Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Análisis, Diseño, Control e Implementación de un Sistema Inteligente de Regadío

Autor:

Moreno Hernangómez, Guillermo

Tutor:

Moya de la Torre, Eduardo Julio

Ingeniería de Sistemas y Automática

Valladolid, Diciembre 2020.

AGRADECIMIENTOS:

Me encantaría agradecer de antemano a todas aquellas personas que, por uno o algún otro motivo, han confiado en mí desde el primer momento. Sobre todo, reconocer el apoyo y ayuda recibida por mis amigos, compañeros y profesores dentro y fuera de la Universidad; en especial agradecimiento al alumno, compañero y, en primer lugar, amigo Mario Montes Sampedro, porque sin sus aportaciones, bien sabe Dios dónde estaría ahora.

CITAR BREVEMENTE CUATRO DICHOS MUY FAMOSOS DEL FILÓSOFO VOLTAIRE PARA LA OCASIÓN QUE ÉL MISMO ENTENDERÁ:

1. "LA IDIOTEZ ES UNA ENFERMEDAD EXTRAORDINARIA, NO ES EL ENFERMO EL QUE SUFRE POR ELLA, SINO LOS DEMÁS".
2. "¡DIOS MÍO, LÍBRAME DE MIS AMIGOS! DE LOS ENEMIGOS YA ME ENCARGO YO".
3. "LOS EJEMPLOS CORRIGEN MUCHO MEJOR QUE LAS REPRIMENDAS".
4. "NO ESTOY DE ACUERDO CON LO QUE DICES, PERO DEFENDERÉ CON MI VIDA TU DERECHO A EXPRESARLO".

Hacer una peculiar noción a mis padres por todo el aporte recibido, tanto económico como afectivo; a mi hermana por todo el cariño y entendimiento recibido y a toda mi familia por preocuparse por mi cada uno a su modo. También mencionar a mis amigos, ellos saben bien quién son, por estos años tan espectaculares de carrera con sus grandes aportaciones en todos los ámbitos de la vida. Gracias 'mi gente'.

Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

MEMORIA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Resumen y Palabras clave:

Resumen:

Este trabajo fin de grado presenta la formar de controlar, de forma inteligente, un sistema de riego, principalmente, mediante sensores y una placa base Arduino Mega2560 R3, entre otros módulos; partiendo desde su diseño, pasando por su desarrollo y hasta llegar a su implementación.

La comprobación de la toma de datos se mostrará por pantalla, tanto en la establecida físicamente en la maqueta, como mediante WiFi por un servidor en la IP local del mismo. Para ello, es necesario ser consciente de la fecha y horas actuales que lo configura un 'Reloj de Tiempo Real'; al mismo tiempo, sirve para delimitar las franjas horarias óptimas de riego. Además, el uso de LEDs determina en cada caso el estado de la variable que se esté comprobando si nos encontramos físicamente en el regadío.

La fuente de alimentación variable de 3.3 o 5 V permite mantener una tensión constante sin que haya fluctuaciones en la misma. Del mismo modo, dos transformadores AC/DC sustentan el que permanezcan encendidos los módulos en todo momento, activando uno la fuente anterior y otro la placa Arduino.

Palabras clave:

Arduino, Entorno de Desarrollo Integrado (IDE), Reloj de Tiempo Real (RTC), fuente de alimentación, transformadores AC/DC, lenguaje de programación, Sistema en un Chip (SoC),

Abstract and Keywords:

Abstract:

This final degree project presents the way to control, in an intelligent way, an irrigation system, mainly through sensors and an Arduino Mega2560 R3 motherboard, among other modules; starting from its design, through its development and up to its implementation.

The verification of the data collection will be shown on the screen, both in the one physically established in the model, and via WiFi by a server on the local IP of the same. For this, it is necessary to be aware of the current date and hours that are set by a "Real Time Clock"; at the same time, it serves to define the optimal irrigation time bands. In addition, the use of LEDs determines in each case the state of the variable that is being checked if we are physically in the irrigation.

The 3.3 or 5 V variable power supply allows maintaining a constant voltage without fluctuations in it. In the same way, two AC / DC transformers support the fact that the modules always remain on, one activating the previous source and the other the Arduino board.

Keywords:

Arduino, Integrated Development Environment (IDE), Real Time Clock (RTC), power supply, AC/DC transformers, programming language, System on a Chip (SoC).

Índice:

Índice de Ilustraciones:

Índice de Figuras:

Índice de Tablas:

1. Introducción y Objetivos:

1.1. Justificación del proyecto:

La idea del presente proyecto surgió a raíz de un trabajo acerca del Canal de Castilla de la asignatura: 'Ingeniería, Tecnología y Sociedad' del módulo correspondiente a la parte 'Historia de la Ingeniería' y, además, de la necesidad de establecer un mejor control del riego, sobre todo en verano, de la parcela del jardín de mi casa. Al mismo tiempo, al haber tenido siempre un cierto afán por la profesión de bombero, este proyecto muestra un acercamiento a ese tema, en cuanto a control y manipulación de agua se refiere.

Un sistema de riego convencional se compone por un circuito de tuberías formado por un conjunto de electroválvulas separadas a cierta distancia, lo que su rango de alcance sea, para abastecer todo el terreno. Además, habitualmente, tiene un programador que establece la hora que se quiere activar el riego y durante cuánto tiempo. El control de las electroválvulas suele hacerse con un par de cables por cada una, lo que supone, en ocasiones, un gran gasto de material.

Este proyecto supone una innovación en cuanto ahorro en comparación con un sistema convencional de riego. Para ello, el principal objetivo es establecer un sistema de comunicación que gestione la eficiencia del sistema mediante sensores de forma automática y, a ser posible, recopile sus datos para poder optimizar su uso.

1.2. Objetivos:

El presente plan pretendió, desde un primer momento, abarcar el control de un sistema de riego inteligente para alcanzar los siguientes objetivos:

1. Realizar un análisis del entorno para poder determinar el funcionamiento óptimo.
2. Implementar un sistema multidireccional de comunicaciones.
3. Establecerse como sistema de riego inteligente inalámbrico.
4. Monitorizar los datos de la humedad y temperatura del ambiente y el terreno mediante sensores.
5. Controlar a partir de los datos de forma automática y, también, de forma manual las electroválvulas (si fuera posible).
6. Permitir la mayor eficiencia de crecimiento del terreno.
7. Generar, en forma de maqueta, tanto el Software y el Hardware a utilizar.

1.3. Definiciones y Conceptos introductorios:

1. ***Broadcast*** = como su propio nombre indica, la traducción literal es *'Difusión Amplia'*; es decir, en comunicaciones es la forma de transmitir información, pero en vez de nodo a nodo, lo hace enviando, de forma simultánea, desde el nodo emisor a varios nodos receptores.

2. ***SPIFFS*** *(Serial Peripheral Interface Flash Fail System)* = son sistemas de archivos, que sirven contenido exclusivamente estático, diseñados para funcionar en memorias flash conectadas por SPI en dispositivos embebidos y con escasa cantidad de RAM, por eso no soporta directorios.

3. ***SPI*** *(Serial Peripheral Interface)* = por definición, SPI corresponden a las siglas de Interfaz de Periféricos en Serie, es un protocolo síncrono de datos en serie utilizado, por microcontroladores, para comunicarse, en distancias cortas, con uno o más dispositivos periféricos o entre ellos mismos, habiendo siempre un maestro. En términos generales, hay cuatro modos de transmisión que controlan: la fase (si los datos entran y salen en el borde ascendente o descendente de la señal del reloj de datos) y la polaridad del reloj (si el reloj está inactivo cuando está alto o bajo).

4. ***Sistema Embebido*** *o* ***Empotrado*** = es un sistema electrónico, de bajo costo y consumo de potencia, diseñado para cubrir necesidades específicas realizando, según sea el caso, pocas funciones en tiempo real; al contrario de lo que ocurre con las computadoras, que están diseñadas para cubrir un amplio rango de necesidades, las cuales tienen un propósito general. Suelen tener un microcontrolador, que es un microprocesador con interfaces de entradas/salidas, y un software que se ejecuta sobre él siendo necesaria el utilizar una cantidad de memoria para el sistema. Estos sistemas tienden a Sistemas inteligentes cuando se les conectan con otro dispositivo (comunicación máquina-máquina) o, en especial, a Internet.

5. ***SoC*** = Sistema en un Chip.

6. ***Websocket*** = es una tecnología diseñada, especialmente, para ser implementada en navegadores y servidores web que proporciona un canal de comunicación bidireccional capaz de recibir y transmitir simultáneamente sobre un único socket TCP.

7. ***TCP*** = Protocolo de Control de Transmisión. Es uno de los protocolos más importantes de Internet, encargado de generar las conexiones para la transmisión de datos de forma fiable y bidireccional, ya que monitorea el flujo de datos garantizando, sin errores, que la información llegue en orden tal cual fue enviada, empleando el Puerto para ello. Evita la saturación de la red.

8. ***API*** = el Interfaz de Programación de Aplicaciones es un conjunto de definiciones, funciones, subrutinas, procedimientos y protocolos que ofrece cierta biblioteca para desarrollar e integrar el software de las aplicaciones y poder ser utilizado por otro software como una capa de abstracción; es decir, permitir el acceso al *'backend'* de aplicaciones de terceros con el fin de reutilizar servicios ya creados; en otras palabras, permiten que los productos y servicios se comuniquen con otros sin necesidad de saber cómo están implementados.

9. ***API REST*** = haciendo referencia a las siglas REST en español, Transferencia de Estado Representacional, corresponden a un conjunto de restricciones con las que se pueden crear un estilo de arquitectura software, en donde las acciones se trasmiten a través de peticiones HTTP y el intercambio de datos se realiza en formato JSON. Se ha convertido en un estándar en la comunicación Web para crear servicios en empresas.

10. ***CMD*** = denominado *Símbolo del Sistema* o *Command Prompt* es el intérprete de comandos en OS/2 (sistema operativo de IBM) y sistemas basados en Windows Nueva Tecnología. Vulgarmente, es la ventana negra, la consola, donde se introducen los comandos del sistema en Windows.

11. ***Vanilla JavaScript*** = *Javascript* se refiere a un lenguaje de programación *'just-in-time'* con funciones de primera clase (son las tratadas como cualquier otra variable). No confundir con el lenguaje de programación Java. El modelo *Vanilla* se refiere a que utiliza el lenguaje sin librerías; es decir, hecho directamente, programado "a pelo".

12. ***JSON*** = es el acrónimo inglés formado por Notación de Objeto de JavaScript, que consiste en un formato de texto plano (carece de información de formato y forma o tipo de letra), independiente del lenguaje por ser subconjunto de la notación literal de objetos de JavaScript, para el intercambio o almacenaje de datos estructurados, siendo una alternativa a XML.

13. ***I2C*** = es un bus de comunicaciones en serie síncrono, denominado Circuitos Inter Integrados, que posee una arquitectura maestro-esclavo (pudiendo ser y transmitir únicamente un maestro cada vez), en donde la comunicación entre sus dispositivos electrónicos se realiza de forma interna en sus artículos, verificándose si la señal ha llegado. Su funcionamiento requiere únicamente de 2 cables: uno para la señal de reloj (CLK) y otro para enviar los datos (SDA). Ahora mismo se considera un estándar.

14. ***URI*** = es un protocolo de Internet, llamado Identificador de Recursos Uniforme, que como su propio nombre indica identifica de forma unívoca los recursos de la red mediante una cadena de caracteres; en otras palabras, es

el autentificador único que accede a un recurso por internet, ya sea físico o abstracto, para interactuar él o consultar información sobre el mismo. A través de la estructura URI se determina a qué, dónde y cómo acceder para la búsqueda de información. En este se engloban las subcategorías ***URL*** y ***URN***.

15. ***URL*** = es una de las dos clases que conforma el ***URI*** y se nombra por Localizador de Recursos Uniforme encargado de indicar dónde se encuentran los recursos y, por ello, también acceder alguna página web.

16. ***URN*** = es la otra clase que conforma el ***URI***, se denomina Nombre de Recurso Uniforme y desempeña la función de determinar un dominio web de forma permanente; es decir, establece un recurso de manera indefinida independientemente de la localización.

17. ***Comandos AT*** = instrucciones que comunican, de forma codificada, al hombre y a un terminal Módem formando un lenguaje entre ellos.

18. ***AP*** = de las siglas en inglés, *'Access Point'*, traducido por Punto de acceso que se refiere a Servidor o Host.

19. ***SSID*** = *'Service Set Identifier'* es un identificador de una red inalámbrica que se encuentra en todos los paquetes que se envían por ella siguiendo una secuencia de 0 a 32 octetos, en la mayoría de los casos, alfanuméricos.

20. ***RSSI*** = *'Received Signal Strength Indicator'* es un Indicador de fuerza de la señal recibida que mide el nivel de potencia de esta en redes WiFi o de telefonía móvil, mayormente, sirviendo de escala de referencia, siendo el 0 el centro, en correspondencia a 1mW.

21. ***WEP, WPA, WPA2*** = diferentes tipos de cifrados para conexiones inalámbricas.

22. ***WDT*** = *'WatchDog Timer'* siglas en inglés correspondientes a Temporizadores de Vigilancia.

23. ***IoT*** = *'Internet de las cosas'* es un concepto achacado a lo que se podría llamar como *la Industria 4.0*, o *cuarta revolución industrial*, al establecer una interconexión entre cualquier tipo de objeto e Internet, sin necesidad de ser manipulado por un humano, para intercambiar información y datos con otros dispositivos. Una de las funciones más importantes es la de poder controlar dichos elementos a través de Internet sin necesidad de estar físicamente en ese lugar.

1.4. Estructura general de la Memoria:

En este apartado se añade una descripción introductoria de las partes en las que se divide el proyecto detallando el contenido de cada una brevemente.

CAPÍTULO 1. *Resumen y palabras clave*. Incluye una breve descripción de la idea principal que se quería llevar a cabo, añadiendo también el Abstract.

CAPÍTULO 2. *Introducción*. Se describe la idea de la que se parte en este proyecto, su justificación. Además, se realiza una breve noción de los objetivos a conseguir, la introducción de ciertas definiciones y conceptos antes de comenzar con el proyecto y se detalla la estructura general de la Memoria.

CAPÍTULO 3. *Estado de la técnica*. El punto abarca la información previa a comenzar con el proyecto y el porqué de su enfoque y realización.

CAPÍTULO 4. *Riego*. Explica en qué consiste y los diferentes tipos que existen.

CAPÍTULO 5. *Análisis del entorno*. Se evalúa la climatología de la región y la zona del terreno.

CAPÍTULO 6. *Arduino*. Introducimos qué es, el porqué de su uso y cómo se utiliza.

CAPÍTULO 7. *Análisis, Diseño y Configuración de los elementos*. Es el desarrollo de la maqueta en sí: Forma de proceder, componentes utilizados, comandos empleados, descripción del sistema, modificaciones realizadas e implementación.

CAPÍTULO 8. *Estudio económico*. Se evaluó el mercado actual, el presupuesto del que se partía y la viabilidad económica.

CAPÍTULO 9. *Pruebas y resultados*. Se tratarán los problemas que nos han surgido a lo largo de su realización y las distintas formas que se han intentado implementar para su mejor funcionamiento. Además, del resultado final que se ha conseguido. Por último, se abordan unas futuras mejoras que se querrían implementar.

CAPÍTULO 10. *Conclusiones*. A lo largo de la realización se ha llevado a cabo un aprendizaje con aspectos, tanto positivos como negativos, que han de ser tratados.

CAPÍTULO 11. *Referencias y Bibliografía*. Indicación de donde se ha obtenido la información y las fuentes empleadas, tanto para desarrollar de forma escrita el trabajo como para aprender las nociones necesarias del mismo.

2. Estado de la técnica:

Debido, en gran parte, a la mejora de las tecnologías en actividades agrícolas, se ha mejorado enormemente su eficiencia y productividad, aumentando hasta tres veces más, desde 1969 hasta hoy en día, el aprovechamiento y utilización del agua, la tierra y demás recursos naturales. Al mismo tiempo, se han visto sometidas la agricultura y la alimentación a un proceso de globalización e industrialización. Por esta razón, según estimó la FAO (Organización de las Naciones Unidas para la Alimentación y la Agricultura) se ha de producir hasta un 50% más que la demanda actual para cubrir las necesidades de los 9700 millones de personas que se estiman que habrá en 2050 en el mundo. Este objetivo ha de conseguirse aportando soluciones a los desafíos, publicados en 2017, dentro del marco que establece la FAO en "El Futuro de la alimentación y la agricultura. Tendencias y desafíos".

"*Actualmente la superficie de regadío en el mundo es de 325,1 millones de hectáreas, representando el 20% de la superficie total de tierra cultivada y suponiendo el 40% de los alimentos producidos en todo el mundo. Asia, con 232,7 millones de hectáreas, es el continente que tiene mayor superficie dedicada al riego (representa más del 70% de la superficie regada mundial), seguido de América con 52,2 millones de hectáreas, Europa con 21,4 millones de hectáreas, África con 15,6 millones de hectáreas y en último lugar Oceanía con 3,2 millones de hectáreas. Se estima que el 70% del agua es utilizada por la agricultura de regadío a nivel mundial, utilizándose el riego por gravedad como método de riego en el 94% de la superficie regada y el riego por aspersión o goteo en el 6% restante de superficie, siendo la eficiencia del riego media a nivel mundial del 56% aproximadamente. En España la superficie de regadío es de 3,4 millones de hectáreas, de las cuales el 70% de la superficie se riega por aspersión o goteo y el 30% restante por gravedad, dedicándose al riego el 79% del agua.*" [1]

En los años venideros se espera un cambio global en la forma de regar en agricultura, en la implementación y aumento de nuevas infraestructuras y superficies acompañadas de nuevas instalaciones para ello con un cambio de cultivos, consiguiendo así mayor eficiencia. Para llevar a cabo esa transformación, se han de tener en cuenta cómo se afrontarán los grandes impactos en el medio ambiente que ya tiene de por sí, en la actualidad, el riego en la agricultura (contaminación de las aguas, sobreexplotación, salinización, …) como el cambio climático venidero que se está viendo acentuado, provocando sequía; lo cual el ahorro del agua es un tema de vital importancia para su futura disponibilidad. A parte, las instalaciones son controladas con energía, siendo su ahorro y optimización un factor principal

en la rentabilidad y sostenibilidad, ya que al cambiar del sistema gravitatorio a emplazamientos que funcionan con presión aumentan el consumo energético.

La solución ante la necesidad de eficiencia, tanto energética como hídrica, es el perfeccionamiento del riego mediante TICs, Tecnologías de la Información y la Comunicación, creándose así el llamado 'Riego Inteligente' que consiste en generar más con menos mediante un aprovechamiento óptimo de todo tipo de recursos, incrementando así la producción con un uso más eficiente de estos medios. La iniciativa del 'Riego Inteligente' se centra en la adquisición, procesamiento y monitorización de datos, además de representar dicha información, consiguiéndose así una eficacia hídrica a la hora de programar el riego, teniendo en cuenta los tiempos, el entorno y demás factores influyentes que determinan la cantidad de agua que necesita la planta en cada momento. La programación del riego se debe determinar tanto por su control de funcionamiento como de la distribución, por el terreno, de la humedad; además de la presión, el terreno, el agua, etc. Aunque la verdadera variable que permite una grata gestión y un ahorro adecuado en los costes operativos en el manejo del riego es el tiempo, al automatizar su control.

En cuanto a un aumento de eficiencia energética se refiere, si se implementan las TICs en designios de riego con energía solar fotovoltaica ('Riego Solar') se disminuyen sustancialmente los costes, al igual que si se emplea otro tipo de energías renovables, pero siendo ésta la más beneficiosa en su utilización. El 'Riego Solar Inteligente' busca una sincronización entre las necesidades del riego y la disponibilidad energética al adaptar la irradiación aprovechable con los distintos sectores de riego, estableciéndose así una administración plena del agua y la energía.

Este tipo de soluciones va a ir, exponencialmente, en aumento para conseguir una reducción de los siguientes indicadores de los productos agrícolas: Huella Hídrica (necesidad de agua para producir una unidad de producto) y Huella de Carbono (para producir una unidad de producto, cuánta cantidad de gases de efecto invernadero han de ser emitidos).

Según el ingeniero y director de proyectos relacionados con la agricultura Manuel Martín Arroyo, concluye que:

"El futuro de la agricultura de regadío a nivel mundial depende, en buena parte, de la implantación de sistemas de riego inteligente en las fincas de cultivo, que permitan la utilización más eficiente de los recursos agua, fertilizante y energía de manera que se aumenten los niveles de producción utilizando menos recursos productivos. Con el riego inteligente se incrementa la rentabilidad de las explotaciones y se minimiza el impacto ambiental de esta actividad al disminuir tanto el uso del agua como la aportación de

elementos contaminantes al entorno. La implantación de sistemas de riego inteligente es fundamental para garantizar la sostenibilidad de la agricultura de regadío." [2]

Por este motivo, podemos observar la importancia de la técnica en el método de riego, ya que es lo que permite mayor validez y control de nuestros medios, que se traduce en eficiencia, sostenibilidad y optimización de los recursos.

He de matizar que esta necesidad de ahorrar agua, al ser un bien esencial y de primera necesidad, se está viendo incrementada sustancialmente por la crisis del COVID-19, al ser muy importante las condiciones de salubridad y desinfección, como lavarse las manos para la prevención y propagación de infecciones. La higiene ha de garantizarse con vital importancia, básicamente, llevada a cabo con el agua que es nuestra materia prima más imprescindible; por eso la salubridad de suministros y saneamiento de aguas residuales han sido declarados por el Gobierno de España como servicio esencial. Por otro lado, a parte de la operadora de agua, la responsabilidad es de todos y cada uno de nosotros.

Toda acción pertinente ha de tomarse conociendo cómo influirá al respecto y cuantos más datos se tengan, en relación con su posible resultado, mejor. Eso es lo que pretende la Industria 4.0 con la recopilación de información a través del 'Big Data', buscar el punto óptimo de mayor eficiencia para incrementar el rendimiento o si incluso es necesario un cambio de sistema para mejorar el desarrollo productivo evitando ineficiencias, particularmente humanas, en vez de buscar que no haya errores en el proceso actual. El intercambio y la compartición de información entre las distintas partes influyentes son primordiales para determinar el resultado más satisfactorio. La Inteligencia Artificial toma un papel principal en la toma de decisiones de forma autónoma y en consecuencia para mejorar el beneficio de sus recursos.

"La Industria 4.0 no está orientada a mejorar tu proceso actual, sino que pretende remplazarlo o evolucionarlo a otro más eficiente." [3]

3. Riego:

3.1. ¿Qué es un riego?

Sistema mediante el cual se aporta agua a los cultivos, utilizado tanto en agricultura como en jardinería, para cumplir sus necesidades hídricas que no se satisficieron mediante la precipitación, considerándose, por ello, un elemento de vital importancia, porque, además, el agua es el encargado de regular la temperatura, transportar las sustancias e influir en la nutrición. En resumidas cuentas, el riego es esencial para un correcto mantenimiento de los cultivos y, por ello, ha de ser controlado eficazmente.

La elección de una técnica de riego u otra depende de diversos factores, siendo los más influyentes, en nuestro caso, la forma del terreno y la topografía, acompañada de la climatología, las plantas a labrar, la calidad del agua y, en menor medida, su disponibilidad y su forma de llevarla a nuestra zona de cultivo entre otros. Una buena elección ha de ser la que permita un correcto ahorro sustancial de nuestra materia prima líquida, el agua, ya que cada tipo de planta consume cierta cantidad en su justa medida y un exceso de la misma podría dificultar el mantenimiento y conservación de nuestros vegetales.

3.2. Tipos de Riego:

En primera instancia, existen diversos tipos de riego, distinguiéndose dos grupos generales, atendiendo a la cronología histórica: ***sistemas tradicionales*** y los ***actuales***:

Los primeros son estructurados mediante la construcción de canales encargados de transportar el agua, pero, debido al excesivo consumo de agua que requieren, se encuentran en desuso por desaprovechar los recursos. Además, era necesario situar las zonas agrícolas cerca de las grandes superficies de agua para que su transporte fuera más eficiente. Existen de diversos tipos, siendo los más habituales los siguientes que detallaremos más adelante:

- Por canales o infiltración.
- Por inundación o sumersión.
- Por surcos o arroyamiento
- Por drenaje.

Todos ellos, como se va a apreciar posteriormente, se engloban dentro de *riegos superficiales*.

El segundo conjunto, al no requerir de construcciones, poseen un *'Sistema de control'* (programadores) que controlan el paso del agua. En los sistemas actuales se distinguen varios subgrupos, que se podrían englobar en 3:

- **Aspersión o Difusión:** modalidad de riego sectorial que esparce gotas muy finas de agua por el aire como si fuera lluvia.
- **Goteo o Localizado:** Sistema de riego mediante el cual el agua cae gota a gota junto al tallo de cada planta; por ello, reducen el consumo de agua al localizarse exactamente en la zona a regar.
- **Hidroponía:** método que sustituye el sustrato del suelo agrícola por disoluciones minerales para cultivar las plantas.

Para todos ellos se pueden especificar dos géneros de controladores, que serán detallados más adelante: los *electromecánicos* y los *inteligentes*.

En cuanto a Ingeniería Electrónica y Automática, es la parte que más nos concierte y en la que verdaderamente se centra nuestro proyecto en sí, al consistir, en mayor medida, en la construcción e implementación de éste.

3.2.1. Riego por superficie:

El agua se aporta para infiltrase en el interior del suelo tras cubrirlo en su totalidad o circular sobre él; por esa razón tiene los graves inconvenientes de las grandes pérdidas ocasionadas, por el contundente consumo de agua y la erosión provocada. En contraposición, esas mismas cualidades se pueden empeñar con función estética, favoreciendo un carácter ambiental y sensorial aportando humedad, frescor y belleza aprovechando la movilidad del agua, pero esto sería para exposición de jardinería y no es el aspecto que nos concierne.

Esta metodología de regar sería adecuada para cultivos arbóreos o terrenos con desnivel que permiten la distribución del agua por todos ellos.

3.2.1.1. Tipos de riegos por superficie:

Dependiendo de la clasificación utilizada se pueden distinguir *según se tenga repartido el agua sobre la superficie*:

- Riego por **infiltración**:
  - Por canales: empleados para transportar el agua hasta las distintas zonas a regar, creando arroyos artificiales.
- Riego por **inundación:** se aplica sobre el terreno, quedándose acumulado en este, hasta que se va, posteriormente, infiltrando.
  - Por alcorques y pozas: áreas de pequeño tamaño y pocas profundidades excavadas alrededor del elemento a regar en

cuestión; es decir, un cubrimiento parcial del terreno. Habitualmente se emplean en arbustos, agrupados o de gran tamaño, y árboles.
    - De compartimentos cerrados: grandes extensiones de terreno donde se empapa todo el suelo.
- Riego por **escurrimiento** o **vertido**: aprovechando el desnivel del terreno, se vierte el agua en la parte superior a regar consiguiendo que se vaya infiltrando a medida que discurre por el mismo.
    - Por surcos: el agua se infiltra, tanto vertical como horizontalmente, mientras se distribuye por el terreno, siendo éstos de distinta forma (U o V), anchura, longitud y pendiente. Predomina su uso en plantas que son sensibles al encharcamiento, pero que han de ser resistentes a la salinidad.
- Riego por **drenaje**: evita la acumulación, en la parte de las raíces, del exceso de agua causado por la saturación del suelo. Su clasificación atiende a tres criterios:
    - *Manera de conducir el agua*:
        - Sistema abierto o superficiales
        - Subterráneos
    - La *disposición*, en el terreno, *de la red*:
        - Disposición de los drenes respecto del colector:
            - En parrilla
            - En peine
            - En espina de pez
        - Posición de las tuberías respecto a las curvas de nivel:
            - Drenes longitudinales
            - Drenes transversales
            - Drenes oblicuos
            - En zig-zag
    - Empleo del *sistema que facilita la entrada del agua*:
        - Drenaje con tuberías, grava y arena
        - Sistema con tuberías y rendijas de drenaje
        - Sistema con tuberías, rendijas de drenaje y tapiz de arena

3.2.2. Riego localizado:

Basado en el aporte continuo del agua sobre la superficie o bajo ésta, si es subterráneo, de manera que únicamente se moja una zona concreta, un volumen que posee el nombre de bulbo húmedo, que depende de la textura del terreno e influye en el transporte de sales hasta su periferia. De esta

forma, es la mejor opción si nuestra agua resulta ser salina; y en caso de ser agua depurada, la opción subterránea es aconsejable. En este tipo de riego es muy importante tener en cuenta la separación entre plantas. Se caracterizan por una alta frecuencia en los riegos con cantidades reducidas de agua, para así mantener una humedad prácticamente constante y servir como soporte, no como almacén. En caso de tratarse de la alimentación de árboles, el riego ha de ser menos frecuente, pero sí más contundente para evitar problemas en las raíces y que éstas consigan una mayor profundidad de agarre.

Poseen la gran ventaja de no tener pérdidas por evaporación, pudiéndose controlar el uso del agua, siendo de esta forma muy eficiente; por esta razón, permite una correcta automatización. Por el contrario, tiene la gran pega que requiere una gran inversión, además de control y mantenimiento.

3.2.3. Riego por aspersión o difusión:

Sistema de cobertura total, porque dispersa gotas de agua por toda la superficie como si fuera lluvia, donde se produce un proceso de infiltración que depende del terreno. Las condiciones externas influyen enormemente en esta forma de riego, como el viento, y reducen su eficiencia, como la evaporación, en gran medida si las gotas son muy finas. En general se emplean para riegos de césped o similares, ya que, al mojar la parte foliar de la planta, si el agua posee cierto tipo y cantidad de sales, podrían causar daños en las hojas y flores.

El control del riego está limitado únicamente a las condiciones climatológicas, siendo la uniformidad del riego independiente de las características del suelo. Representativo por ser riegos ligeros, pero frecuentes, siendo útil en las primeras fases del desarrollo de las plantas, como para servir de apoyo o de supervivencia. Permite a su vez la automatización. En contraposición, esta manera de regar incrementa el riesgo de la aparición de enfermedades superficiales.

3.2.4. Riego por Microaspersión y Nebulización:

Es una variante del método por aspersión, pero más parecida a la de localización por la forma de aplicar el agua; sería la técnica donde confluyen las ventajas de las dos anteriores, siendo la Microaspersión más propia de lugares difíciles de implementar la técnica por goteo y zonas muy arenosas consiguiendo una uniformidad prácticamente constante; y la Nebulización de invernaderos para crear ambientes húmedos y decrementar su temperatura.

3.3. Tipos de Emisores:

Los **emisores** son los dispositivos, dentro de la instalación hidráulica, encargados de distribuir el agua sobre la superficie deseada a regar. Se podrían considerar que son los elementos últimos que influyen voluntariamente sobre el control de nuestra agua a regar, y por eso han de cumplir unos requisitos mínimos de calidad: caudal uniforme (mayormente insensible a variaciones de presión), poco sensible a obturaciones y resistente a las condiciones de trabajo; además, ha de respetar una buena relación calidad/precio, sea fácil su instalación y, en mayor medida, una alta uniformidad en su fabricación.

Para determinar qué tipo de emisor emplear, de forma genérica, el fabricante ha de facilitarnos las principales características de uso, que son:

- Presión nominal: presión a la cual debería funcionar el emisor y para la cual se ha diseñado.
- Caudal nominal: caudal proporcionado al trabajar con la presión nominal.
- Coeficiente de variación de fabricación: la variabilidad que puede llegar a producirse en su fabricación, sirviendo como dato indicativo.

En caso de tratarse de riegos por aspersión, se deberían incluir las características de las boquillas acoplables indicando los datos de presión, caudal y alcance. Por el contrario, si de riegos por localización se tratase, se debe de añadir las propiedades referidas a diámetro mínimo de paso, régimen hidráulico, exponente de descarga y, en caso de ser autocompensantes, el rango de presiones.

3.3.1. Emisores de Riego por Aspersión:

3.3.1.1. Aspersores:

Distribuyen el agua, con movimiento rotatorio, en forma de gotas como si fuera lluvia gracias a la presión ejercida. Hay diversas clasificaciones según el tipo de función que se desempeñe:

- *Según la presión de trabajo*:
  - De baja presión (hasta 1,5 kg/cm^2)
  - De media presión (desde 1,5 kg/cm^2 hasta 4,5 kg/cm^2)
- *Según el mecanismo de giro*:
  - Aspersores de impacto
  - Aspersores de turbina
- *Según la geometría de área a empapar*:
  - Aspersores circulares

- Aspersores sectoriales

- *Según la colocación en la zona de riego*:
  - Aspersores emergentes
  - Aspersores aéreos

3.3.1.2. Difusores:

Son lo mismo que los aspersores, pero carecen de parte móvil; es decir, no tienen elementos móviles y por eso requieren de menor presión, por lo que su alcance es menor, aunque vierten mayor cantidad de agua. Su clasificación típica es:

- *Según la forma de colocarlos en el terreno*:
  - Aspersores emergentes
  - Aspersores aéreos

3.3.2. Emisores de Riego por Localización:

3.3.2.1. Goteros:

Son los más utilizados, disipan la presión y trabajan en torno a 1 kg/cm^2 con bajo caudal, expulsando hasta unos 16 litros/hora. Las clasificaciones dependen de:

- *Donde se encuentran localizados en las tuberías*:
  - Integrados
  - Pinchados
  - Insertados o Interlínea
- *Variaciones en el caudal*:
  - No compensantes
  - Autocompensantes

3.3.2.2. Tuberías emisoras:

Tuberías, habitualmente de polietileno, que aplican el agua, generando una banda continua de humedad, por unos orificios o una pared porosa. Las más utilizadas son:

- Goteadoras: el agua sale gota a gota.
- Perforadas: dependiendo de la presión, la manera en la que sale el agua es por gotas o formando estrechos chorros.
- Porosas o Exudantes: banda de humedad continua a través de los poros.

3.3.2.3. Borboteadores o Inundadores:

Dispositivos que aplican en zonas pequeñas el agua sin llegar a mojar la zona aérea de la planta, por eso entran en este grupo al ser un riego por

inundación al emplear gotas más gruesas. Usados principalmente en el riego de alcorques.

3.4. Tipos de Controladores:

Atendiendo a distintos **parámetros**, se pueden clasificar en diferentes tipos:

- *Según su alimentación eléctrica:*
  - Eléctricos o Electrónicos: 220/230 V de alimentación → Electroválvulas 24 V.
  - Pilas o Autónomos: Alimentación mediante pilas → Electroválvulas tipo latch 9 V.
  - Mediante energía solar.
- *Según la localización del controlador:*
  - Interno: dentro de la arqueta, sumergido.
  - Externo: en el mural, a la intemperie.
- *Según su tecnología:*
  - Analógico
  - Digital
  - Híbrido
- *Según dónde se encuentren las electroválvulas*:
  - Interna en el controlador (Programadores de Grifo)
  - Externas al controlador
- *Según el número de estaciones:* 1, 2, …, números pares hasta 48.
- *Según sus características técnicas:*
  - Programador volumétrico multicable o monocable.
  - Programadores modulares.
  - Programador digital por tiempo con o sin registro de caudal.
  - Programadores horarios con mandos mecánicos.
  - Programadores de batería con válvula incorporada.
  - Programadores de batería con salidas a solenoides “latch”.

Abarcando las distintas clasificaciones, y siendo en la parte que nos focalizaremos, se pueden diferir dos **géneros de controladores** dependiendo de si están *dotados de autonomía suficiente para llevar a cabo decisiones propias o no*:

- ***Electromecánicos*** o ***Electrónicos***: sólo son capaces de realizar la actividad para la cual han sido programados, no pueden adaptarse a inclemencias meteorológicas al no poseer sensores ni elementos similares de control.

- ***Inteligentes*** o ***IoT***: se adaptan a las condiciones del entorno, en gran medida, con la ayuda de los sensores que testean el terreno y ambiente. Poseen la etiqueta 'IoT' al tener conexión con internet y poder almacenar y consultar datos para mejorar su funcionamiento; es decir, son capaces de llevar a cabo el riego cuando mejor convenga, por ello son 'Inteligentes'.

En resumidas cuentas, nuestro dispositivo contemplaría todas las mejoras de los Electromecánicos y serviría de introducción para los Inteligentes, ya que establece unas franjas horarias determinadas para los posibles riegos, pero las determina de forma automática y decide en ellas, mediante los sensores, si es necesario el riego o no.

4. Análisis del terreno y de la climatología:

4.1. Terreno:

La extensión de terreno del jardín está conformada por unos 21m^2 (=7m*3m) en donde cabe destacar una franja conformada por rosales y el resto sería césped con un ciruelo rojo casi en medio. El sistema de riego ya se encuentra instalado, con aspersores en cada una de las cuatro esquinas correspondientes a la parte del césped.

He de destacar la clara diferencia de dos zonas dependiendo de la exposición a la luz respectivamente, donde hay un lado que recibe luz durante todo el día, prácticamente, y otra que se encuentra limitada por la sombra proyectada a lo largo de cierta cantidad de horas. De ahí el uso de 2 sensores de temperatura y 2 de humedad respectivamente. Se prescinde del control de luminosidad por las limitaciones materiales, ya que para determinar la hora se emplea un RTC.

La evaluación del regadío de cada parte sería la siguiente, detallando su frecuencia de regado en el próximo subapartado:

- Césped:

  Al tratarse de un césped ya germinado y cultivado de hace ciertos años, interesa que la profundidad del riego alcance unos 10 cm favoreciendo así el desarrollo radicular.

ILUSTRACIÓN 1: CÉSPED

- Rosales:

  Los rosales al plantarse al mismo tiempo, prácticamente, que la zona del césped; es decir, ya se han desarrollado por completo, interesa que el alcance del riego sea de unos 40 cm, a ser posible, mediante la técnica de goteo para no mojar ni flores ni hojas.

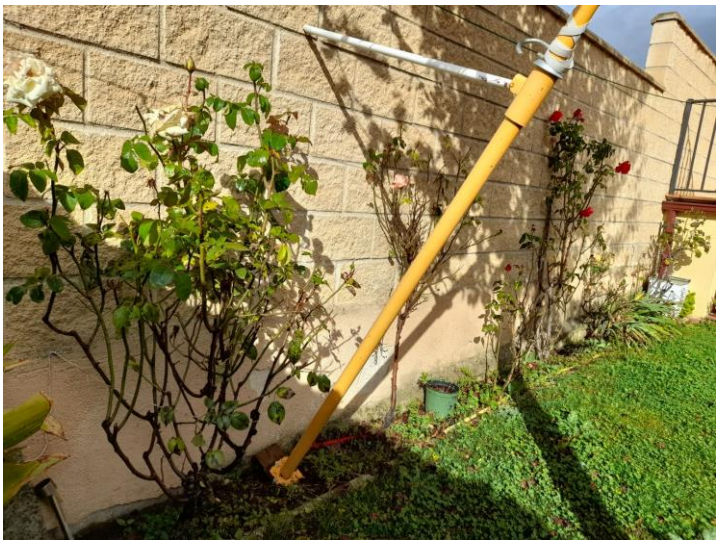


ILUSTRACIÓN 2: ROSALES

- Ciruelo rojo (*Prunus Cerasifera var. 'Pissardii'*):

  Este árbol se plantó a la vez que el césped, por lo que ya se encuentra desarrollado y sin necesidad de un riego por goteo. La profundidad de alcance de su raigambre es somera, debería estar en torno a los 15 cm, ya que se caracteriza por tener unas raíces superficiales.

ILUSTRACIÓN 3: CIRUELO ROJO

4.2. Climatología:

Haciendo un análisis de la zona geográfica, dependiendo de las distintas épocas del año, se ha concretado lo siguiente [mirar Anexo *C. Climatología AEMET*] para un clima Mediterráneo de tipo Continental y de Montaña:

A. Primavera 2019:

a) Temperaturas:

- AL INICIO (01 de marzo – 07 de marzo):
  - Máximas: 14ºC
  - Mínimas: 4ºC
  - Media: 9ºC aprox.
- A FINALES (24 de mayo – 30 de mayo):
  - Máximas: 22ºC
  - Mínimas: 10ºC
  - Media: 16ºC aprox.
- A MEDIADOS (12 de abril – 18 de abril):
  - ✓ Máximas: 15’5ºC
  - ✓ Mínimas: 4’5ºC
  - ✓ Media: 10ºC aprox.

b) Precipitaciones:

- INICIALES (8 de marzo):
  - Precipitación acumulada periodo estudiado: 5 mm
  - Mediana de la precipitación acumulada en el periodo 1981-2010: 2 mm
- FINALES (31 de mayo):
  - Precipitación acumulada periodo estudiado: 77 mm
  - Mediana de la precipitación acumulada en el periodo 1981-2010: 138 mm
- A MEDIADOS (19 de abril):
  - ✓ Precipitación acumulada periodo estudiado: 50 mm
  - ✓ Mediana de la precipitación acumulada en el periodo 1981-2010: 55 mm

B. Verano 2019:

c) Temperaturas:

- AL INICIO (01 de junio – 07 de junio):
  - Máximas: 24ºC
  - Mínimas: 11ºC
  - Media: 17’5ºC aprox.
- A FINALES (24 de agosto – 30 de agosto):
  - Máximas: 28ºC
  - Mínimas: 14ºC

- Media: 21ºC aprox.

- A MEDIADOS (13 de julio – 19 de julio):
  - ✓ Máximas: 30ºC
  - ✓ Mínimas: 15ºC
  - ✓ Media: 22'5ºC aprox.

d) Precipitaciones:

- INICIALES (8 de junio):
  - ➢ Precipitación acumulada periodo estudiado: 8 mm
  - ➢ Mediana de la precipitación acumulada en el periodo 1981-2010: 5 mm
- FINALES (31 de agosto):
  - Precipitación acumulada periodo estudiado: 62 mm
  - Mediana de la precipitación acumulada en el periodo 1981-2010: 65 mm
- A MEDIADOS (20 de julio):
  - ✓ Precipitación acumulada periodo estudiado: 24 mm
  - ✓ Mediana de la precipitación acumulada en el periodo 1981-2010: 51 mm

C. Otoño 2019:

e) Temperaturas:

- AL INICIO (01 de septiembre – 07 de septiembre):
  - ➢ Máximas: 22ºC
  - ➢ Mínimas: 13ºC
  - ➢ Media: 17'5ºC aprox.
- A FINALES (24 de noviembre – 30 de noviembre):
  - Máximas: 10ºC
  - Mínimas: 2ºC
  - Media: 6ºC aprox.
- A MEDIADOS (13 de octubre – 19 de octubre):
  - ✓ Máximas: 18ºC
  - ✓ Mínimas: 8ºC
  - ✓ Media: 13ºC aprox.

f) Precipitaciones:

- INICIALES (8 de septiembre):
  - ➢ Precipitación acumulada periodo estudiado: 0 mm
  - ➢ Mediana de la precipitación acumulada en el periodo 1981-2010: 2 mm
- FINALES (30 de noviembre):
  - Precipitación acumulada periodo estudiado: 150 mm
  - Mediana de la precipitación acumulada en el periodo 1981-2010: 127 mm

- A MEDIADOS (20 de octubre):
  - ✓ Precipitación acumulada periodo estudiado: 30 mm
  - ✓ Mediana de la precipitación acumulada en el periodo 1981-2010: 62 mm

D. Invierno 2019:

g) Temperaturas:

- AL INICIO (01 de diciembre – 07 de diciembre):
  - ➢ Máximas: 9ºC
  - ➢ Mínimas: 2ºC
  - ➢ Media: 5'5ºC aprox.
- A FINALES (23 de febrero – 29 de febrero):
  - ○ Máximas: 12ºC
  - ○ Mínimas: 2ºC
  - ○ Media: 7ºC aprox.
- A MEDIADOS (12 de enero – 18 de enero):
  - ✓ Máximas: 8ºC
  - ✓ Mínimas: 1ºC
  - ✓ Media: 4'5ºC aprox.

h) Precipitaciones:

- INICIALES (8 de diciembre):
  - ➢ Precipitación acumulada periodo estudiado: 15 mm
  - ➢ Mediana de la precipitación acumulada en el periodo 1981-2010: 12 mm
- FINALES (29 de febrero):
  - ○ Precipitación acumulada periodo estudiado: 70 mm
  - ○ Mediana de la precipitación acumulada en el periodo 1981-2010: 115 mm
- A MEDIADOS (19 de enero):
  - ✓ Precipitación acumulada periodo estudiado: 52 mm
  - ✓ Mediana de la precipitación acumulada en el periodo 1981-2010: 68 mm

E. Primavera 2020:

i) Temperaturas:

- AL INICIO (01 de marzo – 07 de marzo):
  - ➢ Máximas: 13'5ºC
  - ➢ Mínimas: 3'5ºC
  - ➢ Media: 8'5ºC aprox.
- A FINALES (18 de mayo - 24 de mayo):
  - ○ Máximas: 21ºC
  - ○ Mínimas: 8ºC

- Media: 14’5ºC aprox.

- A MEDIADOS (12 de abril – 18 de abril):
  - ✓ Máximas: 15’5ºC
  - ✓ Mínimas: 4’5ºC
  - ✓ Media: 10ºC aprox.

j) Precipitaciones:

- INICIALES (8 de marzo):
  - ➢ Precipitación acumulada periodo estudiado: 5 mm
  - ➢ Mediana de la precipitación acumulada en el periodo 1981-2010: 2 mm
- FINALES (24 de mayo):
  - Precipitación acumulada periodo estudiado: 138 mm
  - Mediana de la precipitación acumulada en el periodo 1981-2010: 125 mm
- A MEDIADOS (19 de abril):
  - ✓ Precipitación acumulada periodo estudiado: 70 mm
  - ✓ Mediana de la precipitación acumulada en el periodo 1981-2010: 55 mm

F. Verano 2020:

k) Temperaturas:

- AL INICIO (01 de junio – 07 de junio):
  - ➢ Máximas: 24ºC
  - ➢ Mínimas: 11ºC
  - ➢ Media: 17’5ºC aprox.
- A FINALES (24 de agosto – 30 de agosto):
  - Máximas: 28ºC
  - Mínimas: 14ºC
  - Media: 22ºC aprox.
- A MEDIADOS (13 de julio – 19 de julio):
  - ✓ Máximas: 30ºC
  - ✓ Mínimas: 15ºC
  - ✓ Media: 22’5ºC aprox.

l) Precipitaciones:

- INICIALES (8 de junio):
  - ➢ Precipitación acumulada periodo estudiado: 20 mm
  - ➢ Mediana de la precipitación acumulada en el periodo 1981-2010: 5 mm
- FINALES (31 de agosto):
  - Precipitación acumulada periodo estudiado: 100 mm
  - Mediana de la precipitación acumulada en el periodo 1981-2010: 65 mm

- A MEDIADOS (20 de julio):
  - ✓ Precipitación acumulada periodo estudiado: 73 mm
  - ✓ Mediana de la precipitación acumulada en el periodo 1981-2010: 51 mm

G. Otoño 2020:

m) Temperaturas:

- AL INICIO (01 de septiembre – 07 de septiembre):
  - ➢ Máximas: 28ºC
  - ➢ Mínimas: 13ºC
  - ➢ Media: 21ºC aprox.
- A FINALES (24 de noviembre – 30 de noviembre):
  - ○ Máximas: 12ºC
  - ○ Mínimas: 4ºC
  - ○ Media: 8ºC aprox.
- A MEDIADOS (13 de octubre – 19 de octubre):
  - ✓ Máximas: 17ºC
  - ✓ Mínimas: 6ºC
  - ✓ Media: 12ºC aprox.

n) Precipitaciones:

- INICIALES (8 de septiembre):
  - ➢ Precipitación acumulada periodo estudiado: 0 mm
  - ➢ Mediana de la precipitación acumulada en el periodo 1981-2010: 2 mm
- FINALES (30 de noviembre):
  - ○ Precipitación acumulada periodo estudiado: 135 mm
  - ○ Mediana de la precipitación acumulada en el periodo 1981-2010: 127 mm
- A MEDIADOS (20 de octubre):
  - ✓ Precipitación acumulada periodo estudiado: 85 mm
  - ✓ Mediana de la precipitación acumulada en el periodo 1981-2010: 62 mm

Todas estas medidas localizadas a una altitud de *1005 m*, una latitud de ***40º 56' 43'' N*** y una longitud de ***4º 7' 35'' O*** correspondientes a la localidad de *Segovia*.

Como la humedad del suelo, con precisión, es muy difícil de medir y requiere de un proceso muy elaborado para su determinación, se va a generar una aproximación a partir de unos sensores de humedad y de la humedad relativa de la zona para evaluar cuan cantidad de agua es necesaria en cada momento; es decir, cuanto tiempo ha de estar el riego activado para alcanzar un buen regado del terreno.

5. Arduino:

5.1. ¿Qué es Arduino?

Arduino es una plataforma electrónica de código abierto basada en hardware y software sencillo para implementar, tanto tus primeros proyectos como unos futuros más elaborados. Las placas Arduino pueden leer entradas de diferentes tipos (luz en un sensor, un dedo en un botón o un mensaje de Twitter) y convertirlo en una salida, activando un motor, encendiendo un LED, publicando algo en línea. Enviando un conjunto de instrucciones al microcontrolador puedes establecer qué hacer en el tablero. Para configurarlo, utiliza el lenguaje de programación Arduino (basado en *Wiring*) y el Software Arduino (IDE), basado en *Processing*.

Arduino, como ya he citado antes, ha sido el núcleo de miles de proyectos a lo largo de los años; desde objetos cotidianos hasta complejos instrumentos científicos. Una comunidad mundial de creadores (estudiantes, aficionados, artistas, programadores y profesionales) se ha reunido en torno a esta plataforma de código abierto, contribuyendo a una inmensa cantidad accesible de conocimiento que puede servir de gran ayuda tanto para principiantes como para expertos.

Esta compañía de desarrollo de hardware y software libre nació en el Instituto de Diseño de Interacción en *Ivrea* como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación. Tan pronto como llegó a una comunidad más amplia, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta desde placas simples de 8 bits hasta productos para aplicaciones usables de IoT, impresión 3D y entornos integrados. Todas las placas Arduino son completamente de código abierto, lo que permite a los usuarios construirlas de forma independiente y eventualmente adaptarlas a sus necesidades particulares. El software también es de código abierto y está creciendo gracias a las contribuciones de los usuarios de todo el mundo.

5.2. ¿Por qué Arduino?

La disponibilidad de usuario simple y accesible permite emplearlo en miles de aplicaciones y proyectos diferentes. El fácil uso de su software, al mismo tiempo que flexible, es idóneo tanto para usuarios principiantes como avanzados, pudiendo funcionar en *Mac*, *Windows* y *Linux*. En el ámbito

educativo abarca un sinfín de posibilidades al permitir construir diseños de bajo coste e inicializarse en la programación y la robótica, pudiendo, de esta forma, construir prototipos interactivos que desafíen las leyes de la física y los principios de la química, o para desarrollar proyectos como éste. Al contar con la comunidad de Arduino, existe una amplia congregación de miembros que atribuyen ofrecerse ayuda mutua compartiendo una gran cantidad de conocimiento.

Existen, a su vez, otras muchas plataformas de microcontroladores para el desarrollo computacional con similar funcionalidad, pero éstas no simplifican el proceso de trabajo, lo cual no les hace poseer ciertas ventajas:

- ✓ **Económico:** La versión preensamblada más barata del módulo Arduino tiene un coste inferior a 50$, pudiéndose incluso solicitar para ensamblarla a mano. Por esta, razón entre otras, las placas Arduino son relativamente económicas en comparación con otras plataformas de microcontroladores.
- ✓ **Multiplataforma:** La mayoría de los sistemas de microcontroladores están limitados a *Windows*; en cambio, el software Arduino (IDE) se ejecuta en los sistemas operativos *Windows*, *Macintosh OSX* y *Linux*.
- ✓ **Entorno de programación simple y claro:** está convenientemente centrado en el entorno de programación *Processing*, para que los profesores enseñen a sus alumnos a programar en ese entorno, familiarizándose con el funcionamiento del IDE de Arduino. El IDE de Arduino destaca por su polivalencia, al ser fácil de usar para principiantes, pero lo suficientemente flexible para el aprovechamiento de usuarios avanzados.
- ✓ **Software de código abierto y extensible:** al ser una herramienta de código abierto, tiene el provecho de ser un lenguaje expandible a través de bibliotecas C++, y pudiéndose agregar incluso código AVR-C directamente, ya que es en el que se basa. De esta forma las personas que quieran comprender los detalles técnicos pueden usar Arduino como lazo para aprender el lenguaje de programación AVR-C.
- ✓ **Hardware extensible y de código abierto:** los planos de las placas Arduino se publican bajo una licencia *Creative Commons*; es decir, se encuentran sujeto a un conjunto de instrumentos jurídicos, de carácter gratuito, que facilitan el intercambio de conocimiento, para poder así compartir creatividad e intelecto. De esta forma, es posible crear tu propia versión del módulo, ampliarlo y mejorarlo, sin restricciones permitiéndose así comprender su funcionamiento o, simplemente, para ahorrar dinero en el empleo de materiales.

5.3. ¿Cómo uso Arduino?

En primer lugar, has de conocer el entorno, la ventana de trabajo, en el cual vas a programar; siendo un ejemplo del Sketch de inicio al abrir un trabajo nuevo lo siguiente:

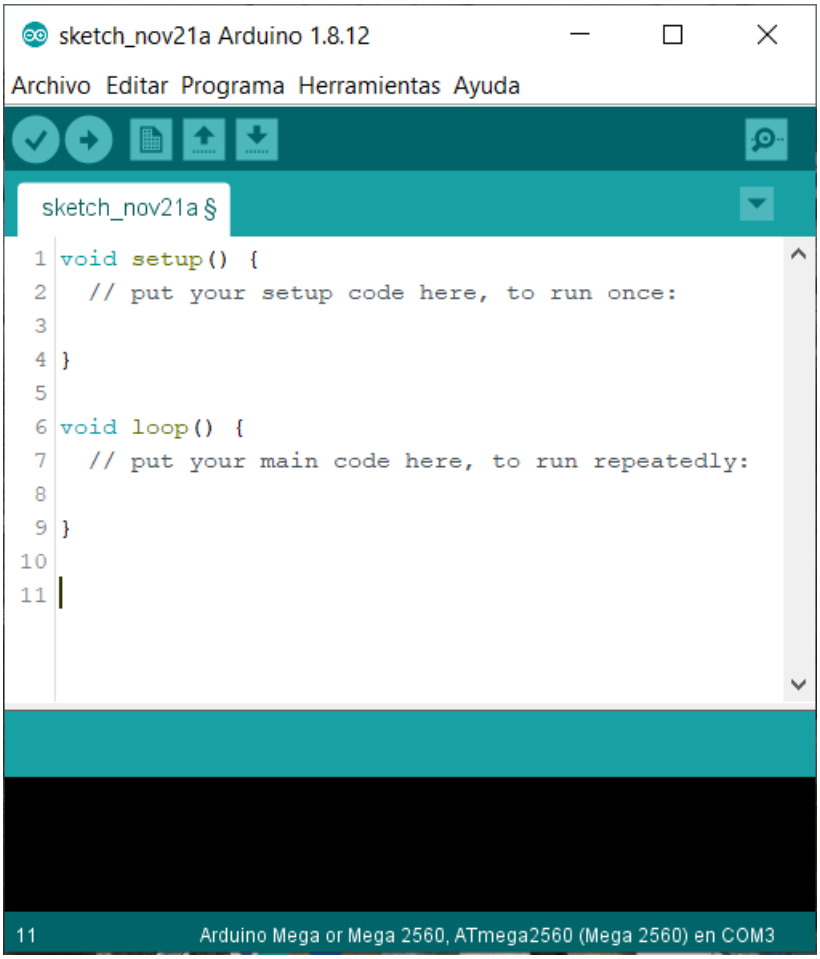


ILUSTRACIÓN 4: ENTORNO ARDUINO

Cada parte en un programa desempeña una función distinta, por esta razón ha de diferenciarse las siguientes partes:

Menú:

Lo conforman las pestañas de 'Archivo, 'Editar', 'Programa', 'Herramientas' y 'Ayuda', que como su propio nombre indican, realizan las distintas funcionalidades que abarca cada una.

