ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Proyecto Fin de Grado

Desarrollo de un Sistema de monitorización y control de un invernadero aplicando Tecnología IoT
(Development of a monitoring and control System for a greenhouse applying IoT Technology)

Para acceder al Título de

GRADUADA EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Eva María Hernández Sanz

Enero – 2019

Agradecimientos

Dedico este Trabajo de Fin de Grado a mi padre, ya que su forma de ver la vida, ha estado presente siempre en la mía. Gracias por tu fuerza y tu temple, y por enseñarme todo lo que sé. Te quiero.

A mi madre y hermanos, por su ayuda, apoyo y cariño incondicional.

A mi pareja y amigos, por su cariño y por aguantar mis días malos.

A mi tutora Elena, por estar disponible en todo momento, y por lidiar conmigo hasta el final.

Gracias

INDICE GENERAL

DOCUMENTO Nº 1:

MEMORIA

Índice general:

Índice de figuras:

Índice de ecuaciones:

Índice de tablas:

1. INTRODUCCIÓN

1.1. MOTIVACIÓN

El presente proyecto nace de la idea de unir las tecnologías IoT (Internet of Things) con el cultivo hortícola en huertos urbanos, un nuevo mercado en auge especialmente en grandes ciudades en las que cada vez es más común encontrar huertos comunitarios o cultivos modestos en terrazas y balcones.

Las razones de esta nueva moda urbana son diversas. El creciente uso de productos químicos en los cultivos tradicionales así como el crecimiento del mercado de productos transgénicos son un motivo de reflexión que está incrementando la venta de productos ecológicos en nuestra sociedad. Cada vez más personas consideran la creación de un huerto urbano una vía de escape en la que invertir su tiempo libre obteniendo además un producto fresco y de calidad. Sin embargo, el clima y la falta de tiempo en el día a día juegan un papel crítico a la hora de mantener este tipo de cultivos, dando lugar muchas veces a la pérdida de los mismos.

El desarrollo de un sistema domótico inteligente que permita controlar los invernaderos domésticos de forma autónoma, consultando su estado y actuando sobre ellos desde cualquier lugar, puede dar solución a éste problema. Este tipo de sistema permitirá a los usuarios delegar las tareas más repetitivas como el control del riego en función de la temperatura y la humedad, enfocándose en otras más artesanales, enriquecedoras y didácticas como la poda o la recolección, que ahora podrán realizarse en cualquier época del año.

Por otro lado, el crecimiento exponencial del Internet de las Cosas o IoT (‘Internet of Things’), en el que se engloban un sinfín de tecnologías emergentes relacionadas con sensórica inalámbrica e inteligencia artificial, nos está permitiendo dar un paso más hacia la vida inteligente y conectada. Ya podemos adquirir en el mercado numerosos sistemas conectados a Internet gracias a estas tecnologías, como neveras inteligentes capaces de analizar necesidades para nuestra próxima compra, y que posteriormente envían un mensaje a nuestro teléfono móvil para confirmar el pedido de aquello que necesitamos en nuestra tienda online predefinida.

Dicho de otro modo, un sistema IoT transformará cualquier objeto físico en un producto de datos digitales. Una vez que se conecta un sensor, el objeto físico comienza a funcionar de manera muy similar a cualquier otro producto digital, aportando datos sobre su uso, ubicación y estado, pudiendo rastrear, controlar, personalizar y actuar de forma remota sobre él. [1]

Por todo ello, el diseño de un sistema domótico inteligente para el control de invernaderos domésticos nos permitirá no solo resolver un problema existente en nuestra sociedad como la pérdida de cultivos urbanos por falta de tiempo, sino hacerlo en base a una clara tendencia del mercado en la actualidad como es el IoT, asentando una base sólida para la futura comercialización del sistema diseñado.

1.2. ANTECEDENTES

1.2.1. INVERNADEROS CONVENCIONALES

Un invernadero (o invernáculo) es un lugar cerrado, estático y accesible a pie que se destina a la horticultura, dotado habitualmente de una cubierta exterior translúcida de vidrio o de plástico, que permite el control de la temperatura, la humedad y otros factores ambientales, y que se utiliza para favorecer el desarrollo de las plantas.[2]

Dentro de un invernadero es posible obtener unas condiciones artificiales de microclima, y con ello cultivar plantas en condiciones óptimas y fuera de temporada.

Un microclima es un entorno que tiene diferentes condiciones ambientales a las encontradas en la misma área. En un invernadero, el microclima está presente gracias a los rayos solares que penetran en el interior quedando retenidos, lo que produce una reacción que presenta las condiciones necesarias para la siembra de un cultivo en común, como es la temperatura, humedad del suelo y humedad relativa.[3]

Explicando el efecto, la luz solar penetra a través de las paredes del invernadero calentando el interior. El recubrimiento de estas paredes puede ser de cristal, policarbonato o film plástico, normalmente compuestos por más de una capa para aislar mejor la temperatura. Estos materiales dispersan la luz entrante de manera que evitan la formación de sombras en el interior. Las plantas, la tierra o los sustratos del interior se calientan por efecto de la radiación solar entrante. Y éstos a su vez, desprenden calor en forma de rayos infrarrojos de onda larga (invisibles), que rebotan (por refracción) en el recubrimiento y aumentan la temperatura interior. Cuanto más caliente se encuentre un objeto, más radiación infrarroja emitirá. El calor se queda acumulado en la parte más alta del invernadero, por lo que es necesario que el mismo tenga un sistema de ventilación para no saturar demasiado el ambiente de temperatura y humedad.

La idea y la necesidad del uso de invernaderos se remonta a 1850, donde la horticultura neerlandesa comenzó a utilizarlos para el cultivo de uvas. Se descubrió que el cultivo en invernaderos con calefacción y con el más alto nivel de cristal incrementaba el rendimiento. Las plantas crecían más rápidamente cuando se les daba más luz y cuando el entorno cálido era constante. Más tarde y de manera continuada se hicieron investigaciones para la mejora de estos invernaderos, generando modelos de gran calidad, como los invernaderos Casta.

En España el cultivo en invernaderos prolifera en las provincias de Alicante, Murcia, Almería y Granada a finales de los 70. [2]

Dado que el clima terrestre es caótico y complejo, debido a factores en los que el hombre no tiene influencia alguna, los diferentes tipos de cultivos se ven afectados de forma directa. En las próximas décadas, la agricultura deberá afrontar, por una parte, una demanda creciente en alimentos y materias primas básicas, y por otra la necesidad de utilizar los recursos disponibles sin causar la degradación del ambiente.

Las ventajas del cultivo en invernadero son diversas residiendo su principal utilidad en la posibilidad de evitar los cambios bruscos del clima como la variación de temperatura, la escasez de lluvia o el exceso de humedad en la tierra. Gracias a ellos también es posible producir cultivos en las épocas del año más difíciles obteniendo cosechas fuera de temporada, sustituyendo el clima de otras regiones y alargando el ciclo del cultivo. Además se obtienen productos de mejor calidad y una mayor producción en la cosecha, algo que permite incrementar la economía de escala reduciendo el coste de producción.

Este incremento del valor de los productos permite que el agricultor pueda invertir tecnológicamente en su explotación mejorando la estructura del invernadero, los sistemas de riego localizado, los sistemas de gestión del clima, etc, que se reflejan posteriormente en una mejora de los rendimientos y de la calidad del producto final.

En los últimos años son muchos los agricultores que han iniciado la instalación de sistemas que permiten la automatización de la apertura de las ventilaciones, radiómetros que indican el grado de luminosidad en el interior del invernadero, equipos de calefacción, etc.

1.2.2. INTERNET DE LAS COSAS IoT

El término “Internet de las cosas” (en inglés, *Internet of Things*, abreviado *IoT*), hace referencia a todos aquellos objetos o dispositivos cotidianos que se encuentran conectados a Internet y que cuentan con algún tipo de inteligencia.

El Internet de las cosas permite que cualquier objeto pueda comunicarse con otras entidades (ya sean estas otros sistemas o personas), obteniendo información útil para llevar a cabo una determinada tarea o función. Así pues, la idea del Internet de las cosas sugiere que, en vez de tener un pequeño número de dispositivos informáticos muy potentes (ordenador portátil,

Tablet, Smartphone, etc), podríamos tener un sinfín de pequeños dispositivos periféricos con poca potencia (como la nevera que avisa de que faltan huevos, un paraguas que avisa que ese día va a llover, una pulsera con tensiómetro que ayuda a controlar el ejercicio, etc).

Para que un objeto esté conectado a Internet debe haber algún tipo de intercambio de información, la cual será analizada y procesada para ayudar en la toma de decisiones sobre qué acciones es necesario realizar.

Estos "objetos inteligentes conectados" necesitan la suficiente autonomía (vida útil de sus baterías o uso de energías renovables) para alimentar un sistema que analice la información (procesador-microcontrolador y programación software) enviándola a otros sistemas de forma inalámbrica (conexión inalámbrica a través de WiFi, Bluetooth, ZigBee, etc). El dispositivo se conectará a un objeto físico que se encuentra en el mundo real, ya sea en casa, en el trabajo, en el coche o junto a nuestro cuerpo. El sistema recibirá continuamente estímulos de diferentes tipos, transformándolos en datos que serán procesados y enviados. Estos dispositivos pueden además generar respuestas en el mundo real por medio de actuadores, algunas de estas respuestas se podrán generar localmente a partir de los datos recogidos y otras veces serán recibidas desde Internet bien por análisis más profundos de la información histórica, bien por acciones directas de un posible "usuario conectado".

El Internet de las cosas tiene sus raíces en el trabajo que realizó Mark Weiser para Xerox PARC en la década de 1990. Weiser no estudió cómo serían las redes que conectarían las redes entre sí, sino qué ocurriría cuando la tecnología fuese tan barata que se pudiese incorporar a objetos cotidianos. Entonces acuñó el término Computación ubicua, o *ubicomp*, también conocida como "tecnología ambiental" (aunque conlleva otras connotaciones que pasan a segundo plano) o "tecnología sosegada": sistemas que no necesitan nuestra atención para funcionar y que están listos para proporcionar información útil cuando la solicitamos.[1]

Gracias a la computación ubicua podemos diseñar y mejorar los invernaderos convencionales que hemos comentado anteriormente. Manejando de forma inteligente y autónoma todos aquellos sistemas instalados en el invernadero, como el sistema de riego y el de ventilación, para mantener los niveles adecuados de temperatura, humedad relativa y calidad del aire. Con ello conseguir la mejor respuesta del cultivo y por tanto, mejoras en el rendimiento, precocidad, calidad del producto y calidad del cultivo.

El usuario introducirá las condiciones ambientales deseadas y el microcontrolador actuará sobre sus salidas para conseguirlas, corrigiendo sus decisiones gracias a la labor de los

sensores, los cuales permiten cerrar el lazo de control. De esta forma, puede crearse un microclima con unas condiciones determinadas de temperatura, humedad y calidad de aire en el invernadero.

Ilustración 1 - Esquema general de un sistema para control de invernaderos domésticos.

1.3. OBJETIVOS Y ALCANCE

El objetivo general del proyecto es diseñar e implementar un sistema de control inteligente para invernaderos domésticos.

1.3.1. OBJETIVOS ESPECÍFICOS

En este punto se desglosa el objetivo general en objetivos más específicos, sin llegar a profundizar en los mismos, para hacerse una idea general del proyecto final deseado.

- El sistema dispondrá de conectividad inalámbrica a Internet pudiendo enviar y recibir información a un servidor que almacenará los datos.
- El sistema recopilará información de diferentes sensores, clasificando las medidas y mostrándolas al usuario final local y remotamente.
- El sistema procesará la información recopilada por los sensores, tomando decisiones localmente gracias a un modo autónomo. Se podrán clasificar estas decisiones en:
    - Alertas de errores o alarmas que se enviarán al usuario final.
    - Acciones sobre actuadores que permitirán reducir el riesgo detectado.
- El sistema dispondrá de un modo manual que permitirá al usuario actuar remotamente sobre los actuadores.
- El sistema se alimentará mediante un panel solar fotovoltaico que cargará una batería dotándolo de una semana de autonomía.

1.3.2. ALCANCE

En este punto se da una visión más concreta de los objetivos específicos, delimitando el alcance que finalmente se ha abordado para el presente proyecto:

- Se diseñará el sistema como un prototipo o modelo experimental, descartando la fabricación de una placa industrial.
- El sistema medirá los siguientes parámetros:
    - Temperatura ambiente
    - Humedad relativa ambiente
    - Humedad de tierra
    - Luminosidad
    - PH agua de riego
    - Nivel depósito de suministro para riego

- El sistema dispondrá de los siguientes medios de actuación:
    - Ventilador
    - Bomba de agua
- El sistema alertará de errores o problemas en determinadas circunstancias, como por ejemplo heladas, falta de agua en la tierra o riesgo por alta temperatura.
- El sistema informará al usuario final mediante:
    - Pantalla local LCD
    - Página web
    - Aplicación móvil
    - Emails automáticos
- Se realizará el dimensionamiento de una placa solar y batería para la autonomía del sistema, pero no se realizará su implementación en el prototipo.

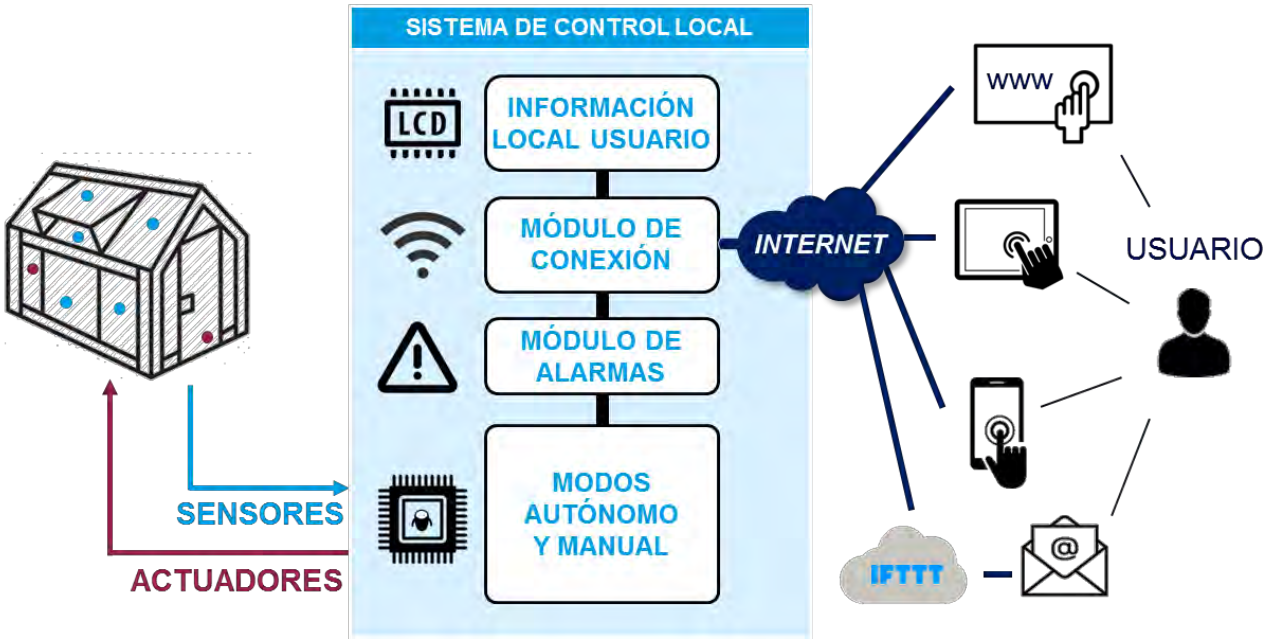


Ilustración 2 - Esquema ampliado del sistema

2. ESTUDIO PREVIO

El presente proyecto tiene sus inicios en Arduino, y por motivos de avance en el proyecto, se han ido haciendo las modificaciones necesarias hasta llegar al actual. Sin embargo, hay una serie de lecciones aprendidas por el camino, de las cuales la más destacable sea, probablemente, el diseño de una placa industrial completa basada en un Arduino Mega 2560.

La idea original suponía emplear un Arduino y un módulo ESP8266 para las comunicaciones, por ello, la PCB diseñada incluye la placa base de Arduino Mega 2560, conectores para los sensores exteriores del invernadero, buzzer para alarma, leds de indicación de estado de los sensores y un zócalo para colocar el módulo wifi basado en ESP8266 (NodeMCU).

Como no es objeto de este punto, hablar de cómo se realizó dicha PCB, se presenta el documento Anexos [1] que incluye el proceso de diseño de la PCB. Así como los esquemáticos del diseño de la placa en el documento Anexos [2], [3] y [4], y los ficheros Gerber generados para su futura fabricación, en el documento Anexos [5].

