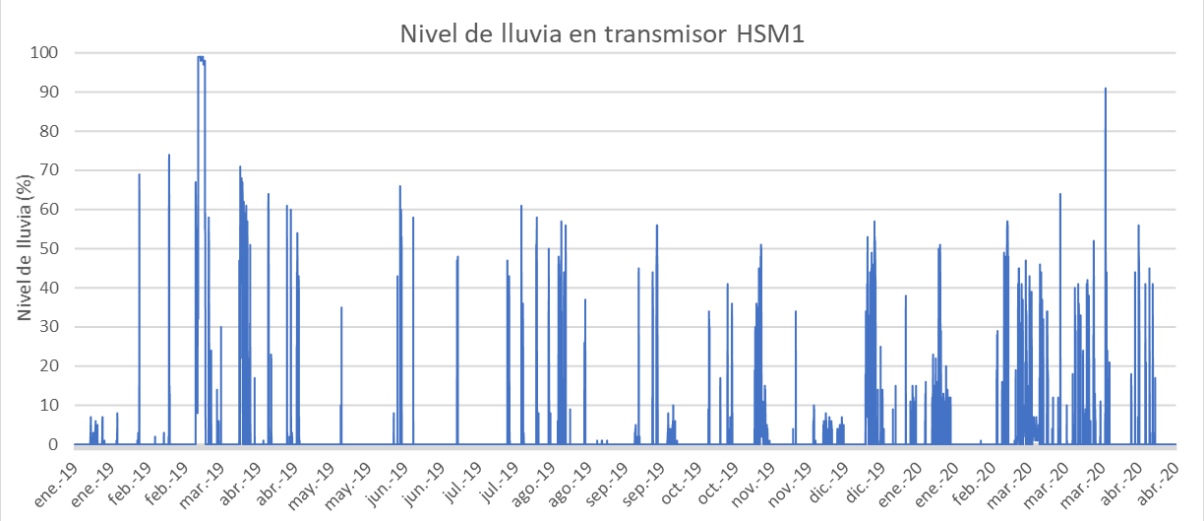


Figura 63. Gráfico del nivel de lluvia medido por el transmisor HSM1.

Otras cajas a las que se ha accedido para su mantenimiento, han presentado señales mínimas de deformaciones y desgaste en su estructura y mantienen su nivel de estanqueidad. De los 10 transmisores instalados con el nuevo diseño, ninguno ha experimentado fallos debidos a defectos estructurales o de estanqueidad. En conclusión, podemos decir que el nuevo diseño cumple significativamente los requisitos establecidos en el proyecto.

6.2. Fiabilidad de medidas

El siguiente aspecto a tener en cuenta en la validación del diseño es la fiabilidad de las medidas tomadas por el dispositivo. Aquí hay que tener en cuenta tanto la precisión y fiabilidad de cada uno de los sensores como la fiabilidad de las comunicaciones.

La transmisión de radiofrecuencia a 433 MHz es susceptible a la aparición de interferencias, que distorsionan los datos y hacen que las medidas sean inservibles. Para evitar la aparición de medidas erróneas e incoherentes en la base de datos, el receptor cuenta con un sistema de filtrado de los paquetes de datos corruptos que recibe. Cuando el receptor detecta un patrón en el mensaje que indica que ha sufrido interferencias, deshecha el mensaje y lo registra en una tabla de la base de datos llamada “ERROR”. Así, se puede realizar una estimación del número total de medidas que el receptor deshecha cada día, con lo que podemos obtener el porcentaje de fiabilidad del sistema.

Tomando como referencia las medidas tomadas en Benicalap en octubre de 2020, tenemos 1381 medidas erróneas para 8 cajas que miden cada 15 minutos aproximadamente. Así pues:

$$\%_{\boldsymbol{fallo}} = \frac{Medidas\ erróneas}{Medidas\ totales} \cdot 100 = \frac{1381\ medidas}{8\ cajas \cdot \frac{1\ medida}{15\ min \cdot caja} \cdot \frac{1440\ min}{dia} \cdot 31\ dias} \cdot 100 = \frac{1381}{23808} \cdot 100 = \mathbf{5.8}\%$$

De esta forma, obtenemos que el porcentaje de pérdidas de medidas es del 5,8% o, lo que es lo mismo, el porcentaje de efectividad es del 94,2%. Poniendo en contexto este dato, significa que cada caja perderá diariamente entre 5 y 6 medidas. Puesto que estos errores aparecen de forma aleatoria y no se suelen producir consecutivamente, podemos decir que no son un problema importante. Variables como la temperatura o la humedad ambiente tienen dinámicas lo suficientemente lentas como para que medidas tomadas puntualmente cada 30 minutos en vez de 15 no resulten una pérdida significativa de información.

6.3. Fiabilidad de sensores

Las condiciones de trabajo de los transmisores y la necesidad de usar sensores de bajo coste, sumando a las largas temporadas de funcionamiento, hacen que en algunos casos los sensores pierdan precisión o que incluso se necesario sustituirlos. A continuación, se detallan los principales problemas de fiabilidad detectados en los sensores y cómo se han intentado corregir:

- **DHT22:** Este sensor de temperatura y humedad (también llamado AM2302) no está diseñado específicamente para trabajar en entornos de muy alta o muy baja humedad durante periodos prolongados.[2] Esto produce que, en algunos transmisores situados en el exterior, el sensor de humedad satura y se mantiene continuamente alrededor del 99%, marcando valores mucho más altos de lo debido. Existe la posibilidad de recalibrar el sensor, pero en todo caso requiere que se sustituya por otro nuevo.

  La solución propuesta para este problema consta de dos partes. Por un lado, para los cálculos que requieran el valor de la humedad relativa, y como solución provisional para las medidas que se han tomado hasta ahora, se han tomado los valores de humedad publicados por el Ajuntament de València pertenecientes a

los Jardins del Real, ajustándolos a la temperatura medida en Benicalap, con lo que se pueden obtener valores aproximados. Por otro lado, se va a proceder próximamente a una actualización de los transmisores en la que el sensor DHT22 va a ser eliminado, y la medida de humedad se tomará a partir del sensor BME280 (hasta ahora usado como sensor de presión). Además, se añadirá el sensor de temperatura y humedad AM2320 (una actualización del DHT22 o AM2302), así que habrá dos datos diferentes de humedad para mayor seguridad.

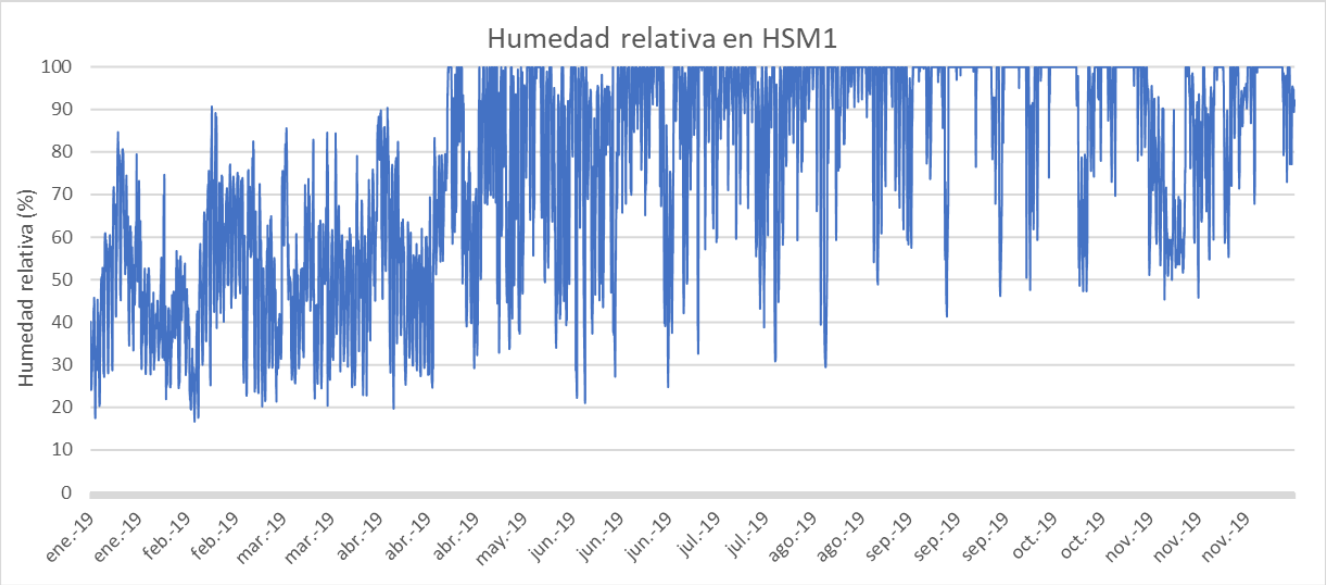


Figura 64. Gráfico de humedad relativa en HSM1 para un sensor DHT22 saturado.

- **DS18B20:** Las sondas de temperatura DS18B20 resultan ideales para el trabajo en la intemperie y el bus de datos digital *OneWire* ofrece una transmisión limpia y segura. Sin embargo, tiene un inconveniente importante. Puesto que las tres sondas están conectadas a través del mismo bus, el microcontrolador asigna automáticamente un identificador a cada una de las sondas para distinguirlas y que no puede modificarse. Por cómo está diseñado el programa Arduino del transmisor, esto provoca que las temperaturas de las tres sondas aparezcan desordenadas en la base de datos de algunos transmisores (WBGT_DS_C en el lugar de TX_DS_C, por ejemplo).

  Una forma rápida de corregirlo es modificar el archivo PHP de Plesk *"conexionbeni.php"*, cambiando manualmente el orden de las temperaturas en la petición SQL que las publica en la base de datos. Por desgracia, eso no cambia las temperaturas anteriores, por lo que es necesario también editar los históricos de datos para corregirlo.

  Una solución permanente que se implementará próximamente en la nueva remesa de repuestos de transmisores será la modificación del circuito de los sensores DS18B20. Lo que se hará será usar GPIO libres del microcontrolador para configurar tres buses de datos. Con ello, se podrá controlar el orden en el que se envían estas temperaturas.

- **MH-RD Sensor:** Este es un sensor de gotas de lluvia, por lo que no ofrece la presición que puede ofrecer un pluviómetro calibrado en litros por metro cuadrado. Por ello, sus medidas se toman como una aproximación a la cantidad de lluvia en un momento determinado, si está lloviendo o no y durante cuánto tiempo. Aunque la variable RAIN indique un porcentaje, éste debe tomarse como un valor aproximado.

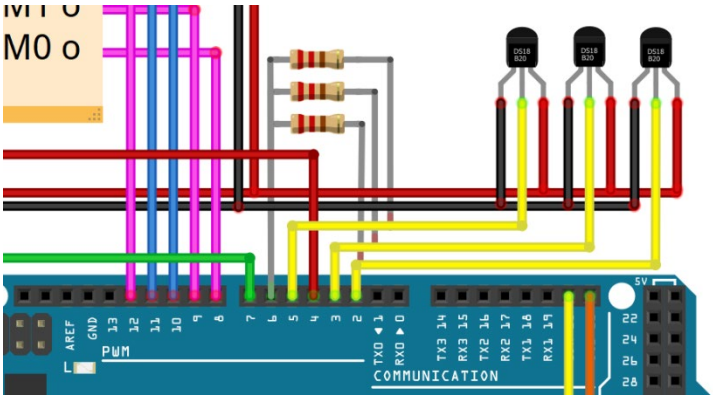


Figura 65. Circuito del transmisor modificado para tres sondas DS18B20.

- **Wind Sensor Rev. C:** Este es un sensor de viento de hilo caliente que se usó en los primeros prototipos. Sin embargo, actualmente se encuentra en desuso y las medidas que ha tomado no se toman en cuenta, puesto que era un sensor muy frágil y su precisión dependía enormemente de la dirección del viento.

- **Anemómetro de tres copas:** Por las limitaciones de espacio del transmisor, este anemómetro está montado boca abajo. Aunque esto no afecta a las medidas tomadas, puesto que no está ideado para trabajar de esta forma, tiende a acumular agua en su rodamiento durante la lluvia, lo que puede provocar oxidación y que no gire. Esto suele requerir la sustitución del sensor. En próximas actualizaciones, se intentará abordar este problema perforando el rotor de los anemómetros para facilitar la evacuación del agua.

- **Célula solar calibrada:** Este sensor de radiación solar viene calibrado por el fabricante y ofrece una buena muy buena precisión. Por desgracia, dejarla a la intemperie significa que la suciedad y la lluvia se acumulan sobre el plástico prismático y en el interior del sensor, porque lo que lo dañan a largo plazo. La solución que se ha adoptado es situarlo debajo de una cúpula de policarbonato transparente que lo protege del agua y la suciedad, pero a costa de que esa cúpula pierda transparencia con el paso del tiempo. La solución puede ser sustituir la cúpula periódicamente o caracterizar la curva de desgaste de la cúpula para ajustar los datos de radiación.

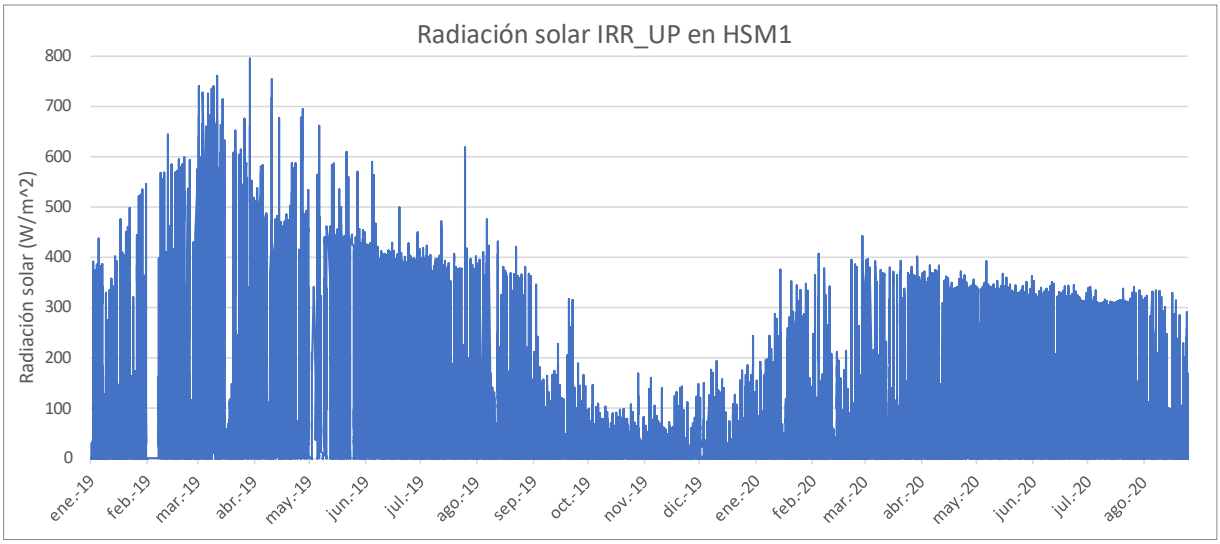


Figura 66. Radiación solar medida en sensor con cúpula con desgaste progresivo.

6.4. Alcance de comunicaciones

Otro parámetro importante a la hora de evaluar el diseño del sistema consiste en el rango de alcance, es decir, la distancia máxima a la que se pueden comunicar entre sí por radiofrecuencia los dispositivos transmisores y el receptor.

Durante el periodo de desarrollo se realizaron una serie de pruebas de rango de diferentes tipos de antena, aunque por aquel momento se estaba desarrollando el sistema con el transmisor de radiofrecuencia nRF24L01 de 2,4 GHz. Las pruebas consistieron en unos pequeños dispositivos portátiles basados en Arduino UNO con un módulo de radiofrecuencia y una antena cada uno, con uno de ellos actuando como transmisor en un lugar fijo, y los otros 4 como receptores desplazándose a diferentes localizaciones para testear el rango de alcance de cada una de las antenas. Los recorridos de la prueba por los alrededores de la UPV fueron los siguientes:

Figura 67. Rutas seguidas durante la prueba de rango de antenas.

El resultado de las pruebas indicó que los módulos de radiofrecuencia (marcados en las especificaciones para una distancia máxima de 1000 m) conseguían transmitir sin excesivas pérdidas hasta a 700 m en línea vista, pero el rango disminuía drásticamente hasta por debajo de los 300 metros cuando había edificios bloqueando la visión, con ligeras variaciones dependiendo del tipo y la longitud de antena. Se comprobó también que apantallar los módulos con papel de aluminio para reducir interferencias no producía ninguna diferencia apreciable. Por tanto, se descartó el uso de transmisiones por radiofrecuencia a 2,4 GHz.

En su lugar, se optó por un módulo de radiofrecuencia de 433 MHz, una frecuencia menos sensible a la presencia de obstáculos sólidos. El módulo escogido, el E32-TTL-1W, ofrecía un rango máximo en línea vista de 7500 m, por lo que era más que suficiente para la aplicación que nos ocupa. Aunque en este caso no se realizó una prueba de rango propiamente dicha, se testeó su alcance directamente sobre el terreno instalando transmisores con estos nuevos módulos en diferentes localizaciones de la UPV, cada una más lejana que la anterior.

Figura 68. Ubicaciones de los transmisores en la prueba de rango en la UPV.

Todos los transmisores instalados funcionaron a la perfección desde el primer minuto y en ningún momento presentaron ningún problema relacionado con la pérdida de medidas por rango. Aunque una buena parte de la distancia a la que se encontraban era línea vista, había un buen número de árboles y construcciones bajas en la trayectoria. Además, la antena del receptor (un modelo preparado para exteriores) se situó en el interior del laboratorio del IIE, en el CPI, por lo que se pudo comprobar que el módulo de radiofrecuencia y las antenas eran capaces de funcionar aún con una buena cantidad de paredes en su trayectoria, por lo que se dio por válido para Benicalap.

6.5. Autonomía

Sin duda, uno de los aspectos más limitantes a la hora de diseñar el sistema ha sido el consumo energético. La necesidad de contar con alimentación independiente de la red ha obligado a controlar este factor. Cada transmisor, dependiendo de su ubicación, monta 2 o 4 baterías de litio recargables de 3,7 V tipo 18650 de diversas capacidades. Por otro lado, cuenta con un panel solar de 165x135 mm y 3,5 W para cargar las baterías durante los periodos de luz solar.

Cabe destacar que en todo momento se ha intentado maximizar el aporte energético de los paneles solares, diseñando la tapa de la caja con un ángulo de 48º respecto al horizonte y durante la instalación se ha procurado que todas las cajas queden orientadas hacia el sur para aprovechar al máximo la luz solar.

Con esto, se puede efectuar un cálculo aproximado de la autonomía del dispositivo. Afortunadamente, el circuito cuenta con diversos sensores de tensión y corriente con lo que podemos medir variables útiles como: VBUS (tensión de entrada), V_PV e I_PV (tensión y corriente en los pines del panel solar), I_CH (corriente que aporta el panel solar tras el módulo cargador de baterías), I_BAT (corriente que aportan las baterías) e I_IN (corriente que consume el circuito, equivalente a I_BAT+I_CH).

6.5.1. Perfiles de carga de baterías

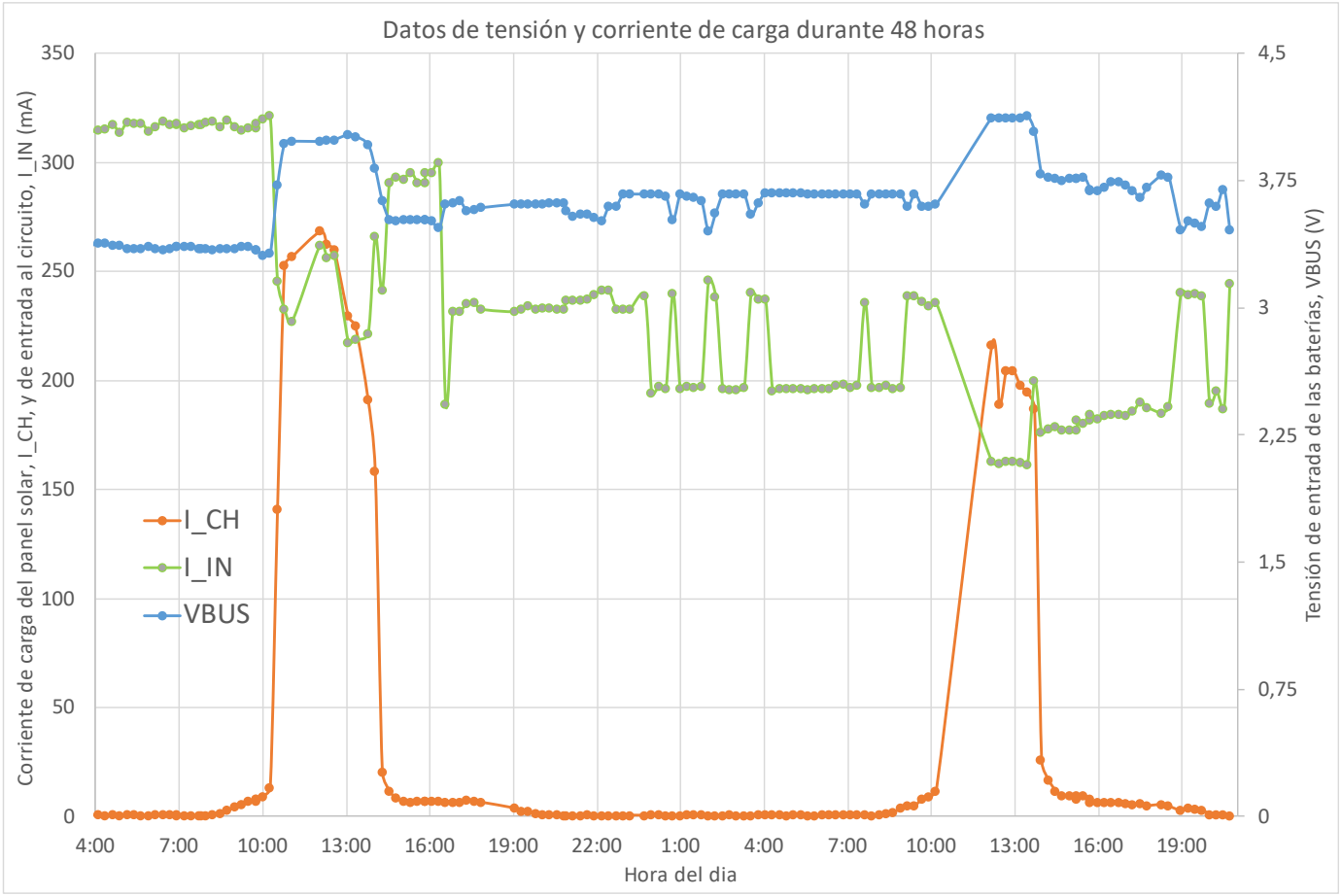


Figura 69. Corrientes y tensiones medidas tras 48 horas en el laboratorio.

En la *Figura 69*, se puede observar cómo la tensión de la batería aumenta tras unas horas con el panel solar cargando. Hay que tener en cuenta que estos valores están tomados en la luz ambiente del laboratorio, por lo que no representan condiciones

de radiación reales. Sin embargo, con estos datos podemos extraer información relevante. Puesto que también medimos la radiación solar (IRR_UP), podemos establecer una relación entre la radiación y la corriente de carga del panel solar, con lo que podríamos realizar una estimación de cuánto tardaría el panel en cargar todas las baterías bajo distintas condiciones.

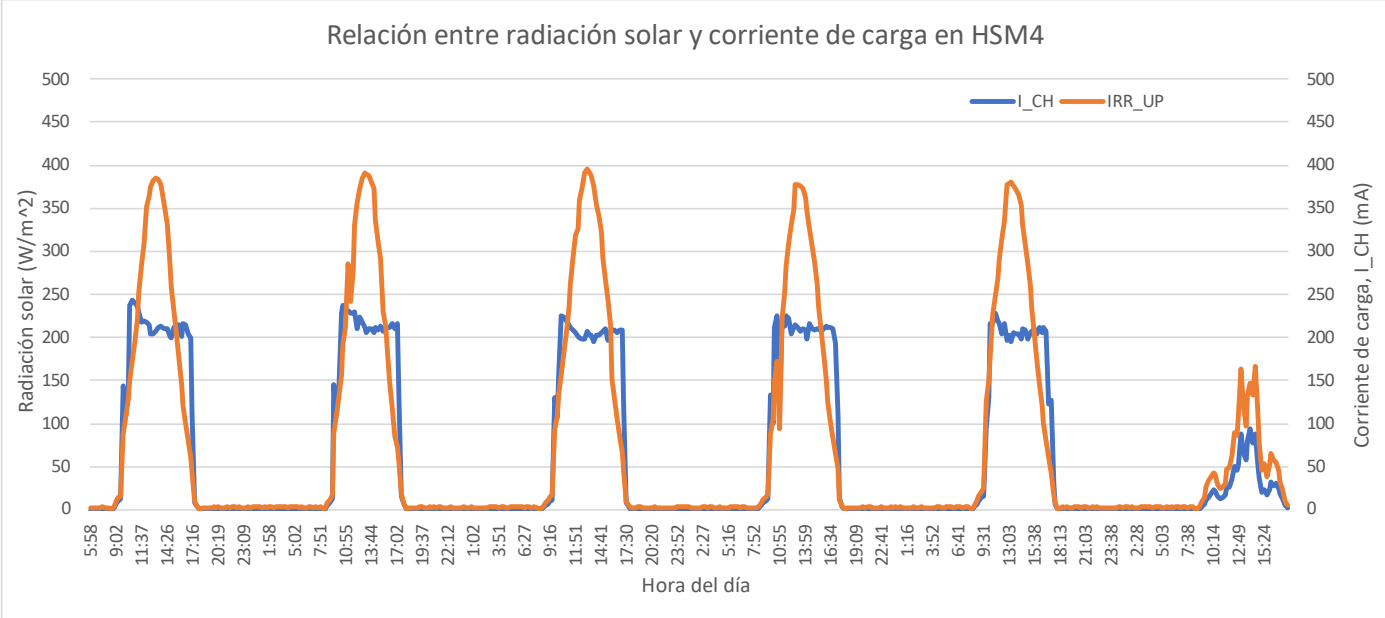


Figura 70. Tabla comparativa entre radiación solar y corriente de carga en HSM4.