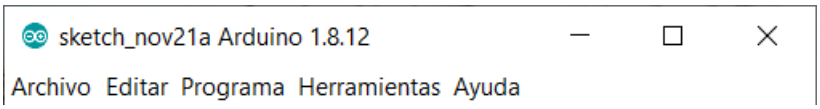


ILUSTRACIÓN 5: MENÚ ARDUINO

Botones de Acceso Rápido:

Estas acciones se pueden encontrar en las distintas pestañas del Menú anterior, pero al emplearse muy frecuentemente han establecido estos botones como atajo.

De izquierda a derecha nos encontramos con las funciones, con botón redondo, de 'Verificar', que comprueba que no haya errores de programación en el código y, justo después, la de 'Subir', que realiza la acción anterior antes de cargar el código en nuestra placa. En tercer lugar, empezando con los de forma cuadrada, está 'Nuevo', que genera un Sketch nuevo, es decir, una ventana nueva con un programa nuevo desde cero; luego, en el medio de los tres, está el de 'Abrir', que carga un programa ya existente guardado anteriormente en una nueva ventana; y, finalmente, el de 'Salvar', que guarda el programa sin realizar una compilación previa, ya que al dar 'Verificar' salva el programa antes de compilarlo. Por último, el botón del lado derecho, es el 'Monitor Serie', que es por donde se visualiza el envío y recepción de datos a través del puerto USB.

ILUSTRACIÓN 6: BOTONES ACCESO RÁPIDO ARDUINO

Editor de texto o Área de programación:

Esta zona es donde se encuentra desarrollado el código en sí. La pestaña que se adentra en la parte superior verde es el Nombre de la ventana actual que se está utilizando; si sale al final del nombre el símbolo '§' significa que el programa no está guardado. La flecha de la esquina superior derecha es la tabulación al menú referido a esta pestaña.

En el área blanca se escribe el programa, es donde se encuentra el 'Setup' y el 'Loop' que son las funciones principales de Arduino. La primera únicamente es realizada una vez, por eso es el lugar de configuración y declaración, mientras la segunda se encuentra realizándose en bucle, siendo así la que contiene el código que queramos que se ejecute en todo momento, las acciones pertinentes a efectuar. Previamente a ellos tienen lugar la declaración de variables globales, constantes, objetos y librerías necesarias para nuestro proyecto.

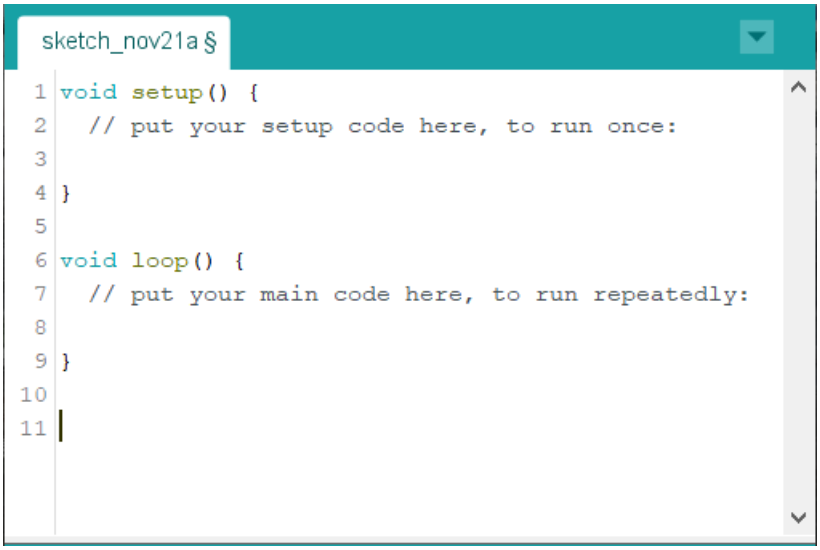


ILUSTRACIÓN 7: EDITOR DE TEXTO ARDUINO

Área de Mensajes y Consola:

Las zonas inferior y superior, las que se encuentran en color verde corresponden al área de mensajes. Arriba se manifiesta el mensaje con la acción que está teniendo lugar o justo ha tenido como un error, guardar o compilar, por ejemplo. Abajo en la esquina izquierda se muestra el número de la línea actual en donde se encuentra establecido el cursor en el editor; luego a su derecha se sitúa el tipo de placa a emplear y el número de puerto utilizado.

La parte en negro es la consola y muestra los errores o la cantidad de espacio ocupada en memoria por las variables, entre otros, tras subir o compilar el programa respectivamente.

ILUSTRACIÓN 8: ÁREA DE MENSAJES Y CONSOLA ARDUINO

6. Análisis, Diseño y Configuración de los Elementos:

Se pretende configurar la instalación del sistema de control con Arduino, dos fuentes de alimentación, una pantalla led LCD4x20 y unos sensores, entre otros módulos. Para ello, se ha realizado un pequeño prototipo, en forma de maqueta, para implementar el resultado.

Aunque el uso de una placa Arduino no es conveniente, por no decir impropio de las industrias (ya que para ello se usan PLCs), es cierto que sirve como base para la realización de pequeñas pruebas y de esta forma es como lo hemos enfocado debido, también, a su bajo impacto económico, facilidad para conseguir el material y la capacidad para soportar los elementos necesarios para el desarrollo de este proyecto. Según avancemos en el proyecto, se irá detallando si fuese posible abarcar, desde este trabajo de partida, un ambiente más industrial.

6.1. Forma de proceder:

Según diversos estudios se acredita que: el regar las plantas cuando más expuestas al calor están y a los rayos solares es contraproducente, ya que facilita la evapotranspiración y podría quemar la vegetación mediante el 'efecto lupa' entre otros problemas. Debido que, y según lo que estipulan órganos que se encuentran en materia, la franja horaria recomendada para una optimización del agua, en horario de verano, es entre las 6:00-9:00 y las 20:00-23:00 (en especial durante las de la mañana, ya que, como las personas, es cuando se pone en marcha su actividad metabólica); por ello, se usa el módulo de tiempo real, para determinar en qué momento del día nos encontramos, si fuera o dentro de dichas franjas. Además, se ha configurado la franja horaria para que se adapte según la época del año encontrada.

Por causas ajenas a mi voluntad, debido a la crisis sanitaria del COVID-19, han surgido limitaciones materiales de espacio, y por ello, se ha realizado una maqueta para enseñar de forma simulada el funcionamiento de este trabajo en vez de poder llevarlo a cabo en el jardín de mi casa. Toda la parte eléctrica y electrónica se ha conseguido implementar con éxito como si de su labor se tratase; en cambio, la parte hidráulica debido a la imposibilidad de conectar una manguera desde el baño hasta la habitación para mostrarlo, se han sustituido las electroválvulas por unos motores que indican, según su funcionamiento o no, si tiene lugar el paso del agua o se encuentra cerrado, vertiéndola ésta desde un recipiente cuando se nos pida y sea necesario.

6.2. Componentes y Elementos:

En la cabecera de este apartado se van a nombrar los componentes utilizados, en forma de epígrafe, siendo éstos los siguientes:

1) Tarjeta Arduino MEGA2560 R3
2) Fuentes de alimentación
3) Pantalla LCD 20x4
4) DHT11 – Sensor de Temperatura y Humedad ambientales
5) DS3231 – Reloj de Tiempo Real
6) DS18B20 – Sensor de Temperatura
7) Higrómetro KY70 – Sensor de Humedad
8) Sensor 'Nivel de agua' – Sensor para la lluvia
9) ESP8266-01S – Módulo WiFi
10) "Electroválvulas" – Motores para la simulación hidráulica
    a. MicroServo SG90
    b. Motor de CC
    c. L293D – Controlador para Motor de CC

En caso de querer comprobar sus características de funcionamiento de cualquiera de ellos, se puede realizar yendo al Anexo correspondiente: *A. 'Datasheets'*.

6.2.1. Arduino MEGA2560 R3:

"El Arduino Mega 2560 es una placa de microcontrolador basada en el ATmega2560. Tiene 54 pines de entrada/salida digital (de los cuales 15 se pueden usar como salidas PWM), 16 entradas analógicas, 4 UART (puertos serie de hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP, y un botón de reinicio." [4]

La tarjeta contiene las mismas prestaciones que si fuera de la compañía Arduino, pero corresponde a la marca ELEGOO.

ILUSTRACIÓN 9: PLACA ARDUINO MEGA2560

La tensión de funcionamiento es de 5V, mientras que el voltaje de entrada puede variar desde 6 a 20V, siendo la recomendable entre 7 y 12V. La corriente, en continua, que circula por los pines I/O es de 20mA (evitando superar el límite de 40mA para no dañar la placa); en cambio, la que circula por el pin de 3'3V es de 50mA. Tiene, además, una memoria flash de 256 KB que emplean 8KB de esos para el gestor de arranque, otros 8KB de memoria SRAM y 4KB de EEPROM. Hay un led integrado en el pin 13 que cuando está a valor alto se encuentra encendido y a valor bajo apagado.

Por otro lado, la revisión 3 de esta placa contiene un microcontrolador ATmega16U2 que sirve como convertidor de USB a serie.

6.2.2. Fuentes de alimentación:

Se han utilizado 3 **fuentes de alimentación**, siendo 2 de ellas 2 transformadores de AC a DC de 9V con 1A y 700mA respectivamente y estando el primero conectado a una fuente del módulo de alimentación que regula la tensión a 5V o 3'3V según la necesidad.

ILUSTRACIÓN 10: FUENTE DE ALIMENTACIÓN 9V – 1A

ILUSTRACIÓN 11: FUENTE DE ALIMENTACIÓN REUTILIZADA 9V – 700MA

ILUSTRACIÓN 12: FUENTE DEL MÓDULO DE ALIMENTACIÓN 3,3V (IZQ.) Y 5V (DERECHA)

6.2.3. Pantalla LCD 4x20:

Pantalla digital de alta calidad, de tamaño 98 x 40 x 9.8 mm, distribuida en 4 filas y 20 columnas de caracteres, siendo cada carácter una matriz de puntos 5x7, con comunicación I2C conformada, además, por un sensor V5.0 (Sensor Shield Arduino IIC) que es quien la permite conectar a la placa Arduino únicamente con 4 cables gracias a dicho protocolo. Tiene la capacidad de regular la luminiscencia de la pantalla mediante un potenciómetro en la tabla. La temperatura de funcionamiento cubre un rango de -30ºC hasta 60ºC. Se alimenta con una tensión de 5 V, a la cual extrae a unos 40mA con la luz de fondo encendida y a unos 5mA con ella apagada.

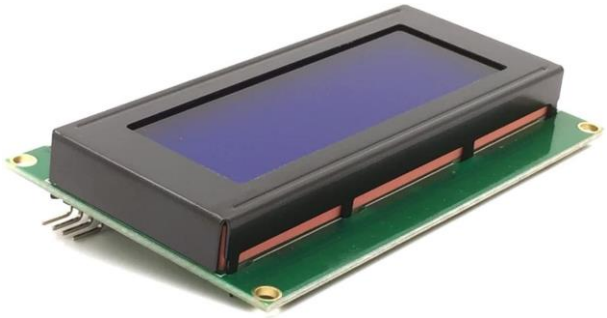


ILUSTRACIÓN 13: PANTALLA LCD 20X4

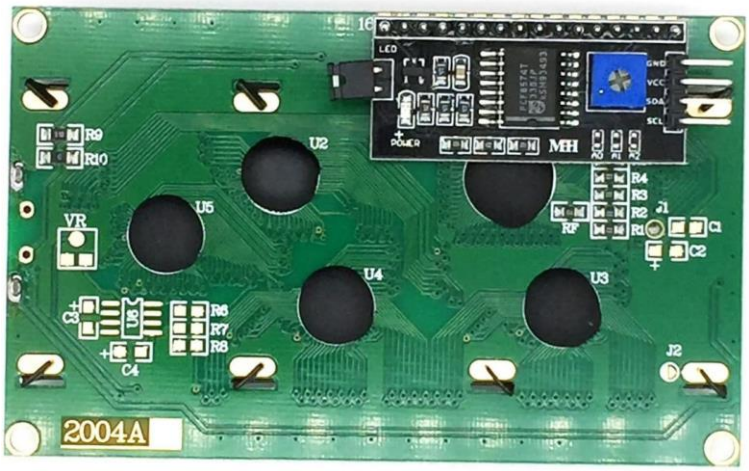


ILUSTRACIÓN 14: PANTALLA LCD 20X4 PARTE TRASERA

Los pines del sensor convertidor para conectar a la placa poseen la siguiente serigrafía: GND, VCC, SDA, SCL; haciendo referencia a 'Tierra', alimentación a 5V, datos y reloj respectivamente.

Para su correcta configuración, fue necesario el empleo de unas bibliotecas diferentes a la típica <*LiquidCrystal.h*> de Arduino que se utiliza para conectar 'a pelo' una LCD 2x16. En este caso se ha hecho uso del *protocolo* ***I2C***, por lo que son necesarias las bibliotecas: ***<Wire.h>*** (simplifica el uso del bus TWI) y ***<LiquidCrystal_I2C.h>***.

6.2.4. DHT 11:

Es un sensor de bajo coste y estabilidad a largo plazo que se encarga de la medida de la temperatura y de la humedad relativa (del ambiente); su uso se va a establecer para ir informando de las mismas en todo momento dentro de lo que sería nuestra "caja negra" con los componentes mediante una salida digital.

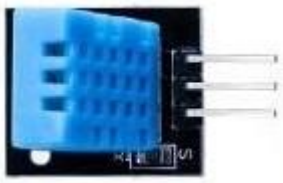


ILUSTRACIÓN 15: SENSOR Tª Y HUMEDAD DHT11

Dicho detector está compuesto por 4 pines, de los cuales se van a usar 3 que son los que salen de la placa en donde está soldado. Los pines corresponden, según se mira de frente y de izquierda a derecha, a V_DD (5V), DATA, el que viene sin conexión y la GND. Su resolución, en cuanto a temperatura y a humedad relativa, es de 16 bits cada. Se hace necesaria una resistencia pull-up de 5'1 KΩ para que cuando el bus esté inactivo, su estado sea a nivel alto.

Su modelado en el entorno IDE de Arduino se conforma con la librería ***<DHT.h>*** estableciéndose en su objeto el pin al que se conecta y del tipo que es el sensor; en este caso, DHT11.

6.2.5. DS3231:

Módulo de Tiempo Real muy preciso, de bajo consumo y alta velocidad (capaz de comunicarse con *interfaz* ***TWI***) que establece la hora y fecha actuales y las mantiene mediante una pila cuando se desconecta el sistema de la fuente de alimentación y de la transmisión por el puerto serie con el PC a través del USB. Dicha pila, es una batería de respaldo CR2032, con vida útil entre 2 y 3 años. La temperatura que soporta abarca desde los -40ºC a 85ºC (con una precisión de ±3,5ppm), siendo su rango de temperatura más funcional de 0ºC

a 40ºC (con precisión de ±2ppm); esto se puede traducir en que el reloj marca la hora con una precisión de ±5 o 6 segundos/año.

Es capaz de emplear el formato de tiempo digital (24 h) o analógico (12 h); e integra, a mayores, un sensor de temperatura digital (±3ºC) y dos despertadores.

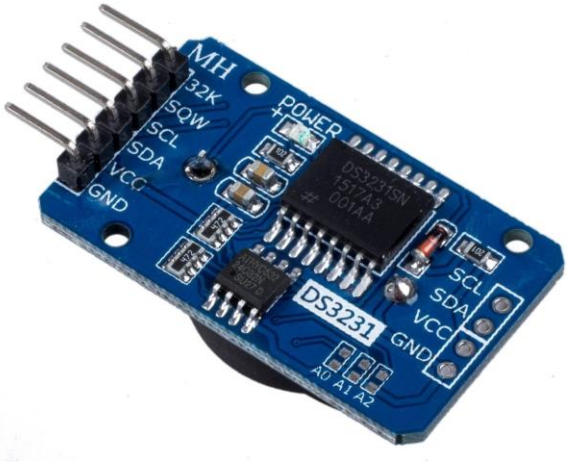


ILUSTRACIÓN 16: RELOJ DE TIEMPO REAL DS3231

Lo conforman 4 pines que en orden descendente en posición de lectura de la serigrafía son: 32K (Salida oscilatoria de 32K), SQW (Salida de onda cuadrada), SCL (Reloj serial [*interfaz I2C*]), SDA (Datos en Serie [*interfaz I2C*]), Vcc (2.3V – 5.5V) y GND (Tierra); de los cuales, los 2 primeros no son de uso obligatorio y los otros 4 se encuentran, también, en el lado contrario del módulo donde se sitúan los 6 pines.

La comunicación al llevarse a cabo con el *protocolo* ***I2C***, se transmiten los datos byte a byte; por lo que se ha establecido la librería ***<RTClib.h>*** como base para obtener la fecha y hora actuales, separando sus diversos campos para poder fijar las franjas horarias correspondientes y transmitir con mayor facilidad.

6.2.6. DS18B20:

Sensor digital de temperatura con encapsulado TO-92 que utiliza únicamente un pin para la transmisión de datos, ya que emplea el *protocolo* ***1-Wire*** que es bidireccional. En nuestro caso, se ha escogido con una cobertura de tubo de acero inoxidable por la conveniencia frente al agua y su robustez añadida. Tiene un alcance desde los -55ºC hasta los 105ºC con hasta 12 bits de resolución programable.

ILUSTRACIÓN 17: LOS 2 SENSORES DE Tª DS18B20

Dicho sensor consta de 3 patillas, unidas a unos cables rojo, amarillo y negro, a las cuales les corresponde una entrada de +5V; Vout y GND respectivamente.

Es necesario usar una resistencia Pull-Up de 4.7KΩ, que al no tener se ha conseguido empleando en paralelo una de 5'1KΩ con una de 10KΩ y dos en serie de 1KΩ y 330Ω

Para un correcto funcionamiento del sensor han sido necesarias instalar las siguientes librerías: ***"OneWire"*** (implementa el protocolo 1-wire) y ***"DallasTemperature"*** (establece las configuraciones y funciones pertinentes para medir la temperatura).

6.2.7. Higrómetro KY70:

Conjunto conformado por un **sensor de humedad HW-080** basado en la variación de la conductividad y un módulo **convertidor analógico HW-103**, ya que sin este último nos devolvería un valor binario al ser digital. Aunque no es muy preciso, para determinar si el suelo se encuentra húmedo (regado/llovido) o no (seco) sirve.

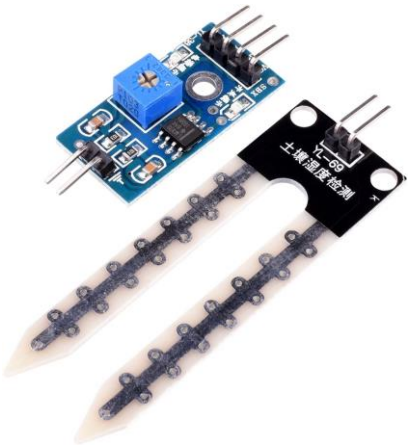


ILUSTRACIÓN 18: ELEMENTOS DEL KY70: HW-103 (ARRIBA) Y HW-080 (ABAJO)

El sensor en sí consta de dos pines que determinan la polaridad a tener en cuenta para conectarlos al convertidor. El convertidor también tiene dichos pines y por el extremo contrario consta de otros 4 que conforman en orden descendente: Vcc (a 5V), GND, D0 y A0. Además, el módulo convertidor, consta de 2 LEDs internos, uno para la alimentación y otro para determinar cuando pasa a HIGH la salida digital D0, que es regulada por un potenciómetro. En nuestro caso, se ha empleado la salida de datos A0.

Mediante un mapeo se establece la conversión de los 10 bits a porcentaje de humedad relativa, siendo el rango de valores de 0 a 1023 y de 100% a 0% respectivamente.

6.2.8. Sensor de Nivel de Agua:

Sensor analógico de dimensiones 62x20x8 mm con un área de detección de 40x16 mm^2 que consiste en 5 pares de trazas paralelas alternándose GND con las del propio sensor que contienen resistencia Pull-up. El fundamento consiste en cerrar dichos pares de líneas estañadas dándose un valor analógico en su medida y pudiendo determinarse así si ha llovido o no. Los valores que recoge se encuentran entre 0 y los 355 aproximadamente, ya que, al encontrarse de forma horizontal con la parte trasera pegada a la tapa, no es capaz de soportar gran cantidad de volumen de líquido sobre sus líneas como si estuviera en posición vertical sumergido en un recipiente.

ILUSTRACIÓN 19: SENSOR 'NIVEL DE AGUA'

Este módulo lo conforman 3 patillas con las serigrafías 'S', '+' y '-', de forma descendiente en la imagen correspondientes a la señal analógica, al valor de tensión positivo (5V) y a su negativo (0V común).

6.2.9. ESP8266-01S:

Módulo WiFi de los más baratos y de los más potentes del mercado con un funcionamiento de *3'3Vcc*. A diferencia del anterior modelo, este solamente tiene 1 led, en vez de 2, que indica cuando GPIO2 está en nivel bajo.

Sus pines al funcionar a 3'3V y el Arduino a 5V es necesario establecer un divisor de tensión que reduzca hasta esos 3'3V si la conexión va desde la placa Arduino hasta el ESP. Por esa razón se emplea el módulo de fuente de alimentación con salidas 3,3V y 5V.

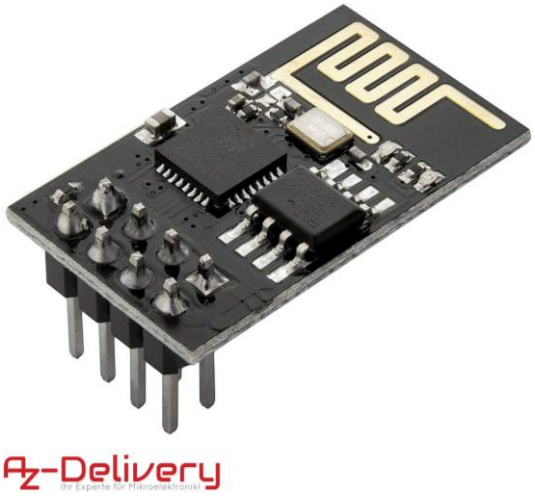


ILUSTRACIÓN 20: MÓDULO WIFI ESP8266-01S

Esta pequeña placa, que puede funcionar de forma autónoma como microcontrolador, se encuentra limitada por el número de pines que puede emplear para controlar elementos externos; siendo estos 8 pines los siguientes:

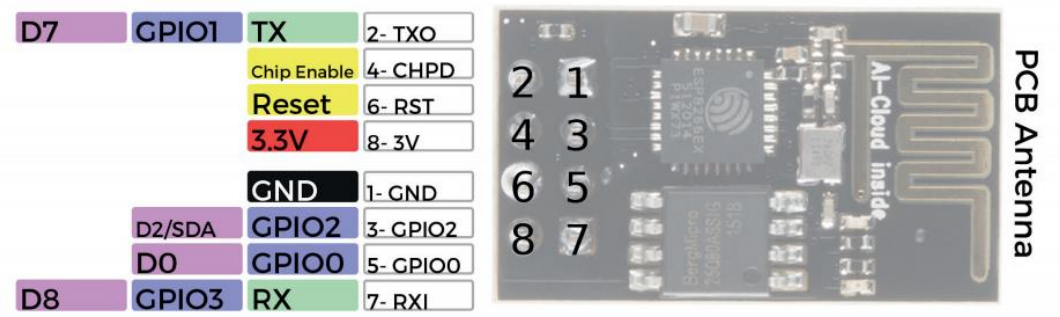


ILUSTRACIÓN 21: ESQUEMA PINES ESP8266-01S

Según la enumeración del esquema, cada pin hace las dotes de:

1) *GND* → 'Tierra'.
2) *TXO* → Transmisión de datos por puerto serie.
3) *GPIO2* → Pin genérico de Entrada/Salida 2.
4) *CHPD* → Habilitación del chip.
5) *GPIO0* → Pin genérico de Entrada/Salida 0 (sirve para flashear).
6) *RST* → Reset a nivel bajo.
7) *RXI* → Recepción de datos por puerto serie.
8) *Vcc* → 3,3V de alimentación.

Se ha de tener en cuenta, en cuanto a memoria se refiere, las características propias de dicha placa, siendo: 1MB de memoria flash, con 64KB y 96KB para las instrucciones y los datos de la RAM, respectivamente, y con 64 KB de

ROM de arranque; ya que son la que nos va a limitar la capacidad de almacenaje de datos y elaboración de la página web.

6.2.10. "Electroválvulas":

Debido a las limitaciones materiales, se prescinde de la parte hidráulica y no se construye como tal, sino que se simula; por lo que es prescindible la compra de las costosas electroválvulas para llevar a cabo esta maqueta. En caso de querer implementar el proyecto, se han de emplear dichas válvulas y, para ello, se hace necesario el uso de conversores de tensión para reducir de 230V hasta los 24V (o 9V) útiles o aumentar de los 5V hasta los 9V (o 24V), siendo mejor opción la reductora con un transformador.

En su lugar, la mejor forma de simular el transito necesario de agua es vertiendo el líquido desde un vaso y que la respuesta eléctrica de su control sea representada y reflejada por unos motores:

6.2.10.1. MicroServo Motor SG90:

Es un servo miniatura de dimensiones diminutas y ligero con alta potencia de salida, gran calidad y bastante económico, que funciona con la mayoría de los sistemas de radio control comerciales y, además, es compatible con casi todas las tarjetas electrónicas de control con microcontroladores; es decir, es capaz de utilizar cualquier hardware, biblioteca o código para controlar estos servos. Especialmente, dadas sus características de torque, tamaño y peso, realiza buena función en aeronaves; siendo ideal para aplicaciones en donde las últimas dos características son de gran importancia. Al mismo tiempo, no requieren de la construcción de un control para el motor con realimentación y caja de cambios; por lo que es una forma muy sencilla para mover lo deseado, ya que también viene con 3 distintos tipos de brazo.

ILUSTRACIÓN 22: MICRO SERVO SG90

Posee unas dimensiones: 22.0 x 11.5 x 27 mm, pesa 9 gramos (10.6 con el conector) y un torque de funcionamiento de 1.2 kg/cm a 4.8V que va a ser nuestra tensión aproximadamente, ya que sus requerimientos de energía son bastante bajos, lo cual permite alimentarlo con la misma fuente de

alimentación que el circuito de control. A ese mismo voltaje, la velocidad de giro es 0.12 seg / 60 °, aunque puede funcionar entre los 4-7,2V. Al contener un conector universal tipo "S", conformados por los cables rojo (Vcc), marrón (GND) y naranja (pulsos PWM), sirve también para los sistemas de radio como se mencionó anteriormente.

En este caso que nos concierne, aunque únicamente sea capaz de girar, en total 180º, unos 90º para un lado y para el otro partiendo de su posición central, es suficiente para ser el motor encargado de simular el paso o no del agua del riego por goteo, indicando así la apertura o cierre del grifo según la posición en la que se encuentre el brazo.

6.2.10.2. Motor de CC (con ventilador):

Motor estándar de CC, de tamaño de 130, que simula el funcionamiento de los aspersores en la parte del césped. Poseen un rango más amplio de funcionamiento, ya que su alimentación abarca desde los 4'5V hasta los 9V en vez de los 1'5-4'5V habituales en este tipo de motores; además de contener 2 cables ya soldados para un prototipado rápido.

ILUSTRACIÓN 23: MOTOR DE CC CON VENTILADOR

Con unas dimensiones de 27,5x20x15 mm^3 del cuerpo y de 8 mm x 2 mm de diámetro del eje, este motor soporta temperaturas desde los -10 ºC hasta los 60 ºC y 10 g de carga nominal, que, sin ella, la corriente máxima es de 70 mA y una velocidad de 9100 ± 1800 rpm. El voltaje nominal oscila por los 6V, pero nuestro uso será de 5V, pudiendo permitir una velocidad de carga de 4500 ± 1500 rpm y corriente máxima de 250 mA; siendo necesario 2V de arranque con un par de 20 g·cm y bloqueándose si alcanza los 500 mA.

Al ser un motor de CC, que pesa 17,5 g, es necesario establecer el control de giro en un sentido o en el contrario; por lo que ese control lo realiza el chip L293D. Independientemente del sentido de giro, se va a prefijar únicamente una dirección, siendo ésta cuando funcione la que indique que los aspersores estarían encendidos y el riego del césped activo.

6.2.10.3. L293D:

Controlador cuádruple medio-H de alta corriente, diseñado por Texas Instruments para conducir cargas inductivas, cuyo modelo 'D' es encargado de proporcionar corrientes de accionamiento bidireccionales de hasta 600 mA desde los 4,5V hasta los 36V y una temperatura de funcionamiento desde los 0º hasta los 70ºC.

ILUSTRACIÓN 24: CHIP L293D

Tienen lugar 4 salidas donde se habilitan por pares, la salida 1 y 2 con la Habilitación1 y la 3 y 4 con la Habilitación2, correspondiéndose cada parte a uno de los lados del chip. Además, por ello, "cada salida es circuito de accionamiento de tótem completo, con un disipador de transistores Darlington y una fuente pseudo-Darlington" como se dice en su ficha de datos del componente. Todas las entradas son tolerantes hasta los 7 V y compatibles con tecnología "*lógica transistor a transistor*" (TTL); es decir, su tensión característica es en el rango en torno a los 5V.

Cuando una entrada de habilitación es alta, los controladores asociados se habilitan y sus salidas están activas y en fase con sus entradas. Cuando la entrada de habilitación es baja, esos controladores están deshabilitados y sus salidas están apagadas y en el estado de alta impedancia. Con las entradas de datos adecuadas, cada par de controladores forma un H completo (o puente) reversible variador adecuado para aplicaciones de motor o solenoide.

En el modelo que vamos a emplear, el L293D, hay unos diodos de alta velocidad de salida ya integrados para supresión de transitorios inductivos; de esta forma, se reduce la complejidad y el tamaño general del sistema. Por otro lado, se tiene un terminal VCC1 separado de otro VCC2 para minimizar la disipación de energía del dispositivo proporcionado para las entradas lógicas.

VCC1 debe suministrarse a 5 V y los datos proporcionarse de niveles de entrada lógica válidos y habilitando las entradas para su funcionamiento. Hay que tener en cuenta las cargas conectadas a las salidas para conectar VCC2 a una fuente de alimentación capaz de suministrar la demanda de corriente y tensión necesarias.

6.3. Comandos AT

6.3.1. ¿Qué son los 'Comandos AT'?

La comunicación entre el hombre y un terminal modem se realiza mediante instrucciones codificadas que conforman un lenguaje, y esas son los comandos AT. Denominados así, por la abreviatura de ***Attention***, y preceden a todos los comandos.

Llamados también *'Conjunto de comandos de Hayes'* haciendo honor a su desarrollador *Dennis Hayes*, por la compañía *Hayes Communications*, para poder configurar y proporcionar instrucciones al interfaz de comunicación con un MÓDEM en 1977. Más adelante, se siguió desarrollando y expandiendo el juego de comandos, gracias al avance del baudio, por las compañías *Microcomm* y *US Robotics* hasta universalizarlo y convertirlo en estándar abierto de comandos para configurar y parametrizar módems.

Los comandos AT tienen como finalidad principal la comunicación con modems, aunque se ha adoptado, para comunicarse con sus terminales, como estándar también en la telefonía móvil GSM. Para éstos últimos, la configuración del interfaz y el proporcionar instrucciones, como enviar mensajes o realizar llamadas, a los terminales se lleva a cabo con una serie de comandos AT específicos, además de muchas otras configuraciones del terminal.

Por esta razón, la implementación de los comandos AT no depende del canal de comunicación, sino que corresponde a los distintos dispositivos GSM a través del cual son enviados estos comandos.

En nuestro caso, el ESP8266-01S al ser un dispositivo que implementa este conjunto de comandos Hayes, se considera ***'compatible Hayes'***.

Los comandos AT más utilizados se ven representados en la siguiente imagen [5]:

Comando AT	Descripción	Parámetros	Ejemplo
AT+RST	Reiniciar módulo		AT+RST
AT+CWMODE=<modo>	Cliente o punto de acceso	1 = cliente 2 = P. Acceso 3 = Cliente & AP	AT+CWMODE=3
AT+CWLAP	Lista de puntos de acceso		AT+CWLAP
AT+CWJAP=<SSID>,<PWD>	Conectar a un punto de acceso	SSID = Nombre de punto de acceso PWD = Contraseña de PA	AT+CWJAP="miwifi","contraseña"
AT+CWQAP	Desconectar del punto de acceso		AT+CWQAP
AT+CWSAP=<SSID>,<PWD>,<CH>,<ENC>	Configurar el punto de acceso	SSID = Nombre de punto de acceso PWD = Contraseña de PA CH = Canal 802.11 (1 a 7) ENC = Encriptado (0 = no encriptado)	AT+CWSAP="miwifi","contraseña"
AT+CIPSTATUS	Estado de la conexión		AT+CIPSTATUS
AT+CIPMUX=<MUX>	Conexión sencilla o multiple.	0 = Sencilla 1 = Multiple	AT+CIPMUX=0
AT+CIPSTART=<TIPE>,<URL>,<PUERTO>	Habilitar una conexión UDP o TCP	CIPMUX = 0: Tipo = TCP o UDP Puerto: Puerto del servidor CIPMUX = 1: ID = ID de conexión de 0 a 4 Tipo = TCP o UDP Puerto: Puerto del servidor	AT+CIPSTART="TCP",www.google.com",80
AT+CIPSEND=<LEN> or AT+CIPSEND=<ID>,<LEN>	Enviar datos al servidor	CIPMUX = 0: LEN = Número de bytes CIPMUX = 1: ID = ID de conexión LEN = Número de bytes	AT+CIPSEND=28 GET http://www.google.com/
AT+CIPCLOSE	Cerrar la conexión		AT+CIPCLOSE
AT+CIFSR	Obtener IP		AT+CIFSR=?
AT+CIPSERVER=<MODO>,<PUERTO>	Configurar servidor TCP	Modo = Cerrar/abrir modo de servidor 0 = cerrar; 1 = abrir	AT+CIPSERVER=1,80

TABLA 1: TABLA 'COMANDOS AT'

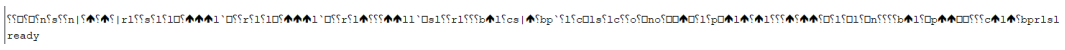
AT:

Es el comando "inicial" a realizar, comprueba si hay envío de datos. Devuelve un '*OK*' de ser así (al igual que va a suceder en casi todos los comandos a utilizar si se ha ejecutado el comando satisfactoriamente).

AT+RST:

Es la instrucción que reinicia el módulo para poder usarlo de nuevo desde "el inicio". Al final del reinicio muestra un '*ready*' indicando que se ha terminado de volver a configurar y anteriormente un '*OK*' para demostrar que se ha cumplido.

```
AT

OK
AT+RST

OK

 ets Jan  8 2013,rst cause:4, boot mode:(3,7)

wdt reset
load 0x40100000, len 1856, room 16
tail 0
chksum 0x63
load 0x3ffe8000, len 776, room 8
tail 0
chksum 0x02
load 0x3ffe8310, len 552, room 8
tail 0
chksum 0x79
csum 0x79

2nd boot version : 1.5
  SPI Speed      : 40MHz
  SPI Mode       : QIO
  SPI Flash Size & Map: 8Mbit(512KB+512KB)
jump to run user1 @ 1000
```



ready

ILUSTRACIÓN 25: COMANDOS AT Y AT+RST

En este caso es un reinicio del temporizador de vigilancia mediante 'Hardware', al ser la causa "4".

Las razones de reinicio a través de código ROM son [6]:

Rst cause No.	Cause
0	Undefined
1	Power reboot
2	External reset or wake-up from Deep-sleep
4	Hardware WDT reset

TABLA 2: TABLA CAUSAS DE REINICIO

AT+GMR:

Este mando muestra la versión del firmware.

```
AT+GMR
AT version:1.2.0.0(Jul  1 2016 20:04:45)
SDK version:1.5.4.1(39cb9a32)
v1.0.0
Mar 11 2018 18:27:31
OK
```

ILUSTRACIÓN 26: COMANDO AT+GMR

AT+CWMODE:

Es la directriz que configura el modelo de operación. Hay 3 modos: estación (cliente), punto de acceso 'AP' (servidor o host) o estación + punto de acceso (ambos). A mayores, hay la opción de '***Test***', para mirar qué modelos válidos existen; la de '***Consulta***', comprueba en qué modo se encuentra configurado, consulta la información del AP que se conecta; y la de '***Ejecución***', establece el modo deseado, configura la información del AP a conectar.

```
AT+CWMODE=?
+CWMODE:(1-3)

OK
AT+CWMODE=3

OK
```

ILUSTRACIÓN 27: COMANDO AT+CWMODE OPCIÓN 'TEST' Y 'EJECUCIÓN'

AT+CIPMUX:

Es la instrucción que permite habilitar la configuración de múltiples conexiones, siendo 0 para una sola conexión y un 1 para varias (hasta un máximo de 4).

Únicamente se puede cambiar el modo tras haber desconectado todas las conexiones, siendo necesario reiniciarlo si se ha iniciado el servidor previamente.

AT+CIPSERVER:

El comando 'CIPSERVER', como su propio nombre indica, crea un servidor, pero solamente puede hacerlo cuando "AT+CIPMUX=1". Puede llegar a contener dos parámetros, siendo obligatorio únicamente el primero: modo y puerto. El ***Modo*** es una variable booleana donde el '1' crea el servidor y el '0' indica la eliminación de este (acompañado del reinicio del módulo para poder llevarse a cabo); y el ***Puerto***, si no se especifica se establece el configurado por defecto, si no, hay que establecer el número del puerto a utilizar.

El monitor del servidor se crea automáticamente al crear el servidor y el cliente que se conecte a un servidor se le asigna una *identificación ID* y ocupará una conexión.

```
AT+CIPMUX=1

OK
AT+CIPSERVER=1,80

OK
```

ILUSTRACIÓN 28: COMANDOS AT+CIPMUX Y AT+CIPSERVER

AT+CWLAP:

Es la instrucción que muestra la lista disponible de los puntos de acceso o los busca según condiciones específicas. En el segundo caso hay que especificar su ***SSID*** y la ***dirección MAC*** mediante una cadena de caracteres y el ***canal*** a utilizar.

En ambos casos se devuelven un conjunto de *parámetros* que son los siguientes:

- *ECN*: Controla el soporte de la notificación de conexión explícita, cabiendo las distintas formas de cifrado:
  - 0 = OPEN → red abierta.
  - 1 = WEP → red de cifrado considerado inseguro.
  - 2 = WPA_PSK → red con cifrado, con clave compartida previamente.
  - 3 = WPA2_PSK → red con cifrado, con clave compartida previamente y con mejoras al ser el nuevo estándar WiFi.
  - 4 = WPA_WPA2_PSK → red con cifrado, que es compatible con los dos estándares anteriores.
- *SSID*: identificador del Punto de Acceso (nombre de la red WiFi).
- *RSSI*: intensidad de señal (nivel con el que recibe el módulo WiFi).
- *MAC*: dirección MAC de la red.

A mayores, también los acompañan:

- *Canal*: por donde se encuentra el estándar WiFI (del 1 al 11)
- Y otros dos parámetros no especificables, que se desconocen qué son por no usarse.

```
AT+CWLAP
+CWLAP:(3,"MiFibra-3278",-83,"94:6a:b0:5b:32:7a",1,-74,0)
+CWLAP:(0,"ConfiguraEduroam",-75,"24:de:c6:a1:92:60",1,32767,0)
+CWLAP:(5,"eduroam",-69,"24:de:c6:a1:90:01",1,-39,0)
```

ILUSTRACIÓN 29: COMANDO AT+CWLAP

AT+CJAP:

Para conectar a un servidor es necesario usar este comando estableciendo, en primer lugar, su ***SSID*** y, acto seguido, la ***contraseña*** proporcionada.

```
AT+CWJAP="Xperia L1_31cc","[redacted]"
WIFI DISCONNECT
WIFI CONNECTED
WIFI GOT IP

OK
```

ILUSTRACIÓN 30: COMANDO AT+CWJAP

AT+CIFSR:

Esta directriz obtiene, primeramente, la dirección del IP local como cliente y/o servidor y, en segundo lugar, la dirección MAC.

```
AT+CIFSR
+CIFSR:APIP,"192.168.4.1"
+CIFSR:APMAC,"a6:cf:12:f4:ae:36"
+CIFSR:STAIP,"192.168.43.63"
+CIFSR:STAMAC,"a4:cf:12:f4:ae:36"

OK
```

ILUSTRACIÓN 31: COMANDO AT+CIFSR

Las líneas 2 y 3 representadas hacen referencia al ***'Access Point'*** (o servidor) y las dos siguientes corresponden a la ***'Station'*** (o cliente); siendo la dirección de la cuarta línea la empleada *(+CIFSR:STAIP, " ...")*.

AT+CIPSEND:

Es el encargado de enviar los datos, teniendo en cuenta 3 variantes, de las cuales, hay 2 que dependen si hay múltiples o no conexiones. En primer lugar, se puede establecer la modalidad de '***Test***', para saber si se están transmitiendo datos o no. Las 2 siguientes, establecen la *longitud del mensaje* que va a ser remitido, hasta un máximo de 2048 caracteres por paquete enviado, refiriéndose a una única conexión con este parámetro, o a varias, si previamente estableces el *ID de la conexión*. Por último, hay que ***enviar el comando***, tal cual, para que envíe el contenido del mensaje con un intervalo de 20 ms entre paquetes, devolviéndose un *">"* cada vez que quiera ejecutar una de las variantes; es decir, cuando esté preparado para aceptar un paquete de datos de determinada longitud establecida, sin excederse del máximo.

```
,355:GET / HTTP/1.1
Host: 192.168.1.108
UsAT+CIPSEND=0,1500

OK
>
```

ILUSTRACIÓN 32: COMANDO AT+CIPSEND DANDO LA ACEPTACIÓN DE ENVÍO DE PAQUETE

AT+CIPCLOSE:

Cierra una conexión TCP o UDP, pudiendo realizarse la modalidad de '***Test***', o cerrarla si es una conexión múltiple o única, ***estableciendo la ID*** para identificarla o no.

```
,381:GET / HTTP/1.1
Host: 192.168AT+CIPSEND=0,1562

OK
>
Recv 1562 bytes

SEND OK
AT+CIPSEND=0,848

OK
> 1,CONNECT

+IPD,1,276:GET /favicon.ico HTTP/1.1
Host: 192.168.43.63
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101 Firefox/72.0
Accept: image/webp,*/*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive


Recv 848 bytes

SEND OK
AT+CIPCLOSE=0
0,CLOSED

OK
```

ILUSTRACIÓN 33: COMANDOS AT+CIPSEND Y AT+CIPCLOSE

6.4. Descripción del Sistema implementado:

Partiendo de la base de usar la herramienta Arduino para nuestro proyecto, se barajaron distintas posibilidades, siendo la final la ***placa MEGA 2560 R3*** (como ya vimos anteriormente), básicamente, por contener un mayor número de pines a emplear y, gracias a eso, una mayor posibilidad de encontrar módulos estandarizados que los abarquen. Por ello, se hizo la compra de un pack que contenía algunos componentes y resistencias, entre otros elementos, que nos iban a servir.

ILUSTRACIÓN 34: KIT ELEGOO

La alimentación del sistema se realiza con ***2 transformadores AC/DC*** y una ***fuente del módulo de alimentación***. El primero de 9V se conectaba a ese último módulo que estabiliza la tensión en 5V (o 3,3V) y son pertenecientes al pack comprado de Arduino; mientras tanto, el transformador restante es reutilizado de un antiguo aparato electrónico ya sin utilidad que, para alimentar la placa Arduino, es suficiente y necesario (más conveniente que realizar la conexión desde la fuente del módulo de alimentación a la entrada V_{in}).

La configuración del LCD se realiza implantando la *dirección I2C* de la pantalla, número de *columnas* y número de líneas o *filas*: "(***0x27,20,4***)" para que pueda funcionar con las librerías anteriormente citadas.

```
LiquidCrystal_I2C lcd_I2C(0x27,20,4); // Conf. el LCD: direc. 0x27, 20 colum. y 4 líneas
```

ILUSTRACIÓN 35: OBJETO 'LCD_I2C' CREADO PARA LA PANTALLA LCD

Para cualquier elemento a utilizar, por norma general, se ha de crear su respectivo objeto para posteriormente iniciarlo.

En primera instancia, el ***Reloj de Tiempo Real DS3231*** viene con la configuración de fábrica; es decir, con la fecha y hora por defecto de cuando se fabricó; por lo que lo primero que se tiene que hacer es configurar las actuales. Esto se consigue buscando la fecha y hora que tiene el PC al cargar el programa con la función: *"**rtc.adjust(DateTime(__DATE__,__TIME__));**"*, pero como esta opción únicamente ha de realizarse la primera vez que se conecta el módulo (o cada vez que se quiera corregir el posible retardo que se haya generado con el paso del tiempo), se ha establecido una variable booleana (*CONFIG_RTC*) que nos establece esta configuración cuando la definimos a '1', porque de no ser así nos daría error al desconectar la placa del puerto serie del ordenador, ya que se encontraría buscando la fecha y hora del PC en todo momento.

```
/*-----Comprobación RTC:-----*/
//  Se actualiza del PC, si procede, la hora para su configuración
if (CONFIG_RTC == 1)
{
  rtc.adjust(DateTime(__DATE__,__TIME__));  //Establece hora ACTUAL del PC

  Serial.println("¡¡¡Fecha y hora actualizadas!!!");
  lcd_I2C.setCursor ( 0, 1 );
  lcd_I2C.print("    Actualizado!    ");
  delay(1000);  // 1 seg. para apreciarlo
}
```

ILUSTRACIÓN 36: AJUSTE DE LA CONFIGURACIÓN DE FECHA Y HORA ACTUALES CON EL PC

Para una mejor visualización de las mismas, se muestran las distintas variables de fecha y hora por separado con el formato *'DD/MM/AAAA – hh:mm:ss'*.

```
/*-----RTC:-----*/
//  Se establece la fecha y la hora actuales
ahora = rtc.now();

// Mostrar por el PC fecha y hora
#if PUERTO_SERIE
  Serial.print("  -·-");
  // FECHA:
  Serial.print(ahora.day(), DEC);
  Serial.print("/");
  Serial.print(ahora.month(), DEC);
  Serial.print("/");
  Serial.print(ahora.year(), DEC);
  Serial.print(" | ");
  // HORA:
  Serial.print(ahora.hour(), DEC);
  Serial.print(":");
  Serial.print(ahora.minute(), DEC);
  Serial.print(":");
  Serial.print(ahora.second(), DEC);
  Serial.println("-·-  ");
  Serial.println(" ");
#endif  // Cierre de PUERTO_SERIE

// Imprimir por la LCD fecha y hora actuales
lcd_I2C.setCursor ( 0, 0 );
lcd_I2C.print(" MENU FECHA Y HORA: ");
// FECHA:    (DD/MM/AAAA)
lcd_I2C.setCursor ( 2, 2 );
lcd_I2C.print("Fecha: ");
lcd_I2C.print(ahora.day(), DEC);
lcd_I2C.print("/");
lcd_I2C.print(ahora.month(), DEC);
lcd_I2C.print("/");
lcd_I2C.print(ahora.year(), DEC);
lcd_I2C.print(" ");
// HORA:     (hh:mm:ss)
lcd_I2C.setCursor ( 2, 3 );
lcd_I2C.print("Hora:  ");
lcd_I2C.print(ahora.hour(), DEC);
lcd_I2C.print(":");
lcd_I2C.print(ahora.minute(), DEC);
lcd_I2C.print(":");
lcd_I2C.print(ahora.second(), DEC);
    delay(2000);
```

ILUSTRACIÓN 37: FORMATO FECHA Y HORA: 1º PUERTO SERIE Y 2º PANTALLA LCD

Como se puede apreciar, se plasma dicho formato por la pantalla LCD y/o por el Puerto Serie, según se estipule la variable booleana *PUERTO_SERIE*, que si se encuentra predeterminada a '1' lo mostraría, además de por la pantalla LCD, por el Puerto Serie para apreciarlo por el Monitor Serie del Arduino IDE. Esta forma de reflejar los datos y variables se llevará a cabo en cada uno de los distintos componentes del proyecto.

Además, para saber en qué franja horaria nos ubicamos, se ha establecido una variable entera '*h*' que identifica en qué hora nos encontramos para poder usarla como *Condición de Testeo*.

En segundo lugar, se hace aconsejable conocer las variables de nuestro entorno; es decir, saber la temperatura y humedad relativa del ambiente. El conocerlas lo va a llevar a cabo el sensor ***DHT 11***, que, aunque no sea de gran precisión, pero sí muy barato, hace que sea el módulo ideal para desempeñar la función, ya que, simplemente, es conocer una aproximación del marco en el que vamos a trabajar. Las funciones '***readHumidity()***' y '***readTemperature()***' se van a encargar de establecer la Humedad y Temperatura ambientes respectivamente.

```
/*-----DHT 11:-----*/
//  Se establece la temp. y hum. ambientales
humAmb = dht.readHumidity();
tempAmb = dht.readTemperature();

// Mostrar por el PC temp. y hum. amb.
#if PUERTO_SERIE
  Serial.print("  Temperatura: ");
  Serial.print(tempAmb);
  Serial.println("°C"); // Ej.: Temperatura: __°C
  Serial.print("  Humedad: ");
  Serial.print(humAmb);
  Serial.println("%");
  Serial.println(" ");
#endif  // Cierre de PUERTO_SERIE

// Imprimir por la LCD temp. y hum. amb.
lcd_I2C.clear();
lcd_I2C.setCursor ( 0, 0 );
lcd_I2C.print("    T");
lcd_I2C.write(7);   // &ordf (o 170) = 'ª'
lcd_I2C.print("yHum Amb:    ");   // Titulo:"    TªyHum Amb:    "
lcd_I2C.setCursor ( 2, 2 );
lcd_I2C.print("T");
lcd_I2C.write(7);
lcd_I2C.print(" amb: ");   // "Tª amb: "
lcd_I2C.print(tempAmb);
lcd_I2C.print((char)223);   // &deg (o 176) = '°'
lcd_I2C.print("C");
lcd_I2C.setCursor ( 2, 3 );
lcd_I2C.print("Hum amb: ");
lcd_I2C.print(humAmb);
lcd_I2C.print("%");
    delay(2000);
```

ILUSTRACIÓN 38: FORMATO DHT11: 1º PUERTO SERIE Y 2º PANTALLA LCD

Los ***sensores de temperatura DS18B20***, aun pudiéndose establecer varios a un único pin de Arduino gracias al *protocolo 1-Wire*, se ha escogido el separar cada sensor en su correspondiente pin, por la sencillez que ello supone y al tener disponibilidad suficiente en nuestra placa Arduino Mega. En caso de haberlos implementado en un solo pin, habría que haber distinguido la dirección de cada sensor para diferenciarlos al usar el mismo bus, siendo este procedimiento más engorroso y menos claro de vista al comprador por su mayor dificultad en la programación. El único inconveniente que posee nuestra configuración de emplear 2 pines es que son necesarias 2 resistencias Pull-Up de 4'7K.

En este caso, con este tipo de sensor, para establecerse la temperatura, primero ha de enviarse el comando para leerse y después se obtiene el dato, pero determinando el ***formato*** en el cual se quiere: si en *grados Celsius* (0) o *grados Fahrenheit* (1).

```
/*-----DS18B20:-----*/
// Se establecen las Temp. actuales
sensorTemp1.requestTemperatures();      //Se envía el comando para leer la temperatura
temp1= sensorTemp1.getTempCByIndex(0); //Se obtiene la temperatura en °C (=0) o en K (=1)
sensorTemp2.requestTemperatures();
temp2= sensorTemp2.getTempCByIndex(0);
```

ILUSTRACIÓN 39: MANERA DE ESTABLECER LA TEMPERATURA EN LOS DS18B20

Los sensores de humedad, nombrados ***Higrómetros KY70***, pueden usarse como entradas digitales o analógicas, habiéndose empleado el segundo caso. Aunque no consiga detectar hasta el 100% de humedad, en valor absoluto, al encontrarse en torno al 80% su valor máximo (al establecer el mapeo desde el valor 0 hasta el 1023) es suficiente para determinar si se ha realizado un riego (o ha llovido) o no.

```
/*-----KY70:-----*/
n=1;
humedad_1 = analogRead(sensorKY70_1);   // Valor analógico
  selH1 = SelecHum(humedad_1, n);   // Determina el estado de humedad
  hum1 = map(analogRead(sensorKY70_1), 0, 1023, 100, 0);  // Valor en %

n=2;
humedad_2 = analogRead(sensorKY70_2);   // Valor analógico
  selH2 = SelecHum(humedad_2, n);   // Determina el estado de humedad
  hum2 = map(analogRead(sensorKY70_2), 0, 1023, 100, 0);  // Valor en %

n=3;
humedad_3 = analogRead(sensorKY70_3);   // Valor analógico
  selH3 = SelecHum(humedad_3, n);   // Determina el estado de humedad
  hum3 = map(analogRead(sensorKY70_3), 0, 1023, 100, 0);  // Valor en %

n=4;
humedad_4 = analogRead(sensorKY70_4);   // Valor analógico
  selH4 = SelecHum(humedad_4, n);   // Determina el estado de humedad
  hum4 = map(analogRead(sensorKY70_4), 0, 1023, 100, 0);  // Valor en %
```

ILUSTRACIÓN 40: MANERA DE ESTABLECER LA HUMEDAD EN LOS HIGRÓMETROS KY70

Como se visualiza en la imagen, se emplea una función creada llamada '***SelecHum***' que ordena los distintos rangos en los que se puede encontrar la humedad del suelo. La jerarquía de más húmedo a menos sería: "*Empapado*", "*Mojado*", "*Húmedo*", "*Seco*" y, si el sensor se encuentra demasiado salido del terreno, "*Aire*".

Otro sensor analógico utilizado es el de '***Detección de Lluvia***', llamado comercialmente como 'Nivel de agua'. Su rango es más limitado que el anterior (siendo el mínimo 0, aunque el máximo unos 350 de 1023), pero al ser un sistema enfocado a su uso en verano, sirve para determinar si ha llovido o no únicamente, con la ayuda de los sensores de humedad del suelo.

```
/*-----LLUVIA:-----*/
sensLluvia=analogRead(Sensor_Lluvia);
Serial.print("   //Nivel lluvia: ");
Serial.println(sensLluvia);
Serial.println("\\");
if(sensLluvia>=280)
{
  lcd_I2C.clear();
  lcd_I2C.setCursor(0, 1);
  lcd_I2C.print("¡¡Lluvia detectada!!");
  Serial.println("     ¡¡Lluvia detectada!!");
}
```

ILUSTRACIÓN 41: CONDICIÓN A PARTIR DE LA QUE SE DETECTA LLUVIA

Para determinar si es necesario regar o no, se han establecido las siguientes condiciones:

```
/*-----RIEGO:-----*/
if(((hum1+hum2)/2<=28 && sensLluvia<=280 || (humAmb<=15 && tempAmb>=30)) && franja==true )
{
  riego_cesped=true;
  digitalWrite(Enable1_L293D,HIGH);
}
if(((hum3+hum4)/2<=28 && sensLluvia<=280 || (humAmb<=15 && tempAmb>=30)) && franja==true )
{
  riego_rosales=true;
}
lcd_I2C.clear();
```

ILUSTRACIÓN 42: CONDICIÓN DE RIEGO

Esto quiere decir que si cada zona, independiente, tiene una media de menos del 28% de humedad del suelo o la temperatura ambiente supera de los 30ºC con una humedad ambiente inferior al 15%, se ha de realizar riegos consecutivos hasta que cambien dichas condiciones, siempre y cuando se encuentren dentro de la franja de riego.