Además de la PCB, se implementaron los primeros pasos en Arduino, sin estudio ni análisis de cultivo previos. Se empleó en lugar de una pantalla LCD para el aviso de estados en el invernadero, unos leds que proporcionaban información según estuviesen encendidos unos u otros. Además, esta información de los sensores se transmitía al ESP8266 por un puerto serial, como una cadena de caracteres generada por el mismo programador. Esto era bastante problemático, ya que no es una comunicación rápida y se necesitan además dos programas: uno para cargar en la placa Arduino Mega y otro para cargar en el módulo ESP8266.

Es por eso que, más tarde, se decidió pasar toda la carga a un solo programa y por tanto, surgió la necesidad de buscar un dispositivo compatible con el ide de Arduino, con capacidad y pines suficientes para todos los sensores y actuadores empleados, así como disponer de comunicación WiFi.

Además, destacar que se comenzó publicando datos al servidor *CloudMQTT,* pero el websocket no tenía una interfaz gráfica muy llamativa y, además, las limitaciones eran mayores, pues la versión gratuita limitaba hasta 5 conexiones (5 topics o temas de publicación/subscripción).

A continuación se muestran unas imágenes del primer control sobre invernadero diseñado con Arduino:

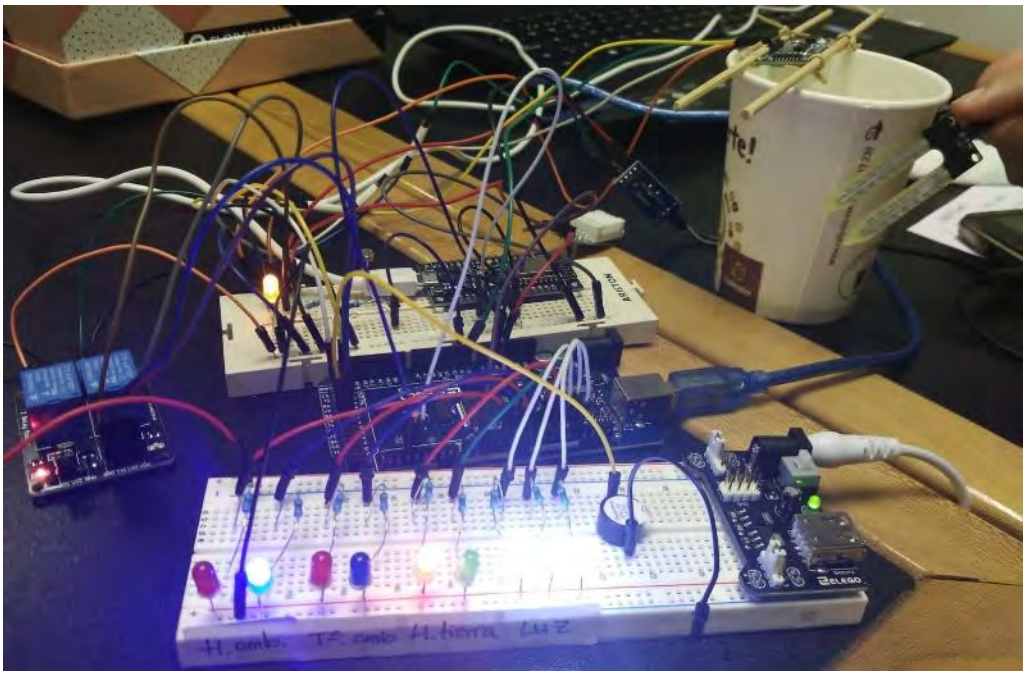


Ilustración 3 - Proyecto inicial con Arduino y ESP8266.

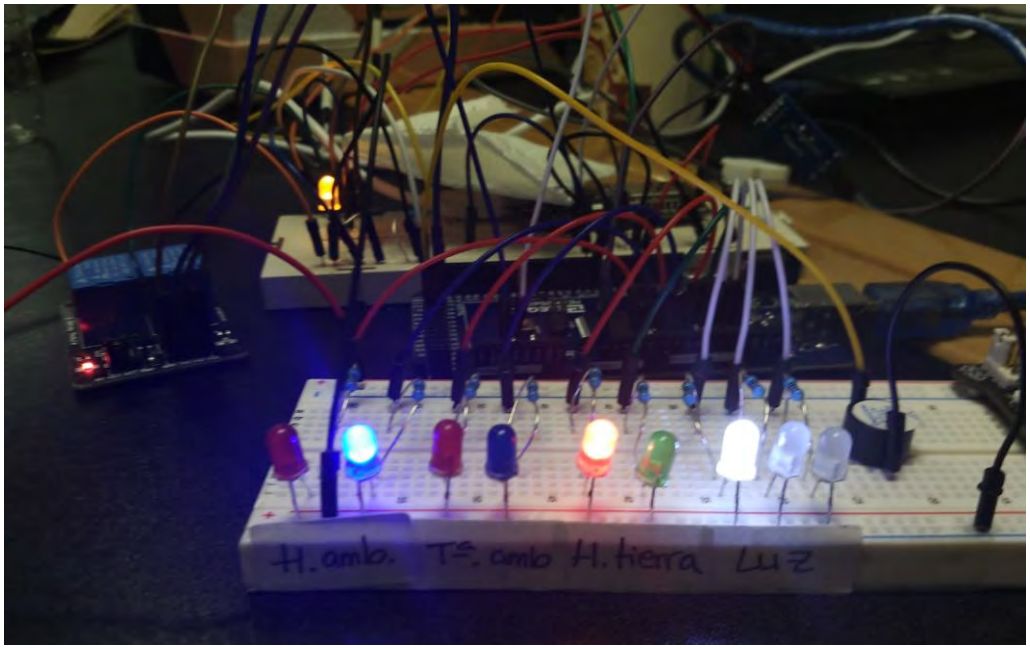


Ilustración 4 - Proyecto inicial con leds informativos de estado del invernadero.

Un ejemplo de los datos publicados en el Websocket del servidor CloudMQTT:

/NodeMCU/humedadAmbiente	65.90
/NodeMCU/temperatura	24.00
/NodeMCU/luminosidad	alta
/NodeMCU/humedadTierra	humedo
/NodeMCU/NivelDepositoAgua	92.74

Ilustración 5 - Publicación de datos en CloudMQTT.

3. PROCESO DE CULTIVO EN INVERNADERO

3.1. ANÁLISIS INICIAL

Dentro de un invernadero podemos controlar las necesidades de nuestros cultivos respecto a las condiciones climáticas del mismo. Cabe destacar, que cada tipo de cultivo tiene unas necesidades específicas, y por tanto unos límites admisibles de climatización. Además, se debe tener constancia de la existencia de multitud de parámetros a medir y controlar en un cultivo.

Por esta razón, es necesario limitar el sistema, de tal forma que se establezcan:

- **Los parámetros a medir:** temperatura, humedad relativa, humedad de suelo, luminosidad, ph de agua de riego y nivel de agua de depósito.
- **Límites admisibles para un tipo de cultivo.** En el ejemplo del presente trabajo, se propone el cultivo de tomate.
- **Relaciones entre los valores medidos.** Este punto es importante para el procesado y control sobre los actuadores del invernadero y la gestión de alertas. En el proyecto desarrollado, se ha elegido actuar sobre riego y ventilación.

En los siguientes apartados, se expondrá de manera general los parámetros a medir en el invernadero del proyecto desarrollado. Así como de manera más específica, se ha elegido un cultivo concreto (el tomate) para establecer los límites de actuación. Cabe señalar que por falta de tiempo, no se ha podido diseñar un método de selección de cultivo, pero se pretende implementar en próximos pasos.

3.2. PARÁMETROS DE CULTIVO

3.2.1. TEMPERATURA Y HUMEDAD AMBIENTE

Estos parámetros son, junto con la humedad de tierra, los más importantes a tener en cuenta dentro de un invernadero. Sin embargo, para el manejo de los mismos, es importante conocer las necesidades y limitaciones de la especie cultivada, tal y como se expone en [4]:

- Temperatura mínima letal. Aquella por debajo de la cual se producen daños en la planta.
- Temperaturas máximas y mínimas biológicas. Indican valores, por encima o por debajo respectivamente del cual, no es posible que la planta alcance una determinada fase vegetativa, como floración, fructificación, etc.
- Temperaturas nocturnas y diurnas. Indican los valores aconsejados para un correcto desarrollo de la planta.

A continuación se muestra una tabla, obtenida de la referencia [4], que recoge las exigencias de temperatura para distintas especies de cultivo:

	TOMATE	PIMIENTO	BERENJENA	PEPINO	MELÓN	SANDÍA
Tª mínima letal	(-2)	(-1)	0	(-1)	0-1	0
Tª mínima biológica	10-12	10-12	10-12	10-12	13-15	11-13
Tª óptima	16-26	16-18	17-22	18-18	18-21	17-20
Tª máxima biológica	25-27	23-27	22-27	20-25	25-30	23-28
Tª máxima letal	35-38	33-35	43-53	31-35	33-37	33-37

Tabla 1 - Exigencias de temperatura para distintas especies de cultivos en Invernadero

Para el cultivo de tomate se ha hecho un estudio un poco más exhaustivo, en el que se obtienen diferentes rangos de temperatura dependiendo de la especie de tomate cultivado. En este proyecto se establecen límites para el cultivo de cualquier especie de tomate, gracias a la rusticidad asociada la planta que permite su cultivo en condiciones adversas.

El tomate prospera mejor en climas secos con temperaturas moderadas. En general, las plantas de tomate crecen en un rango de temperatura de 10 a 35ºC y sobreviven a temperaturas de hasta 38ºC y de 0ºC. No obstante, el tomate es una especie de estación

cálida, su temperatura óptima de desarrollo varía entre 16 y 26°C, por ello, el cultivo al aire libre se realiza en climas templados. Temperaturas extremas pueden ocasionar diversos trastornos, ya sea en la maduración, precocidad o color. Temperaturas bajo 10°C afectan en la formación de flores y temperaturas mayores a 35°C pueden afectar en la fructificación por mal desarrollo de óvulos y al desarrollo de la planta en general y del sistema radicular en particular. Asimismo, la temperatura nocturna puede ser determinante en la producción, ya que, cuando es inferior a 10°C origina problemas en el desarrollo de la planta y frutos, provocando deformidades.

Una planta de tomate indeterminada desarrolla tres hojas por semana a una temperatura promedio diaria de 20°C en el tallo principal y, dependiendo del cultivo, entre 6 y 8 semanas después de que los frutos alcancen la etapa madura (roja). La maduración del fruto está muy influenciada por la temperatura en lo referente tanto a la precocidad como a la coloración, de forma que valores cercanos a los 10ºC así como superiores a los 30ºC originan tonalidades amarillentas.

En el cultivo de invernadero la gran parte de la radiación solar es absorbida y reflejada por la cubierta. Sin embargo, habrá que tener en cuenta que en los invernaderos comerciales, el 70–80% de la radiación solar exterior llega a las plantas, mientras que en invernaderos experimentales esta fracción es mucho más pequeña, especialmente durante el invierno.

Las temperaturas más cálidas suelen conllevar un tiempo de germinación más corto, pero una temperatura excesiva tampoco es favorable. La temperatura óptima de germinación se sitúa normalmente entre los 16 y 26°C. Por debajo de 16°C, la germinación se retrasará, y lo hará conforme baja la temperatura. En cambio, a temperaturas cercanas a los 26°C, la germinación se demora solamente unos 5 o 6 días. En general, el tiempo de germinación estará comprendido entre 5 y 8 días.

Temperatura	Estado de la planta	
-2ºC	Se hiela la planta	
10-12ºC	Se detiene su desarrollo	
18-25ºC	Desarrollo normal	
21-24ºC	Mayor desarrollo	
16-26ºC	Germinación óptima	
Temperaturas óptimas		
Desarrollo	Diurno	23-26ºC
	Nocturno	13-16ºC
Floración	Diurna	23-26ºC
	Nocturna	15-18ºC
Maduración		15-22ºC

Tabla 2 - Temperaturas críticas para el cultivo de tomate.

No obstante, se debe considerar que los valores de temperaturas por sí solos son referenciales, puesto que su interacción con otros factores repercute mayormente. Por ejemplo, la combinación de altas temperaturas con humedad baja, puede generar aborto floral y baja viabilidad del polen.

La humedad relativa es la cantidad de agua contenida en el aire, en relación con la máxima que sería capaz de contener a la misma temperatura. Existe una relación inversa de la temperatura con la humedad por lo que a elevadas temperaturas, aumenta la capacidad de contener vapor de agua y por tanto disminuye la HR. Con temperaturas bajas, el contenido en HR aumenta.[3]

Cada especie tiene una humedad ambiental idónea para vegetar en perfectas condiciones: al pimiento y berenjena les gusta una HR sobre el 50-60%; al melón, entre el 60-70%; al calabacín, entre el 65-80% y al pepino entre el 70-90%.

La HR del aire es un factor climático que puede modificar el rendimiento final de los cultivos. Cuando la HR es excesiva las plantas reducen la transpiración y disminuyen su crecimiento, se producen abortos florales por apelmazamiento del polen y un mayor desarrollo de enfermedades criptogámicas (hongos). Por el contrario, si es muy baja, las plantas transpiran en exceso, pudiendo deshidratarse, además de los comunes problemas de mal cuaje.

Respecto al desarrollo del tomate, la humedad relativa puede oscilar entre 60% y 80%, siendo entre el 65 y el 75% el óptimo para el crecimiento, floración, fructificación y crecimiento de la fruta en la planta de tomate. Considerando que humedades relativas muy elevadas (superiores al 85%) y de larga duración, favorecen el desarrollo de enfermedades fungosas y bacterianas (hongos, como el mildiú polvoriento), además en combinación con una temperatura superior a 30°C es crítica para la polinización, debido a que el polen se compacta abortando parte de las flores. También está vinculado al agrietamiento de fruto, cuando se presenta un período de estrés hídrico y luego se produce un exceso de humedad en el suelo por riego abundante.

Por debajo del 30% de humedad relativa, las plantas de tomate todavía crecen pero no de manera óptima. Particularmente en combinación con la temperatura alta, la morfología y la fisiología cambian: el contenido de materia seca vegetal aumenta y el área foliar específica disminuye. La fotosíntesis puede disminuir debido al cierre de los estomas a temperaturas muy altas y baja humedad relativa.

En la mayoría de las situaciones, la humedad relativa del aire se puede evitar con un control de clima adecuado, es decir, ventilando el invernadero y, si es necesario, calentando al mismo tiempo. Todos los datos de estudio de temperatura y humedad se han contrastado en diversos artículos. [5], [6], [7], [8] y [9].

3.2.2. HUMEDAD DEL TERRENO

Se denomina humedad del suelo a la cantidad de agua por volumen de tierra que hay en un terreno. La aplicación de riego en el momento y cantidad apropiada es fundamental para obtener un buen rendimiento en el cultivo. El exceso de agua reduce el crecimiento al arrastrar los nitratos a una profundidad tal que las raíces de los cultivos no puedan alcanzar. Además el exceso de agua puede desplazar el aire contenido en el interior del suelo, lo que provoca la escasez de oxígeno en las raíces de dichos cultivos.

No solo el exceso de agua es perjudicial, la escasez de ésta también lo es. La falta de humedad provoca las caídas de las hojas y la aparición de plagas. El exceso de humedad por otro lado, puede provocar podredumbre, puntas marrones o manchas en las hojas, así como la aparición de enfermedades de carácter fúngico.

La rusticidad de la planta de tomate permite que sea poco exigente a las condiciones de suelo. Sin embargo, debe tener un buen drenaje; de aquí la importancia de un suelo con alto contenido de materia orgánica. En suelos arcillosos y arenosos, se desarrolla con un mínimo de 40 cm de profundidad.

En cuanto al pH de suelo, el óptimo debe oscilar entre 6 y 6,5 para que la planta se desarrolle y disponga de nutrientes adecuadamente. Los suelos pueden ser desde ligeramente ácidos hasta ligera a medianamente alcalinos. Al respecto, es posible encontrar cultivos de tomate establecidos en suelos que presentan pH 8, siendo un factor posible de manejar, ya que el tomate es la especie cultivada en invernadero que mejor tolera las condiciones de pH.

Todos los datos de estudio de humedad de tierra y pH se han contrastado en diversos artículos: [4], [5], [6], [7], [8] y [9].

3.2.3. LUMINOSIDAD

De manera general, a mayor luminosidad en el interior del invernadero se debe aumentar la temperatura y la humedad, para que la fotosíntesis sea máxima; por el contrario, si hay poca luz pueden descender las necesidades de otros factores. Para mejorar la luminosidad natural se usan los siguientes medios:

- Materiales de cubierta con buena transparencia.
- Orientación adecuada del invernadero.
- Materiales que reduzcan el mínimo las sombras interiores.
- Aumento del ángulo de incidencia de las radiaciones sobre las cubiertas.
- Acolchados del suelo con plástico blanco.

En verano para reducir la luminosidad se emplean:

- Blanqueo de cubiertas.
- Mallas de sombreo.
- Acolchados de plástico negro.