Como se aprecia en la *Figura 70*, el panel solar otorga una corriente de carga máxima a partir de un cierto valor de radiación, a excepción de los primeros días. Este cambio se debe a las fluctuaciones en las corrientes debido a que las baterías están en estado de carga máxima. Cuando la radiación no llega a ese valor umbral, la corriente que otorga el panel es aproximadamente proporcional a la radiación, por lo que aún sigue otorgando carga.

Así pues, podemos tomar como referencia dos valores promedio de radiación, para un día nublado y uno soleado, y con ello estimar el valor de la corriente de carga del panel solar. Teniendo en cuenta que el módulo de carga de baterías convierte la tensión a 4,2 V, podemos calcular el tiempo que tardaría el panel en recargar 4 baterías 18650 de 3,7 V y distintas capacidades usando la Ecuación 2, suponiendo un rendimiento de carga conservador del 90%:

$$E = V \cdot I \cdot t \cdot \eta \rightarrow t = 4 \cdot \frac{C_{BAT} \cdot V_{BAT}}{V_{CH} \cdot I_{CH} \cdot \eta} \qquad (2)$$

Tipo de día	IRR_UP	I_CH	Tiempo de carga 4 baterías de capacidad:			
			1650 mAh	2000 mAh	3000 mAh	3400 mAh
Día nublado	80 W/m^2	40 mA	161,5 h	195,8 h	293,7 h	332,8 h
Día soleado	300 W/m^2	200 mA	32,3 h	39,2 h	58,7 h	66,6 h

Figura 71. Tabla de tiempos de carga completa del panel solar.

Aunque a primera vista parezcan tiempos muy elevados, hay que tener en cuenta que largos periodos sin radiación no son comunes en Valencia, y será muy poco habitual que tenga que realizarse una carga completa de baterías. A pesar de ello, se deben tener también en cuenta los tiempos de descarga.

6.5.2. Perfiles de descarga de baterías

Durante el desarrollo se realizaron diversas pruebas de consumo y autonomía del transmisor mediante descargas de batería. Sin embargo, tras la instalación en Benicalap se dieron dos casos de transmisores que tuvieron que seguir funcionando sólo con baterías hasta que se agotaron completamente: HSM1, al que tras la instalación se le detectó un fallo en el panel solar; y HSM9 que, al estar conectado a la red eléctrica en el interior de un aula de colegio, se quedó sin alimentación tras el cierre forzado en marzo de 2020 por la crisis de la COVID-19.

A pesar del inconveniente, estos hechos nos ofrecen una oportunidad de obtener datos reales sobre la autonomía de los transmisores.

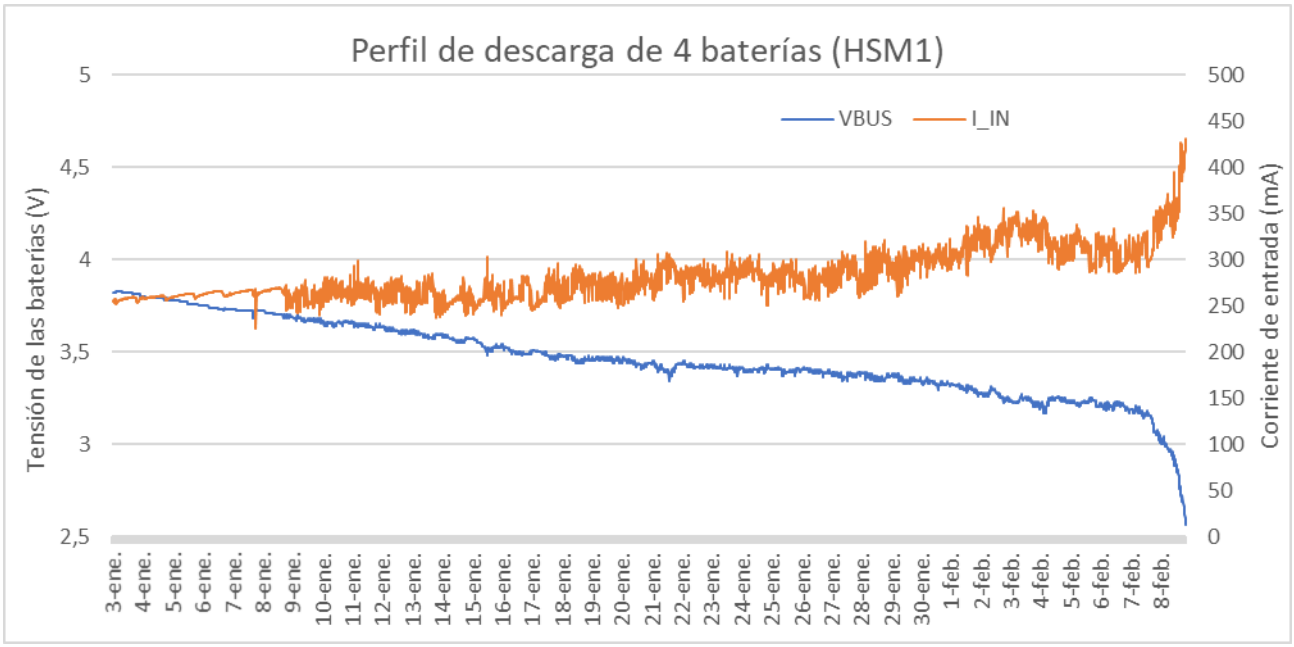


Figura 73. Perfil de descarga de 4 baterías en HSM1.

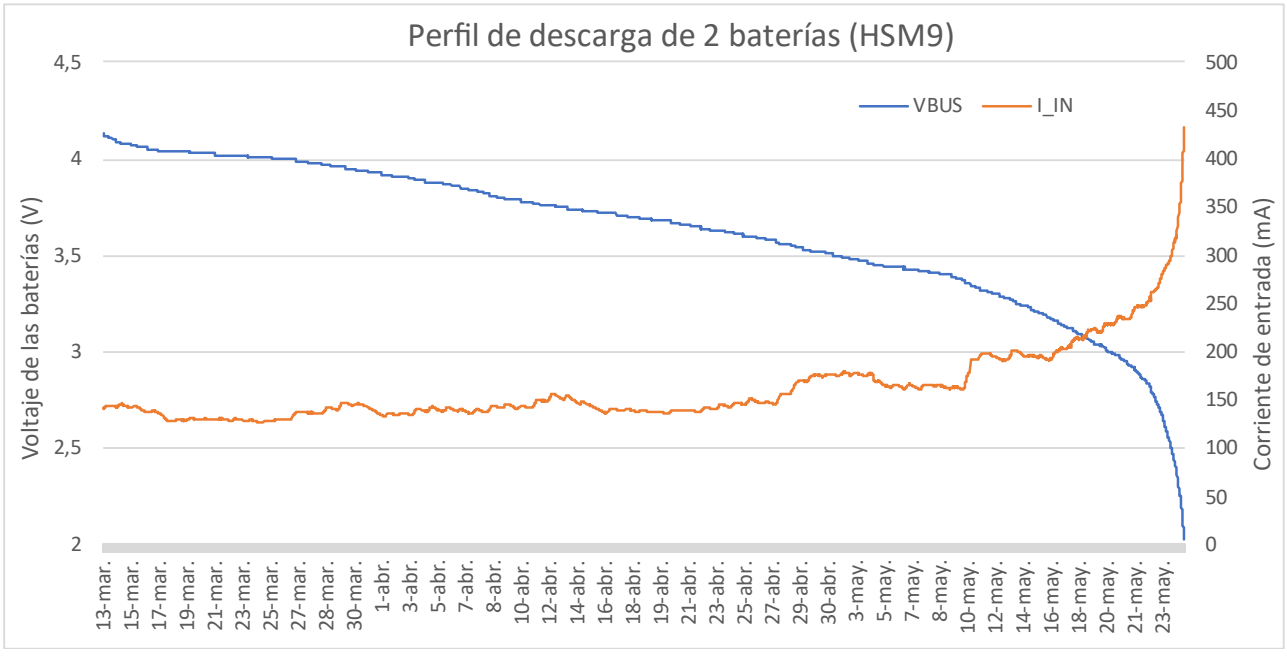


Figura 72. Perfil de descarga de 2 baterías en HSM9.

Como se puede observar en la *Figura 72* y la *Figura 73*, en los dos casos de 2 y 4 baterías, los transmisores se mantienen funcionando durante largos periodos de tiempo. Concretamente, el transmisor HSM1 se mantuvo durante 36 días en funcionamiento solar, y HSM9 estuvo 71 días alimentado sólo a través de las baterías, sin conexión a la red. Las diferencias se deben, claro está, en la cantidad de sensores que lleva cada uno, como se puede apreciar en los datos de corriente de entrada. A pesar de tener también la mitad de baterías, HSM9 comenzó la descarga en un estado de carga mayor (4,1 V contra 3,8 V). Además, un menor consumo de corriente permite descargas más profundas de las baterías, por lo que se aprovecha mejor la energía.

6.5.3. Caracterización del porcentaje de carga

Estas gráficas de descarga nos permiten extraer una información interesante que no siempre es fácil de obtener: la curva del porcentaje de carga de las baterías. Por un lado, tenemos el valor de la tensión de las baterías en cada una de las medidas de todo un ciclo de descarga. Tomando la aproximación de que todas las medidas se toman en intervalos de tiempo constante y que en todas las medidas de consume la misma energía, se puede considerar cada medida como un porcentaje del total del ciclo de descarga, siendo el 100% la carga inicial y el 0% cuando se toma la última medida antes de apagarse completamente. Con estas premisas, se puede crear la gráfica de la *Figura 74*, que describiría el porcentaje de medidas que se han realizado respecto del total para cada valor de tensión. Así se obtiene entonces una relación entre el porcentaje de carga de la batería y su voltaje.

Figura 74. Gráfica del nivel de carga respecto de la tensión de la batería.

A partir de aquí, esta curva puede interpretarse de diferentes formas para extraer el cálculo del porcentaje de carga. Al fin y al cabo, no es más que un valor orientativo. En nuestro caso, se ha dividido la curva en 4 segmentos lineales (aunque se podría dividir en más) para representar la zona de trabajo lineal de la batería, por lo que se obtendría la siguiente formulación, siendo *x* la tensión de la batería e *y* el nivel de carga:

$$\begin{Bmatrix} \forall x \in [4 \quad \infty) \rightarrow & y = 100 \\ \forall x \in [3{,}1 \quad 4) \rightarrow & y = 100x - 300 \\ \forall x \in [2{,}7 \quad 3{,}1) \rightarrow & y = 25x - 67{,}5 \\ \forall x \in (-\infty \quad 2{,}7) \rightarrow & y = 0 \end{Bmatrix} \tag{3}$$

6.6. Capacidades de conexión distribuida

Una característica muy destacable sobre el sistema HSM consiste en que es totalmente escalable. A su configuración actual, de 11 transmisores y un receptor en Benicalap, se le suma un segundo receptor en el laboratorio del Instituto de Ingeniería Energética en el CPI. Este, a su vez, recibe mensajes continuamente de un número indeterminado de transmisores tanto colgados en la UPV como en pruebas dentro del propio laboratorio. Este esquema de receptores y transmisores puede escalarse de forma indefinida, tan solo es necesario crear nuevas tablas en la base de datos para alojar las mediciones que se requiera.

Pero estos no son los únicos dispositivos que pueden conectarse a la base de datos. El hecho de usar Arduino como intermediario para la transmisión de datos al computador Raspberry Pi tiene una ventaja importante, y es que cualquier dispositivo compatible con Arduino puede adaptarse a este sistema. Esto lo vuelve una herramienta muy versátil, puesto que casi cualquier circuito electrónico con cualquier tipo de sensor podría conectarse a un microcontrolador Arduino por cualquiera de las interfaces de las que dispone. Ya sea con transmisión por radio o directamente al receptor, enviando la trama de datos adecuada, la Raspberry Pi reconocería sin problema los datos. Y no solo eso, cualquier dispositivo capaz de realizar peticiones MySQL a través de internet podría escribir en la base de datos actuando como si fuera un receptor, lo que amplía enormemente las posibilidades de este sistema.

Figura 75. Montaje de transmisor con Arduino UNO y analizador de redes.

Estas capacidades se demostraron en el laboratorio durante la fase de diseño. El proyecto GrowGreen Valencia propuso la instalación de un analizador de redes Siemens Sentron Pac3200 con el objetivo de monitorizar la red eléctrica en el centro de mayores y el Pabellón Deportivo de Benicalap. La intención era estudiar patrones de consumo eléctrico en estos lugares para detectar posibles efectos relacionados con las acciones del proyecto GrowGreen como, por ejemplo, un posible ahorro energético en el consumo de aire acondicionado por la instalación de la cubierta ajardinada.

Finalmente, este sistema no llegó a instalarse, pero el desarrollo se completó y se probó con éxito. El montaje consistía en una placa Arduino UNO con un *shield* Ethernet y conectado por cable Ethernet al analizador de redes. El Arduino enviaba las peticiones y recibía los parámetros solicitados a través del protocolo Modbus TCP y finalmente formateaba y enviaba los datos por radio como cualquier otro transmisor.

6.7. Uso de los datos en investigación

Como se ha podido comprobar, el sistema HSM es capaz de entregar medidas de todas las variables necesarias, con la calidad y la frecuencia requeridas. De hecho, todas las medidas que el sistema ha tomado hasta ahora ya han servido para realizar algunas publicaciones científicas con diversos objetivos. Estas son las más destacadas:

- Alfonso-Solar, D., Bastida-Molina, P., Montuori, L., Vargas-Salgado, C. (2019). ***Monitoring and evaluation of thermal comfort in urban areas: application to Valencia city***. CARPE Conference 2019: Horizon Europe and beyond. [1]

Este artículo realiza una valoración del estrés térmico a partir del indicador PET. Utiliza 6 meses de datos obtenidos entre enero y junio de 2019, de dos localizaciones de Benicalap, correspondientes a los transmisores HSM4 y HSM1. Para ello, se ha usado un software de modelado llamado RAYMAN para obtener el índice PET y establecer el porcentaje de horas de confort térmico al mes que se detectan en cada una de las localizaciones (una siendo una zona despejada de alta insolación y la otra una zona con vegetación). Para ello se ha usado la tabla de referencia de la *Figura 76*, que relaciona el valor del índice PET con unos valores de percepción térmica. [9]

Max.	Min.	Thermal perception
	>41	Extreme heat stress
41	35	Strong heat stress
35	29	Moderate heat stress
29	23	Slight heat stress
23	18	No termal stress
18	13	Slight cold stress
13	8	Moderate cold stress
8	4	Strong cold stress
	<4	Extreme cold stress

Figura 76. Percepción térmica en tramos de PET. Fuente: (Matzarakis et al.,1999).

Además, usa los datos del PET para compararlos con los de la temperatura de globo negro con la intención de probar la viabilidad de usar un sensor de temperatura de globo negro como una forma fiable y de bajo coste de obtener el índice PET. Las conclusiones del estudio indican que existe una fuerte relación lineal entre el índice PET y la temperatura de globo negro, con una desviación estándar de 1,5 grados.

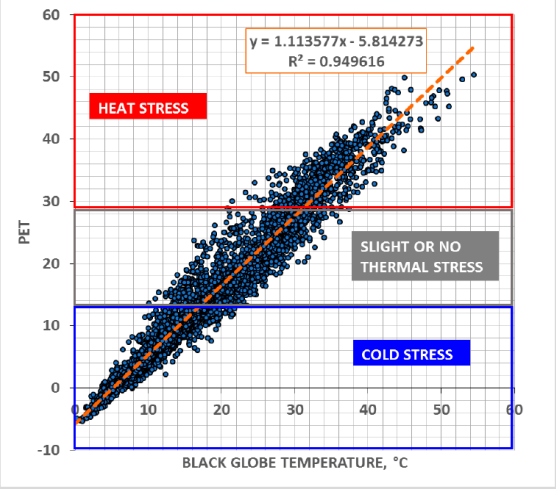


Figura 77. Regresión lineal entre el índice PET y la temperatura de globo negro. Fuente: (Alfonso-Solar, D. et al., 2019).

- Vargas-Salgado, C., Chiñas-Palacios, C., Aguila-León, J. and Alfonso-Solar, D. (2019). ***Measurement of the black globe temperature to estimate the MRT and WBGT indices using a smaller diameter globe than a standardized one: Experimental analysis.*** Proceedings 5th CARPE Conference: Horizon Europe and beyond. [11]

Como ya se ha comentado en el *Apartado 5*, este estudio analiza la viabilidad de usar un sensor de temperatura de globo negro de 9 cm de diámetro en lugar del estándar de 15 cm que marca (UNE ISO 7726, 2002), [9] así como observar las posibles desviaciones que puedan aparecer en el cálculo de la temperatura radiante media. Las conclusiones del trabajo muestran que es posible usar un diámetro de globo negro más pequeño, con un margen de error aceptable. Y todo este estudio se basa en los datos que ha recolectado el sistema.

6.8. Cálculo de los indicadores de estrés térmico

Con estos datos que hemos obtenido a lo largo de la fase de operación, en que los transmisores han estado en funcionamiento en Benicalap, se pueden obtener los indicadores del estrés térmico que estábamos buscando: la temperatura radiante media (MRT), el índice de temperatura de globo y bulbo húmedo (WBGT) y el índice PET (*Physiological Equivalent Temperature*). Con estos valores, seremos capaces de obtener información acerca del estrés térmico.

6.8.1. Temperatura de globo negro

Así pues, para concluir, realizaremos el cálculo de los principales indicadores de estrés térmico a partir de las mediciones de la base de datos. Partiremos de los datos de temperatura de globo negro (de diámetro 9 cm) captados por uno de los transmisores los días 4 y 5 de junio de 2019, representando un ejemplo de 48 horas típicas.

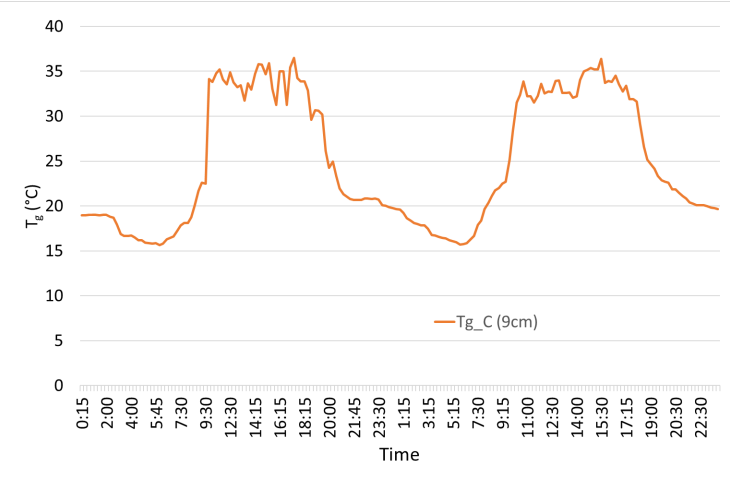


Figura 78. Gráfico de temperatura de globo negro a lo largo de 48 horas.

6.8.2. Cálculo de la temperatura radiante media (MRT)

Para calcular la MRT, podemos utilizar la Ecuación 4 que se indicó en el *Apartado 1.3.: Conceptos teóricos*. Sin embargo, recordaremos que esa ecuación es adecuada para un globo negro de 15 cm de diámetro, y que el diámetro habitual de los transmisores HSM es de 9 cm. Por tanto, se usará la Ecuación 6 según (UNE ISO 7726, 2002), [11] para convección forzada cuando el diámetro del globo es inferior a 15 cm:

$$\bar{t}_r = \left[\left(t_g + 273\right)^4 + \frac{1{,}1 \cdot 10^8 \cdot \left(w_{speed}\right)^{0.6} \cdot \left(t_g - t_a\right)}{\varepsilon_g \cdot D^{0.4}} \right]^{1/4} - 273 \quad (6)$$

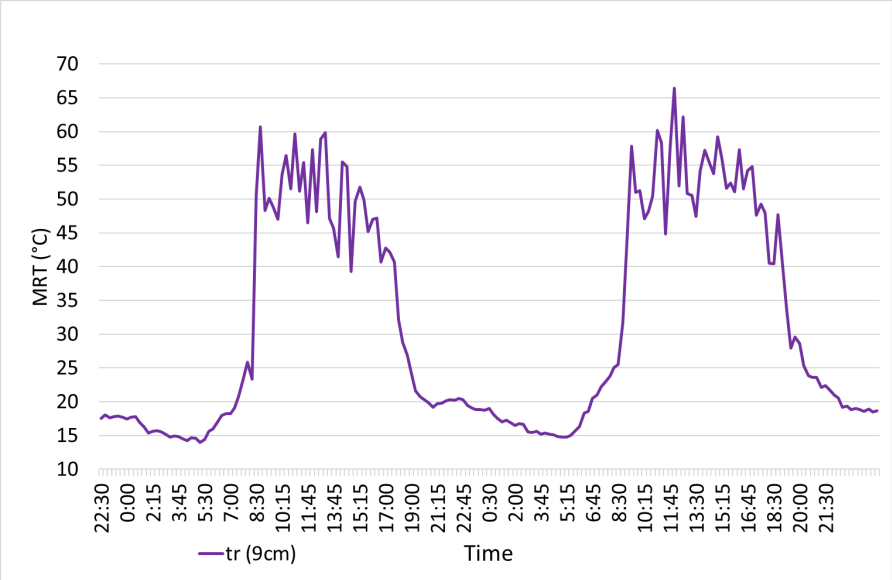


Figura 79. Gráfico de temperatura radiante media a lo largo de 48 horas.

6.8.3. Cálculo del índice WBGT

Para el cálculo de la temperatura de globo y bulbo húmedo (WBGT), usaremos las Ecuaciones 1, 2 y 3 descritas en el *Apartado 1.3: Conceptos teóricos.*

$$WBGT_{outdoor} = 0{,}7t_{nwb} + 0{,}2t_g + 0{,}1t_a \quad (1)$$

$$t_{nwb} = t_w + 0{,}0021S - 0{,}42w_{speed} + 1{,}93 \quad (2)$$

$$t_w = t_a \cdot \tan^{-1}\left[0{,}151977(RH\% + 8{,}131659)^{1/2}\right] + \tan^{-1}(t_a + RH\%) - \tan^{-1}(RH\% - 1{,}676331) + 0{,}00391838(RH\%)^{\frac{3}{2}} \cdot \tan^{-1}(0{,}023101RH\%) - 4{,}686035 \quad (3)$$

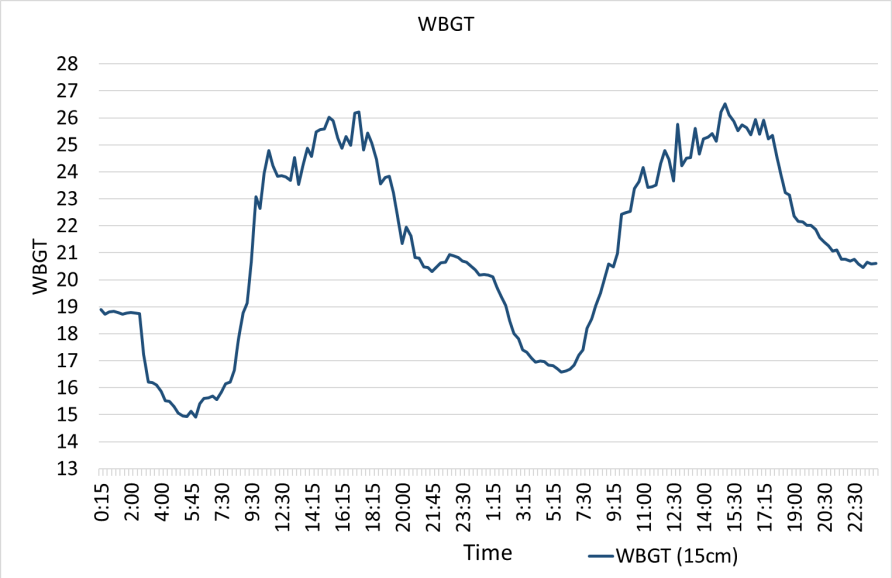


Figura 80. Gráfica del índice WBGT durante 48 horas.

6.8.4. Cálculo del índice PET

Finalmente, calcularemos el índice PET. Para ello, usaremos la ecuación descrita en (Alfonso-Solar, D. et al., 2019). Como ya hemos comentado en el *Apartado 6.7*, este estudio describe una correlación lineal entre la temepratura de globo negro y el índice PET (véase *Figura 77*). Así pues, hemos usado esta ecuación para calcular el índice PET en ese intervalo de 48 horas.

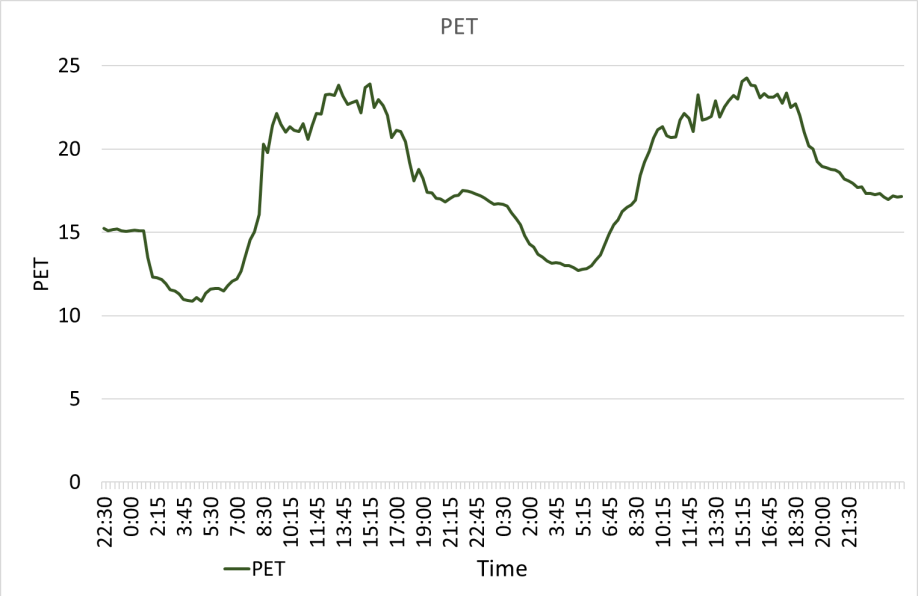


Figura 81. Gráfica del índice PET durante 48 horas.