En cuanto al ***módulo WiFi*** se refiere, la configuración del ESP8266-01S, se ha seguido las referencias tomadas para la programación y carga de un ESP-01.

En primer lugar, para establecer la *carga del programa* en la memoria es necesario establecer el pin GPIO0 a nivel bajo; es decir, conectarlo a tierra y establecer así el ***Modo UART***. A continuación, para '***flashearlo***', se dejan los pines GPIO0 y GPIO2 sin conectar, ya que en su configuración interna se encuentran conectados a un Pull-Up; por lo que, su valor por defecto es a nivel alto, y en este microcontrolador es a 3'3V. Teniendo en cuenta esto último, es necesario establecer un divisor de tensión a la salida del pin TX_1 de nuestro Arduino para que no se queme el pin RX de nuestro ESP-01S, ya que los pines de Arduino funcionan a 5V y los del ESP-01S a 3'3V.

Antes de proseguir con todo ello, es necesario recordar las características propias de dicha placa que nos van a delimitar el entorno en el cual vamos a poder trabajar: 1MB de memoria flash, con 64KB y 96KB para las instrucciones y los datos de la RAM, respectivamente, y con 64 KB de ROM de arranque.

Para la creación del servidor, finalmente, no fue necesario implantar la biblioteca oficial en el entorno Arduino correspondiente a la comunidad ESP8266, ya que la conexión se ha realizado mediante los *'**Comandos AT**'* y se comprueba si hay envío o no de datos, a través de un identificador, como se observa en la siguiente imagen:

```
/*-----SERVIDOR:-----*/
if(Serial1.available())  // Comprobación de que ESP está enviando datos
{
  if(Serial1.find("+IPD,"))    // Se comprueba si hay Identificador asociado
  {
    delay(1000);

    digitalWrite(LED_Amarillo, HIGH);

    int conexionID = Serial1.read()-48;
```

ILUSTRACIÓN 43: GENERACIÓN DEL SERVIDOR WEB

Se ha empleado el lenguaje, sin usar librerías, en donde todas las acciones las realiza el cliente; es decir, se ha programado con formato *'**Vanilla Javascript**'*. Lo que sí que se construyó fue una función que se encarga de enviar los datos carácter a carácter para desarrollar la página web, siendo ésta la que viene a continuación:

```
/*     ---FUNCIÓN DE ENVÍO DE DATOS:---
 * Envía cualquier dato o conjunto de comandos
 * al servidor ESP-01S, si se encuentra disponible,
 * que se ha de escribir o config. en la pág. web.
 * mediante la lectura de caracter a caracter.
/* --- . --- . --- . --- . --- . --- . --- . --- . --- */
String sendData(String command, const int timeout, boolean debug)
{
  String response = "";
  Serial1.print(command);    // Envia lo escrito para que lo lea el ESP
  long int time = millis ();
  while( (time+timeout) > millis() )
  {
    while( Serial1.available() )
    {
      char c = Serial1.read();    // Lee el siguiente caracter
      response+=c;
    }
  }

  if(debug)
  {
    Serial.print(response);
  }
  return response;
}
```

ILUSTRACIÓN 44: FUNCIÓN PARA ENVIAR LOS DATOS A LA PÁGINA WEB

Si no se conoce qué IP hemos de implementar, se accede al *'cmd'* del sistema y se escribe: "ipconfig" para saber la Dirección IPv4 que tenemos a utilizar como base. En mi caso, el router asigna a mi ordenador la IP *192.168.1.***, por lo que tenemos que establecer una que se encuentre dentro de ese rango. Con la ayuda del comando ***AT+CIFSR*** se vincula la IP que se ha asignado al ESP-01S y a partir de ella ya se puede acceder sin problema desde el WiFi de casa. Para poder consultar los valores actuales, otra opción será emplear la dirección MAC, realizando una matriz de bytes, a partir de esa dirección IP que hemos fijado antes.

Por otro lado, para poder acceder desde fuera de nuestro WiFi, es necesario conocer la *dirección IP Pública*, que es dinámica y, por tanto, cambiante cada cierto tiempo. Al ser volátil, es conveniente usar un *servicio de direccionamiento dinámico* (***DDNS***) para que no haya que reescribirla cada vez que se modifique. La que se ha obtenido anteriormente es la dirección Privada de nuestro dispositivo, que, para poder vincularla a la Pública, es necesario hacer uso de un Puerto o Salida.

A la hora de ahorrar en memoria de las variables, se hace aconsejable utilizar *'**const char[] PROGMEM**'*, ya que almacena el contenido en memoria flash

permitiendo así quitarle carga de trabajo a la otra. Se emplea habitualmente al incrustar cadenas HTML que no se van a modificar, que es como se ha llevado a cabo la mayoría de nuestra página web.

Previamente, para poder realizarse toda esta toma de datos anteriores, ha de establecerse las condiciones de testeo que se han conformado de la siguiente manera:

```
// CONDICIONES DE TESTEO:
// - Dentro de las horas establecidas:
//    * Verano -> [6:00-9:00) y [20:00-23:00)
//    * Primavera & Otoño -> [7:00-10:00) y [19:00-22:00)
// - Testear recargando la Pág Web (Fuera de ellas tambien)

epoca = EstAnual(ahora.day(),ahora.month());

if( ((((6 <= h) && (h < 9)) || ((20 <= h) && (h < 23))) && epoca==2) ||
    ((((7 <= h) && (h < 10)) || ((19 <= h) && (h < 22))) && (epoca==1 || epoca==3)) ||
    (Testear==true) )
{
  lcd_I2C.clear();
  digitalWrite(LED_Test, HIGH);
```

ILUSTRACIÓN 45: CONDICIÓN DE TESTEO

Finalmente, para compactar y proteger todo este "tinglado" se ha utilizado un Tupperware de dimensiones 14'5x20x6 cm^3 de plástico duro como carcasa. Se le han realizado unos cortes y ciertos agujeros para colocar la LCD, introducir los respectivos cables de alimentación y puerto serie USB, además de los correspondientes huecos para sacar los sensores.

Todo el código desarrollado, tanto su función principal donde se encuentra el '*Setup*', el '*Loop*' y la función de enviar los datos, como las funciones externas, se han plasmado de manera ordenada en el Anexo *B. Código*. Su lectura introducirá en materia, complementará y facilitará el del subcapítulo próximo a través de su siguiente diagrama.

6.5. Proceso ejecutado:

A través de un Diagrama de Actividades se muestra cómo se desarrolla el código que hemos implementado y cómo quedó finalmente haciéndose una clara distinción entre la parte del '*Setup*' y del '*Loop*'.

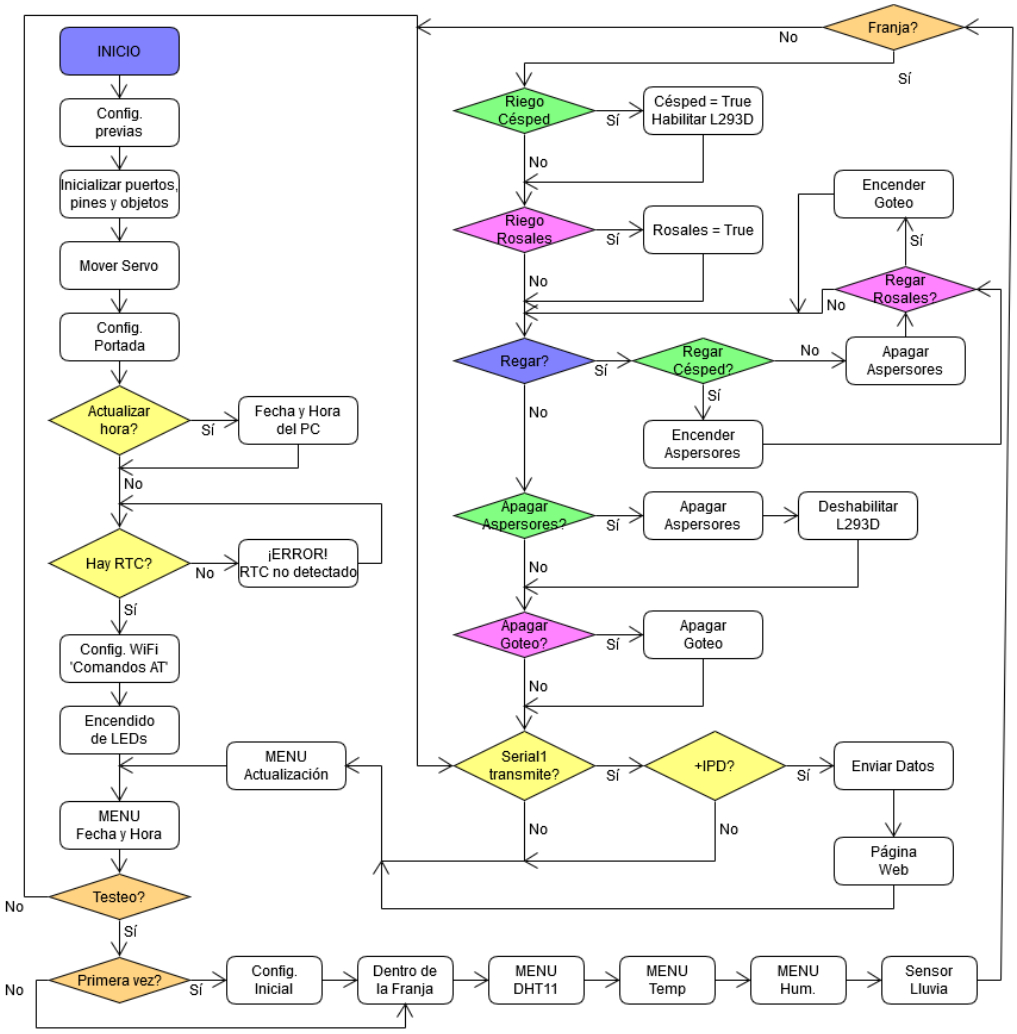


FIGURA 1: DIAGRAMA DE ACTIVIDADES DEL PROYECTO

La primera acción realizada es lo que se ha llamado en el esquema como *'Configuraciones previas'* y esto es lo que se lleva a cabo antes del '***Setup***'. Consiste en determinar las librerías, funciones, pines, variables globales y asignación de los objetos a utilizar.

La siguiente parte que comprende desde *'Inicializar puertos, pines y objetos'* hasta después del *'Encendido de LEDs'* es lo que abarca nuestro '***Setup***'. Básicamente lo que realiza es inicializar las partes que se van a emplear, moviendo acto seguido el servo para saber que se ha terminado la configuración de ellas; después se pinta en el LCD las portadas del proyecto para después comprobar si es necesario actualizar la hora y si se encuentra el Reloj de Tiempo Real conectado. Finalmente se establece la conexión a Internet y se iluminan todos los LEDs para saber que ha terminado de realizarse.

La última parte y la que más abarca, es donde se empieza a repetir el proceso una y otra vez en *bucle*, por esa razón es el '***Loop***'. Iría determinado desde antes del *'Menú Fecha y Hora'*, que es donde se establece la lectura de la hora actual, hasta después del *'Menú Actualización'*, que es donde se ha terminado de visualizar todos los datos si no ha tenido lugar ningún error en el envío. En esta zona, si se establece la condición de testeo, se realiza la toma de datos de las distintas variables y determina si es necesario activar un tipo de riego u otro para después mostrarlo por la página web; en caso de no cumplir con la condición, comprueba si se insta la petición mediante la recarga de la página web y si tal circunstancia es positiva, enseña por la página web los últimos datos testeados; si por el contrario no hay solicitud realiza la actualización y sigue mostrando la fecha y la hora únicamente.

6.6. Modificaciones realizadas:

En primera instancia se pensó en mantener activos los sensores de humedad en todo momento para poder acceder a los datos desde cualquier dispositivo en cualquier instante, pero esta opción se sustituyó por otra opción más factible en cuanto a mantenimiento de la vida útil de los mismos. Se estableció que estos sensores únicamente se activaran cuando se les enviara una solicitud para ello; es decir, permanecen inactivos hasta que se pide mostrar su valor. Esto tiene lugar a lo largo de todo el día, pero cuando se encuentran en las franjas horarias de riego, se establece que testee los valores cada cierto tiempo, para comprobar que se cumplen las condiciones óptimas o es necesario alargar el riego realizando otro más.

A la hora de mostrar por la LCD los caracteres "º" y "ª" salían símbolos totalmente distintos, por lo que ha sido necesario crearlos de forma manual con la ayuda de la matriz de puntos 5x8. Al tener que realizar este trabajo, se construyó una función encargada de ello, en donde se guardaron los "moldes" de las distintas matrices y se generan directamente el carácter deseado. Se crearon además de otros comandos que hacen la función de bordes en el menú mientras se conecta al WiFi, imágenes que representan los logos de la EII y de la Uva, que se representan también dentro de la función. Los últimos se emplean para configurar una portada del proyecto en el LCD. Finalmente, "º" se identificó el código y se escribe directamente él: *'(char)223'* dentro de la función de imprimir por pantalla.

Partiendo de la base para la configuración de la página web, en un primer momento se programó para que únicamente pudiera acceder un cliente a observar los datos, ya que al ser para un jardín particular no sería necesario más, por lo que si quería comprobar cómo se encontraba el jardín, no podía hacerlo a la vez desde el PC y el teléfono móvil. De esta forma, se estableció

que se pudieran conectar hasta 4 clientes, siendo los distintos inquilinos; así se corregía el "error" de no poder acceder desde 2 puntos distintos a la vez.

Al mismo tiempo, se estableció la IP local del router como conexión web, pero resultaba que en caso de querer acceder a ella desde fuera del WiFi de casa (con los datos propios desde el móvil) no permitía que se cargara, por lo que se tuvo que hacer uso de la dirección MAC del dispositivo, en vez de la IP, para poder conectarse desde fuera de casa.

En cuanto a la comunicación se refiere, en primera instancia, se contempló la idea de usar formularios web, ya que son la manera más sencilla y simple de transmisión de información, pero éstos obligan a recargar la página para enviarla y eso, a fin de cuentas, es un mecanismo anticuado; por lo que fue descartado desde un primer momento.

A continuación, se intentó crear la página web mediante una función que imprimiera en pantalla todo lo que se quería visualizar empleando código 'html', pero, aunque plasmaba en la web todo lo deseado, esta opción no permitía tener control en las decisiones que se querían tomar, como que el cliente enviara una petición para obtener algún dato. Además, esta forma de obrar almacenaba el contenido en la memoria correspondiente a variables, siendo ésta de gran peso, incluso llegando a sobrepasar los límites. Principalmente por esta razón, fue necesario buscar otra forma de proceder.

Una solución, intermedia, que se encontró fue guardar la parte estática del contenido en memoria flash, en vez de en memoria de las variables; así se reduce el consumo de RAM, ya que es escasa en nuestro dispositivo.

Para el uso de contenido exclusivamente estático (no varía entre peticiones) es aconsejable el uso de sistema de archivos SPIFFS. Para su utilización se necesita la librería ***<FS.h>.*** Se va a encargar de servir ficheros (html, css, js,...) consiguiendo el funcionamiento dinámico mediante código JavaScript y llamadas al 'Backend' al cargar el cliente este contenido. Es aconsejable, por no decir necesario si es una carga de trabajo extremadamente grande, que se sirvan los archivos comprimidos en formato GZIP, para así ocupar menor ancho de banda, su velocidad de transmisión sea mejor y suponga menor carga para el servidor, ya que el descomprimirlo recae sobre los clientes, no sobre el servidor. Esto supone una mejora de rendimiento importante, sobre todo si el entorno de trabajo (framework) son extensos JavaScripts.

Se barajó después la opción de emplear AJAX (Asynchronous JavaScript And XML), pero esta es una petición que requiere que el cliente sea el que inicie la comunicación y la lanza al HTTP; es decir, no hay un mecanismo desde el servidor para poder llamar al cliente, y no es lo que buscamos. Para ello se barajaron las opciones: Websockets o MQTT.

El empleo de Websockets permite una comunicación bidireccional con bajo lag, debido a que la conexión se mantiene en todo momento abierta, por lo que es conveniente para nuestro caso, ya que, además, su velocidad de transmisión, entre servidor y cliente, es alta para obtener datos en tiempo real. Por consiguiente, el tener la conexión continuamente abierta consume, como es de esperar, recursos, disminuyendo la atención a clientes, pero esto no es problema, ya que con uno o dos clientes nos bastaría. Además, no tendrían que definir '*endpoints*', sirviéndose el contenido directamente desde SPIFFS (Serial Peripheral Interface Flash Fail System); es decir, comunicando mediante '*frontend'*.

El caso anterior es correcto, pero poder atender varios clientes con una sola dirección o puerto, es mejor, ya que se puede lanzar la página web y el Websocket desde el mismo puerto. Esto es lo que sucede cuando se implementan Websockets Asíncronos.

También se sopesó la opción de usar protocolo UDP, pero, aunque sea una solución de transmisión muy rápida, no existe confiabilidad en la comunicación; es decir, no comprueba si los paquetes han llegado al cliente, no hay acuse de recibo; por tanto, no verifica errores ni son necesarios los paquetes que generan la conexión; por consiguiente, suponen menor carga al servidor.

En la actualidad, lo más común es utilizar un API REST para la comunicación y la información se intercambie mediante ficheros Json. En cambio, a nuestro pesar, por la sencillez que suponía el emplear los ***Comandos AT*** sin la necesidad de adentrarse en un largo estudio de las librerías para los distintos casos citados anteriormente (aunque esa investigación exhaustiva se llevó a cabo), finalmente se realizó una función que enviara carácter a carácter lo que se quería plasmar en la página web mediante código, especialmente, 'html'. No es la mejor opción, como ya se ha evaluado en este apartado, pero sí la más sencilla y apta por las limitaciones de memoria.

6.7. Implementación del conexionado:

En este apartado se va a tratar de visualizar cómo se procedió a la hora de ir configurando los módulos y de qué manera se fueron colocando para que, finalmente, todos tuvieran su respectivo lugar en el prototipo.

Como es lógico, obviamente, se fueron configurando de forma individual cada sensor para luego ir implementándolos, todos a la vez, uno a uno hasta formar el conjunto final. La forma de proceder fue la siguiente:

En primer lugar, debido al exceso de unidades que se tenían a nuestra disposición, se realizaron las primeras pruebas con la estructuración del ***DHT11***, conectándose de la forma que se puede observar en la imagen de a continuación, con el cable de color rojo va dirigido a la alimentación de 5V, el negro a Tierra y el azul al *pin digital 31* para la transmisión de datos.

Previamente, se configuró la ***pantalla LCD*** para poderse mostrar los datos por ella, empleando un módulo de I2C quedando de la siguiente manera:

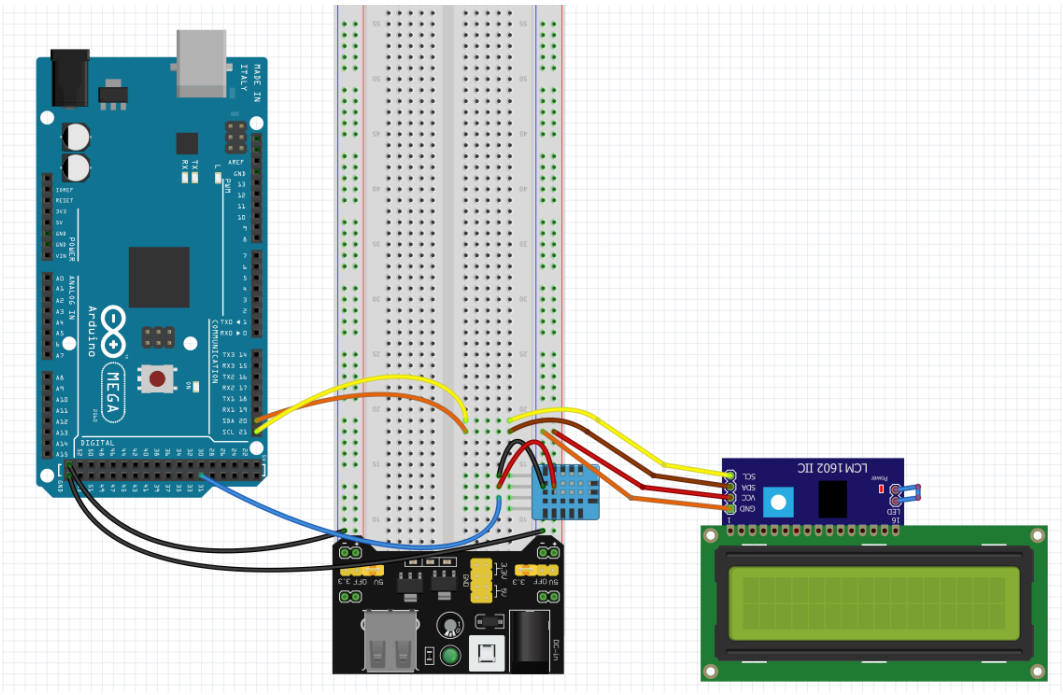


FIGURA 2: FRITZING LCD & DHT11

En este caso el cable naranja hace referencia al GND, el rojo sigue siendo Vcc, el marrón conecta la señal de datos y el amarillo la del reloj.

A continuación, se implementaron unos ***LEDs*** para hacer referencias luminosas con lo que se pedía en cada momento en nuestro código y, además, se determinó la fecha y horas actuales con un ***Reloj de Tiempo Real***.

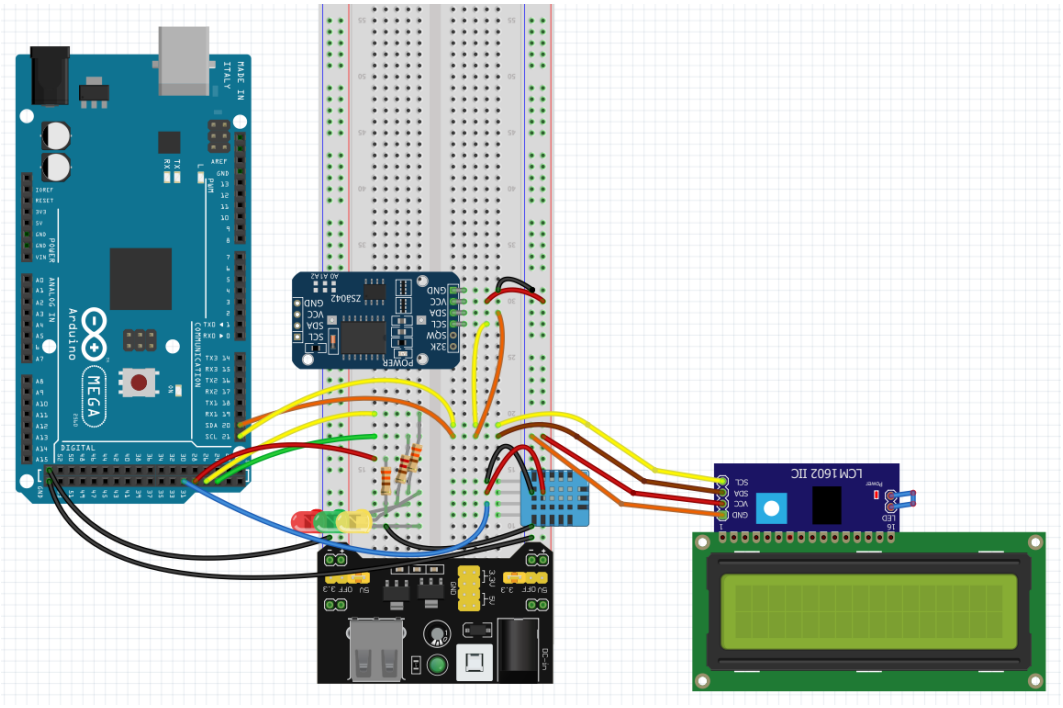


FIGURA 3: FRITZING ... + LEDS Y RTC

Como se puede apreciar a simple vista, en ambos bloques, la Alimentación va referenciada en rojo y la Tierra en negro. Los LEDs van independientemente de cualquier otro elemento, salvo de sus respectivas resistencias para regular la corriente que circule por ellos; por eso, dependiendo de su color, se conectan a los *pines digitales 23*, *25* y *27*, referenciándose al verde, amarillo y rojo respectivamente.

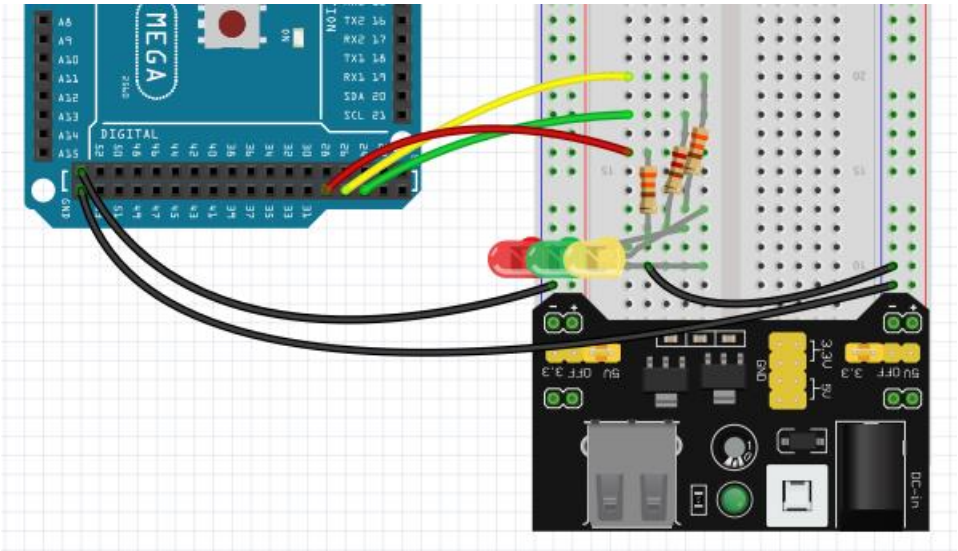


FIGURA 4: FRITZING LEDS SOLO

El RTC al compartir los pines SDA y SCL con la pantalla LCD, se va a usar el mismo color amarillo para el SCL, pero el naranja para el SDA, porque es el

color empleado para conectar ambos con el Arduino MEGA; correspondientes a los *pines 20 y 21* (SDA y SCL).

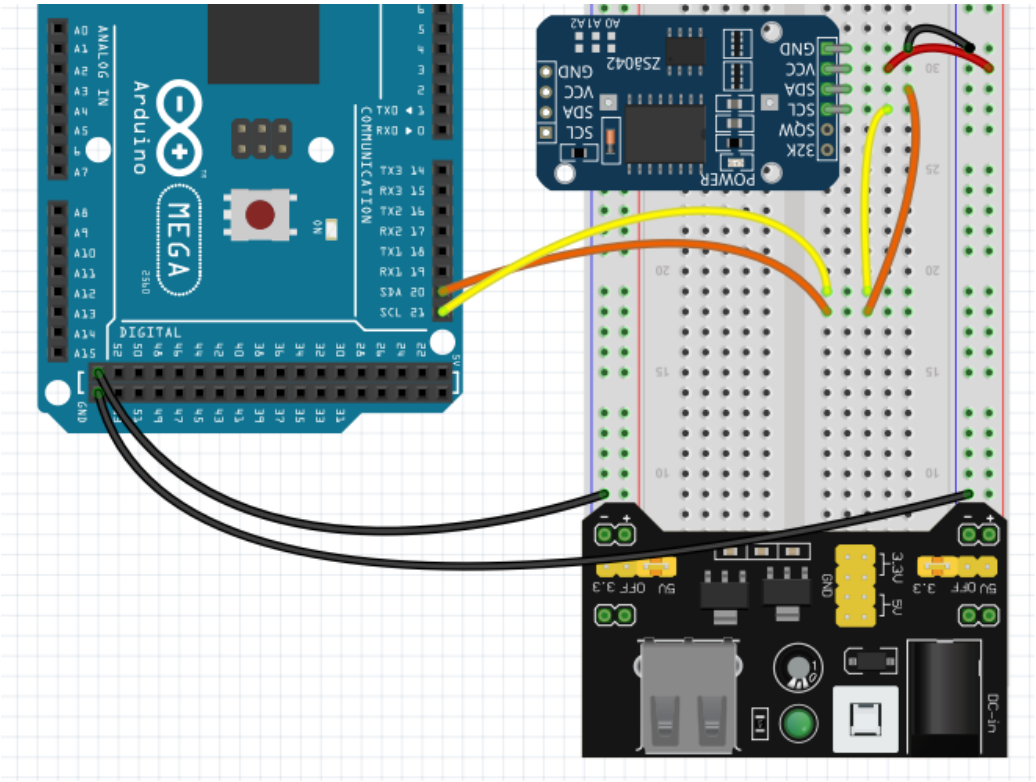


FIGURA 5: FRITZING RTC SOLO

Se siguió con los ***sensores de temperatura*** *y* ***humedad*** respectivamente.

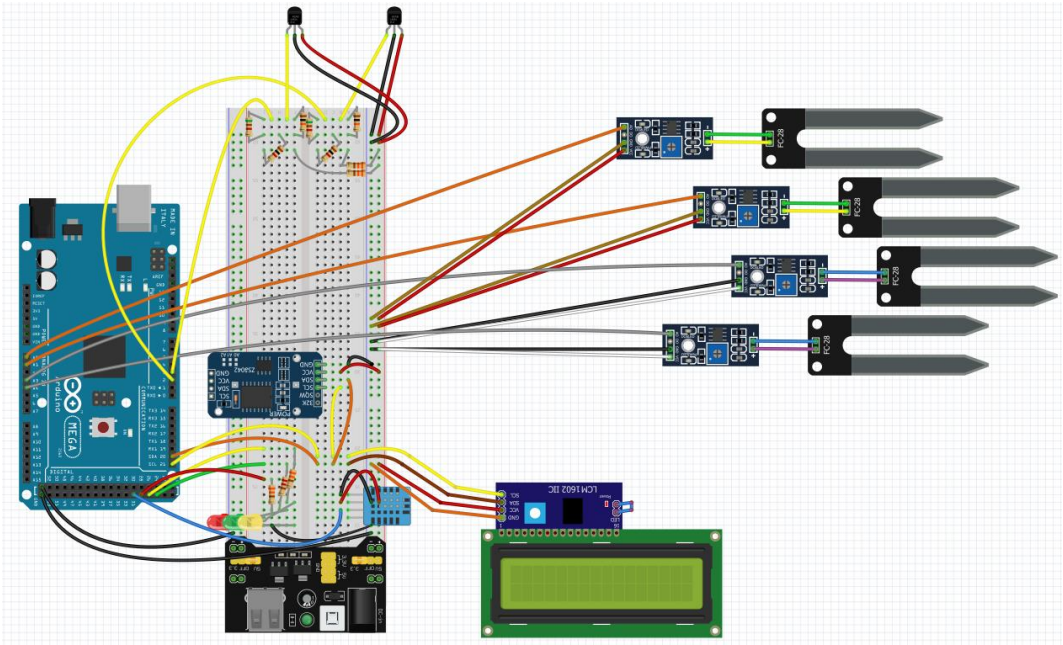


FIGURA 6: FRITZING ... + DS18B20 Y KY70

Para los de temperatura fue necesario estañar cables con terminación macho a los que ya venían para poder manipularlo mejor en nuestro prototipo; se optó por el color rojo típicamente para la tensión, el negro habitual para los 0V y el de los datos a uno de color blanco (representado con amarillo en la imagen). Los pines empleados para los datos son para el ***Sensor1*** (el de la derecha) el 2 y para el ***Sensor2*** (el de la izquierda) el 3 como se aprecia a continuación:

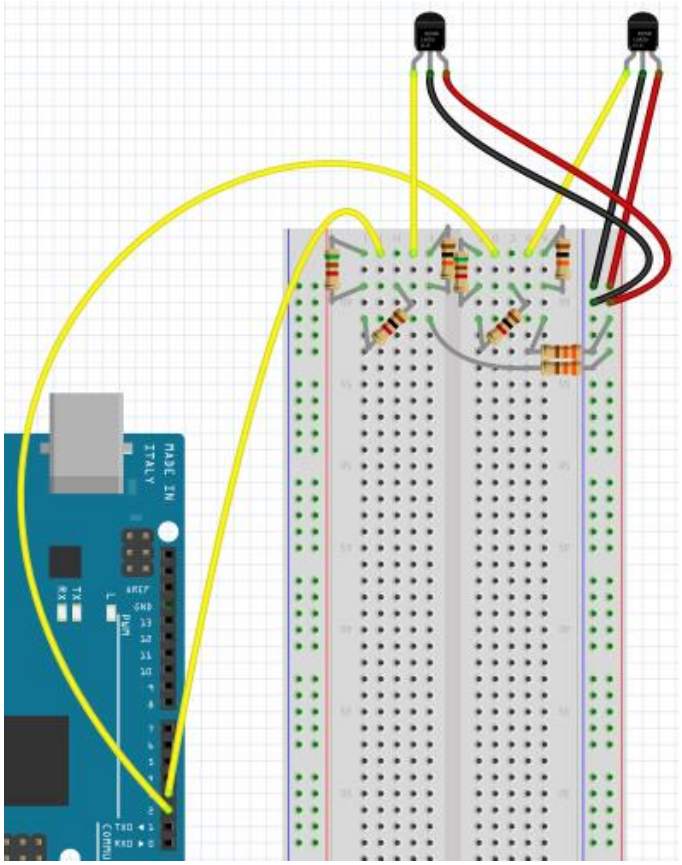


FIGURA 7: FRITZING DS18B20 SOLO

En cambio, en los de humedad se realizó una clara distinción por colores, dependiendo si eran los correspondientes a la *parte del* ***Césped*** o de los ***Rosales***.

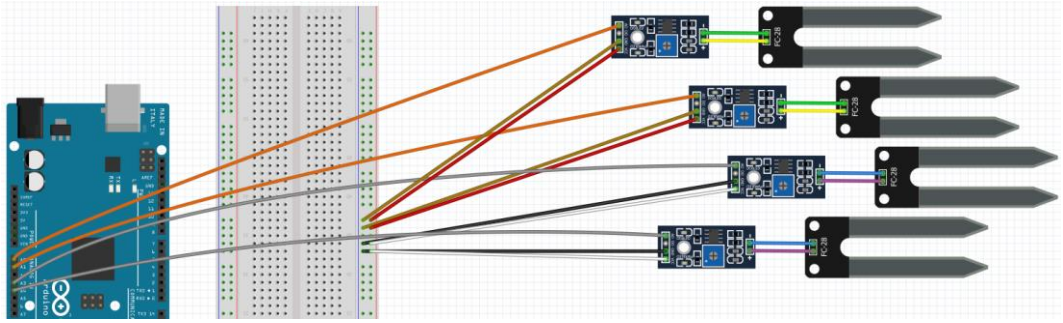


FIGURA 8: FRITZING KY70 SOLO

El primer grupo conecta la primera parte, de la sonda al módulo I2C, mediante el color verde referenciando al negativo y el amarillo al positivo; la que va desde este último hasta el Arduino, la segunda parte, son Vcc (rojo), GND (marrón) y conexión analógica (naranja). El segundo grupo conecta el positivo con el morado y el azul con el negativo en primera parte; siendo lo que procede blanco para Vcc, negro para GND y gris para los pines analógicos.

Los ***detectores de humedad*** están enumerados del 1 al 4 en orden descendente en la imagen; por ello, se sabe con certeza que el número ***1*** establece conexión con el *pin A0*, el ***2*** con el *A1* y el ***3*** y el ***4*** con el *A3 y A4*, porque se dejó un pin de separación para distinguir entre los correspondientes al ***Césped*** *(los 2 primeros)* y a ***los Rosales*** *(los 2 siguientes)*.

Luego, la parte más complicada a mi parecer, ya que existen diferentes maneras de poder conectarlo para uno u otro funcionamiento, es la configuración del ***módulo WiFi***.

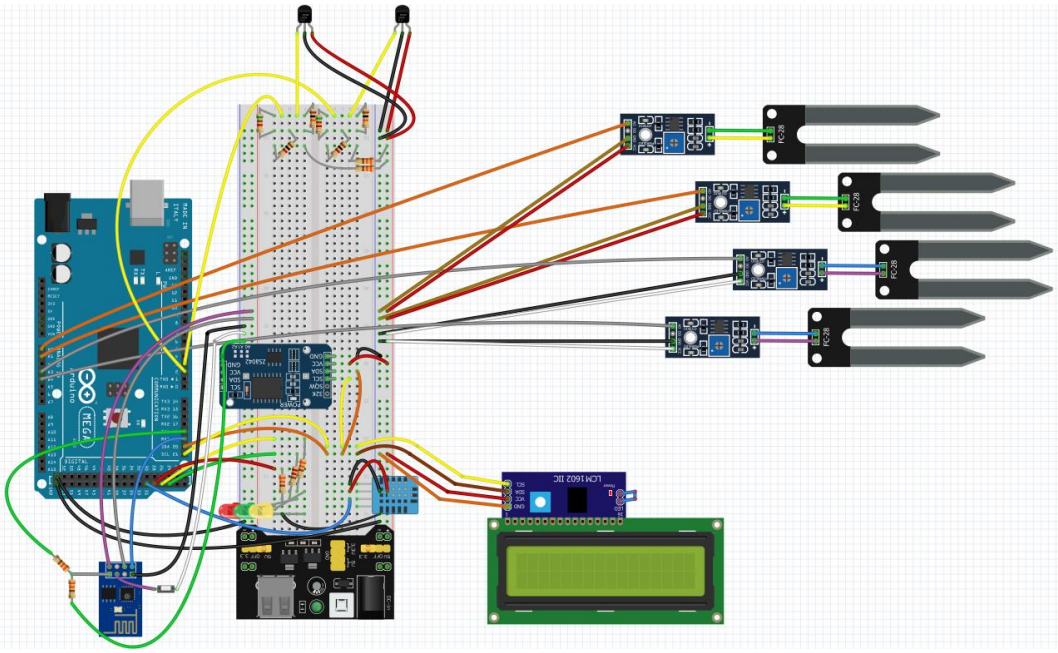


FIGURA 9: FRITZING ... + ESP8266-01S

En este caso, los colores son de vital importancia, ya que una confusión entre GND y 3,3V podría implicar la avería del dispositivo. Empezando, en el módulo, por el pin de arriba a la izquierda según se ve en la siguiente imagen, el cable morado se conecta a 3,3V al igual que el siguiente del siguiente de color gris oscuro. Simplemente, el siguiente se encontraría sin conexionado alguno, como el pin de debajo del gris oscuro. El último de la fila de arriba, con cable azul marino, se conecta al puerto Rx1 (*pin 19 de la placa*).

Comenzando en la hilera de pines inferior, según la vista de la imagen, el cable negro, que se encuentra a la derecha del todo, está vinculado con GND.

El cable morado que se adhiere con uno blanco sirve para *flashear* y se encuentran unidos únicamente *al subir el programa*. El último cable gris, divido por dos cables verdes, el superior establece conexión con Tx1 (*pin 18 de la placa*) y el inferior con Tierra.

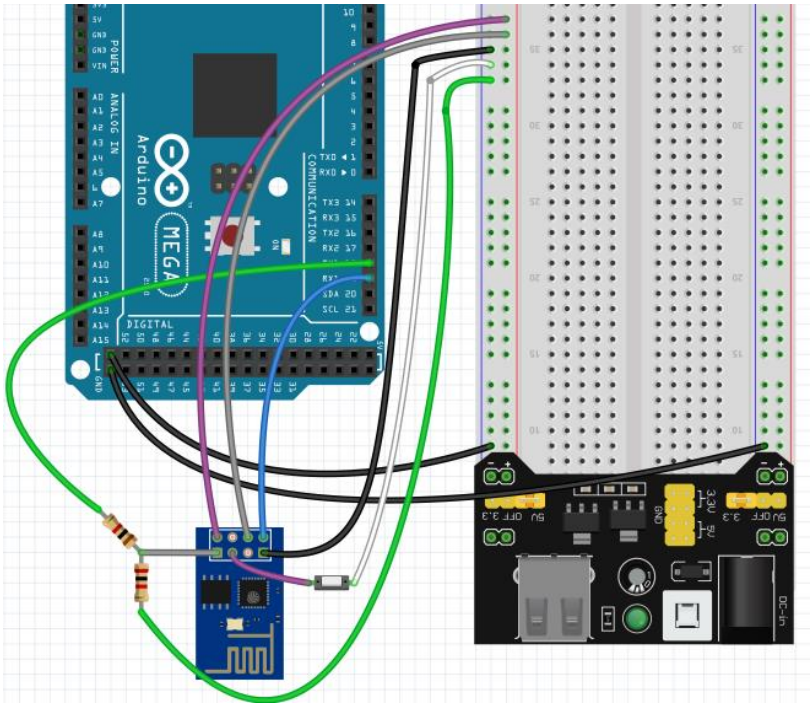


FIGURA 10: FRITZING ESP8266-01S SOLO

Por último, se instaló el ***sensor de detección de nivel*** que se va a emplear como sensor detector de lluvia.

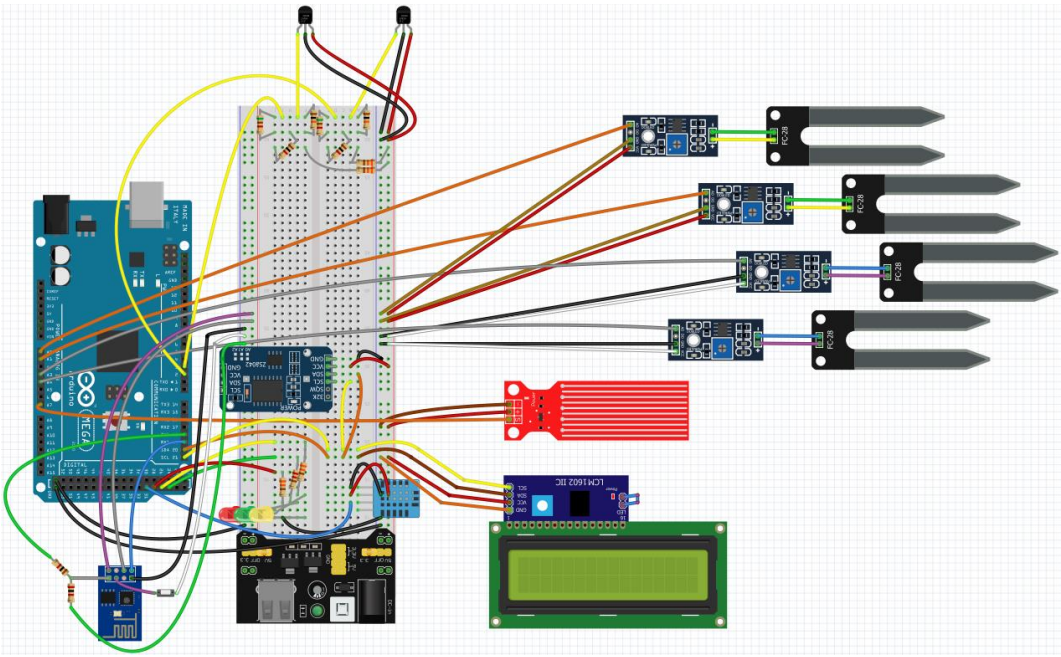


FIGURA 11: FRITZING ... + SENSOR LLUVIA

Los colores típicamente usados en nuestro proyecto se encuentran también en este dispositivo: Alimentación (rojo), 0V (marrón) y datos analógicos (naranja). Este último conectado al *pin A7* del Arduino MEGA.

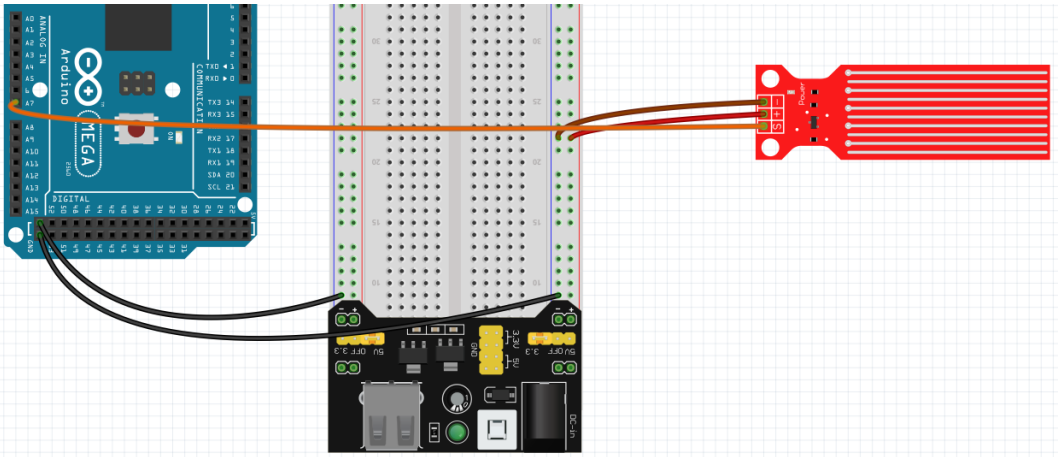


FIGURA 12: FRITZING SENSOR LLUVIA SOLO

Finalmente, para la simulación de la parte hidráulica de los aspersores y del goteo, como se encuentran controlados de manera eléctrica, se conectaron un ***motor de CC con ventilador*** y un ***micro servo***.

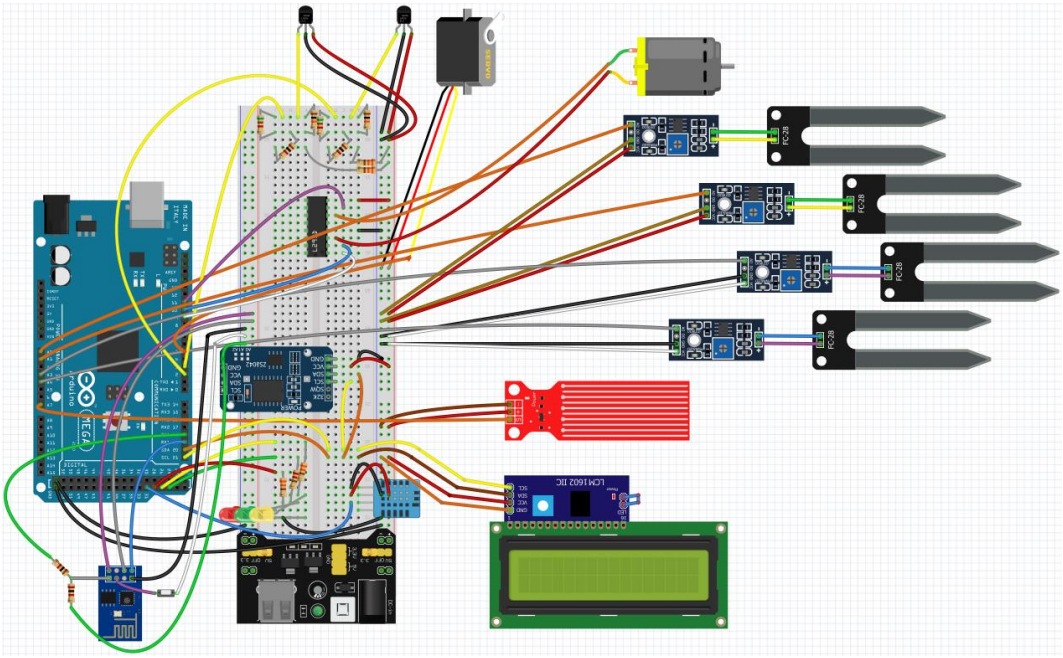


FIGURA 13: FRITZING ... + MOTORES

El Motor de Corriente Continua tiene 2 terminales representados por los colores rojo (+) y naranja (-), que han de ser controlados por el *chip L293D* con las *patillas 3* y *6* respectivamente. De este chip empezando por la patilla de abajo a la derecha (referenciado como *pin 1*), según la imagen, sale el cable blanco que permite la habilitación o no de este lado del chip, estando

conectado al *pin 9* del Arduino. A su vez, los cables azul y morado, enlazados a los *pines 10 y 11* de la placa, son los que determinan si el giro es horario o antihorario respectivamente. Como viene siendo habitual, en el chip su *patilla 8* se encuentra conectada a 5V con un cable rojo y la *patilla 4* a GND con uno negro.

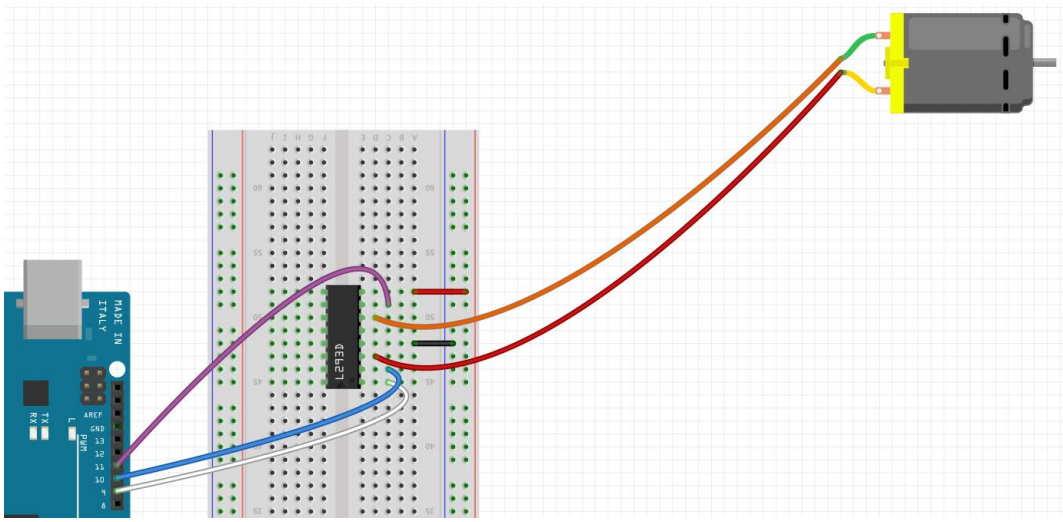


FIGURA 14: FRITZING MOTOR CC SOLO

En este micro servo, de forma similar, otra vez, se emplearon los colores habituales: 5V (rojo), GND (negro) y pulsos PWM (naranja). Este último conectado al *pin 5* de la tarjeta.

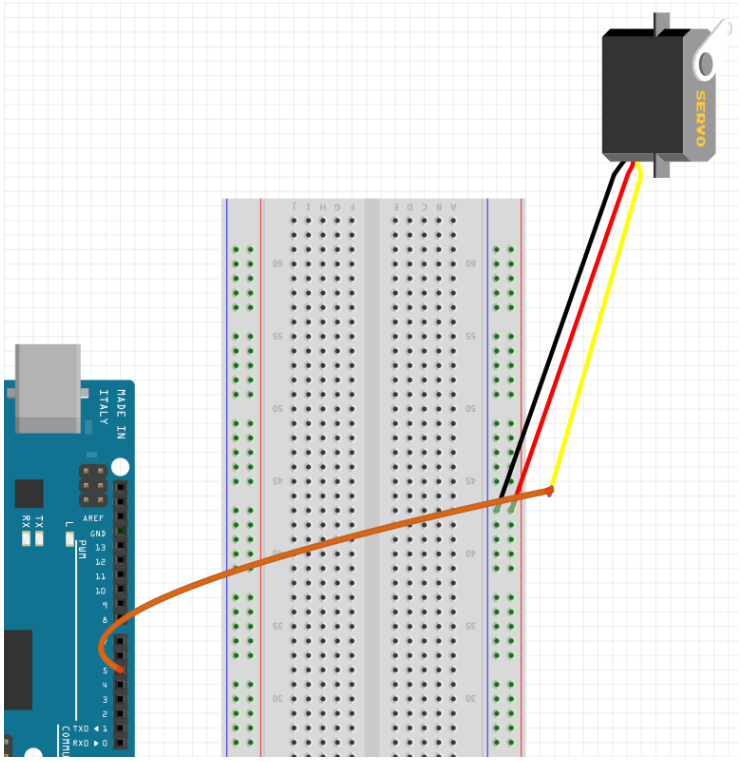


FIGURA 15: FRITZING MICRO SERVO SOLO

6.8. Maqueta construida:

En este apartado se van a mostrar unas fotos de la maqueta realizada para simular el proceso distinguiéndose, claramente, sus distintas partes. En la zona de arriba a la izquierda en la siguiente imagen se va a poder apreciar la parte que simboliza los Rosales, al igual que arriba a la derecha lo que sería el Césped:

ILUSTRACIÓN 46: MAQUETA VISTA SUPERIOR

ILUSTRACIÓN 47: MAQUETA PERSPECTIVA IZQUIERDA

ILUSTRACIÓN 48: MAQUETA PERSPECTIVA DERECHA

En cuanto a la parte de Césped se refiere, se puede apreciar los 2 sensores de humedad con las etiquetas 1 y 2, siendo el primero el del fondo y el segundo el más cercano a la tarjeta del título. Luego se distingue el sensor de temperatura C.1 a la derecha y el ventilador a la izquierda de la imagen, haciendo referencia a lo que sería el funcionamiento del aspersor:

ILUSTRACIÓN 49: ZONA CÉSPED

Dirigiéndonos a la otra zona, la de los Rosales, se va a enfocar desde adelante y desde atrás:

ILUSTRACIÓN 50: ZONA ROSALES FRONTAL

ILUSTRACIÓN 51: ZONA ROSALES TRASERA

Centrándonos en la segunda fotografía, a la izquierda abajo se situaría el sensor de humedad 3 y encima suyo el Micro Servo que interpreta la llave que accionaría el Goteo; en cambio, a la derecha arriba se encuentra el higrómetro 4 y abajo suyo el cable para medir la temperatura etiquetado con una pegatina como C.2.