Es interesante destacar el uso del blanqueo ya que esta labor está en función del desarrollo del cultivo y de las temperaturas, y tiene efectos contradictorios que hay que conocer para hacer un correcto uso. Hay que saber que la planta sombreada se ahila y se producen abortos de flores en determinadas especies sensibles a la luz (especialmente tomate, pimiento y berenjena), por lo que el manejo del riego y de la solución nutritiva tiene que ir unida al efecto

que produce el blanqueo. Los plásticos sucios o envejecidos provocan el mismo efecto que el blanqueo.

La luminosidad en el cultivo de tomate cumple un rol importante, más allá del crecimiento vegetativo de la planta, ya que el tomate requiere al menos 6 horas diarias de luz directa para florecer. Estos valores reducidos pueden incidir de forma negativa sobre este proceso y la fecundación. En zonas de alto polvo en suspensión, durante períodos de recambio de cultivo, se realizan frecuentes lavados de la cubierta de los invernaderos con el objetivo de mejorar la producción y evitar un posterior exceso de crecimiento vegetativo.

Sin embargo, estudios indican que el fotoperiodo no sería un factor crítico a diferencia de la intensidad de radiación, que si es muy alta se pueden producir golpes de sol, partiduras, coloración irregular, entre otros.

El estudio de luminosidad se ha contrastado en diversos artículos y una tesis:[3], [4], [5], [6], [7], [8] y [9].

3.2.4. PH AGUA DE RIEGO

Antes de conocer los niveles de pH entre los que debe encontrarse nuestro sistema, es imprescindible conocer qué tipo de parámetro vamos a medir y porqué.

El pH es una medida de la acidez o alcalinidad de una solución, y cuya escala varía de 0 a 14. El pH indica la concentración de iones de hidrógeno $[H]^+$ presente en determinadas sustancias. Se puede cuantificar con precisión usando un sensor que mide la diferencia de potencial entre dos electrodos: un electrodo de referencia (plata / cloruro de plata) y un electrodo de vidrio que es sensible al ión de hidrógeno. Esto es lo que formará la sonda. Además, hay que usar un circuito electrónico para condicionar la señal de manera apropiada. Hablaremos de éste sensor en detalle más adelante. Por ahora, nos interesa saber que las necesidades de pH de una planta varían si se cultivan en el suelo o en cultivos hidropónicos, por lo que es importante adaptar su nivel de pH a su método de crecimiento. Establecer un rango de valores de pH toma importancia especialmente para la capacidad de absorción de nutrientes del cultivo en cuestión.

En lo referente al agua de riego, que sea de alta calidad es un requisito previo para el riego en experimentos. En general, hay varias fuentes de agua disponibles, pero los recursos

hídricos pueden ser limitados: agua desionizada (opción más segura), agua de lluvia, agua del grifo o de pozo, y agua superficial de lagos u otras fuentes (la opción peor). La calidad del agua de las diferentes fuentes varía enormemente debido a sus concentraciones de iones y contaminaciones con agentes biológicos.

La contaminación por patógenos se puede encontrar en el agua de lluvia y en las fuentes de agua superficial y por lo que pueden necesitar desinfección. Si la instalación está equipada con un sistema de desinfección, el agua de lluvia será una buena alternativa al agua desionizada.

El agua del grifo es la fuente de agua más aplicada. Sin embargo, los usuarios a veces no consideran que el agua del grifo puede afectar al crecimiento de las plantas e incluso en la toxificación del tomate. Cuando se usa, en todos los casos, la composición iónica del agua del grifo y el agua de fuentes superficiales debe probarse y conocerse.

En el presente proyecto nos centramos en controlar únicamente el pH, recogiendo datos del mismo para esas diferentes fuentes de agua, podemos destacar que el pH del agua de lluvia oscila entre 4 y 6, y el agua de pozo suele tener un pH de 7 a 8.8.

Considerando importante fijar unos valores de pH límites admisibles para el cultivo del tomate, encontramos que el pH de riego óptimo acepta valores de 5 a 6 y el pH de la disolución del sustrato es de 5,5 a 6,8 que es el óptimo para la absorción de nutrientes.

El estudio de nivel de pH se ha contrastado en diversos artículos y una tesis: [3], [4], [5], [6], [7], [8] y [9].

3.2.5. NIVEL DE AGUA EN DEPÓSITO DE SUMINISTRO

Es habitual si queremos montar un huerto urbano o un invernadero casero, pensar en un sistema de riego adecuado y con la suficiente capacidad para suministrar agua al menos durante una semana si no es más, principalmente para no preocuparnos de forma muy continuada, ya sea por trabajo o porque nos vamos de vacaciones. Por ello resulta útil emplear un sensor de nivel que avise de cuánta agua queda en el depósito.

Cabe añadir que, se empleará una bomba de agua sumergible con capacidad para bombear agua a todos los cultivos. La bomba no puede trabajar en vacío, por lo que resulta útil saber el nivel de agua que queda en el depósito, y así evitar que la bomba se rompa por cavitación.

Un Sensor de Nivel es un dispositivo electrónico que mide la altura del material, generalmente líquido, dentro de un tanque u otro recipiente. Les hay de nivel de punto o de medición continua, así como de contacto o sin contacto, les hay ultrasonidos, de capacitancia y con radar. En este trabajo se emplea un sensor de ultrasonidos compatible con Arduino.

3.2.6. SISTEMA DE RIEGO

Existen diversos procedimientos para la distribución eficiente de agua sobre la superficie del suelo. En la actualidad, hay distintos tipos de riego que facilitan este punto, aportando el suministro necesario para el buen crecimiento de las plantas. Acotando estos procedimientos, nos centramos en los más empleados en el cultivo dentro de invernaderos [10] y [11].

Sistema de riego por goteo:

Los sistemas de riego por goteo son muy ventajosos y de los más empleados actualmente. Son ideales para conservar el agua, ya que el caudal se puede medir y regular, con el añadido de que llevan directamente el agua a la tierra y a las raíces de las plantas.

El sistema de riego por goteo se basa en el transporte de agua por tubos de PVC que se ramifican para llegar a todos los puntos necesarios del invernadero. El ensamblaje es sencillo, cada pequeño tubo va con boquillas individuales para el suministro de agua a cada planta. Además, se pueden conectar temporizadores, electroválvulas y sensores como caudalímetros para su automatización.

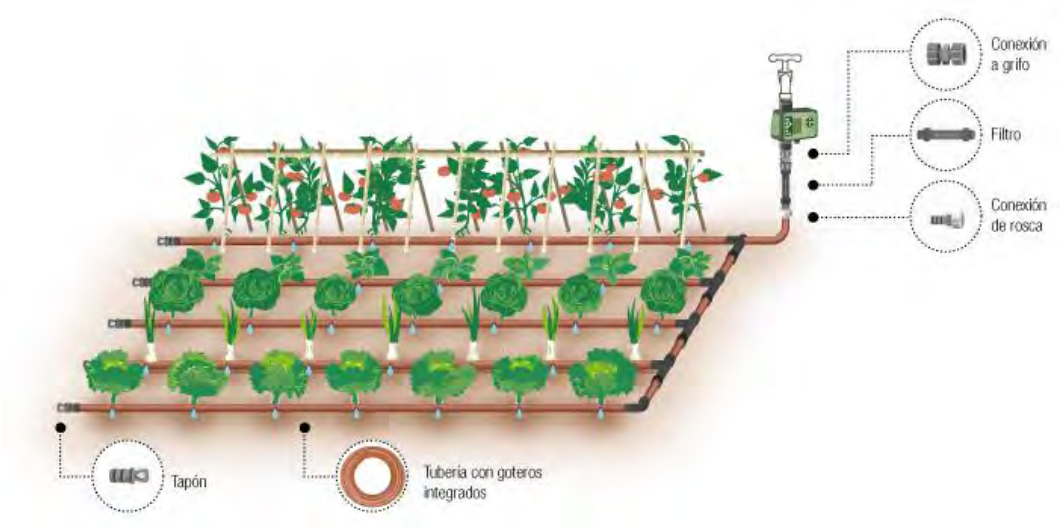


Ilustración 6 - Sistema de riego por goteo con caudalímetro.

Gracias al riego por goteo, se impide el crecimiento de malezas y moho y la pérdida de agua. Sin embargo, la desventaja radica en la posibilidad de que los goteros y las boquillas se obstruyan.

Ilustración 7 - Riego por goteo.

Riego por tapete:

El riego de tapete se emplea principalmente cuando las plantas del invernadero se encuentran en macetas o bandejas, como para semilleros o pequeñas plántulas. El sistema funciona por acción capilar manteniendo mojado un tapete espeso, el cual está instalado sobre un banco con un extremo fijado en un canal lleno de agua. Las plantas absorben el agua del tapete por las perforaciones de las macetas que las contienen. Las principales desventajas de éste sistema son, por un lado que los tapetes a medida que pasa el tiempo se obstruyen con algas, y por otro lado la limitación de su uso a plantas de pequeño tamaño.

Ilustración 8 - Riego por tapete.

Rocío:

Este sistema de riego se instala por encima de las plantas, de tal forma que el agua llega a las mismas por unas finas boquillas de los tubos de instalación. El rocío mantiene los esquejes de las plantas húmedos y ayuda a controlar la humedad, ya que la fina niebla tiene a evaporarse. Además, ayuda a enfriar la temperatura interior del invernadero. Este sistema es ideal para plantas epífitas y tropicales.

Ilustración 9 - Riego por rocío.

Sistema de agua de lluvia:

Un sistema de agua de lluvia es tal vez el sistema de riego más personalizable que existe. Una técnica común es utilizar tuberías de PVC y barriles para recoger agua de lluvia. Este es un sistema muy ecológico y que reduce los gastos también.

Ilustración 10 - Depósito de almacenamiento de agua de lluvia.

El sistema de riego implementado en el presente trabajo es el riego por goteo, ya que podemos controlar el flujo de agua necesaria independientemente del cultivo que tengamos, además de la facilidad de instalación y economía por ahorro de agua.

Cabe destacar, que el sistema de agua de lluvia es admisible si queremos economizar aún más el sistema, sin embargo, habrá que tener en cuenta que el agua de lluvia puede no ser la más adecuada para los cultivos, por lo que tendremos que medir y controlar los parámetros nutritivos del agua almacenada, como el pH.

Finalmente, comentar que la parte interesante de este punto no es el tipo de sistema de riego empleado, sino la activación del mismo. Por lo que el usuario final podrá decantarse por el sistema más adecuado para su cultivo, siempre y cuando debe permitir actuar sobre la activación y parada del sistema de riego en base a unos parámetros que automatizarán el proceso.

Así pues, la activación del riego podrá hacerse manualmente y por descontado, de forma automática. Esta selección de modo, podrá hacerse remotamente gracias al servidor web del que hablaremos más adelante en detalle. El objetivo, se trata ni más ni menos, de automatizar y facilitar el sistema de riego, incluso, podría servir para tareas de abonado y tratamientos fitosanitarios, incluyendo en la disolución del agua los mismos.

3.2.7. SISTEMA DE VENTILACIÓN

Como se ha comentado anteriormente, en un invernadero puede producirse un sobrecalentamiento debido a que la radiación infrarroja que emiten los cultivos queda atrapada. Por esta razón, se plantea un sistema de ventilación que sustituirá el aire más caliente que se encuentra en interior (en la parte más superior) por otra masa de aire más frío que procede del exterior. De esta manera gran parte de la sobrecarga de calor puede evacuarse, disminuyendo la temperatura y, a su vez, modificando la concentración de gases y la humedad.

Se puede optar por la ventilación natural o la mecánica activa. La diferencia radica en que en la natural no tenemos casi ninguna capacidad de controlar las condiciones climáticas dentro del invernadero y en la activa sí. A continuación se muestra un cuadro resumen de estas dos opciones de ventilación, mostrando sus ventajas y desventajas.

TIPO DE VENTILACIÓN	DESCRIPCIÓN	VENTAJAS	DESVENTAJAS
Natural	Extracción y admisión de aire mediante aperturas con rejilla	• Coste bajo de instalación y mantenimiento • No necesita electricidad	• Baja capacidad de controlar y determinar las condiciones climáticas internas • Alta dependencia de las condiciones climáticas externas
Activa (Mecánica simple + mecánica húmeda)	Extracción de aire mediante ventiladores electromecánicos Admisión de aire mediante ventilador y paneles húmedos	• Control de las condiciones climáticas internas • Mejores resultados anuales sin dependencia de condiciones externas	• Mayores costos en comparación con la ventilación natural • Dependencia del suministro de electricidad

Tabla 3 - Resumen tipos de Ventilación para Invernaderos.

A continuación se detallan estos tipos de ventilación. La información al respecto se ha obtenido de las siguientes referencias: [12] y [13].

Ventilación natural

En la ventilación natural, el aire caliente que se encuentra en el interior del invernadero asciende y sale al exterior por dos aperturas situada en la cubierta, mientras que la admisión se realiza desde dos aperturas en la parte baja de las fachadas laterales. De esta forma se crea un flujo de aire que abarca todo el recinto interior. Para este tipo de ventilación son necesarias grandes aberturas, entre un 15 % y un 25 % de la superficie de la cubierta, y no permite controlar la incidencia de la velocidad del aire sobre las plantas.

Cabe destacar que existen medios tecnológicos, como circuladores de aire, para aumentar el flujo de aire interno y obtener mejores resultados, aunque éstos estén influenciados por las condiciones climáticas externas y la ubicación geográfica de los cultivos.

Ventilación mecánica simple

La ventilación mecánica se basa en la renovación del aire instalando ventiladores electromecánicos en la cubierta o en la parte alta de un lateral del invernadero, mientras que las entradas de aire que proviene del exterior se localizan en la parte baja de la pared opuesta. Con este sistema la temperatura mínima interior no suele exceder de la del aire exterior. Los ventiladores deben distribuirse a lo largo del invernadero, en la cubierta o en los laterales y separados entre 7 y 10 metros. En cuanto a los ventiladores laterales, dispondrán de persianas por gravedad para evitar corrientes contrarias cuando los aparatos no están funcionando. Se utilizarán rejas anti-pájaros o roedores para proteger las entradas de aire hacia el exterior, mientras se dispondrán de deflectores hacia el interior si el aire exterior que entra incidiese directamente sobre los cultivos más próximos.

Ventilación mecánica húmeda

El sistema de ventilación mecánica húmeda se basa en saturar de humedad el aire de entrada al atravesar unos paneles de grandes dimensiones construidos con un material fibroso empapado en agua. Si el invernadero es muy ancho debe adoptarse la disposición de ventiladores de techo y entradas de aire con paneles húmedos en ambos laterales de la nave. Mediante este sistema el aire se renueva, se altera el nivel de humedad del interior y se enfría. Cuanto más seco sea el aire exterior introducido, mayor será el grado de enfriamiento.

Ventilación activa

Se puede considerar como un sistema que unifica la ventilación mecánica simple con la húmeda. La ventilación en los invernaderos es esencial y tiene efectos decisivos en los resultados de los cultivos. Los aumentos en los niveles de temperatura y humedad dañan el rendimiento y afectan la calidad de los cultivos. El sistema de ventilación activa permite lograr condiciones climáticas internas ideales produciendo mayor rendimiento y calidad en el cultivo.

En el presente proyecto se ha optado por implementar un sistema de ventilación mecánica simple. Con un único ventilador colocado en la parte superior del invernadero para la extracción de aire caliente acumulado llegado el caso en que la temperatura y humedad ambiente aumenten considerablemente.

Cabe destacar, al igual que en la selección del sistema de riego, se deja abierto a posibles modificaciones de ventilación, pudiendo incluir un sistema de ventilación activa y con ello haciendo más completo el control sobre el ambiente dentro del invernadero.

3.3. RESUMEN

3.3.1. LÍMITES

A modo resumen, se recogen en una tabla los límites establecidos de los parámetros a controlar. En la última columna se detallan las alertas que se mostrarán al usuario final.

PARÁMETROS	ÓPTIMOS	MÍNIMO ADMISIBLE	MÁXIMO ADMISIBLE	ALERTAS
Temperatura (Tª en ºC)	**16 – 26 ºC**	**10 ºC**	**35 ºC**	⚠ **Helada (0ºC)** ⚠ **Tª Baja (10ºC)** ⚠ **Tª Alta (35ºC)**
Humedad Relativa (%HR)	**65 – 75 %HR**	**55 %HR**	**85 %HR**	⚠ **HR Baja (55%HR)** ⚠ **HR Alta (85%HR)**
Humedad de tierra (en %)	**26 – 45 %**	**25 %**	**46%**	⚠ **Suelo seco (25%)** ⚠ **Suelo humedo (46%)**
Luminosidad (en %)	**30 – 60 % (Media)**	**60 % (Baja)**	**30 % (Alta)**	⚠ **No se alertará**
pH de agua de riego	**5 – 6 pH**	**5 pH**	**6,5 pH**	⚠ **Agua ácida (pH 5)** ⚠ **Agua básica (pH 6,5)**
Nivel de agua en depósito de suministro (%)	**20 – 95 %**	**15 %**	**95 %**	⚠ **Nivel de agua crítico. Peligro bomba (15%)** ⚠ **Peligro desborde en depósito (95%)**

Tabla 4 - Resumen límites de cultivo y alertas.