Además, se ha realizado un estudio del índice PET a partir de las medidas de ese transmisor para el intervalo de tiempo entre 13 de febrero y el 5 de julio. Así, luego se ha establecido una clasificación con el porcentaje de tiempo en que esa localización se mantiene en diferentes niveles de estrés térmico, usando los mismos valores que en la *Figura 76*, agrupados en tres bloques:

Percepción térmica	Min	Max	Valores	%
Extreme/Strong/Moderate heat stress	29		32	**0.4%**
Slight/Slight heat stress	13	35	4937	**55.6%**
Cold Stress		13	3903	**44.0%**
		Total	8872	100%

Valor máximo de índice PET	30.72
Valor mínimo de índice PET	-0.48

7. Presupuesto

A continuación, se va a proceder a detallar el coste aproximado del proyecto. Se calculará el coste base de componentes, herramientas y mano de obra correspondientes a la fabricación de 11 dispositivos transmisores y un receptor, es decir, el número de dispositivos que están actualmente instalados en Benicalap. Por otro lado, también se calculará el coste total del proyecto añadiendo el tiempo aproximado de mano de obra durante la fase de diseño, aunque no se contará el coste por componentes invertidos en el desarrollo del proyecto. Así pues, se va a desglosar el coste en diferentes categorías:

- **Electrónica del dispositivo transmisor:** Se incluyen en esta categoría los componentes electrónicos y materiales utilizados en la fabricación de un dispositivo transmisor completo, con todos los sensores que pudieran instalarse, incluyendo cableado, sin contar elementos mecánicos o estructurales.

Descripción del artículo	Uds.	Precio unitario	Precio total
Anemómetro de copas 0-5V	1	32,32 €/ud.	32,32 €
Antena 433 MHz	1	1,05 €/ud.	1,05 €
Batería de litio recargable 18650 3,7 V	4	3,99 €/ud.	15,96 €
Bornera de tornillo 2 pines 5,05 mm THT	6	0,12 €/ud.	0,72 €
Célula solar calibrada Cebekit C-0121	2	33,92 €/ud.	67,84 €
Conector enchufable PCB Molex 2,54 THT hembra	11	0,16 €/ud.	1,76 €
Conector enchufable PCB Molex 2,54 THT macho	11	0,16 €/ud.	1,76 €
Conectores de alimentación macho/hembra (par)	5	0,04 €/ud.	0,20 €
Diodo Shottky THT Vf < 0.3 V	1	0,03 €/ud.	0,03 €
Interruptor DIP-switch de 8 contactos THT	1	0,69 €/ud.	0,69 €
Módulo convertidor de tensión de 5 a 12 V	1	0,60 €/ud.	0,60 €
Módulo convertidor de tensión VMA-403	1	3,28 €/ud.	3,28 €
Módulo de carga VMA-321	1	3,28 €/ud.	3,28 €
Módulo de radiofrecuencia E32-TTL-1W	1	5,27 €/ud.	5,27 €
Módulo de sensor de presión BME280	1	1,91 €/ud.	1,91 €
Módulo sensor de corriente INA219	3	0,77 €/ud.	2,31 €
Módulo sensor de corriente INA3221	1	1,95 €/ud.	1,95 €
Módulo temporizador TPL5110	1	4,10 €/ud.	4,10 €
Panel solar 165x135 mm 6 V	1	7,26 €/ud.	7,26 €
Placa de circuito impreso GrowGreen v2	1	0,80 €/ud.	0,80 €
Placa de desarrollo ATMega2560	1	31,54 €/ud.	31,54 €
Portapilas de 4 huecos para pilas 18650	1	2,50 €/ud.	2,50 €
Regleta de pines hembra THT 2,54 mm recto	10	0,08 €/ud.	0,80 €
Regleta de pines macho THT 2,54 mm angulares	1	0,18 €/ud.	0,18 €
Regleta de pines macho THT 2,54 mm recto	4	0,18 €/ud.	0,72 €
Sensor de lluvia MH-RD	1	0,40 €/ud.	0,40 €
Sensor de temperatura DS18B20 encapsulado	2	0,88 €/ud.	1,76 €
Sensor de temperatura y humedad DHT22	1	2,30 €/ud.	2,30 €
Set de cables Dupont 20 cm	1	2,99 €/ud.	2,99 €
Set de resistencias THT ±1%	1	0,50 €/ud.	0,50 €
Coste adicional por utilización de herramientas		5%	9,84 €
Coste adicional por extras y pérdidas de fabricación		5%	9,84 €
		Subtotal	**216,46 €**

- **Elementos mecánicos del dispositivo transmisor:** En esta categoría se incluyen los elementos mecánicos y de sujeción utilizados en el dispositivo transmisor, como la tornillería o el globo negro, sin contar las piezas de la estructura impresas en 3D.

Descripción del artículo	Uds.	Precio unitario	Precio total
Abrazadera isofónica reforzada M8-10 Ø90	1	4,30 €/ud.	4,30 €
Arandela autoblocante M10 DIN 6795-A	2	0,03 €/ud.	0,06 €
Arandela plana M10 DIN 9021	2	0,12 €/ud.	0,24 €
Arandela plana M3 DIN 9021	4	0,03 €/ud.	0,12 €
Barniz transparente para exteriores	0,2	5,10 €/ud.	1,02 €
Cúpula de metacrilato transparente Ø160	1	2,00 €/ud.	2,00 €
Esfera de latón Ø80	1	5,00 €/ud.	5,00 €
Pintura en spray blanco mate	0,2	2,95 €/ud.	0,59 €
Pintura en spray negro mate	0,05	3,75 €/ud.	0,19 €
Tornillo autoblocante DIN 7504-N Ø4,2 25 mm	8	0,05 €/ud.	0,40 €
Tornillo de estrella M3 DIN 965 30 mm	4	0,03 €/ud.	0,12 €
Tornillo M10 DIN 933 100 mm	2	0,41 €/ud.	0,82 €
Tuerca M10 DIN 934	4	0,17 €/ud.	0,68 €
Tuerca M3 DIN 934	4	0,03 €/ud.	0,12 €
Varilla de aluminio hueca Ø6	0,392 m	2,10 €/m	0,82 €
Varilla de aluminio hueca Ø8	0,212 m	2,10 €/m	0,45 €
Coste adicional por utilización de herramientas		5%	0,85 €
Coste adicional por extras y pérdidas de fabricación		5%	0,85 €
		Subtotal	**18,62 €**

- **Piezas impresas en 3D:** Aquí se detalla el precio aproximado de las piezas impresas en 3D. Se va a tomar como referencia el precio por gramo de una bobina de filamento de 1 kg, y el peso de filamento que indica el archivo de impresión del software Cura.

Descripción de la pieza	Material	Duración	Longitud	Peso	Uds.	Precio unitario	Precio total
Caja nueva	PLA	13h 04'	103,77 m	309 g	1	0,02 €/g	6,18 €
Disco	PLA	0h 21'	2,75 m	8 g	1	0,02 €/g	0,16 €
Palo	PLA	1h 08'	9,11 m	27 g	1	0,02 €/g	0,54 €
Plato inferior	PLA	0h 29'	3,97 m	11 g	4	0,02 €/g	0,88 €
Plato superior	PLA	0h 36'	4,64 m	14 g	1	0,02 €/g	0,28 €
Reja	PLA	0h 05'	0,63 m	2 g	1	0,02 €/g	0,04 €
Rosca tapa	PLA	0h 31'	3,37 m	10 g	1	0,02 €/g	0,20 €
Soporte anemo. tapado	PLA	0h 37'	4,33 m	13 g	1	0,02 €/g	0,26 €
Soporte cúpula	PLA	0h 52'	7,22 m	22 g	1	0,02 €/g	0,44 €
Tapa simple nueva	PLA	5h 20'	45,63 m	136 g	1	0,02 €/g	2,72 €
Coste herramientas						5%	0,59 €
Coste extra y pérdidas						5%	0,59 €
						Subtotal	**12,87 €**

- **Componentes del dispositivo receptor y comunicaciones:** En esta parte se van a incluir todos los elementos utilizados en la fabricación del dispositivo receptor, incluyendo tanto electrónica y cableado, como elementos mecánicos, tornillería y los costes de los servicios de telecomunicaciones.

Descripción del artículo	Uds.	Precio unitario	Precio total
Cable USB-A a USB-B macho-macho 0,3 m	1	2,33 €/ud.	2,33 €
Módem 3G USB Huawei E1750	1	15,52 €/ud.	15,52 €
Módulo de radiofrecuencia E32-TTL-1W	1	5,27 €/ud.	5,27 €
Placa de desarrollo ATMega328 UNO	1	14,26 €/ud.	14,26 €
Placa de desarrollo Raspberry Pi 3B	1	38,51 €/ud.	38,51 €
Set de cables Dupont 20 cm	1	2,99 €/ud.	2,99 €
Set de espaciadores metálicos M3 para PCB	1	6,85 €/ud.	6,85 €
Tarjeta SIM con tarifa mensual de datos	48	1,00 €/mes	48,00 €
Teclado inalámbrico RF con touchpad	1	11,99 €/ud.	11,99 €
Touchscreen Display 7" para Raspberry Pi	1	69,90 €/ud.	69,90 €
Transformador de alimentación 5 V 2,5 A micro USB	1	8,99 €/ud.	8,99 €
Coste adicional por utilización de herramientas		5%	11,23 €
Coste adicional por extras y pérdidas de fabricación		5%	11,23 €
		Subtotal	**247,07 €**

- **Mano de obra:** Finalmente, se va a calcular el coste correspondiente a la mano de obra de un operario para la fabricación, por un lado, de un dispositivo transmisor y, por otro lado, de un receptor. De nuevo, se trata de una estimación, ya que con fabricación en serie se reducirían los tiempos de dedicación.

Dispositivo transmisor			
Concepto	**Duración**	**Salario bruto**	**Coste total**
Preparación y uso de la impresora 3D	0,5 h	10,00 €/h	5,00 €
Tratamiento de las piezas impresas en 3D	1,0 h	10,00 €/h	10,00 €
Preparación de componentes	0,5 h	10,00 €/h	5,00 €
Montaje de la estructura del dispositivo transmisor	2,0 h	10,00 €/h	20,00 €
Montaje y soldadura del circuito electrónico	2,0 h	10,00 €/h	20,00 €
Cableado del dispositivo transmisor	2,0 h	10,00 €/h	20,00 €
Programación y puesta en funcionamiento	0,5 h	10,00 €/h	5,00 €
Comprobaciones finales	0,5 h	10,00 €/h	5,00 €
Instalación	0,5 h	10,00 €/h	5,00 €
Coste adicional por tareas extra y mantenimiento		10%	4,75 €
		Subtotal	**99,75 €**
Dispositivo receptor			
Concepto	**Duración**	**Salario bruto**	**Coste total**
Preparación de componentes	0,5 h	10,00 €/h	5,00 €
Programación del dispositivo receptor	1,0 h	10,00 €/h	10,00 €
Montaje del dispositivo receptor	4,0 h	10,00 €/h	40,00 €
Puesta en funcionamiento y comprobaciones finales	0,5 h	10,00 €/h	5,00 €
Instalación	0,5 h	10,00 €/h	5,00 €
Coste adicional por tareas extra y mantenimiento		10%	3,25 €
		Subtotal	**68,25 €**

- **Coste final de fabricación de los prototipos:** Una vez establecidos los costes de cada una de las categorías, se calculará el coste final aproximado del proyecto, contando tan solo la fabricación de los 11 prototipos de transmisor y un receptor.

Concepto	Uds.	Coste unitario	Coste total
Electrónica del dispositivo transmisor	11	216,46 €/ud.	2381,04 €
Elementos mecánicos del dispositivo transmisor	11	18,62 €/ud.	204,80 €
Piezas impresas en 3D	11	12,87 €/ud.	141,57 €
Mano de obra del dispositivo transmisor	11	149,63 €/ud.	1097,25 €
Componentes del dispositivo receptor y comunicaciones	1	247,07 €/ud.	247,07 €
Mano de obra del dispositivo receptor	1	102,38 €/ud.	68,25 €
		TOTAL	**4131,98 €**

- **Mano de obra de la etapa de diseño del sistema:** También se va a realizar una estimación del coste de mano de obra empleada durante la fase de diseño y desarrollo del sistema. Se ha tomado como referencia un sueldo base mensual para 20 horas semanales durante 12 meses.

Concepto	Duración	Salario bruto	Coste total
Diseño y desarrollo de los prototipos (20 h/semana)	12 meses	750,00 €/mes	9000,00 €
		TOTAL	**9000,00 €**

8. Conclusiones

En conclusión, podemos decir que este trabajo ha conseguido alcanzar los objetivos que se había planteado en el inicio del mismo.

Se ha logrado diseñar un sistema replicable y de bajo coste, del cual el equipo del proyecto posee toda la propiedad intelectual y todos los datos de diseño, por lo que puede realizar modificaciones y actualizaciones fácilmente sin necesidad de acudir a terceros. Además, utiliza componentes y tecnologías de baja complejidad, por lo que puede ser fabricado y replicado fácilmente por otros técnicos sin necesidad de una formación excesivamente profunda.

Por otro lado, cumple las especificaciones de diseño en cuanto a variables de medida, frecuencia de medición de datos y almacenamiento en base de datos. El almacenamiento en la nube permite un acceso sencillo a los datos de forma que se pueden exportar a voluntad para su posterior análisis.

Los dispositivos instalados, tras sucesivas iteraciones de diseño y actualizaciones, finalmente han conseguido ser capaces, generalmente, de trabajar tanto en intemperie como en interiores durante largas temporadas bajo condiciones de fuerte radiación solar, lluvia intensa y condiciones de poca luminosidad y, a día de hoy, desde su instalación a principios de 2019, siguen en su mayoría funcionando correctamente. Se espera que los próximos repuestos de transmisores funcionen ininterrumpidamente hasta la finalización del proyecto en mayo de 2022. Por su lado, el dispositivo receptor instalado en octubre de 2018 sigue funcionando hasta hoy con poco o nulo mantenimiento.

El sistema diseñado en este trabajo ha permitido recolectar importantes datos relativos a indicadores del estrés térmico y a las condiciones meteorológicas del barrio de Benicalap. Actualmente, estos datos se encuentran en estudio y se está analizando el impacto que las acciones realizadas en la zona han tenido sobre el clima y el bienestar de los ciudadanos, así como el ahorro energético y la gestión de los recursos. Hasta el momento, los datos que los HSM han recolectado ya han sido utilizados para llevar a cabo distintas publicaciones científicas.

Así, este trabajo ha permitido apoyar el desarrollo teórico y científico de estas soluciones inspiradas en la naturaleza como una alternativa viable para mejorar la calidad de vida en las ciudades. De esa forma, se puede alcanzar una gestión sostenible de los recursos naturales para tratar de hacer frente a los retos que el cambio climático nos planteará en los próximos años.

9. Referencias

[1] Alfonso-Solar, D., Bastida-Molina, P., Montuori, L., Vargas-Salgado, C. (2019). *Monitoring and evaluation of thermal comfort in urban areas: application to Valencia city*. CARPE Conference 2019: Horizon Europe and beyond. [online] Disponible en: dx.doi.org/10.4995/CARPE2019.2019.10198 [Consulta: 21/11/2020].

[2] Aosong Electronics Co., Ltd. *AM2302 Product Manual.* [online] Disponible en: robotchip.ru/download/datasheet/AM2302-Datasheet.pdf [Consulta: 21/11/2020].

[3] Ayuntamiento de Valencia. (2020). *El Ayuntamiento instala una cubierta ajardinada en el centro de mayores de Benicalap para mejorar "la salud y el medio ambiente"*. Disponible en: valencia.es/valencia/noticias/NOTICIA_071735 [Consulta: 22/11/2020].

[4] GrowGreen. (2017). *Growgreen project website*. [online] Disponible en: growgreenproject.eu [Consulta: 21/11/2020].

[5] GrowGreen. (2017). *Growgreen Valencia*. [online] Disponible en: growgreenproject.eu/city-actions/valencia [Consulta: 21/11/2020].

[6] IPCC. (2013). *Climate Change 2013: The Physical Science Basis. Summary for Policymakers: A report of Working Group I of the IPCC.* [Consulta: 21/11/2020].

[7] Las Provincias. (2020). *Un nuevo bosque para Valencia.* [news] Disponible en: lasprovincias.es/valencia-ciudad/nuevo-bosque-valencia-20200906001732-ntvo.html [Consulta: 22/11/2020].

[8] Levante – El mercantil valenciano. (2019). *El colegio verde de Benicalap: un jardín vertical y agua reutilizada para regar*. [news] Disponible en: levante-emv.com/valencia/2019/11/20/colegio-verde-benicalap-jardin-vertical-13725635.html [Consulta: 22/11/2020].

[9] Matzarakis, A., Mayer, H., Iziomon, M.G., (1999). *Applications of a universal thermal index: physiological equivalent temperature*. Int. J. Biometerol. 43, 76–84.

[10] Reyes Tovar, AF. (2019). *Análisis del estrés térmico basado en mediciones locales y simulación: Aplicación al barrio de Benicalap de la ciudad de Valencia.* [online] Disponible en: hdl.handle.net/10251/128447 [Consulta: 22/11/2020].

[11] UNE ISO 7726, U. I. (2002). Norma española UNE ISO 7726.

[12] Vargas-Salgado, C., Chiñas-Palacios, C., Aguila-León, J. and Alfonso-Solar, D. (2019). *Measurement of the black globe temperature to estimate the MRT and WBGT indices using a smaller diameter globe than a standardized one: Experimental analysis.* Proceedings 5th CARPE Conference: Horizon Europe and beyond. [online]. Disponible en: hdl.handle.net/10251/132198 [Consulta: 22/11/2020].

[13] Vargas-Salgado, C., Montuori, L., Bastida-Molina, P., Alfoso-Solar, D. (2019). *Arduino-based prototype to estimate heat stress indices in urban environments.* [online] Disponible en: dx.doi.org/10.4995/CARPE2019.2019.10199 [Consulta: 22/11/2020].

[14] Wikipedia, la enciclopedia libre. (2020). *Estrés Térmico*. [online] Disponible en: es.wikipedia.org/wiki/Estrés_térmico [Fecha de consulta: 21/11/2020].

ANEXOS

ANEXO I: Esquema de circuito para PCB en EasyEDA

ANEXO II: Esquema de montaje completo en Fritzing de Transmisor v4.2

ANEXO III: Esquema de montaje simple en Fritzing de Transmisor v4.2

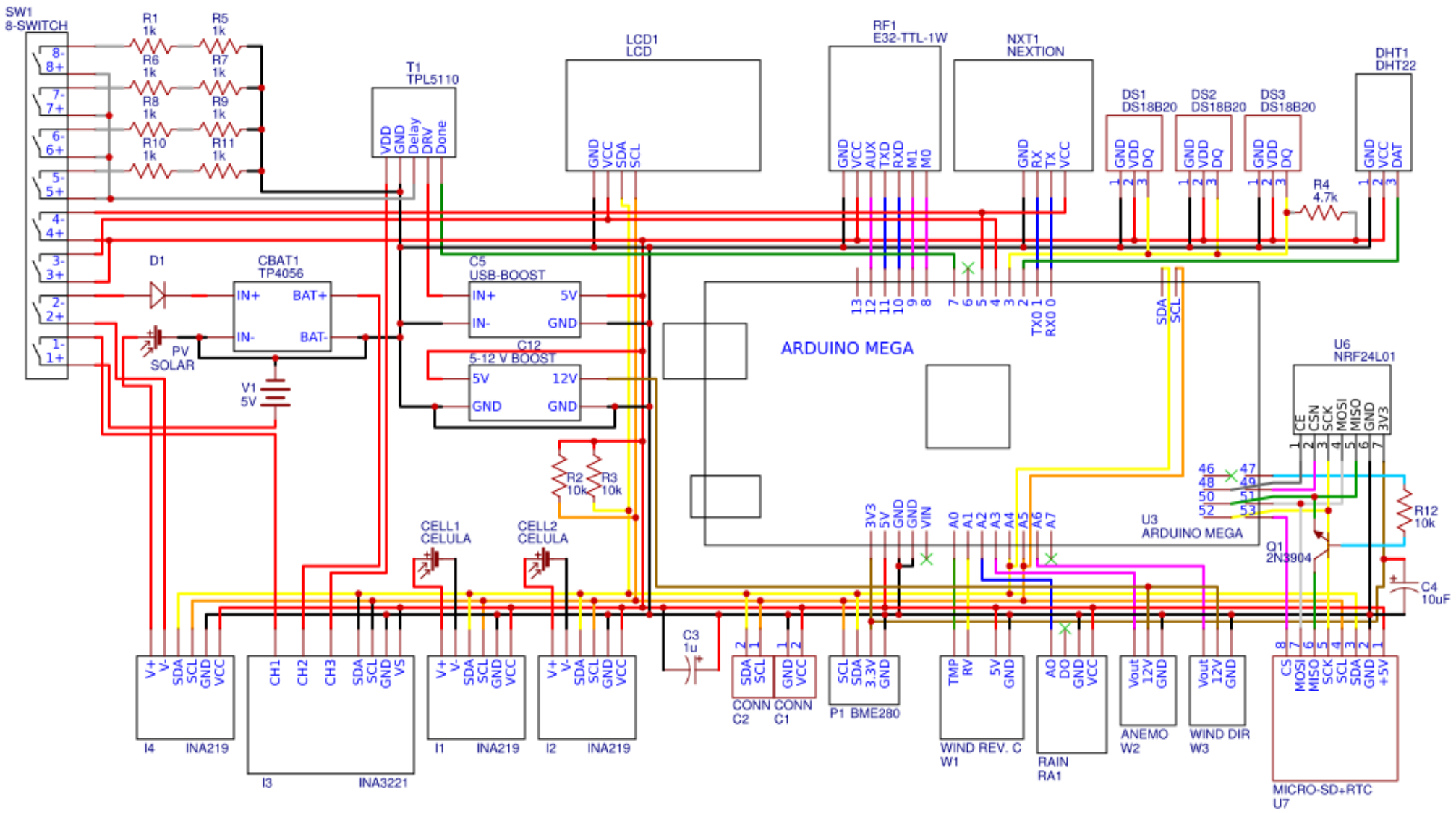
ANEXO IV: Código de Arduino para Transmisor v4.2

ANEXO V: Código de Arduino para Receptor v4.2

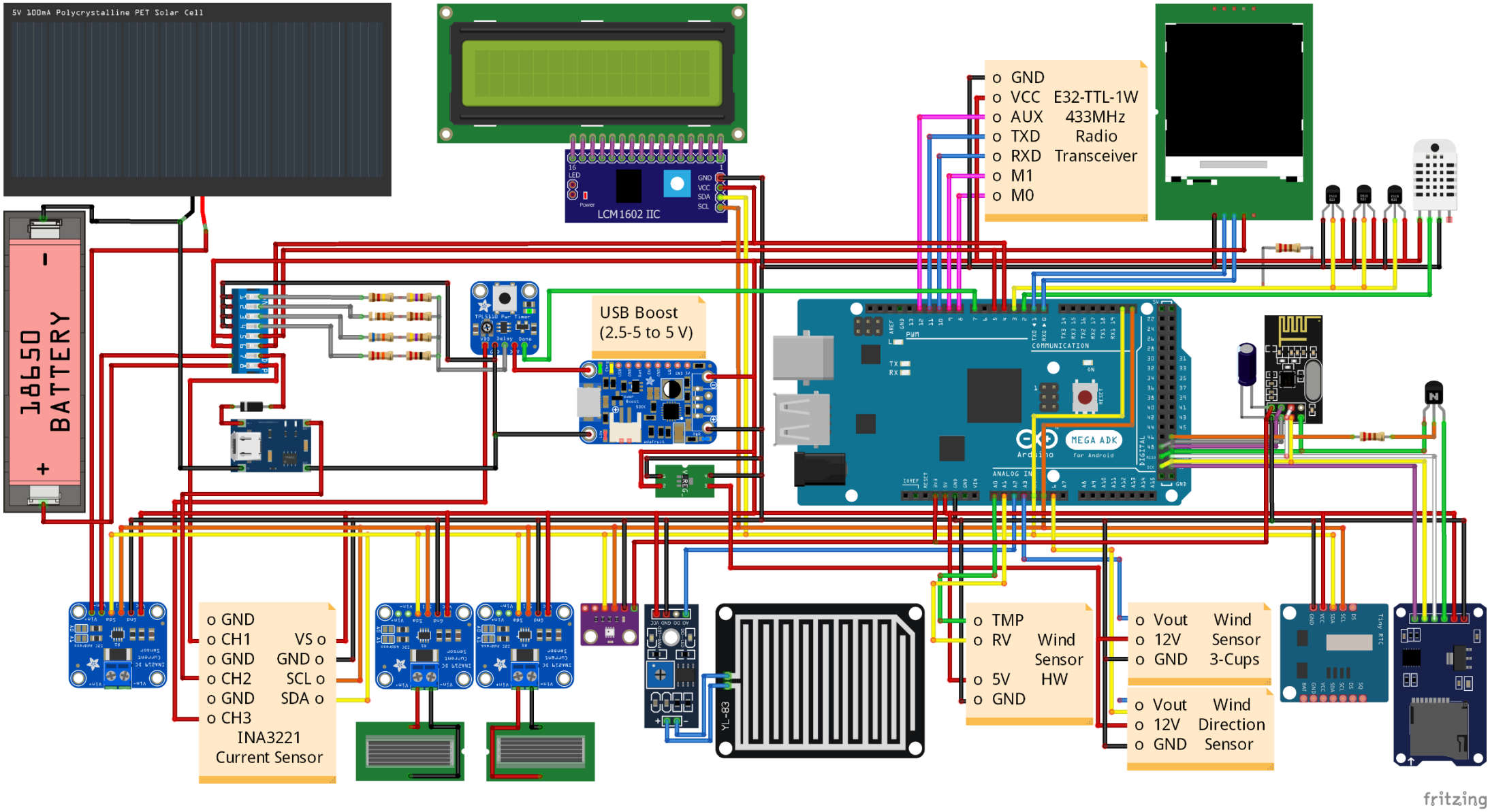
ANEXO VI: Código Python para tratamiento de datos en Raspberry Pi