Por último, la parte que engloba prácticamente todos los conexionados se encuentra metida dentro de la carcasa; por eso, para poder visualizarla echamos a un lado la tapa apreciándose lo siguiente:

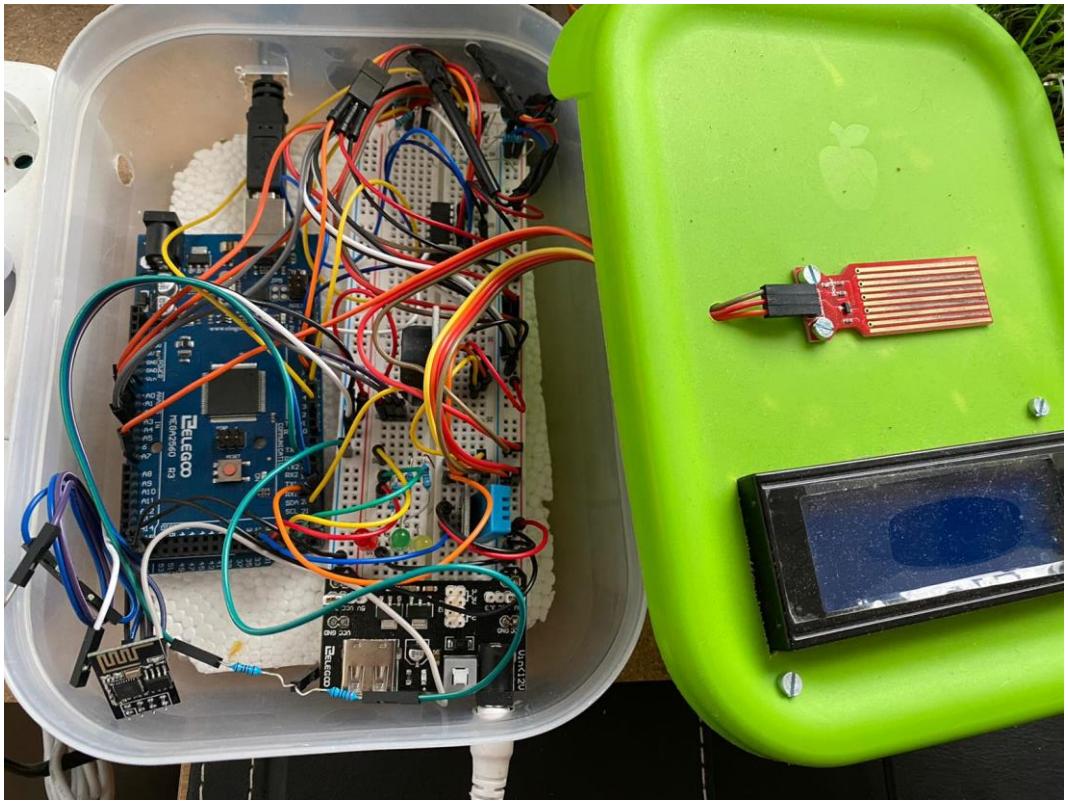


ILUSTRACIÓN 52: CARCASA ABIERTA DEL PROTOTIPO

En su parte inferior tiene un agujero realizado para la conexión del jack de la fuente de alimentación blanca al módulo de alimentación; arriba a su izquierda hay otro agujero similar para la otra fuente de alimentación negra que iría directamente conectada a la tarjeta Arduino MEGA. En la parte superior de la imagen se ha cortado una pestaña con forma cuadrada para poder pasar el puerto USB y a su derecha otra de forma rectangular para pasar todos los cables de los sensores. Justo encima de esta última hay dos taladros por donde pasan cada uno de los cables de los sensores de temperatura. En la tapa se aprecia que se ha encajonado la pantalla LCD amarrada, además, con dos tornillos y encima suyo se encuentra el sensor para la lluvia, también atornillado, que necesita un agujero para poder pasar sus tres cables.

7. Estudio económico:

7.1. Análisis de mercado:

En primera instancia, se sopesó el precio que se tenía hoy en día la instalación de un riego convencional y lo que supondría, a mayores, automatizar dicho sistema para que actúe de forma autónoma.

Comprobando el valor de las primeras marcas, como *Rain Bird*, *Regaber*, *Vyrsa*, *Irritrol*, *Ris*, *RiegoPro*, *Toro AG Irrigation*, *Viagua*, *Nelson Irrigation España*, ... se determinó lo siguiente:

> Aproximadamente, para la instalación hidráulica de una superficie de unos 20 m^2 rectangular con 2 electroválvulas y 4 aspersores, sale por un módico precio medio de unos: 120€ (sin contar el trabajo hombre). Para su control, a mayores, la compra de un programador (sin considerar la mano de obra de la instalación ni de la configuración) saldría por una media de 70-90€, sin ser autónomo; si nos interesa que sea inteligente, con sus respectivos sensores de calidad media-alta, oscilaría en torno a los 180-220€.

Por esta razón, nuestro proyecto al considerarse dentro de los del último grupo, pero de una calidad más simple, el precio final del producto ha de ser bastante menor a esos dos centenares de euros. El resultado del prototipo debería costar la mitad de dicha cantidad.

7.2. Presupuesto:

Partiendo de la base que no se buscaba un gasto mayor a 150€ en la compra de material y demás menesteres, el punto de partida del trabajo fue la compra del kit ELEGOO más completo y avanzado de iniciación compatible para proyectos con Arduino IDE MEGA 2560 que contiene la mayoría de los elementos utilizados como cables, resistencias, puerto serie USB, motores, etc. A su vez se fue pensando los sensores y módulos que se iban a necesitar para realizar el proyecto de forma adecuada; por ello se pidieron 2 cables DS18B20 (sensor de temperatura digital de acero inoxidable), un WINGONEER DS3231 AT24C32 I2C Tiny (módulo de reloj de precisión en Tiempo Real para Arduino), 5 sensores de humedad KY70 (o higrómetros) Kuman con sus respectivos puentes de cables y 3 módulos WiFi ESP8266 ESP-01S de la marca AZDelivery para Arduino y Raspberry Pi con E-book incluido. A mayores, se compraron por si fueran necesarios, en caso de avería o mal configuración de la instalación, 3 bloques de conexión Sensor de Temperatura y Humedad DHT11 de la marca AZDelivery con eBook incluido, además de 2 piezas, de la marca HiLetgo, LM2596S (convertidor de fuente de

alimentación reductor DC-DC con puerto USB), aunque finalmente no fueron necesario su uso.

Como la pantalla que venía en el conjunto ELEGOO era pequeña a mi parecer, un LCD 2x12, se decidió invertir en una pantalla más grande de 4x20, una pantalla LCD con retroiluminación azul WINGONEER IIC / I2C / TWI serie 2004. También, para comprobar que las soldaduras de fábrica venían sin fallo, se gastó lo necesario en un multímetro digital profesional Tacklife DM01M de 6000 conteos, 1000V reales RMS con rango automático del voltímetro con NCV y LIVE.

A continuación, se detalla la lista de los elementos anteriormente citados para construir la maqueta de simulación del proyecto:

COMPONENTES:	Cantidad:	Uso (nº):	Coste total con IVA:	Coste total utilizado:
Kit: Arduino MEGA 2560 R3 + cables + …	1	Sí	54,99 €	38,50 € [estimando uso del 70% del kit]
Sensor Tª. (DS18B20)	2	Sí (2)	5,99 €	5,99 €
Reloj Tiempo Real (DS3231)	1	Sí	6,49 €	6,49 €
Higrómetro (KY70)	5	Sí (4)	7,99 €	6,40 €
Sensor Tª y Hum. (DHT11)	3	No*	9,99 €	0 €
Módulo WiFi (ESP8266 ESP-01S)	3	Sí (1)	9,99 €	3,33 €
Convertidor reductor fuente de alimentación (LM2596S)	2	No	10,99 €	0 €
Pantalla LCD 4x20	1	Sí	8,99 €	8,99 €
Multímetro (DM01M)	1	Sí	23,79 €	23,79 €
TOTAL:			139,21 €	93,49 €

** De los 3 sensores DHT11 no se utilizó ninguno de los comprados, ya que en el kit Arduino venía uno que fue el que se empleó.*

TABLA 3: LISTA DE COMPONENTES Y COSTES PARTE ELECTRÓNICA

En mi caso, los costes de la parte hidráulica no se tienen en cuenta, porque ya se encuentran instalados, pero haciendo una estimación a medida, ya que serían habituales a cualquier sistema de riego, se desmembrarían tal que:

COMPONENTES:	Cantidad:	Coste unitario:	Coste total:
Electroválvula 24V	1	18€ aprox.	18€
Aspersores	4	7€ aprox.	28€
Tubos PVC	14 m.	1€/m aprox	14€
TOTAL:			60€

TABLA 4: LISTA DE COMPONENTES Y COSTES PARTE HIDRÁULICA

7.3. Viabilidad económica:

En caso de una posible comercialización, el tiempo promedio sería de unas 2 horas de la configuración hardware y software por dispositivo. Por consiguiente, si fuese necesario añadir otro dispositivo a nuestra instalación, el tiempo se vería incrementado en unos 15-20 min por dispositivo a mayores que se configure en conjunto [en el cálculo se consideraran 15 min = 0,25 h].

Como es lógico, la instalación ha de ser montada por un técnico especializado que ha de ser pagado por su mano de obra; por lo que, realizando un cálculo aproximado a partir de la media de salarios actuales, se desglosa lo siguiente basándonos en la página https://es.indeed.com/salaries/t%C3%A9cnico-electronico-Salaries y suponiendo que es salario bruto:

CONCEPTO (Trabajador + Empresa):	Coste:	Coste empresa por hora (150h* mensuales):
Salario Bruto Mensual:	1.231 €	8,2107 €
Contingencias comunes (4,7 + 23,6 %)	57,857 + 290,516 = 348,373 €	1,937 €
Desempleo (1,55 + 5,5%)	19,0805 + 67,705 = 86,7855 €	0,451 €
FOGASA (0 + 0,2 %)	2,462 €	0,016 €
Formación profesional (0,1 + 0,6 %)	1,231 + 7,386 = 8,617 €	0,05 €
TOTAL EMPRESA:	**1599,07 €**	10,665 €

* *Considerando un año 1800h teniendo ya en cuenta 3 semanas naturales de vacaciones anuales (=8x5x4x12-8x5x3).*

TABLA 5: COSTES SALARIO BRUTO TÉCNICO

Teniendo en cuenta el número de dispositivos a instalar en cierto recinto, se obtendrían los siguientes costes de mano de obra aproximadamente:

Nº dispositivos:	Horas:	Coste total (€):	Coste unitario (€/u.):
1	2	21,33	21,33
2	2,25	24	12
3	2,5	26,66	8,89
10	4,25	45,33	4,53
22	7,25	77,32	3,52
50	14,25	151,98	3,04
100	26,75	288,29	2,89

TABLA 6: COSTES SEGÚN Nº DE DISPOSITIVOS

A medida que el número de dispositivos aumenta, el coste asociado al montaje disminuye considerablemente; por lo que dichos gastos serán insignificantes cuanto mayor sea la plantación. Considerando la jornada laboral del trabajador, se podrán instalar como máximo 22 dispositivos en un mismo recinto un mismo día.

Haciendo una estimación, para los costes no considerados hasta ahora, se obtendría, a través de un sondeo, que la caja plástica donde iría el prototipo saldría por unos 1,2€/u. cada 100 unidades de producto.

Coste total del prototipo, sin incluir la instalación hidráulica:

CONCEPTO:	COSTE:
Materiales	94,6€
Mano de obra	21,33€
TOTAL:	116€

TABLA 7: COSTE TOTAL DEL PROTOTIPADO

Comparando con el análisis intensivo realizado acerca del estudio de mercado actual, se obtiene como resultado que nuestro prototipo es inferior al precio medio evaluado de sistemas de características similares.

Se ha de tener en cuenta que no se han considerado las economías de escala al haber calculado el precio para nuestra maqueta únicamente; por lo tanto, el coste material se verá disminuido considerablemente si se produce con fines comerciales en grandes cantidades. Ante esas circunstancias, al no usarse el kit completo de ELEGOO, se llegaría a un acuerdo con dicha entidad para personalizar dicho conjunto de materiales para reducir costes y de esta forma salir ambos beneficiados.

En caso de comercializar el prototipo, los costes asociados a la creación y mantenimiento de la empresa están exentos de ser evaluados en este proyecto al ser asunto, en cuestión, de un posterior estudio de viabilidad empresarial.

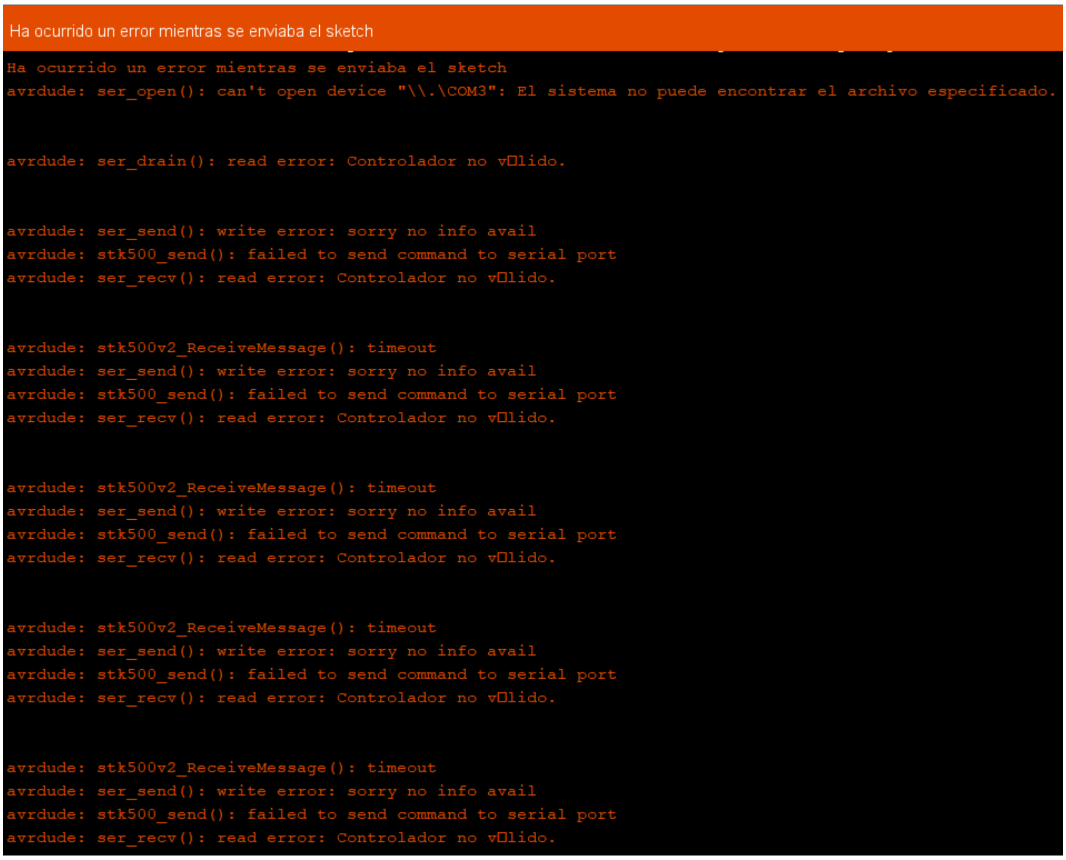
8. Pruebas y Resultados:

8.1. Errores de Hardware:

Principales errores encontrados, en cuanto a una razón física se refieren, son:

8.1.1. Subir programa sin conectar USB:

En ocasiones, al emplearse la propia entrada USB del Puerto Serie como alimentación, ya que si hay conectado una conexión externa mediante el Jack y otra a través del USB predomina la segunda, se suele desconectar si se va a estar programando durante largo tiempo para que no se encuentre en funcionamiento en todo momento. Ante esta situación, la de usar únicamente el USB como alimentación, hay veces que se termina de escribir el código y se sube el programa a la placa, teniendo desconectada dicha toma de tensión, y nos da error al no estar conectada, siendo este el siguiente:



```
Ha ocurrido un error mientras se enviaba el sketch
Ha ocurrido un error mientras se enviaba el sketch
avrdude: ser_open(): can't open device "\\.\COM3": El sistema no puede encontrar el archivo especificado.


avrdude: ser_drain(): read error: Controlador no v□lido.


avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.


avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.


avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.


avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.


avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.
```

ILUSTRACIÓN 53: ERROR SIN CONEXIÓN USB [I]

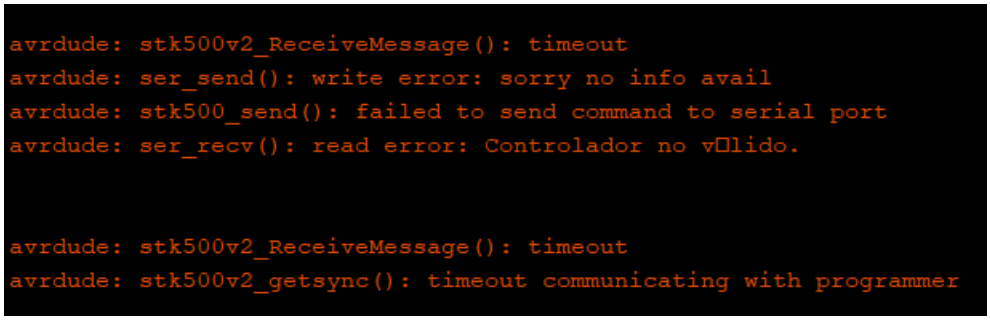


ILUSTRACIÓN 54: ERROR SIN CONEXIÓN USB [II]

Básicamente nos viene a decir, en primer lugar, que no encuentra el archivo especificado; por lo que, inmediatamente después, establece la lectura de un error que el controlador no es válido. Después, con un error de tipo escritura, no detecta envío de datos al no haber y determina que hay fallo al enviar por el Puerto Serie. A continuación, se vuelve a decir que el controlador no es válido con una anomalía de tipo lectura. Finalmente, se repite esta secuencia varias veces, pero debido a un '*timeout*' al excederse del tiempo de espera; siendo el último el de comunicarse con el programador.

8.1.2. No detectar algún componente:

Debido a la mala conexión o por una extracción involuntaria de un componente o de su conexionado, no lo detecta correctamente y, por tanto, da fallo.

Los ejemplos donde más clara se ve esta situación es en los que hay comunicación mediante el bus de datos I2C.

Hay en otras ocasiones en las que no se detecta que el elemento esté desconectado como tal, pero eso se puede percatar uno por las medidas poco usuales que se toman o que no ejecuten la acción pertinente para la cual ha sido reconfigurado, por ejemplo.

8.1.3. Flashear ESP8266:

Es uno caso particular de la última acepción del anterior apartado, muy importante por interferir en la conexión con Internet, propia de los objetos 'IoT'. Se hace necesario tener en cuenta la conexión o no del GPIO0 a la hora de subir el código o ejecutarlo respectivamente. Era un fallo común al comienzo del trabajo, ya que al realizar cualquier cambio era preciso establecer dicha conexión al subir el programa nuevo y desconectarlo para que lo ejecutara correctamente.

En caso de subir el programa teniendo enlazado el GPIO0 y ejecutarlo sin realizar la desconexión, el código se lleva a cabo de manera normal, pero el problema reside en que la configuración WiFi no llega a establecerse. Para ello hay que reiniciar la placa, apagándola, por ejemplo, y desenlazar dicho pin en nuestro caso.

8.1.4. Subiendo a la Placa I/O:

A la hora de conectar el puerto USB y, rápidamente, darle subir el programa a nuestro microprocesador, hay en ocasiones que nos sale en la Consola una extensión de texto en color naranja, con título: "***Subiendo a la placa I/O...***" en el Área de Mensajes. El contenido dice lo siguiente:

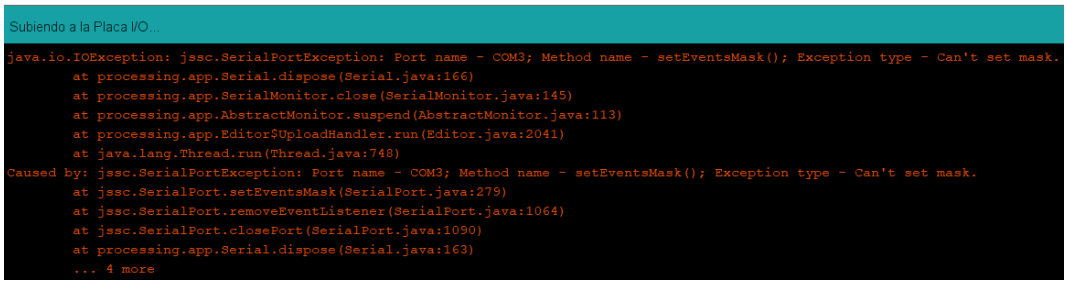


ILUSTRACIÓN 55: ERROR 'SUBIENDO A LA PLACA I/O'

Básicamente lo que nos quiere comunicar es que no puede establecer la máscara debido a que la conexión del respectivo puerto no se ha establecido correctamente. Lo que se aprecia en el *Monitor Serie* al enviar datos por el Puerto Serie se muestra a continuación:

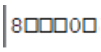


ILUSTRACIÓN 56: ERROR 'SUBIENDO A LA PLACA I/O' MONITOR SERIE (1)

Es su manera de indicar que se está produciendo un error y no se ha solucionado. Tras seguir recibiendo y enviando datos la cadena de caracteres aumenta, siguiendo una similar secuencia:

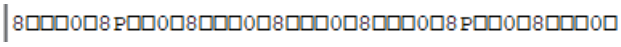


ILUSTRACIÓN 57: ERROR 'SUBIENDO A LA PLACA I/O' MONITOR SERIE (2)

8.2. Errores de Software:

Son los correspondientes a la carga del programa y a la ejecución durante el mismo, y no a una condición física del prototipo.

8.2.1. Excepciones fatales:

Uno de los problemas encontrados a la hora de la realización de las diferentes pruebas fue la aparición de anomalías, que durante se ejecutaba el código, no se mostraba primeramente nada por el Monitor Serie, pero que después de que se cargara dicha parte salía representado entre lo que se tenía que visualizar ciertos mensajes, siendo los más destacados:

Excepción fatal '0':

```
Fatal exception 0(IllegalInstructionCause):
epc1=0x4021362e, epc2=0x00000000, epc3=0x00000000, excvaddr=0x0000000c, depc=0x00000000
```

ILUSTRACIÓN 58: CAUSA DE "INSTRUCCIÓN ILEGAL"

Esta singularidad ocurre durante el arranque cuando GPIO2 se encuentra conectado a Tierra debiendo estar éste en tal situación sin conexión, o en estado 'Pull-up' (o nivel alto) que en este caso es lo mismo. Este tipo de enlace involuntario es a lo que se llama 'puntero salvaje'; la otra causa por la que puede darse es mediante BIN binarios dañados.

Excepción fatal '9':

```
Fatal exception 9(LoadStoreAlignmentCause):
epc1=0x40213620, epc2=0x00000000, epc3=0x00000000, excvaddr=0x00000022, depc=0x00000000
```

ILUSTRACIÓN 59: CAUSA DE "ALINEAMIENTO DE CARGA ALMACENADA"

Esta irregularidad ocurre cuando las direcciones de operación de lectura/escritura no se encuentran alineadas. Puede tener lugar como la del anterior por 'punteros salvajes', o porque la cache no se encuentra alineada.

Excepción fatal '28':

```
Fatal exception 28(LoadProhibitedCause):
epc1=0x4021364b, epc2=0x00000000, epc3=0x00000000, excvaddr=0x0000000c, depc=0x00000000
```

ILUSTRACIÓN 60: CAUSA DE "CARGA PROHIBIDA"

La exclusión número 28 hace referencia cuando se accede a una dirección inválida. El acceso a la caché después de que se apaga o contacto mediante un 'puntero salvaje' son las razones más frecuentes por las que puede ocurrir.

8.2.2. Port Busy:

Este desacierto se notifica cuando tenemos dos programas de Arduino abiertos a la vez, pero estando inicializado el Monitor Serie en uno de ellos y al querer abrirlo en el otro se informa lo siguiente:

```
Error abriendo puerto "COM3" (Port busy)                Copiar mensajes de error
El Sketch usa 6382 bytes (2%) del espacio de almacenamiento de programa. El máximo es 253952 bytes.
Las variables Globales usan 684 bytes (8%) de la memoria dinámica, dejando 7508 bytes para las variables locales. El máximo es 8192 bytes.
Ha ocurrido un error mientras se enviaba el sketch
avrdude: ser_open(): can't open device "\\.\COM3": Acceso denegado.


avrdude: ser_drain(): read error: Controlador no v□lido.


avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.
```

ILUSTRACIÓN 61: PORT BUSY [I]

```
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.


avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.


avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.


avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.


avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: ser_send(): write error: sorry no info avail
avrdude: stk500_send(): failed to send command to serial port
avrdude: ser_recv(): read error: Controlador no v□lido.


avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_getsync(): timeout communicating with programmer
Error abriendo puerto "COM3" (Port busy)
```

ILUSTRACIÓN 62: PORT BUSY [II]

A fin de cuentas, lo que nos quiere decir esta notificación es que nos deniega el acceso al no poder abrir el puerto, siendo el resto del proceso similar al error “Subir programa sin conectar USB” decretando finalmente que existe una inexactitud al intentar abrir el puerto.

8.2.3. 'Busy':

Este tipo de equivocación surge, principalmente, cuando ya se está acabando de programar el código, por contener gran cantidad de contenido y, por tanto, ocupar más memoria. La palabra '***Busy***' mostrada por el Monitor Serie quiere decir que el comando AT previo no ha terminado de ejecutarse y este puede ser precedido por un '***p...***' o un '***s...***'. El primer suceso tiene lugar porque el comando se está ejecutando todavía; en cambio, si sale el segundo indica que hay envío de datos de por medio en ese instante.

```
AT+CWJAP="MiFibra-F84E","[redacted]"
WIFI CONNECTED
AT+CIFSR
busy p...
```

ILUSTRACIÓN 63: EJEMPLO DE 'BUSY P...' AL ESTAR CONECTANDO EL WIFI

```
,355:GET / HTTP/1.1
Host: 192.168b>4: </b>60%</td><
busy s...

Recv 1500 bytes
SEND FAIL
/tIndustriales src=https://www.eii.uva.es/escuela/logos/logotipo_neg.jpg width=240 height=150></div>AT+CIPCLOSE=0
0,CLOSED

OK
0,CONNECT

+IPD,0,355:GET / HTTP/1.1
Host: 192.168.1.108
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101 Firefox/72.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

ILUSTRACIÓN 64: EJEMPLO DE 'BUSY S...' AL ESTAR ENVIANDO LA PÁG. WEB GENERADA

ILUSTRACIÓN 65: RESULTADO DEL EJEMPLO ANTERIOR DE 'BUSY P...' Y 'BUSY S...'

Otra impropiedad similar a ésta es si nos sale directamente la palabra 'ERROR', pero en esta situación es debido a que no se ha realizado correctamente la configuración del comando AT correspondiente. Un ejemplo sería el del comando AT+CWMODE al no especificar cuál de los 3 modos seleccionar:

```
AT+CWMODE=?
+CWMODE:(1-3)

OK
AT+CWMODE

ERROR
AT+CWMODE=3

OK
```

ILUSTRACIÓN 66: EJEMPLO DE 'ERROR' CON EL COMANDO AT+CWMODE

8.2.4. Fragmentación de memoria:

Se produce, en la mayoría de las ocasiones, cuando se concatenan grandes cadenas de caracteres a enviar repetidas veces. Si internamente ha habido una asignación errónea, esto puede dañar los datos, por copiarse mal la cadena, y bloqueándose finalmente nuestro módulo ESP.

El emplear el operador "+()" nos permite concatenar, y si se emplea varias veces seguidas pueden fallar como se ha explicado anteriormente dando lugar a 'fallos silenciosos'. Esto se debe a una falta de asignación correcta de memoria por la causa de haberse quedado sin pila, aunque aún haya suficiente memoria en la pila total; es decir, es necesario encontrar un hueco acorde con el tamaño que se solicita.

Este tipo de fallos requieren del cumplimiento de interfaces específicos, por un lado, y de que el código de asignación no está accesible en algunas librerías para modificarse, por otro lado; todo ello en cuanto al módulo WiFi se refiere.

8.2.5. Servidor:

A la hora de crear nuestro servidor, se ha de tener cuidado cuán grande se quiera que sea la configuración de éste en cuanto a memoria se refiere. Se ha cometido varias veces el error que consiste en un bloqueo causado por quedarse sin pila de almacenamiento retentiva; es decir, sin memoria.

El tipo de error citado es un ejemplo concreto del anterior. Los objetos de tipo String no son capaces de devolver un error de "la vieja escuela", es decir, no se puede, a nivel de aplicación de usuario, controlar esos errores. Al mismo tiempo el empleo de errores no permite adentrarse en el código de forma subyacente, porque las implementaciones estándar se fundamentan en excepciones y ellas no son detectadas por el ESP.

8.3. Pruebas:

Como ya se dijo anteriormente, el software del proyecto se fue configurando componente a componente, elemento a elemento, para posteriormente juntarlo todo en un único programa dotado de las funciones deseadas para que lo realice de una manera determinada. Por ello, una vez agrupados los distintos subconjuntos, se fue desarrollando y adaptando la idea en las diferentes versiones realizadas, guardando así el progreso, siendo las más destacadas las siguientes:

La primera versión se configuró las condiciones de testeo, pero teniendo el resto de los bloques de la programación fuera de ellas para poder realizar las pruebas pertinentes hasta pulir el código. Es donde se conformó la manera de crear y enviar el código html para generar la página web mediante la función pertinente.

En la segunda, a mayores de la anterior, se procuró generar una variable que detectara, mediante texto, que se quería testear, porque obtendría el resultado tras leerlo de la propia página web si estaba en 'ON' u 'OFF'. De esta forma, se tuvo que construir una función, en vez de enviar, que es la que se sigue usando, para recibir esa información. Finalmente, ese método se descartó por la ineficacia de su lentitud.

La tercera revisión se caracterizó por intentar, mediante la biblioteca del ESP8266 "WiFi", obtener el vínculo con Internet evitando el uso de 'Comandos AT', ya fuera mediante protocolo UDP o TCP, entre otros. El problema de hacerlo de esta manera es que había que implantar al ESP como la placa base, obviando al Arduino MEGA y eso no nos interesaba. Además, se implementó la "***Configuración inicial***" para realizar un primer testeo de seguridad. También se modeló el envío de un formulario para obtener el 'Historial' de los datos y una casilla para la selección de los mismos, pero no se desarrolló con éxito.

En la cuarta se decidió realizar la recreación de los aspersores y goteos mediante los motores y, por eso, fueron añadidos aquí. Su conteo se efectuaba mediante el reinicio de la función 'millis', pero esto suponía un problema, porque interfería a la hora de transmitir la página web y por eso se

descartó. En cambio, aquí también fue donde se configuró las portadas implementando la función '*generarChar*' entre otros caracteres. Al mismo tiempo se consideró la posibilidad de lluvia en el riego.

La quinta prueba, únicamente, se procuró generar el título de la página web en el 'Setup' en vez de en el 'Loop' al ser contenido estático y evitando, así, una carga repetitiva de trabajo al microprocesador y errores silenciosos. Para intentar solucionar el contratiempo de la interposición de la función 'millis', se procuró cambia la forma de conteo, produciéndolo y reiniciándolo con la función 'micros' que es igual que la anterior, pero nos muestra los microsegundos, no los milisegundos. En esta situación ocurría una anomalía al no reiniciarse a cero debido a los registros base.

En la penúltima versión se añadió un aviso de "*¡Warning!*", que mientras no se detectará el RTC y siguiera sin llevarse una acción a cabo, estuviera mostrándose el mensaje cada 2 segundos. Esto fue debido a que se dejó medio configurado (indicándose únicamente como error), pero sin establecerse esa realimentación que obligara a tener un RTC disponible para poderse ejecutar el programa y, por tanto, su correcto funcionamiento; ya que sin él no se sabría si la hora actual se encuentra dentro de las franjas correspondientes.

La última prueba, antes de terminar de conformar y adaptar el 'Resultado final', fue la de, que automáticamente, se configurara la franja horaria para verano, primavera-otoño o invierno según los solsticios y equinoccios. Esta distinción es importante por las horas de sol disponibles en uno y en otros periodos, al igual que la temperatura y humedad ambientes. Se implementó en una pestaña externa llamada "*EstacionHoraria.hpp*".

8.4. 'Resultado final':

Este capítulo que nos precede va a desarrollarse de forma detallada la resolución finalmente implementada y cómo se expone en caso de ejecutar el código a manera de demostración.

Lo primero es establecer la velocidad, en baudios, a lo cual queremos que funcione el procesamiento de la placa; es decir, el número de unidades de señal que haya por segundo. No ha de confundirse velocidad de baudios con tasa de bits, ya que esta última se refiere al número de bits transmitidos por unidad de tiempo; en cambio, la primera indica el número de símbolos pudiendo comprender 1 o más bits.

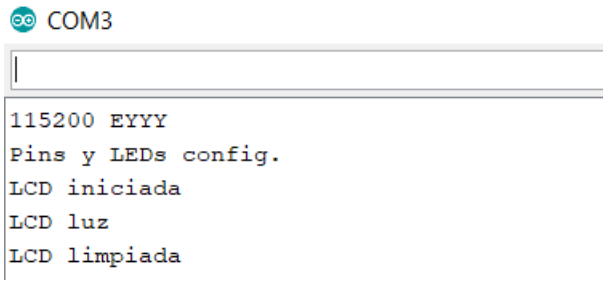


ILUSTRACIÓN 67: PUERTO SERIE 'RESULTADO FINAL' [I]

Acto seguido se configuran los distintos pines y LEDs a utilizar. Al mismo tiempo se representarán por la LCD las diferentes ***portadas*** creadas:

ILUSTRACIÓN 68: PORTADA LCD LOGO 'EII'

ILUSTRACIÓN 69: PORTADA LCD LOGO 'UVA'

Para establecer la conexión WiFi se modelan los '***Comandos AT***' de la siguiente manera, mientras se muestran estos dos mensajes por pantalla y finalmente lucen los 3 LEDs a la vez para indicarlo:

```
AT

OK
AT+GMR
AT version:1.2.0.0(Jul  1 2016 20:04:45)
SDK version:1.5.4.1(39cb9a32)
v1.0.0
Mar 11 2018 18:27:31
OK
AT+CWMODE=3

OK
AT+CIPMUX=1

OK
AT+CIPSERVER=1,80

OK
AT+CWJAP="MiFibra-F84E","[redacted]"
WIFI DISCONNECT
WIFI CONNECTED
WIFI GOT IP

OK
AT+CIFSR
+CIFSR:APIP,"192.168.4.1"
+CIFSR:APMAC,"a6:cf:12:f4:ae:36"
+CIFSR:STAIP,"192.168.43.63"
+CIFSR:STAMAC,"a4:cf:12:f4:ae:36"

OK
```

ILUSTRACIÓN 70: PUERTO SERIE 'RESULTADO FINAL' [II]

ILUSTRACIÓN 71: LCD 'COMANDOS AT'

ILUSTRACIÓN 72: LCD RED WIFI

Tras fijar la configuración del 'Setup', en primera instancia, está instaurado que cumpla un ***primer testeo de seguridad*** para comprobar que los sensores desempeñan su función correctamente y no hay fallo en sus conexionados.

```
-·-30/11/2020 | 17:14:7-·-

      17 horas
#*# Estación: OTONO #*#

  Temperatura: 24.80°C
  Humedad: 40.00%

< Temperatura1= 23.19 °C >
< Temperatura2= 22.25 °C >

   · SensorH1: Mojado
   · SensorH2: Humedo
   · SensorH3: Mojado
   · SensorH4: Mojado

   # Hum1: 494 >> 52%
   # Hum2: 707 >> 31%
   # Hum3: 438 >> 58%
   # Hum4: 388 >> 62%

- - - - - - - - - - - - - - - -

  //Nivel lluvia: 5\\
   Aspersores = 'OFF'
   Goteo = 'OFF'
```

ILUSTRACIÓN 73: PUERTO SERIE 'RESULTADO FINAL' [III]

Todo lo que se muestra por pantalla, en relación con la anterior imagen, es parte del '***Loop***' visualizándose en el siguiente orden:

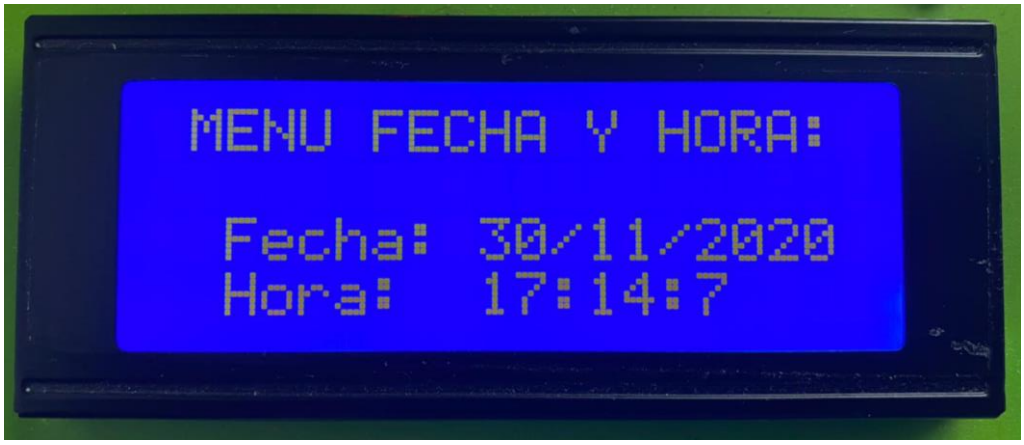


ILUSTRACIÓN 74: MENÚ FECHA Y HORA LCD

ILUSTRACIÓN 75: MENÚ FECHA Y HORA LCD FIJANDO HORA ACTUAL PARA EL TESTEO

El inicio de la "*Configuración inicial*" indicando el primer testeo de seguridad, se ve de la siguiente forma:

ILUSTRACIÓN 76: LCD CONFIGURACIÓN INICIAL

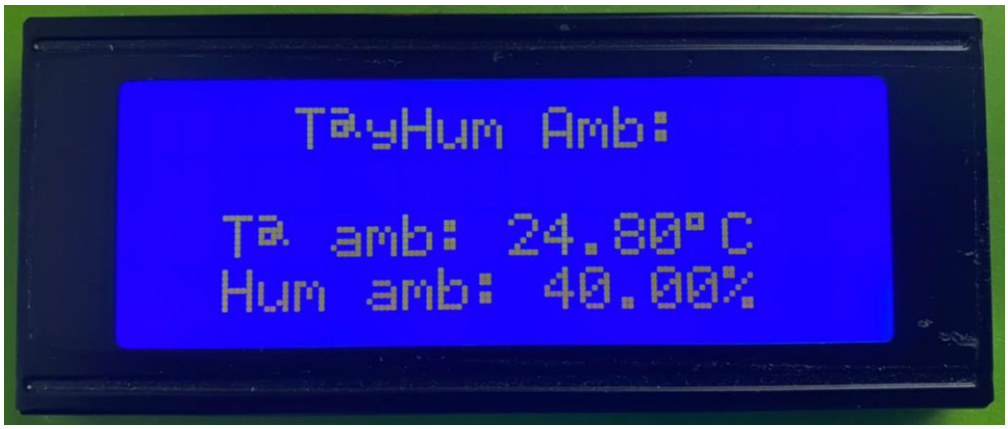


ILUSTRACIÓN 77: LCD MENÚ Tª Y HUM AMBIENTES

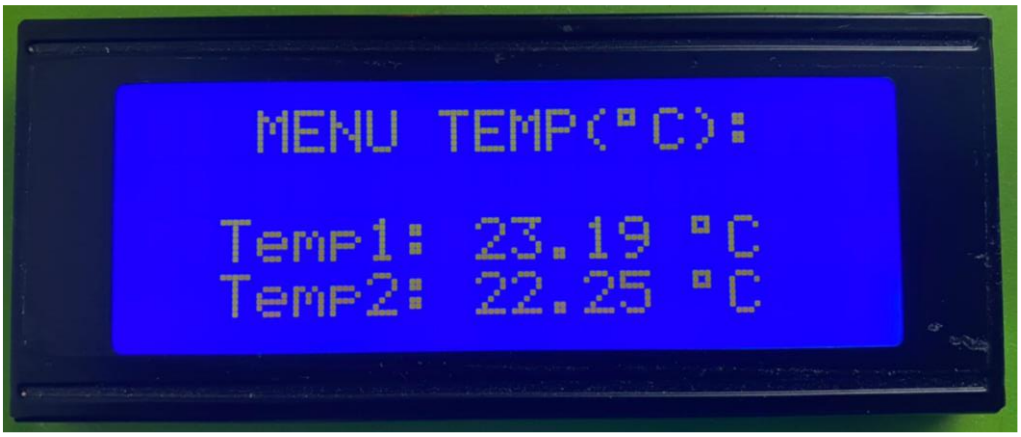


ILUSTRACIÓN 78: LCD MENÚ Tª

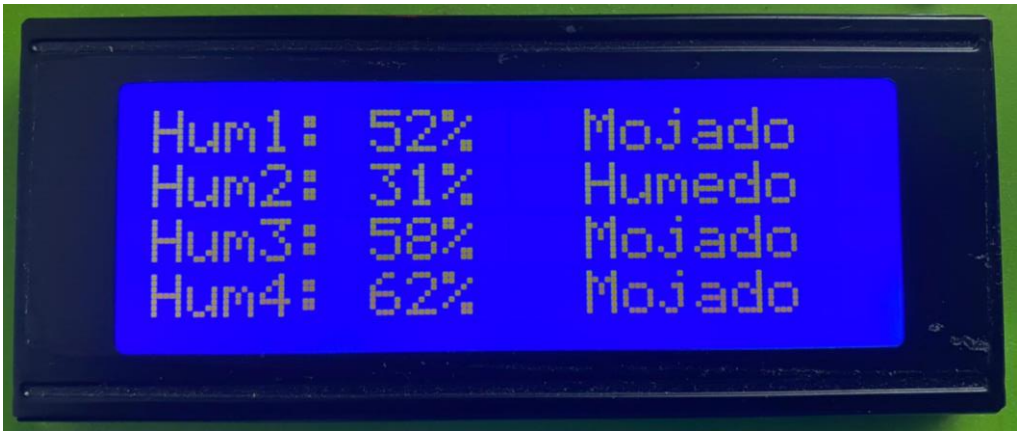


ILUSTRACIÓN 79: LCD MENÚ HUM

ILUSTRACIÓN 80: LCD MENÚ RIEGO

En caso de encontrarse en 'ON' ambas zonas, la de Césped (Aspersores) y la de los Rosales (Goteo), se plasmarían en la segunda fila y tercera respectivamente, en vez de en la primera y la cuarta. Seguido de un mensaje de que se encuentra regando cierta parte, reescribiendo dichas líneas correspondientes. A continuación, se muestra "***Regando Césped***":

ILUSTRACIÓN 81: LCD REGANDO CÉSPED

ILUSTRACIÓN 82: MENÚ ACTUALIZACIÓN LCD

Después al localizarse fuera de las franjas horarias de riego, preestablecidas dependiendo de la época del año, se mantiene entre el Menú de Fecha y Hora y el de Actualización de datos; comprobándose primero si se ha enviado petición de Testeo al recargar la página. Si no hay solicitud, se aprecia lo siguiente por el Puerto Serie:

```
-·-30/11/2020 | 17:14:33-·-

      17 horas
#*# Estación: OTONO #*#


-·-30/11/2020 | 17:14:41-·-

      17 horas
#*# Estación: OTONO #*#


-·-30/11/2020 | 17:14:49-·-
```

ILUSTRACIÓN 83: PUERTO SERIE 'RESULTADO FINAL' [III]

Por el contrario, si ha tenido lugar esa demanda de ***testear***, se dibuja en la LCD:

ILUSTRACIÓN 84: LCD TESTEO

En caso de encontrarnos ***dentro de la franja*** de testeo se visualizaría lo siguiente:

ILUSTRACIÓN 85: LCD 'ESTOY EN LA FRANJA'

```
-··-30/11/2020 | 21:44:15-·-

      21 horas
#*# Estación: OTONO #*#

(| Estoy en la franja |)

  Temperatura: 27.50°C
  Humedad: 36.00%
```

ILUSTRACIÓN 86: PUERTO SERIE 'RESULTADO FINAL' FRANJA

Saliendo por la ***página web*** el resultado de esta última acción de los sensores:

ILUSTRACIÓN 87: PÁGINA WEB 'RESULTADO FINAL'

8.5. Mejoras futuras:

Para alargar la vida útil de los componentes eléctricos, al estar en contacto con el agua, sería necesario testear únicamente llegadas las franjas horarias establecidas de forma periódica para no cometer, en un futuro, fallos ni falsos positivos por deterioro del material por corrosión. Esto quiere decir que, en vez de estar en tensión, en todo momento, los componentes únicamente lo estarían cuando se solicitara, que en esta acepción es durante toda la franja horaria pertinente.

Otra forma de afrontar el problema anterior, para cualquier hora del día, sería cuando se envíe la petición de testeo mediante un botón virtual en la página web para que sondee solamente en ese momento (intentando evitar el uso de Relés si eso fuera posible, para ahorrar en material). Complementándose, se haría necesario establecer que, durante los tramos horarios, el testeo automático se realizara especialmente cada 5 minutos, por ejemplo, para que no estén activados en todo momento, pero que, si es solicitado mediante el botón virtual, lo haga en ese preciso instante.

ILUSTRACIÓN 88: SERVIDOR ENVÍO DATOS (ON-OFF)

Establecer un historial semanal de los datos que se quieran almacenar empleando una base de datos, guardadas en un archivo .txt, por ejemplo, para que no ocupe mucha memoria, al ser limitada en nuestro dispositivo ESP-01S.

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *
 *  ___CONTROL DE UN SISTEMA INTELIGENTE DE REGADÍO___      *
 *                      ~~~ TFG ~~~                         *
 *    Universdidad de Valladolid - Esc. Ing. Indus.         *
 *      Grado en Ing. Electrónica Indus. y Autom.           *
 *                                                          *
 *            Guillermo Moreno Heranangómez                 *
 *                                                          *
 * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

- - - - - - - - - - - - - - - - - MEMORIA HISTORIAL: - - - - - - - - - - - - - - - - -

_____________________________________________________________________________________

                                        >>> DÍA: 22/10/2020 <<<
                    /// HORA: 18:51 \\\
Temp_Amb: 25.30ºC       |   Temp1: 22.00ºC   |   Hum1: 25%   |   Hum3: 51%
Hum_Amb:  66.00%        |   Temp2: 22.50ºC   |   Hum2: 37%   |   Hum4: 44%
_____________________________________________________________________________________

                                        >>> DÍA: 05/11/2020 <<<
                    /// HORA: 13:01 \\\
Temp_Amb: 26.40ºC       |   Temp1: 23.60ºC   |   Hum1: 55%   |   Hum3: 68%
Hum_Amb:  48.00%        |   Temp2: 22.80ºC   |   Hum2: 52%   |   Hum4: 63%
_____________________________________________________________________________________

                                        >>> DÍA: 09/11/2020 <<<
                    /// HORA: 12:46 \\\
Temp_Amb: 24.60ºC       |   Temp1: 21.70ºC   |   Hum1: 41%   |   Hum3: 14%
Hum_Amb:  33.00%        |   Temp2: 22.20ºC   |   Hum2: 34%   |   Hum4: 20%

                    /// HORA: 18:34 \\\
Temp_Amb: 25.50ºC       |   Temp1: 20.90ºC   |   Hum1: 59%   |   Hum3: 48%
Hum_Amb:  37.00%        |   Temp2: 21.60ºC   |   Hum2: 57%   |   Hum4: 46%
_____________________________________________________________________________________

                                        >>> DÍA: 16/11/2020 <<<
                    /// HORA: 11:27 \\\
Temp_Amb: 23.20ºC       |   Temp1: 21.30ºC   |   Hum1: 13%   |   Hum3: 56%
Hum_Amb:  62.00%        |   Temp2: 20.70ºC   |   Hum2: 44%   |   Hum4: 53%
```

FIGURA 16: DOCUMENTO .TXT - DATOS GUARDADOS 'BLOCK DE NOTAS'

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *
 *  ___CONTROL DE UN SISTEMA INTELIGENTE DE REGADÍO___      *
 *                      ~~~ TFG ~~~                         *
 *    Universdidad de Valladolid - Esc. Ing. Indus.         *
 *      Grado en Ing. Electrónica Indus. y Autom.           *
 *                                                          *
 *            Guillermo Moreno Heranangómez                 *
 *                                                          *
 * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

- - - - - - - - - - - - - - MEMORIA HISTORIAL: - - - - - - - - - - - - - -

_____________________________________________________________________________

                        >>> DÍA: 22/10/2020 <<<
                /// HORA: 18:51 \\\
Temp_Amb: 25.30°C   |   Temp1: 22.00°C   |   Hum1: 25%   |   Hum3: 51%
Hum_Amb:  66.00%    |   Temp2: 22.50°C   |   Hum2: 37%   |   Hum4: 44%
_____________________________________________________________________________

                        >>> DÍA: 05/11/2020 <<<
                /// HORA: 13:01 \\\
Temp_Amb: 26.40°C   |   Temp1: 23.60°C   |   Hum1: 55%   |   Hum3: 68%
Hum_Amb:  48.00%    |   Temp2: 22.80°C   |   Hum2: 52%   |   Hum4: 63%
_____________________________________________________________________________

                        >>> DÍA: 09/11/2020 <<<
                /// HORA: 12:46 \\\
Temp_Amb: 24.60°C   |   Temp1: 21.70°C   |   Hum1: 41%   |   Hum3: 14%
Hum_Amb:  33.00%    |   Temp2: 22.20°C   |   Hum2: 34%   |   Hum4: 20%

                /// HORA: 18:34 \\\
Temp_Amb: 25.50°C   |   Temp1: 20.90°C   |   Hum1: 59%   |   Hum3: 48%
Hum_Amb:  37.00%    |   Temp2: 21.60°C   |   Hum2: 57%   |   Hum4: 46%
_____________________________________________________________________________