Los parámetros de luminosidad y humedad de tierra son convertidos de una escala de 0 a 4095 a una escala en tanto por ciento (de 0 a 100). Se establece un porcentaje para los niveles analíticos de proceso, luminosidad (alta, media o baja), y humedad de suelo (ideal, seco o húmedo). Cabe señalar, que la escala de luminosidad es inversa, por lo que un nivel alto se traduce en una luminosidad baja.

3.3.2. PROCESO

Una vez definidos los límites de control, se plantea una solución de actuación para regular los parámetros del cultivo. Para ello se toman medidas sobre la actuación del riego y la ventilación del invernadero.

Para el control de la humedad del suelo, actuaremos activando el riego siempre que no tengamos una humedad suficientemente alta. En la práctica, este sistema puede provocar que el riego se active en momentos no óptimos como por ejemplo, mediodía, momento en el que el sol radia con más intensidad generando más temperatura en el invernadero. Para regar el cultivo con eficacia, el mejor momento del día es al atardecer, ya que la temperatura va a menos y por lo tanto la evaporación del agua aportada al suelo se reducirá sustancialmente hasta el día siguiente. Por lo tanto, dará más tiempo a que las raíces de las plantas se hidraten bien y retengan el agua con mayor eficacia para cuando salga de nuevo el sol y se eleve la temperatura. Teniendo en cuenta éste efecto, incorporamos un sensor que nos informa de las condiciones lumínicas del invernadero.

Tomando esto como base, el riego se activará siempre que haya una humedad menor a la deseada y la luz incidente sea “baja”, con baja quedan contempladas prácticamente todas las horas del día menos las centradas en torno a mediodía, cuando la luz incidente es considerablemente mayor.

Como medida adicional, tendremos en cuenta que la bomba de agua no puede funcionar en vacío, por lo que aseguraremos que el riego se active únicamente si hay agua suficiente en el depósito de suministro.

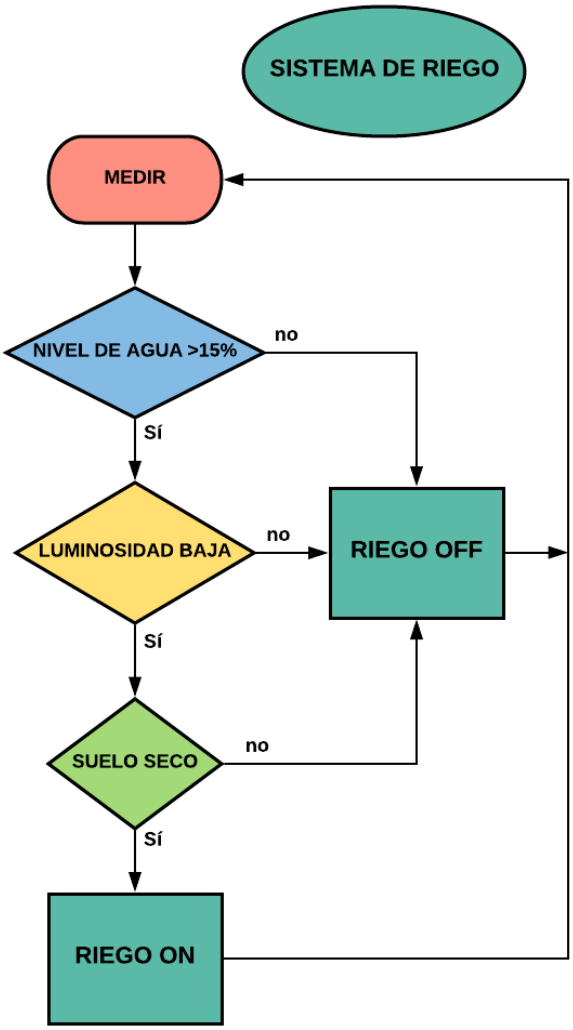


Ilustración 11 - Diagrama de flujo del proceso del Sistema de Riego Automático.

En cuanto al sistema de ventilación, se diferenciará entre condiciones climáticas nocturnas y diurnas. Si la luminosidad es baja, entraremos en el lazo de control nocturno, y en caso contrario en modo diurno. El límite de humedad relativa será el mismo en ambos casos (75%HR) y los límites que cambian son los de temperatura ambiente. Es decir, tanto para condiciones diurnas como nocturnas, se activará el ventilador en el caso en que la temperatura y humedad ambiente aumenten considerablemente.

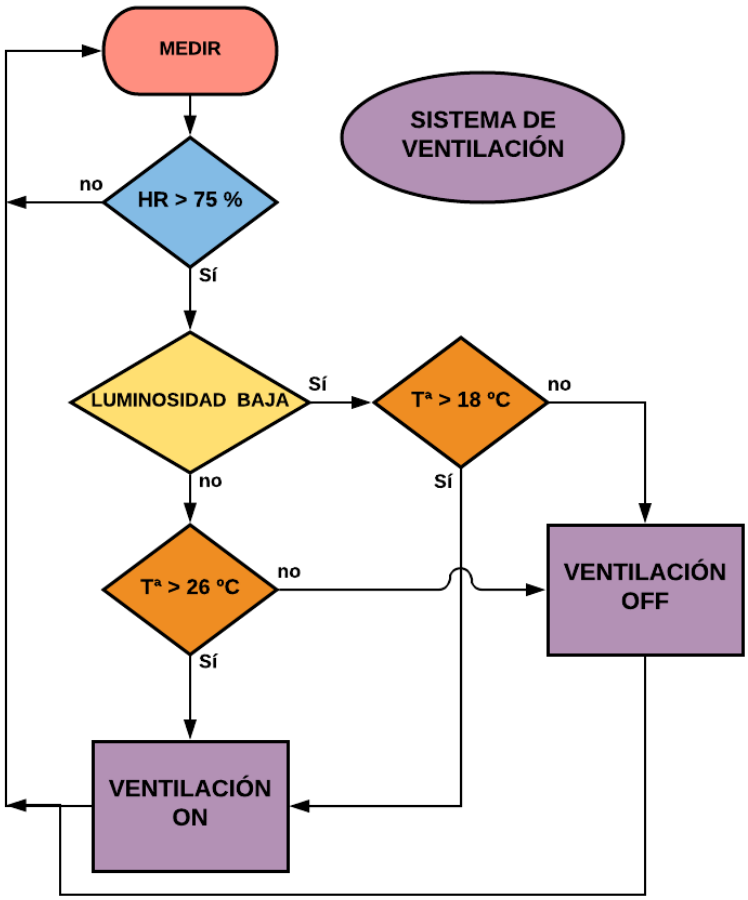


Ilustración 12 - Diagrama de flujo del proceso del Sistema de Ventilación Automática.

3.3.3. NIVELES DE ALERTAS

Las alertas se irán clasificando por niveles y gracias a un menú, según el nivel de alerta, se mostrará al usuario final el mensaje correspondiente de forma local por el display y por publicación en lista en la página web.

Cabe señalar, que por limitaciones del servidor gratuito que se utiliza, sólo podremos enviar un estado de alarma a la vez. Esto implica, que si se producen varias alarmas (por ejemplo: peligro helada, suelo seco y nivel de agua critico), solo se podrá enviar la primera alerta generada. Este problema se solucionaría ampliando el servidor a más conexiones, ya que el servidor libre solo admite hasta 10, y en el proyecto se usa este total, dejando una sola variable para todas las alarmas. Es por eso, que priorizaremos en la clasificación las alertas más importantes. La clasificación de alertas tendrá la siguiente forma:

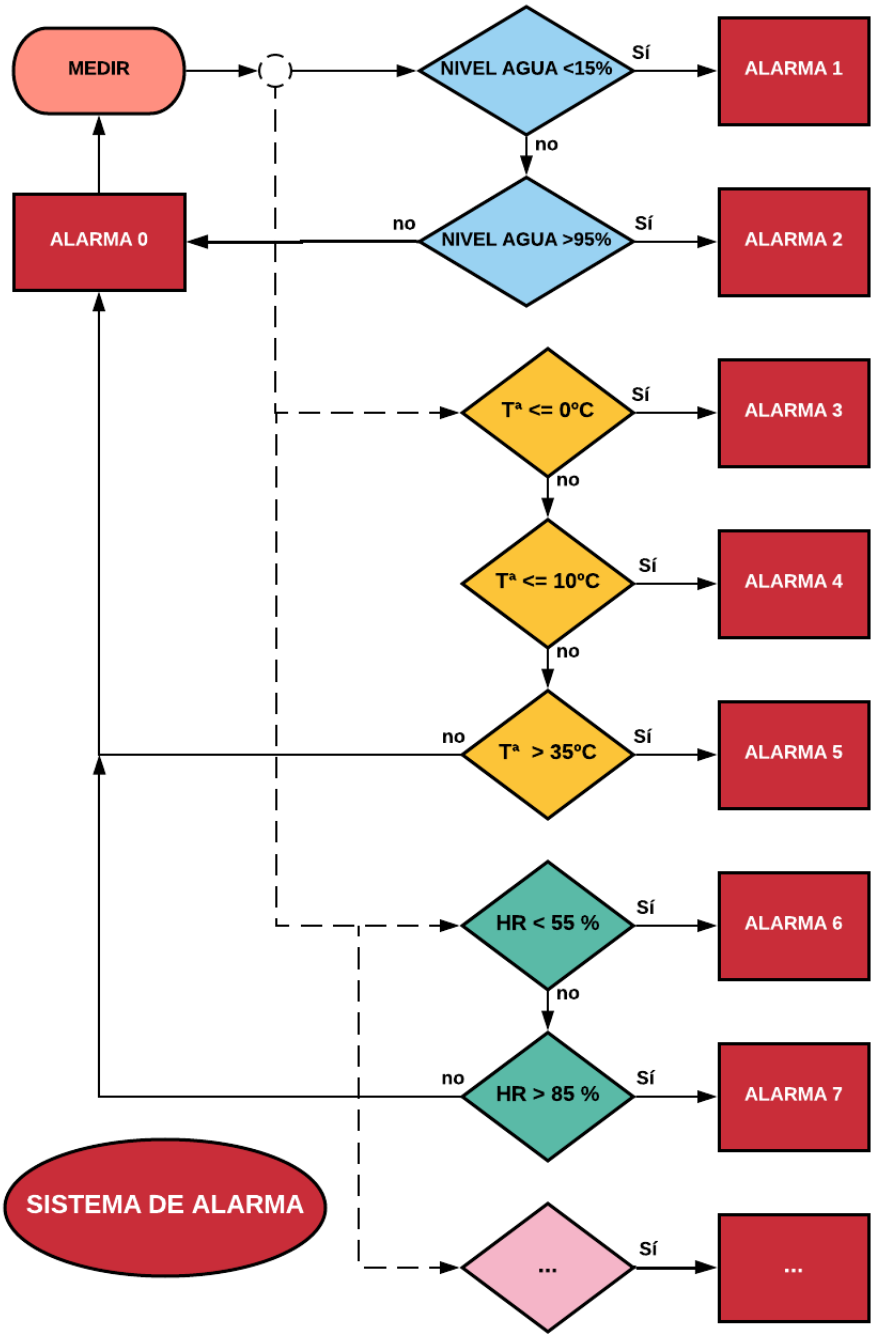


Ilustración 13 - Diagrama de flujo de proceso del Sistema de Alarma.

De manera independiente, se compararán los límites establecidos para alertar sobre los distintos parámetros de control (nivel de agua, temperatura, humedad relativa…). Como todas las alarmas siguen la misma estructura de comparación, se entiende que se pueden añadir más alarmas según el parámetro del que queramos generar una alerta. El sistema es sencillo, una única variable (ALARMA) puede tomar distintos valores (1, 2, 3,…) para su posterior clasificación en un menú de alarmas.

Como una solución al problema señalado, se propone el uso de la plataforma IFTTT, en la que se compararán los parámetros del cultivo publicados en el servidor con los límites de alerta definidos, y se generarán automáticamente los emails informativos de las alarmas en cuestión. Hablaremos de esta plataforma en detalle más adelante.

4. HARDWARE

4.1. SELECCIÓN DE COMPONENTES

Existen múltiples plataformas hardware y software para el desarrollo del Internet de las Cosas, entre ellas Galileo y Edison de Intel, Raspberry PI, Arduino, etc. En este apartado no se pretende poner en manifiesto toda la variedad de placas existentes en la actualidad, por lo que se expondrán más bien, los puntos básicos para la selección del dispositivo de control. Es decir, en función de los requisitos de la aplicación, será más interesante usar una placa u otra, ya que existen diferencias entre ellas, como el número de entradas y salidas, tamaño, potencia, precio, características del microcontrolador…

A la hora de pensar en la plataforma con la que se trabajará surgen tantas dudas como dispositivos hay actualmente. La decisión de decantarse por una u otra dependerá de la marca, precio, rendimiento y funcionalidades, y el hecho de decantarse por una no implica que sea la mejor opción. Cabe destacar, que hay soluciones de proyectos parecidos al presente y a la vez completamente diferentes.

Para seleccionar la plataforma se deberá tener en cuenta qué se quiere conseguir y qué se necesita para su desarrollo. Lo mejor es pensar en la influencia de los siguientes factores, los cuales se han escrito en referencia a [1]:

- **Velocidad del procesador o velocidad de reloj:**

Indica la velocidad a la que se procesa una instrucción, escrita en código máquina procedente de un programa. Lo que implica que cuanto más rápido sea el procesador, más instrucciones ejecutará por segundo. La velocidad de los microprocesadores se mide en decenas de MHz mientras que en un SoC se hace en cientos de MHz o en unos pocos GHz. Si el proyecto no va a necesitar mucho poder de cálculo, bastará con un microcontrolador, sin embargo si hay que manipular grandes cantidades de datos, como supone el procesado de un vídeo en tiempo real, deberíamos emplear una plataforma SoC.

Esto es interesante si por ejemplo queremos añadir una cámara a nuestro proyecto de tal forma que podamos ver el estado del invernadero siempre que queramos. Cabe destacar, que se ha intentado probar con una cámara OV7670 con memoria FIFO, sin embargo, por falta de tiempo no se ha podido implementar.

- **Memoria RAM:**

La RAM es la memoria de trabajo del sistema. A más RAM, más cosas se podrán hacer o más facilidad se tendrá a la hora de codificar un algoritmo. Es complicado determinar la cantidad exacta de memoria RAM que se necesita antes de empezar un proyecto, ya que este es ampliable en todos los sentidos. Aun así, se podrían descartar los microcontroladores que tengan menos de 1KB de RAM. Si se trabajase con protocolos encriptados lo recomendable es partir de 4KB. Y si se emplea un SoC, será como mínimo de 256MB, sobretodo si se utiliza Linux.

- **Redes:**

Si se trabaja en IoT es importante determinar cómo se va a conectar el dispositivo a Internet. La conexión más sencilla y barata es emplear un cable Ethernet, pero es una conexión física. Una conexión inalámbrica solucionaría esta limitación pero su configuración es más compleja. La solución más popular es el WiFi, pero es la más cara y la que más energía consume. Otras conexiones inalámbricas de corto alcance (como ZigBee o el Bluetooth 4.0) pueden resultar más baratas, pero el WiFi funciona mejor si el ancho de banda es limitado.

- **USB:**

Si el dispositivo se pudiese conectar a un ordenador por USB contaría con potencia y acceso a la red. Se puede elegir un microcontrolador o SoC que incluya soporte para USB ahorrando la necesidad de añadir un nuevo chip al circuito.

- **Consumo energético:**

Cuanto más rápido es el procesador, más energía consume, lo que supone un problema para los dispositivos portátiles o que tengan que emplear sistemas no convencionales de alimentación, como por ejemplo paneles solares. Conviene reducir el consumo aunque se pueda conectar a una toma de corriente eléctrica. Hay procesadores que pueden pasar a un estado de hibernación y consumir muy poca energía, por lo que se puede emplear un procesador muy rápido para llevar a cabo las tareas más pesadas y luego volver al estado de aletargamiento para minimizar el consumo de energía.

- **Trabajar con sensores y otros circuitos:**

El dispositivo que se busca deberá, además de conectarse a Internet, interactuar con sensores y actuadores, que serán los encargados de obtener datos de su entorno, mostrar los datos recogidos y actuar en consecuencia a una lógica de control definida. Para conectar el circuito, se tendrá que utilizar algún tipo de bus periférico, como SPI o I2C, y también se deberá garantizar la lectura y/o escritura tanto de datos digitales (como uso de GPIOs para entradas y salidas On/Off) como analógicos (gracias al uso de DACs y CADs para entradas y salidas de voltaje variable). Los microcontroladores o las placas SoC que se encuentran en el mercado ofrecen diferentes combinaciones de estas interfaces.