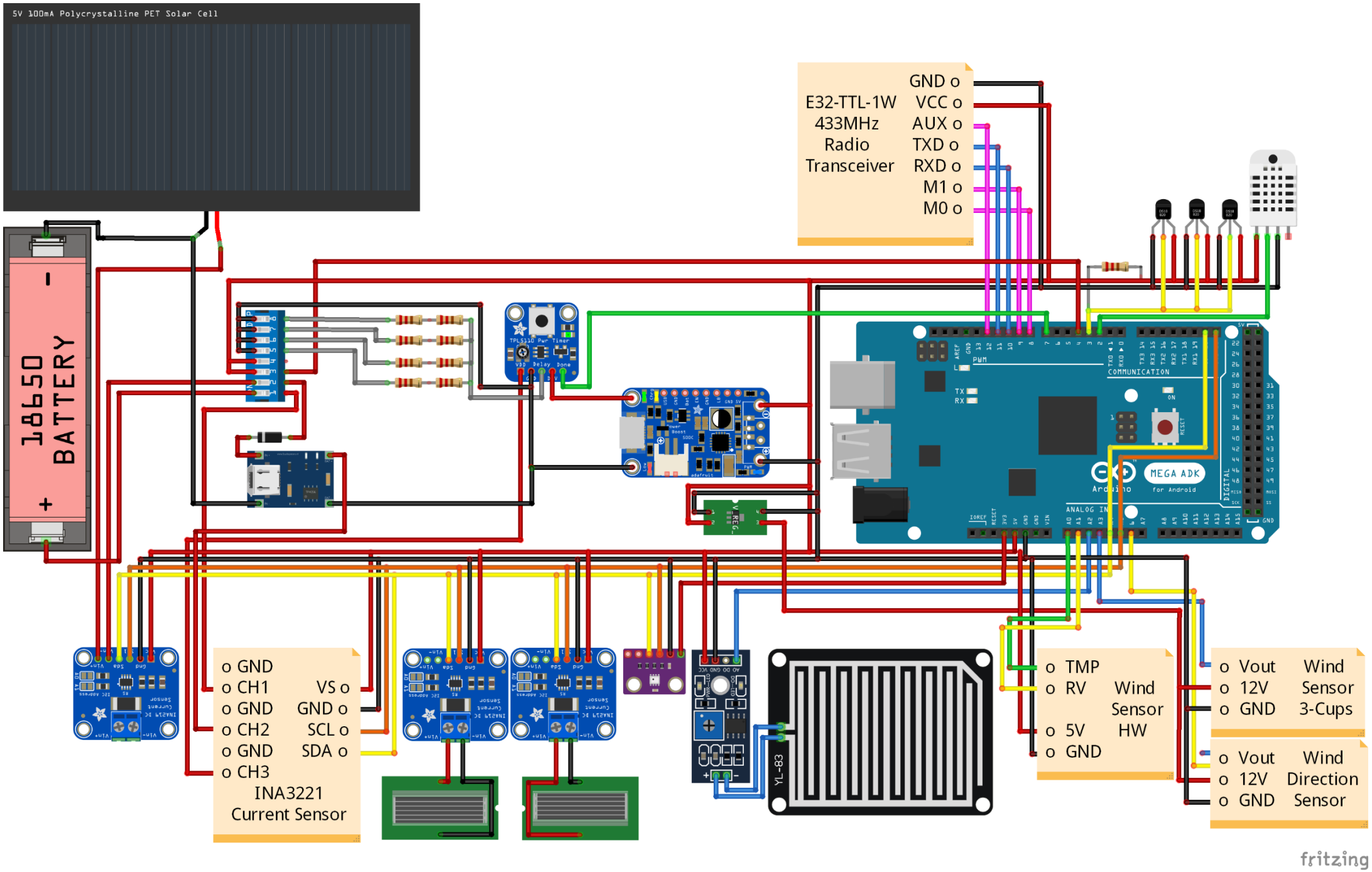
ANEXO VII: Código PHP de escritura en la Base de Datos en Plesk

ANEXO VIII: Manual de montaje, soldadura y programación de Transmisor



TITLE: Esquemático Transmitter 4 v2		REV: 1.0
Date: 2018-05-24		Sheet: 1/1
EasyEDA V5.1.3	Drawn By: Àlex Sánchez	


5V 100mA Polycrystalline PET Solar Cell
LCM1602 IIC
GND
VCC
SDA
SCL
18650 BATTERY
+
-
USB Boost
(2.5-5 to 5 V)
o GND
o VCC E32-TTL-1W
o AUX 433MHz
o TXD Radio
o RXD Transceiver
o M1
o M0
MEGA ADK
for Android
o GND
o CH1 VS o
o GND GND o
o CH2 SCL o
o GND SDA o
o CH3
INA3221
Current Sensor
YL-83
o TMP
o RV Wind
Sensor
o 5V HW
o GND
o Vout Wind
o 12V Sensor
o GND 3-Cups
o Vout Wind
o 12V Direction
o GND Sensor
fritzing


5V 100mA Polycrystalline PET Solar Cell
18650 BATTERY
-
+
E32-TTL-1W
433MHz
Radio
Transceiver
GND o
VCC o
AUX o
TXD o
RXD o
M1 o
M0 o
MEGA ADK
for Android
COMMUNICATION
ANALOG IN
o GND
o CH1
o GND
o CH2
o GND
o CH3
VS o
GND o
SCL o
SDA o
INA3221
Current Sensor
YL-83
o TMP
o RV
o 5V
o GND
Wind
Sensor
HW
o Vout
o 12V
o GND
Wind
Sensor
3-Cups
o Vout
o 12V
o GND
Wind
Direction
Sensor
fritzing

ANEXO IV: Código de Arduino para Transmisor v4.2

```
/**
 ****************************************************************************************************************************************************
   PROYECTO GROWGREEN VALENCIA - TRANSMITTER v4.2
   (ARDUINO MEGA / ARDUINO UNO / E32-TTL-1W)

   @file    Transmitter_4.2_MEGA_E32.ino
   @date    21/05/2018
   @author  Carlos Vargas Salgado
            Alejandro Sánchez Aduna
 ****************************************************************************************************************************************************
   - DHT22 (Humidity - temperature sensor)
   Source: http://www.electroschematics.com/11291/arduino-dht22-am2302-tutorial-library/

   - DS18B20 (Digital temperature sensors)
   Source: https://create.arduino.cc/projecthub/TheGadgetBoy/ds18b20-digital-temperature-sensor-and-arduino-9cc806

   - Wind Sensor Rev.C (Hot wire anemometer)
   Source: https://moderndevice.com/product/wind-sensor/

   - Anemometer Kit (0-5V,3 Wind Cups And Circuit Module)
   Source: https://www.dfrobot.com/wiki/index.php/Wind_Speed_Sensor_Voltage_Type(0-5V)_SKU:SEN0170
   Vin require  9-24V

   - BME280 (Presure, barometer and temperature sensor)
   Source: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout.pdf
   Uses Wire.h for I2C operation and SPI.h for SPI operation. Library allows either I2C or SPI, so include both.

   - LCD_2004A + LCN1602(LCD de 20x4 I2C)
   Source: http://wiki.sunfounder.cc/index.php?title=I2C_LCD2004
   Conect SDA to A4 pin in arduino uno and to pin20 in mega2560
   Conect SCL to A5 pin in arduino uno and to pin21 in mega2560

   - INA219/INA3221 (Current Sensor)
   Source INA219:  https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ina219-current-sensor-breakout.pdf
   Source INA3221: https://tech.scargill.net/ina3221-triple-voltage-current-monitor/
   Up to 4 boards may be connected. Addressing is as follows:
   Board 0: Address = 0x40, Offset = B00000 (no jumpers required)
   Board 1: Address = 0x41, Offset = B00001 (bridge A0)
   Board 2: Address = 0x44, Offset = B00100 (bridge A1)
   Board 3: Address = 0x45, Offset = B00101 (bridge A0 & A1)
```

```
   - TPL5110 (Power Timer Breakout)
   Source: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-tpl5110-power-timer-breakout.pdf7
   Datasheet: http://www.ti.com/lit/ds/symlink/tpl5110.pdf

   - E32-TTL-1W (433MHz Radio/Wireless Transceiver)
   Source: http://www.cdebyte.com/en/product-view-news.aspx?id=108

   - MH-RD Raindrops Module (Sensor de lluvia)
   Source: https://www.luisllamas.es/arduino-lluvia/

   - Wind Direction Sensor JL-FS2
   Source: https://www.banggood.com/es/Wind-Sensor-JL-FS2-Wind-Direction-Sensor-4-20mA-0-5V-Signal-Output-p-971174.html

   - Micro SD Storage Board (MicroSD_Card_Adapter_Bus_SPI)
   Source: http://howtomechatronics.com/tutorials/arduino/arduino-sd-card-data-logging-excel-tutorial

   - Tiny RTC DS1307 (Real time clock I2C)
   Source: http://www.tuelectronica.es/tutoriales/arduino/reloj-rtc-i2c-con-arduino.html
   Be aware, the first time set the time is requiered, for that purpose a program must be sent to the rtc after using it definitely
 ************************************************************************************************************************
*/

/**
 ************************************************************************************************************************
   LIBRERÍAS ************************************************************************************************************
 ************************************************************************************************************************
*/

#include <DHT.h>                 //DHT22
#include <OneWire.h>             //DS18B20
#include <DallasTemperature.h>   //DS18B20
#include <Adafruit_BME280.h>     //BME280
#include <Adafruit_Sensor.h>     //BME280
#include <Wire.h>                //LCD, INA219, INA3221
#include <LiquidCrystal_I2C.h>   //LCD
#include <Adafruit_INA219.h>     //INA219
#include <SDL_Arduino_INA3221.h> //INA3221
#include <SoftwareSerial.h>      //E32-TTL-1W
#include <RTClib.h>              //RTC
#include <SD.h>                  //microSD
#include <SPI.h>                 //microSD
#include "defines.h"             //Archivo de constantes de programa
```

```
/**
 ********************************************************************************************************************************
   PROGRAMA PRINCIPAL - FUNCIÓN SETUP ********************************************************************************************
 ********************************************************************************************************************************
*/

void setup() {

  // VARIABLES DE PROGRAMA ********************************************************************************************************

  SoftwareSerial      radio(RF_TX, RF_RX);  //SOFT_RX, SOFT_TX - Inicializar puerto serie para E32-TTL-1W
  DHT                 dht(DHT_PIN, DHT22);  //Init DHT for 16MHz Arduino. For board with a faster processor, use "DHT22 dht(DHT_PIN, 30);"
  OneWire             oneWire(DS_PIN);      //Setup a oneWire instance to communicate with any OneWire devices
  DallasTemperature   ds18b20(&oneWire);    //Pass our oneWire reference to Dallas Temperature.
  Adafruit_BME280     bme;                  //Declaración para I2C. Para SPI, consultar documentación
  LiquidCrystal_I2C   lcd(0x27, 20, 4);     //Set the LCD address to 0x27 for a 20 chars and 4 line display (I2C can be different)
  SDL_Arduino_INA3221 ina3221;              //I2C address: 0x40
  Adafruit_INA219     ina_cell(CELL1_ADD);  //I2C address: 0x41
  Adafruit_INA219     ina_cell2(CELL2_ADD); //I2C address: 0x44
  Adafruit_INA219     ina_pv(PV_ADD);       //I2C address: 0x45
  RTC_DS1307          rtc;                  //Variable para el RTC

  File myFile;                              //Archivo para guardar datos en la microSD
  const char RX_ADDRESS[] = {0xF0 | (uint8_t)(RX_ID), 0x00};  //Dirección del Receptor: [0x F + N°Receptor(hex) + 00]
  char datos[57];                           //Vector de bytes para transmisión por radio
  uint8_t index = 0;                        //Variable índice para el vector de datos

  // VARIABLES DE MEDIDA **********************************************************************************************************

  float vDHT_HR[ITER_NUM];
  float vDHT_TEX[ITER_NUM];
  float vDS_T1[ITER_NUM];
  float vDS_T2[ITER_NUM];
  float vDS_T3[ITER_NUM];
  float vWIND_HW[ITER_NUM];
  float vWIND_CUP[ITER_NUM];
  float vWIND_DIR[ITER_NUM];
  float vPATM[ITER_NUM];
  float vV_BUS[ITER_NUM];
  float vV_PV[ITER_NUM];
  float vI_BAT[ITER_NUM];
  float vI_CH[ITER_NUM];
  float vI_IN[ITER_NUM];
```

```
float vI_PV[ITER_NUM];
float vIRR_UP[ITER_NUM];
float vIRR_DOWN[ITER_NUM];
float vRAIN[ITER_NUM];
float vIR_T[ITER_NUM];
float vGAS[ITER_NUM];

uint8_t  BAT_LV;
uint16_t DHT_HR;
int16_t  DHT_TEX;
int16_t  DS_T1;
int16_t  DS_T2;
int16_t  DS_T3;
uint16_t WIND_HW;
uint16_t WIND_CUP;
uint8_t  WIND_DIR;
int32_t  PATM;
int16_t  V_BUS;
int16_t  V_PV;
int32_t  I_BAT;
int32_t  I_CH;
int32_t  I_IN;
int32_t  I_PV;
int32_t  IRR_UP;
int32_t  IRR_DOWN;
uint8_t  RAIN;
uint16_t IR_T;
uint16_t GAS;

// INICIALIZACIÓN DE SENSORES Y PERIFÉRICOS ***************************************************************************************************

// SERIAL MONITOR ****************************************
Serial.begin(115200);

// DHT22 (HUMIDITY AND TEMPERATURE SENSOR) *****************
dht.begin();

// DS18B20 (DIGITAL TEMPERATURE SENSORS) *******************
ds18b20.begin();

// BME280 (PRESSURE AND BAROMETER SENSOR) ******************
bme.begin(0x76);
```

```
// INA219, INA3221 (CURRENT AND VOLTAGE SENSOR) ************
ina3221.begin();
ina_cell.begin();
ina_cell2.begin();
ina_pv.begin();

//INA219 by default will use the largest range (32V, 2A). You can call a setCalibration function to change this range:
//ina.setCalibration_32V_1A(), ina.setCalibration_16V_400mA()
ina_cell.setCalibration_16V_400mA();
ina_cell2.setCalibration_16V_400mA();

// LCD_2004A + LCN1602 (LCD 20x4 I2C) **********************
pinMode(LCD_EN, INPUT);
lcd.init();       //initialize the lcd
lcd.backlight();  //open the backlight
lcd.clear();

// TPL5110 (POWER TIMER BREAKOUT) **************************
pinMode(TPL_DONE, OUTPUT);
digitalWrite(TPL_DONE, LOW);

// E32-TTL-1W (433MHz RADIO/WIRELESS TRANSCEIVER) **********
pinMode(RF_M0, OUTPUT);
pinMode(RF_M1, OUTPUT);
pinMode(RF_AUX, INPUT);
radio.begin(9600);

//Modos de funcionamiento - Normal: M0=0 M1=0 / Wake-up: M0=0 M1=1 / Power saving: M0=1 M1=0 / Sleep: M0=1 M1=1
digitalWrite(RF_M0, HIGH);
digitalWrite(RF_M1, HIGH);

// TINY RTC DS1307 (REAL TIME CLOCK I2C) *******************
rtc.begin();

// MICRO-SD CARD ADAPTER BOARD SPI *************************
pinMode(SD_CS, OUTPUT);
pinMode(SD_EN, OUTPUT);
digitalWrite(SD_EN, HIGH);

if (SD.begin(SD_CS)) {
  lcd.setCursor(0, 0);
  lcd.print("SD card is ready    ");
```

```
  if (!SD.exists(FILE_NAME)) {
    myFile = SD.open(FILE_NAME, FILE_WRITE);
    if (!myFile) {
      lcd.setCursor(0, 0);
      lcd.print("Cannot open file.   ");
    }
    myFile.print("Tx_id,");
    myFile.print("Date,");
    myFile.print("BAT_LV%,");
    myFile.print("HR_DHT_%,");
    myFile.print("TEX_DHT_C,");
    myFile.print("DS_T1_C,");
    myFile.print("DS_T2_C,");
    myFile.print("DS_T3_C,");
    myFile.print("V_WIND_HW_m/s,");
    myFile.print("V_WIND_CUP_m/s,");
    myFile.print("DIR_WIND,");
    myFile.print("PATM_Pa,");
    myFile.print("V_BUS_V,");
    myFile.print("I_BAT_mA,");
    myFile.print("I_BOOST_mA,");
    myFile.print("I_IN_mA,");
    myFile.print("I_PV_mA,");
    myFile.print("IRR_UP_Wm2,");
    myFile.print("IRR_DOWN_Wm2,");
    myFile.print("RAIN,");
    myFile.print("T_IR_T,");
    myFile.print("GAS,");

    myFile.println();
    myFile.close();
  }
}
else {
  lcd.setCursor(0, 1);
  lcd.print("SD card init failed ");
}
digitalWrite(SD_EN, LOW);

lcd.setCursor(0, 0);
lcd.print("Midiendo: 0/");
lcd.print(ITER_NUM);
```

```
for (int i = 0; i < ITER_NUM; i++) {
  unsigned long startMillis = millis();

  // 1. DHT22 (Humidity - temperature sensor) ************************************************************************************
  // Reading temperature or humidity takes about 250 ms. Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  // Wait a few seconds between measurements. The DHT22 should not be read at a higher frequency of about once every 2 seconds.

  vDHT_HR[i]  = dht.readHumidity();
  vDHT_TEX[i] = dht.readTemperature();

  if (!DHT_ON || isnan(vDHT_TEX[i]) || isnan(vDHT_HR[i]) || (vDHT_HR[i] == 0 && vDHT_TEX[i] == 0)) { // Check if any reads failed
    vDHT_HR[i]  = ERROR_VAL;
    vDHT_TEX[i] = ERROR_VAL;
  }

  // 2. DS18B20 (Digital temperature sensors) ************************************************************************************

  ds18b20.requestTemperatures(); // Send the command to get temperature readings
  vDS_T1[i] = ds18b20.getTempCByIndex(0);
  vDS_T2[i] = ds18b20.getTempCByIndex(1);
  vDS_T3[i] = ds18b20.getTempCByIndex(2);

  if (!DS_T1_ON || vDS_T1[i] > 80 || vDS_T1[i] < -30) vDS_T1[i] = ERROR_VAL;
  if (!DS_T2_ON || vDS_T2[i] > 80 || vDS_T2[i] < -30) vDS_T2[i] = ERROR_VAL;
  if (!DS_T3_ON || vDS_T3[i] > 80 || vDS_T3[i] < -30) vDS_T3[i] = ERROR_VAL;

  // 4. Anemometer Kit (0-5V,3 Wind Cups And Circuit Module) ************************************************************************
  //Input voltage 0-5 v to pin A0. The level of wind speed is proportional to the output voltage (x6)

  if (WIND_CUP_ON) vWIND_CUP[i]  = ANEMO_ADJ * (analogRead(ANEMO_PIN) * 5.0 / 1024.0);
  else             vWIND_CUP[i]  = ERROR_VAL;

  // 5. BME280 (Presure, barometer and temperature sensor) ***********************************************************************
  //Start with temperature, as that data is needed for accurate compensation.
  //Reading the temperature updates the compensators of the other functions in the background.

  if (PATM_ON) {
    vPATM[i] = bme.readPressure();
    if (!(vPATM[i] > 80000 && vPATM[i] < 120000))
      vPATM[i] = ERROR_VAL;
  }
  else vPATM[i] = ERROR_VAL;
```

```
// 8. INA219/INA3221 (Current Sensor) ********************************************************************************************

if (INA3221_ON) {
  vV_BUS[i] = ina3221.getBusVoltage_V(1);
  vI_BAT[i] = -ina3221.getCurrent_mA(1);
  vI_CH[i]  = -ina3221.getCurrent_mA(2);
  vI_IN[i]  = ina3221.getCurrent_mA(3);
}
else {
  vV_BUS[i] = ERROR_VAL;
  vI_BAT[i] = ERROR_VAL;
  vI_CH[i]  = ERROR_VAL;
  vI_IN[i]  = ERROR_VAL;
}

if (CELL1_ON) vIRR_UP[i] = vAbs(ina_cell.getCurrent_mA() * 1000 / CELL1_ADJ);
else          vIRR_UP[i] = ERROR_VAL;

if (CELL2_ON) vIRR_DOWN[i] = vAbs(ina_cell2.getCurrent_mA() * 1000 / CELL2_ADJ);
else          vIRR_DOWN[i] = ERROR_VAL;

if (INA_PV_ON) {

  vV_PV[i] = ina_pv.getBusVoltage_V();
  vI_PV[i] = ina_pv.getCurrent_mA();
}
else {
  vV_PV[i] = ERROR_VAL;
  vI_PV[i] = ERROR_VAL;
}

// 9. Sensor de lluvia ***********************************************************************************************************

if (RAIN_ON) {
  vRAIN[i] = int((1023 - analogRead(RAIN_AO)) / 10.23);
}
else vRAIN[i] = 255;

Serial.print(i + 1);
Serial.print(", ");
lcd.setCursor(0, 0);
lcd.print("Midiendo: ");
lcd.print(i + 1);
```

```
  lcd.print("/");
  lcd.print(ITER_NUM);

  // Sensor de temperatura infrarroja ************************************************************************************************

  if (IR_ON) {
    vIR_T[i] = 0; //MODIFICAR
  }
  else vIR_T[i] = ERROR_VAL;

  // Sensor de gas *******************************************************************************************************************

  if (GAS_ON) {
    vGAS[i] = 0; //MODIFICAR
  }
  else vGAS[i] = ERROR_VAL;


  if (i != ITER_NUM - 1)
    while ((millis() - startMillis) < ITER_T);
}

// 3. Wind Sensor Rev.C (Hot wire anemometer) ***************************************************************************************

if (WIND_HW_ON) {
  //These equations are all derived from regressions from raw data as such they depend on a lot of experimental factors
  //such as accuracy of temp sensors, and voltage at the actual wind sensor, (wire losses) which were unaccouted for.
  float TMP_Therm_ADunits = analogRead(WINDC_TMP);
  float RV_Wind_Volts     = analogRead(WINDC_RV) *  0.0048828125; //ADC 10 bits
  float zeroWind_ADunits  = -0.0006 * (TMP_Therm_ADunits * TMP_Therm_ADunits) + 1.0727 * TMP_Therm_ADunits + 47.172; //13.0C 553 482.39
  float zeroWind_volts    = (zeroWind_ADunits * 0.0048828125) - (0.15);
  //This from a regression from data in the form of "Vraw = V0 + b * WindSpeed ^ c"
  //V0 is zero wind at a particular temperature, the constants b and c were determined by some Excel wrangling with the solver.
  //0.15 is zeroAdj: negative numbers yield smaller wind speeds and vice versa

  WIND_HW = pow((vAbs(RV_Wind_Volts - zeroWind_volts) / .2300) , 2.7265) * 1.60934 / 3.6;
  //El valor absoluto evita medidas faltantes cuando la base es negativa
}
else WIND_HW = ERROR_VAL;

// 11. Wind Direction Sensor JL-FS2 *************************************************************************************************

if (WIND_DIR_ON) WIND_DIR = analogRead(DIR_PIN) * 16 / 1024 + 1;
```

```
else             WIND_DIR = 0;

Serial.println("Calculando...");
lcd.setCursor(0, 1);
lcd.print("Calculando...");

// Cálculo de las medias aritméticas de las medidas ************************************************************************************

DHT_HR   = (uint16_t) CalcularMedia(vDHT_HR);
DHT_TEX  = (int16_t)  CalcularMedia(vDHT_TEX) ;
DS_T1    = (int16_t)  CalcularMedia(vDS_T1);
DS_T2    = (int16_t)  CalcularMedia(vDS_T2);
DS_T3    = (int16_t)  CalcularMedia(vDS_T3);
WIND_CUP = (uint16_t) CalcularMedia(vWIND_CUP);
PATM     = (int32_t)  CalcularMedia(vPATM) / 100;
if (PATM == ERROR_VAL / 100) PATM = ERROR_VAL;
V_BUS    = (int16_t)  CalcularMedia(vV_BUS);
V_PV     = (int16_t)  CalcularMedia(vV_PV);
I_BAT    = (int32_t)  CalcularMedia(vI_BAT);
I_CH     = (int32_t)  CalcularMedia(vI_CH);
I_IN     = (int32_t)  CalcularMedia(vI_IN);
I_PV     = (int32_t)  CalcularMedia(vI_PV);
IRR_UP   = (int32_t)  CalcularMedia(vIRR_UP);
IRR_DOWN = (int32_t)  CalcularMedia(vIRR_DOWN);
RAIN     = (uint8_t)  (CalcularMedia(vRAIN) / 100);
IR_T     = (uint16_t) CalcularMedia(vIR_T);
GAS      = (uint16_t) CalcularMedia(vGAS);

// Cálculo del porcentaje de batería ***************************************************************************************************

if (V_BUS != ERROR_VAL) {
  if (V_BUS >= 310)
    BAT_LV = V_BUS - 300;
  else if (V_BUS * 0.25 - 67.5 > 0)
    BAT_LV = V_BUS * 0.25 - 67.5;
  else
    BAT_LV = 0;

  if (BAT_LV > 100) BAT_LV = 100;
}
else BAT_LV = 0;

// 8. Serial Monitor *******************************************************************************************************************
```

```
Serial.print("TX_ID    = "); Serial.print(TX_ID);    Serial.println("");
Serial.print("BAT_LV   = "); Serial.print(BAT_LV);   Serial.println(" %");
Serial.print("DHT_HR   = "); Serial.print(DHT_HR);   Serial.println(" %");
Serial.print("DHT_TEX  = "); Serial.print(DHT_TEX);  Serial.println(" *C");
Serial.print("DS_T1    = "); Serial.print(DS_T1);    Serial.println(" *C");
Serial.print("DS_T2    = "); Serial.print(DS_T2);    Serial.println(" *C");
Serial.print("DS_T3    = "); Serial.print(DS_T3);    Serial.println(" *C");
Serial.print("WIND_HW  = "); Serial.print(WIND_HW);  Serial.println(" m/s");
Serial.print("WIND_CUP = "); Serial.print(WIND_CUP); Serial.println(" m/s");
Serial.print("WIND_DIR = "); Serial.print(WIND_DIR); Serial.println("");
Serial.print("PATM     = "); Serial.print(PATM);     Serial.println(" Pa");
Serial.print("V_BUS    = "); Serial.print(V_BUS);    Serial.println(" V");
Serial.print("V_PV     = "); Serial.print(V_PV);     Serial.println(" V");
Serial.print("I_BAT    = "); Serial.print(I_BAT);    Serial.println(" mA");
Serial.print("I_CH     = "); Serial.print(I_CH);     Serial.println(" mA");
Serial.print("I_IN     = "); Serial.print(I_IN);     Serial.println(" mA");
Serial.print("I_PV     = "); Serial.print(I_PV);     Serial.println(" mA");
Serial.print("IRR_UP   = "); Serial.print(IRR_UP);   Serial.println(" W/m2");
Serial.print("IRR_DOWN = "); Serial.print(IRR_DOWN); Serial.println(" W/m2");
Serial.print("RAIN     = "); Serial.print(RAIN);     Serial.println("");
Serial.print("IR_T     = "); Serial.print(IR_T);     Serial.println(" *C");
Serial.print("GAS      = "); Serial.print(GAS);      Serial.println(" ppm");

// 7. LCD_2004A + LCN1602(LCD de 20x4 I2C) ***************************************************************************************

if (digitalRead(LCD_EN) == HIGH) {
  lcd.clear();
  lcd.setCursor(0, 0); lcd.print("BAT_LV  ="); lcd.print(BAT_LV);   lcd.print(" %");
  lcd.setCursor(0, 1); lcd.print("DHT_HR  ="); lcd.print(DHT_HR);   lcd.print(" %");
  lcd.setCursor(0, 2); lcd.print("DHT_TEX ="); lcd.print(DHT_TEX);  lcd.print(" C");
  lcd.setCursor(0, 3); lcd.print("DS_T1   ="); lcd.print(DS_T1);    lcd.print(" C");
  delay(5000);

  lcd.clear();
  lcd.setCursor(0, 0); lcd.print("DS_T2   ="); lcd.print(DS_T2);    lcd.print(" C");
  lcd.setCursor(0, 1); lcd.print("DS_T3   ="); lcd.print(DS_T3);    lcd.print(" C");
  lcd.setCursor(0, 2); lcd.print("PATM    ="); lcd.print(PATM);     lcd.print(" Pa");
  lcd.setCursor(0, 3); lcd.print("WIND_HW ="); lcd.print(WIND_HW);  lcd.print(" ms");
  delay(5000);

  lcd.clear();
  lcd.setCursor(0, 0); lcd.print("WIND_CUP="); lcd.print(WIND_CUP); lcd.print(" ms");
```

```
  lcd.setCursor(0, 1); lcd.print("V_BUS   ="); lcd.print(V_BUS);    lcd.print(" V");
  lcd.setCursor(0, 2); lcd.print("I_BAT   ="); lcd.print(I_BAT);    lcd.print(" mA");
  lcd.setCursor(0, 3); lcd.print("I_CH    ="); lcd.print(I_CH);     lcd.print(" mA");
  delay(5000);

  lcd.clear();
  lcd.setCursor(0, 0); lcd.print("I_IN    ="); lcd.print(I_IN);     lcd.print(" mA");
  lcd.setCursor(0, 1); lcd.print("IRR_UP  ="); lcd.print(IRR_UP);   lcd.print(" Wm2");
  lcd.setCursor(0, 2); lcd.print("IRR_DOWN="); lcd.print(IRR_DOWN); lcd.print(" Wm2");
  lcd.setCursor(0, 3); lcd.print("RAIN    ="); lcd.print(RAIN);     lcd.print("");
  delay(5000);
  lcd.clear();
}

// 11. E32-TTL-1W (433MHz Radio/Wireless Transceiver) ************************************************************************************

digitalWrite(RF_M0, LOW);
digitalWrite(RF_M1, LOW);
delay(50);

datos[index++] = RX_ADDRESS[0];
datos[index++] = RX_ADDRESS[1];
datos[index++] = CHANNEL;

datos[index++] = TX_ID;

datos[index++] = BAT_LV;

datos[index++] = DHT_HR >> 8;
datos[index++] = DHT_HR & 0x00FF;
datos[index++] = DHT_TEX >> 8;
datos[index++] = DHT_TEX & 0x00FF;

datos[index++] = DS_T1 >> 8;
datos[index++] = DS_T1 & 0x00FF;
datos[index++] = DS_T2 >> 8;
datos[index++] = DS_T2 & 0x00FF;
datos[index++] = DS_T3 >> 8;
datos[index++] = DS_T3 & 0x00FF;

datos[index++] = WIND_HW >> 8;
datos[index++] = WIND_HW & 0x00FF;
datos[index++] = WIND_CUP >> 8;
```

```
datos[index++] = WIND_CUP & 0x00FF;
datos[index++] = WIND_DIR;

datos[index++] = PATM >> 24;
datos[index++] = PATM >> 16 & 0x000000FF;
datos[index++] = PATM >> 8  & 0x000000FF;
datos[index++] = PATM & 0x000000FF;

datos[index++] = V_BUS >> 8;
datos[index++] = V_BUS & 0x00FF;

datos[index++] = V_PV >> 8;
datos[index++] = V_PV & 0x00FF;

datos[index++] = I_BAT >> 24;
datos[index++] = I_BAT >> 16 & 0x000000FF;
datos[index++] = I_BAT >> 8  & 0x000000FF;
datos[index++] = I_BAT & 0x000000FF;

datos[index++] = I_CH >> 24;
datos[index++] = I_CH >> 16 & 0x000000FF;
datos[index++] = I_CH >> 8  & 0x000000FF;
datos[index++] = I_CH & 0x000000FF;

datos[index++] = I_IN >> 24;
datos[index++] = I_IN >> 16 & 0x000000FF;
datos[index++] = I_IN >> 8  & 0x000000FF;
datos[index++] = I_IN & 0x000000FF;

datos[index++] = I_PV >> 24;
datos[index++] = I_PV >> 16 & 0x000000FF;
datos[index++] = I_PV >> 8  & 0x000000FF;
datos[index++] = I_PV & 0x000000FF;

datos[index++] = IRR_UP >> 24;
datos[index++] = IRR_UP >> 16 & 0x000000FF;
datos[index++] = IRR_UP >> 8  & 0x000000FF;
datos[index++] = IRR_UP & 0x000000FF;

datos[index++] = IRR_DOWN >> 24;
datos[index++] = IRR_DOWN >> 16 & 0x000000FF;
datos[index++] = IRR_DOWN >> 8  & 0x000000FF;
datos[index++] = IRR_DOWN & 0x000000FF;
```

```
  datos[index++] = RAIN;

  datos[index++] = IR_T >> 8;
  datos[index++] = IR_T & 0x00FF;

  datos[index++] = GAS >> 8;
  datos[index++] = GAS & 0x00FF;

  radio.write(datos, sizeof(datos));

  //lcd.setCursor(0, 0);
  //lcd.print("Writing completed");
  delay(3000);
}

/**
 ********************************************************************************************************************************************
   PROGRAMA PRINCIPAL - FUNCIÓN LOOP ********************************************************************************************************
 ********************************************************************************************************************************************
*/

void loop() {
  digitalWrite(TPL_DONE, HIGH);
}

/**
 ********************************************************************************************************************************************
   FUNCIÓN CALCULARMEDIA() ******************************************************************************************************************
 ********************************************************************************************************************************************
*/

float CalcularMedia(float v[]) {
  float suma = 0;
  int medidas = 0;
  for (int i = 0; i < ITER_NUM; i++) {
    if (v[i] != ERROR_VAL) {
      medidas++;
      suma += v[i];
    }
  }
  if (medidas == 0)
    return ERROR_VAL;
```

```
  else
    return 100 * suma / medidas;
}

/**
 ************************************************************************************************************************
   FUNCIÓN VALOR ABSOLUTO ************************************************************************************************
 ************************************************************************************************************************
*/

float vAbs(float v) {
  if (v < 0) return -v;
  else       return v;
}

/* FIN DE PROGRAMA ********************************************************************************************************/
```

```
/**
 ****************************************************************************************************************************
   PROYECTO GROWGREEN VALENCIA - TRANSMITTER v4.2
   (ARDUINO MEGA / ARDUINO UNO / E32-TTL-1W)

   @file    defines.h
   @date    21/05/2018
   @author  Carlos Vargas Salgado
            Alejandro Sánchez Aduna
 ****************************************************************************************************************************
*/

/**
 ****************************************************************************************************************************
  DEFINICIONES DE VARIABLES DE PROGRAMA ************************************************************************************
 ****************************************************************************************************************************
*/

// VARIABLES DE TRANSMISIÓN DE RADIO ***************************************************************************************
#define TX_ID       12       //Variable ID: Identificador del Transmisor [1, 2, 3...] (decimal)
#define RX_ID       2       //Identificador del Receptor [1, 2, 3...] (decimal) Default: 2
#define CHANNEL     0x17    //Canal de comunicación de radio. [Freq = (410 + CHANNEL(dec)*10^6) MHz] Default: 0x17 (433 MHz)

// ACTIVACIÓN Y DESACTIVACIÓN DE SENSORES (true/false) **********************************************************************
#define DHT_ON      true    //Sensor de temperatura y humedad DHT22
#define DS_T1_ON    true    //Sensor de temperatura digital DS18B20 (T1)
#define DS_T2_ON    true    //Sensor de temperatura digital DS18B20 (T2)
#define DS_T3_ON    true    //Sensor de temperatura digital DS18B20 (T3)
#define WIND_HW_ON  false   //Sensor de viento de hilo caliente REV C.
#define WIND_CUP_ON true    //Sensor de viento de 3 copas
#define WIND_DIR_ON false   //Sensor de dirección de viento (Default: false)
#define PATM_ON     true    //Sensor de presión BME280
#define INA3221_ON  true    //Sensor de tensión y corriente de 3 canales INA3221
#define CELL1_ON    true    //Sensor de radiación superior (INA219)
#define CELL2_ON    true    //Sensor de radiación inferior (INA219)
#define INA_PV_ON   true    //Sensor de tensión y corriente del panel solar (INA219)
#define RAIN_ON     true    //Sensor de lluvia MH-RD
#define IR_ON       false   //Sensor de temperatura infrarrojos (Default: false)
#define GAS_ON      false   //Sensor de gas (Default: false)

#define SD_ON       false   //Módulo microSD + RTC (Default: false)
#define RF24_ON     false   //Módulo de comunicación radio nRF24L01 (Default: false)
```

```
// DECLARACIÓN DE PINES DIGITALES (NO MODIFICAR) ********************************************************************************
#define DHT_PIN     2
#define DS_PIN      3
#define LCD_EN      4
#define NXT_EN      5
#define TPL_DONE    7
#define RF_M0       8
#define RF_M1       9
#define RF_RX       10
#define RF_TX       11
#define RF_AUX      12
#define SD_EN       47
#define RF24_CE     48
#define RF24_CS     49
#define SD_CS       53

// DECLARACIONES DE PINES ANALÓGICOS (NO MODIFICAR) *****************************************************************************
#define WINDC_TMP   0
#define WINDC_RV    1
#define RAIN_AO     2
#define ANEMO_PIN   3
#define DIR_PIN     6

// OTRAS VARIABLES ***************************************************************************************************************
#define ITER_NUM    10      //Número de medidas
#define ITER_T      1000    //Periodo de cada iteración. Tiempo entre medidas (ms)
#define ANEMO_ADJ   6.48    //Constante del anemómetro - Metálico: 6 (0-5V, 0-30m/s), Plástico: 6.48 (0-5V, 0-32.4m/s)
#define CELL1_ADJ   307.0   //Constante de calibración del sensor de radiación superior (Etiqueta del reverso. Default: 302.0)
#define CELL2_ADJ   302.0   //Constante de calibración del sensor de radiación inferior (Etiqueta del reverso. Default: 302.0)
#define CELL1_ADD   0x41    //Dirección del bus I2C del INA_CELL1 (Default 0x41)
#define CELL2_ADD   0x44    //Dirección del bus I2C del INA_CELL2 (Default 0x44)
#define PV_ADD      0x45    //Dirección del bus I2C del INA_PV    (Default 0x45)
#define FILE_NAME   "hsmupv.csv" //Nombre de archivo para microSD
#define ERROR_VAL   32767   //Valor de error (Default: 32767)

/* FIN DE PROGRAMA ***************************************************************************************************************/
```