                        >>> DÍA: 16/11/2020 <<<
                /// HORA: 11:27 \\\
Temp_Amb: 23.20°C   |   Temp1: 21.30°C   |   Hum1: 13%   |   Hum3: 56%
Hum_Amb:  62.00%    |   Temp2: 20.70°C   |   Hum2: 44%   |   Hum4: 53%
```

FIGURA 17: DOCUMENTO .TXT - DATOS GUARDADOS 'NOTEPAD++'

Optimizando el punto anterior, de forma inclusiva, se establecería mediante una 'checkbox' en la página web principal los datos concretos que se quisieran guardar, desechando así los no necesarios o los que son muy

similares por periodicidad; pudiendo así, a su vez, aumentar la temporalidad de guardado pasando de ser semanal a mensual, por ejemplo.

ILUSTRACIÓN 89: SERVIDOR FINAL CON HISTORIAL

A mayores, se podría establecer otra página web secundaria titulada "*Historial*" para visualizar dicha información guardada en cualquier momento y desde cualquier instante.

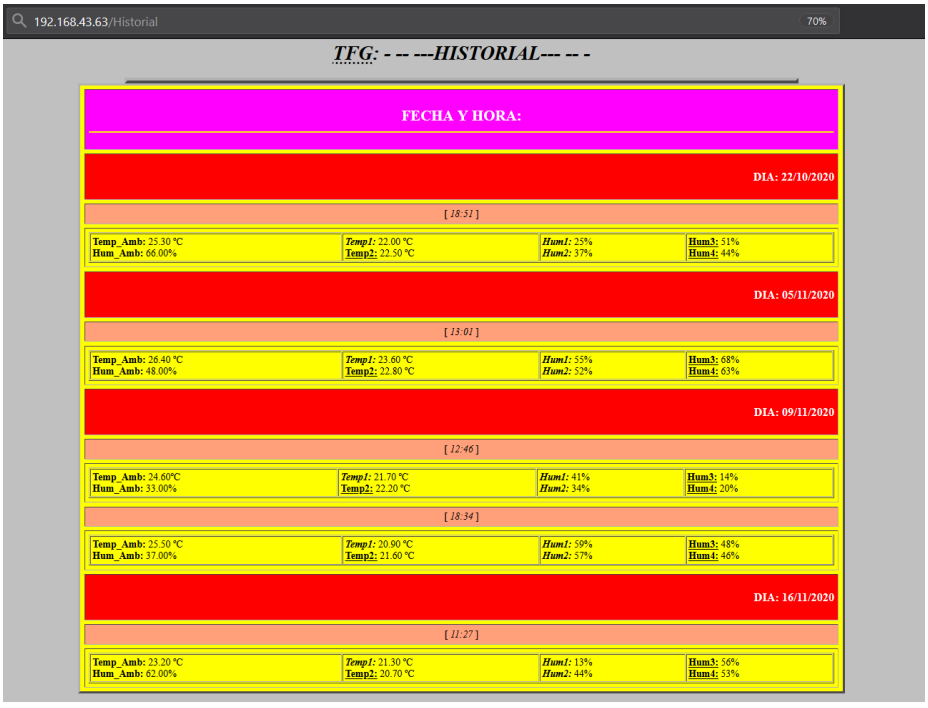


ILUSTRACIÓN 90: SERVIDOR FINAL 'HISTORIAL'

9. Conclusiones:

En primer lugar, se hace necesario conocer muy detenidamente el entorno, la manera cómo se va a trabajar, de qué forma se va a enfocar el proyecto, el material del cuál se dispone y los medios que se van a necesitar. Teniendo determinado estos aspectos resultará más sencillo la realización del proyecto y el evitar toparnos con incertidumbres a posteriori.

El saber utilizar las herramientas de desarrollo y tener práctica con ellas nos agilizará el proceso de realización y seremos más eficaces; por esa razón, es muy aconsejable el trabajar en espacios anteriormente empleados con regularidad para sacar todo nuestro potencial y evitar fallos. Aunque el emplear nuevos métodos y formas de realizar un trabajo también sirve como manera de aprendizaje autónoma que, por otra parte, estimula el desarrollo de adaptación.

También, he de decir, es sugerible comprender de forma global el cómo funciona el campo que vamos a tratar y no solamente el ámbito, o la parte, en cuestión que se va a abordar, porque tener una visión colectiva de todos los aspectos a tratar, en su forma más plena, enriquece el entendimiento y mejora a la hora de su explicación.

Además, la primera idea es la que nos introduce en el tema, no teniendo por qué ser ésta la mejor opción a desarrollar, siendo comprensible un cambio o modificación en el transcurso de un trabajo que mejorará la puesta a punto del proyecto. De este modo, no ha de haber reparo en cambiar, quitar o eliminar cualquier parte, siempre y cuando sirva para progresar o nos quedemos estancados; al igual que, si es necesario, pedir ayuda en cualquier momento, ya que, por suerte o por desgracia, 'no nacimos aprendidos'.

Resumiendo, se podría decir, que he adquirido las competencias de:

1. Familiarizarme, más a fondo, con el entorno IDE de Arduino.
2. Ciertos conocimientos de botánica.
3. Profundizar en la evaluación de extraer la información pertinente de los componentes en los diferentes 'Datasheets', al igual que de los distintos elementos que los componen.
4. Generar páginas web mediante código html, css, json y websockets.
5. Inicializar mi aprendizaje con dichos lenguajes de programación.
6. Configuración y análisis del módulo ESP8266.
7. Ampliar conocimientos acerca de la herramienta Fritzing.
8. Adaptación de la planificación del proyecto ante incertidumbres.
9. Resolución de problemas ante limitaciones de cualquier tipo.
10. Evaluación del proyecto mediante diagramas.

10. Referencias y Bibliografía:

Partiendo como base para el prototipado se empleó el estudio de los apuntes correspondientes a las asignaturas '*Mecatrónica*' y '*Control y Comunicaciones*'. El resto de información necesaria fue buscada, sobre todo, a través de Internet siendo el resultado bibliográfico el siguiente:

Referencias:

[1] Arroyo, M. M. (2017, 11 septiembre). *El Riego Inteligente en la agricultura*. iAgua. https://www.iagua.es/blogs/manuel-martin-arroyo/riego-inteligente-agricultura

[2] Arroyo, M. M. (2017, 11 septiembre). *El Riego Inteligente en la agricultura*. iAgua. https://www.iagua.es/blogs/manuel-martin-arroyo/riego-inteligente-agricultura

[3] López, R. (2020, 28 octubre). *El impacto de la Industria 4.0 en el sector del agua*. iAgua. https://www.iagua.es/blogs/ramon-lopez/impacto-industria-40-sector-agua

[4] Arduino. (s. f.). *Arduino Mega 2560 Rev3 | Arduino Official Store*. Recuperado 2 de junio de 2020, de https://store.arduino.cc/arduino-mega-2560-rev3

[5] Gómez, E. (2015, mayo). *Comando-AT*. Rincón Ingenieril. https://www.rinconingenieril.es/wp-content/uploads/2015/05/Comando-AT.pdf

[6] Espressif. (2016). *ESP8266 Reset Causes and Common Fatal Exception Causes*. https://www.espressif.com/sites/default/files/documentation/esp8266_reset_causes_and_common_fatal_exception_causes_en.pdf

[7] Llamas, L. (2015, 28 octubre). *Esquema de patillaje (pinout) de Arduino Uno, Nano, Mini y Mega*. Luis Llamas. https://www.luisllamas.es/esquema-de-patillaje-de-arduino-pinout/

Bibliografía:

1&1 IONOS España S.L.U. (2020, 23 marzo). *URI (identificador de recursos uniformes)*. IONOS Digitalguide. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/uri-identificador-de-recursos-uniformes/ [Consultado: 17/09/2020]

A. (2005). *Riego del césped*. Césped.es. https://www.cesped.es/riego-del-cesped/ [Consultado: 25/05/2020]

A. (2020, 16 abril). *Cómo conectar el módulo ESP8266 a WiFi*. Proyectos con Arduino. https://proyectosconarduino.com/curso/como-conectar-modulo-esp8266-a-wifi/ [Consultado: 17/08/2020]

Amanecer y atardecer en España. (s. f.). DatosMundial.com. Recuperado 26 de noviembre de 2020, de https://www.datosmundial.com/europa/espana/puesta-del-sol.php

Alex7 Tutoriales. (2018, 16 junio). *No Muestra Caracteres en LCD 20x4 Comunicación I2C con ARDUINO (SOLUCIÓN)* [Vídeo]. YouTube. https://www.youtube.com/watch?v=jOe4WFksBSA [Consultado: 09/06/2020]

Alonso, S., de Castro, M. y Martín-Vide, J. (2004). *Impactos del Cambio Climático en España – 1. El clima de España: Pasado, Presente y escenarios de clima para el siglo XXI*. CSIC (Consejo Superior de Investigaciones Científicas). https://digital.csic.es/bitstream/10261/35782/1/090471228000f24f_tcm7-12417.pdf [Consultado: 25/05/2020]

Bonad Gascon, S. y Sala Galan, J. (1970). *El Ciruelo*. Ministerio de Agricultura, Pesca y Alimentación Gobierno de España. https://www.mapa.gob.es/ministerio/pags/biblioteca/hojas/hd_1970_19-20.pdf [Consultado: 25/05/2020]

Cambatronics Online. (2017, 16 mayo). *ESP8266 #2: Conexionado y puesta en marcha.* [Vídeo]. YouTube. https://www.youtube.com/watch?v=yEGaafxRi6A [Consultado: 14/07/2020]

Campbell, S. (2015, 13 noviembre). *How to Set Up the DHT11 Humidity Sensor on an Arduino*. Circuit Basics. https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/ [Consultado: 15/06/2020]

colaboradores de Wikipedia. (2020, 27 octubre). *Sistema embebido*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Sistema_embebido [Consultado: 10/09/2020]

Comandos AT - 3Cu Electrónica. (s. f.). 3Cu Electrónica. Recuperado 14 de octubre de 2020, de https://sites.google.com/site/3cuelectronica/home/comandos-at-1 [Consultado: 14/10/2020]

Cómo elegir programadores de riego. (2019, 5 noviembre). LEROY MERLIN. https://www.leroymerlin.es/jardin-terrazas/riego/programadores-riego/como-elegir-programadores-de-riego [Consultado: 4/11/2020]

CSS básico. (2020, 10 noviembre). Documentación web de MDN. https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/CSS_basics [Consultado: 26/08/2020]

Danés, J. (2020, 27 enero). *La cotización por contingencias comunes*. Loentiendo. https://loentiendo.com/cotizacion-contingencias-comunes/ [Consultado: 07/10/2020]

designthemes. (2019, 22 octubre). *Servidor Web con WIFI ESP8266*. PROMETEC. https://www.prometec.net/servidor-web-esp8266/ [Consultado: 09/10/2020]

E. (2019, diciembre 11). *Arduino: Usando la función millis() en lugar de delay()* | *Robots Didácticos*. Robots Didácticos. http://robots-argentina.com.ar/didactica/arduino-usando-la-funcion-millis-en-lugar-de-delay/ [Consultado: 25/09/2020]

educ8s.tv. (2017, 6 mayo). *Arduino Tutorial: 20x4 I2C Character LCD display with Arduino Uno from Banggood.com* [Vídeo]. YouTube. https://www.youtube.com/watch?v=F9IVtKa8C7Q [Consultado: 10/06/2020]

El profe García. (2016, 18 mayo). ✔ *WIFI Modulo ESP8266 Conectarse y enviar datos por Internet ✔Comandos AT✔* [Vídeo]. YouTube. https://www.youtube.com/watch?v=7gXcTBHLCRc [Consultado: 27/05/2020]

El profe García. (2017, 28 junio). ✔ *Que es un Reloj de Tiempo Real (RTC) como se usa?* [Vídeo]. YouTube. https://www.youtube.com/watch?v=tYuSah73zxY [Consultado: 04/06/2020]

El profe García. (2019, 27 agosto). ✔ *Sin Errores Conectar i2C al LCD y Arduino, encontrando el código Hex correcto* [Vídeo]. YouTube. https://www.youtube.com/watch?v=M_xxEkMkN74 [Consultado: 14/06/2020]

Engineers, L. M. (2019a, diciembre 13). *How Water Level Sensor Works and Interface it with Arduino*. Last Minute Engineers. https://lastminuteengineers.com/water-level-sensor-arduino-tutorial/ [Consultado: 24/10/2020]

Engineers, L. M. (2019, 13 diciembre). *Interface DS3231 Precision RTC Module with Arduino*. Last Minute Engineers. https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/ [Consultado: 24/09/2020]

Espressif. (2016). *ESP8266 Reset Causes and Common Fatal Exception Causes*. https://www.espressif.com/sites/default/files/documentation/esp8266_reset_causes_and_common_fatal_exception_causes_en.pdf [Consultado: 13/11/2020]

ESP8266 - AT Command Reference · room-15. (2015, 26 marzo). room-15. https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/ [Consultado: 09/10/2020]

ESP8266 - NURDspace. (s. f.). NURDs. Recuperado 27 de julio de 2020, de https://nurdspace.nl/ESP8266 [Consultado: 27/07/2020]

Fernandez, I. (2019, 3 noviembre). *Sensor de humedad para tierra: HW-080 – BETA Weblog*. BETAweblog. http://irisfernandez.com.ar/betaweblog/index.php/2019/11/03/sensor-de-humedad-para-tierra-hw-080/ [Consultado: 31/07/2020]

Forgiarini, L. (2020a, marzo 14). *≫ 100 Codigos HTML para páginas web 【en bloc de notas】* . LuisForgiariniBlog. https://luisforgiariniblog.com/codigos-html-para-paginas-web-bloc-notas/ [Consultado: 26/08/2020]

G., A. (2020, 1 junio). *Cómo utilizar un sensor de humedad de suelo con Arduino*. Automatización para Todos. https://www.automatizacionparatodos.com/sensor-de-humedad-de-suelo-con-arduino/ [Consultado: 30/07/2020]

Grokhotkov, I. (s. f.). *Guía para PROGMEM sobre ESP8266 y Arduino IDE — documentación de ESP8266 Arduino Core - 2.4.0*. ESP8266 Arduino Core. Recuperado 18 de octubre de 2020, de

https://esp8266-arduino-spanish.readthedocs.io/es/latest/PROGMEM.html [Consultado: 18/10/2020]

Grokhotkov, I. (s. f.). *Librería ESP8266WiFi — documentación de ESP8266 Arduino Core - 2.4.0*. ESP8266 Arduino Core. Recuperado 22 de septiembre de 2020, de https://esp8266-arduino-spanish.readthedocs.io/es/latest/esp8266wifi/readme.html [Consultado: 22/09/2020]

Grokhotkov, I. (s. f.). *Mi ESP se bloquea al correr el programa. ¿Como lo resuelvo? — documentación de ESP8266 Arduino Core - 2.4.0*. ESP8266 Arduino Core. Recuperado 16 de noviembre de 2020, de https://esp8266-arduino-spanish.readthedocs.io/es/latest/faq/a02-my-esp-crashes.html [Consultado: 16/11/2020]

Grokhotkov, I. (2017, 14 mayo). *ESP8266 Arduino Core Documentation*. Read the Docs. https://buildmedia.readthedocs.org/media/pdf/arduino-esp8266/docs_to_readthedocs/arduino-esp8266.pdf [Consultado: 30/07/2020]

H. (2018, 18 abril). *Conexión WiFi con módulo ESP8266 y Arduino Mega*. HOMEWORKERS. https://homeworkers.co/conexion-wifi-con-modulo-esp8266-y-arduino-mega/ [Consultado: 21/07/2020]

Hernández, L. D. V. (2020, 12 mayo). *Reloj con Arduino, cómo controlar los tiempos con un RTC*. Programar fácil con Arduino. https://programarfacil.com/blog/arduino-blog/reloj-con-arduino-rtc/ [Consultado: 28/05/2020]

Hernández, L. D. V. (2020, 15 noviembre). *Introducción a HTML5 para comenzar con tus proyectos del IoT*. Programar fácil con Arduino. https://programarfacil.com/blog/programacion/introduccion-a-html5/ [Consultado: 14/09/2020]

Hernández, L. D. V. (2020, 30 mayo). *Guía para configurar un ESP-01, el módulo WiFi basado en ESP8266*. Programar fácil con Arduino. https://programarfacil.com/podcast/como-configurar-esp01-wifi-esp8266/ [Consultado: 23/07/2020]

HTML: básico. (2020, 8 noviembre). Documentación web de MDN. https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics [Consultado: 26/08/2020]

Humedad relativa Segovia - Observaciones | Castilla y León Retrospectiva del tiempo. (s. f.). woespana.es. Recuperado 26 de mayo de 2020, de https://www.woespana.es/weather/maps/city?LANG=es&WMO=08213&ART=RLF&CONT=eses&R=0&LEVEL=150&REGION=0005&LAND=ECL&NOREGION=0&MOD=&TMX=&TMN=&SON=&PRE=&MONAT=&OFFS=&SORT= [Consultado: 26/05/2020]

I. (2019, 30 mayo). *¿Qué es una API y para qué sirve?* Viewnext. https://www.viewnext.com/que-es-una-api-y-para-que-sirve/ [Consultado: 10/09/2020]

INFOTEC. (s. f.). *Sistemas Embebidos: Innovando hacia los Sistemas inteligentes*. SemanticWebBuilder. Recuperado 10 de septiembre de 2020, de http://www.semanticwebbuilder.org.mx/es_mx/swb/Sistemas_Embebidos_Innovando_hacia_los_Sistemas_Inteligentes_ [Consultado: 10/09/2020]

I2C LCD2004 - Wiki. (2019, 11 abril). Sunfounder. http://wiki.sunfounder.cc/index.php?title=I2C_LCD2004 [Consultado: 11/06/2020]

J. (s. f.). *Alineación, estilos de fuente y separadores horizontales en documentales HTML*. Con Clase. Recuperado 25 de agosto de 2020, de http://html.conclase.net/w3c/html401-es/present/graphics.html [Consultado: 25/08/2020]

J. (2016, 9 noviembre). *Alimentación Arduino*. Aprendiendo Arduino. https://aprendiendoarduino.wordpress.com/2016/11/09/alimentacion-arduino/ [Consultado: 27/05/2020]

J. (2017, 20 junio). *Placas Arduino*. Aprendiendo Arduino. https://aprendiendoarduino.wordpress.com/2017/06/19/placas-arduino-2/ [Consultado: 27/05/2020]

J. (2017, 22 octubre). *Web Server*. Aprendiendo Arduino. https://aprendiendoarduino.wordpress.com/tag/web-server/ [Consultado: 17/08/2020]

jadsa tv. (2018, 30 mayo). *SERIE SENSORES Y MODULOS #4: SENSOR DE HUMEDAD DE SUELO HL-69 / YL-69* [Vídeo]. YouTube. https://www.youtube.com/watch?v=ppvKm-5BG-0 [Consultado: 28/05/2020]

Jardinday. (2020, 11 julio). *Programadores de riego: tipos y características*. jardínday. https://jardinday.com/programador-de-riego-tipos-y-caracteristicas/ [Consultado: 4/11/2020]

JavaScript. (2020, 23 noviembre). Documentación web de MDN. https://developer.mozilla.org/es/docs/Web/JavaScript [Consultado: 11/09/2020]

León, D. P. (s. f.). *Elemento button (type=submit)*. HTMLquick. Recuperado 18 de octubre de 2020, de https://www.htmlquick.com/es/reference/tags/button-submit.html [Consultado: 18/10/2020]

Llamas, L. (2015, 11 noviembre). *Encender un LED con Arduino*. Luis Llamas. https://www.luisllamas.es/encender-un-led-con-arduino/ [Consultado: 08/06/2020]

Llamas, L. (2016, 18 mayo). *El bus I2C en Arduino*. Luis Llamas. https://www.luisllamas.es/arduino-i2c/ [Consultado: 14/09/2020]

Llamas, L. (2016, 19 enero). *Medir la humedad del suelo con Arduino e higrómetro FC-28*. Luis Llamas. https://www.luisllamas.es/arduino-humedad-suelo-fc-28/ [Consultado: 10/06/2020]

Llamas, L. (2016, 22 mayo). *Conectar un display LCD Hitachi a Arduino por bus I2C*. Luis Llamas. https://www.luisllamas.es/arduino-lcd-i2c/ [Consultado: 10/06/2020]

Llamas, L. (2017, 27 mayo). *Conectar Arduino por WiFi con el módulo ESP8266 ESP01*. Luis Llamas. https://www.luisllamas.es/arduino-wifi-esp8266-esp01/ [Consultado: 28/07/2020]

Llamas, L. (2019, 3 agosto). *Cómo servir contenido desde memoria Flash en el ESP8266*. Luis Llamas. https://www.luisllamas.es/como-servir-contenido-desde-memoria-flash-en-el-esp8266/ [Consultado: 29/07/2020]

Llamas, L. (2019, 5 mayo). *Cómo emplear el ESP8266 como servidor HTTP*. Luis Llamas. https://www.luisllamas.es/como-emplear-el-esp8266-como-servidor/ [Consultado: 29/07/2020]

Llamas, L. (2019, 7 marzo). *Cómo emplear el ESP8266 como cliente HTTP*. Luis Llamas. https://www.luisllamas.es/como-emplear-el-esp8266-como-cliente-http/ [Consultado: 28/07/2020]

Martínez de Azagra Paredes, A., & del Río San José, J. (s. f.). *Riegos de apoyo y socorro*. ACADEMIA. Recuperado 26 de octubre de 2020, de https://www.academia.edu/25724076/Riegos_de_apoyo_y_socorro [Consultado: 26/10/2020]

Michael Goering. (2016, 17 febrero). *I2C LCD 20x4* [Vídeo]. YouTube. https://www.youtube.com/watch?v=vlEcaOLjEv0 [Consultado: 09/06/2020]

Ministerio de Educación - Gobierno de España. (s. f.). *Proyecto Biosfera*. Ministerio de Educación, Cultura y Deporte. Recuperado 26 de noviembre de 2020, de http://recursos.cnice.mec.es/biosfera/alumno/1ESO/Astro/contenido12.htm

Morales, J. (2017, 24 marzo). *ESP8266, módulo wifi ESP-01S*. Playbyte.es. http://www.playbyte.es/electronica/arduino/esp-01s-modulo-wifi-basado-en-esp8266/ [Consultado: 30/07/2020]

Ordóñez, J. (2019, 24 julio). *▷ ¿Qué es una API REST? | Guía 【2020】*. Idento. https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/ [Consultado: 11/09/2020]

Programar fácil. (2017, 20 marzo). *Tutorial DHT11 con Arduino, medir temperatura y humedad* [Vídeo]. YouTube. https://www.youtube.com/watch?v=hImSF9xNARU [Consultado: 15/06/2020]

Research, T. (2020, 24 abril). *Qué es WPA PSK TKIP CCMP – Información de seguridad WiFi*. Acrylic WiFi. https://www.acrylicwifi.com/blog/que-es-wpa-psk-tkip-ccmp/ [Consultado: 17/10/2020]

Rubén Loredo. (2012, 11 agosto). *Medición de temperatura con Arduino+LM35+LCD2X16* [Vídeo]. YouTube. https://www.youtube.com/watch?v=c60mj78oojo [Consultado: 24/05/2020]

Seguridad Social: Cotización / Recaudación de Trabajadores. (2019). Seguridad Social ESP. http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537 [Consultado: 07/10/2020]

Seguridad Social: Cotización / Recaudación de Trabajadores. (2019, 2 febrero). Seguridad Social ESP. http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/10721/10957/9932/4315 [Consultado: 07/10/2020]

Servomotor Mini SG-90. (2020, 11 noviembre). Geekbot Electronics. http://www.geekbotelectronics.com/producto/servomotor-mini-sg-90/ [Consultado: 06/10/2020]

SODIMAC. (2015, 14 diciembre). *¿Cómo implementar un sistema de riego automático en el jardín?* [Vídeo]. YouTube. https://www.youtube.com/watch?v=J3YatOrbEBM [Consultado: 25/05/2020]

Temporizador de Riego. (s. f.). NOVAGRIC. Recuperado 4 de noviembre de 2020, de https://www.novagric.com/es/temporizador-de-riego [Consultado: 4/11/2020]

Tera Electronics. (2017, 6 enero). *Introducción Arduino #1 : Conociendo Arduino Uno | Partes y sus funciones* [Vídeo]. YouTube. https://www.youtube.com/watch?v=J2w-lg0EzbA [Consultado: 26/05/2020]

Tipos de datos básicos de HTML. (s. f.-b). Con Clase. Recuperado 18 de octubre de 2020, de http://html.conclase.net/w3c/html401-es/types.html#h-6 [Consultado: 18/10/2020]

Tutorial ESP8266 Parte I. (s. f.). Naylamp Mechatronics. Recuperado 16 de noviembre de 2020, de https://www.naylampmechatronics.com/blog/21_Tutorial-ESP8266-Parte-I.html [Consultado: 16/11/2020]

Tutorial HTML - Imágenes. (s. f.). Tutorialehtml. Recuperado 25 de agosto de 2020, de https://tutorialehtml.com/es/tutoriales-html-imagenes/ [Consultado: 25/08/2020]

Tutorial sensor digital de temperatura DS18B20. (s. f.). Naylamp Mechatronics. Recuperado 17 de junio de 2020, de https://www.naylampmechatronics.com/blog/46_Tutorial-sensor-de-temperatura-DS18B20.html [Consultado: 17/06/2020]

Valencia, H. G. |. (2014, 13 abril). Cómo triunfar con las rosas. *Levante-EMV*. https://www.levante-emv.com/valencia/2014/04/13/triunfar-rosas-12800835.html [Consultado: 25/05/2020]

Viagua S.L. (2019, 12 agosto). *Tipos de programadores para riego automático*. Viagua. https://viagua.es/tipos-programadores-riego-automatico/ [Consultado: 4/11/2020]

Yumpu.com. (s. f.). *Riego de jardines*. págs. 47 – 51, 77 – 112, 113 – 131, 133 – 163, 191 - 215 Recuperado 8 de octubre de 2020, de

https://www.yumpu.com/es/document/read/19654134/riego-de-jardines [Consultado: 08/10/2020]

Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

ANEXOS:

Grado en Ingeniería Electrónica Industrial y Automática

A. 'Datasheets':

En este anejo se van a disponer las 'Fichas técnicas' de cada componente para manifestar de dónde se han adquirido sus características.

Arduino MEGA2560 R3:

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

FIGURA 18: DATASHEET CARACTERÍSTICAS ARDUINO MEGA2560 R3

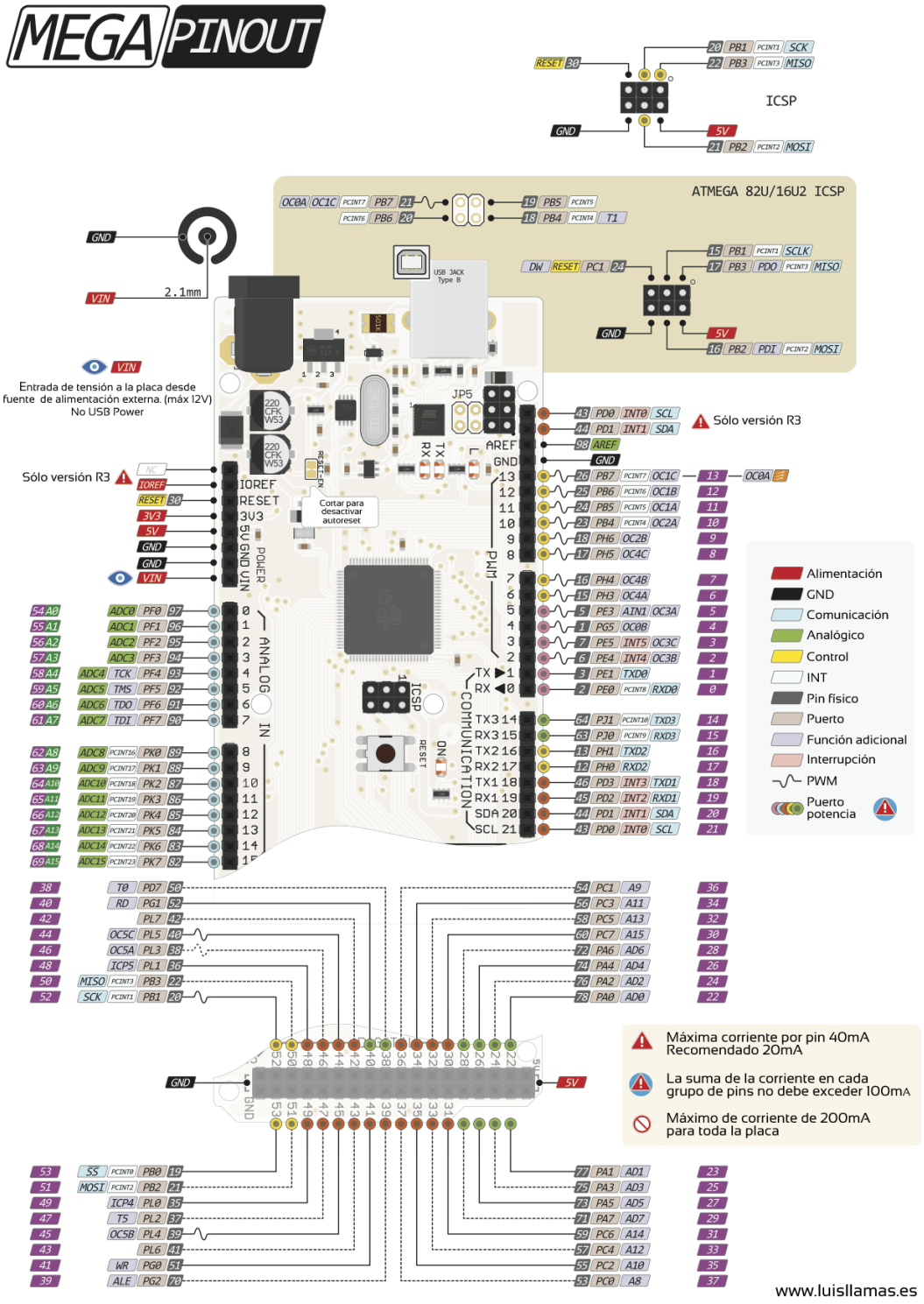


FIGURA 19: DATASHEET PINES ARDUINO MEGA2560 R3 [7]

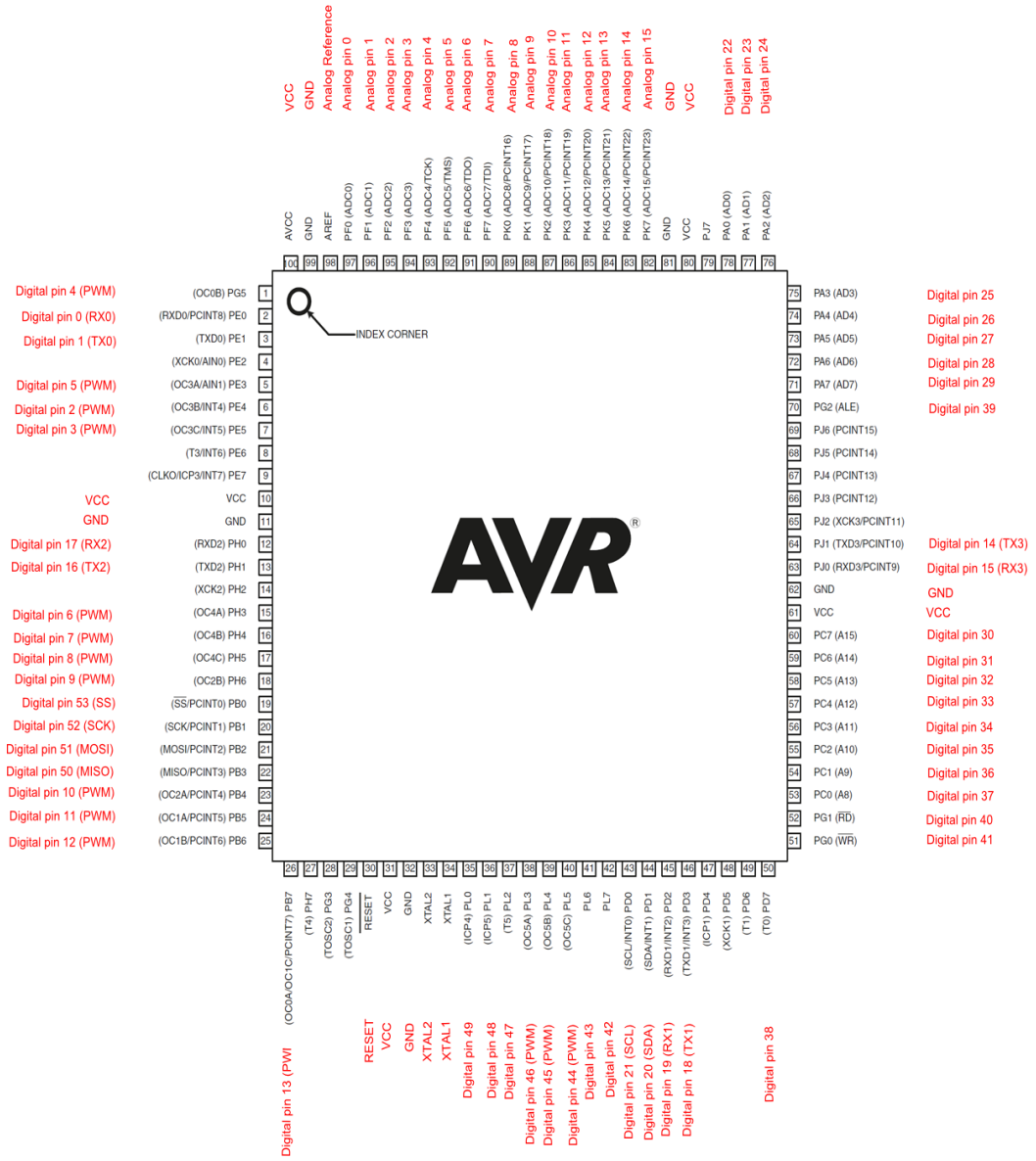


FIGURA 20: DATASHEET MAPA PIN ATMEGA2560 REV2

Fuentes de alimentación:

Fuente de alimentación blanca 9V – 1A:

Dimensions

Wall wart

- 46.8mm x 32.15mm x 32.84mm
- 1.84in x 1.26in x 1.29in

- Cable length: 146.05cm / 57.5in
- Cable Diameter: 3.8mm / 0.15in

FIGURA 21: DATASHEET DIMENSIONES FUENTE DE ALIMENTACIÓN BLANCA

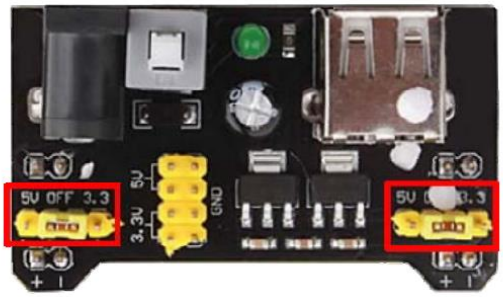
Fuente de módulo de alimentación:

Product Specifications:

- Locking On/Off Switch
- LED Power Indicator
- Input voltage: 6.5-9v (DC) via 5.5mm x 2.1mm plug
- Output voltage: 3.3V/5v
- Maximum output current: 700 mA
- Independent control rail output. 0v, 3.3v, 5v to breadboard
- Output header pins for convenient external use
- Size: 2.1 in x 1.4 in
- USB device connector onboard to power external device

FIGURA 22: DATASHEET ESPECIFICACIONES MODULO FUENTE DE ALIMENTACIÓN

Setting up output voltage:



3.3 volt configuration

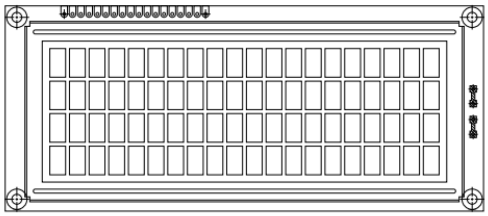
5 volt configuration

The left and right voltage output can be configured independently. To select the output voltage, move jumper to the corresponding pins. Note: power indicator LED and the breadboard power rails will not power on if both jumpers are in the "OFF" position.

FIGURA 23: DATASHEET AJUSTE MODULO FUENTE DE ALIMENTACIÓN

Pantalla LCD 20x4

20 x 4 Character LCD



FEATURES

- Type: Character
- Display format: 20 x 4 characters
- Built-in controller: ST 7066 (or equivalent)
- Duty cycle: 1/16
- 5 x 8 dots includes cursor
- + 5 V power supply (also available for + 3 V)
- LED can be driven by pin 1, pin 2, pin 15, pin 16 or A and K
- N.V. optional for + 3 V power supply
- Material categorization: For definitions of compliance please see www.vishay.com/doc?99912

Pb-free RoHS COMPLIANT

MECHANICAL DATA

ITEM	STANDARD VALUE	UNIT
Module Dimension	146.0 x 62.5	mm
Viewing Area	123.5 x 43.0	
Dot Size	0.92 x 1.10	
Dot Pitch	0.98 x 1.16	
Mounting Hole	139.0 x 55.5	
Character Size	4.84 x 9.22	

ABSOLUTE MAXIMUM RATINGS

ITEM	SYMBOL	STANDARD VALUE			UNIT
		MIN.	TYP.	MAX.	
Power Supply	V_{DD} to V_{SS}	- 0.3	-	7.0	V
Input Voltage	V_I	- 0.3	-	V_{DD}	

Note

- V_{SS} = 0 V, V_{DD} = 5.0 V

ELECTRICAL CHARACTERISTICS

ITEM	SYMBOL	CONDITION	STANDARD VALUE			UNIT
			MIN.	TYP.	MAX.	
Input Voltage	V_{DD}	V_{DD} = + 5 V	4.7	5.0	5.3	V
		V_{DD} = + 3 V	2.7	3.0	5.3	
Supply Current	I_{DD}	V_{DD} = + 5 V	-	8.0	10.0	mA
Recommended LC Driving Voltage for Normal Temperature Version Module	V_{DD} to V_0	- 20 °C	5.0	5.1	5.7	V
		0 °C	4.6	4.8	5.2	
		25 °C	4.1	4.5	4.7	
		50 °C	3.9	4.2	4.5	
		70 °C	3.7	3.9	4.3	
LED Forward Voltage	V_F	25 °C	-	4.2	4.6	V
LED Forward Current	I_F	25 °C	-	540	1080	mA
EL Power Supply Current	I_{EL}	V_{EL} = 110 V_{AC}, 400 Hz	-	-	5.0	mA

FIGURA 24: DATASHEET CARACTERÍSTICAS LCD

INTERFACE PIN FUNCTION		
PIN NO.	SYMBOL	FUNCTION
1	V_{SS}	Ground
2	V_{DD}	+ 3 V or + 5 V
3	V_0	Contrast adjustment
4	RS	H/L register select signal
5	R/$\overline{W}$	H/L read/write signal
6	E	H → L enable signal
7	DB0	H/L data bus line
8	DB1	H/L data bus line
9	DB2	H/L data bus line
10	DB3	H/L data bus line
11	DB4	H/L data bus line
12	DB5	H/L data bus line
13	DB6	H/L data bus line
14	DB7	H/L data bus line
15	A	Power supply for LED (4.2 V)
16	K	Power supply for B/L (0 V)
17	NC/V_{EE}	NC or negative voltage output
18	NC	NC connection

DIMENSIONS in millimeters

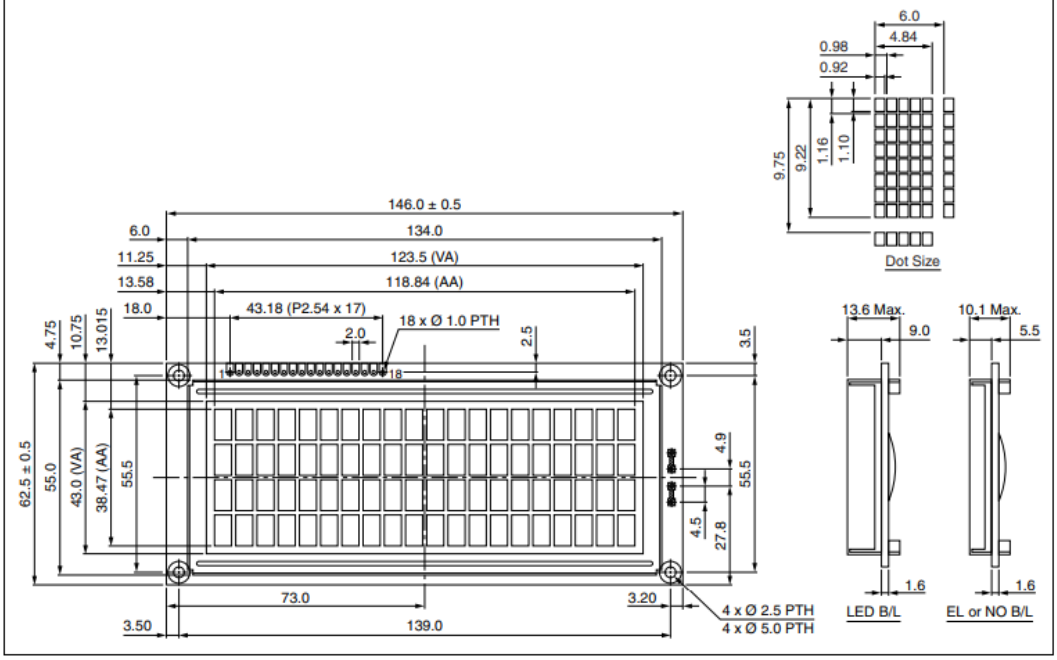


FIGURA 25: DATASHEET INTERFAZ PINES & DIMENSIONES LCD

DHT11:

Product parameters

Relative humidity
Resolution: 16Bit
Repeatability: ±1% RH
Accuracy: At 25℃ ±5% RH
Interchangeability: fully interchangeable
Response time: 1 / e (63%) of 25℃ 6s
1m / s air 6s
Hysteresis: <± 0.3% RH
Long-term stability: <± 0.5% RH / yr in

Temperature
Resolution: 16Bit
Repeatability: ±0.2℃
Range: At 25℃ ±2℃
Response time: 1 / e (63%) 10S

Electrical Characteristics
Power supply: DC 3.5 ~ 5.5V
Supply Current: measurement 0.3mA standby 60μ A
Sampling period: more than 2 seconds

Pin Description
1, the VDD power supply 3.5 ~ 5.5V DC
2 DATA serial data, a single bus
3, NC, empty pin
4, GND ground, the negative power

FIGURA 26: DATASHEET PARÁMETROS DHT11

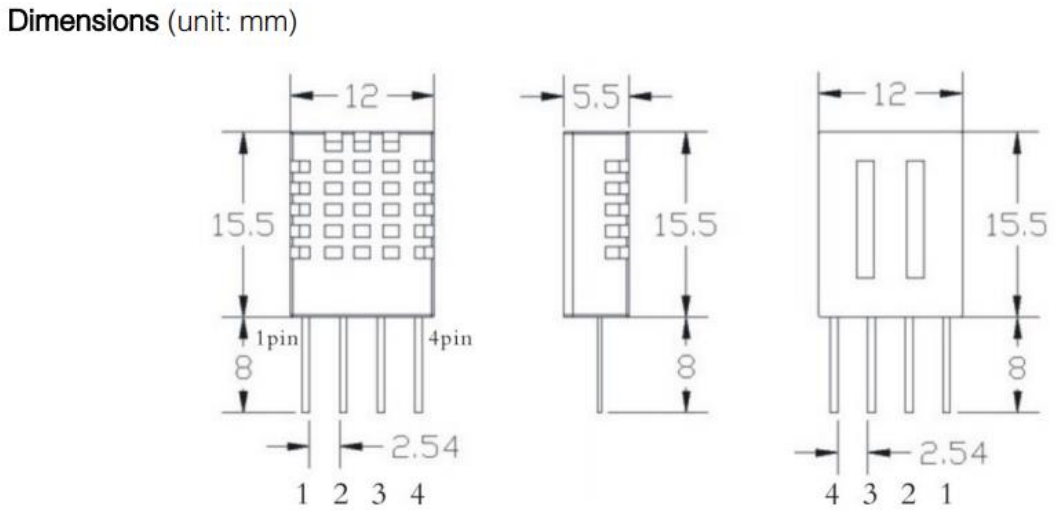


FIGURA 27: DATASHEET DIMENSIONES DHT11

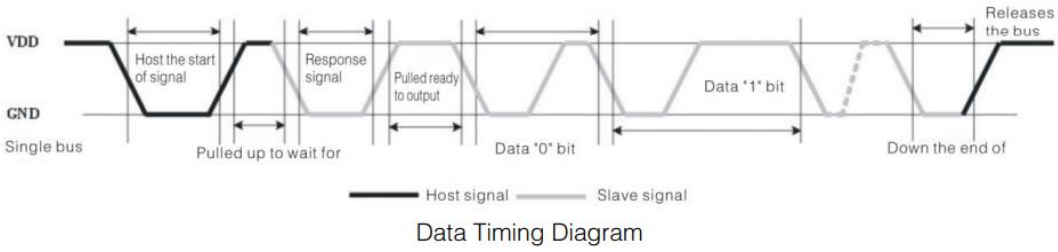


FIGURA 28: DATASHEET DIAGRAMA DE TIEMPO DE DATOS DHT11

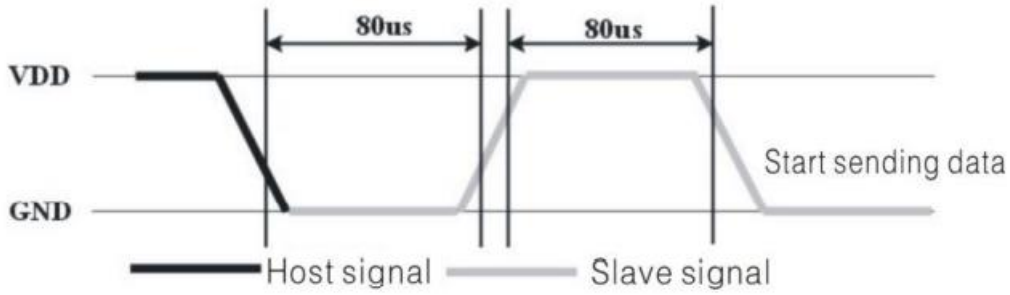


FIGURA 29: DATASHEET DIAGRAMA DE ENVÍO DE DATOS DHT11

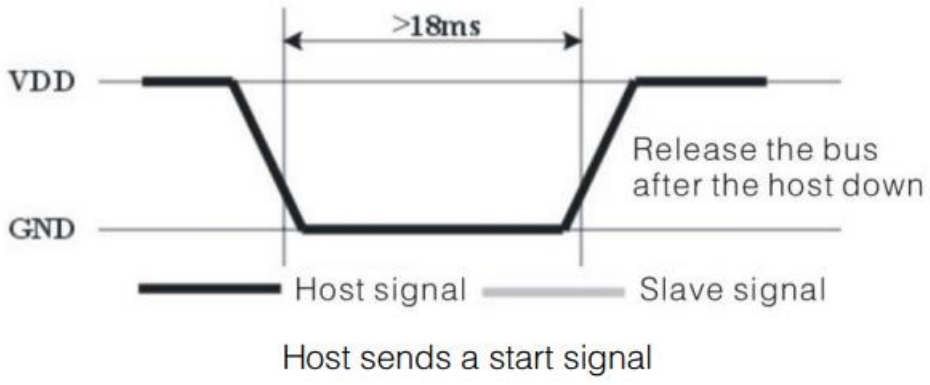


FIGURA 30: DATASHEET DIAGRAMA DE SEÑAL DE COMIENZO DHT11

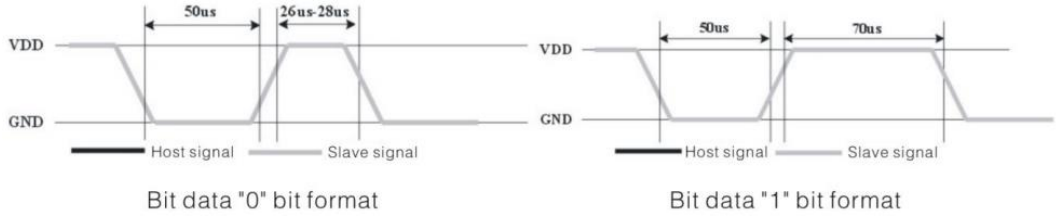


FIGURA 31: DATASHEET DIAGRAMA DE FORMATO DE SEÑAL DE BIT '0' Y '1' DHT11

Example 1: 40 data is received:

0011 0101	0000 0000	0001 1000	0000 0000	0100 1101
High humidity 8	Low humidity 8	High temp. 8	Low temp. 8	Parity bit

Calculate:

0011 0101+0000 0000+0001 1000+0000 0000= 0100 1101

Received data is correct:

Humidity: 0011 0101=35H=53%RH

Temperature: 0001 1000=18H=24℃

Example 2: 40 data is received:

0011 0101	0000 0000	0001 1000	0000 0000	0100 1001
High humidity 8	Low humidity 8	High temp. 8	Low temp. 8	Parity bit

Calculate:

0011 0101+0000 0000+0001 1000+0000 0000 = 0100 1101

01001101≠0100 1001

The received data is not correct, give up, to re-receive data.

FIGURA 32: DATASHEET EJEMPLO DE PARIDAD DHT11

DS3231:

Benefits and Features

- Highly Accurate RTC Completely Manages All Timekeeping Functions
  - Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year, with Leap-Year Compensation Valid Up to 2100
  - Accuracy ±2ppm from 0°C to +40°C
  - Accuracy ±3.5ppm from -40°C to +85°C
  - Digital Temp Sensor Output: ±3°C Accuracy
  - Register for Aging Trim
  - $\overline{RST}$ Output/Pushbutton Reset Debounce Input
  - Two Time-of-Day Alarms
  - Programmable Square-Wave Output Signal
- Simple Serial Interface Connects to Most Microcontrollers
  - Fast (400kHz) I^2C Interface
- Battery-Backup Input for Continuous Timekeeping
  - Low Power Operation Extends Battery-Backup Run Time
  - 3.3V Operation
- Operating Temperature Ranges: Commercial (0°C to +70°C) and Industrial (-40°C to +85°C)
- Underwriters Laboratories® (UL) Recognized

FIGURA 33: DATASHEET BENEFICIOS Y CARACTERÍSTICAS DS3231

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground ....-0.3V to +6.0V
Junction-to-Ambient Thermal Resistance (θ_{JA}) (Note 1)73°C/W
Junction-to-Case Thermal Resistance (θ_{JC}) (Note 1)....23°C/W
Operating Temperature Range
DS3231S .....0°C to +70°C
DS3231SN..... -40°C to +85°C
Junction Temperature.....+125°C
Storage Temperature Range..... -40°C to +85°C
Lead Temperature (soldering, 10s).....+260°C
Soldering Temperature (reflow, 2 times max) .....+260°C
(see the *Handling, PCB Layout, and Assembly* section)

FIGURA 34: DATASHEET CALIFICACIÓN MÁX. ABSOLUTA DS3231

Recommended Operating Conditions

(T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		2.3	3.3	5.5	V
	V_{BAT}		2.3	3.0	5.5	V
Logic 1 Input SDA, SCL	V_{IH}		0.7 x V_{CC}		V_{CC} + 0.3	V
Logic 0 Input SDA, SCL	V_{IL}		-0.3		0.3 x V_{CC}	V

FIGURA 35: DATASHEET CONDICIONES DE OPERACIÓN RECOMENDABLES DS3231

Electrical Characteristics

(V_{CC} = 2.3V to 5.5V, V_{CC} = Active Supply (see Table 1), T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Typical values are at V_{CC} = 3.3V, V_{BAT} = 3.0V, and T_A = +25°C, unless otherwise noted.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Active Supply Current	I_{CCA}	(Notes 4, 5)	V_{CC} = 3.63V			200	µA
			V_{CC} = 5.5V			300	
Standby Supply Current	I_{CCS}	I²C bus inactive, 32kHz output on, SQW output off (Note 5)	V_{CC} = 3.63V			110	µA
			V_{CC} = 5.5V			170	
Temperature Conversion Current	$I_{CCSCONV}$	I²C bus inactive, 32kHz output on, SQW output off	V_{CC} = 3.63V			575	µA
			V_{CC} = 5.5V			650	
Power-Fail Voltage	V_{PF}			2.45	2.575	2.70	V
Logic 0 Output, 32kHz, $\overline{INT}$/SQW, SDA	V_{OL}	I_{OL} = 3mA				0.4	V
Logic 0 Output, $\overline{RST}$	V_{OL}	I_{OL} = 1mA				0.4	V
Output Leakage Current 32kHz, $\overline{INT}$/SQW, SDA	I_{LO}	Output high impedance		-1	0	+1	µA
Input Leakage SCL	I_{LI}			-1		+1	µA
$\overline{RST}$ Pin I/O Leakage	I_{OL}	$\overline{RST}$ high impedance (Note 6)		-200		+10	µA
V_{BAT} Leakage Current (V_{CC} Active)	I_{BATLKG}				25	100	nA

FIGURA 36: DATASHEET CARACTERÍSTICAS ELÉCTRICAS DS3231

Electrical Characteristics (continued)

(V_{CC} = 2.3V to 5.5V, V_{CC} = Active Supply (see Table 1), T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Typical values are at V_{CC} = 3.3V, V_{BAT} = 3.0V, and T_A = +25°C, unless otherwise noted.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Output Frequency	f_{OUT}	V_{CC} = 3.3V or V_{BAT} = 3.3V			32.768		kHz
Frequency Stability vs. Temperature (Commercial)	$\Delta f/f_{OUT}$	V_{CC} = 3.3V or V_{BAT} = 3.3V, aging offset = 00h	0°C to +40°C			±2	ppm
			>40°C to +70°C			±3.5	
Frequency Stability vs. Temperature (Industrial)	$\Delta f/f_{OUT}$	V_{CC} = 3.3V or V_{BAT} = 3.3V, aging offset = 00h	-40°C to <0°C			±3.5	ppm
			0°C to +40°C			±2	
			>40°C to +85°C			±3.5	
Frequency Stability vs. Voltage	$\Delta f/V$				1		ppm/V
Trim Register Frequency Sensitivity per LSB	$\Delta f/LSB$	Specified at:	-40°C		0.7		ppm
			+25°C		0.1		
			+70°C		0.4		
			+85°C		0.8		
Temperature Accuracy	Temp	V_{CC} = 3.3V or V_{BAT} = 3.3V		-3		+3	°C
Crystal Aging	$\Delta f/f_O$	After reflow, not production tested	First year		±1.0		ppm
			0–10 years		±5.0		

Electrical Characteristics

(**V_{CC} = 0V, V_{BAT} = 2.3V to 5.5V**, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Active Battery Current	I_{BATA}	$\overline{EOSC}$ = 0, BBSQW = 0, SCL = 400kHz (Note 5)	V_{BAT} = 3.63V			70	µA
			V_{BAT} = 5.5V			150	
Timekeeping Battery Current	I_{BATT}	$\overline{EOSC}$ = 0, BBSQW = 0, EN32kHz = 1, SCL = SDA = 0V or SCL = SDA = V_{BAT} (Note 5)	V_{BAT} = 3.63V		0.84	3.0	µA
			V_{BAT} = 5.5V		1.0	3.5	
Temperature Conversion Current	I_{BATTC}	$\overline{EOSC}$ = 0, BBSQW = 0, SCL = SDA = 0V or SCL = SDA = V_{BAT}	V_{BAT} = 3.63V			575	µA
			V_{BAT} = 5.5V			650	
Data-Retention Current	I_{BATTDR}	$\overline{EOSC}$ = 1, SCL = SDA = 0V, +25°C				100	nA

FIGURA 37: DATASHEET CARACTERÍSTICAS ELÉCTRICAS (II) DS3231

AC Electrical Characteristics

(V_{CC} = $V_{CC(MIN)}$ to $V_{CC(MAX)}$ or V_{BAT} = $V_{BAT(MIN)}$ to $V_{BAT(MAX)}$, V_{BAT} > V_{CC}, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Note 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
SCL Clock Frequency	f_{SCL}	Fast mode	100		400	kHz
		Standard mode	0		100	
Bus Free Time Between STOP and START Conditions	t_{BUF}	Fast mode	1.3			µs
		Standard mode	4.7			
Hold Time (Repeated) START Condition (Note 7)	$t_{HD:STA}$	Fast mode	0.6			µs
		Standard mode	4.0			
Low Period of SCL Clock	t_{LOW}	Fast mode	1.3			µs
		Standard mode	4.7			
High Period of SCL Clock	t_{HIGH}	Fast mode	0.6			µs
		Standard mode	4.0			
Data Hold Time (Notes 8, 9)	$t_{HD:DAT}$	Fast mode	0		0.9	µs
		Standard mode	0		0.9	
Data Setup Time (Note 10)	$t_{SU:DAT}$	Fast mode	100			ns
		Standard mode	250			
START Setup Time	$t_{SU:STA}$	Fast mode	0.6			µs
		Standard mode	4.7			
Rise Time of Both SDA and SCL Signals (Note 11)	t_R	Fast mode	20 + $0.1C_B$		300	ns
		Standard mode			1000	
Fall Time of Both SDA and SCL Signals (Note 11)	t_F	Fast mode	20 + $0.1C_B$		300	ns
		Standard mode			300	
Setup Time for STOP Condition	$t_{SU:STO}$	Fast mode	0.6			µs
		Standard mode	4.7			
Capacitive Load for Each Bus Line	C_B	(Note 11)			400	pF
Capacitance for SDA, SCL	$C_{I/O}$			10		pF
Pulse Width of Spikes That Must Be Suppressed by the Input Filter	t_{SP}			30		ns
Pushbutton Debounce	PB_{DB}			250		ms
Reset Active Time	t_{RST}			250		ms
Oscillator Stop Flag (OSF) Delay	t_{OSF}	(Note 12)		100		ms
Temperature Conversion Time	t_{CONV}			125	200	ms

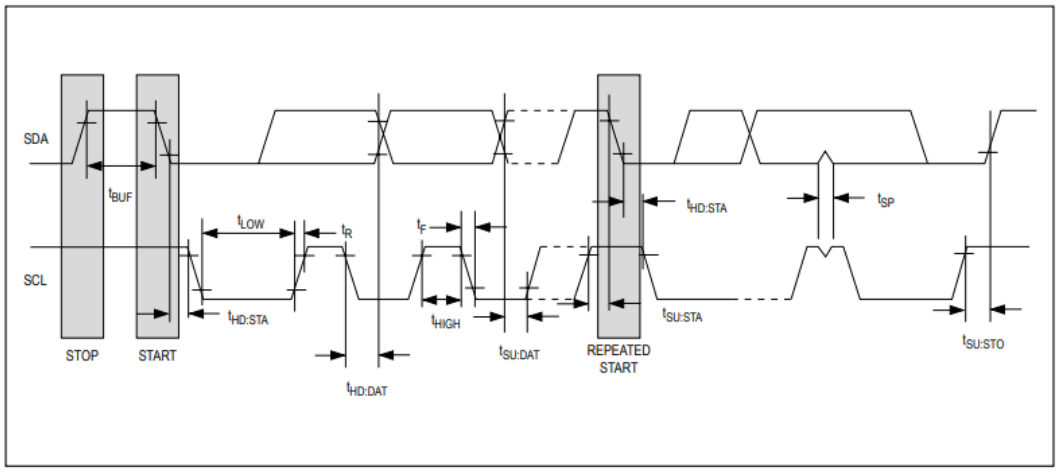
Power-Switch Characteristics

(T_A = T_{MIN} to T_{MAX})

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{CC} Fall Time; $V_{PF(MAX)}$ to $V_{PF(MIN)}$	t_{VCCF}		300			µs
V_{CC} Rise Time; $V_{PF(MIN)}$ to $V_{PF(MAX)}$	t_{VCCR}		0			µs
Recovery at Power-Up	t_{REC}	(Note 13)		250	300	ms

FIGURA 38: CARACTERÍSTICAS ELÉCTRICAS CA Y DE ENCENDIDO DS3231

Data Transfer on I²C Serial Bus



WARNING: Negative undershoots below -0.3V while the part is in battery-backed mode may cause loss of data.

Note 2: Limits at -40°C are guaranteed by design and not production tested.

Note 3: All voltages are referenced to ground.

Note 4: I_{CCA}—SCL clocking at max frequency = 400kHz.

Note 5: Current is the averaged input current, which includes the temperature conversion current.

Note 6: The $\overline{RST}$ pin has an internal 50kΩ (nominal) pullup resistor to V_{CC}.

Note 7: After this period, the first clock pulse is generated.

Note 8: A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the $V_{IH(MIN)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.

Note 9: The maximum $t_{HD:DAT}$ needs only to be met if the device does not stretch the low period (t_{LOW}) of the SCL signal.

Note 10: A fast-mode device can be used in a standard-mode system, but the requirement $t_{SU:DAT} \geq 250ns$ must then be met. This is automatically the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line $t_{R(MAX)} + t_{SU:DAT} = 1000 + 250 = 1250ns$ before the SCL line is released.

Note 11: C_B—total capacitance of one bus line in pF.

Note 12: The parameter t_{OSF} is the period of time the oscillator must be stopped for the OSF flag to be set over the voltage range of $0.0V \leq V_{CC} \leq V_{CC(MAX)}$ and $2.3V \leq V_{BAT} \leq 3.4V$.

Note 13: This delay applies only if the oscillator is enabled and running. If the $\overline{EOSC}$ bit is a 1, t_{REC} is bypassed and $\overline{RST}$ immediately goes high. The state of $\overline{RST}$ does not affect the I²C interface, RTC, or TCXO.

FIGURA 39: DATASHEET TRANSFERENCIA DE DATOS BUS I2C DS3231

Typical Operating Characteristics

(V_{CC} = +3.3V, T_A = +25°C, unless otherwise noted.)

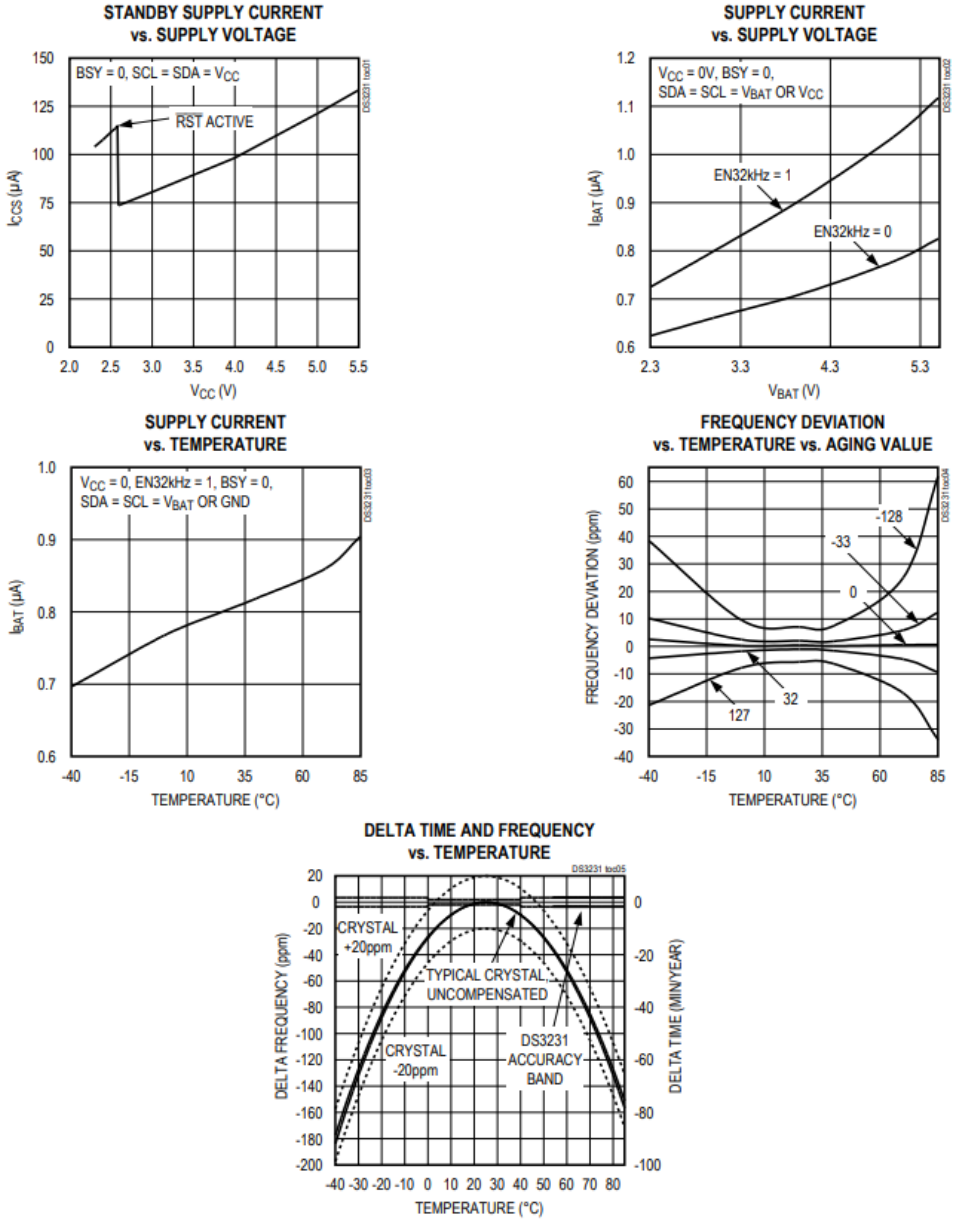


FIGURA 40: DATASHEET CARACTERÍSTICAS TÍPICAS DE OPERACIÓN DS3231

Block Diagram

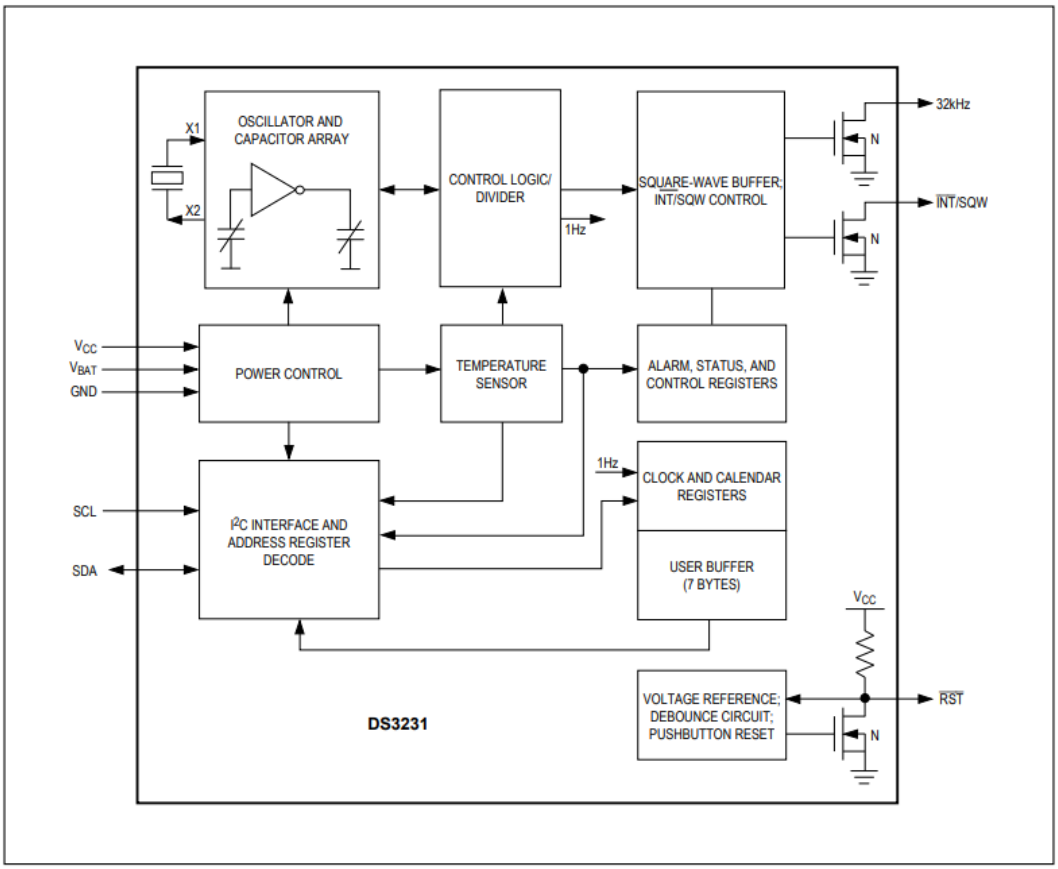


FIGURA 41: DATASHEET DIAGRAMA DE BLOQUES DS3231

Pin Description

PIN	NAME	FUNCTION