Selección final:

En los comienzos del proyecto se planteó usar un Arduino Mega (ya que era el disponible en ese momento) y un módulo WiFi (ESP8266) para la conexión a Internet. Sin embargo, a medida que avanzaba el proyecto, era necesario hacer un cambio. ¿Por qué utilizar dos componentes si puedo hacerlo con uno? El problema del Arduino Mega era que no tenía capacidad para conectar a Internet y en el caso del SoC ESP8266 el problema residía en no poder conectar más de un sensor analógico, por disponer únicamente de un puerto con ADC. Por esta razón, se buscó otra solución que resolviese el problema, encontrando al hermano mayor del ESP8266, el ESP32. Una solución compatible con el código generado hasta el momento en Arduino, con más puertos ADC para los sensores analógicos y con la capacidad para gestionar toda lógica de control. A continuación se expone más en detalle este dispositivo.

4.1.1. MÓDULO ESP32-DevKitC

El ESP32 es un SoC (System on Chip) diseñado por la compañía china Espressif y fabricado por TSMC. Integra en un único chip un procesador Tensilica Xtensa de doble núcleo de 32bits a 160Mhz (con posibilidad de hasta 240Mhz), conectividad WiFi y Bluetooth. [14]

El ESP32 es el sucesor del ESP8266, añade muchas funciones y mejoras respecto a este último, como son mayor potencia, Bluetooth 4.0, encriptación por hadware, sensor de temperatura, sensor hall, sensor táctil, reloj de tiempo real (RTC), más puertos, más buses... Además, se han desarrollado firmwares, documentación, herramientas y, aunque su soporte es aún inferior al del ESP8266, en la actualidad es fácil encontrar tutoriales sobre el mismo y continuamente se publican nuevos artículos.

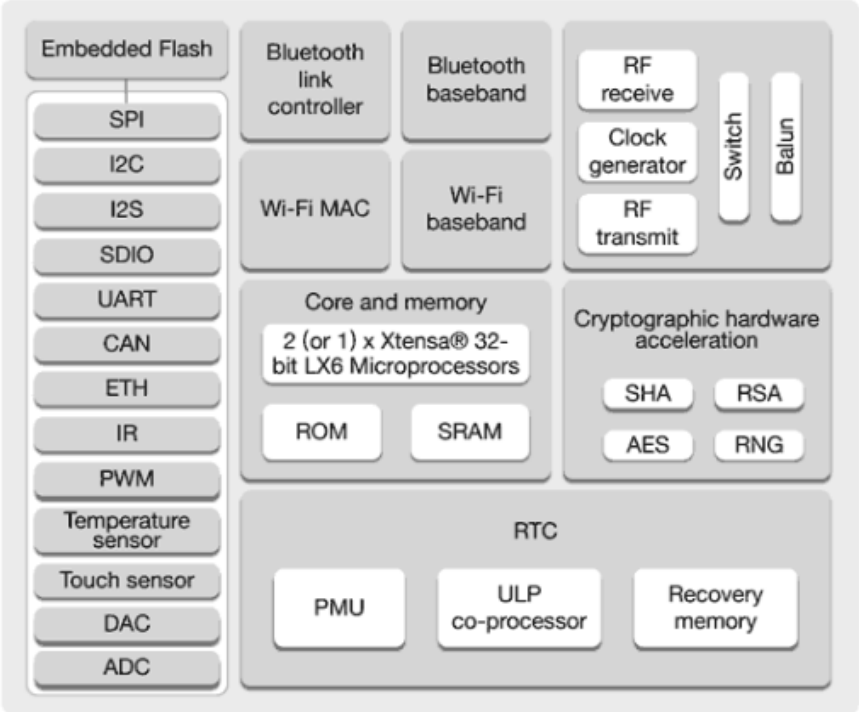


Ilustración 14 - Estructura del ESP32.

También empiezan a verse artículos y productos comerciales que emplean el ESP32 como núcleo. No obstante, de momento, encontramos más artículos que montan el ESP8266, seguramente por su bajo precio o por llevar más tiempo en el mercado.

Ocupa un lugar destacado en aplicaciones IoT por su enorme potencial. A continuación se muestra en una tabla resumen sus características técnicas, aprovechamos también para ver las diferencias con el ESP8266. Características obtenidas de las referencias [15] y [16].

CARACTERÍSTICAS	ESP8266	ESP32
Procesador	Tensilica LX106 32 bits a 80 MHz (hasta 160 MHz)	Tensilica Xtensa LX6 32 bits Dual-core a 160 MHz (hasta 240 MHz)
SRAM	80 kB (40 disponibles)	520 kB
Memoria Flash	Hasta 4 MB	Hasta 16 MB
ROM	No	448
Alimentación	3.0 a 3.6V	2.3 a 3.6V
Rango temperaturas	-40ºC a 85ºC	-40ºC a 125ºC
Consumo promedio	80 mA	80 mA, 225 max

Consumo modo Deep Sleep	20 µA (RTC+memoria RTC)	2.5 µA
WiFi	802.11 b/g/n/e/i (hasta +20dBm) WEP, WPA	802.11 b/g/n/e/i (hasta +20dBm) WEP, WPA
Encriptación	TLS 1.2 (por software)	AES, SHA, RSA, ECC (por hardware)
Ethernet	No	10/100 Mbps MAC
Bluetooth	No	v4.2 BR/EDR y BLE
UART	2	3
I2C	1	2
SPI	2	4
GPIO	17	34
PWM	4	16
ADC	1 (10 bits)	2 (12 bits) con preamplificador de bajo ruido, hasta 60 dB
DAC	No	2 (8 bits)
1-wire	por software	por software
I2S	1	2
CAN bus	No	1(2.0)
Sensor temperatura	No	Si
Sensor efecto HALL	No	Si
Sensor capacitivo	No	10
IR	Si	Si
Temporizadores	3	4 (64 bits)
Gen. nº aleatorios	No	Si
Encriptación de la memoria flash	No	Si
Arranque seguro	No	Si

Tabla 5 - Características y diferencias entre ESP8266 y ESP32

Vemos que las características del ESP32 son muy superiores, sin embargo para muchas aplicaciones puede ser suficiente con el ESP8266, y por ende mucho más económico. El fabricante proporciona este módulo soldado a una placa de desarrollo con un conector microUSB y un par de microinterruptores, conocida como ESP32-DevKitC.

4.1.2. SENSÓRICA

4.1.2.1. SENSOR TEMPERATURA Y HUMEDAD

Para el control de la temperatura y humedad relativa encontramos los sensores DHT11 y DHT22. Para el prototipo se eligió este tipo de sensor por la sencilla razón de que con un único sensor se puede medir temperatura y humedad. Una particularidad de estos sensores es que la señal de salida es digital, por lo tanto, lo tendremos que conectar a pines digitales.

Ilustración 15 - Sensores DHT22 y DHT11.

Llevan un pequeño microcontrolador interno para hacer el tratamiento de señal. Los DHT11 y 22 se componen de un sensor capacitivo para medir la humedad y de un termistor para medir temperatura. La principal diferencia entre ambos es que el ciclo de operación es menor en el DHT11 que en el DHT22, y que el DHT22 tiene rangos de medida más amplios y mayor resolución. La diferencia en precio es pequeña, algo más caro el DHT22, pero merece la pena decantarse por éste. Ambos sensores están calibrados en laboratorio y tienen una buena fiabilidad. Esto supone una gran ventaja, ya que simplifican las conexiones a realizar en la placa. [17]

En el mercado existen tres tipos de encapsulado:

- El sensor suelto, con un encapsulado azul y cuatro pines disponibles para conectar. (Será necesario añadir la resistencia pull-up).
- El sensor con una placa soldada, con tres pines disponibles para conectar y una resistencia pull-up (normalmente de 4,7-10 kΩ).
- El mismo formato que el anterior, pero con un condensador de filtrado (normalmente de 100 nF).

La captación de los valores de temperatura y humedad en el prototipo de invernadero realizado se ha hecho escogiendo el sensor DHT22 con resistencia pull-up para evitar tener que soldarla.

A continuación, se muestra una tabla con las características de estos sensores y la comparativa entre ellos, obtenida de la referencia [18].

Parámetro	**DHT11**	**DHT22**
Alimentación	3Vdc ≤ Vcc ≤ 5Vdc	3.3Vdc ≤ Vcc ≤ 6Vdc
Consumo máximo durante conversión (al tiempo que solicita los datos)	2,5 mA	2,5 mA
Señal de Salida	Digital	Digital
Rango de medida Temperatura	De 0 a 50 °C	De -40°C a 80 °C
Precisión Temperatura	±2 °C	<±0.5 °C
Resolución Temperatura	0.1°C	0.1°C
Rango de medida Humedad	De 20% a 90% RH	De 0 a 100% RH
Precisión Humedad	4% RH	2% RH
Resolución Humedad	1%RH	0.1%RH
Tiempo de respuesta	1s	2s
Tamaño	12 x 15.5 x 5.5mm	14 x 18 x 5.5mm

Tabla 6 - Características y comparativa sensores DHT

En lo que se refiere al pinout, los pines del DHT22 siguen el mismo orden:

De izquierda a derecha se tiene:

1. VCC: Alimentación del sensor
2. Señal: Información de temperatura y humedad
3. NC: No Conexión
4. GND: Masa del circuito

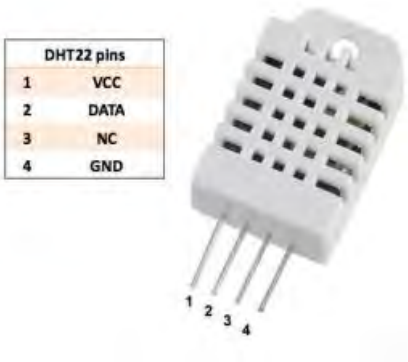


Ilustración 16 - Pines del sensor DHT22.

4.1.2.2. HIGRÓMETRO YL69

Buscando un sensor para medir el nivel de humedad del terreno de cultivo, nos encontramos con el higrómetro YL-69, muy barato y compatible con Arduino. Se trata de un sensor capaz de medir la humedad del suelo aplicando una pequeña tensión entre sus terminales. La corriente que circula entre ellos depende de la resistencia que se genera en el suelo, que a su vez depende de la humedad. La sonda YL-69 es alimentada mediante dos cables por el módulo YL-38. Dicho módulo incluye un comparador LM393, un led de encendido y otro de activación de salida digital. Dispone de cuatro pines: alimentación (VCC), tierra (GND), salida digital (D0) y salida analógica (AO). La salida analógica proporciona un valor de voltaje en función del nivel de humedad, mientras que la salida digital para de estado bajo a alto cuando se ha alcanzado el nivel ajustado mediante potenciómetro que regula el nivel de humedad. Referencias de los datos obtenidos en [19, 20].

Características:

- Voltaje de entrada: 3.3 - 5 V
- Voltaje de salida: 0 – 4.2 V
- Corriente (Imáx): 35 mA
- Dimensiones YL-69: 60 x 30 mm
- Dimensiones YL-38: 30 x 16 mm
- Salida analógica (A0): Entrega una tensión proporcional a la humedad.
- Salida digital (D0): Ajusta el nivel lógico de la salida mediante el potenciómetro.

Ilustración 17 - Higrómetro YL69.

El funcionamiento es sencillo. Aplicando una pequeña tensión entre los terminales del módulo YL-69, se hace pasar una corriente que depende de la resistencia generada en el suelo, y ésta depende en gran medida de la humedad. A medida que aumenta o disminuye la humedad, la corriente aumenta o disminuye en consecuencia, coincidiendo un 0 con humedad total (sumergido en agua) y el máximo de la escala con un suelo muy seco (o en el aire). La escala depende del ADC de nuestro controlador, en nuestro caso será una escala de 0 a 4095, ya que el ADC que proporciona el ESP32 es de 12 bits como vimos con anterioridad. En el caso de usar un ESP8266, el ADC de los mismos es de 10 bits, lo que nos proporciona una escala de valores de 0 a 1023.

Por otro lado, un suelo ligeramente húmedo daría valores típicos de 2400-2800, mientras que un suelo seco tendrá valores de 3200-4095. Para no trabajar directamente con esta escala, se propone una conversión a una escala de 0 a 100, para trabajar en una lógica de control basándonos en un tanto por ciento de humedad.

En un proyecto como el presente, este sensor cumple con las expectativas, sin embargo para algo más industrial, necesitaremos otro tipo de higrómetro del que hablaremos más adelante.

4.1.2.3. FOTORRESISTENCIA LDR

Para el control de la luminosidad en el prototipo de invernadero se usa una fotocélula o LDR, la cual actúa como una resistencia, cambiando su valor conforme al nivel de luz que incide sobre ella. La relación entre la resistencia de la LDR y la iluminación puede modelarse mediante la ecuación 1:

$$R_L = R_0 \left(\frac{L_0}{L} \right)^{\alpha}$$

Ecuación 1 - Modelo fotorresistencia

Donde L es la iluminación (en lux), α es una constante que depende del material y RL y R0 son las resistencias a los niveles L y L0, respectivamente.

Respecto a las características y funcionamiento de una fotorresistencia se ha encontrado información en la referencia [21].

Ilustración 18 - Fotorresistencia

Características:

- Voltaje máximo: 150 V
- Temperatura: [-30,70] ºC
- Pico espectral: 540 nm
- Salida analógica
- Resistencia oscuridad: 50 kΩ
- Resistencia luz brillante: 500 Ω
- Gamma = 0.9
- Tiempo de respuesta = 25 ms

Debemos convertir ese valor de resistencia en un voltaje para poder medirlo analógicamente, la forma más sencilla de hacerlo es creando un divisor de tensión, junto con una resistencia fija.

$$V_{AO} = \frac{R_{LDR}}{R_{LDR} + 1k} \cdot 5\ V$$

Ecuación 2 - Conversión voltaje fotorresistencia

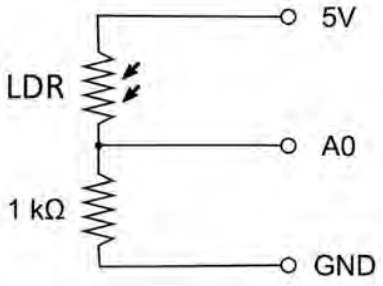


Ilustración 19 - Divisor de Tensión.

Cuando detecta luminosidad alta, la resistencia de la LDR es muy baja en comparación con la fija, por lo que la salida analógica tiende a la tensión de alimentación, mientras que para situaciones de oscuridad, la resistencia que presenta la LDR es mucho mayor que la fija, por lo que la salida tiende a cero.

Cabe destacar, que actualmente ya se venden módulos con LDR con la resistencia fija incluida. En el prototipo, la resistencia va soldada al resto del circuito para que sea más fácil reponer la LDR en caso de deterioro, ya que la colocaremos fuera del sistema electrónico.

4.1.2.4. SENSOR DE PH

Anteriormente hablamos de la importancia que adquiere controlar el nivel de pH de riego para nuestro cultivo, por lo que nos centraremos ahora en el sensor que emplearemos en nuestro sistema.

Para medir el pH del agua de riego empleamos un sensor de pH analógico, especialmente diseñado para controladores como Arduino. Como ya comentamos, el sensor se compone de una sonda y un circuito electrónico de acondicionamiento de señal. El pH se mide por la diferencia de potencial entre los dos electrodos que componen la sonda: un electrodo de referencia (plata / cloruro de plata) y un electrodo de vidrio es sensible al ión de hidrógeno. A continuación se muestra dicho sensor:

Ilustración 20 - Sensor de pH. Sonda y circuito de acondicionamiento

Ilustración 21 - Circuito de acondicionamiento sensor pH. Vista planta.

Especificaciones:

- Potencia: 5.00V
- Tamaño del módulo: 43 x 32 mm (1.69x1.26 ")
- Rango de medición: 0 - 14PH
- Temperatura de medición: 0 - 60 ºC
- Precisión: ± 0.1pH (25ºC)
- Tiempo de respuesta: <1 min
- Sensor de pH con conector BNC
- Controlador pH 2.0 (3 pines)
- Potenciómetro de ajuste de ganancia

- Indicador de encendido LED pH significa potencial de hidrógeno y nos dice si una solución o sustancia es BÁSICA (pH superior a 7,0), ACIDIC (pH inferior a 7,0) o NEUTRAL (pH de 7.0).

Las soluciones ácidas contienen una mayor concentración de iones de hidrógeno (H+), mientras que las soluciones básicas contienen una mayor concentración de iones hidroxilo (OH-). Estos son algunos ejemplos de sustancias cotidianas y su pH:

Sustancia	**pH aproximado**
Jugo de limón	2,4 – 2,6
Coca cola	2,5
Vinagre	2,5 – 2,9
Zumo de naranja o manzana	3,5
Cerveza	4,5
Café	5,0
Té	5,5
Leche	6,5
Agua	7,0
Saliva	6,5 – 7,4
Sangre	7,38 – 7,42
Agua de mar	8,0
Jabón	9,0 – 10,0
Lejía	13

Tabla 7 - Niveles de pH de algunas sustancias

Lo primero que se debe hacer antes de comenzar a medir, es calibrar el sensor. Como se puede observar, en el circuito hay dos mandos. El que está más conectado al conector BNC de la sonda es el que regula el desplazamiento, el otro es el límite de pH.