ANEXO V: Código de Arduino para Receptor v4.2

```
/**
 ****************************************************************************************************************************************
   PROYECTO GROWGREEN VALENCIA - RECEIVER v4.2
   (ARDUINO UNO / E32-TTL-1W)

   @file    Receiver_Rasp_4.2_E32.ino
   @date    21/05/2018
   @author  Carlos Vargas Salgado
            Alejandro Sánchez Aduna
            Jose Ignacio Marqués Ortega
 ****************************************************************************************************************************************
*/

//###########################################################################################################################################
// ÍNDICE DE PROGRAMA

// 1. E32-TTL-1W (433MHz Radio/Wireless Transceiver) ********************************************************************************
//    Source: http://www.cdebyte.com/en/product-view-news.aspx?id=108

//###########################################################################################################################################
// PROGRAMA

#include <SoftwareSerial.h>

// VARIABLES DE TRANSMISIÓN DE RADIO
#define RF_M0       8
#define RF_M1       9
#define RF_RX       10
#define RF_TX       11
#define RF_AUX      12

#define ERROR_VAL   "32767"

SoftwareSerial radio(RF_TX, RF_RX);  //SOFT_RX, SOFT_TX

//###########################################################################################################################################
// FUNCIÓN SETUP

void setup() {
```

```
  // 9. Serial Monitor ***************************************************************************************************

  Serial.begin(9600);  //Connetion with serial port

  // E32-TTL-1W (433MHz Radio/Wireless Transceiver) ********************************************************************

  pinMode(RF_M0, OUTPUT);
  pinMode(RF_M1, OUTPUT);
  pinMode(RF_AUX, INPUT);
  radio.begin(9600);

  // Modos de funcionamiento - Normal: M0=0 M1=0 / Wake-up: M0=0 M1=1 / Power saving: M0=1 M1=0 / Sleep: M0=1 M1=1
  digitalWrite(RF_M0, LOW);
  digitalWrite(RF_M1, LOW);
}

//#####################################################################################################################################
// FUNCIÓN LOOP

void loop() {

  String   data;

  uint8_t  TX_ID    = 0;
  uint8_t  BAT_LV   = 0;
  uint16_t DHT_HR   = 0;
  int16_t  DHT_TEX  = 0;
  int16_t  DS_T1    = 0;
  int16_t  DS_T2    = 0;
  int16_t  DS_T3    = 0;
  uint16_t WIND_HW  = 0;
  uint16_t WIND_CUP = 0;
  uint8_t  WIND_DIR = 0;
  int32_t  PATM     = 0;
  int16_t  V_BUS    = 0;
  int16_t  V_PV     = 0;
  int32_t  I_BAT    = 0;
  int32_t  I_CH     = 0;
  int32_t  I_IN     = 0;
  int32_t  I_PV     = 0;
  int32_t  IRR_UP   = 0;
  int32_t  IRR_DOWN = 0;
  uint8_t  RAIN     = 0;
```

```
uint16_t IR_T     = 0;
uint16_t GAS      = 0;

while (true) {
  if (radio.available()) {
    TX_ID    = radio.read();

    BAT_LV   = radio.read();

    DHT_HR   = radio.read();
    DHT_HR   = (DHT_HR << 8) | radio.read();
    DHT_TEX  = radio.read();
    DHT_TEX  = (DHT_TEX << 8) | radio.read();

    DS_T1    = radio.read();
    DS_T1    = (DS_T1 << 8) | radio.read();
    DS_T2    = radio.read();
    DS_T2    = (DS_T2 << 8) | radio.read();
    DS_T3    = radio.read();
    DS_T3    = (DS_T3 << 8) | radio.read();

    WIND_HW  = radio.read();
    WIND_HW  = (WIND_HW << 8) | radio.read();
    WIND_CUP = radio.read();
    WIND_CUP = (WIND_CUP << 8) | radio.read();
    WIND_DIR = radio.read();

    PATM     = radio.read();
    PATM     = (PATM << 8) | radio.read();
    PATM     = (PATM << 8) | radio.read();
    PATM     = (PATM << 8) | radio.read();

    V_BUS    = radio.read();
    V_BUS    = (V_BUS << 8) | radio.read();

    V_PV     = radio.read();
    V_PV     = (V_PV << 8) | radio.read();

    I_BAT    = radio.read();
    I_BAT    = (I_BAT << 8) | radio.read();
    I_BAT    = (I_BAT << 8) | radio.read();
    I_BAT    = (I_BAT << 8) | radio.read();
```

```
I_CH     = radio.read();
I_CH     = (I_CH << 8) | radio.read();
I_CH     = (I_CH << 8) | radio.read();
I_CH     = (I_CH << 8) | radio.read();

I_IN     = radio.read();
I_IN     = (I_IN << 8) | radio.read();
I_IN     = (I_IN << 8) | radio.read();
I_IN     = (I_IN << 8) | radio.read();

I_PV     = radio.read();
I_PV     = (I_PV << 8) | radio.read();
I_PV     = (I_PV << 8) | radio.read();
I_PV     = (I_PV << 8) | radio.read();

IRR_UP   = radio.read();
IRR_UP   = (IRR_UP << 8) | radio.read();
IRR_UP   = (IRR_UP << 8) | radio.read();
IRR_UP   = (IRR_UP << 8) | radio.read();

IRR_DOWN = radio.read();
IRR_DOWN = (IRR_DOWN << 8) | radio.read();
IRR_DOWN = (IRR_DOWN << 8) | radio.read();
IRR_DOWN = (IRR_DOWN << 8) | radio.read();

RAIN     = radio.read();

IR_T     = radio.read();
IR_T     = (IR_T << 8) | radio.read();

GAS      = radio.read();
GAS      = (GAS << 8) | radio.read();

// 8. Montamos el string de datos para comunicacion serie con la Raspberry Pi ************************************************

String DIR;
switch (WIND_DIR) {
  case 1:  DIR = "S";   break;
  case 2:  DIR = "SSO"; break;
  case 3:  DIR = "SO";  break;
  case 4:  DIR = "OSO"; break;
  case 5:  DIR = "O";   break;
  case 6:  DIR = "ONO"; break;
```

```
  case 7:  DIR = "NO";  break;
  case 8:  DIR = "NNO"; break;
  case 9:  DIR = "N";   break;
  case 10: DIR = "NNE"; break;
  case 11: DIR = "NE";  break;
  case 12: DIR = "ENE"; break;
  case 13: DIR = "E";   break;
  case 14: DIR = "ESE"; break;
  case 15: DIR = "SE";  break;
  case 16: DIR = "SSE"; break;
  default: DIR = "0";   break;
}

data =  "ID="    + String(TX_ID);

data += "&BAT="  + String(BAT_LV);

if (DHT_HR == 32767) data += "&DHTH=" + String(ERROR_VAL);
else data += "&DHTH=" + String((float)DHT_HR   / 100, 2);

if (DHT_TEX == 32767) data += "&DHTT=" + String(ERROR_VAL);
else data += "&DHTT=" + String((float)DHT_TEX  / 100, 2);

if (DS_T1 == 32767) data += "&T1=" + String(ERROR_VAL);
else data += "&T1="   + String((float)DS_T1    / 100, 2);

if (DS_T2 == 32767) data += "&T2=" + String(ERROR_VAL);
else data += "&T2="   + String((float)DS_T2    / 100, 2);

if (DS_T3 == 32767) data += "&T3=" + String(ERROR_VAL);
else data += "&T3="   + String((float)DS_T3    / 100, 2);

if (WIND_HW == 32767) data += "&WHW=" + String(ERROR_VAL);
else data += "&WHW="  + String((float)WIND_HW  / 100, 2);

if (WIND_CUP == 32767) data += "&WCUP=" + String(ERROR_VAL);
else data += "&WCUP=" + String((float)WIND_CUP / 100, 2);

data += "&WDIR=" + DIR;

if (PATM == 327) data += "&PATM=" + String(ERROR_VAL);
else data += "&PATM=" + String(PATM);
```

```
      if (V_BUS == 32767) data += "&VBUS=" + String(ERROR_VAL);
      else data += "&VBUS=" + String((float)V_BUS    / 100, 2);

      if (V_PV == 32767) data += "&VPV=" + String(ERROR_VAL);
      else data += "&VPV="  + String((float)V_PV     / 100, 2);

      if (I_BAT == 32767) data += "&IBAT=" + String(ERROR_VAL);
      else data += "&IBAT=" + String((float)I_BAT    / 100, 2);

      if (I_CH == 32767) data += "&ICH=" + String(ERROR_VAL);
      else data += "&ICH="   + String((float)I_CH    / 100, 2);

      if (I_IN == 32767) data += "&IIN=" + String(ERROR_VAL);
      else data += "&IIN="  + String((float)I_IN     / 100, 2);

      if (I_PV == 32767) data += "&IPV=" + String(ERROR_VAL);
      else data += "&IPV="  + String((float)I_PV     / 100, 2);

      if (IRR_UP == 32767) data += "&IRRU=" + String(ERROR_VAL);
      else data += "&IRRU=" + String((float)IRR_UP   / 100, 2);

      if (IRR_DOWN == 32767) data += "&IRRD=" + String(ERROR_VAL);
      else data += "&IRRD=" + String((float)IRR_DOWN / 100, 2);

      if (RAIN == 327) data += "&RAIN=" + String(ERROR_VAL);
      else data += "&RAIN=" + String(RAIN);

      if (IR_T == 32767) data += "&IRT=" + String(ERROR_VAL);
      else data += "&IRT="  + String((float)IR_T     / 100, 2);

      if (GAS == 32767) data += "&GAS=" + String(ERROR_VAL);
      else data += "&GAS="  + String(GAS);

      Serial.println(data); //Enviamos por Serial a la RaspBerry.

      data = "";  //Limpiamos el string
    }
    delay(1000);
  }
}

// FIN DE PROGRAMA
//#########################################################################################################################
```

ANEXO VI: Código Python para tratamiento de datos en Raspberry Pi

```
# NOMBRE DE ARCHIVO: "CONEXIONRASP.PY" ##########################################################################################################
#! /usr/bin/pyhton3
# Librerias a utilizar
import os, sys                                                  # librerias del sistema Python 3.4
import serial                                                   # Lectura serie Rspberry Pi
import time                                                     # Tiempo de ejecución para hacer los delays
import requests                                                 # Protocolo HTTP
import datetime                                                 # Fecha y hora registro
import csv                                                      # Gestor de CSV

puertoserie = '/dev/ttyACM0'                                    # Puerto USB Raspberry Pi
peticion = 'https://growgreenvlc.webs.upv.es/conexionrasp.php?' # Cabecera de la Peticion

# Funciones
def datalogg(x,z):
    '''
    Logea la salida a Plesk
    '''
    t = datetime.datetime.now().strftime('%d-%m-%Y %H:%M:%S')   # Obtiene la fecha local y le damos formato
    fecha = t[0:10]                                             # Seleccionamos la fecha
    hora = t[11:]                                               # Seleccionamos la hora
    mylisten = [[z, fecha, hora, x]]                            # Escritura de linea
    file = open('/home/pi/Documents/log.csv','a')               # Abrimos el fichero

    with file:
        writer = csv.writer(file)                               # Activamos la escritura
        writer.writerows(mylisten)                              # Escribimos la linea

    return (0)

def data2plesk(data,pid):
    '''
    Esta función hace un una petición GET para escribir en la base de datos MySql.
    Retorna la respuesta del servidor sobre la inscripcion de la base de datos
    '''
    dataurl = peticion + data + '&R=2'   # <== Cambiar segun el ID de la Raspberry (Recptor 1 o 2) # Se monta el string de la petición
    datalogg(dataurl, pid)                                      # Logeamos la URL que se envia a Plesk
    r = requests.get(dataurl)                                   # Petición GET (pleskserver)
    return r
```

```
# PROGRAMA PRNICIPAL ##########################################################################################################################################

if __name__ == '__main__':

    pid = str(os.getpid())
    time.sleep(35)                                                  # Esperamos a que la Raspberry se conecte a internet
    arduino = serial.Serial(puertoserie, baudrate=9600, timeout=1.0) # Iniciamos la comunicación serie con el Arduino.

    # Reseteamos el arduino

    arduino.setDTR(False)
    time.sleep(1)
    arduino.flushInput()
    arduino.setDTR(True)

    with arduino:                                                   # Si hay conexion al arduino

        while 1:                                                    # Bucle infinito

            try:
                line = arduino.readline()                           # Espera a que se reciba una nueva línea de bytes.
                data = line.decode('utf-8').rstrip()                # Transformamos bytes en strings decodificando a UTF-8, ASCII también vale
                                                                    # Quitamos el espacio final (da problemas si no lo haces)

                ## Completar aqui con un parse para comprobar el tipo de datos que tenemos y mandar la URL limpia  con Null o algo por determina
                ## def parse y analisys ??

                if not data:
                    # HACK: Descartamos líneas vacías porque fromstring produce resultados erróneos.
                    # Ver https://github.com/numpy/numpy/issues/1714
                    continue

                data2plesk(data, pid)                               # Respuesta del servidor. La utilizaremos para gestion de errores
                time.sleep(0.5)

            # Errores
            except ValueError:
                warnings.warn("Line {} didn't parse, skipping".format(data))
            except KeyboardInterrupt:
                print("Exiting")
                break
```