1	32kHz	32kHz Output. This open-drain pin requires an external pullup resistor. When enabled, the output operates on either power supply. It may be left open if not used.
2	V_{CC}	DC Power Pin for Primary Power Supply. This pin should be decoupled using a 0.1µF to 1.0µF capacitor. If not used, connect to ground.
3	$\overline{INT}$/SQW	Active-Low Interrupt or Square-Wave Output. This open-drain pin requires an external pullup resistor connected to a supply at 5.5V or less. This multifunction pin is determined by the state of the INTCN bit in the Control Register (0Eh). When INTCN is set to logic 0, this pin outputs a square wave and its frequency is determined by RS2 and RS1 bits. When INTCN is set to logic 1, then a match between the timekeeping registers and either of the alarm registers activates the $\overline{INT}$/SQW pin (if the alarm is enabled). Because the INTCN bit is set to logic 1 when power is first applied, the pin defaults to an interrupt output with alarms disabled. The pullup voltage can be up to 5.5V, regardless of the voltage on V_{CC}. If not used, this pin can be left unconnected.
4	$\overline{RST}$	Active-Low Reset. This pin is an open-drain input/output. It indicates the status of V_{CC} relative to the V_{PF} specification. As V_{CC} falls below V_{PF}, the $\overline{RST}$ pin is driven low. When V_{CC} exceeds V_{PF}, for t_{RST}, the $\overline{RST}$ pin is pulled high by the internal pullup resistor. The active-low, open-drain output is combined with a debounced pushbutton input function. This pin can be activated by a pushbutton reset request. It has an internal 50kΩ nominal value pullup resistor to V_{CC}. No external pullup resistors should be connected. If the oscillator is disabled, t_{REC} is bypassed and $\overline{RST}$ immediately goes high.
5–12	N.C.	No Connection. Must be connected to ground.
13	GND	Ground
14	V_{BAT}	Backup Power-Supply Input. When using the device with the V_{BAT} input as the primary power source, this pin should be decoupled using a 0.1µF to 1.0µF low-leakage capacitor. When using the device with the V_{BAT} input as the backup power source, the capacitor is not required. If V_{BAT} is not used, connect to ground. The device is UL recognized to ensure against reverse charging when used with a primary lithium battery. Go to www.maximintegrated.com/qa/info/ul.
15	SDA	Serial Data Input/Output. This pin is the data input/output for the I²C serial interface. This open-drain pin requires an external pullup resistor. The pullup voltage can be up to 5.5V, regardless of the voltage on V_{CC}.
16	SCL	Serial Clock Input. This pin is the clock input for the I²C serial interface and is used to synchronize data movement on the serial interface. Up to 5.5V can be used for this pin, regardless of the voltage on V_{CC}.

FIGURA 42: DATASHEET DESCRIPCIÓN DE PINES DS3231

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM / 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM / 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM / 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM / 00–23
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM / 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM / 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

FIGURA 43: DATASHEET REGISTROS DE CRONOMETRAJE DS3231

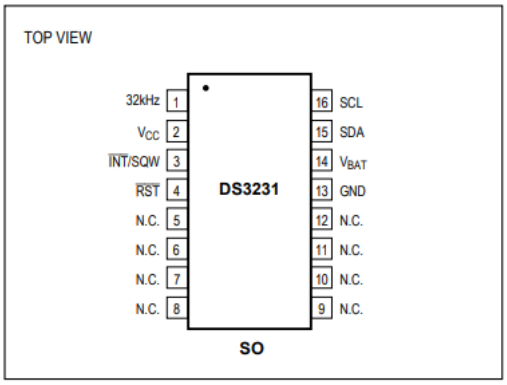
Table 2. Alarm Mask Bits

DY/$\overline{DT}$	ALARM 1 REGISTER MASK BITS (BIT 7)				ALARM RATE
	A1M4	A1M3	A1M2	A1M1	
X	1	1	1	1	Alarm once per second
X	1	1	1	0	Alarm when seconds match
X	1	1	0	0	Alarm when minutes and seconds match
X	1	0	0	0	Alarm when hours, minutes, and seconds match
0	0	0	0	0	Alarm when date, hours, minutes, and seconds match
1	0	0	0	0	Alarm when day, hours, minutes, and seconds match

DY/$\overline{DT}$	ALARM 2 REGISTER MASK BITS (BIT 7)			ALARM RATE
	A2M4	A2M3	A2M2	
X	1	1	1	Alarm once per minute (00 seconds of every minute)
X	1	1	0	Alarm when minutes match
X	1	0	0	Alarm when hours and minutes match
0	0	0	0	Alarm when date, hours, and minutes match
1	0	0	0	Alarm when day, hours, and minutes match

FIGURA 44: DATASHEET MÁSCARAS DE ALARMA DS3231

Pin Configuration



Chip Information

SUBSTRATE CONNECTED TO GROUND

PROCESS: CMOS

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
DS3231S#	0°C to +70°C	16 SO
DS3231SN#	-40°C to +85°C	16 SO

#Denotes an RoHS-compliant device that may include lead (Pb) that is exempt under RoHS requirements. The lead finish is JESD97 category e3, and is compatible with both lead-based and lead-free soldering processes. A "#" anywhere on the top mark denotes an RoHS-compliant device.

Package Information

For the latest package outline information and land patterns (footprints), go to www.maximintegrated.com/packages. Note that a "+", "#", or "-" in the package code indicates RoHS status only. Package drawings may show a different suffix character, but the drawing pertains to the package regardless of RoHS status.

PACKAGE TYPE	PACKAGE CODE	OUTLINE NO.	LAND PATTERN NO.
16 SO	W16#H2	21-0042	90-0107

FIGURA 45: DATASHEET CONFIGURACIÓN PINES E INFORMACIÓN DS3231

DS18B20:

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations

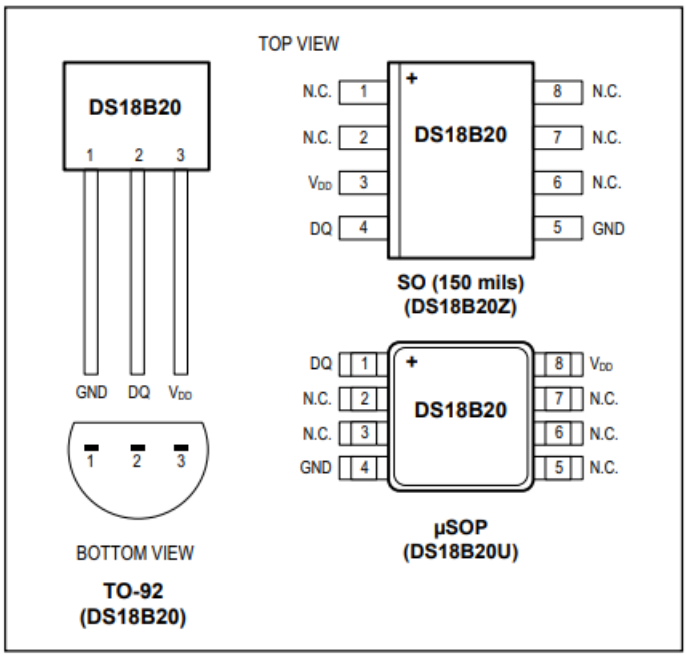


FIGURA 46: DATASHEET CARACTERÍSTICAS Y PINES DS18B20

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground ....-0.5V to +6.0V
Operating Temperature Range..... -55°C to +125°C
Storage Temperature Range..... -55°C to +125°C
Solder Temperature.....Refer to the IPC/JEDEC J-STD-020 Specification.

These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

DC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Supply Voltage	V_{DD}	Local power (Note 1)		+3.0		+5.5	V
Pullup Supply Voltage	V_{PU}	Parasite power	(Notes 1, 2)	+3.0		+5.5	V
		Local power		+3.0		V_{DD}	
Thermometer Error	t_{ERR}	-10°C to +85°C	(Note 3)			±0.5	°C
		-30°C to +100°C				±1	
		-55°C to +125°C				±2	
Input Logic-Low	V_{IL}	(Notes 1, 4, 5)		-0.3		+0.8	V
Input Logic-High	V_{IH}	Local power	(Notes 1,6)	+2.2		The lower of 5.5 or V_{DD} + 0.3	V
		Parasite power		+3.0			
Sink Current	I_L	$V_{I/O}$ = 0.4V		4.0			mA
Standby Current	I_{DDS}	(Notes 7, 8)			750	1000	nA
Active Current	I_{DD}	V_{DD} = 5V (Note 9)			1	1.5	mA
DQ Input Current	I_{DQ}	(Note 10)			5		µA
Drift		(Note 11)			±0.2		°C

Note 1: All voltages are referenced to ground.
Note 2: The Pullup Supply Voltage specification assumes that the pullup device is ideal, and therefore the high level of the pullup is equal to V_{PU}. In order to meet the V_{IH} spec of the DS18B20, the actual supply rail for the strong pullup transistor must include margin for the voltage drop across the transistor when it is turned on; thus: $V_{PU\_ACTUAL} = V_{PU\_IDEAL} + V_{TRANSISTOR}$.
Note 3: See typical performance curve in Figure 1. Thermometer Error limits are 3-sigma values.
Note 4: Logic-low voltages are specified at a sink current of 4mA.
Note 5: To guarantee a presence pulse under low voltage parasite power conditions, V_{ILMAX} may have to be reduced to as low as 0.5V.
Note 6: Logic-high voltages are specified at a source current of 1mA.
Note 7: Standby current specified up to +70°C. Standby current typically is 3µA at +125°C.
Note 8: To minimize I_{DDS}, DQ should be within the following ranges: GND ≤ DQ ≤ GND + 0.3V or V_{DD} – 0.3V ≤ DQ ≤ V_{DD}.
Note 9: Active current refers to supply current during active temperature conversions or EEPROM writes.
Note 10: DQ line is high ("high-Z" state).
Note 11: Drift data is based on a 1000-hour stress test at +125°C with V_{DD} = 5.5V.

FIGURA 47: DATASHEET VALORES MÁX. Y CARACTERÍSTICAS ELÉC. CC DS18B20

AC Electrical Characteristics–NV Memory

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
NV Write Cycle Time	t_{WR}			2	10	ms
EEPROM Writes	N_{EEWR}	-55°C to +55°C	50k			writes
EEPROM Data Retention	t_{EEDR}	-55°C to +55°C	10			years

AC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Temperature Conversion Time	t_{CONV}	9-bit resolution	(Note 12)			93.75	ms
		10-bit resolution				187.5	
		11-bit resolution				375	
		12-bit resolution				750	
Time to Strong Pullup On	t_{SPON}	Start convert T command issued				10	µs
Time Slot	t_{SLOT}	(Note 12)		60		120	µs
Recovery Time	t_{REC}	(Note 12)		1			µs
Write 0 Low Time	t_{LOW0}	(Note 12)		60		120	µs
Write 1 Low Time	t_{LOW1}	(Note 12)		1		15	µs
Read Data Valid	t_{RDV}	(Note 12)				15	µs
Reset Time High	t_{RSTH}	(Note 12)		480			µs
Reset Time Low	t_{RSTL}	(Notes 12, 13)		480			µs
Presence-Detect High	t_{PDHIGH}	(Note 12)		15		60	µs
Presence-Detect Low	t_{PDLOW}	(Note 12)		60		240	µs
Capacitance	$C_{IN/OUT}$					25	pF

Note 12: See the timing diagrams in Figure 2.
Note 13: Under parasite power, if t_{RSTL} > 960µs, a power-on reset can occur.

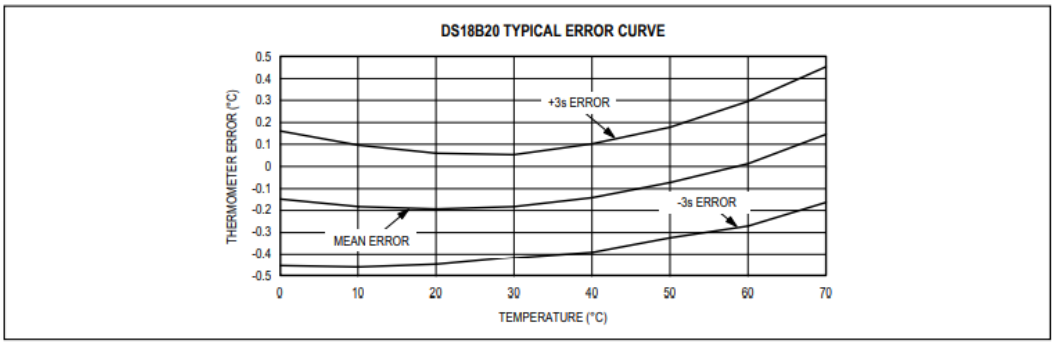


Figure 1. Typical Performance Curve

FIGURA 48: DATASHEET CARACTERÍSTICAS ELÉC. CA DS18B20

Pin Description

PIN			NAME	FUNCTION
SO	µSOP	TO-92		
1, 2, 6, 7, 8	2, 3, 5, 6, 7	—	N.C.	No Connection
3	8	3	V_{DD}	Optional V_{DD}. V_{DD} must be grounded for operation in parasite power mode.
4	1	2	DQ	Data Input/Output. Open-drain 1-Wire interface pin. Also provides power to the device when used in parasite power mode (see the *Powering the DS18B20* section.)
5	4	1	GND	Ground

FIGURA 49: DATASHEET DESCRIPCIÓN PINES DS18B20

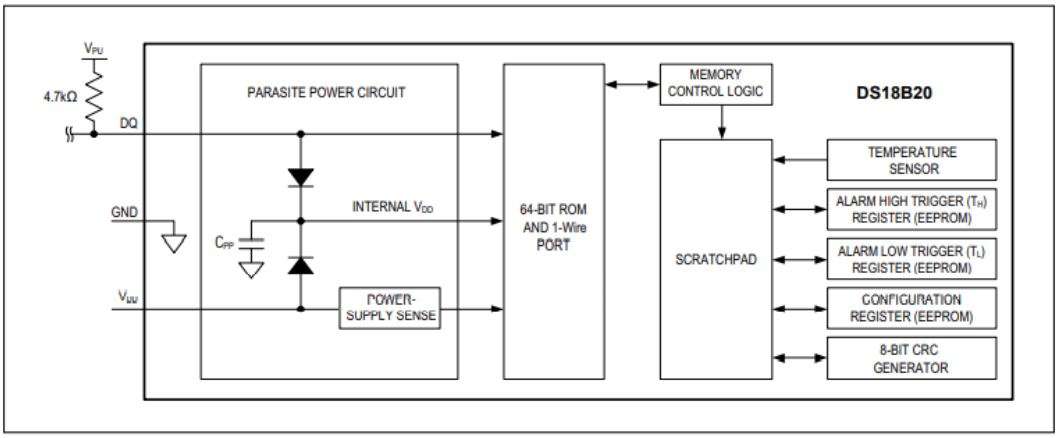


Figure 3. DS18B20 Block Diagram

FIGURA 50: DATASHEET DIAGRAMA DE BLOQUES DS18B20

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	**BIT 14**	**BIT 13**	**BIT 12**	**BIT 11**	**BIT 10**	**BIT 9**	**BIT 8**
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Figure 4. Temperature Register Format

Table 1. Temperature/Data Relationship

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

**The power-on reset value of the temperature register is +85°C.*

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Figure 5. T_H and T_L Register Format

FIGURA 51: DATASHEET FORMATO DATOS DS18B20

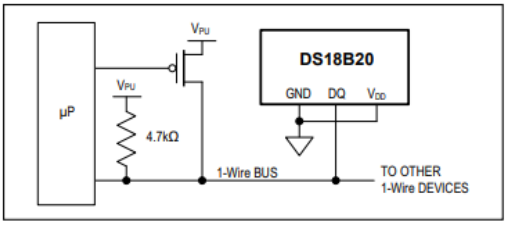


Figure 6. Supplying the Parasite-Powered DS18B20 During Temperature Conversions

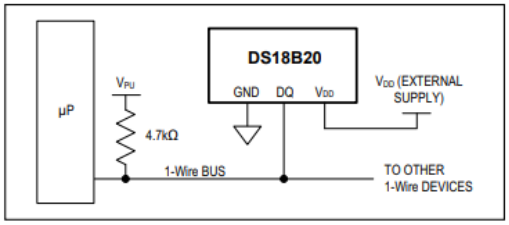


Figure 7. Powering the DS18B20 with an External Supply

FIGURA 52: DATASHEET FORMAS DE ALIMENTACIÓN DS18B20

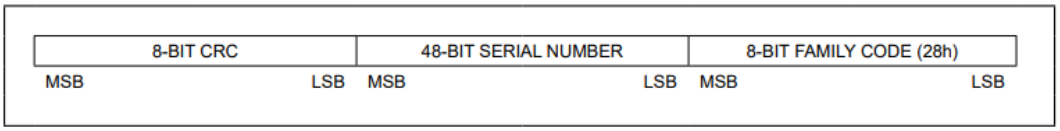


Figure 8. 64-Bit Lasered ROM Code

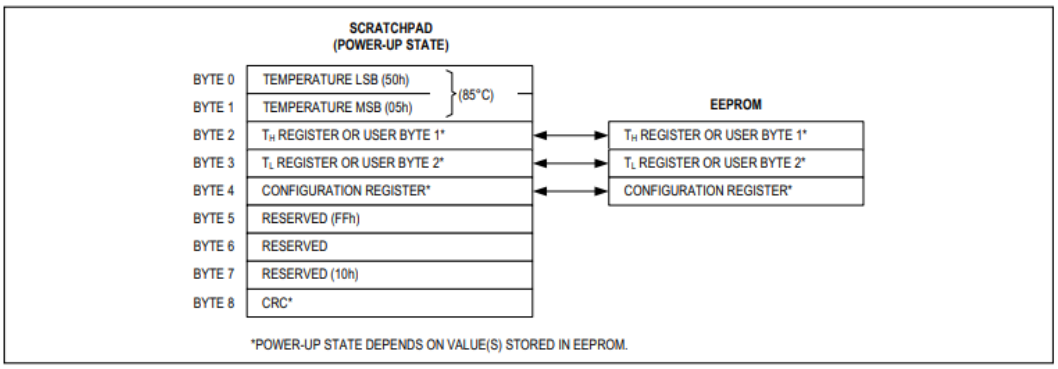


Figure 9. DS18B20 Memory Map

FIGURA 53: DATASHEET MAPA DE MEMORIA Y CODIGO ROM DS18B20

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	R1	R0	1	1	1	1	1

Figure 10. Configuration Register

Table 2. Thermometer Resolution Configuration

R1	R0	RESOLUTION (BITS)	MAX CONVERSION TIME	
0	0	9	93.75ms	(t_{CONV}/8)
0	1	10	187.5ms	(t_{CONV}/4)
1	0	11	375ms	(t_{CONV}/2)
1	1	12	750ms	(t_{CONV})

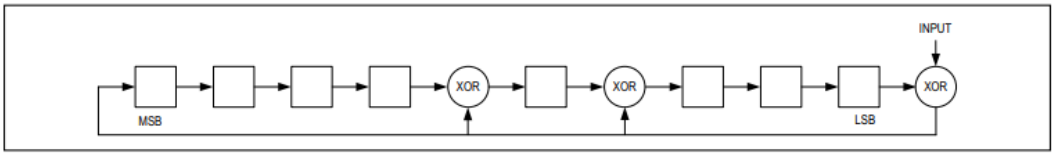


Figure 11. CRC Generator

FIGURA 54: DATASHEET CONFIGURACIÓN RESOLUCIÓN DS18B20

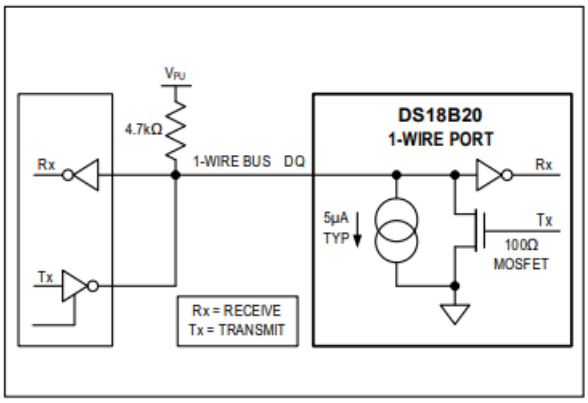


Figure 12. Hardware Configuration

FIGURA 55: DATASHEET CONFIGURACIÓN HARDWARE DS18B20

Table 3. DS18B20 Function Command Set

COMMAND	DESCRIPTION	PROTOCOL	1-Wire BUS ACTIVITY AFTER COMMAND IS ISSUED	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18B20 transmits conversion status to master (not applicable for parasite-powered DS18B20s).	1
MEMORY COMMANDS				
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18B20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2, 3, and 4 (T_H, T_L, and configuration registers).	4Eh	Master transmits 3 data bytes to DS18B20.	3
Copy Scratchpad	Copies T_H, T_L, and configuration register data from the scratchpad to EEPROM.	48h	None	1
Recall E^2	Recalls T_H, T_L, and configuration register data from EEPROM to the scratchpad.	B8h	DS18B20 transmits recall status to master.	
Read Power Supply	Signals DS18B20 power supply mode to the master.	B4h	DS18B20 transmits supply status to master.	

Note 1: For parasite-powered DS18B20s, the master must enable a strong pullup on the 1-Wire bus during temperature conversions and copies from the scratchpad to EEPROM. No other bus activity may take place during this time.
Note 2: The master can interrupt the transmission of data at any time by issuing a reset.
Note 3: All three bytes must be written before a reset is issued.

FIGURA 56: DATASHEET COMANDOS FUNCIÓN DS18B20

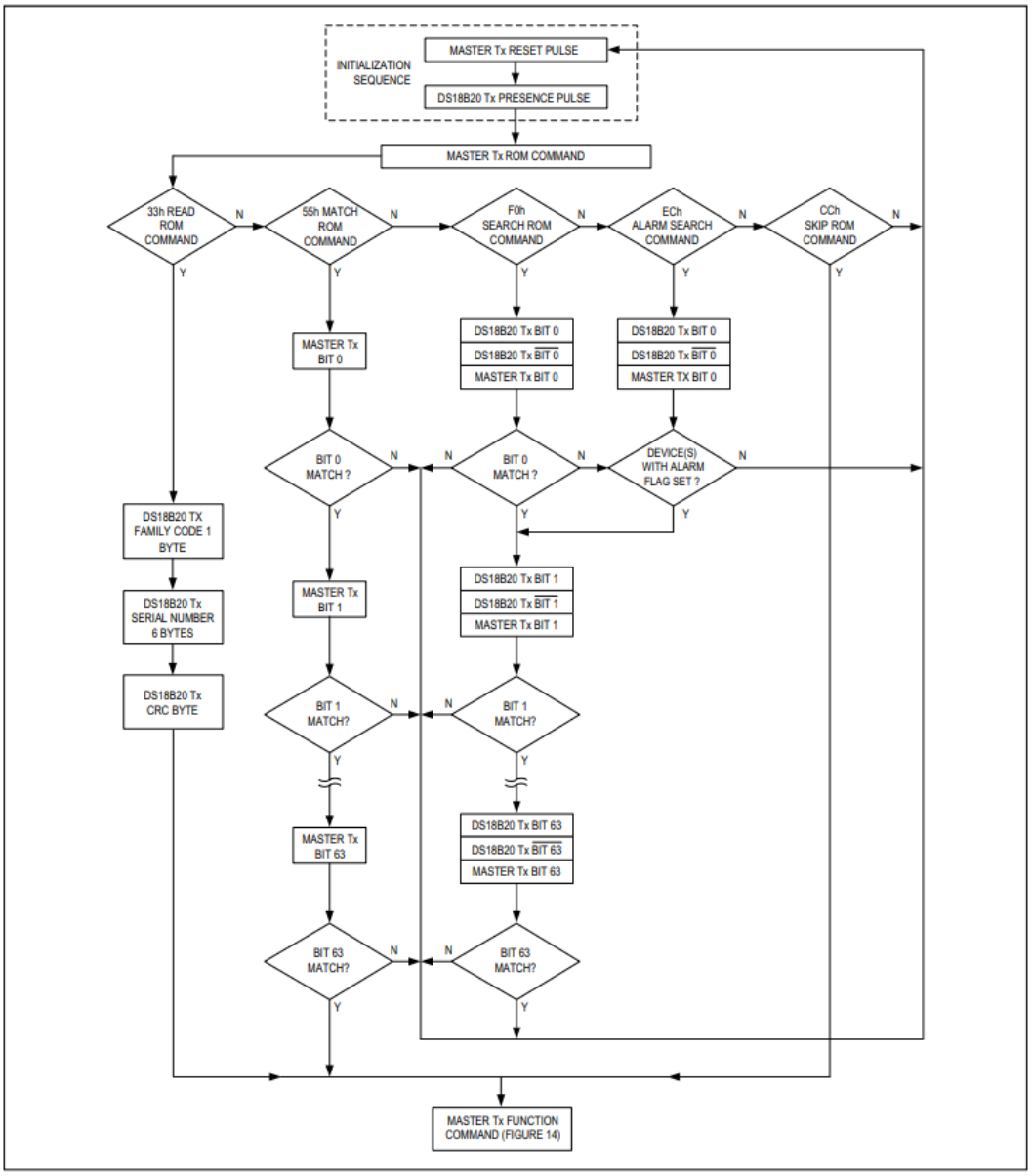


Figure 13. ROM Commands Flowchart

FIGURA 57: DATASHEET DIAGRAMA DE FLUJO COMANDOS ROM DS18B20

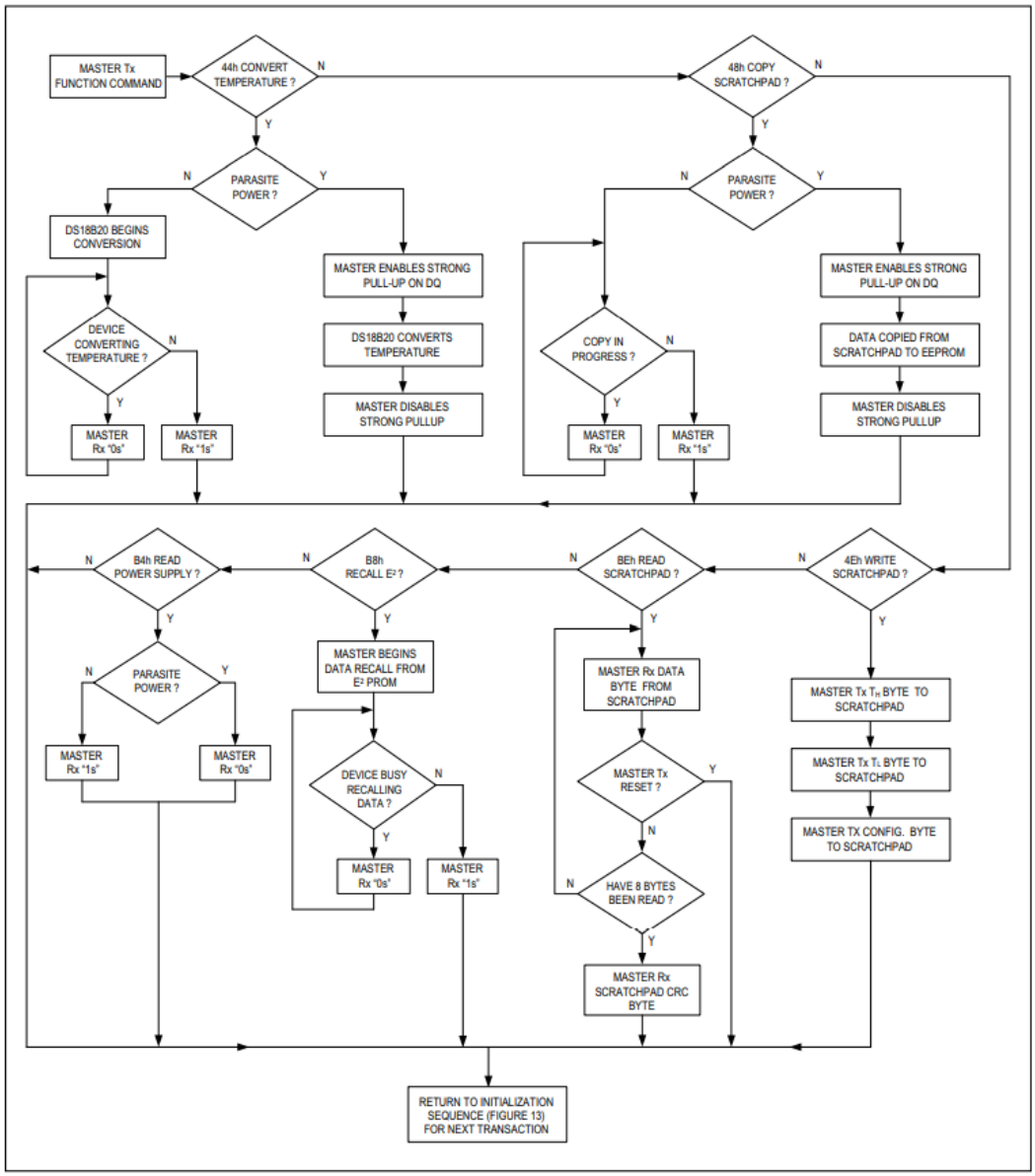


Figure 14. DS18B20 Function Commands Flowchart

FIGURA 58: DATASHEET DIAGRAMA DE FLUJO COMANDOS FUNCIÓN DS18B20

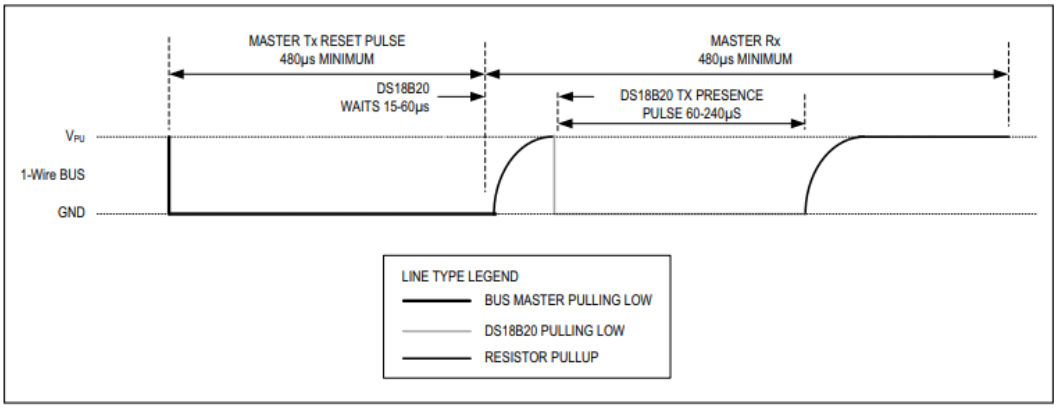


Figure 15. Initialization Timing

FIGURA 59: DATASHEET TIEMPO DE INICIALIZACIÓN DS18B20

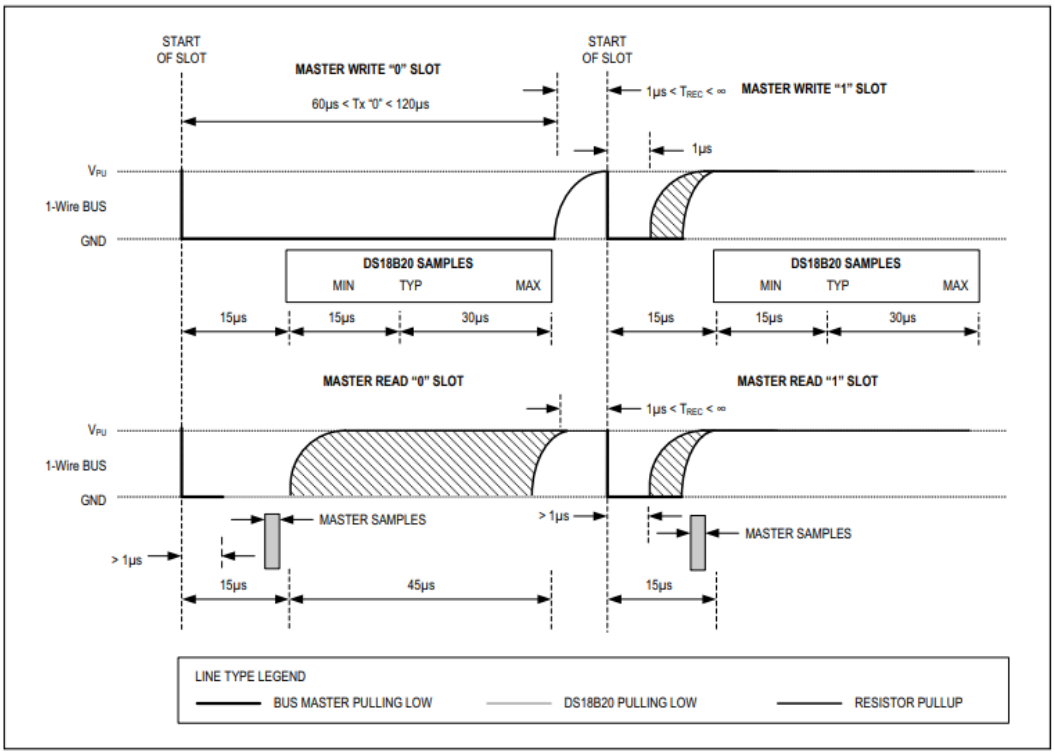


Figure 16. Read/Write Time Slot Timing Diagram

FIGURA 60: DATASHEET DIAGRAMA DE INTERVALO DE TIEMPO LECTURA-ESCRITURA DS18B20

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE	TOP MARK
DS18B20	-55°C to +125°C	3 TO-92	18B20
DS18B20+	-55°C to +125°C	3 TO-92	18B20
DS18B20/T&R	-55°C to +125°C	3 TO-92 (2000 Piece)	18B20
DS18B20+T&R	-55°C to +125°C	3 TO-92 (2000 Piece)	18B20
DS18B20-SL/T&R	-55°C to +125°C	3 TO-92 (2000 Piece)*	18B20
DS18B20-SL+T&R	-55°C to +125°C	3 TO-92 (2000 Piece)*	18B20
DS18B20U	-55°C to +125°C	8 FSOP	18B20
DS18B20U+	-55°C to +125°C	8 FSOP	18B20
DS18B20U/T&R	-55°C to +125°C	8 FSOP (3000 Piece)	18B20
DS18B20U+T&R	-55°C to +125°C	8 FSOP (3000 Piece)	18B20
DS18B20Z	-55°C to +125°C	8 SO	DS18B20
DS18B20Z+	-55°C to +125°C	8 SO	DS18B20
DS18B20Z/T&R	-55°C to +125°C	8 SO (2500 Piece)	DS18B20
DS18B20Z+T&R	-55°C to +125°C	8 SO (2500 Piece)	DS18B20

+Denotes a lead-free package. A "+" will appear on the top mark of lead-free packages.

T&R = Tape and reel.

**TO-92 packages in tape and reel can be ordered with straight or formed leads. Choose "SL" for straight leads. Bulk TO-92 orders are straight leads only.*

FIGURA 61: DATASHEET INFORMACIÓN SOBRE PEDIDOS DS18B20

Higrómetro KY70:

Comparador LM393:

1 Features

- Wide Supply
  - Voltage Range: 2.0 V to 36 V
  - Single or Dual Supplies: ±1.0 V to ±18 V
- Very Low Supply Current Drain (0.4 mA) — Independent of Supply Voltage
- Low Input Biasing Current: 25 nA
- Low Input Offset Current: ±5 nA
- Maximum Offset voltage: ±3 mV
- Input Common-Mode Voltage Range Includes Ground
- Differential Input Voltage Range Equal to the Power Supply Voltage
- Low Output Saturation Voltage: 250 mV at 4 mA
- Output Voltage Compatible with TTL, DTL, ECL, MOS and CMOS logic systems
- Available in the 8-Bump (12 mil) DSBGA Package
- See AN-1112 (SNVA009) for DSBGA Considerations
- Advantages
  - High Precision Comparators
  - Reduced V_{OS} Drift Over Temperature
  - Eliminates Need for Dual Supplies
  - Allows Sensing Near Ground
  - Compatible with All Forms of Logic
  - Power Drain Suitable for Battery Operation

2 Applications

- Battery Powered Applications
- Industrial Applications

3 Description

The LM193-N series consists of two independent precision voltage comparators with an offset voltage specification as low as 2.0 mV max for two comparators which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage. These comparators also have a unique characteristic in that the input common-mode voltage range includes ground, even though operated from a single power supply voltage.

Application areas include limit comparators, simple analog to digital converters; pulse, squarewave and time delay generators; wide range VCO; MOS clock timers; multivibrators and high voltage digital logic gates. The LM193-N series was designed to directly interface with TTL and CMOS. When operated from both plus and minus power supplies, the LM19-N series will directly interface with MOS logic where their low power drain is a distinct advantage over standard comparators.

The LM393 and LM2903 parts are available in TI's innovative thin DSBGA package with 8 (12 mil) large bumps.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM193-N	TO-99 (8)	9.08 mm x 9.08 mm
LM293-N		
LM393-N	SOIC (8)	4.90 mm x 3.91 mm
	DSBGA (8)	1.54 mm x 1.54 mm
LM2903-N	SOIC (8)	4.90 mm x 3.91 mm
	DSBGA (8)	1.54 mm x 1.54 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

FIGURA 62: DATASHEET CARACTERÍSTICAS Y DESCRIPCIÓN LM393

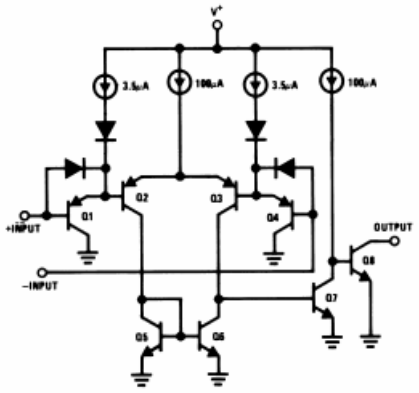
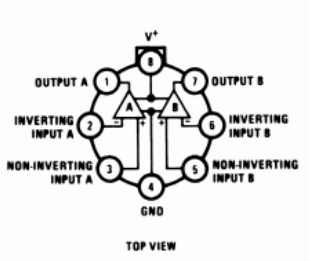


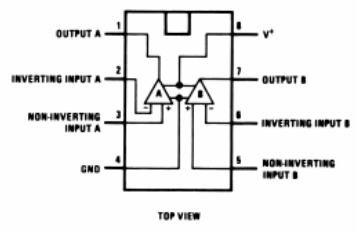
FIGURA 63: DATASHEET ESQUEMA SIMPLIFICADO LM393

5 Pin Configuration and Functions

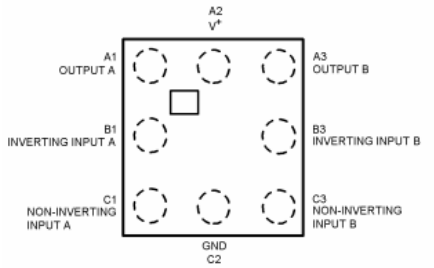
LMC Package
8-Pin TO-99
Top View



P and D Package
8-Pin CDIP, PDIP, SOIC
Top View



YZR Package
8-Pin DSBGA
Top View



Pin Functions

PIN			I/O	DESCRIPTION
NAME	NO.			
	PDIP/SOIC/ TO-99	DSBGA		
OUTA	1	A1	O	Output, Channel A
-INA	2	B1	I	Inverting Input, Channel A
+INA	3	C1	I	Noninverting Input, Channel A
GND	4	C2	P	Ground
+INB	5	C3	I	Noninverting Input, Channel B
-INB	6	B3	I	Inverting Input, Channel B
OUTB	7	A3	O	Output, Channel B
V+	8	A2	P	Positive power supply

FIGURA 64: DATASHEET CONFIGURACIÓN PINES Y FUNCIONES LM393

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)(2)(3)

			MIN	MAX	UNIT
Differential Input Voltage (4)				36	V
Input Voltage			−0.3	36	V
Input Current (V_{IN}<−0.3 V) (5)				50	mA
Power Dissipation (6)	PDIP			780	mW
	TO-99			660	mW
	SOIC			510	mW
	DSBGA			568	mW
Output Short-Circuit to Ground (7)				Continuous	
Lead Temperature (Soldering, 10 seconds)				260	°C
Soldering Information	PDIP Package Soldering (10 seconds)			260	°C
	SOIC Package	Vapor Phase (60 seconds)		215	°C
		Infrared (15 seconds)		220	°C
Storage temperature, T_{stg}			-65	150	°C

(1) *Absolute Maximum Ratings* indicate limits beyond which damage may occur. *Recommended Operating Conditions* indicate conditions for which the device is intended to be functional, but specific performance is not guaranteed. For guaranteed specifications and test conditions, see the Electrical Characteristics.
(2) Refer to RETS193AX for LM193AH military specifications and to RETS193X for LM193H military specifications.
(3) If Military/Aerospace specified devices are required, please contact the TI Sales Office/Distributors for availability and specifications.
(4) Positive excursions of input voltage may exceed the power supply level. As long as the other voltage remains within the common-mode range, the comparator will provide a proper output state. The low input voltage state must not be less than −0.3V (or 0.3V below the magnitude of the negative power supply, if used).
(5) This input current will only exist when the voltage at any of the input leads is driven negative. It is due to the collector-base junction of the input PNP transistors becoming forward biased and thereby acting as input diode clamps. In addition to this diode action, there is also lateral NPN parasitic transistor action on the IC chip. This transistor action can cause the output voltages of the comparators to go to the V^+ voltage level (or to ground for a large overdrive) for the time duration that an input is driven negative. This is not destructive and normal output states will re-establish when the input voltage, which was negative, again returns to a value greater than −0.3V.
(6) For operating at high temperatures, the LM393 and LM2903 must be derated based on a 125°C maximum junction temperature and a thermal resistance of 170°C/W which applies for the device soldered in a printed circuit board, operating in a still air ambient. The LM193/LM193A/LM293 must be derated based on a 150°C maximum junction temperature. The low bias dissipation and the "ON-OFF" characteristic of the outputs keeps the chip dissipation very small (P_D≤100 mW), provided the output transistors are allowed to saturate.
(7) Short circuits from the output to V^+ can cause excessive heating and eventual destruction. When considering short circuits to ground, the maximum output current is approximately 20 mA independent of the magnitude of V^+.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±1300	V

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

	MIN	NOM	MAX	UNIT
Supply Voltage (V+) - Single Supply	2.0		36	V
Supply Voltage (V+) - Dual Supply	±1.0		±18	V
Operating Input Voltage on (VIN pin)	0		(V+) -1.5V	V
Operating junction temperature, T_J : LM193/LM193A	-55		125	°C
Operating junction temperature, T_J : LM2903	-40		85	°C
Operating junction temperature, T_J : LM293	-25		85	°C
Operating junction temperature, T_J : LM393	0		70	°C

FIGURA 65: DATASHEET ESPECIFICACIONES LM393

6.4 Thermal Information

THERMAL METRIC(1)		LMx93 TO-99 8 PINS	UNIT
$R_{\theta JA}$	Junction-to-ambient thermal resistance	170	°C/W

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, SPRA953.

FIGURA 66: DATASHEET ESPECIFICACIONES (II) LM393

6.7 Electrical Characteristics: LMx93 and LM2903 V^+= 5 V, T_A = 25°C

Unless otherwise stated.

PARAMETER	TEST CONDITIONS		LM193-N			LM293-N, LM393-N			LM2903-N			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
Input Offset Voltage	See (1)			1.0	5.0		1.0	5.0		2.0	7.0	mV
Input Bias Current	$I_{IN}(+)$ or $I_{IN}(-)$ with Output In Linear Range, V_{CM} = 0 V (2)			25	100		25	250		25	250	nA
Input Offset Current	$I_{IN}(+)-I_{IN}(-)$ V_{CM} = 0 V			3.0	25		5.0	50		5.0	50	nA
Input Common Mode Voltage Range	V+ = 30 V (3)		0		V+−1.5	0		V+−1.5	0		V+−1.5	V
Supply Current	R_L=∞	V^+=5 V		0.4	1		0.4	1		0.4	1.0	mA
		V^+=36 V		1	2.5		1	2.5		1	2.5	mA
Voltage Gain	R_L≥15 kΩ, V^+=15 V V_O = 1 V to 11 V		50	200		50	200		25	100		V/mV
Large Signal Response Time	V_{IN}=TTL Logic Swing, V_{REF}=1.4 V V_{RL}=5 V, R_L=5.1 kΩ			300			300			300		ns
Response Time	V_{RL}=5 V, R_L=5.1 kΩ (4)			1.3			1.3			1.5		µs
Output Sink Current	$V_{IN}(-)$=1 V, $V_{IN}(+)$=0, V_O≤1.5 V		6.0	16		6.0	16		6.0	16		mA
Saturation Voltage	$V_{IN}(-)$=1 V, $V_{IN}(+)$=0, I_{SINK}≤4 mA			250	400		250	400		250	400	mV
Output Leakage Current	$V_{IN}(-)$=0, $V_{IN}(+)$=1V, V_O=5 V			0.1			0.1			0.1		nA

(1) At output switch point, V_O=1.4V, R_S= 0 Ω with V^+ from 5V to 30V; and over the full input common-mode range (0V to V^+−1.5V), at 25°C.
(2) The direction of the input current is out of the IC due to the PNP input stage. This current is essentially constant, independent of the state of the output so no loading change exists on the reference or input lines.
(3) The input common-mode voltage or either input signal voltage should not be allowed to go negative by more than 0.3V. The upper end of the common-mode voltage range is V^+−1.5 V at 25°C, but either or both inputs can go to 36 V without damage, independent of the magnitude of V^+.
(4) The response time specified is for a 100 mV input step with 5 mV overdrive. For larger overdrive signals 300 ns can be obtained, see *LMx93 and LM193A Typical Characteristics* .

FIGURA 67: DATASHEET CARACTERISTICAS ELÉCTRICAS LM393

6.8 Electrical Characteristics: LMx93 and LM2903 (V+ = 5 V)(1)

PARAMETER	TEST CONDITIONS	LM193-N			LM293-N, LM393-N			LM290-N			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
Input Offset Voltage	See (2)			9			9		9	15	mV
Input Offset Current	$I_{IN(+)}-I_{IN(-)}$, V_{CM}=0 V			100			150		50	200	nA
Input Bias Current	$I_{IN}(+)$ or $I_{IN}(-)$ with Output in Linear Range, V_{CM}=0 V (3)			300			400		200	500	nA
Input Common Mode Voltage Range	V^+=30V (4)	0		V^+−2.0	0		V^+−2.0	0		V^+−2.0	V
Saturation Voltage	$V_{IN}(-)$=1V, $V_{IN}(+)$=0, I_{SINK}≤4 mA			700			700		400	700	mV
Output Leakage Current	$V_{IN}(-)$=0, $V_{IN(+)}$=1V, V_O=30 V			1.0			1.0			1.0	µA
Differential Input Voltage	Keep All V_{IN}'s≥0 V (or V^-, if Used), (5)			36			36			36	V

(1) These specifications are limited to −55°C≤T_A≤+125°C, for the LM193/LM193A. With the LM293 all temperature specifications are limited to −25°C≤T_A≤+85°C and the LM393 temperature specifications are limited to 0°C≤T_A≤+70°C. The LM2903 is limited to −40°C≤T_A≤+85°C.
(2) At output switch point, V_O=1.4V, R_S= 0 Ω with V^+ from 5V to 30V; and over the full input common-mode range (0V to V^+−1.5V), at 25°C.
(3) The direction of the input current is out of the IC due to the PNP input stage. This current is essentially constant, independent of the state of the output so no loading change exists on the reference or input lines.
(4) The input common-mode voltage or either input signal voltage should not be allowed to go negative by more than 0.3V. The upper end of the common-mode voltage range is V^+−1.5 V at 25°C, but either or both inputs can go to 36 V without damage, independent of the magnitude of V^+.
(5) Positive excursions of input voltage may exceed the power supply level. As long as the other voltage remains within the common-mode range, the comparator will provide a proper output state. The low input voltage state must not be less than −0.3V (or 0.3V below the magnitude of the negative power supply, if used).

FIGURA 68: DATASHEET CARACTERISTICAS ELÉCTRICAS (II) LM393

6.9 Typical Characteristics: LMx93 and LM193A

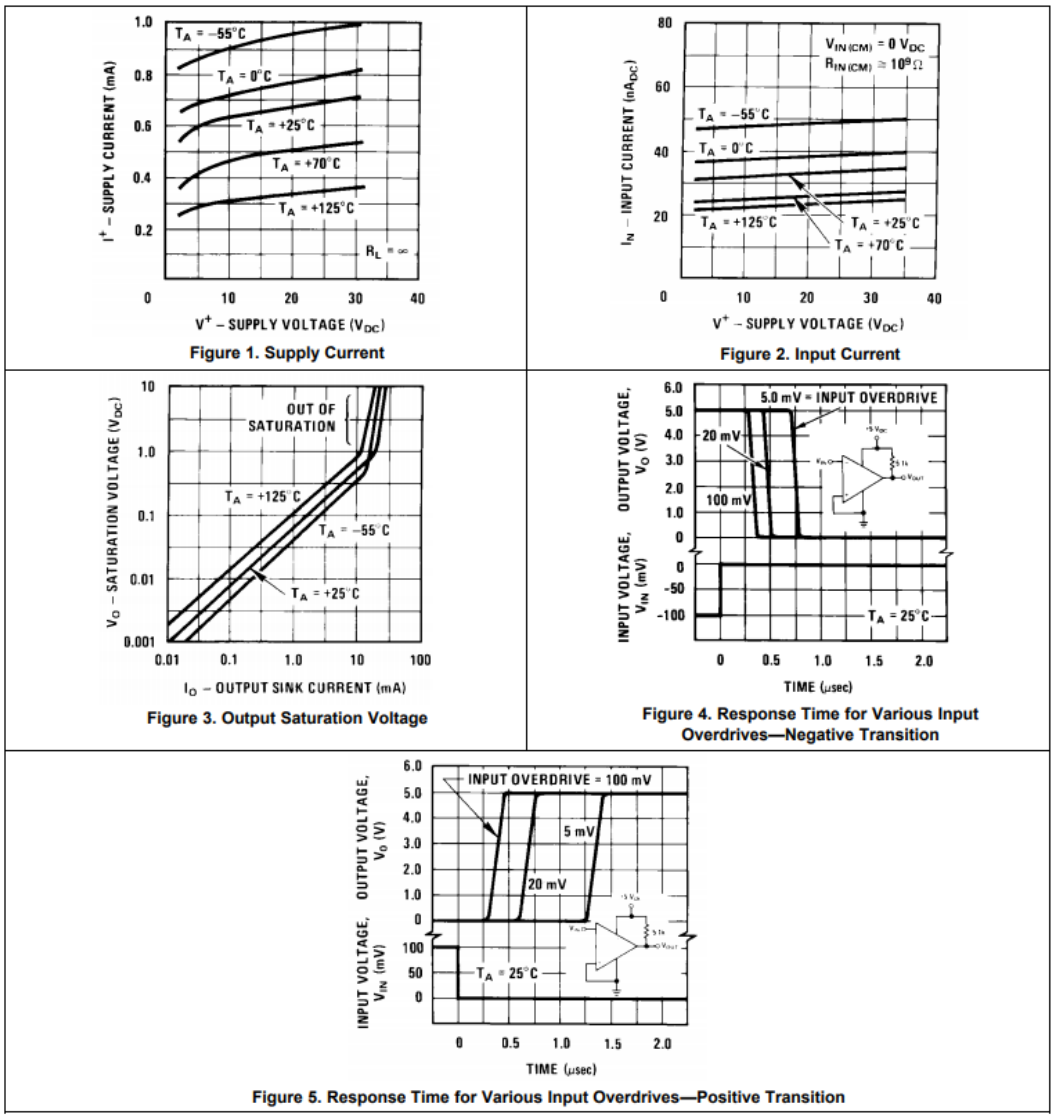


Figure 1. Supply Current

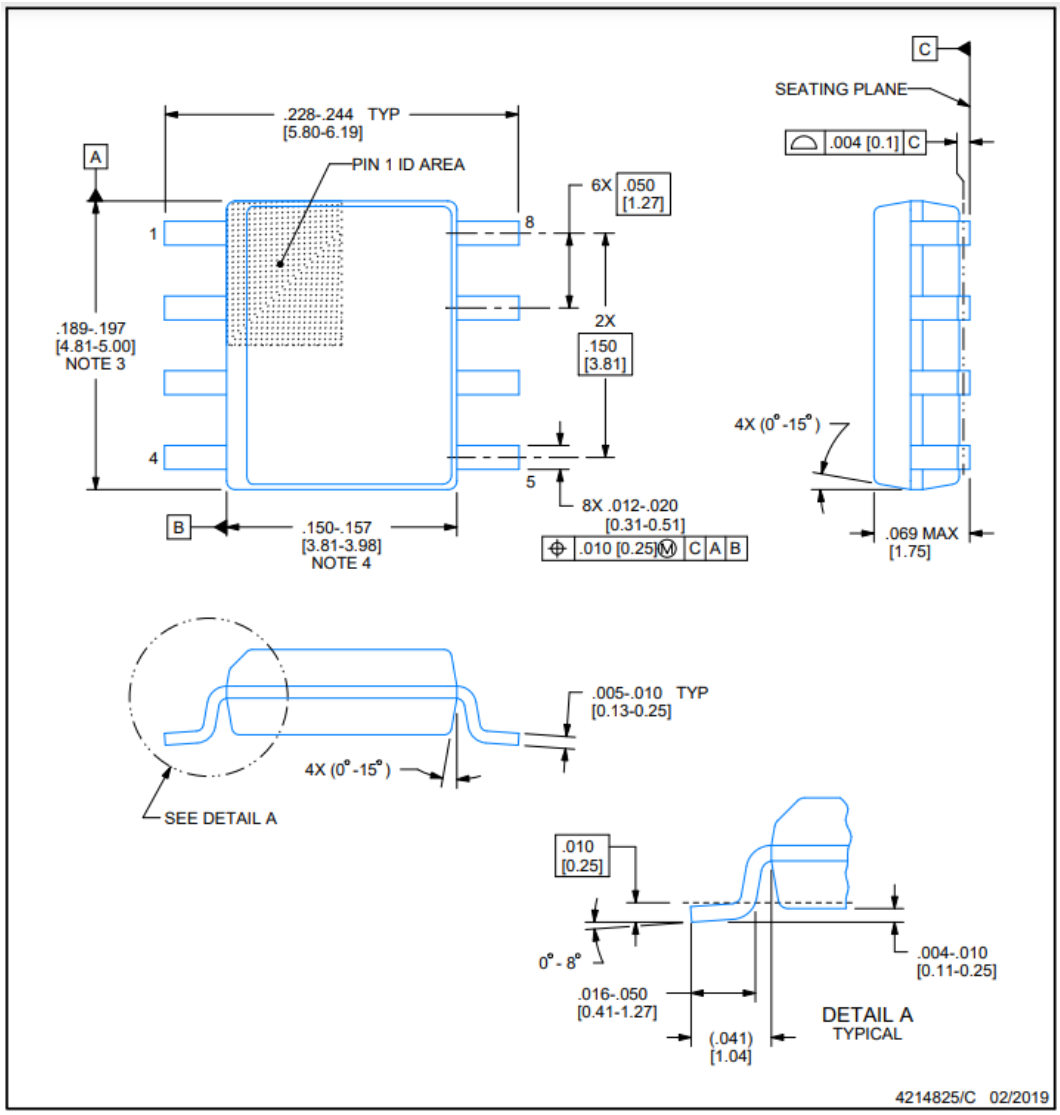
Figure 2. Input Current

Figure 3. Output Saturation Voltage

Figure 4. Response Time for Various Input Overdrives—Negative Transition

Figure 5. Response Time for Various Input Overdrives—Positive Transition

FIGURA 69: DATASHEET CARACTERÍSTICAS TÍPICAS LM393



NOTES:

1. Linear dimensions are in inches [millimeters]. Dimensions in parenthesis are for reference only. Controlling dimensions are in inches. Dimensioning and tolerancing per ASME Y14.5M.
2. This drawing is subject to change without notice.
3. This dimension does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed .006 [0.15] per side.
4. This dimension does not include interlead flash.
5. Reference JEDEC registration MS-012, variation AA.

FIGURA 70: DATASHEET EMPAQUETADO LM393

Sensor ‘Nivel de agua’:

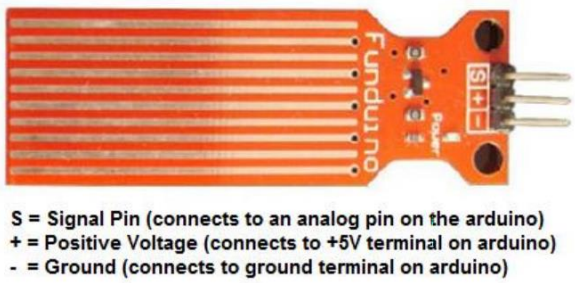


FIGURA 71: DATASHEET PINES SENSOR ‘NIVEL DE AGUA’

ESP8266-01S:

2.Specifications

Power
VCC-3.0-3.6V
Standby ~ 0.9uA
Running ~60-215mA,
Average ~ 80mA

Wifi Features
802.11 b/g/n
2.4GHz
WPA/WPA2
Wifi Direct
+20dBm output power (802.11b)

I/O Features
Integrated TCP/IP I
ntegrated TR switch, LNA,
balun

Memory/Speed Features
80MHz
64KB instruction RAM
96KB data RAM
64K boot ROM
1MB* Flash Memory

Basic Connection
VCC - 3.3V
GND - GND
TX - RX on Arduino or FTDI
RX - TX on ARduino or FTDI
Chip Enable - 3.3V

Default Baud Rate
11520* 8N1

LEDs
Red: Power
Blue: TX

*milage may vary on different version of the board

FIGURA 72: DATASHEET ESPECIFICACIONES ESP-01S

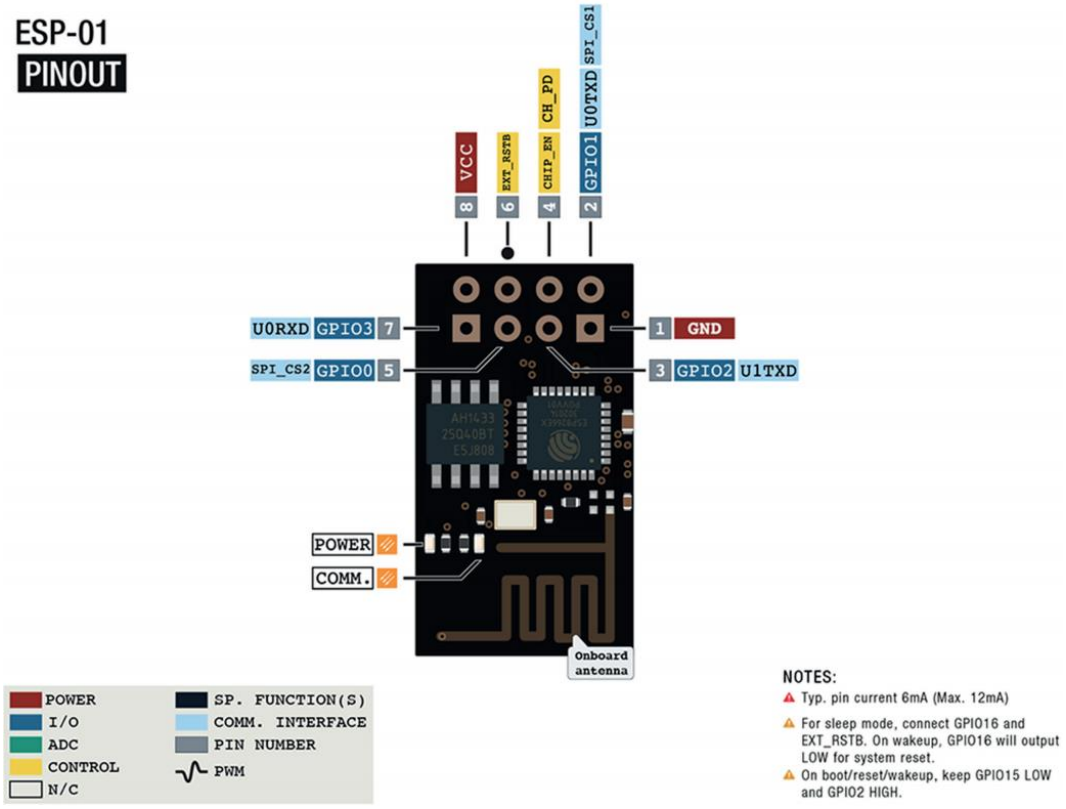


FIGURA 73: DATASHEET PINOUT ESP-01S

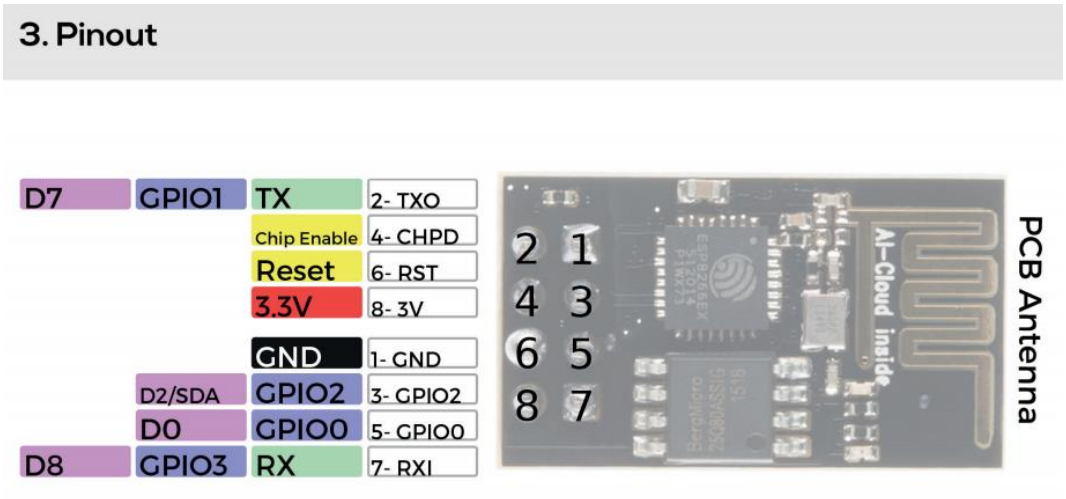
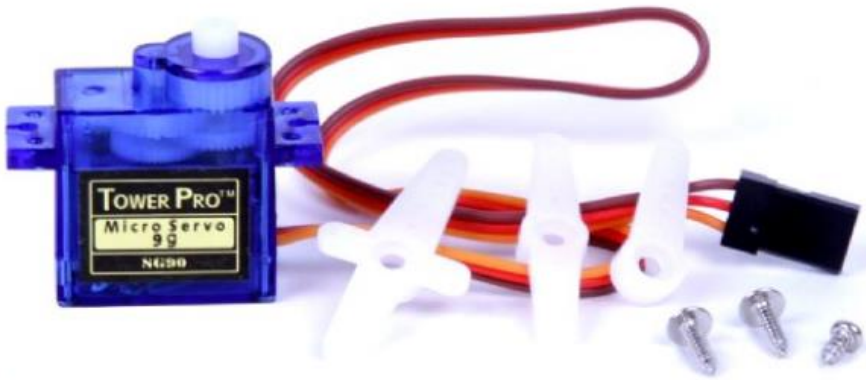


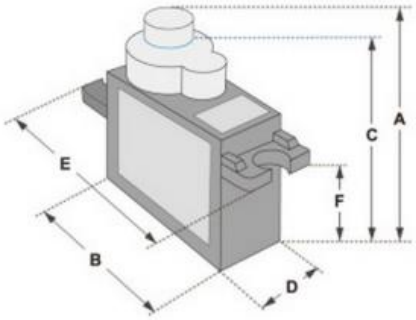
FIGURA 74: DATASHEET ENUMERACIÓN PINOUT ESP-01S

MicroServo SG90:

SERVO MOTOR SG90 **DATA SHEET**



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.



Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

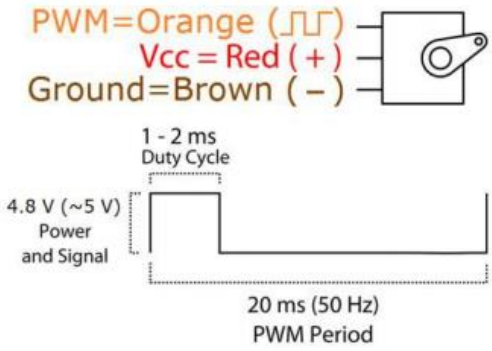


FIGURA 75: DATASHEET SERVO MOTOR SG90

Motor CC:

adafruit
INDUSTRIES

DC Toy / Hobby Motor - 130 Size

PRODUCT ID: 711

DESCRIPTION

These are standard '130 size' DC hobby motors. They come with a wider operating range than most toy motors: from 4.5 to 9VDC instead of 1.5-4.5V. This range makes them perfect for controlling with an Adafruit Motor Shield, or with an Arduino where you are more likely to have 5 or 9V available than a high current 3V setting. They'll fit in most electronics that already have 130-size motors installed and there's two breadboard-friendly wires soldered on already for fast prototyping

TECHNICAL DETAILS

- Operating Temperature: -10°C ~ +60°C
- Rated Voltage: 6.0VDC
- Rated Load: 10 g*cm
- No-load Current: 70 mA max
- No-load Speed: 9100 ±1800 rpm
- Loaded Current: 250 mA max
- Loaded Speed: 4500 ±1500 rpm
- Starting Torque: 20 g*cm
- Starting Voltage: 2.0
- Stall Current: 500mA max
- Body Size: 27.5mm x 20mm x 15mm
- Shaft Size: 8mm x 2mm diameter
- Weight: 17.5 grams

FIGURA 76: DATASHEET MOTOR CC

L293D:

TEXAS INSTRUMENTS

L293, L293D

SLRS008D – SEPTEMBER 1986 – REVISED JANUARY 2016

L293x Quadruple Half-H Drivers

1 Features

- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- High-Noise-Immunity Inputs
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

2 Applications

- Stepper Motor Drivers
- DC Motor Drivers
- Latching Relay Drivers

3 Description

The L293 and L293D devices are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN.

The L293 and L293D are characterized for operation from 0°C to 70°C.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
L293NE	PDIP (16)	19.80 mm × 6.35 mm
L293DNE	PDIP (16)	19.80 mm × 6.35 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Logic Diagram

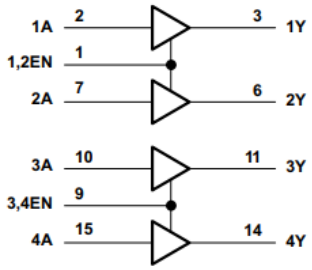
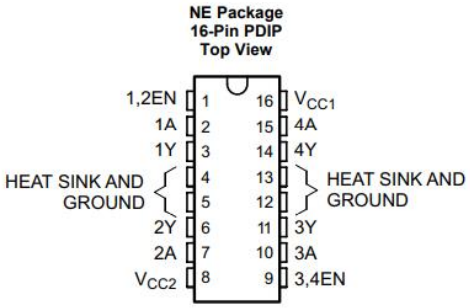


FIGURA 77: DATASHEET CARACTERÍSTICAS Y DIAGRAMA LÓGICO L293D

5 Pin Configuration and Functions



Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect to printed-circuit-board ground plane with multiple solid vias
V_{CC1}	16	—	5-V supply for internal logic translation
V_{CC2}	8	—	Power VCC for drivers 4.5 V to 36 V

FIGURA 78: DATASHEET CONFIGURACIÓN PINES L293D

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)

	MIN	MAX	UNIT
Supply voltage, V_{CC1}(2)		36	V
Output supply voltage, V_{CC2}		36	V
Input voltage, V_I		7	V
Output voltage, V_O	–3	V_{CC2} + 3	V
Peak output current, I_O (nonrepetitive, t ≤ 5 ms): L293	–2	2	A
Peak output current, I_O (nonrepetitive, t ≤ 100 µs): L293D	–1.2	1.2	A
Continuous output current, I_O: L293	–1	1	A
Continuous output current, I_O: L293D	–600	600	mA
Maximum junction temperature, T_J		150	°C
Storage temperature, T_{stg}	–65	150	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) All voltage values are with respect to the network ground terminal.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±2000	V
		Charged-device model (CDM), per JEDEC specification JESD22-C101(2)	±1000	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

			MIN	NOM	MAX	UNIT
	Supply voltage	V_{CC1}	4.5		7	V
		V_{CC2}	V_{CC1}		36	
V_{IH}	High-level input voltage	V_{CC1} ≤ 7 V	2.3		V_{CC1}	V
		V_{CC1} ≥ 7 V	2.3		7	V
V_{IL}	Low-level output voltage		–0.3(1)		1.5	V
T_A	Operating free-air temperature		0		70	°C

(1) The algebraic convention, in which the least positive (most negative) designated minimum, is used in this data sheet for logic voltage levels.

6.4 Thermal Information

THERMAL METRIC(1)		L293, L293D NE (PDIP) 16 PINS	UNIT
$R_{\theta JA}$	Junction-to-ambient thermal resistance (2)	36.4	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	22.5	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	16.5	°C/W
ψ_{JT}	Junction-to-top characterization parameter	7.1	°C/W
ψ_{JB}	Junction-to-board characterization parameter	16.3	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report, SPRA953.
(2) The package thermal impedance is calculated in accordance with JESD 51-7.

FIGURA 79: DATASHEET ESPECIFICACIONES L293D

6.5 Electrical Characteristics

over operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		L293: I_{OH} = −1 A		V_{CC2} − 1.8	V_{CC2} − 1.4		V
			L293D: I_{OH} = − 0.6 A					
V_{OL}	Low-level output voltage		L293: I_{OL} = 1 A			1.2	1.8	V
			L293D: I_{OL} = 0.6 A					
V_{OKH}	High-level output clamp voltage		L293D: I_{OK} = −0.6 A			V_{CC2} + 1.3		V
V_{OKL}	Low-level output clamp voltage		L293D: I_{OK} = 0.6 A			1.3		V
I_{IH}	High-level input current	A	V_I = 7 V			0.2	100	μA
		EN				0.2	10	
I_{IL}	Low-level input current	A	V_I = 0			−3	−10	μA
		EN				−2	−100	
I_{CC1}	Logic supply current		I_O = 0	All outputs at high level		13	22	mA
				All outputs at low level		35	60	
				All outputs at high impedance		8	24	
I_{CC2}	Output supply current		I_O = 0	All outputs at high level		14	24	mA
				All outputs at low level		2	6	
				All outputs at high impedance		2	4	

6.6 Switching Characteristics

over operating free-air temperature range (unless otherwise noted) V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output from A input	L293NE, L293DNE	C_L = 30 pF, See Figure 2		800		ns
		L293DWP, L293N L293DN			750		
t_{PHL}	Propagation delay time, high-to-low-level output from A input	L293NE, L293DNE			400		ns
		L293DWP, L293N L293DN			200		
t_{TLH}	Transition time, low-to-high-level output	L293NE, L293DNE			300		ns
		L293DWP, L293N L293DN			100		
t_{THL}	Transition time, high-to-low-level output	L293NE, L293DNE			300		ns
		L293DWP, L293N L293DN			350		

6.7 Typical Characteristics

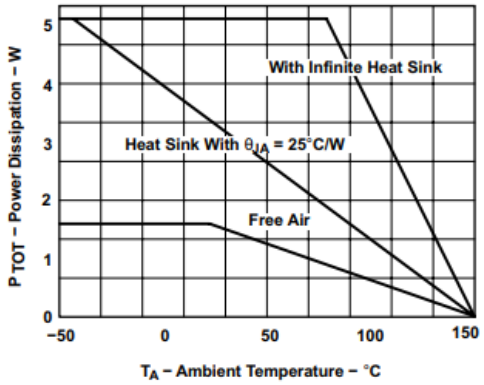


Figure 1. Maximum Power Dissipation vs Ambient Temperature

FIGURA 80: DATASHEET ESPECIFICACIONES (II) L293D

8.4 Device Functional Modes

Table 1 lists the fuctional modes of the L293x.

Table 1. Function Table (Each Driver)(1)

INPUTS (2)		OUTPUT (Y)
A	EN	
H	H	H
L	H	L
X	L	Z

(1) H = high level, L = low level, X = irrelevant, Z = high impedance (off)
(2) In the thermal shutdown mode, the output is in the high-impedance state, regardless of the input levels.

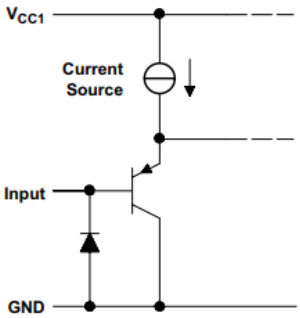


Figure 3. Schematic of Inputs for the L293x

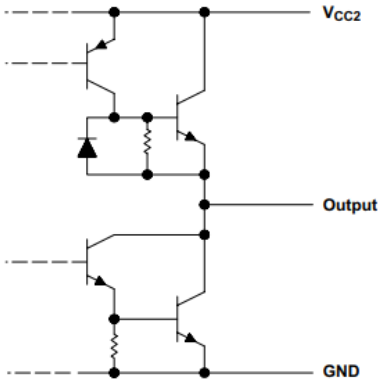


Figure 4. Schematic of Outputs for the L293

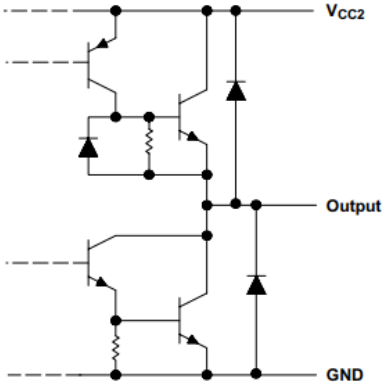


Figure 5. Schematic of Outputs for the L293D

FIGURA 81: DATASHEET MODOS FUNCIONALES L293D

11 Layout

11.1 Layout Guidelines

Place the device near the load to keep output traces short to reduce EMI. Use solid vias to transfer heat from ground pins to ground plane of the printed-circuit-board.

11.2 Layout Example

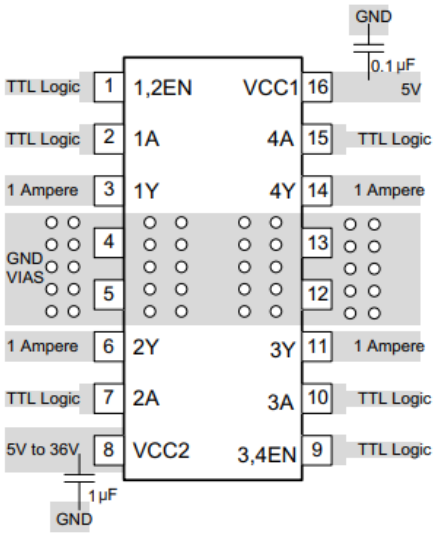


Figure 13. Layout Diagram

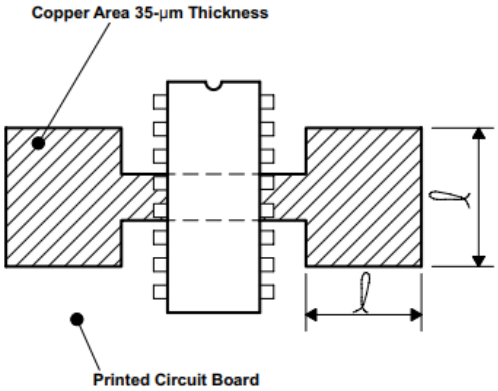


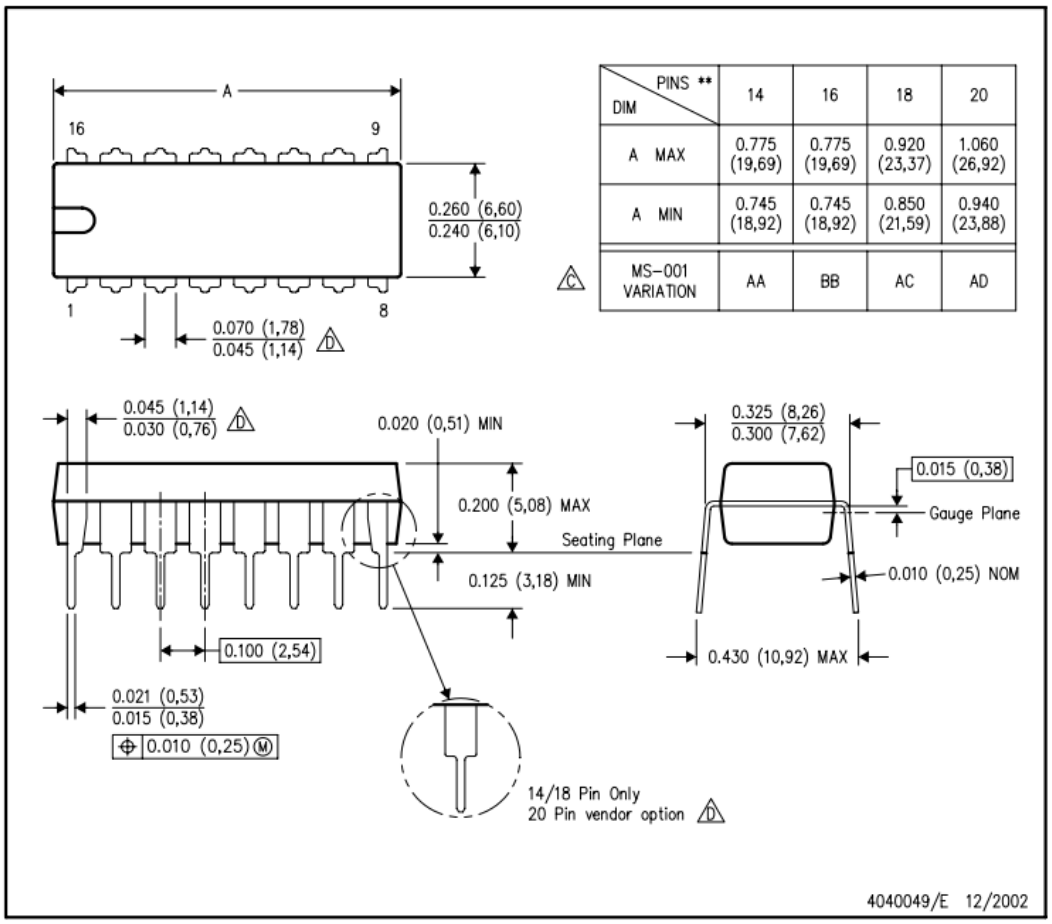
Figure 14. Example of Printed-Circuit-Board Copper Area (Used as Heat Sink)

FIGURA 82: DATASHEET DISEÑO L293D

MECHANICAL DATA

N (R–PDIP–T**) PLASTIC DUAL–IN–LINE PACKAGE

16 PINS SHOWN



DIM \ PINS **	14	16	18	20
A MAX	0.775 (19,69)	0.775 (19,69)	0.920 (23,37)	1.060 (26,92)
A MIN	0.745 (18,92)	0.745 (18,92)	0.850 (21,59)	0.940 (23,88)
MS–001 VARIATION	AA	BB	AC	AD

NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS–001, except 18 and 20 pin minimum body length (Dim A).
D. The 20 pin end lead shoulder width is a vendor option, either half or full width.

FIGURA 83: DATASHEET EMPAQUETADO L293D

B. Código:

"Configuraciones previas":

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 *   ___CONTROL DE UN SISTEMA INTELIGENTE DE REGADÍO___
 *                      ~~~ TFG ~~~
 *  Universdidad de Valladolid - Esc. Ing. Indus.
 *  Grado en Ing. Electrónica Indus. y Autom.
 *
 *  Guillermo Moreno Heranangómez
 *
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*----- PINES ESP8266 (Cara Superior) -----*/
// TX(o GPIO1) · · GND          _|_____¬
// CH-PD       · · GPIO2         |   ___|
// RST         · · GPIO0         | |___¬
// VCC(=3'3V)  · · RX(o GPIO3)   |     |
/*----- ----- ----- ----- ----- ----- -----*/