• **Desplazamiento:** la media (rango) de la sonda oscila entre valores negativos y positivos. El 0 representa un pH de 7.0. Para poder usarlo con Arduino, este circuito agrega un valor de compensación al valor medido por la sonda, de esta manera el ADC solo tendrá que tomar una muestra de valores positivos de voltaje. Por lo tanto, forzaremos un pH de 7.0 desconectando la sonda del circuito y conectando la parte interior del conector BNC con el exterior. Con un multímetro, medimos el valor del pin Po y ajustamos el potenciómetro a 2,5 V.

• **Límite de pH:** este botón sirve para configurar un valor límite del circuito para el sensor de pH que hace que el LED rojo se encienda y la señal del pin se ponga en estado alto.

Además, se debe calcular la conversión del voltaje que nos dará el sensor de pH a lo que necesitaremos dos valores de referencia de pH y medir el voltaje devuelto por el sensor en el pin Po. Es mejor usar una solución de calibración activada, también hay en líquido, pero es más fácil mantenerla encendida. Estas soluciones se venden en diferentes valores, aunque las más comunes son pH 4.01, pH 6.86 y pH 9.18.

Usando los sobres con pH 4.01 y pH 6.86 obtenemos los voltajes en el pin Po 3.04 V y 2.54 V, respectivamente. El sensor es lineal, de modo que al tomar dos puntos, podemos deducir la ecuación para convertir el voltaje medido a pH. La fórmula general sería y = mx + b, por lo que tenemos que calcular 'm' y 'b', ya que 'x' sería la tensión e 'y' el pH. El resultado es y = -5.70 x + 21.34.

Para conectarlo con ESP32, se necesita una entrada analógica, potencia (5V) y dos GND, que en realidad están separados en el circuito del sensor, pero se puede usar el mismo.

Los datos, características e información recogida sobre el sensor de pH sean obtenido de las siguientes referencias: [22, 23, 24].

4.1.2.5. ULTRASONIDOS

El sensor empleado es el HC-SR04, es un módulo que incorpora un par de transductores de ultrasonido que se utilizan de manera conjunta para determinar la distancia del sensor con un objeto colocado enfrente de este. Su funcionamiento se basa en medir cuanto tiempo tarda en viajar una onda ultrasónica desde que sale del sensor (por el transductor trigger) hasta que regresa (por el transductor echo).

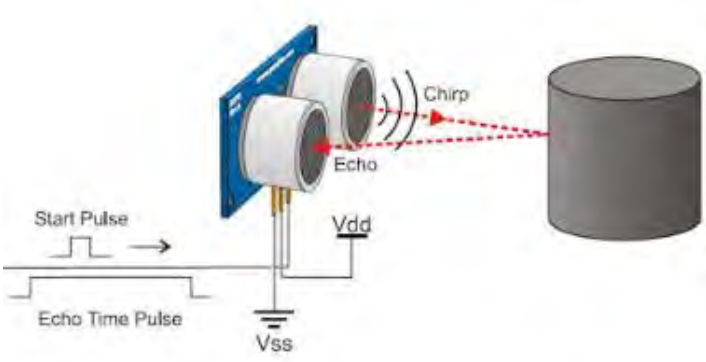


Ilustración 22 - Funcionamiento del sensor de Ultrasonidos.

Puede detectar objetos desde 2 centímetros hasta 400 cm, con una precisión de 3 mm y un ángulo de cobertura de medición de 15 grados. El funcionamiento es sencillo, en el momento que el Trigger recibe del microcontrolador una señal de 5V y duración de 10us, la membrana de éste emite una onda sonora de 8 pulsos y 40 Hz. Esa onda sonará rebotará en el objeto frente al sensor y la recibirá el segundo transconductor (Echo). Cuando el sensor recibe los 8 pulsos, calcula el tiempo que ha tardado y desde el pin Echo emitirá una señal de 5V proporcional al tiempo calculado para que la reciba el microcontrolador.

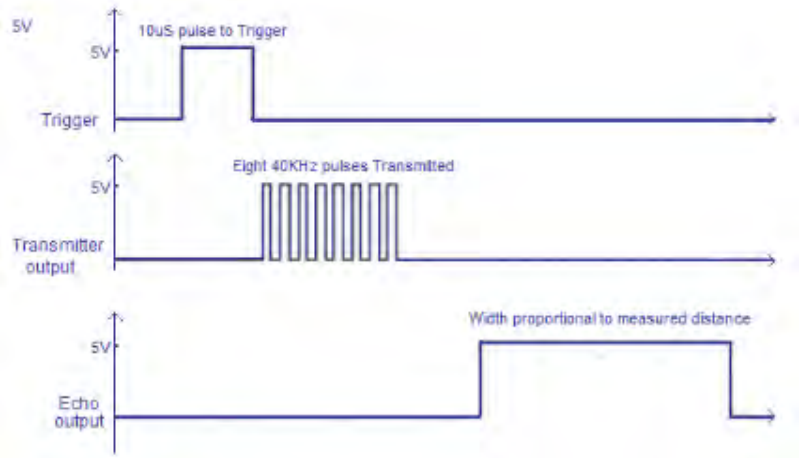


Ilustración 23 - Envío y recepción de pulsos de señal del sensor de Ultrasonidos.

Características:

- Vcc = 5 V
- Imáx = 15 mA
- 4 pines: Vcc, Trigger, Echo y GND
- Frecuencia de trabajo = 40 kHz
- Ángulo efectivo < 15 °
- Distancia = [2,400] cm
- Resolución = 0.3 cm
- Dimensiones = 45x20x15

Ilustración 24 - Sensor de Ultrasonidos.

Para conocer el % de agua disponible en el depósito de suministro, lo primero es tomar dos medidas: Nivel de depósito en vacío (0% nivel de líquido), nivel de depósito lleno (lo que corresponderá a un 100% de nivel de líquido). Además, se entiende que el sensor leerá la cantidad de espacio vacío que hay en el depósito (es decir, mide la distancia que hay hasta alcanzar el objeto, en éste caso el agua). Sabiendo esto, podremos calcular la capacidad total del depósito, sustrayendo la cantidad de espacio vacío a la capacidad total del mismo.

Es necesario que el depósito sea homogéneo, ya sea cilíndrico o cúbico, pero siempre el diámetro o perímetro de la base debe ser el mismo que la parte superior del recipiente.

Por otro lado, hay que tener en cuenta el ruido que se genera debido al movimiento del recipiente y el agua, incluso debido al propio sensor o a la placa que utilicemos. El ruido se traduce en una variación o fluctuación en las medidas que vamos obteniendo a través del sensor. Para su eliminación se suele aplicar un filtro paso bajo que elimina las frecuencias más bajas donde se encuentra el ruido. En el presente proyecto, este ruido se disminuirá aplicando una media de los valores que se van obteniendo, y en función de la cantidad de muestras que se recojan, podremos tener un valor con más o menos precisión.

Es decir, si se toma un gran número de muestras, se obtendrá un valor de gran precisión, sin embargo se tardará mucho en obtenerlo. Si por el contrario, se escoge hacer la media con un pequeño número de muestras, la precisión será muy baja, pero se obtendrá el valor final mucho antes. Es en este punto en el que hay que tomar una decisión, precisión o rapidez. Cabe destacar, que interesa mucho saber la capacidad del depósito, ya que si ésta es muy pequeña un pico en la medida (como por ejemplo una fuga inesperada de agua o un movimiento brusco en el depósito) va a afectar en gran medida a la media realizada, mientras que en un depósito de más capacidad la variación del nivel agua es sensiblemente menor, y por tanto la media puede no variar tanto. En el proyecto se hará una media de 10 muestras porque interesa la rapidez para ver resultados rápidos en los ensayos, sin embargo, al tratarse de un depósito pequeño, deberíamos tomar muchas más muestras para asegurar una buena medida.

Imágenes recogidas de la referencia [25]. Información y características del Sensor y ultrasonidos recogidas de la referencia [26].

4.1.3. ACTUADORES

En este apartado vamos a incluir los mecanismos que nos ayudarán a controlar las variables del entorno del invernadero, siendo el agente en contacto con el mismo. A continuación, se muestran los dispositivos empleados en el prototipo para el sistema de riego y ventilación.

4.1.3.1. BOMBA DE AGUA PARA RIEGO

En el prototipo diseñado se implementa un sistema de riego por goteo en el que el actuador empleado es una bomba de agua sumergible sin escobillas de 12 V DC. Su objetivo no es otro que extraer agua de un depósito en el que se encuentra sumergida e impulsarlo hacia un tubo exterior mediante su potencia eléctrica.

Ilustración 25 - Bomba de agua 12V DC.

La bomba elegida tiene las siguientes características, según [27]:

- Marca: LEDGLE
- Material: ABS
- Voltaje: DC 12V
- Tasa de energía: 4.8W
- Resistencia al agua: IP68 a prueba de agua
- Consumo: 400mA
- Altura máxima de elevación: 3 m
- Flujo máximo: 240L / H
- Diámetro de entrada / salida: 8 mm
- Volumen de ruido: ≦ 40dB
- Temperatura máxima del agua: 60°C
- Dimensiones del producto: 5,5 x 3,4 x 4,1 cm ; 68 g
- Longitud del cable 17.7 " / 45cm
- Referencia del fabricante: 10L5AL24488TJTSPOF1BKW755

Para su correcto funcionamiento, es necesario que la bomba esté sumergida y no trabaje en vacío o podría romperse por cavitación. Por ello, para evitar en la medida de lo posible que esto ocurra, se impone en la lógica de control (código) que se desactive el riego siempre que el depósito alcance un nivel inferior al 15% de su capacidad. Además alertaremos de esto gracias a la gestión de alarmas.

Se ha elegido esta bomba como ejemplo para el prototipo, de modo que la activaremos con un módulo relé conectado al controlador. El relé cerrará el circuito si el riego se activa en consecuencia a la lógica de control implementada.

Señalar que se ha decidido emplear este sistema pensando en una futura extrapolación a un invernadero de mayores dimensiones, ya que en ese caso, se necesitará una bomba de riego de mayor potencia con capacidad de suministro a mayores distancias.

4.1.3.2. VENTILADOR PARA EXTRACCIÓN DE AIRE

En el prototipo diseñado se implementa un sistema de ventilación mecánica simple en el que el actuador empleado no es más que un ventilador de 12 V DC. Su objetivo es extraer el aire cargado (exceso de temperatura y humedad) del interior del invernadero.

El ventilador dispone de un rotor con alabes que se acciona por un motor eléctrico. Al girar el motor, las aspas impulsan las moléculas de aire aumentando su energía cinética sin casi variación de volumen, motivo por el cual se consideran máquinas hidráulicas en lugar de turbo máquinas. En electrónica se encuentran multitud de ventiladores para la disipación de calor, por ejemplo para ordenadores. Por otro lado, en industria se emplean no solo para el control de temperatura, sino también en instalaciones de climatización, gases, humos, refrigeración, maquinaria, entre otros. A continuación se muestran las características del ventilador empleado, según [28]:

Ilustración 26 - Ventilador 12V DC.

- Tamaño: 40X40X10mm
- Conector: 2pin(2.0mm)
- Voltaje de operación: 12V DC
- Corriente de operación: 0.11±0.02 A
- Velocidad de operación: 6500 ±10%rpm
- Flujo de aire: 9.8CFM
- Ruido: 15±10%dBA
- Largo del cable: 13cm
- Peso: 14g

Cabe destacar, que se deja abierta la opción de colocar un ventilador de mejores características, como la bomba de agua, para una posible ampliación del invernadero. Por otro lado, señalar que en un futuro se desea implementar un ventilador adicional en la parte inferior del invernadero para la inclusión de aire del exterior, para un buen control sobre la climatización del invernadero. Es decir, se desea implementar un sistema de ventilación activa.

4.1.4. OTROS COMPONENTES

4.1.4.1. DISPLAY I2C LCD

Cuando diseñamos un proyecto con arduino es muy común incluir un display para poder monitorear cierta información. Existe una amplia gama de pantallas LCD sobre las que elegir, son baratas, consumen poco y tienen diversas medidas. El principal inconveniente de todas ellas es la cantidad de pines necesarios para conectarla, por esta razón encontramos en el mercado una solución muy útil, el adaptador Serial I2C PCF8574. Con este adaptador podemos convertir cualquier pantalla estándar de 16×2 caracteres o 20×4 en una pantalla LCD serial que se conecta al ESP32 con solo 2 pines digitales (SDA y SCL) al bus I2C.

Por descontado, podemos comprar directamente la pantalla con el adaptador ya soldado, de esta forma evitamos tener errores a la hora de soldarlo nosotros mismos.

Ilustración 27 - Adaptador de pantalla LCD I2C.

Ilustración 28 - Display LCD I2C. Vistas delantera y trasera

A continuación se muestran las características del adaptador pantalla LCD 16×2 serial I2C PCF8574:

- Basado en el expansor I/O PCF8574.
- Solamente 4 lineas en total (incluyendo alimentación) para conectar una pantalla al microcontrolador.
- Soporta múltiples dispositivos en el mismo bus I2C.
- Jumpers soldables para selección de dirección en el bus I2C, pueden coexistir varios de estos módulos en el mismo bus I2C.
- Compatible con pantalla LCD 16×2 o 20×4.
- Utiliza el protocolo I2C, por lo que puede compartir el bus con otros dispositivos.
- Se pueden colocar varias pantallas en el mismo bus.
- Control de la iluminación del display por software y hardware (jumper).
- Librería dispone para arduino que facilita el uso de este dispositivo.

Para comprender el uso y funcionamiento de este Display LCD, se ha recogido información, incluidas las características técnicas del mismo, en tres fuentes distintas: [29], [30], [31].

4.1.4.2. MÓDULO RELÉS

Un relé es un dispositivo electromecánico que sirve para controlar aparatos eléctricos que funcionan con voltajes de hasta 220V. Es decir, el relé permite a un procesador de bajo voltaje de funcionamiento, controlar cargas a un nivel de tensión o intensidad muy superior a las que su electrónica puede soportar. Con un relé es posible encender o apagar cargas de corriente alterna a 220V e intensidades de 10A, así como cargas de hasta 30V de continua.

Ilustración 29 - Módulo de dos relés. [32]

Características:

- Alimentación: 3.3 V - 5 V
- Número de relevadores: 2 de 1 polo 2 tiros
- Número de canales: 2 canales independientes protegidos con optoacopladores
- Cada canal máximo convierte la corriente: 3 A
- Contactos independientes para conexionado
- El voltaje de la bobina del relé es de 5 VDC
- Voltaje de carga: 220 V AC o 30 V DC
- Corriente máxima de carga: 10 A
- Corriente de control: para cada canal de 15 a 20 mA
- Leds indicadores de funcionamiento del módulo y para cada canal (enciende cuando la bobina del relé esta activa)
- Terminales de conexión de tornillo (clemas)
- Terminales de entrada de señal lógica con headers macho de 0.1″
- Vida eléctrica y mecánica: 100.000 ciclos

- Temperatura de trabajo: -30° C a +85° C
- Tamaño: 4.7 x 2.9 x 1.8 cm

El opto acoplador es un dispositivo que aísla galvánicamente el circuito primario y secundario, lo que supone una protección adicional para el módulo frente a un fallo catastrófico en la placa del relé. El circuito primario está formado por un electroimán (consta de una bobina arrollada a un núcleo metálico), mientras que el circuito secundario, encargado de alimentar la carga, está formado por unos contactos eléctricos instalados en láminas de metal flexible. Estos contactos son tres: Común (C), Normalmente Abierto (NO) y Normalmente Cerrado (NC). Dos de ellos son fijos y uno móvil encargado de cerrar el circuito con uno de los contactos fijos.

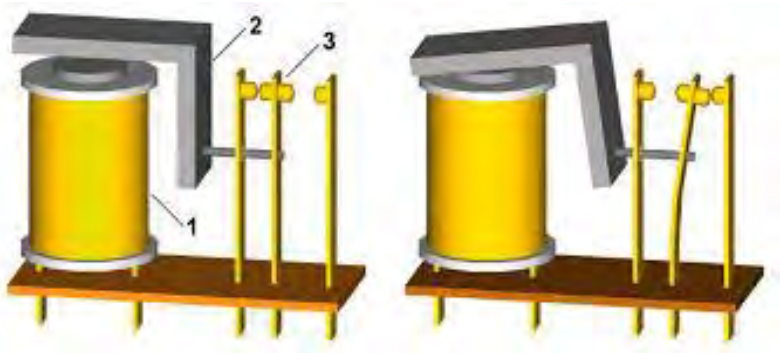


Ilustración 30 - Funcionamiento relé. [33]

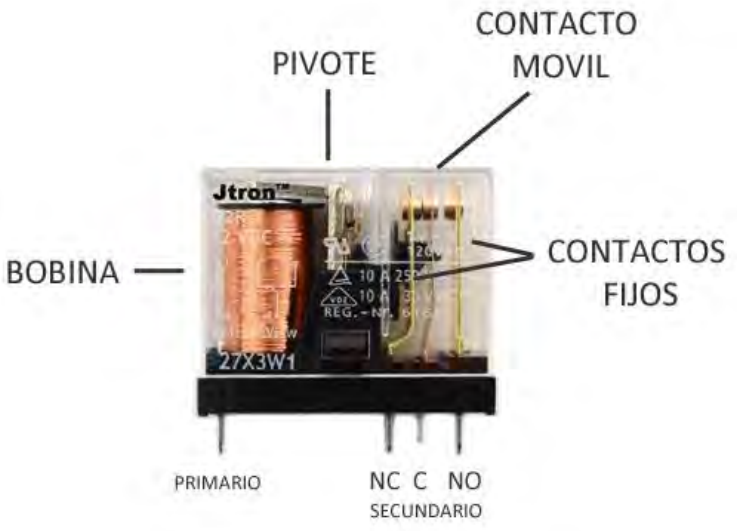


Ilustración 31 - Partes de un relé. [34]

El funcionamiento es sencillo, el circuito primario recibe una señal electrónica de baja tensión del microcontrolador, la corriente circula por la bobina del circuito primario generando un campo magnético que hace pivotar una armadura, que a su vez empuja al contacto móvil, cerrando el circuito con el contacto fijo NO. Mientras, se separa y abre el circuito con el terminal NC.