ANEXO VII: Código PHP de escritura en la Base de Datos en Plesk

```
<?php
// ********************************************************************************************************************************
// @file    conexionbeni.php
// @author  Alejandro Sánchez Aduna
//          Jose Ignacio Marqués Ortega
// ********************************************************************************************************************************

// Recoge el ID del emisor de datos
$tx_id = $_GET['ID'];

// Parametros de conexion BBDD
$db = mysqli_connect("localhost:3306", "user", "password", "database") or die("No conectado a la BD");

  if ($db){
    echo 'Conexion establecida';
    if ($tx_id < 99){
        // Tomando los datos de la consulta
        $eq = mysqli_real_escape_string($db, $tx_id);                  // Variable de id Arduino Numero de equipo "Eq_1"
        $lvl = mysqli_real_escape_string($db, $_GET['BAT']);           // Nivel de la bateria
        $dht_h = mysqli_real_escape_string($db, $_GET['DHTH']);        // Variable Humdead %
        $dht_t = mysqli_real_escape_string($db, $_GET['DHTT']);        // Variable Temperatura °C
        $t1 = mysqli_real_escape_string($db, $_GET['T1']);             // Variable Sensor T1 °C
        $t2 = mysqli_real_escape_string($db, $_GET['T2']);             // Variable Sensor T2 °C
        $t3 = mysqli_real_escape_string($db, $_GET['T3']);             // Variable Sensor T3 °C
        $whw = mysqli_real_escape_string($db, $_GET['WHW']);           // Variable Sensor Viento ms
        $wcp = mysqli_real_escape_string($db, $_GET['WCUP']);          // Variable Anemometro copas ms
        $dir = mysqli_real_escape_string($db, $_GET['WDIR']);          // Variable direccion del viento
        $bme = mysqli_real_escape_string($db, $_GET['PATM']);          // Variable Sensor BME (presion atmosférica)
        $vbus = mysqli_real_escape_string($db, $_GET['VBUS']);         // Variable Sensor tensión batería
        $pv = mysqli_real_escape_string($db, $_GET['VPV']);            // Variable Sensor tensión placa solar
        $ibat = mysqli_real_escape_string($db, $_GET['IBAT']);         // Variable Sensor Corriente Bateria
        $ich = mysqli_real_escape_string($db, $_GET['ICH']);           // Variable Sensor Corriente del driver
        $iin = mysqli_real_escape_string($db, $_GET['IIN']);           // Variable Sensor Corriente Arduino
        $ipv = mysqli_real_escape_string($db, $_GET['IPV']);           // Variable Sensor Corriente del panel
        $cellu = mysqli_real_escape_string($db, $_GET['IRRU']);        // Variable Radiacion up
        $celld = mysqli_real_escape_string($db, $_GET['IRRD']);        // Variable Radiacion down
        $rain = mysqli_real_escape_string($db, $_GET['RAIN']);         // Variable Sensor lluvia
        $inf = mysqli_real_escape_string($db, $_GET['IRT']);           // Variabls Sensor infrarrojo
        $gas = mysqli_real_escape_string($db, $_GET['GAS']);           // Variable GAS
        $r = mysqli_real_escape_string($db, $_GET['R']);               // Variable Receptor
```

```
        // Registro Medidas erroneas
        if($gas != '32767'){
            echo ' Medida erronea';

            $sql = "INSERT INTO ERROR(I, DATE, ID, RASP) VALUES(NULL, CURRENT_TIMESTAMP, '".$eq."','".$r."')" ;

            if (!mysqli_query($db,$sql)){
                echo(" Error description: " . mysqli_error($db));
            }

            else {
                echo " Escritura correcta";
            }

            // Cerrar base de datos
            mysqli_close($db);
        }

    else{
        switch ($tx_id){

            case '1':
                echo " Transmisor 1 :";
                $sql = "INSERT INTO HSM1(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TX_DS_C, TBOX_DS_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t3."','".$t2."','".$t1."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '2':
                echo " Transmisor 2 :";
                $sql = "INSERT INTO HSM2(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TX_DS_C, TBOX_DS_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t1."','".$t2."','".$t3."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '3':
                echo " Transmisor 3 :";
                $sql = "INSERT INTO HSM3(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TX_DS_C, TBOX_DS_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t3."','".$t2."','".$t1."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
```

```
                break;

            case '4':
                echo " Transmisor 4 :";
                $sql = "INSERT INTO HSM4(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TX_DS_C, TBOX_DS_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t2."','".$t3."','".$t1."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '5':
                echo " Transmisor 5 :";
                $sql = "INSERT INTO HSM5(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TX_DS_C, TBOX_DS_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t3."','".$t2."','".$t1."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '6':
                echo " Transmisor 6 :";
                $sql = "INSERT INTO HSM6(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TX_DS_C, TBOX_DS_C, V_WIND_HW,
      V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
      VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t2."','".$t3."','".$t1."','".$whw."','".$wcp."',
      '".$dir."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '7':
                echo " Transmisor 7 :";
                $sql = "INSERT INTO HSM7(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TX_DS_C, TBOX_DS_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t2."','".$t3."','".$t1."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '8':
                echo " Transmisor 8 :";
                $sql = "INSERT INTO HSM8(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TWALL1_C, TWALL2_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t1."','".$t2."','".$t3."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '9':
```

```
                echo " Transmisor 9 :";
                $sql = "INSERT INTO HSM9(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TWALL1_C, TWALL2_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t1."','".$t2."','".$t3."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '10':
                echo " Transmisor 10 :";
                $sql = "INSERT INTO HSM10(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TWALL1_C, TWALL2_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t1."','".$t2."','".$t3."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '11':
                echo " Transmisor 11 :";
                $sql = "INSERT INTO HSM11(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TWALL1_C, TWALL2_C, V_WIND_HW,
        V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
        VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t1."','".$t2."','".$t3."','".$whw."','".$wcp."',
        '".$dir."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '12':
                echo " Transmisor 12 :";
                $sql = "INSERT INTO HSM12(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, TX_DS_C, TWALL1_C, TWALL2_C, V_WIND_HW,
        V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
        VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t2."','".$t1."','".$t3."','".$whw."','".$wcp."',
        '".$dir."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '13':
                echo " Transmisor 13 :";
                $sql = "INSERT INTO HSM13(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TX_DS_C, TBOX_DS_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t1."','".$t2."','".$t3."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '14':
                echo " Transmisor 14 :";
```

```
                $sql = "INSERT INTO HSM14(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TX_DS_C, TBOX_DS_C, V_WIND_HW,
        V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
        VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t1."','".$t2."','".$t3."','".$whw."','".$wcp."',
        '".$dir."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            case '15':
                echo " Transmisor 15 :";
                $sql = "INSERT INTO HSM15(Input, Receptor, Tx_id, Date, BAT_LV, HR_DHT, TEX_DHT_C, WBGT_DS_C, TX_DS_C, TBOX_DS_C, V_WIND_HW,
V_WIND_CUP, DIR_WIND, PATM_Pa, V_BUS_V, V_PV_V, I_BAT_mA, I_CH_mA, I_IN_mA, I_PV_mA, IRR_UP_Wm2, IRR_DOWN_Wm2, RAIN)
VALUES(NULL,'".$r."','".$eq."',CURRENT_TIMESTAMP,'".$lvl."','".$dht_h."','".$dht_t."','".$t1."','".$t2."','".$t3."','".$whw."','".$wcp."','".$dir
."','".$bme."','".$vbus."','".$pv."','".$ibat."','".$ich."','".$iin."','".$ipv."','".$cellu."','".$celld."','".$rain."')";
                break;

            default:
                echo " No se quien eres....";
                break;
        }
    }

        // Cosulta de Insercion
        if (!mysqli_query($db,$sql)){ echo("Error description: " . mysqli_error($db)); }
        else { echo "Escritura correcta"; }

        // Cerrar base de datos
        mysqli_close($db);
    }

    else{ // DISPOSITIVOS DE REDES
        $eq = mysqli_real_escape_string($db, $tx_id);
        $van = mysqli_real_escape_string($db, $_GET['VAN']);
        $vbn = mysqli_real_escape_string($db, $_GET['VBN']);
        $vcn = mysqli_real_escape_string($db, $_GET['VCN']);
        $vab = mysqli_real_escape_string($db, $_GET['VAB']);
        $vbc = mysqli_real_escape_string($db, $_GET['VBC']);
        $vca = mysqli_real_escape_string($db, $_GET['VCA']);
        $ia = mysqli_real_escape_string($db, $_GET['IA']);
        $ib = mysqli_real_escape_string($db, $_GET['IB']);
        $ic = mysqli_real_escape_string($db, $_GET['IC']);
        $sa = mysqli_real_escape_string($db, $_GET['SA']);
        $sb = mysqli_real_escape_string($db, $_GET['SB']);
        $sc = mysqli_real_escape_string($db, $_GET['SC']);
        $pa = mysqli_real_escape_string($db, $_GET['PA']);
```

```
        $pb = mysqli_real_escape_string($db, $_GET['PB']);
        $pc = mysqli_real_escape_string($db, $_GET['PC']);
        $xa = mysqli_real_escape_string($db, $_GET['XA']);
        $xb = mysqli_real_escape_string($db, $_GET['XB']);
        $xc = mysqli_real_escape_string($db, $_GET['XC']);
        $f = mysqli_real_escape_string($db, $_GET['F']);
        $s = mysqli_real_escape_string($db, $_GET['S']);
        $p = mysqli_real_escape_string($db, $_GET['P']);
        $x = mysqli_real_escape_string($db, $_GET['X']);
        $pf = mysqli_real_escape_string($db, $_GET['PF']);
        $r = mysqli_real_escape_string($db, $_GET['R']);

        switch ($tx_id){

            case 101:
                echo " Transmisor RED 1 :";
                $sql = "INSERT INTO PM1(DATE, Tx_id, V_AN_V, V_BN_V, V_CN_V, V_AB_V, V_BC_V, V_CA_V,I_A_A, I_B_A, I_C_A, S_A_VA, S_B_VA, S_C_VA,
P_A_W, P_B_W, P_C_W, X_A_var, X_B_var, X_C_var, F_Hz, S_VA, P_W, x_var, PF)
VALUES(CURRENT_TIMESTAMP,'".$eq."','".$van."','".$vbn."','".$vcn."','".$vab."','".$vbc."','".$vca."','".$ia."','".$ib."','".$ic."','".$sa."','".$
sb."','".$sc."','".$pa."','".$pb."','".$pc."','".$xa."','".$xb."','".$xc."', '".$f."','".$s."','".$p."','".$x."','".$pf."')";
                break;

            case 102:
                echo " Transmisor RED 2 :";
                $sql = "INSERT INTO PM2(DATE, Tx_id, V_AN_V, V_BN_V, V_CN_V, V_AB_V, V_BC_V, V_CA_V,I_A_A, I_B_A, I_C_A, S_A_VA, S_B_VA, S_C_VA,
P_A_W, P_B_W, P_C_W, X_A_var, X_B_var, X_C_var, F_Hz, S_VA, P_W, x_var, PF)
VALUES(CURRENT_TIMESTAMP,'".$eq."','".$van."','".$vbn."','".$vcn."','".$vab."','".$vbc."','".$vca."','".$ia."','".$ib."','".$ic."','".$sa."','".$
sb."','".$sc."','".$pa."','".$pb."','".$pc."','".$xa."','".$xb."','".$xc."', '".$f."','".$s."','".$p."','".$x."','".$pf."')";
                break;

            default:
                echo " No se quien eres....";
                break;
        }
        // cosulta de Insercion
        if (!mysqli_query($db,$sql)){ echo("Error description: " . mysqli_error($db)); }
        else { echo "Escritura correcta"; }

        //cerrar base de datos
        mysqli_close($db);
    }
}
?>
```

ANEXO VIII: Manual de montaje, soldadura y programación del Transmisor

Proyecto GrowGreen Valencia

Alejandro Sánchez Aduna

Instituto de Ingeniería Energética

Universitat Politècnica de València

Valencia, diciembre de 2020

Índice

Índice de figuras

Nota: Todas las figuras de este documento son de fuente propia, a no ser que se especifique lo contrario.

1. Objeto del documento

El presente documento es un manual de montaje y programación del dispositivo *Transmitter* para la medida del estrés térmico, *Heat Stress Monitoring* (HSM), del proyecto GrowGreen Valencia.

Se tratarán una amplia variedad de temas: una descripción breve de cada uno de sus componentes y sensores, y se pasará al montaje, soldadura y programación del dispositivo. El objetivo final es que el usuario sea capaz de replicar el trabajo realizado hasta ahora y continuar con su desarrollo o resolver problemas que puedan aparecer en caso de que sea necesario.

A continuación, se describirá el proceso de montaje de un dispositivo transmisor. El proceso se referirá a un transmisor completo, con todos los compontes que pueden integrarlo, salvo aquellos que no son utilizados habitualmente y se dejan como opcionales (como el sensor de gas o el módulo de almacenamiento RTC + microSD). Este montaje es válido para transmisores más simples, pudiendo el usuario saltarse el montaje de aquellos sensores y componentes que no considere necesarios.

Figura 1. Transmisor completo listo para instalar.

2. Ensamblaje de la estructura

En primer lugar, se procederá al montaje de la estructura del dispositivo, que incluye la caja y los accesorios impresos en 3D, y los elementos mecánicos y de sujeción, como tornillos y abrazaderas. A continuación, se detalla la lista de elementos necesarios para el montaje, y cómo obtener cada uno de ellos. Posteriormente, se detallará el proceso de ensamblaje. A todas las piezas de la lista que sean impresas en 3D deben retirársele los soportes y pintarse con pintura en spray blanca y barniz.

2.1. Lista de componentes necesarios

- **Caja:** Caja del dispositivo impresa en 3D a partir del archivo "*caja nueva.stl*".
- **Soporte del sensor de temperatura de globo negro:** Impreso en 3D a partir del archivo "*palo.stl*". Si no se va a usar el sensor de globo negro, puede sustituirse por una tapa enroscada a partir del archivo *"rosca palo.stl"*.
- **Soporte del sensor de radiación:** Soporte del sensor de radiación i de la cúpula protectora. Impreso en 3D a partir del archivo "*soporte cupula.stl*".
- **Soporte del sensor de viento:** Soporte del sensor de viento de hilo caliente REV. C. Impreso en 3D a partir del archivo "*soporte anemo.stl*". Alternativamente, si no se va a usar el sensor de hilo caliente, puede sustituirse esta pieza por la del archivo *"soporte anemo tapado.stl"*.
- **Platos de la pantalla de protección de sensores:** Platos impresos en 3D a partir del archivo *"platos.stl"*. Se trata en total de tres platos inferiores (*"plato inferior.stl"*) y un plato superior (*"plato superior.stl"*).
- **Rejilla para sensores:** Rejilla de soporte para los sensores en el interior de la pantalla de protección. Impresa en 3D a partir del archivo *"reja.stl"*.
- **Tapón con rosca:** Pieza usada para sellar el agujero de acceso al botón de reinicio de la caja. Impresa en 3D a partir del archivo *"rosca tapa.stl"*.
- **Disco de soporte:** Disco para dar estabilidad a la pantalla de sensores. Impreso en 3D a partir del archivo *"disco.stl"*.
- **4x Varillas de aluminio Ø6:** Varillas huecas de aluminio de diámetro 6 milímetros cortadas a 98 mm de longitud (o, alternativamente, a 45 mm).
- **4x Varillas de aluminio Ø8:** Varillas huecas de aluminio de diámetro 8 milímetros cortadas a 53 mm de longitud.
- **2x Tornillos M10 DIN 933:** Tornillos M10 para las abrazaderas, de aproximadamente 10 cm de longitud.
- **2x Arandelas planas M10 DIN 9021:** Arandelas planas M10 DIN 9021.
- **2x Arandelas autoblocantes M10 DIN 6795-A:** Arandelas autoblocantes M10 DIN 6795-A, preferiblemente negras o de acero inoxidable.

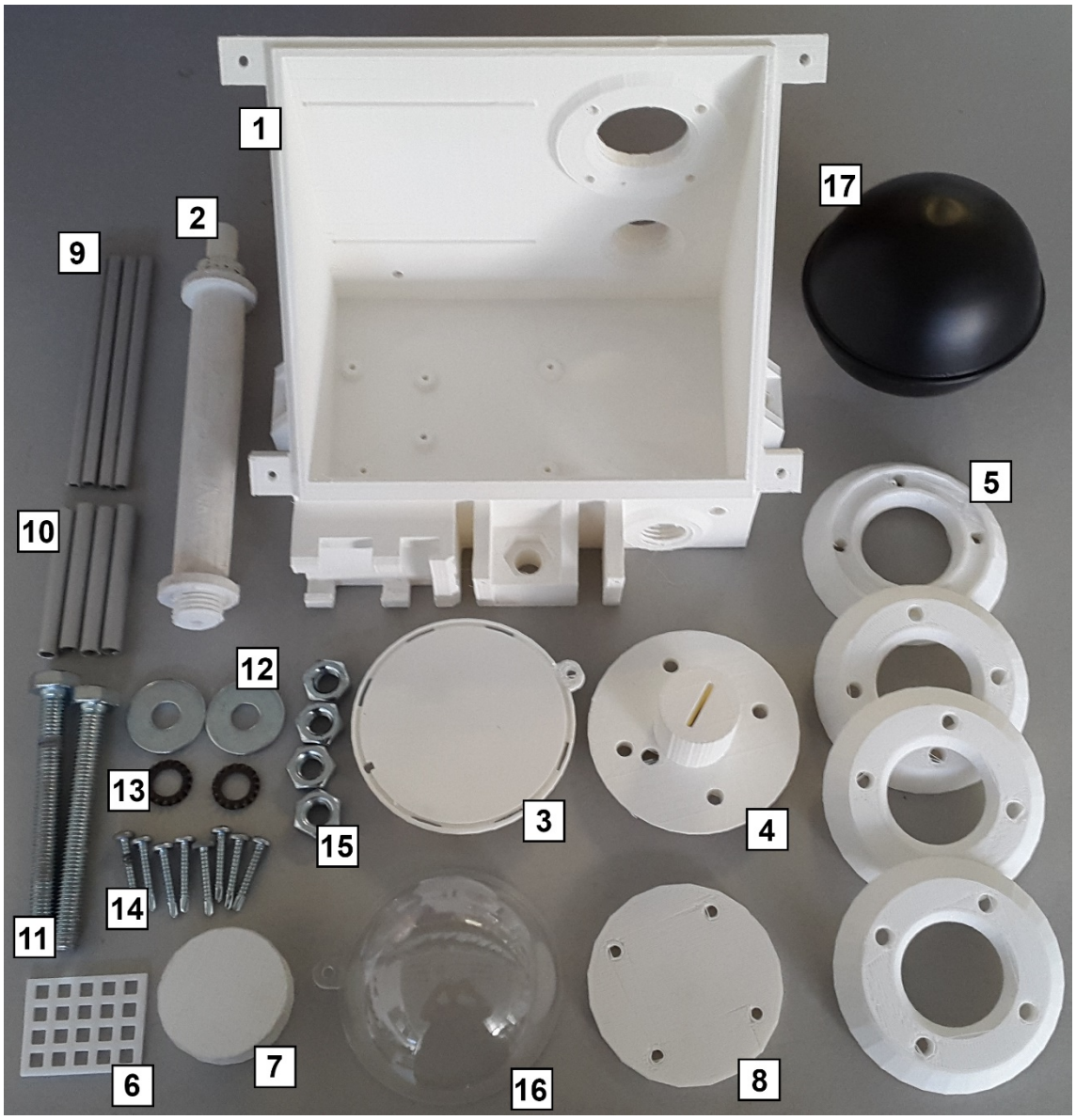


Figura 2. Piezas necesarias para el montaje de la estructura.

- **8x Tornillos autotaladrantes DIN 7504-N Ø4,2:** Tornillos autotaladrantes de estrella DIN 7504-N de diámetro 4,2 mm y alrededor de 25 mm de longitud.
- **4x Tuercas M10 DIN 934:** Tuercas M10 DIN 934.
- **Cúpula de metacrilato:** Cúpula de protección para el sensor de radiación de metacrilato transparente y radio 80 mm.
- **Esfera de "globo negro":** Esfera de latón de 80 mm de diámetro pintada con pintura en spray negra, con una rosca hembra de PVC de 20 mm de diámetro.
- **Extras:** Tornillo de estrella M3 DIN 965 de 20 y 30 mm, tuercas M3 DIN 934, arandelas planas M3 DIN 9021.
- **Abrazaderas:** Abrazaderas isofónicas reforzadas con goma de diferentes diámetros, dependiendo del montaje (por defecto 125 mm).

2.2. Ensamblaje de la estructura

Teniendo todos los componentes preparados, es momento de iniciar el montaje. El primer paso consiste en montar la pantalla de protección para los sensores. Para ello, usaremos las **barras de aluminio**, que servirán de pilares de sujeción. Deberán haber sido cortadas a las medidas indicadas anteriormente con la ayuda de una sierra de mano, y preferiblemente con los bordes limados con una lima para metal o con una taladradora tipo Dremel. Primero, introducimos la barra de diámetro 6 mm y la introduciremos en la barra de diámetro 8 mm con la ayuda de un martillo hasta que coincidan en uno de los extremos. Repetimos el proceso 4 veces.

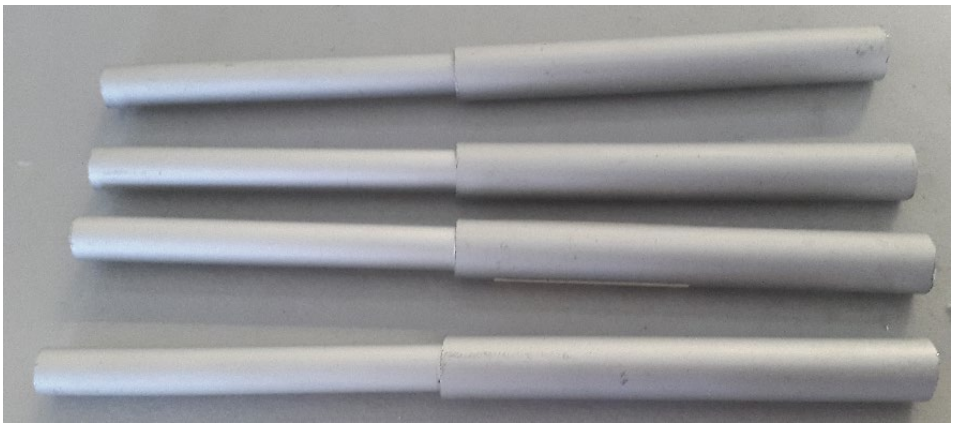


Figura 3. Pilares de aluminio para la pantalla de sensores.

Una vez obtenidos los pilares, introdúzcalos por los orificios de la pieza *"**Soporte anemo**"* (o, por defecto, *"Soporte anemo tapado"* si no se va a usar el sensor de viento de hilo caliente REV C.), de forma que quede como se muestra en la *Figura 4*.

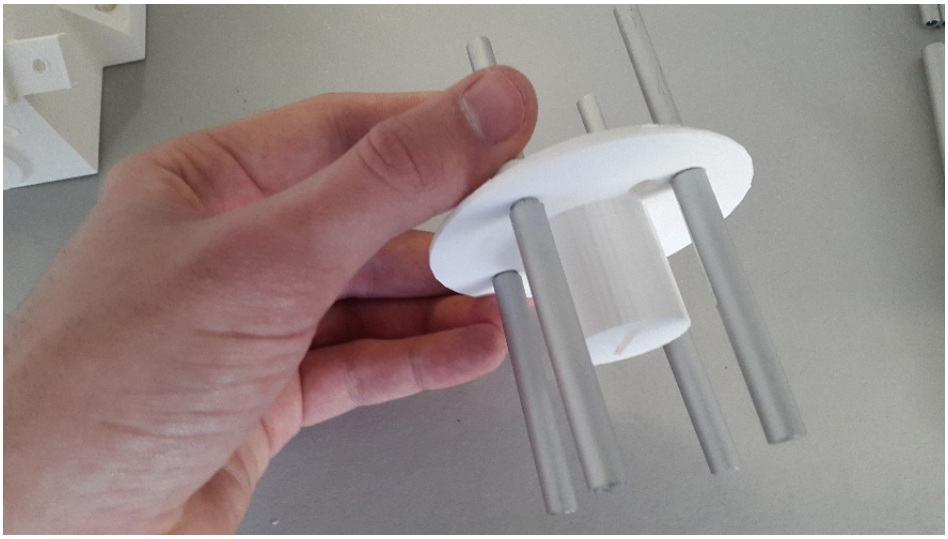


Figura 4. Montaje del soporte del sensor de viento.

Sobre esta pieza, introduciremos los **cuatro platos** que conforman la pantalla. Introduciremos primero los tres platos inferiores y después el plato superior, de forma que el luego coincida con las muescas presentes en la caja. Es posible que sea necesario agrandar los orificios de los platos con la ayuda de un destornillador ancho para que las barras de aluminio puedan introducirse fácilmente. También puede usarse un martillo, con cuidado de no dañar la pieza impresa.

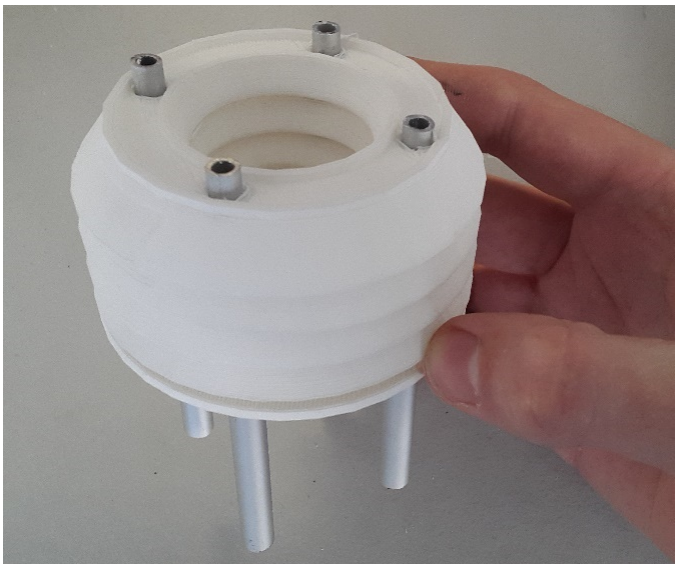


Figura 5. Montaje de los platos para la pantalla de sensores.