//////////////////MODO UART (CARGAR programa en la
memoria)//////////////////
/*----- CONEXIONES PARA CARGAR EL PROGRAMA ESP8266(_E) CON
ARDUINO(_A) MEGA-----*/
// TX_E -> TX_A
// RX_E -> RX_A
// VCC_E y CH-PD -> 3'3V
// GND_E y GPIO0 -> GND_A       ¡¡¡LOW!!!
// RST_E y GPIO2_E -> (Libres)
/*----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----
- -----*/

//////////////////MODO FLASH (EJECUTAR programa)//////////////////
/*----- CONEXIONES PARA FLASHEAR ESP8266(_E) CON ARDUINO(_A) MEGA
-----*/
// TX_E -> TX_A
// RX_E -> RX_A
// VCC_E y CH-PD -> 3'3V
// GND_E -> GND_A
// RST_E , GPIO2_E y GPIO0_E -> (Libres)
/*----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ----
- -----*/

/*** NOTA: GPIO2 y GPIO0 tienen Pull-Up interno; es decir, ***/
/*** desconectados por defecto están a HIGH (¡¡¡3'3V!!!)   ***/

//// LIBRERIAS:
  // Pantalla 20x4 con I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
  // Bibloteca del DHT11
#include <DHT.h>
  // Bibloteca del RTC
#include <RTClib.h>   // Reloj de Tiempo Real
  // Sensor DS18B20 (TEMP)
#include <OneWire.h>    // Contiene protocolo 1-wire
#include <DallasTemperature.h>    // Contiene las func. y conf.
```

```
  // Bibloteca del SG90 Servo Motor
#include <Servo.h>

// FUNCIONES EXTERNAS:
#include "EstacionHoraria.hpp"   // Función que establece la
estación del año
#include "SeleccionHum.hpp"   // Función que determina el estado
de la humedad del suelo
#include "GeneracionChar.hpp"   // Función que genera un caracter
nuevo

#define PUERTO_SERIE 1 //1: Envia datos; 0: No envia datos
#define CONFIG_RTC 0  //1: Busca y establece la hora del PC; 0: No
la busca
  //NOTA: Es necesario cargar 2 veces el programa (la primera vez
de todas)
  //  La 1ª(con 1) -> Busca y establece fecha-hora ACTUAL del PC
  //  La 2ª(con 0) -> No busca la fecha-hora, deben estar ya
actualizadas
  //                  (con el paso previo), y se puede implementar
sin error

// DEFINICIÓN DE PINES:
const int LED_Rojo = 27;
const int LED_Verde = 23;
const int LED_Amarillo = 25;
const int LED_Test = 13;   // <- LED incluido en Arduino MEGA
// Asignación de los pines a utilizar:
const int Sensor_Lluvia = A7;
const int tipoDHT = 11;       //  Tipo de sensor DHT: 11
const int sensorDHT = 31;     // Salida digital: pin31
const int TEMP_Cable1 = 2;    // Sensor Temp Cesped
const int TEMP_Cable2 = 3;    // Sensor Temp Rosales
    // A0 y A1 para el Cesped (Sensores Hum)
const int sensorKY70_1 = A0;  // Etiquetado [1]
const int sensorKY70_2 = A1;  // Etiquetado [2]
    // A3 y A4 para los Rosales (Sensores Hum)
const int sensorKY70_3 = A3;  // Etiquetado [3]
const int sensorKY70_4 = A4;  // Etiquetado [4]

/*----- PATILLAS CHIP L293D -----*/
//            °
// Enable1  ·       · +V
// In1      ·       · In4
// Out1     ·       · Out4
// 0V       · L293D · 0V
// 0V       ·       · 0V
// Out2     ·       · Out3
// In2      ·       · In3
// +V_Motor ·       · Enable2
/*----- ----- ----- ----- -----*/
/*** NOTA: Enable1 controla Out1 y Out2, y Enable2 controlo Out3 y
Out4; ***/
/*** por esa razón, es posible controlar 2 motores a la vez con un
chip  ***/
// Pines riego:
      // Simulado por un Motor CC (Ventilador)
const int Enable1_L293D = 9;       //Enable1 chip L293D, patilla 1
-> pin9 (PWM)
```

```
const int dirDrch1_L293D = 10;      //In1 chip L293D, patilla 3 ->
pin10 (PWM)
const int dirIzq1_L293D = 11;       //In2 chip L293D, patilla 6 ->
pin11 (PWM)
      // Simulado por un Servo Motor (SG90)
const int pinServo = 5;             //Pulsos del Servo Motor

// VARIABLES GLOBALES:
// Variables contador:
int i;     // Variable contador
int n;     // Variable (contador) sensor humedad
// Variables del Detector de agua:
int sensLluvia;    // De 0 a 330 (de Seco a Empapado)
// Variables del sensor DHT11: Temperatura y Humedad ambientes
float tempAmb, humAmb;
// Variables de DS18B20: Temperaturas
float temp1, temp2;
// Variables de KY70: Humedad
int humedad_1, humedad_2, humedad_3, humedad_4;  // Valores
analógicos
int hum1, hum2, hum3, hum4;                      // Valores en %
char *selH1, *selH2, *selH3, *selH4;                // Estado de
humedad
// Variables de RTC: hora
int h;
int epoca;
// Variables para la pag web:
String webpage, webpage1, webpage2;    // Máx 2048 caracteres a
enviar cada
// Variables de testeo:
bool Testear=true;
bool config_ini=true;
// Variables de riego:
bool franja=true;
bool riego_cesped=false;     // <- Sensores: cable amarillo-verde y
cable1
bool riego_rosales=false;    // <- Sensores: cable morado-azul y
cable2
    // Valores actuales en millis() al empezar a regar:
unsigned long regando_cesped=0;
unsigned long regando_rosales=0;
    // Diferencia en millis() al empezar a regar:
unsigned long contador_cesped=0;
unsigned long contador_rosales=0;

// OBJETOS CREADOS:
// Asignación del nombre del objeto
//    DHT(uint8_t pin, uint8_t type, uint8_t count = 6);
DHT dht(sensorDHT, tipoDHT);

LiquidCrystal_I2C lcd_I2C(0x27,20,4); // Conf. el LCD: direc.
0x27, 20 colum. y 4 líneas

RTC_DS3231 rtc; // Reloj

DateTime ahora; // Fecha y hora

OneWire cableTemp1(TEMP_Cable1);   // Bus de datos 'OneWire': pin2
```

```
OneWire cableTemp2(TEMP_Cable2);   // Bus de datos 'OneWire': pin3

DallasTemperature sensorTemp1(&cableTemp1); //Se declara una
variable u objeto para nuestro sensor [C1]
DallasTemperature sensorTemp2(&cableTemp2); //Se declara una
variable u objeto para nuestro sensor [C2]

Servo ServoMotor;

// PARÁMETROS Enviar Datos & WiFi:
#define DEBUG true   // Para depurar código a la hora de 'Enviar
datos'

const char* nombre_red = "MiFibra-F84E";
const char* contrasenia_red = "Jz****E4";
// IP Privada -> http://192.168.43.63/

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

'Setup':

```
void setup()
{
  delay(500);
  Serial.begin(115200);
  Serial1.begin(115200);
  Serial.println("115200 EYYY");

  /*-----Config de pines:-----*/
  pinMode(sensorKY70_1, INPUT); //Pin SENSOR1
  pinMode(sensorKY70_2, INPUT); //Pin SENSOR2
  pinMode(sensorKY70_3, INPUT); //Pin SENSOR3
  pinMode(sensorKY70_4, INPUT); //Pin SENSOR4

    //Pines Goteo (rosales)
  pinMode(Enable1_L293D,OUTPUT);
  pinMode(dirDrch1_L293D,OUTPUT);
  pinMode(dirIzq1_L293D,OUTPUT);

  pinMode(pinServo,OUTPUT); //Pin Aspersores (césped)

  pinMode(LED_Rojo, OUTPUT);     //Pin LED Rojo
  pinMode(LED_Verde, OUTPUT);    //Pin LED Verde
  pinMode(LED_Amarillo, OUTPUT); //Pin LED Amarillo
  pinMode(LED_Test, OUTPUT);     //Pin LED Testeo
    Serial.println("Pins y LEDs config.");

  /*-----Inicialización de los objetos de los distintos módulos:--
---*/
  dht.begin(); // Inicialización del sensor DHT11
  rtc.begin(); // Inicialización del RTC
  sensorTemp1.begin();   // Se inician los sensores de temp.
  sensorTemp2.begin();
  // Comprobación física de objetos inicializados conectando,
además, Servo:
  ServoMotor.attach(pinServo);  //Conecta el pin9 con la línea de
control
```

```
    ServoMotor.write(90);       //Mueve el Servo a la posición
central -> Las "12h"
    delay(1000);
    ServoMotor.write(-90);      //Mueve el Servo a la posición izq
-> Las "9h"

  lcd_I2C.begin();
    Serial.println("LCD iniciada");
  lcd_I2C.backlight();
    Serial.println("LCD luz");
    delay(1000);

/*-----PORTADAS LCD:-----*/
  generarChar(lcd_I2C,"EII");;          // Genera y representa
logo EII
  lcd_I2C.setCursor ( 7, 0 );
  lcd_I2C.print("Escuela de");
  lcd_I2C.setCursor ( 7, 1 );
  lcd_I2C.print("Ingenierias");
  lcd_I2C.setCursor ( 7, 2 );
  lcd_I2C.print("Industriales");
  lcd_I2C.setCursor ( 0, 3 );
  lcd_I2C.print("Trabajo Fin de Grado");
  delay(3000);

  lcd_I2C.clear();
  generarChar(lcd_I2C,"UVa");           // Genera y representa logo
UVa
  lcd_I2C.setCursor ( 7, 0 );
  lcd_I2C.print("Universidad");
  lcd_I2C.setCursor ( 6, 1 );
  lcd_I2C.print("de Valladolid");
  lcd_I2C.setCursor ( 11, 2 );
  lcd_I2C.print("2020");
  lcd_I2C.setCursor ( 0, 3 );
  lcd_I2C.print("Trabajo Fin de Grado");
  delay(3000);

  /*-----Comprobación RTC:-----*/
  //  Se actualiza, si procede, la hora para su configuración
  if (CONFIG_RTC == 1){
    rtc.adjust(DateTime(__DATE__,__TIME__));  //Establece hora
ACTUAL del PC

    Serial.println("¡¡¡Fecha y hora actualizadas!!!");
    lcd_I2C.setCursor ( 0, 1 );
    lcd_I2C.print("    Actualizado!    ");
    delay(1000);  // 1 seg. para apreciarlo
  }

  //SISTEMA CONECTADO:
  //  Se comprueba si el RTC está conectado correctamente
  if (!rtc.begin())
  {
    lcd_I2C.clear();
    lcd_I2C.setCursor(4, 1);
    lcd_I2C.print("!!!ERROR:!!!");
    lcd_I2C.setCursor(2, 2);
```

```
    lcd_I2C.print("RTC no detectado");
    #if PUERTO_SERIE
        Serial.println("ERROR: RTC no detectado");
    #endif  // Cierre de PUERTO_SERIE

    delay(2000);  // 2 seg. para apreciarlo
    lcd_I2C.clear();

    while(!rtc.begin())
    {
      lcd_I2C.setCursor(1, 1);
      lcd_I2C.print("RTC no disponible!");
      generarChar(lcd_I2C,"Warning!");    // Genera y representa
logo Warning ⚠
      delay(2000);
    }
  }

  //  Se establece la fecha y la hora actuales
  ahora = rtc.now();

  delay(2000);
   // Se borra la LCD
  lcd_I2C.clear();
    Serial.println("LCD limpiada");
    Serial.println(" ");

  /*-----Creación de Caracteres:-----*/
    generarChar(lcd_I2C,"0");   // Genera el comando: ° (0)
    generarChar(lcd_I2C,"1");   // Borde Horiz
    generarChar(lcd_I2C,"2");   // Borde Vert
    generarChar(lcd_I2C,"3");   // Esquina Sup Izq
    generarChar(lcd_I2C,"4");   // Esquina Sup Drch
    generarChar(lcd_I2C,"5");   // Esquina Inf Izq
    generarChar(lcd_I2C,"6");   // Esquina Inf Drch

  /*-----Config CONEXIÓN WiFi:-----*/
    lcd_I2C.setCursor ( 0, 1 );
    lcd_I2C.print("Conexion a traves de");
    lcd_I2C.setCursor ( 3, 2 );
    lcd_I2C.print("'Comandos AT':");
    lcd_I2C.setCursor ( 2, 0 );
    lcd_I2C.write(6);
    lcd_I2C.setCursor ( 17, 0 );
    lcd_I2C.write(5);
    lcd_I2C.setCursor ( 2, 3 );
    lcd_I2C.write(4);
    lcd_I2C.setCursor ( 17, 3 );
    lcd_I2C.write(3);

  sendData("AT\r\n",1000,DEBUG);      //Test de comunicación
  sendData("AT+GMR\r\n",1000,DEBUG);  //Versión del firmware
  //delay(1000);
//  sendData("AT+RST\r\n",1000,DEBUG);  //Reiniciar módulo
  sendData("AT+CWMODE=?\r\n",1000,DEBUG); //Modos de operación que
soporta
  delay(5000);

  sendData("AT+CWMODE=3\r\n",1000,DEBUG); //Modo de operación
```

```
      //Modos: 1-Estación (Cliente), 2-Punto de Acceso (Servidor)
y 3-Ambas
  sendData("AT+CIPMUX=1\r\n",1000,DEBUG); //N° de
conexiones/clientes (?)
  sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); //Función de
Servidor por el puerto 80
  delay(10000);
  sendData("AT+CWLAP\r\n",1000,DEBUG);  //Lista de AccessPoint
disponibles
  delay(5000);
  /* sendData("AT+CWJAP=\"nombre-red\",\"contraseña-
red\"\r\n",1000,DEBUG); */
  sendData("AT+CWJAP=\"MiFibra-
F84E\",\"Jz9ge3E4\"\r\n",1000,DEBUG);
      // Red WiFi, Usuario y Clave
    lcd_I2C.clear();
    lcd_I2C.setCursor ( 0, 0 );
    //lcd_I2C.print("╔*~~~*~~~**~~~*~~~*╗");
    lcd_I2C.write(3);   // Esquina Sup Izq
    for(i=0; i<=17; i++)
    {
      lcd_I2C.write(1); // Borde Horiz
    }
    lcd_I2C.write(4);   // Esquina Sup Drch
    lcd_I2C.setCursor ( 0, 1 );
    //lcd_I2C.print("║                  ║");
    lcd_I2C.write(2);   // Borde Vert
    lcd_I2C.setCursor ( 19, 1 );
    lcd_I2C.write(2);   // Borde Vert
    lcd_I2C.setCursor ( 4, 1 );
    lcd_I2C.print(nombre_red);
    lcd_I2C.setCursor ( 0, 2 );
    //lcd_I2C.print("║                  ║");
    lcd_I2C.write(2);   // Borde Vert
    lcd_I2C.setCursor ( 19, 2 );
    lcd_I2C.write(2);   // Borde Vert
    lcd_I2C.setCursor ( 6, 2 );
    lcd_I2C.print(contrasenia_red);
    lcd_I2C.setCursor ( 0, 3 );
    //lcd_I2C.print("╚*~~~*~~~**~~~*~~~*╝");
    lcd_I2C.write(5);   // Esquina Inf Izq
    for(i=0; i<=17; i++)
    {
      lcd_I2C.write(1); // Borde Horiz
    }
    lcd_I2C.write(6);   // Esquina Inf Drch
  delay(5000);
  sendData("AT+CIFSR\r\n",1000,DEBUG);  //Dirección IP local
(Host)
  delay(5000);

  //  Se encienden los 3 LEDs a la vez durante 1 seg.
  //  De esta manera se determina que se ha establecido la
conexión
  digitalWrite(LED_Rojo, HIGH);
  digitalWrite(LED_Verde, HIGH);
  digitalWrite(LED_Amarillo, HIGH);
  digitalWrite(LED_Test, HIGH);
```

```
  delay(1000);
  digitalWrite(LED_Rojo, LOW);
  digitalWrite(LED_Verde, LOW);
  digitalWrite(LED_Amarillo, LOW);
  digitalWrite(LED_Test, LOW);
  delay(500);

}  // Cierre Bucle Setup
```