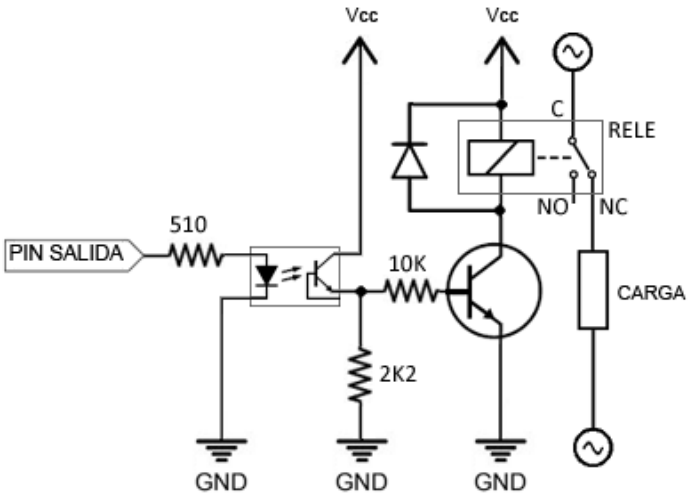


Ilustración 32 - Esquemático de un relé. [34]

Por tanto, para controlar la carga con un relé como si fuera un interruptor siempre se conectará uno de los polos al terminal C (común), que está unido al contacto móvil del secundario. El otro polo de la carga se debe conectar a uno de los terminales NO o NC, en función de si queremos que al entrar en funcionamiento el relé el circuito se cierre (carga encendida) o se abra (carga apagada). Funcionamiento de un relé referenciado en [34].

4.2. DIMENSIONAMIENTO PANEL FOTOVOLTAICO

4.2.1. ENERGÍA FOTOVOLTAICA

Para la autonomía del sistema diseñado se propone el uso de energía solar fotovoltaica, un tipo de energía renovable usada para generar electricidad.

La energía fotovoltaica transforma la radiación solar de forma directa en electricidad usando paneles formados por células fotovoltaicas. En los paneles, la energía de los fotones se transmite a los átomos de silicio de las células fotovoltaicas, los electrones reciben la energía que los excita generando movimiento entre ellos y por tanto electricidad.

La tecnología fotovoltaica es modular, no contamina, y los gastos de mantenimiento y explotación son casi nulos, y además es inagotable. Los principales sistemas fotovoltaicos que se pueden instalar son los sistemas aislados, los conectados a red y los sistemas híbridos. Sin entrar en detalle de cada uno de ellos, vemos las diferencias generales que presentan cada uno de ellos. Información obtenida en el TFG referenciado en [35].

Sistemas fotovoltaicos aislados:

Son aquellos que precisan de sistemas de acumulación de energía producida. Durante el día produce energía suficiente para abastecer la demanda durante la tarde y la noche (ya que en estas horas no produce) gracias a la acumulación de energía en baterías. Por tanto, estos sistemas están formados por: módulos fotovoltaicos necesarios, un regulador de carga, un inversor de corriente alterna y un sistema de acumulación formado por baterías. Todo ello debidamente dimensionado para abastecer dicha demanda energética.

Sistemas fotovoltaicos conectados a la red:

Son aquellos que proporcionan toda o parte de su producción a una red de distribución. Están constituidos por: un generador fotovoltaico formado por módulos interconectados, contadores de producción y consumo para medir la energía producida por el sistema fotovoltaico y un inversor o convertidor de corriente continua-alterna para optimizar el paso de energía entre el módulo fotovoltaico y la carga. Estos sistemas son los habituales en centrales fotovoltaicas y en sistemas fotovoltaicos integrados en edificios.

Sistemas fotovoltaicos híbridos:

Son aquellos que utilizan la energía de los paneles solares combinados con otra fuente de energía (eólica, diésel, hidroeléctrica…). Estos sistemas cubren la demanda gracias al aprovechamiento de las fuentes disponibles. Si los consumos son elevados en un momento puntual, se aprovechará la energía suministrada por la segunda fuente, de esta forma se puede disponer de un número menor de paneles fotovoltaicos, optimizando con ello el precio total de la instalación.

Al no ser objeto de proyecto introducirse de lleno en el mundo de la energía solar fotovoltaica, lo que se propone es un pequeño estudio y dimensionamiento de una instalación fotovoltaica de autoconsumo (aislada) que dote al sistema del invernadero de una semana de autonomía.

Para ello se seguirán los siguientes pasos:

- Cálculo de los consumos individuales del sistema y del consumo total del mismo para un día de funcionamiento.
- Obtención de la radiación solar disponible de la localización donde se pretende instalar el conjunto fotovoltaico.
- Cálculo del número de paneles solares necesarios para la instalación.
- Diseño de la capacidad de los acumuladores o baterías según la autonomía deseada.
- Selección del regulador de carga y del convertidor de corriente.

En los siguientes puntos se detallan estos pasos a seguir, explicando a su vez algunos conceptos necesarios para la interpretación de los datos usados. Toda la información contenida en estos pasos se ha obtenido de dos fuentes: [36] y [37].

4.2.2. CONSUMO ESTIMADO

Este es, probablemente, el punto más importante para el dimensionamiento de la instalación solar fotovoltaica de autoconsumo (aislada de la red). Se debe dimensionar teniendo en mente dos objetivos, abastecer la demanda energética con garantías y reducir lo que se pueda el coste económico de la instalación.

Para calcular el consumo total diario de nuestro sistema hay que empezar por los consumos individuales que lo componen. El consumo total se mide en Watios hora (Wh). Se calcularán primero los consumos individuales en Watios, y como todos ellos estarán conectados todo el día, se multiplicará la suma de éstos por 24 horas. A continuación se resumen todos los

consumos individuales, sacados según la ecuación [3], en una tabla. Cabe destacar, que sólo se emplea un componente de cada tipo, si hubiese más de uno, habría que multiplicar el consumo de uno por en número de componentes del mismo tipo.

Consumo individual: $$\boldsymbol{P(W) = V \cdot I_{max}}$$

Ecuación 3 - Consumo individual

COMPONENTE	V – Tensión de funcionamiento (V)	Imax – Corriente máxima (mA)	Pi – Consumo individual (w)
ESP32-DevKitC	3.3 V	225 mA	0.7425 W
DHT22	5 V	2.5 mA	0.0125 W
LDR	5 V	0.6 mA	0.003 W
YL-69, YL-38	5 V	35 mA	0.175 W
Sensor PH	5 V	10 mA	0.05 W
Ultrasonidos HC-SR04	5 V	15 mA	0.075 W
Display LCD I2C	5 V	250 mA	1.25 W
Relés	5 V	40 mA	0.2 W
Ventilador	12 V	110 mA	1.32 W
Bomba de agua	12 V	400 mA	4.8 W

Tabla 8 - Consumos individuales de los componentes del sistema

El consumo total (Pt) será el sumatorio de los consumos individuales (Pi):

$$\boldsymbol{Pt} = \sum Pi = \boldsymbol{8.63\ W}$$

Ecuación 4 - Consumo o potencia total

Total consumo por día estimado (Cde):

$$\boldsymbol{Cde} = Pt \cdot 24\frac{h}{día} = 8.63(W) \cdot 24\left(\frac{h}{dia}\right) \cong \boldsymbol{207\ Wh/dia}$$

Ecuación 5 - Consumo total por día estimado

Si se aplica un rendimiento de instalación del 75%, obtenemos la energía total necesaria de la instalación (Ten):

$$\boldsymbol{Ten} = \frac{Cde}{\eta} = \frac{207\ (Wh/dia)}{0.75} \cong \boldsymbol{276\ Wh/dia}$$

Ecuación 6 - Energía total necesaria de la instalación

Para el caso de estudio se necesita una potencia fotovoltaica de al menos 276 W por hora y día para abastecer la demanda.

4.2.3. RADIACIÓN SOLAR DISPONIBLE

Se necesita saber la cantidad de radiación solar disponible en la zona de instalación del sistema fotovoltaico. Para ello existen aplicaciones con bases de datos sobre la radiación solar en distintos puntos del mundo y en todos los meses del año. Para el cálculo se emplea la aplicación **PVGIS** (Photovoltaic Geographical Information System – European Commission, Joint Research Center), la cual tiene una plataforma on-line en que se pueden obtener los datos de insolación para toda Europa de forma fácil y rápida [38].

Lo siguiente que debemos saber, es el tipo de placas solares de las que disponemos en el mercado. En función de las características de la placa escogida, se definirán los datos necesarios para el cálculo de radiación solar disponible. Se pueden encontrar placas de distintos materiales, distinguiendo dos grandes grupos:

- Células basadas en Silicio (monocristalino, policristalino o amorfo). Son las células fotovoltaicas más convencionales y las más usadas dentro del mercado por ser las más fiables, las que presentan mejores características en cuanto a rendimiento (en torno al 20% en laboratorio y en torno al 10-15% en producción).
- Células menos convencionales (Silicio amorfo Tandem, CdTe, CIS, etc), se obtienen porcentajes de rendimiento sensiblemente menores que las anteriores.

En cuanto a lo que se busca en este proyecto, no es necesario el uso de células muy prometedoras, ya que la demanda es muy pequeña y el precio podría aumentar considerablemente. Para el caso actual, se ha encontrado en una tienda de electrónica convencional, un fabricante que proporciona información sobre sus placas, y destaca por emplear células de Silicio policristalino, ofreciendo una amplia variedad de paneles de

distintas potencias de funcionamiento. Tal y como se puede comprobar en las especificaciones técnicas del documento [39].

Estas células trabajan a una potencia de pico de 1000 W/m2 según las especificaciones de radiación solar incidente. Se estimarán unas pérdidas del 25% en la instalación, esto no es más que un coeficiente seguridad para asegurar un funcionamiento adecuado.

Por otro lado, en el caso de estudio se propone colocar la instalación en Boo de Piélagos (Cantabria), por lo que se busca este punto en el mapa. Se establecen a continuación, la base de datos sobre la que se hará el cálculo, en este caso *Climate-SAF PVGIS,* ya que es la que posee las últimas actualizaciones. Se especifica la tecnología y potencia de las células que se emplearán y se establece un montaje de pie a 45º de inclinación, debido a la latitud del lugar de instalación (Siempre se establece Latitud = Angulo de inclinación de las placas solares), en el caso de Boo de Piélagos la latitud es de unos 43º pero se redondea a 45º por facilidades de instalación. A su vez, se supone que la placa estará orientada hacia el Sur para conseguir un mayor aprovechamiento de la luz solar incidente, por lo que pondremos un azimut a 0º.

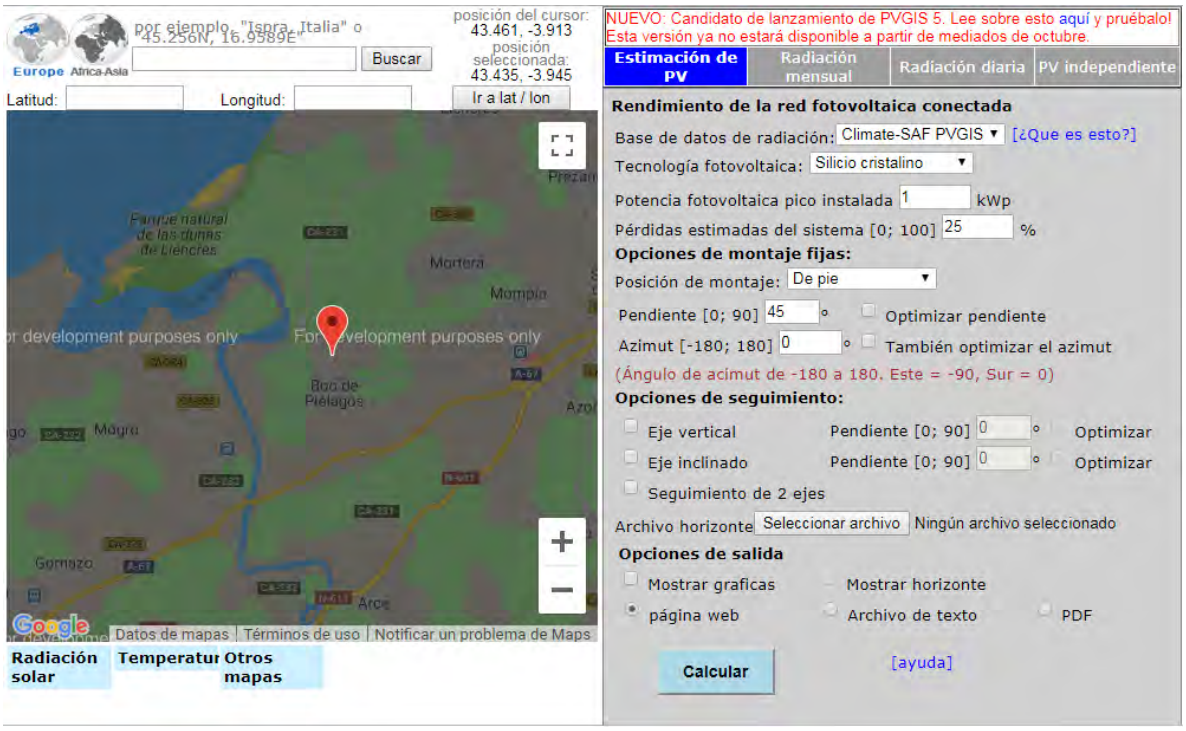


Ilustración 33 - Calculo de Radiación Solar en PVGIS.

Según la estimación se obtienen los siguientes resultados:

Estimaciones PVGIS de generación eléctrica solar.

Ubicación: 43 ° 26'5 "Norte, 3 ° 56'43" Oeste, Elevación: 23 m snm,

Base de datos de radiación solar utilizada: PVGIS-CMSAF

Potencia nominal del sistema fotovoltaico: 1,0 kW (silicio cristalino)

Pérdidas estimadas debido a la temperatura y baja irradiancia: 8,9% (utilizando la temperatura ambiente local)

Pérdida estimada debido a los efectos de reflectancia angular: 2,7%

Otras pérdidas (cables, inversor, etc.): 25,0%

Pérdidas del sistema fotovoltaico combinadas: 33,5%

Sistema fijo: inclinación = 45 °, orientación = 0 °				
Mes	E_d	E_m	H_d	H_m
ene	1.85	57.4	2,66	82.4
feb	2.50	69.9	3.62	101
mar	3,27	101	4.90	152
abr	3,26	97.7	4.93	148
Mayo	3.18	98.6	4.83	150
jun	3,21	96.3	4.93	148
jul	3,32	103	5.13	159
ago	3,29	102	5.09	158
sep	3,41	102	5.23	157
oct	2,70	83.7	4,09	127
nov	1.88	56.3	2,75	82.4
dic	1.86	57.8	2,70	83.7
Promedio anual	**2,81**	**85.5**	**4.24**	**129**
Total por año		**1030**		**1550**

Tabla 9 - Resultado de cálculo de Radiación Solar por PVGIS

Donde:

- **Ed:** Producción diaria promedio de electricidad a partir del sistema dado **(kWh)**
- **Em:** Producción mensual promedio de electricidad a partir del sistema dado **(kWh)**
- **Hd:** Suma diaria promedio de irradiación global por metro cuadrado recibida por los módulos del sistema dado **(kWh) / m2)**
- **Hm:** Suma promedio de la irradiación global por metro cuadrado recibida por los módulos del sistema dado **(kWh / m2)**

Los meses más desfavorables de radiación solar incidente son en invierno (Enero, Noviembre y Diciembre). Para dimensionar la instalación, es necesario hacerlo para las condiciones

mensuales más desfavorables para asegurar que se cubre la demanda energética durante todo el año. En este caso, se observa que Enero es el mes más desfavorable con 2,66 kWh·m2 por día (Hd).