El siguiente paso consiste en el montaje del **anemómetro de copas**. Use los tornillos autotaladrantes para fijar el anemómetro en el lado ancho de los pilares a través de los orificios del propio sensor. No es necesario apretar los tornillos ahora, se hará en el siguiente paso. En caso de que no se vaya a instalar el anemómetro de copas, atornille la pieza *"Disco"* en su lugar.

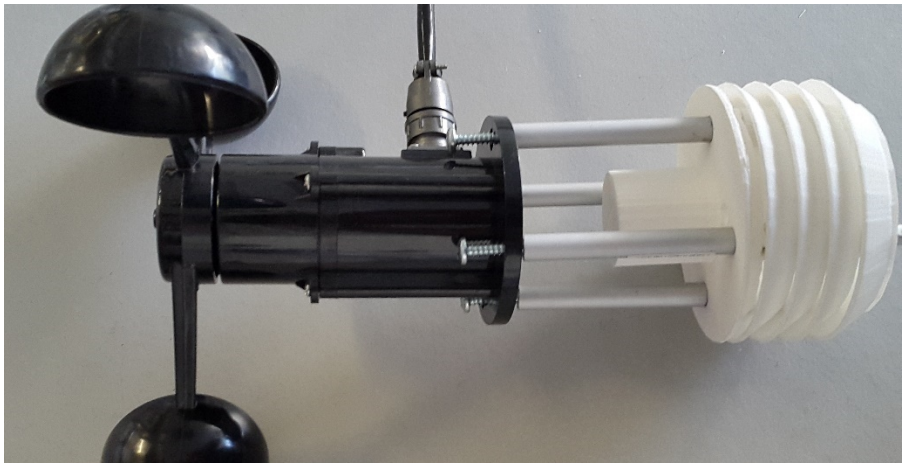


Figura 6. Montaje del anemómetro de copas.

Como **alternativa**, si no se va a usar ningún sensor de viento, también es posible cortar las barras de aluminio finas a 45 mm de longitud, introducir la pieza *"Soporte anemo tapado"* y los platos de la misma forma que antes y colocar los tornillos directamente sobre el soporte del anemómetro, de forma que se prescinde de las barras gruesas y se obtiene un dispositivo más compacto.

A continuación, se procederá a **acoplar la pantalla de sensores** a la caja. Una el conjunto a las muescas presentes en la forma que coincidan los cuatro orificios de los pilares. Introduzca cuatro tornillos más y usando dos destornilladores o un taladro apriete los ocho tornillos intentando que no quede ningún espacio entre el plato superior y la caja. Ahora puede introducir el cable del anemómetro por el orificio del soporte. Asegúrese de haber orientado la pantalla tal como se muestra en la *Figura 7*. Esto será importante más adelante, ya que de lo contrario este cable y la posición del soporte pueden dificultar la colocación del resto de sensores.

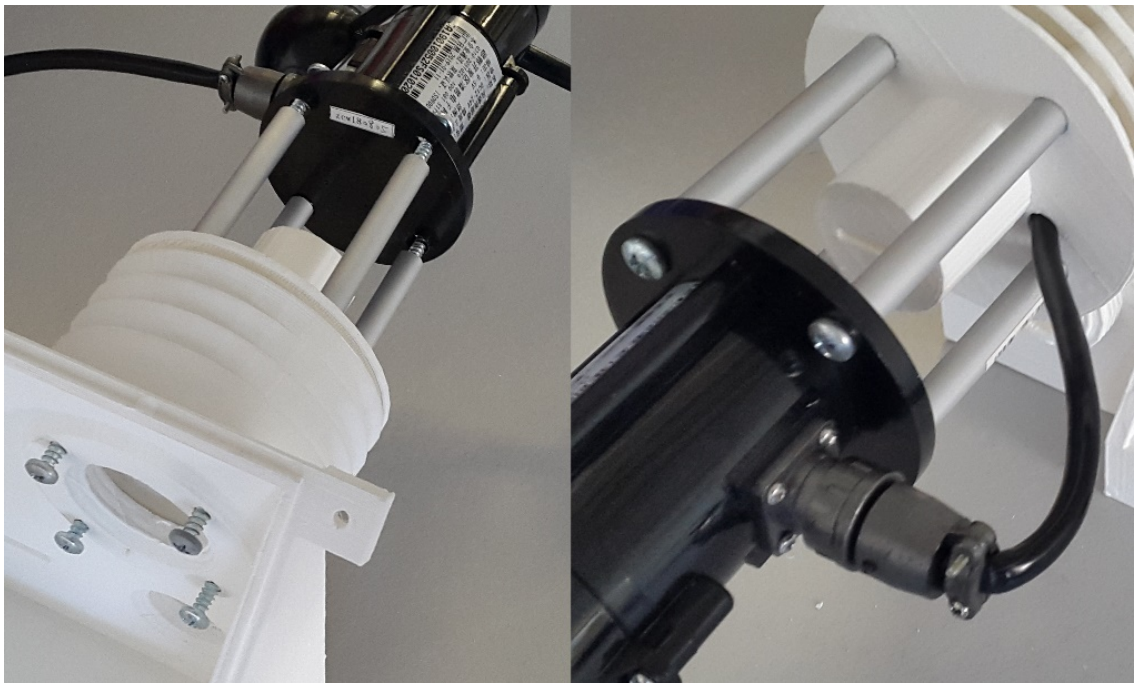


Figura 7. Acople de la pantalla de sensores en la caja.

Tras esto, se procederá a **instalar las abrazaderas** para la instalación. Para ello, se colocarán primero las sujeciones mecánicas, usando los huecos hexagonales presentes en la parte superior e inferior de la caja. Los elementos se colocarán en el siguiente orden (véase *Figura 8*): tornillo M10, arandela plana M10, arandela autoblocante M10 y dos tuercas M10.

Finalmente, se atornillarán las abrazaderas del diámetro adecuado dependiendo del lugar en que vayan a ser instaladas (por defecto, suelen ser de diámetro 125 mm) y se apretarán las tuercas con una llave inglesa. Si se desea, se puede saltar este paso y colocar las abrazaderas al final, justo antes de la instalación.

Figura 8. Instalación de las abrazaderas.

Siguiendo con el montaje, a continuación, se montará el **sensor de temperatura de globo negro**. Tome la pieza *"Palo"*, y envuelva ligeramente la rosca ancha de la pieza con cinta de teflón, de forma que al enroscar la pieza en el orificio superior de la caja éste quede bien sellado, con cuidado de no forzar el plástico (posteriormente, se usará silicona de montaje para completar el sellado). Fíjese en el sentido de giro en el que envuelve la cinta de teflón para que al enroscar la pieza ésta no se retire. Ahora introduzca también el sensor de temperatura DS18B20 encapsulado por el orificio de la pieza, dejando una longitud de cable restante de aproximadamente unos 20 cm. Corte el exceso de cable con unos alicates. Finalmente, enrosque el globo negro en la rosca superior de la pieza cogiéndolo por su soporte hexagonal. Nunca enrosque el globo agarrándolo por la parte esférica, ya que podría arrancarla de su soporte. Si es necesario, ayúdese de una llave inglesa.

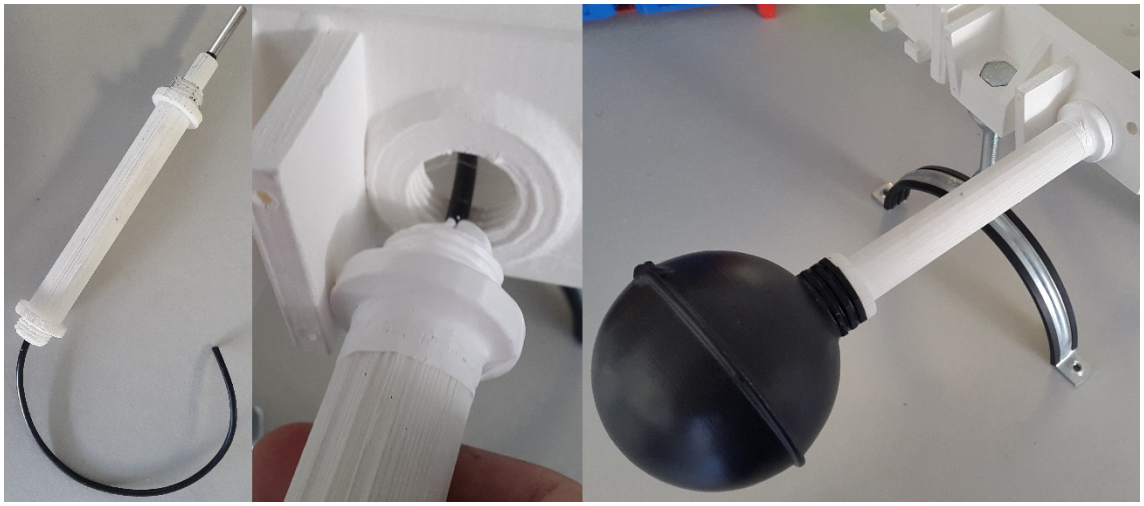


Figura 9. Montaje del sensor de temperatura de globo negro.

Por último, y si el dispositivo va a usar sensor de radiación superior, introduzca la pieza *"Soporte cúpula"* en su lugar correspondiente. Ayúdese de un martillo si es necesario.

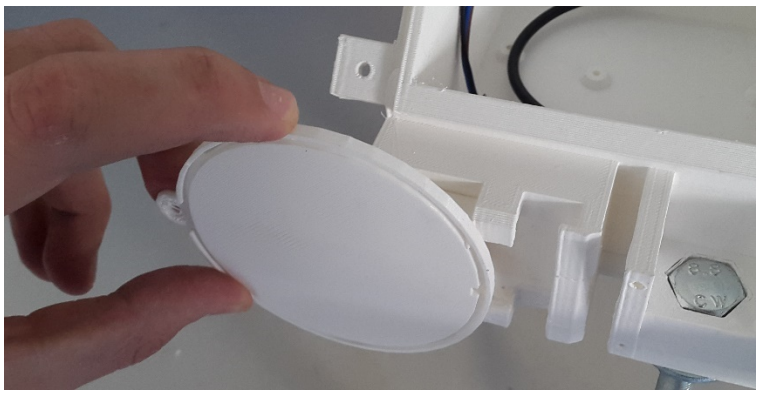


Figura 10. Montaje del soporte de la cúpula.

3. Soldadura del circuito electrónico

Una vez ensamblados la mayoría de los elementos mecánicos, es el momento de proceder a la soldadura de la electrónica del dispositivo. Al igual que en el apartado anterior, primero se detallará la lista de componentes necesarios y, posteriormente el proceso de soldadura y montaje.

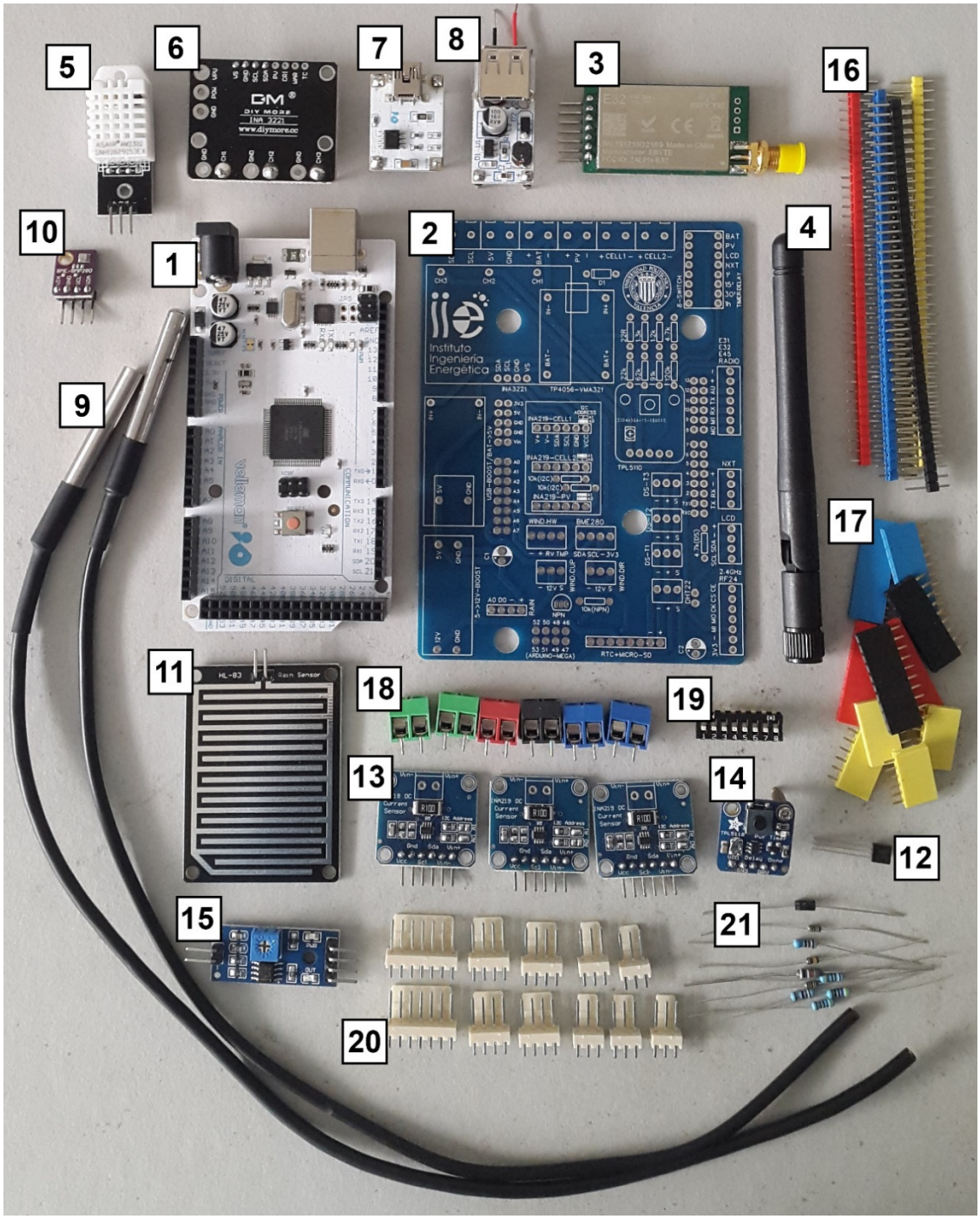


Figura 11. Piezas necesarias para el montaje de la electrónica.

3.1. Lista de componentes necesarios

- **Placa Arduino Mega 2560:** Placa de desarrollo Arduino Mega 2560, con procesador ATmega2560 y pines ya soldados.
- **PCB GrowGreen v2:** Segunda versión de la placa de circuito impreso desarrollada para el proyecto, de color azul.
- **Módulo de radiofrecuencia E32-TTL-1W:** Módulo de comunicaciones por radiofrecuencia a 433 MHz E32-TTL-1W, conector hembra.
- **Antena 433 MHz:** Antena de 433 MHz con conector macho.
- **Sensor DTH22:** Sensor de humedad y temperatura DHT22 ya soldado. (En desuso. Usar en su lugar el sensor AM2320).
- **Sensor INA3221:** Sensor de tensión y corriente de tres canales INA3221.
- **Módulo de carga VMA-321:** Módulo cargador de baterías de litio VMA-321.
- **Convertidor de tensión VMA-403:** Convertidor DC-DC step-up a 5V con USB.
- **2x Sensor DS18B20 encapsulado:** Sensor de temperatura digital DS18B20 con encapsulado metálico. El número de sensores varía en función de las necesidades (por defecto, 2).
- **Sensor BME280:** Sensor de presión BME280 con comunicación I2C.
- **Placa sensor de lluvia:** Placa resistiva de sensor de lluvia MH-RD.
- **Chip sensor DS18B20:** Sensor DS18B20 en encapsulado TO-92. (En desuso)
- **3x Sensor INA219:** Sensor de tensión y corriente INA219. El número de sensores varía en función de las necesidades (por defecto, 3).
- **Temporizador TPL5110:** Módulo temporizador de bajo consumo TPL5110.
- **Módulo para sensor de lluvia:** Módulo convertidor analógico digital y analógico de 5 V para sensor de lluvia MH-RD.
- **Tiras de pines macho:** Tiras de pines rectos y acodados macho de 2,54 mm en varios colores.
- **Tiras de pines hembra:** Tiras de pines hembra de 2,54 mm en varios colores.
- **Borneras de dos pines:** Borneras atornilladas de dos pines y 5,08 mm.
- **Interruptor 8 canales:** Interruptor THT de 8 canales y 2,54 mm.
- **Conectores para PCB macho:** Conectores para PCB Molex, 2,54 mm, rectos THT macho, de una fila y diferente número de pines.
- **Resistencias estándar 1% y diodo Schottky:** Resistencias estándar de precisión con tolerancia 1% de diferentes valores: 22 kΩ, 22 Ω, 68 kΩ, 470 Ω,

91 kΩ, 1,2 Ω, 120 kΩ, 2x 4,7 kΩ. Diodo Schottky THT con tensión umbral inferior a 0,3 V.

- **Extras:** 3x Resistencias estándar de 10 kΩ. Transistor NPN tipo 2N2222. Tornillos de cabeza redonda M3, de aproximadamente 5 mm y preferiblemente magnéticos. Espaciadores pequeños para PCB de diferentes longitudes.

- **Sensor AM2320:** Sensor de temperatura y humedad. Versión actualizada del DHT22, con comunicación por I2C.

3.2. Soldadura y montaje de la electrónica

Una vez enumerados los componentes necesarios, procedemos a describir el proceso de soldadura y montaje de la electrónica. En primer lugar, tomaremos la PCB y, con la ayuda de un cúter, **cortaremos una de las pistas** de cobre situadas en la parte trasera. Se trata de la pista que va desde el pin *Done* del *timer* TPL5110 y el pin 13 del Arduino MEGA. Compruebe que no haya continuidad entre estos puntos con la ayuda de un multímetro.

Figura 12. Corte de la pista de cobre en la PCB.

A continuación, soldaremos las **tiras de pines macho y hembra** a la PCB, que nos permitirán acoplarla al Arduino MEGA y usarla como *shield*. Empezaremos soldando los pines macho, situados en la fila interior, introduciéndolos por la cara inferior de la PCB. Una buena forma de hacerlo es cortar las tiras de pines a la medida correcta y encajarlos en su sitio en el Arduino MEGA, de tal forma que simplemente se coloque la PCB encima y se puedan soldar todos los pines fácilmente, quedando así bien orientados.

Después, se soldarán las tiras de pines hembra, que nos permitirán conectar los distintos sensores, además de acceder fácilmente a los pines del Arduino MEGA. Para realizarlo, se recomienda cortar las tiras al tamaño adecuado (retirando el plástico sobrante), poner un poco de estaño en la punta del soldador y, mientras sujeta la tira en su lugar con una mano, con la otra suelde uno de los pines de la tira, lo que sujetará toda la tira en su lugar y le permitirá soldar el resto de pines. Asegúrese que la tira está bien recta antes de soldar todos los pines. En la *Figura 13*, tiene una referencia de dónde va cada una de las tiras y su longitud. El esquema de colores es tan solo orientativo.

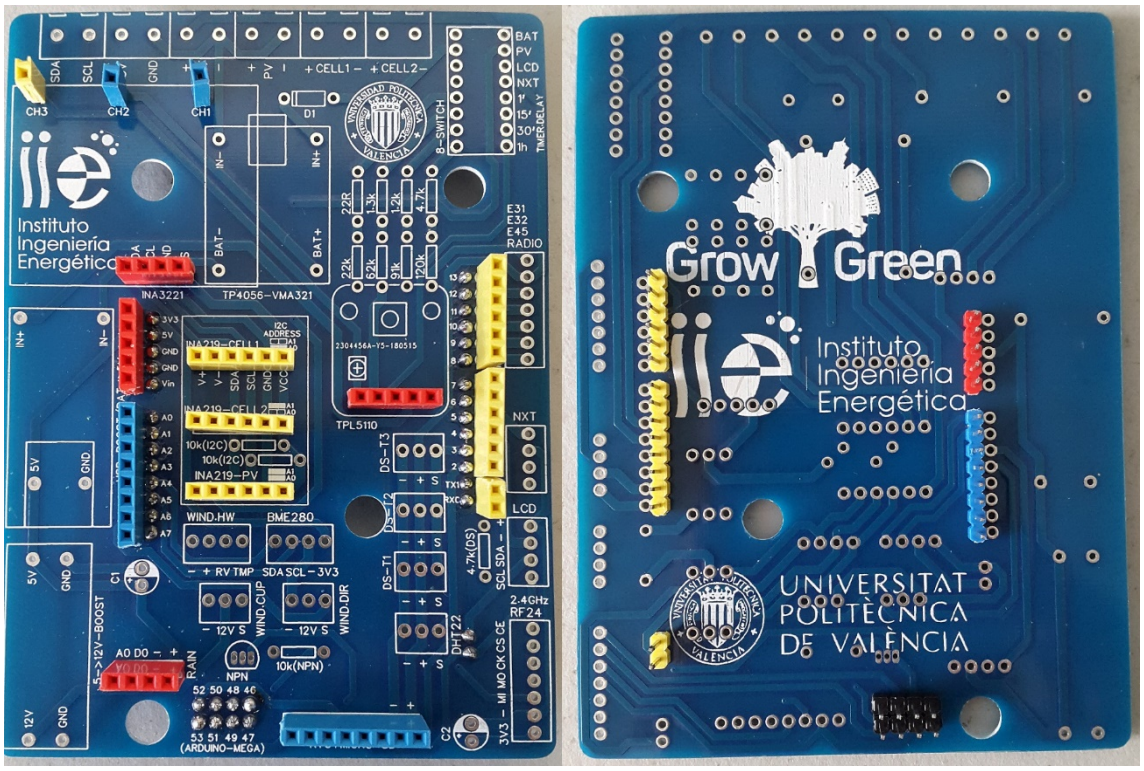


Figura 13. Posición de las tiras de pines en la PCB.

A continuación, soldaremos la tira de **borneras**, encajándolas las unas con las otras en sus pestañas e introduciéndolas en los agujeros correspondientes en la PCB. En este caso es recomendable seguir un patrón de colores para diferenciar la entrada de baterías, panel solar y sensores de radiación. Nótese que, después de la soldadura, se ha cortado uno de los pines con unos alicates. Esto es necesario para evitar que este pin entre en contacto con el conector USB del Arduino MEGA.

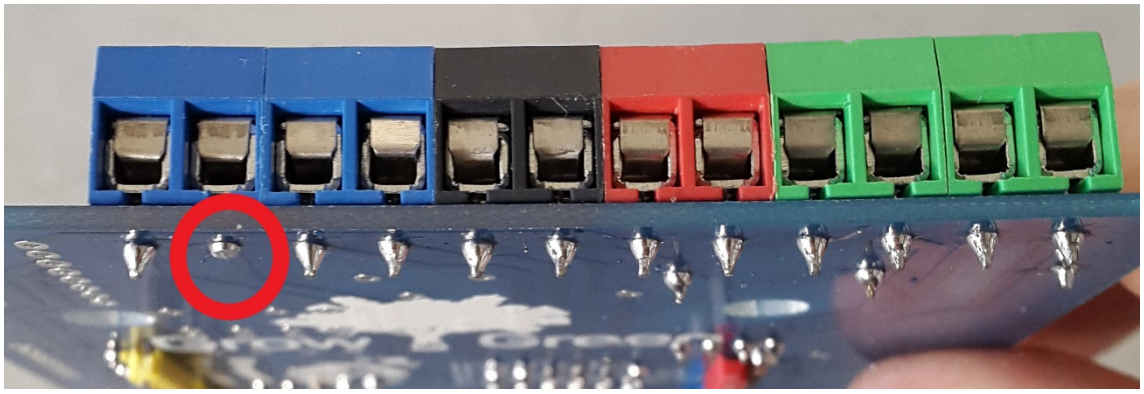


Figura 14. Detalle de la colocación de las borneras.

El siguiente paso consiste en **soldar los pines de todos los componentes**, para que puedan encajarse en su lugar adecuadamente. En muchos casos, es muy útil colocar tanto los pines como el componente en su lugar de la PCB y soldarlo directamente en su lugar, como en el caso del módulo de carga, el INA3221 o el convertidor de 12 V. En la *Figura 16*, puede ver cómo deben quedar los pines de cada uno de estos componentes.

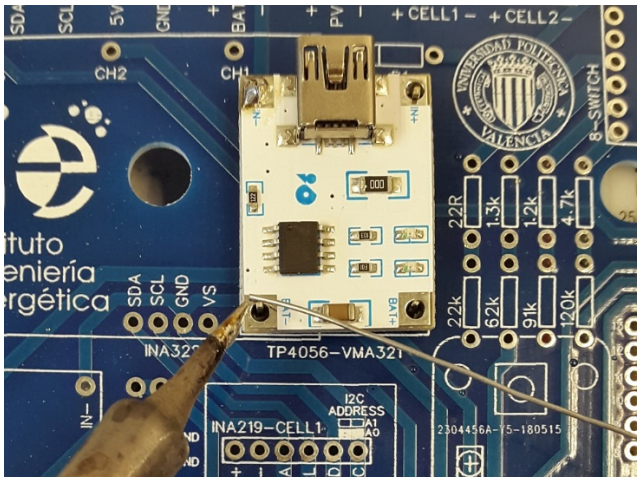


Figura 15. Soldadura de los componentes directamente en la placa.