'Loop':

```
void loop()
{
  /*-----RTC:-----*/
  //  Se establece la fecha y la hora actuales
  ahora = rtc.now();

  // Mostrar por el PC fecha y hora
  #if PUERTO_SERIE
    Serial.println();
    Serial.print("  -·-");
    // FECHA:
    Serial.print(ahora.day(), DEC);
    Serial.print("/");
    Serial.print(ahora.month(), DEC);
    Serial.print("/");
    Serial.print(ahora.year(), DEC);
    Serial.print(" | ");
    // HORA:
    Serial.print(ahora.hour(), DEC);
    Serial.print(":");
    Serial.print(ahora.minute(), DEC);
    Serial.print(":");
    Serial.print(ahora.second(), DEC);
    Serial.println("-·-  ");
    Serial.println(" ");
  #endif  // Cierre de PUERTO_SERIE

  lcd_I2C.clear();

  // Imprimir por la LCD fecha y hora actuales
  lcd_I2C.setCursor ( 0, 0 );
  lcd_I2C.print(" MENU FECHA Y HORA: ");
  // FECHA:   (DD/MM/AAAA)
  lcd_I2C.setCursor ( 2, 2 );
  lcd_I2C.print("Fecha: ");
  lcd_I2C.print(ahora.day(), DEC);
  lcd_I2C.print("/");
  lcd_I2C.print(ahora.month(), DEC);
  lcd_I2C.print("/");
  lcd_I2C.print(ahora.year(), DEC);
  lcd_I2C.print(" ");
  // HORA:    (hh:mm:ss)
  lcd_I2C.setCursor ( 2, 3 );
  lcd_I2C.print("Hora:  ");
  lcd_I2C.print(ahora.hour(), DEC);
  lcd_I2C.print(":");
```

```
  lcd_I2C.print(ahora.minute(), DEC);
  lcd_I2C.print(":");
  lcd_I2C.print(ahora.second(), DEC);
      delay(2000);

  h=ahora.hour();
  lcd_I2C.setCursor ( 9, 1 );
  lcd_I2C.print(h);
  Serial.print("         ");
  Serial.print(h);
  Serial.println(" horas");
    delay(1000);

  // CONDICIONES DE TESTEO:
  // - Dentro de las horas establecidas:
  //     * Verano -> [6:00-9:00) y [20:00-23:00)
  //     * Primavera & Otoño -> [7:00-10:00) y [19:00-22:00)
  // - Testear recargando la Pág Web (Fuera de ellas tambien)

  epoca = EstAnual(ahora.day(),ahora.month());

  if( ((((6 <= h) && (h < 9)) || ((20 <= h) && (h < 23))) &&
epoca==2) ||
      ((((7 <= h) && (h < 10)) || ((19 <= h) && (h < 22))) &&
(epoca==1 || epoca==3)) ||
      (Testear==true) )
  {
    lcd_I2C.clear();
    digitalWrite(LED_Test, HIGH);

    if(Testear==false)
    {
      franja=true;
    }

    if(config_ini == true)
    {
      lcd_I2C.setCursor ( 2, 0 );
      for(i=0; i<=15; i++)
      {
        lcd_I2C.write(1); // Borde Horiz
      }
      lcd_I2C.setCursor ( 3, 1 );
      lcd_I2C.print("Configuracion");
      lcd_I2C.setCursor ( 6, 2 );
      lcd_I2C.print("inicial:");
      lcd_I2C.setCursor ( 2, 3 );
      for(i=0; i<=15; i++)
      {
        lcd_I2C.write(1); // Borde Horiz
      }
    }
    else if((Testear==true) && (franja==false))
    {
      // Página recargada desde el navegador
      lcd_I2C.setCursor ( 3, 0 );
      lcd_I2C.write(3);   // Esquina Sup Izq
        lcd_I2C.setCursor ( 7, 0 );
```

```
      lcd_I2C.write(1); // Borde Horiz
      lcd_I2C.setCursor ( 12, 0 );
      lcd_I2C.write(1); // Borde Horiz
    lcd_I2C.setCursor ( 16, 0 );
    lcd_I2C.write(4);   // Esquina Sup Drch
    lcd_I2C.setCursor ( 3, 1 );
    lcd_I2C.write(1); // Borde Horiz
    lcd_I2C.setCursor ( 5, 1 );
    lcd_I2C.print("!!TESTEO!!");
    lcd_I2C.setCursor ( 16, 1 );
    lcd_I2C.write(1); // Borde Horiz
    lcd_I2C.setCursor ( 3, 2 );
    lcd_I2C.write(5);   // Esquina Inf Izq
      lcd_I2C.setCursor ( 7, 2 );
      lcd_I2C.write(1); // Borde Horiz
      lcd_I2C.setCursor ( 12, 2 );
      lcd_I2C.write(1); // Borde Horiz
    lcd_I2C.setCursor ( 16, 2 );
    lcd_I2C.write(6);   // Esquina Inf Drch
    lcd_I2C.setCursor ( 2, 3 );
    lcd_I2C.print("(Pag. recargada)");
  }
  else
  {
    lcd_I2C.setCursor ( 0, 0 );
    lcd_I2C.write(2); // Borde Vert
    lcd_I2C.setCursor ( 8, 0 );
    for(i=0; i<=3; i++)
    {
      lcd_I2C.write(2); // Borde Vert
    }
    lcd_I2C.setCursor ( 19, 0 );
    lcd_I2C.write(2); // Borde Vert
    lcd_I2C.setCursor ( 0, 1);
    lcd_I2C.write(2); // Borde Vert
    lcd_I2C.setCursor ( 1, 1 );
    lcd_I2C.print("Estoy en la franja");
    Serial.println("  (| Estoy en la franja |)");
    Serial.println("");
    lcd_I2C.setCursor ( 19, 1);
    lcd_I2C.write(2); // Borde Vert
    lcd_I2C.setCursor ( 0, 2);
    lcd_I2C.write(2); // Borde Vert
    lcd_I2C.setCursor ( 8, 2 );
    for(i=0; i<=3; i++)
    {
      lcd_I2C.write(2); // Borde Vert
    }
    lcd_I2C.setCursor ( 19, 2);
    lcd_I2C.write(2); // Borde Vert
    lcd_I2C.setCursor ( 0, 3);
    lcd_I2C.write(2); // Borde Vert
    lcd_I2C.setCursor ( 8, 3 );
    for(i=0; i<=3; i++)
    {
      lcd_I2C.write(2); // Borde Vert
    }
    lcd_I2C.setCursor ( 19, 3);
    lcd_I2C.write(2); // Borde Vert
```

```
      franja=true;
    }
    delay(2000);

    Testear=false;
    digitalWrite(LED_Test, LOW);

    /*-----DHT 11:-----*/
    //  Se establece la temp. y hum. ambientales
    humAmb = dht.readHumidity();
    tempAmb = dht.readTemperature();

    // Mostrar por el PC temp. y hum. amb.
    #if PUERTO_SERIE
      Serial.print("    Temperatura: ");
      Serial.print(tempAmb);
      Serial.println("°C"); // Ej.: Temperatura: __°C
      Serial.print("    Humedad: ");
      Serial.print(humAmb);
      Serial.println("%");
      Serial.println(" ");
    #endif  // Cierre de PUERTO_SERIE

    generarChar(lcd_I2C,"7");  // Genera el comando: ª (7)
    // Imprimir por la LCD temp. y hum. amb.
    lcd_I2C.clear();
    lcd_I2C.setCursor ( 0, 0 );
    lcd_I2C.print("    T");
    lcd_I2C.write(7);   // &ordf (o 170) = 'ª'
    lcd_I2C.print("yHum Amb:    ");   // Titulo:"    TªyHum
Amb:    "
    lcd_I2C.setCursor ( 2, 2 );
    lcd_I2C.print("T");
    lcd_I2C.write(7);
    lcd_I2C.print(" amb: ");   // "Tª amb: "
    lcd_I2C.print(tempAmb);
    lcd_I2C.print((char)223);   // &deg (o 176) = '°'
    lcd_I2C.print("C");
    lcd_I2C.setCursor ( 2, 3 );
    lcd_I2C.print("Hum amb: ");
    lcd_I2C.print(humAmb);
    lcd_I2C.print("%");
        delay(2000);

    /*-----DS18B20:-----*/
    // Se establecen las Temp. actuales
    sensorTemp1.requestTemperatures();   //Se envía el comando
para leer la temperatura
    temp1= sensorTemp1.getTempCByIndex(0); //Se obtiene la
temperatura en °C (=0) o en K (=1)
    sensorTemp2.requestTemperatures();
    temp2= sensorTemp2.getTempCByIndex(0);

    // Mostrar por el PC las Temp. actuales
    #if PUERTO_SERIE
      Serial.print("  < Temperatura1= ");
      Serial.print(temp1);
```

```
      Serial.println(" °C >");
      Serial.print("  < Temperatura2= ");
      Serial.print(temp2);
      Serial.println(" °C >");
      Serial.println(" ");
    #endif  // Cierre de PUERTO_SERIE

    // Imprimir por la LCD las Temp. actuales
    lcd_I2C.clear();
    lcd_I2C.setCursor ( 0, 0 );
    lcd_I2C.print("   MENU TEMP(");
    lcd_I2C.print((char)223);
    lcd_I2C.print("C):   ");   // Titulo:"   MENU TEMP(°C):   "
    lcd_I2C.setCursor ( 2, 2 );
    lcd_I2C.print("Temp1: ");
    lcd_I2C.print(temp1);
    lcd_I2C.print(" ");
    lcd_I2C.print((char)223);
    lcd_I2C.print("C");
    lcd_I2C.setCursor ( 2, 3 );
    lcd_I2C.print("Temp2: ");
    lcd_I2C.print(temp2);
    lcd_I2C.print(" ");
    lcd_I2C.print((char)223);
    lcd_I2C.print("C");
        delay(2000);

    /*-----KY70:-----*/
    n=1;
    humedad_1 = analogRead(sensorKY70_1);   // Valora analógico
      selH1 = SelecHum(humedad_1, n);   // Determina el estado de
humedad
      hum1 = map(analogRead(sensorKY70_1), 0, 1023, 100, 0);  //
Valor en %

    n=2;
    humedad_2 = analogRead(sensorKY70_2);   // Valora analógico
      selH2 = SelecHum(humedad_2, n);   // Determina el estado de
humedad
      hum2 = map(analogRead(sensorKY70_2), 0, 1023, 100, 0);  //
Valor en %

    n=3;
    humedad_3 = analogRead(sensorKY70_3);   // Valora analógico
      selH3 = SelecHum(humedad_3, n);   // Determina el estado de
humedad
      hum3 = map(analogRead(sensorKY70_3), 0, 1023, 100, 0);  //
Valor en %

    n=4;
    humedad_4 = analogRead(sensorKY70_4);   // Valora analógico
      selH4 = SelecHum(humedad_4, n);   // Determina el estado de
humedad
      hum4 = map(analogRead(sensorKY70_4), 0, 1023, 100, 0);  //
Valor en %

      delay(4000);

    // Mostrar por el PC las Hum. actuales
```

```
#if PUERTO_SERIE
  Serial.println(" ");
  Serial.print("    # Hum1: ");
  Serial.print(humedad_1);
  Serial.print(" >> ");   // &raquo (o 187): '>>'
  Serial.print(hum1);
  Serial.println("%");
  Serial.print("    # Hum2: ");
  Serial.print(humedad_2);
  Serial.print(" >> ");   // &raquo (o 187): '>>'
  Serial.print(hum2);
  Serial.println("%");
  Serial.print("    # Hum3: ");
  Serial.print(humedad_3);
  Serial.print(" >> ");   // &raquo (o 187): '>>'
  Serial.print(hum3);
  Serial.println("%");
  Serial.print("    # Hum4: ");
  Serial.print(humedad_4);
  Serial.print(" >> ");   // &raquo (o 187): '>>'
  Serial.print(hum4);
  Serial.println("%");

  Serial.println(" ");
  Serial.println("- - - - - - - - - - - - - - -");
  Serial.println(" ");
#endif  // Cierre de PUERTO_SERIE

// Imprimir por la LCD las Hum. actuales
lcd_I2C.clear();
lcd_I2C.setCursor(0, 0);
lcd_I2C.print("Hum1: ");
lcd_I2C.print(hum1);
lcd_I2C.print("%");
lcd_I2C.setCursor(11, 0);
lcd_I2C.print(selH1);
  delay(200);

lcd_I2C.setCursor(0, 1);
lcd_I2C.print("Hum2: ");
lcd_I2C.print(hum2);
lcd_I2C.print("%");
lcd_I2C.setCursor(11, 1);
lcd_I2C.print(selH2);
  delay(200);

lcd_I2C.setCursor(0, 2);
lcd_I2C.print("Hum3: ");
lcd_I2C.print(hum3);
lcd_I2C.print("%");
lcd_I2C.setCursor(11, 2);
lcd_I2C.print(selH3);
   delay(200);

lcd_I2C.setCursor(0, 3);
lcd_I2C.print("Hum4: ");
lcd_I2C.print(hum4);
lcd_I2C.print("%");
```

```
    lcd_I2C.setCursor(11, 3);
    lcd_I2C.print(selH4);
      delay(2000);

    /*-----LLUVIA:-----*/
    sensLluvia=analogRead(Sensor_Lluvia);
    Serial.print("   //Nivel lluvia: ");
    Serial.print(sensLluvia);
    Serial.println("\\\\");
    if(sensLluvia>=280)
    {
      lcd_I2C.clear();
      lcd_I2C.setCursor(0, 1);
      lcd_I2C.print("¡¡Lluvia detectada!!");
      Serial.println("     ¡¡Lluvia detectada!!");
    }

    /*-----RIEGO:-----*/
    if(((hum1+hum2)/2<=28 && sensLluvia<=280 || (humAmb<=15 &&
tempAmb>=30)) && franja==true )
    {
      riego_cesped=true;
      digitalWrite(Enable1_L293D,HIGH);
    }
    if(((hum3+hum4)/2<=28 && sensLluvia<=280 || (humAmb<=15 &&
tempAmb>=30)) && franja==true )
    {
      riego_rosales=true;
    }
    lcd_I2C.clear();

    //  Encendido de Aspersores:
    if(riego_cesped == true)
    {
      digitalWrite(dirDrch1_L293D,HIGH);
      digitalWrite(dirIzq1_L293D,LOW);
      regando_cesped=millis();
      digitalWrite(LED_Verde, HIGH);
      Serial.println("    Aspersores = 'ON'");
      lcd_I2C.setCursor(1, 0);
      lcd_I2C.print("Aspersores = 'ON'");
      delay(500);
    }
    //  Encendido de Goteo:
    if(riego_rosales == true)
    {
      ServoMotor.write(90);
      regando_rosales=millis();
      digitalWrite(LED_Rojo, HIGH);
      Serial.println("    Goteo = 'ON'");
      lcd_I2C.setCursor(3, 3);
      lcd_I2C.print("Goteo == 'ON'");
      delay(500);
    }

    while( (riego_cesped == true && (millis()-
regando_cesped)<=6000) ||
           (riego_rosales == true && (millis()-
regando_rosales)<=12000) )
```

```
    {
      lcd_I2C.clear();

      if(riego_cesped == true && (millis()-regando_cesped)<=6000)
      {
        contador_cesped=millis()-regando_cesped;
        lcd_I2C.setCursor(1, 1);
        lcd_I2C.print("{{Regando cesped}}");
        Serial.print("  {{{{Regando cesped}}}} ");
        Serial.println(contador_cesped);
      }
      if(riego_rosales == true && (millis()-
regando_rosales)<=12000)
      {
        contador_rosales=millis()-regando_rosales;
        lcd_I2C.setCursor(1, 2);
        lcd_I2C.print("{Regando__rosales}");
        Serial.print("  {~{~Regando rosales~}~} ");
        Serial.println(contador_rosales);
      }
      delay(950);

        //  Apagado de Aspersores (mientras Goteo activado):
      if((millis()-regando_cesped)>=6000)   // 60000ms = 1 min
tiene que haber estado regando
      {                                                        //
Establezco 6s para hacer la prueba
        digitalWrite(dirDrch1_L293D,LOW);
        //digitalWrite(dirIzq1_L293D,LOW);   /* -> Empleamos
únicamente un sentido de giro para la simulación */
        digitalWrite(LED_Verde, LOW);
        riego_cesped=false;
        Serial.println("    Aspersores = 'OFF'");
        lcd_I2C.setCursor(0, 0);
        lcd_I2C.print("Aspersores /=/ 'OFF'");
      }
    }
    lcd_I2C.clear();

    //  Apagado de Aspersores:
    if(riego_cesped == false || (millis()-
regando_cesped)>=6000)   // 60000ms = 1 min tiene que haber estado
regando
    {                                                          //
Establezco 6s para hacer la prueba
      digitalWrite(dirDrch1_L293D,LOW);
      //digitalWrite(dirIzq1_L293D,LOW);   /* -> Empleamos
únicamente un sentido de giro para la simulación */
      digitalWrite(LED_Verde, LOW);
      riego_cesped=false;
      Serial.println("    Aspersores = 'OFF'");
      lcd_I2C.setCursor(0, 0);
      lcd_I2C.print("Aspersores /=/ 'OFF'");

      digitalWrite(Enable1_L293D,LOW);
    }
    //  Apagado de Goteo:
```

```
    if(riego_rosales == false || (millis()-
regando_rosales)>=12000)  // 120000ms = 2 min tiene que haber
estado regando
    {                                                          //
Establezco 12s para hacer la prueba
      ServoMotor.write(-90);
      digitalWrite(LED_Rojo, LOW);
      riego_rosales=false;
      Serial.println("    Goteo = 'OFF'");
      lcd_I2C.setCursor(2, 3);
      lcd_I2C.print("Goteo =/= 'OFF'");
    }
    Serial.println("");
    delay(2000);

  }   //<- FIN CONDICION DE TESTEO

  /*-----SERVIDOR:-----*/
  if(Serial1.available())  // Comprobación de que ESP está
enviando datos
  {
    if(Serial1.find("+IPD,"))   // Se comprueba si hay
Identificador asociado
    {
      delay(1000);

      digitalWrite(LED_Amarillo, HIGH);

      int conexionID = Serial1.read()-48;

      if((franja == true) || (config_ini == true))
      {
        config_ini = false;
        webpage = "<html><head><meta http-equiv=""refresh""
content=""5"">";  //Refresca la pag cada 5 seg
      }
      else
      {
        webpage = "<html><head><meta http-
equiv=""refresh"">";  //Refrescar para que cargue la pág.
      }
      webpage+="<title data-l10n-id=""newtab-page-title"">CONTROL
INTELIGENTE SISTEMA DE REGADÍO</title>";
      webpage+="<meta charset='utf-8'></head>";    // Establece el
lenguaje para que sea con acentos y 'º'.

      webpage+="<body bgcolor=""silver"">";
        webpage+="<div align=""center"">";
          webpage+="<a
href=""http://www.uva.es/export/sites/uva/"" title=""UVa
Universidad de Valladolid""><img alt=""UVa Universidad de
Valladolid""
src=""https://stic.uva.es/export/sites/stic/_imagenes/header.png""
></a>";
        webpage+="<h1><i><b><acronym
title=""Trabajo_Fin_de_Grado"">TFG</acronym></b>:~^~^CONTROL
INTELIGENTE SISTEMA DE REGADÍO:^~^~</i></h1>";
          webpage+="<a href=""https://eii.uva.es/"" title=""EII
Escuela de Ingenierías Industriales""><img alt=""EII Escuela de
```

```
Ingenierías Industriales - UVa""
src=""https://www.eii.uva.es/images/cabecera.jpg""></a></div>";

        webpage+="<div align=""center""><table><tr>";

        webpage+="<div align=""center"">";
          webpage+="<table bgcolor=""lime"" cellspacing=""6""
cellpadding=""6"" border=""3"" style=""margin: 0 auto;"">";
            webpage+="<tr bgcolor=""purple""><td colspan=""4""><h2
style=""text-align:center;""> <font color=""yellow""> FECHA Y
HORA: </font><hr color=""red"">";//<br>
              webpage+="<font color=""lightsalmon""> ···>>> ";
              webpage+=ahora.day();
              webpage+="/";
              webpage+=ahora.month();
              webpage+="/";
              webpage+=ahora.year();
              webpage+="  -  ";
              webpage+=ahora.hour();
              webpage+=":";
              webpage+=ahora.minute();
              webpage+=":";
              webpage+=ahora.second();
              webpage+=" <<<··· ";
              webpage+="</font></h2></td></tr>";
            webpage+="<tr><td> Temperatura ambiente:
</div></td><td>";
              webpage+=tempAmb;
              webpage+="°C </td>";
            webpage+="<tr><td> Humedad ambiente: </div></td><td>";
              webpage+=humAmb;
              webpage+="% </td></tr>";

            webpage1 = "<tr><td colspan=""4"">";
              webpage1+="<div align=""center""><table><tr>";
                webpage1+="<td><div align=""center""><img
alt=""Universidad de Valladolid""
src=""http://www.uva.es/resources/uva50/img/logoUVa.png""></div></
td>";
                webpage1+="<td><table bgcolor=""aqua""
border=""1""><tr><td><i><u>Césped</u></i></td><td><i><u>Rosales</u
></i></td></tr>";
                  webpage1+="<tr><td colspan=""2""><b><div
style=""text-align:center;""> Temperatura: </div></b></td></tr>";
                    webpage1+="<tr><td>Cable 1: ";
                    webpage1+=temp1;
                    webpage1+="°C</td>";
                    webpage1+="<td>Cable 2: ";
                    webpage1+=temp2;
                    webpage1+="°C</td></tr>";
                  webpage1+="<tr><td colspan=""2""><b><div
style=""text-align:center;""> Humedad: </div></b></td></tr>";
                    webpage1+="<tr><td><b>1: </b>";
                      webpage1+=hum1;
                      webpage1+="%</td>";
                      webpage1+="<td><b>3: </b>";
                      webpage1+=hum3;
                      webpage1+="%</td></tr>";
```

```
                    webpage1+="<tr><td><b>2: </b>";
                      webpage1+=hum2;
                      webpage1+="%</td>";
                      webpage1+="<td><b>4: </b>";
                      webpage1+=hum4;
                      webpage1+="%</td></tr>";
                    webpage1+="</tr></table></td>";
                webpage1+="<td><div align=""center""><img
alt=""EII Escuela de Ingenierías Industriales""
src=""https://www.eii.uva.es/escuela/logos/logotipo_neg.jpg""
width=""240"" height=""150""></div></td>";
              webpage1+="</tr></table></div></td></tr></table></di
v></body></body></html>";

      digitalWrite(LED_Amarillo, LOW);
      delay(1000);
      digitalWrite(LED_Amarillo, HIGH);

      sendData("AT+CIPSEND=" + String(conexionID) + "," +
webpage.length() + "\r\n",500,true);
      sendData(webpage, 1000,true);

      delay(1000);

      sendData("AT+CIPSEND=" + String(conexionID) + "," +
webpage1.length() + "\r\n",500,true);
      sendData(webpage1, 1000,true);

      digitalWrite(LED_Amarillo, LOW);
      delay(1000);
      digitalWrite(LED_Amarillo, HIGH);

      sendData("AT+CIPCLOSE=" + String(conexionID) +
"\r\n",1000,true);

      digitalWrite(LED_Amarillo, LOW);

      if(franja==false)
      {
        Testear=true;
      }
      franja=false;
    }
  }

  /*-----MENÚ ACTUALIZACIÓN:-----*/
  // Mostrar durante 5 s. los datos
  //    La frecuencia del DHT11 hace que se tomen datos cada 5
seg.
  delay(4500);
  lcd_I2C.clear();
  lcd_I2C.setCursor ( 0, 1 );
  lcd_I2C.print("   (Actualizando)   ");
  delay(500);
  lcd_I2C.setCursor ( 0, 1 );
  lcd_I2C.print("                    ");

}   // Cierre Bucle Loop
```

Función 'sendData':

```
/*     ---FUNCIÓN DE ENVÍO DE DATOS:---
 * Envía cualquier dato o conjunto de comandos
 * al servidor ESP-01S, si se encuentra disponible,
 * que se ha de escribir o config. en la pág. web.
 * mediante la lectura de caracter a caracter.
/* --- . --- . --- . --- . --- . --- . --- . --- . --- */
String sendData(String command, const int timeout, boolean debug)
{
  String response = "";
  Serial1.print(command);    // Envia lo escrito para que lo lea el
ESP
  long int time = millis ();
  while( (time+timeout) > millis() )
  {
    while( Serial1.available() )
    {
      char c = Serial1.read();    // Lee el siguiente caracter
      response+=c;
    }
  }

  if(debug)
  {
    Serial.print(response);
  }
  return response;
}
```

Función 'EstacionHoraria':

```
/*             ---FUNCIÓN DE ESTACIÓN DEL AÑO:---
 * Establece en qué época del año se encuentra
 * y establece la franja horaria correspondiente.
/* --- . --- . --- . --- . --- . --- . --- . --- . --- */

// ESTACIONES DEL AÑO (Comienzos):
// - Primavera: 21 de Marzo     [= Equinoccio de Primavera]
// - Verano: 22 de Junio        [= Solsticio de Verano]
// - Otoño: 21 de Septiembre    [= Equinoccio de Otoño]
// - Invierno: 22 de Diciembre [= Solsticio de Invierno]
/* ~~~ CAMBIOS DE HORA: ~~~*/
/* Horario de Verano -> Último domingo de Marzo, se ADELANTA la
hora   */
/* Horario de Invierno -> Último domingo de Octubre, se ATRASA la
hora */

// Pag. web donde vislumbrar los datos de la región a lo largo del
año,
// estableciendose la localidad más cercana para ello:
//   http://www.aemet.es/es/serviciosclimaticos/vigilancia_clima/a
nalisis_estacional?l=2465
//   (enlace correspondiente a la Temperatura de la estación
actual para la localidad de Segovia)
```

```
int EstAnual(int dia, int mes)
{
   int estacion;

   if((mes > 3) && (mes <= 6))  //PRIMAVERA [1]
   {
      if((mes = 6) && (dia >= 22))
      {
        Serial.println("  #*# Estación: VERANO #*#");
        Serial.println("");
        estacion = 2;
      }
      else
      {
        Serial.println("  #*# Estación: PRIMAVERA #*#");
        Serial.println("");
        estacion = 1;
      }

   }
   else if((mes > 6 ) && (mes <= 9))  //VERANO [2]
   {
      if((mes = 9) && (dia >= 21))
      {
        Serial.println("  #*# Estación: OTONO #*#");
        Serial.println("");
        estacion = 3;
      }
      else
      {
        Serial.println("  #*# Estación: VERANO #*#");
        Serial.println("");
        estacion = 2;
      }
   }
   else if((mes > 9) && (mes <= 12))  //OTOÑO [3]
   {
      if((mes = 12) && (dia >= 22))
      {
        Serial.println("  #*# Estación: INVIERNO #*#");
        Serial.println("");
        estacion = 4;
      }
      else
      {
        Serial.println("  #*# Estación: OTONO #*#");
        Serial.println("");
        estacion = 3;
      }
   }
   else if(mes <= 3)  //INVIERNO [4]
   {
      if((mes = 3) && (dia >= 21))
      {
        Serial.println("  #*# Estación: PRIMAVERA #*#");
        Serial.println("");
        estacion = 1;
      }
      else
```

```
        {
          Serial.println("  #*# Estación: INVIERNO #*#");
          Serial.println("");
          estacion = 4;
        }
    }

    return estacion;
}
```

Función 'GeneracionChar':

```
/*              ---FUNCIÓN DE CREACIÓN DE UN CARACTTER:---
 * Genera un nuevo caracter o conjunto de caracteres a partir
 * de una matriz de byte de 5x8.
 * Luego, si es un conjunto, lo representa.
/* --- . --- . --- . --- . --- . --- . --- . --- . --- */
//// LIBRERIAS:
  // Pantalla 20x4 con I2C
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

/*-------- Caracteres sueltos: --------*/
//  Matriz que genera el comando: ° (0)
byte grados[] = {
  B00000,
  B01110,
  B01010,
  B01110,
  B00000,
  B00000,
  B00000,
  B00000
};
//  Matriz que genera el comando: ª (7)
byte ord_fem[] = {
  B11100,
  B00010,
  B11110,
  B10110,
  B11101,
  B00000,
  B00000,
  B00000
};
//  Matriz que genera el borde Horizontal (1)
byte bordeHoriz[] = {
  B00000,
  B00000,
  B11011,
  B00100,
  B00100,
  B11011,
  B00000,
  B00000
};
//  Matriz que genera el borde Vertical (2)
```

```
byte bordeVert[] = {
  B00100,
  B01010,
  B01010,
  B00100,
  B00100,
  B01010,
  B01010,
  B00100
};
//  Matriz que genera esquina Superior Izq (3)
byte esqSupIzq[] = {
  B00000,
  B00000,
  B01111,
  B01000,
  B01000,
  B01011,
  B01010,
  B01010
};
//  Matriz que genera esquina Superior Drch (4)
byte esqSupDrch[] = {
  B00000,
  B00000,
  B11110,
  B00010,
  B00010,
  B11010,
  B01010,
  B01010
};
//  Matriz que genera esquina Inferior Izq (5)
byte esqInfIzq[] = {
  B01010,
  B01010,
  B01011,
  B01000,
  B01000,
  B01111,
  B00000,
  B00000
};
//  Matriz que genera esquina Inferior Drch (6)
byte esqInfDrch[] = {
  B01010,
  B01010,
  B11010,
  B00010,
  B00010,
  B11110,
  B00000,
  B00000
};
/*-------------------------------------*/
/*--------*/
/*-------- LOGO EII pequeño: --------*/
//  Matrices que generan logo EII pequeño
// Van enumeradas de izquierda a derecha desde arriba a abajo
```

```
byte logoEIImini_11[] = {
  B00000,
  B00001,
  B00111,
  B01000,
  B01000,
  B11111,
  B11000,
  B11000
};
byte logoEIImini_12[] = {
  B11111,
  B11111,
  B11111,
  B01001,
  B01001,
  B11111,
  B01001,
  B01001
};
byte logoEIImini_13[] = {
  B00000,
  B10000,
  B11100,
  B10010,
  B10010,
  B11111,
  B10011,
  B10011
};
byte logoEIImini_21[] = {
  B11111,
  B11000,
  B01000,
  B01000,
  B00111,
  B00001,
  B00000,
  B00000
};
byte logoEIImini_22[] = {
  B11001,
  B01001,
  B01001,
  B01001,
  B11111,
  B11111,
  B11111,
  B00000
};
byte logoEIImini_23[] = {
  B10011,
  B10011,
  B10010,
  B10010,
  B11100,
  B10000,
  B00000,
```

```
  B00000
};
/*-------- LOGO UVa pequeño: --------*/
//  Matrices que generan logo UVa pequeño
// Van enumeradas de izquierda a derecha desde arriba a abajo
byte logoUVamini_11[] = {
  B11111,
  B11111,
  B11111,
  B11111,
  B11111,
  B10010,
  B10010,
  B10010
};
byte logoUVamini_12[] = {
  B11111,
  B11111,
  B11111,
  B11111,
  B11111,
  B10110,
  B10110,
  B10110
};
byte logoUVamini_13[] = {
  B11111,
  B11111,
  B11111,
  B11111,
  B11111,
  B11111,
  B10011,
  B11101
};
byte logoUVamini_21[] = {
  B10010,
  B10010,
  B11001,
  B11111,
  B11111,
  B11111,
  B11111,
  B11111
};
byte logoUVamini_22[] = {
  B10110,
  B11001,
  B11001,
  B11111,
  B11111,
  B11111,
  B11111,
  B11111
};
byte logoUVamini_23[] = {
  B10001,
  B10101,
  B10001,
```

```
  B11111,
  B11111,
  B11111,
  B11111,
  B11111
};
/*-------- LOGO Warning ⚠: --------*/
//  Matrices que generan logo ⚠ pequeño
// Van enumeradas de izquierda a derecha desde arriba a abajo
byte logoWarning_1[] = {
  B00001,
  B00011,
  B00010,
  B00110,
  B00110,
  B01111,
  B01110,
  B11111
};
byte logoWarning_2[] = {
  B10000,
  B11000,
  B01000,
  B01100,
  B01100,
  B11110,
  B01110,
  B11111
};
//// - - - - - -


void generarChar(LiquidCrystal_I2C lcd_I2C, char *CharACrear)
{
        /*-----Selección de Caracteres:-----*/
  if(CharACrear == "0")
  {
    lcd_I2C.createChar(0, grados);  // Genera el comando: ° (0)
    lcd_I2C.write(0);
  }
  else if(CharACrear == "1")
  {
    lcd_I2C.createChar(1, bordeHoriz);
    lcd_I2C.write(1);   // Borde Horiz
  }
  else if(CharACrear == "2")
  {
    lcd_I2C.createChar(2, bordeVert);
    lcd_I2C.write(2);   // Borde Vert
  }
  else if(CharACrear == "3")
  {
    lcd_I2C.createChar(3, esqSupIzq);
    lcd_I2C.write(3);   // Esquina Sup Izq
  }
  else if(CharACrear == "4")
  {
    lcd_I2C.createChar(4, esqSupDrch);
```

```
  lcd_I2C.write(4);   // Esquina Sup Drch
}
else if(CharACrear == "5")
{
  lcd_I2C.createChar(5, esqInfIzq);
  lcd_I2C.write(5);   // Esquina Inf Izq
}
else if(CharACrear == "6")
{
  lcd_I2C.createChar(6, esqInfDrch);
  lcd_I2C.write(6);   // Esquina Inf Drch
}
else if(CharACrear == "7")
{
  lcd_I2C.createChar(7, ord_fem);  // Genera el comando: ª (7)
  lcd_I2C.write(7);
}
else if(CharACrear == "EII")
{
  lcd_I2C.load_custom_character(1, logoEIImini_11);
  lcd_I2C.load_custom_character(2, logoEIImini_12);
  lcd_I2C.load_custom_character(3, logoEIImini_13);
  lcd_I2C.load_custom_character(4, logoEIImini_21);
  lcd_I2C.load_custom_character(5, logoEIImini_22);
  lcd_I2C.load_custom_character(6, logoEIImini_23);

  lcd_I2C.setCursor ( 2, 0 );
  lcd_I2C.write(1);
  lcd_I2C.write(2);
  lcd_I2C.write(3);
  lcd_I2C.setCursor ( 2, 1 );
  lcd_I2C.write(4);
  lcd_I2C.write(5);
  lcd_I2C.write(6);
}
else if(CharACrear == "UVa")
{
  lcd_I2C.load_custom_character(1, logoUVamini_11);
  lcd_I2C.load_custom_character(2, logoUVamini_12);
  lcd_I2C.load_custom_character(3, logoUVamini_13);
  lcd_I2C.load_custom_character(4, logoUVamini_21);
  lcd_I2C.load_custom_character(5, logoUVamini_22);
  lcd_I2C.load_custom_character(6, logoUVamini_23);

  lcd_I2C.setCursor ( 2, 0 );
  lcd_I2C.write(1);
  lcd_I2C.write(2);
  lcd_I2C.write(3);
  lcd_I2C.setCursor ( 2, 1 );
  lcd_I2C.write(4);
  lcd_I2C.write(5);
  lcd_I2C.write(6);
}
else if(CharACrear == "Warning!")
{
  lcd_I2C.load_custom_character(1, logoWarning_1);
  lcd_I2C.load_custom_character(2, logoWarning_2);

  lcd_I2C.setCursor ( 9, 2 );
```

```
      lcd_I2C.write(1);
      lcd_I2C.write(2);
    }
}
```

Función 'SeleccionHum':

```
/*              ---FUNCIÓN DE SELECCIÓN DE HUMEDAD:---
 * Determina en qué franja de humedad se encuentra
 * el sensor y establece la etiqueta correspondiente.
/* --- . --- . --- . --- . --- . --- . --- . --- . --- */

char *SelecHum(int humedad, int n)
{
   char *selec = NULL;
   if(humedad <= 250)
   {
      Serial.print("     · SensorH");
            Serial.print(n);
            Serial.println(": Empapado ");
            selec = " Empapado";
   }
   else if(humedad  > 250 && humedad <= 500)
   {
      Serial.print("     · SensorH");
            Serial.print(n);
            Serial.println(": Mojado   ");
            selec = " Mojado  ";
   }
   else if(humedad > 500 && humedad <= 750)
   {
      Serial.print("     · SensorH");
          Serial.print(n);
          Serial.println(": Humedo   ");
          selec = " Humedo  ";
   }
   else if(humedad > 750 && humedad <= 930)
   {
      Serial.print("     · SensorH");
          Serial.print(n);
          Serial.println(": Seco     ");
      selec = " Seco    ";
   }
   else if(humedad > 930)
   {
      Serial.print("     · SensorH");
            Serial.print(n);
            Serial.println(": Aire     ");
      selec = " Aire    ";
   }

  return selec;
}
```

C. Climatología AEMET:

En este anexo se añaden las capturas pertenecientes a la Temperatura y Precipitaciones de la página web de la Agencia Estatal de Meteorología (AEMET).

La información que se muestra es a partir del mes de Otoño 2019, aunque se hayan obtenido datos desde Primavera 2019, porque se reescribieron en la propia página por los de la época del nuevo año sin haberlos guardado previamente de forma correcta.

Temperatura:

Hay que indicar que la gráfica en rojo corresponde a las temperaturas máximas diarias en dicho periodo, mientras que la azul se refiere a las mínimas en el mismo tiempo.

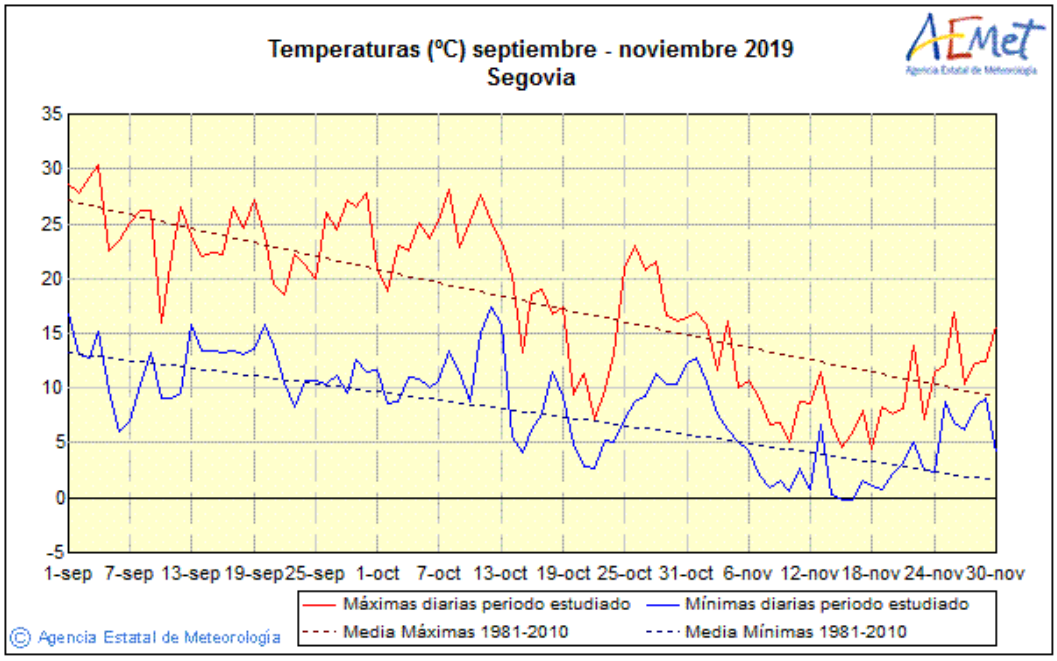


FIGURA 84: AEMET TEMP OTOÑO2019

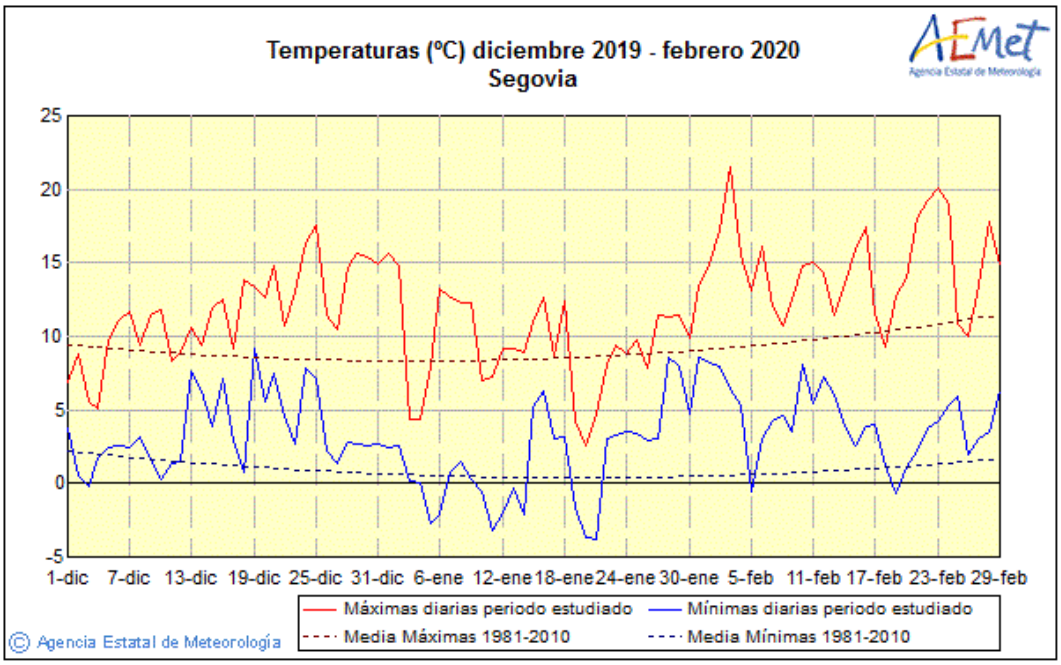


FIGURA 85: AEMET TEMP INVIERNO2019

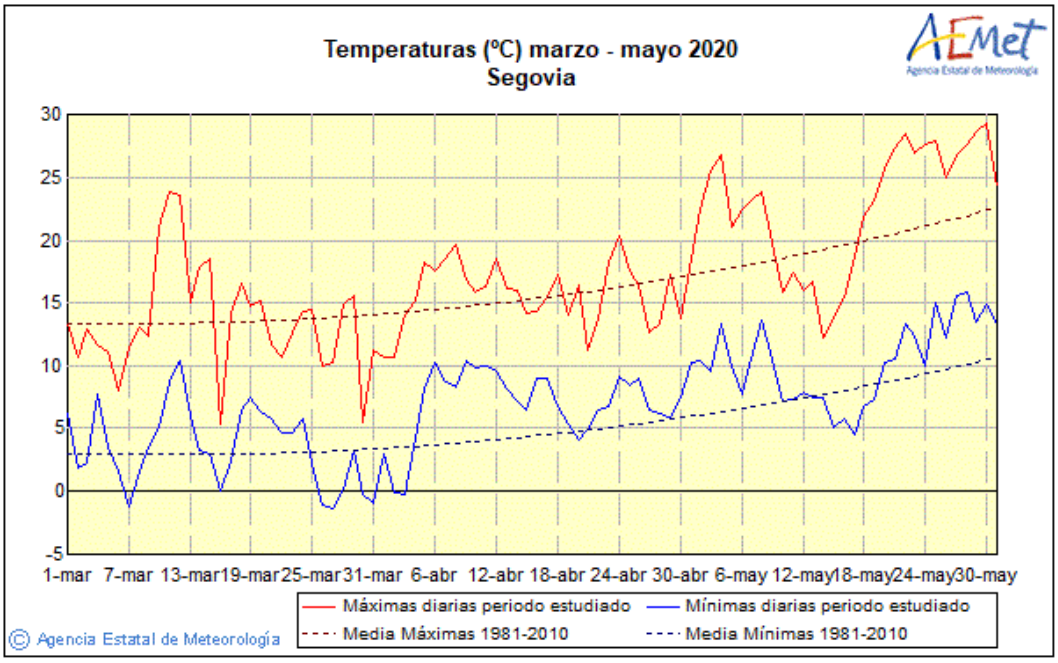


FIGURA 86: AEMET TEMP PRIMAVERA2020

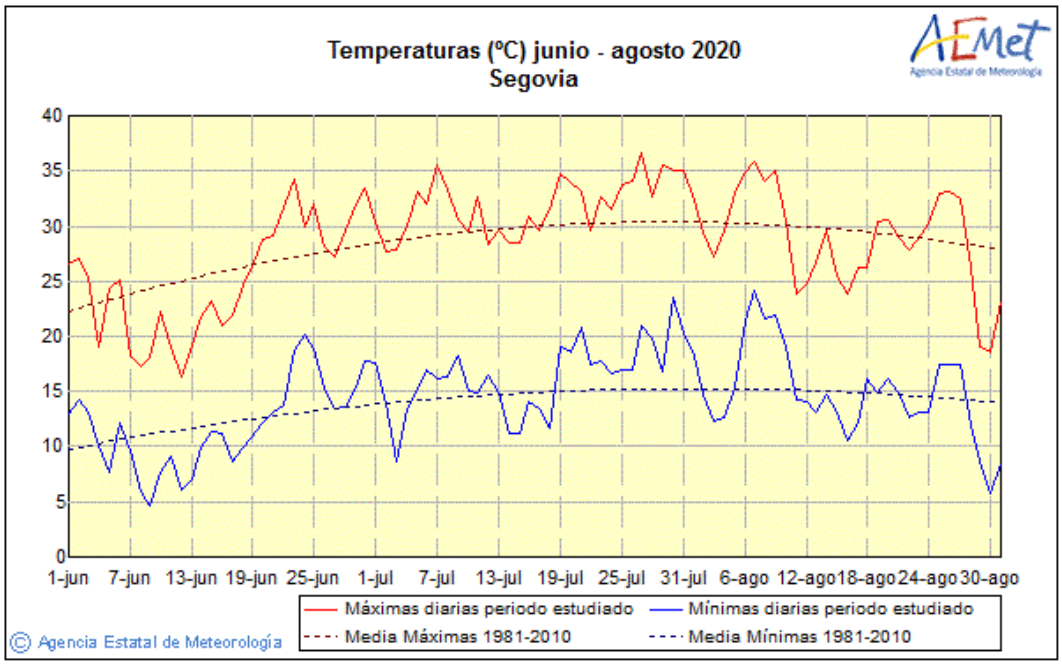


FIGURA 87: AEMET TEMP VERANO2020

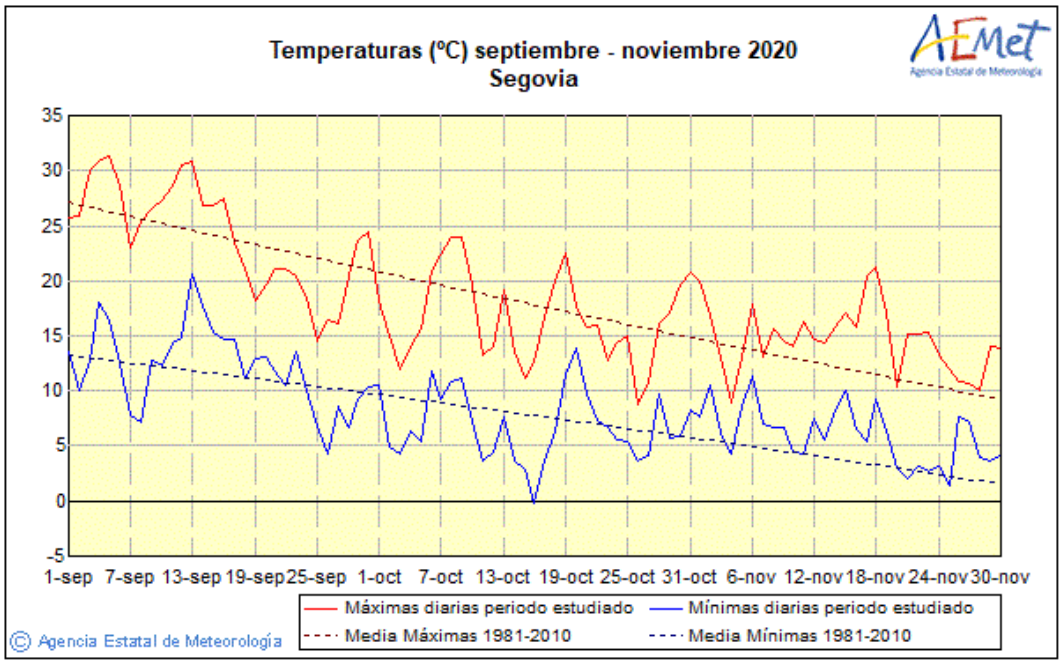


FIGURA 88: AEMET TEMP OTOÑO2020

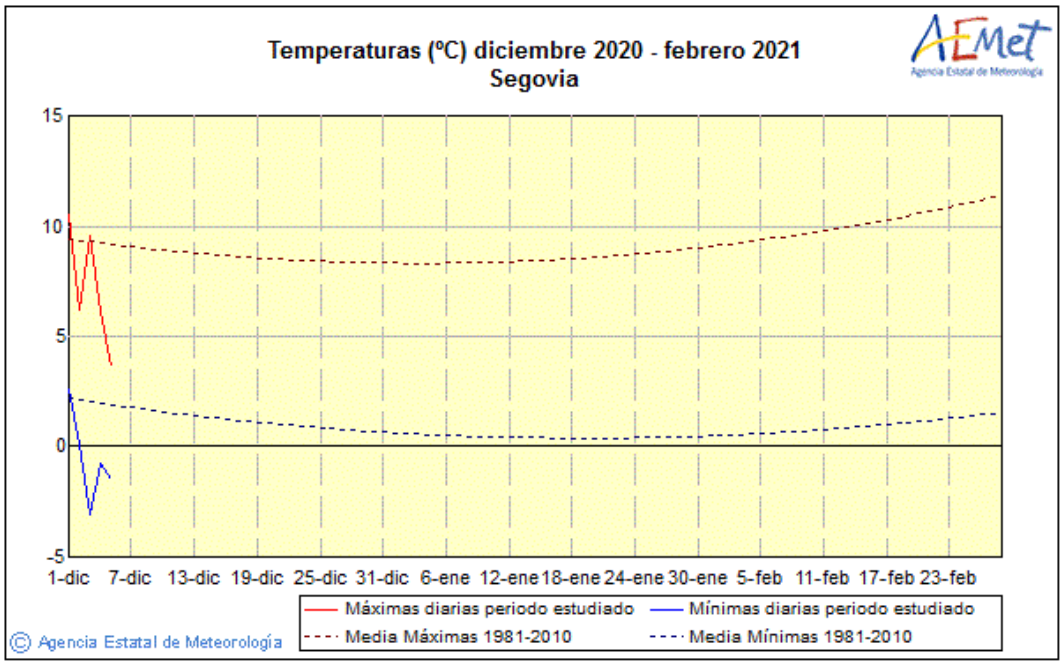


FIGURA 89: AEMET ESTIMACIÓN TEMP INVIERNO2020

Precipitaciones:

Se ha de señalar que la precipitación actual en ese momento concreto es la representada en rojo y la mediana establecida entre el periodo 1981 y 2010 se señala mediante las barras azules.

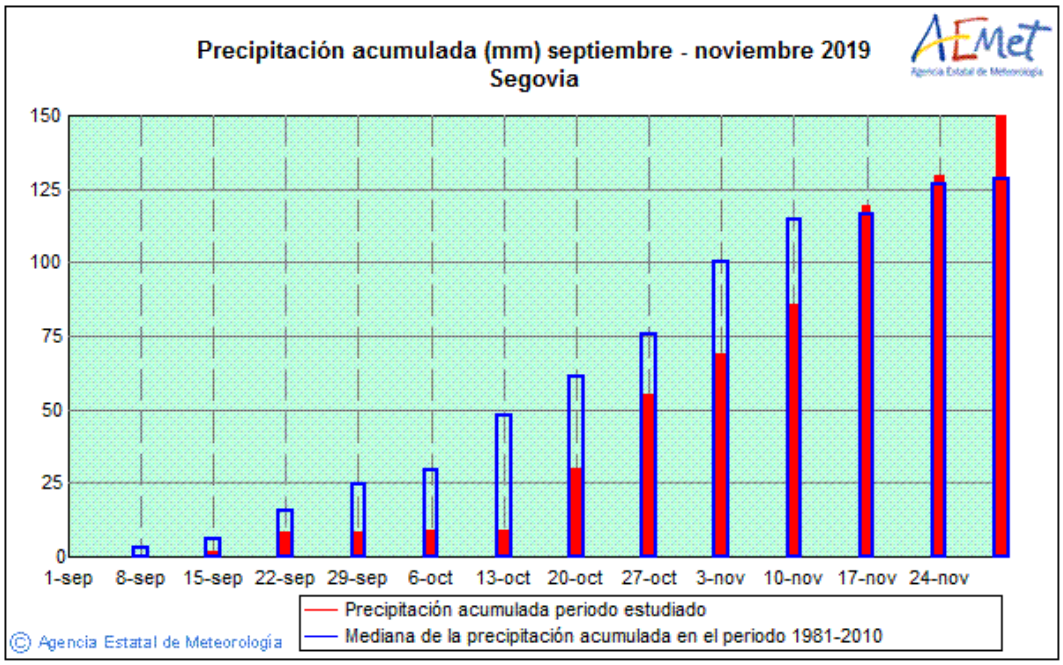


FIGURA 90: AEMET PRECIPITACIONES OTOÑO2019

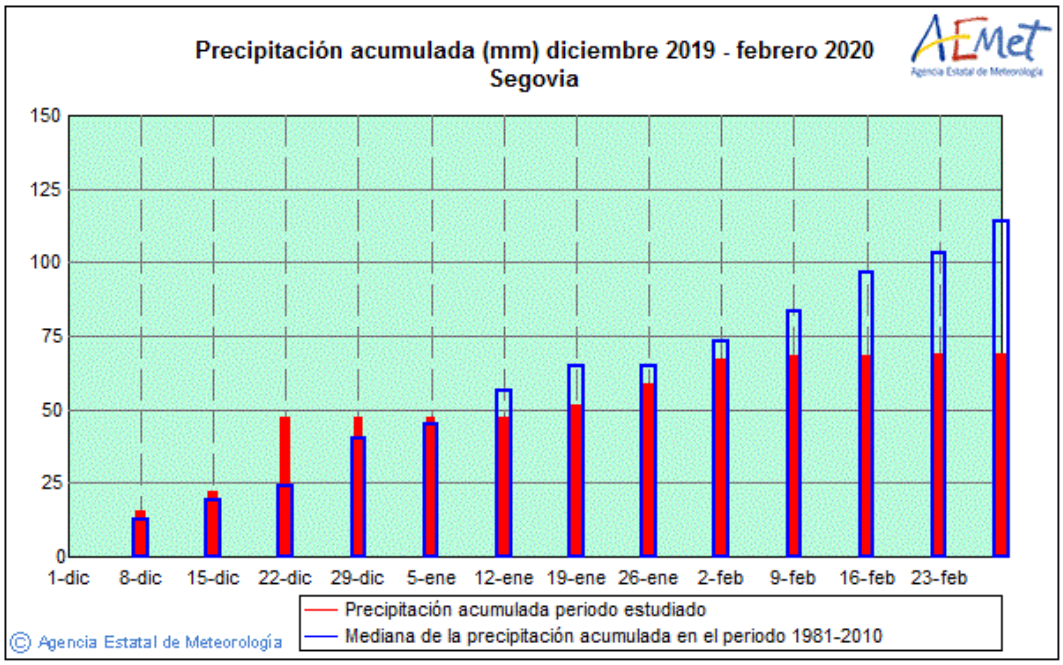


FIGURA 91: AEMET PRECIPITACIONES INVIERNO2019

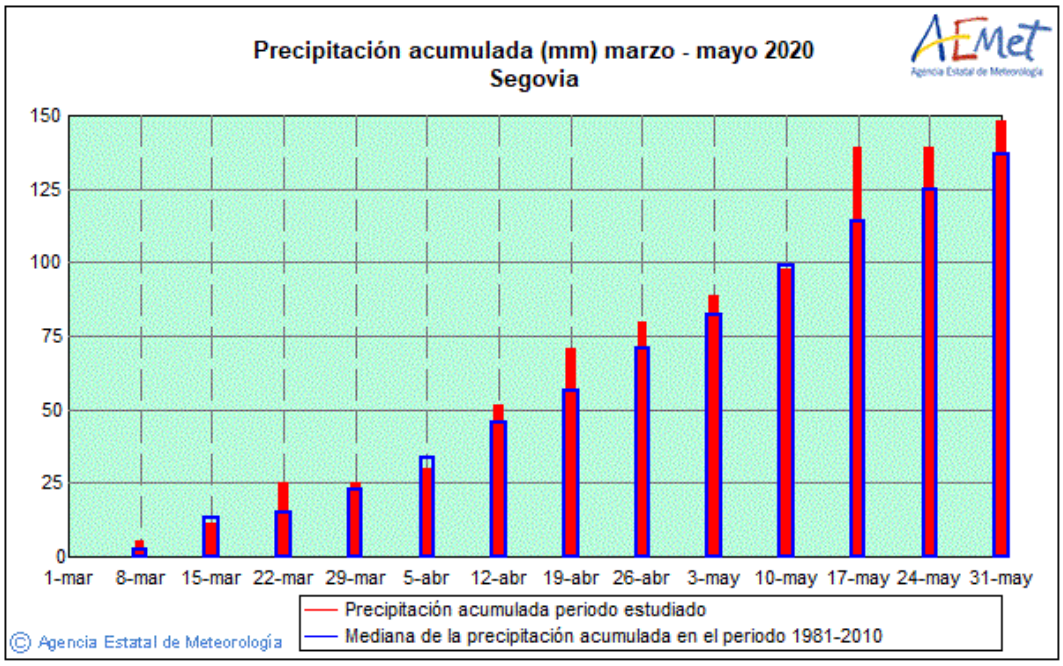


FIGURA 92: AEMET PRECIPITACIONES PRIMAVERA2020

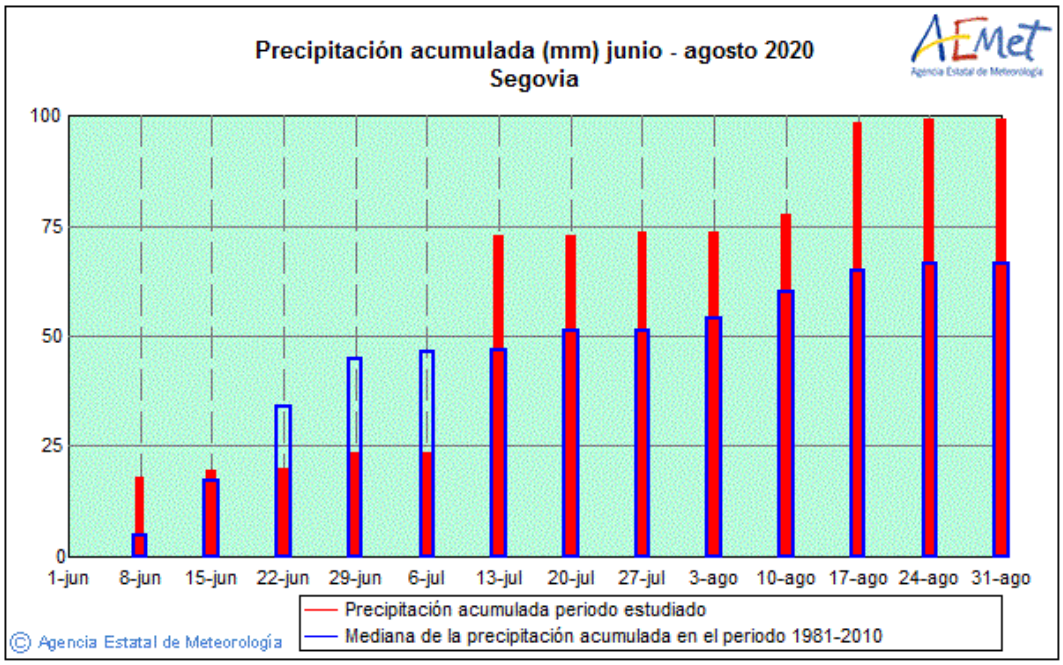


FIGURA 93: AEMET PRECIPITACIONES VERANO2020

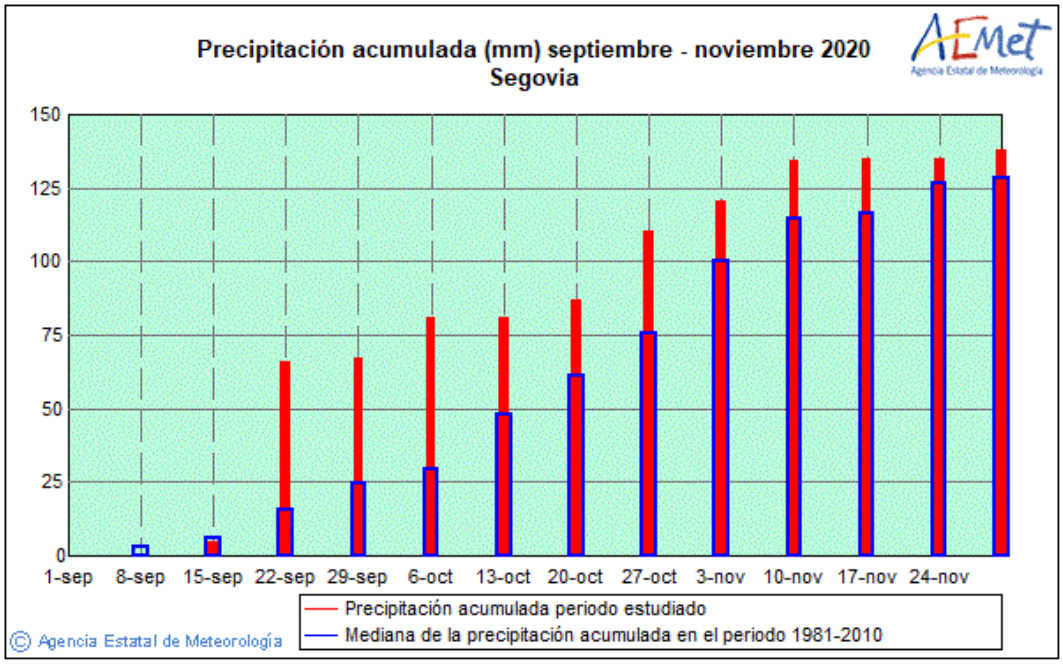


FIGURA 94: AEMET PRECIPITACIONES OTOÑO2020

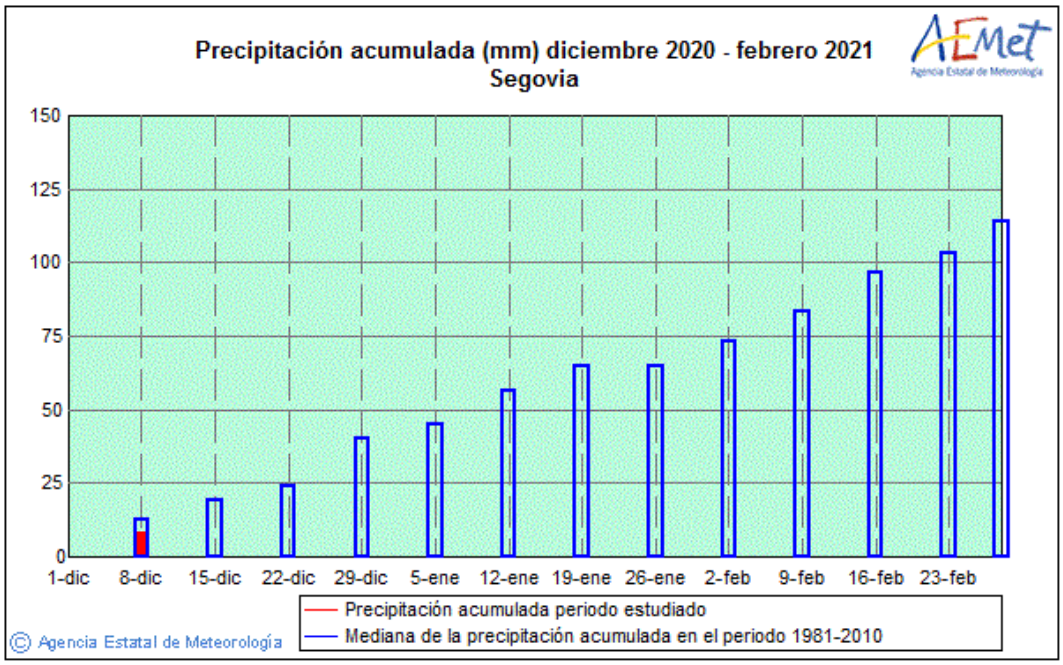


FIGURA 95: AEMET ESTIMACIÓN PRECIPITACIONES INVIERNO2020