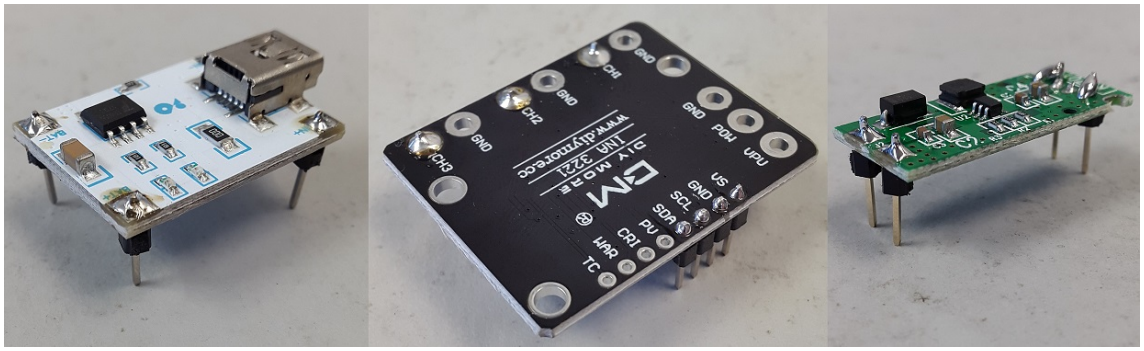


Figura 16. Referencia de soldadura de algunos componentes.

Además, en el INA3221, será necesario unir el puente de estaño marcado como GND. Esto permite seleccionar en el bus I2C la dirección 0x40, de forma que no interfiera con los demás componentes del bus, especialmente los INA219.

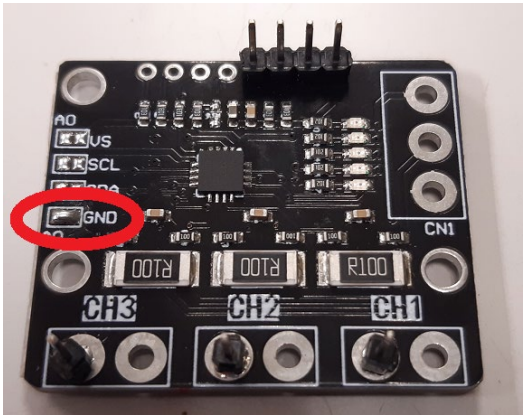


Figura 17. Puente de estaño en el sensor INA3221.

En otros casos, como en el ***timer* TPL5110**, es mejor sujetar los pines con una mano o con un tercer brazo para soldadura. Para este componente, además, será necesario cortar con un cúter el *jumper* de cobre situado en la parte posterior del sensor etiquetado como *"Trim enable"*. Esto permitirá que se pueda regular el tiempo de desconexión del sensor mediante las resistencias externas de la PCB en vez de por el potenciómetro que trae incorporado. Finalmente, se le pueden atornillar unos espaciadores que sirvan de soporte para poder pulsar el botón de reinicio fácilmente.

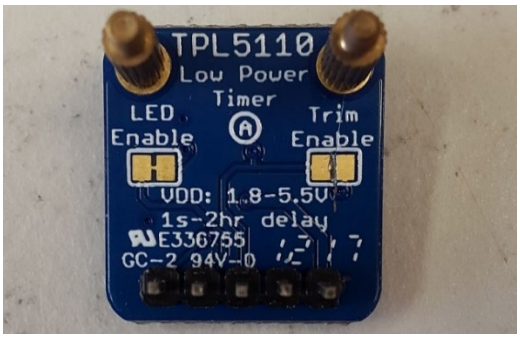


Figura 18. Soldadura y preparación del *timer* TPL5110.

En el caso del **módulo USB**, será necesario soldar un par de cables cortos directamente en los pines del conector USB, correspondientes a la salida de 5 V. Asegúrese de que queden bien sujetos y que no se suelten al moverlos. Una vez sujetos, se puede soldar el módulo USB en su sitio, insertando primero los cables en sus orificios y doblándolos ligeramente hasta que todo el módulo esté en la posición adecuada y pueda soldar los pines.

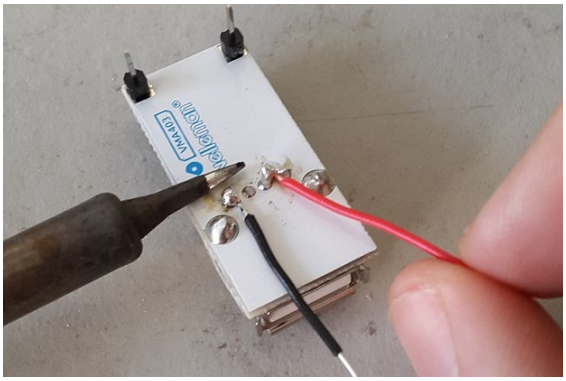


Figura 19. Soldadura de los cables del módulo de tensión de 5 V.

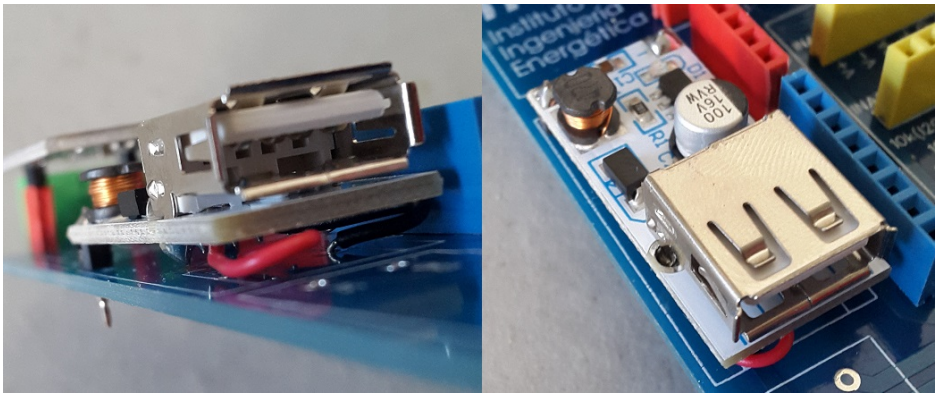


Figura 20. Módulo USB soldado.

Tenga especial cuidado con el **módulo de radio** E32-TTL-1W, ya que será necesario retirar la tira de pines rectos que trae de fábrica y sustituirlos por una tira de pines acodados. Este módulo tiene una gran disipación de calor, por lo que fundir el estaño de sus pines puede ser complicado, especialmente el pin de GND. En este caso, se recomienda cortar los pines entre sí y retirarlos uno a uno con unos alicates, calentando con la punta del soldador y añadiendo nuevo estaño en caso de que sea necesario. Finalmente, retire el estaño sobrante que obstruye el pad con la ayuda de una bomba desoldadora y más estaño nuevo. Cuando todos los agujeros estén limpios,

introduzca la nueva tira de pines y suelde los pines uno a uno asegurándose de que quede recta. No introduzca nunca la punta del soldador directamente en el agujero, ya que podría hacer saltar el pad. En caso de que eso suceda, rasque con cuidado la capa de recubrimiento aislante en los alrededores del pad (en la pista o plano de masa que esté conectado a éste) hasta que el cobre quede al descubierto, lo que le permitirá soldar el pin directamente al plano de masa.

Figura 21. Soldadura de los pines del módulo de radio.

Finalmente, prepararemos los **sensores de corriente INA219**. El número de sensores que necesitemos dependerá de los que queramos incluir en el dispositivo, véase: uno para el sensor de radiación superior, uno para el sensor de radiación inferior y otro para el panel solar. Los dos primeros dependerán del lugar donde se coloque el dispositivo. Sin embargo, por lo general, se puede prescindir de la medida de corriente del panel solar, ya que el INA3221 ya nos da una medida de la corriente que sale del módulo de carga. Si prescindimos de alguno de los sensores de radiación, será tan sencillo como no colocar su sensor de corriente. Si prescindimos del sensor del panel solar, será necesario colocar un *jumper* entre los pines V+ y V- del sensor para que se pueda cerrar el circuito del panel solar. Este *jumper* se puede fabricar simplemente colocando una tira de dos pines macho entre los pines V+ y V- y unirlos con el soldador.

Para los INA219 que sí coloquemos, deberemos **retirar las dos resistencias *pull-up*** que vienen incorporadas en el sensor, ya que pueden interferir en el buen funcionamiento del bus I2C, reduciendo en exceso la resistencia del bus. Recordemos que es recomendable mantener una resistencia entre los pines SDA y VCC, y SCL y VCC de alrededor de 5 kΩ. Finalmente, seleccionaremos la **dirección I2C** de los módulos, uniendo o separando con estaño los puentes A0 y A1 presentes en la placa. En la propia PCB vienen indicados los puentes que es necesario cerrar para cada una de las posiciones.

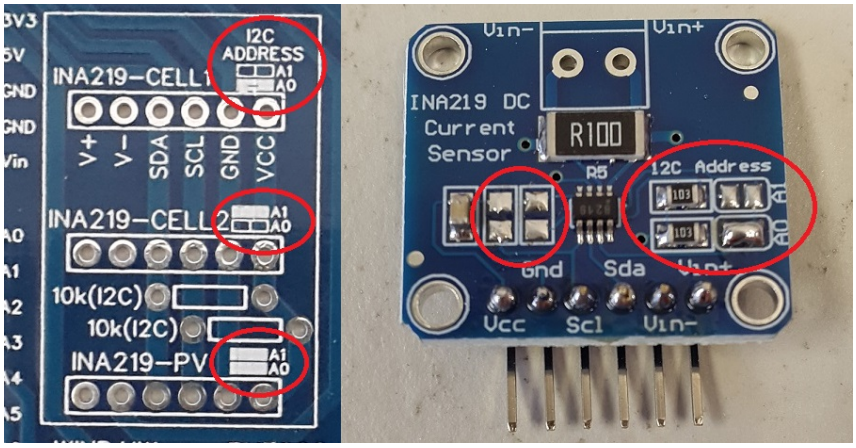


Figura 22. Preparación de los sensores INA219.

Una vez preparados los sensores que se insertan en la PCB, el siguiente paso consiste en soldar los **componentes pequeños**. Así, soldaremos el interruptor de 8 canales, diodo, resistencias, transistor NPN (en desuso), respetando la polaridad en cada caso. Deberá prestar especial atención a las **resistencias del *timer* TPL5110**. Estas resistencias se encargan de controlar el tiempo de reinicio del *timer*, y una pequeña variación en estas resistencias puede suponer una diferencia de minutos en la toma de las medidas. Para soldar los valores adecuados, tome como referencia la *Figura 25*, ya que los valores marcados en la serigrafía de la PCB no son correctos.

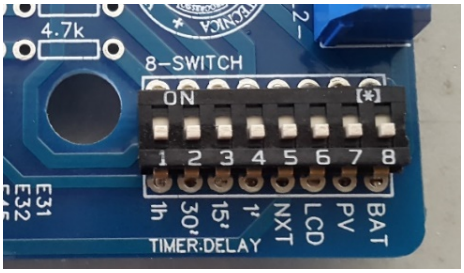


Figura 23. Orientación correcta del interruptor de 8 canales.

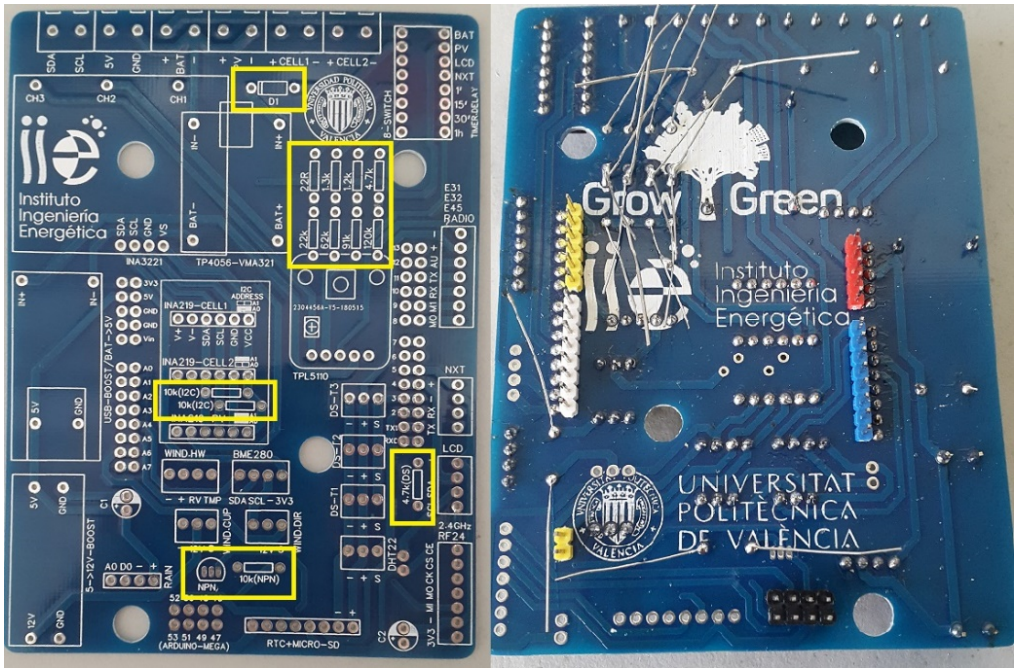


Figura 24. Soldadura de componentes pequeños.

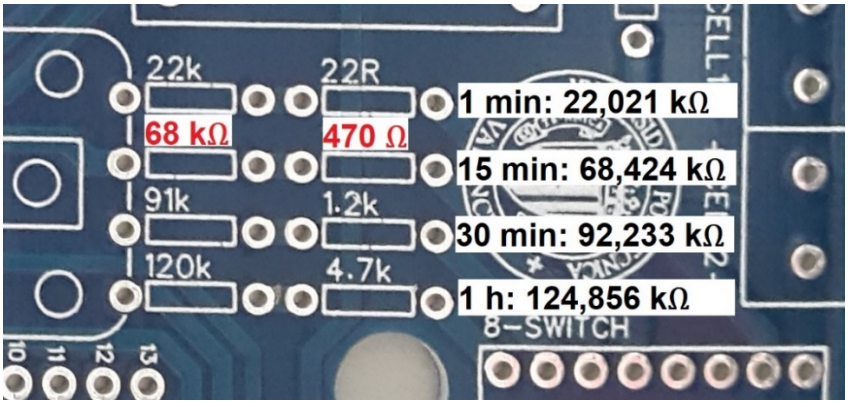


Figura 25. Valores de las resistencias del *timer* TPL5110.

El último paso en la soldadura de la electrónica consiste en soldar los **conectores Molex** para el resto de sensores y periféricos. El método más recomendable para que queden colocados correctamente es el siguiente: colocar el conector en su lugar, fundir un poco de estaño en la punta del soldador y acercar la punta a uno de los orificios manteniendo el conector bien orientado con los dedos (con cuidado de evitar quemaduras) hasta que el estaño se haya enfriado y el conector se mantenga fijo. Entonces se podrá soldar normalmente el resto de pines y, finalmente, repasar con estaño el primero que se ha colocado. En la posición asignada al sensor DS-T1 se puede soldar el chip del sensor DS18B20, que correspondería a la temperatura del interior de la caja (TBOX). Sin embargo, esta medida está en desuso y se recomienda soldar en su lugar otro conector Molex de 3 pines.

Una vez terminado todo este proceso, se pueden insertar en su sitio los sensores preparados anteriormente, y el **resultado final** debería ser similar al de la *Figura 26*. Cuando el usuario se haya familiarizado con todos los pasos, es conveniente que se establezca una **soldadura en paralelo** para acelerar el proceso y soldar más placas al mismo tiempo.

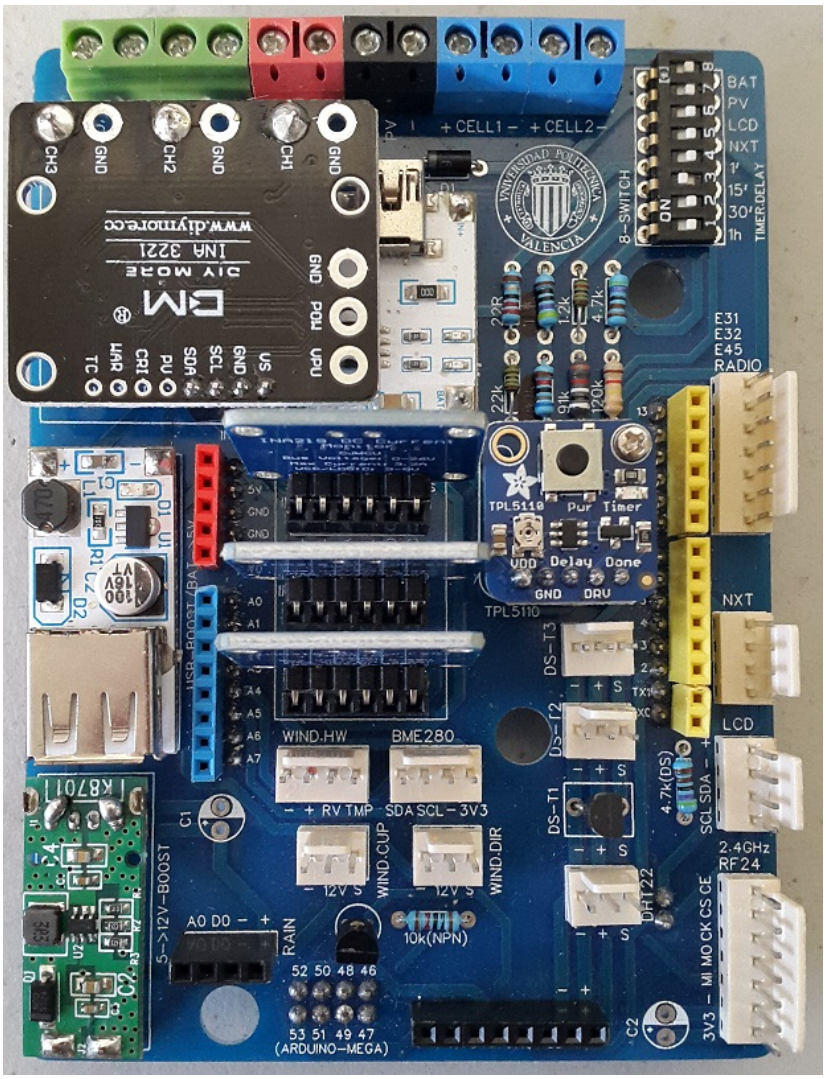


Figura 26. Aspecto final de una PCB terminada.

4. Programación del dispositivo

En esta sección, se describirá el proceso de programación del dispositivo, es decir, la carga de los respectivos programas a los dispositivos del transmisor, tanto a la placa base como al módulo de radiofrecuencia.

Antes que nada, asegúrese de que tiene instalada en su PC la **aplicación oficial de Arduino** y que tiene instaladas las **librerías** necesarias. Para instalar las librerías, copie las carpetas presentes en el directorio *"Librerías utilizadas"* y péguelas en la carpeta *"C:\Users\user\Documents\Arduino\libraries"*.

4.1. Configuración del módulo de radiofrecuencia

Antes de realizar el ensamblaje de los módulos de radiofrecuencia y las antenas, es necesario configurar adecuadamente el módulo de comunicación, sobre todo si acaba de sacarse de su embalaje. También es recomendable reprogramar el módulo si hay un error en las comunicaciones o si no se está seguro de si se ha programado o no.

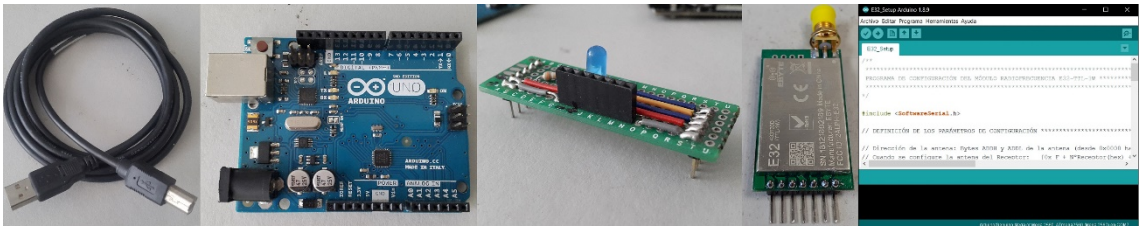


Figura 27. Elementos para la configuración del módulo de radio.

Primero, debemos reunir los **elementos necesarios**: módulo de radiofrecuencia E32-TTL-1W, placa Arduino UNO con *shield* personalizado para programación de radio, cable USB-A macho a USB-B macho y PC con la aplicación oficial de Arduino instalada. Este *shield* que se ha nombrado se trata de una placa para prototipos casera diseñada a medida especialmente para la programación del módulo de radio de forma que encaje en una placa Arduino UNO y en la que se pueda conectar el módulo fácilmente. El circuito es simple: sencillamente, conecta los pines digitales de la antena a las entradas y salidas digitales de la placa Arduino (los mismos que se usan en el transmisor) y los pines de alimentación al puerto de 5 V. Incluye además un diodo LED para poder observar el proceso de configuración.

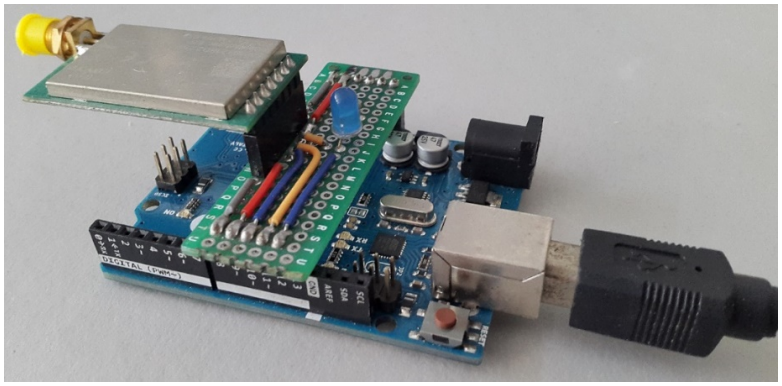


Figura 28. Conjunto Arduino+*shield* para configuración de radio.

Así pues, el proceso de configuración es el siguiente:

1) Acoplar el *shield* de configuración de radio a la placa Arduino UNO según se describe en el *Anexo III*, y conectar placa Arduino al PC con el cable USB.

2) Abrir en la aplicación para PC de Arduino el programa *"E32_Setup.ino"*.

3) Modifique la línea de código *"#define ADDRESS 0x????"* según corresponda siguiendo las anotaciones en el programa.

4) En el desplegable *"Herramientas - Placa"* seleccionar el modelo *"Arduino/Genuino UNO"*. A continuación, en el desplegable *"Herramientas - Puerto"*, seleccionar el puerto correspondiente a la placa conectada.

5) Pulsar la opción *"Verificar"* y, una vez compilado el programa, pulse *"Subir"*.

6) Nada más termine la subida del programa, observará en el diodo LED del *shield* una serie de parpadeos cortos y rápidos. Desconecte el cable USB del PC, espere un par de segundos y vuelva a conectarlo.

7) Observará un único parpadeo corto. Desconecte el cable USB y extraiga el módulo de radio. La configuración se ha realizado correctamente.

Si ya se ha cargado el programa en la placa Arduino anteriormente, puede omitir los pasos 2 a 7. Nada más conectar el Arduino al PC deberían producirse una serie de parpadeos rápidos y, tras desconectar y volver a conectar, un único parpadeo corto que indica que la configuración ha sido correcta.

4.2. Programación del transmisor

Para la programación del dispositivo transmisor, tan sólo son necesarios una placa Arduino MEGA, un PC con la aplicación oficial de Arduino instalada y un cable USB-A macho a USB-B macho. Por ello, este paso puede realizarse en cualquier momento del montaje, aunque se recomienda realizarse en la fase final del montaje, durante o después del cableado. Así, se pueden conocer tanto los sensores que va a contener el dispositivo como el identificador que se le va a asignar. Así pues, el proceso de programación es el siguiente:

1) Conectar placa Arduino al PC con el cable USB. Esto puede realizarse tanto con la placa Arduino sola como con la PCB acoplada, ya sea cableada o no, e incluso con las baterías conectadas (siempre que se haya comprobado la ausencia de cortocircuitos).

2) Abrir en la aplicación de Arduino el programa *"Transmitter_4.2_MEGA_E32.ino"*.

3) Acceda a la pestaña del programa llamada *"defines.h"*. En esa pestaña, se deben modificar los siguientes parámetros:

- *TX_ID*: Número identificador del dispositivo, y el que aparecerá como identificador de la base de datos.
- Variables de activación de sensores (como *DHT_ON* y *CELL1_ON*), cambiando a *true* aquellos sensores que estén presentes en el dispositivo y dejando en *false* aquellos que no se hayan instalado.

- *ITER_NUM* e *ITER_T*: Corresponde al número de medidas que se van a tomar y con las que se va a calcular la media (por defecto, 10), y el intervalo de tiempo entre cada una de las medidas (por los tiempos de lectura habituales de los sensores, es recomendable no introducir un valor inferior a 1000 ms). Cuanto mayor sea el valor, mayor será el tiempo de encendido y el consumo energético del dispositivo.
- *ANEMO_ADJ*: Constante de calibración del sensor de viento de copas. Depende del modelo de sensor que se instale.
- *CELL1_ADJ* y *CELL2_ADJ*: Constantes de calibración de los sensores de radiación superior e inferior. Este valor se encuentra anotado en el reverso de los sensores, y se recomienda apuntarlo también en el interior de la caja. En caso de no disponer de ese valor, puede dejarse por defecto.
- Además de los anteriores, aparecen otros parámetros destinados solamente a tareas de desarrollo, por lo que no se recomienda modificarlos.

4) En el desplegable *"Herramientas - Placa"* seleccionar el modelo *"Arduino/Genuino MEGA or MEGA 2560"*. A continuación, en el desplegable *"Herramientas - Puerto"*, seleccionar el puerto correspondiente a la placa conectada.

5) Pulsar la opción *"Verificar"* y, una vez compilado el programa, pulse *"Subir"*. Es posible que salte un mensaje de tipo *warning* en relación a una conversión de unidades. Puede ignorarse sin problema.

6) Una vez se muestre el programa como subido, puede desconectar el cable USB. La programación se ha realizado correctamente.