A continuación, se calcula la cantidad de **horas sol pico (HSP)**, esto es equivalente al número de horas que tendría que incidir el sol sobre los paneles a una intensidad de 1kW/m2 (radiación solar incidente de las células fotovoltaicas de los paneles elegidos) para obtener la insolación total de un día.

$$HSP = \frac{radiación\ solar\ Enero}{radiación\ solar\ de\ pico\ de\ las\ células} = \frac{2{,}66\ \mathrm{kWh \cdot m2}}{1\ \mathrm{kWh \cdot m2}} = \mathbf{2{,}66\ HSP}$$

Ecuación 7 - Radiación Solar Disponible

Esta es la radiación solar disponible para las condiciones más desfavorables.

4.2.4. CALCULO DE PANELES SOLARES NECESARIOS

En función de las condiciones más desfavorables se calcula a continuación el número de módulos necesarios para la instalación. Para ello es necesario definir el rendimiento de trabajo, para lo que se tendrá en cuenta que las placas estarán colocadas en el exterior del invernadero y por ello se prevén deterioros por causas diversas, como ensuciamiento, deterioro por oxidación o clima, etc. Estas pérdidas son estimativas, y dependerá del criterio del instalador. Se han considerado unas pérdidas del 25%, por lo que el rendimiento de trabajo se establece en el 75%.

Otra cosa a tener en cuenta, es la potencia de pico de los módulos elegidos. En las características técnicas del modelo y fabricante seleccionado, se detallan diferentes placas con sus respectivas potencias de pico. Se han escogido módulos de 80 W para el cálculo.

Como se pretende diseñar una instalación para uso diario, el cálculo del número de módulos se hará mediante la siguiente fórmula:

$$Nº\ modulos = \frac{Ten}{(HSP \cdot \eta \cdot Ppm)} = \frac{276\ Wh/dia}{2{,}66\ h/dia \cdot 0.75 \cdot 80\ W} = 1{,}73 \sim \mathbf{2\ módulos}$$

Ecuación 8 - Cálculo del número de módulos fotovoltaicos necesarios

Dónde:

- **Ten:** Energía total necesaria
- **HSP:** Horas sol pico
- **η:** Rendimiento de trabajo
- **Ppm:** Potencia de pico del módulo

Se redondea a 2 módulos, por lo que se obtiene una instalación solar fotovoltaica de 160 W de pico totales (2 módulos de 80W de pico). Estos módulos trabajan a 12 V. El voltaje de funcionamiento dependerá del sistema de acumuladores que se elija.

4.2.5. CAPACIDAD DE LOS ACUMULADORES

Lo siguiente será diseñar la capacidad de las baterías o acumuladores, para lo que se necesita establecer la autonomía deseada en caso de tener días nublados o de baja insolación. De manera general, la decisión se toma a criterio del instalador, según la previsión de días sin sol que pueda haber en una determinada zona, en Cantabria se puede prever 4 o 5 días nublados en invierno si no son más. Sin embargo, como ya se detalló al principio, se pretende diseñar el sistema de alimentación para una semana de autonomía (7 días naturales).

Se define la capacidad de la batería por la siguiente ecuación:

$$Capacidad\ de\ acumulación = \frac{Energía\ necesaria\ (Ten) \cdot Días\ de\ autonomía}{Voltaje\ bateria \cdot Profundidad\ de\ descarga\ batería}$$

Ecuación 9 - Capacidad de acumulación

Ahora bien, es necesario elegir una batería, ya que la profundidad de descarga depende de las características técnicas según el modelo y fabricante que se escoja. Se ha encontrado un modelo (Batería AGM Blackbull 12V 250Ah) que tolera una capacidad de descarga de hasta un 60%.

$$\boldsymbol{Capacidad\ de\ la\ batería} = \frac{276\,\frac{Wh}{dia} \cdot 7dias}{12\ V \cdot 0.6} = \boldsymbol{268,3\ Ah\ (c100)}$$

Ecuación 10 - Capacidad de la batería

El valor c100 indica que la capacidad de la batería será la suministrada por ciclos de carga de 100 h. Se requiere una mínima intensidad de carga para asegurar que la batería se cargue correctamente y alargar su vida útil.

4.2.6. SELECCIÓN DEL REGULADOR

Finalmente, queda elegir un regulador de carga. Los reguladores se determinan por la intensidad máxima de trabajo y el voltaje en que se ha diseñado la instalación. En este caso, la intensidad máxima de trabajo será:

$$\boldsymbol{Imáx} = \frac{P\ (Consumo\ total\ componentes)}{V\ (Tension\ trabajo)} = \frac{8.63(W)}{12\ (V)} = \mathbf{0.72}\ \boldsymbol{A}$$

Ecuación 11 - Intensidad máxima de trabajo de la instalación

Sólo se necesitan 0,7 A de continua, ya que en este caso no se tiene ningún componente que consuma un pico de corriente de arranque elevado, es decir, el consumo es continuo. Señalar que en este proyecto no hace falta pasar de corriente alterna a continua, por lo que no será necesario un convertidor de corriente.

4.3. VISIÓN GLOBAL DEL SISTEMA

A continuación se recopila a modo resumen todo el hardware empleado para el prototipo del proyecto diseñado.

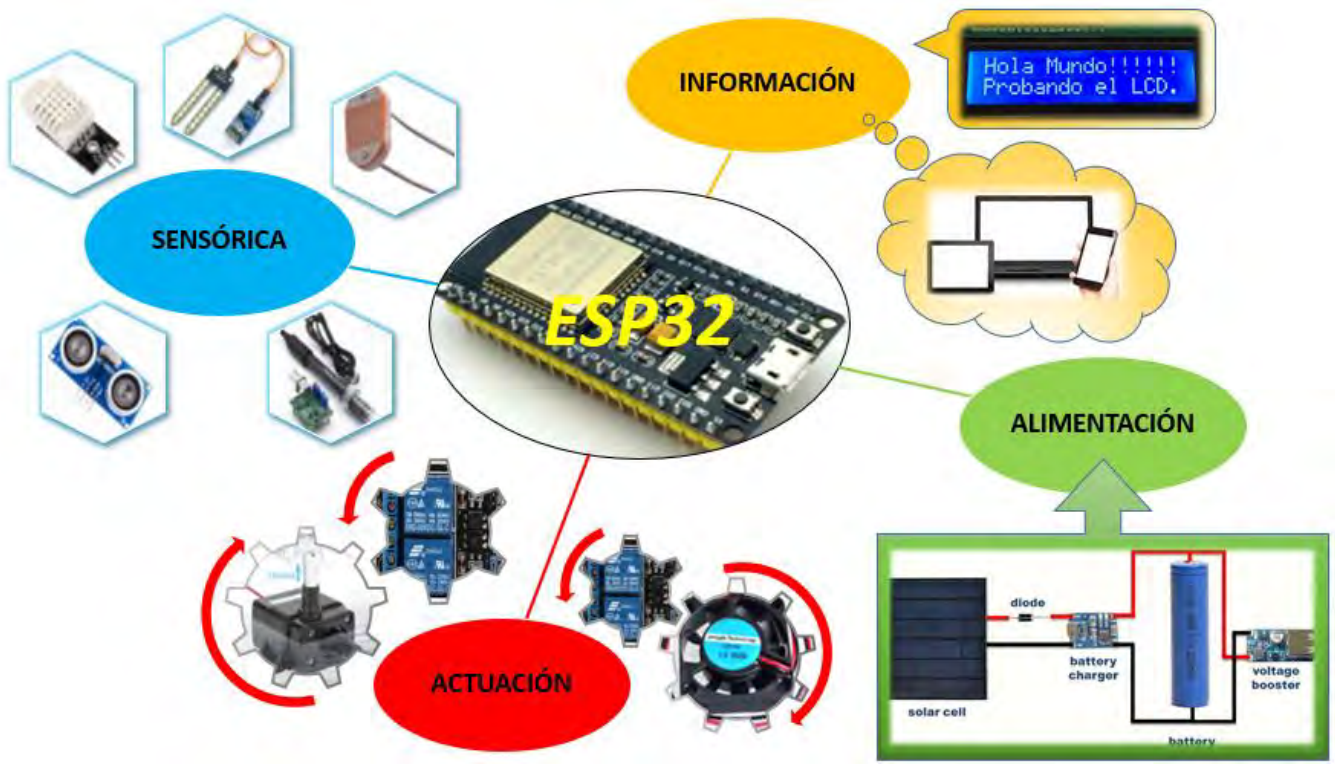


Ilustración 34 - Visión global del sistema.

- **ESP32:** Como controlador principal del sistema
- **Sensórica:** Uso de 5 sensores para la lectura de parámetros del sistema (DHT22, YL69, LDR, Ultrasonidos y Sensor de PH)
- **Sistema de actuación:** Gracias a un módulo de dos relés accedemos a los actuadores (bomba de agua y ventilador)
- **Sistema de alimentación:** Compuesto por dos placas solares, un acumulador y un regulador de descarga.
- **Información del sistema:** El usuario podrá acceder a la información recogida localmente mediante el Display LCD I2C y remotamente mediante un ordenador, portátil, móvil o Tablet.

5. CONECTIVIDAD

5.1. CONECTIVIDAD WiFi

Antes de comenzar a enviar o recibir datos por WiFi es imprescindible conocer el módulo y sus protocolos o vías de comunicación con Internet. Es una de las partes más complejas y que personalmente me ha generado quebraderos de cabeza. Es complicado entender un protocolo de comunicación si no estás familiarizado con él o con las bases de Internet, lo que resulta una parte dura y tediosa de investigación y desarrollo dentro del proyecto.

Es importante valorar y contrastar bien la información de Internet y/o libros, ya que una búsqueda concreta puede derivar en diversas respuestas y no todas pueden responder nuestras dudas de una manera clara, concisa y correcta.

5.1.1. COMUNICACIONES POR INTERNET

En éste apartado se resumen las bases y conceptos de comunicación de Internet. Además para un mayor entendimiento de estos conceptos, se recomienda leer el Libro “Internet de las Cosas” [1], donde de forma cercana y con multitud de ejemplos, dan a conocer la tecnología IoT.

El paquete de protocolos TCP / IP:

Es una combinación de los protocolos TCP (Transmission Control Protocol) e IP (Internet Protocol).

En el **protocolo IP** la información se envía de una máquina a otra dentro de un paquete de datos, donde se incluyen las direcciones del destinatario y del remitente, utilizando un formato estándar (un protocolo). La máquina que envía el mensaje (enrutador) no siempre conoce cuál es la mejor ruta para llegar al destino, ya que la mayoría de las veces los paquetes viajan a través de una red inmensa de enrutadores que no siempre es la misma. Lo malo de éste protocolo, es que no ofrece garantías de que el destinatario reciba la información y además solo permite enviar los datos que quepan en un paquete.

El **protocolo TCP** es el protocolo de transporte más básico de Internet y se ha construido en una capa por encima del protocolo IP. TCP aporta una secuencia de números, acuses de recibo y retransmisiones. Permite enviar mensajes de cualquier longitud y ofrece cierta garantía de que los paquetes de datos lleguen intactos a su destino.

La combinación de ambos en uno resulta muy útil, y por ello se define el paquete de protocolos TCP/IP como un apilamiento de protocolos donde cada uno aprovecha las propiedades de los protocolos que se encuentran en capas inferiores. A continuación se muestra un esquema aclaratorio sobre el paquete de protocolos y las capas que lo componen:

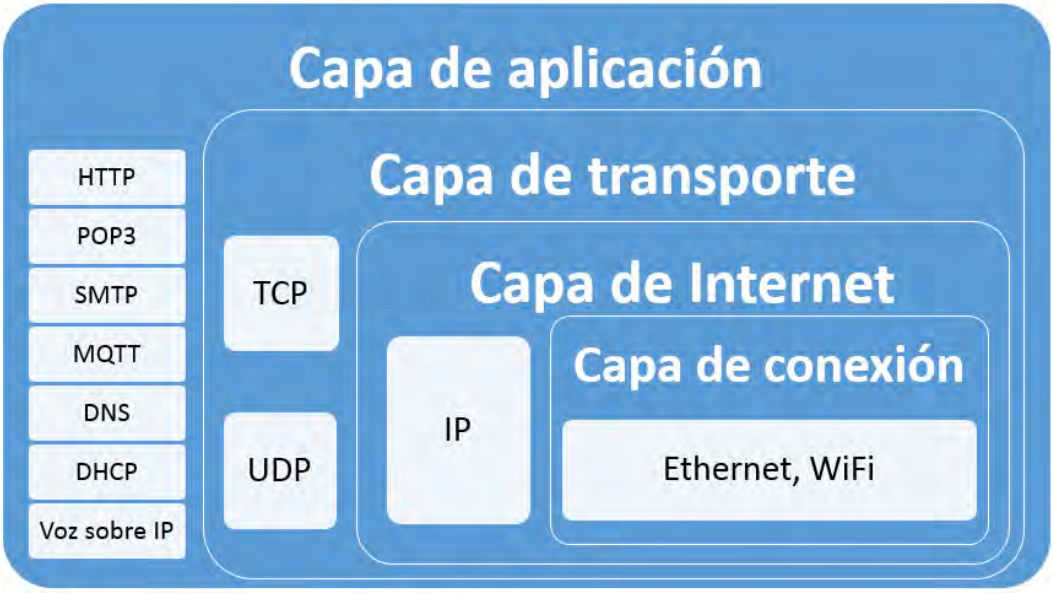


Ilustración 35 - Paquete de protocolos y capas que lo componen.

Las capas más internas representan los protocolos de bajo nivel, mientras que la más externa representa un nivel de abstracción más alto.

Los protocolos de bajo nivel, los que se encuentran en la capa de conexión, gestionan la transferencia de bits de información a través de la red de trabajo. Esta red podría ser un cable Ethernet o Wi-Fi, una conexión que se establezca a través de un teléfono o incluso conexiones que utilicen estándares de radio de onda corta.

La capa de Internet se encarga de simplificar todos los detalles relacionados con las conexiones y el intercambio de información, se pueden emplear direcciones sencillas para dirigirse al destinatario.

Encima de ésta capa se encuentra la capa de transporte y en ella el protocolo TCP, gracias al cual se consigue un control más sofisticado sobre el envío de mensajes.

Por último está la capa de aplicación, la cual contiene los protocolos que se utilizan para trabajar con páginas Web, enviar correos electrónicos y trabajar con la telefonía sobre Internet. El protocolo más utilizado en la Web es el HTTP, ideal para emplear en el IoT. Sin

embargo, en éste proyecto se emplea el protocolo MQTT del que entraremos en detalle en los siguientes apartados.

Cabe destacar, que el protocolo TCP no es el único en la capa de transporte. El protocolo UDP se descarta la mayoría de las veces por no garantizar que el mensaje llegue al destinatario (al igual que ocurría con IP), sin embargo, lo hace atractivo y útil en aplicaciones como el streaming de datos, donde no puede haber retrasos en la trasmisión y se puede prescindir de la pérdida de algunos datos. Un ejemplo de aplicación es el servicio Skype, un sistema de teléfono basado en el ordenador.

5.1.2. CONEXIÓN MODULO WiFi

Una vez visto del módulo WiFi que se utilizará en el proyecto, queda concretar cómo se efectuará su conexión a Internet. Lo primero es saber en qué entorno de programación se va a trabajar, al igual que el ESP8266, el ESP32 puede programarse desde diferentes entornos como el ESPlorer, ESP Flash Donwload Tool y Arduino IDE. El entorno de programación de este proyecto será el IDE de Arduino, por ser un entorno abierto e intuitivo.

Ahora bien, el objetivo de este punto es explicar cómo conectar el ESP32 a una red WiFi. Lo primero será instalar la última versión de Arduino IDE, ya que si la versión es anterior a la 1.8 no funcionará.

Por otro lado, para poder usar cualquier módulo basado en ESP32 en el emulador de Arduino, es necesario seguir una serie de pasos para su instalación. Este proceso es algo más pesado que el de instalación para ESP8266, ya que para éste último se usa el gestor de tarjetas de Arduino y solo es necesario especificar una dirección web que descargue directamente los controladores del mismo. El desarrollo del entorno ESP32 no ha alcanzado ese punto de madurez, por lo que para su instalación es necesario usar Phyton 2.7 y el entorno gráfico de GIT GUI. No se entra en detalles de instalación, ya que se trata de un trámite del que ya hay información de cómo hacerlo paso a paso, por ejemplo en la página web referenciada como [40].

Una vez que se puede acceder a la placa de desarrollo en Arduino, (en Herramientas escogemos la placa: "*ESP Dev Module*") se procede a instalar la librería necesaria para poder conectar el ESP32 a una red WiFi. Necesitamos incluir la librería *"WiFi.h"* (para ESP8266 será "*ESP8266WiFi.h*"). En el menú del IDE seleccionar: *Programa* > *Incluir librería* > *Gestionar librerías*. Y en el buscador escribir *WiFi*. En la imagen se muestra la librería ya instalada. Este paso se hará siempre que se quiera instalar una nueva librería para su uso.

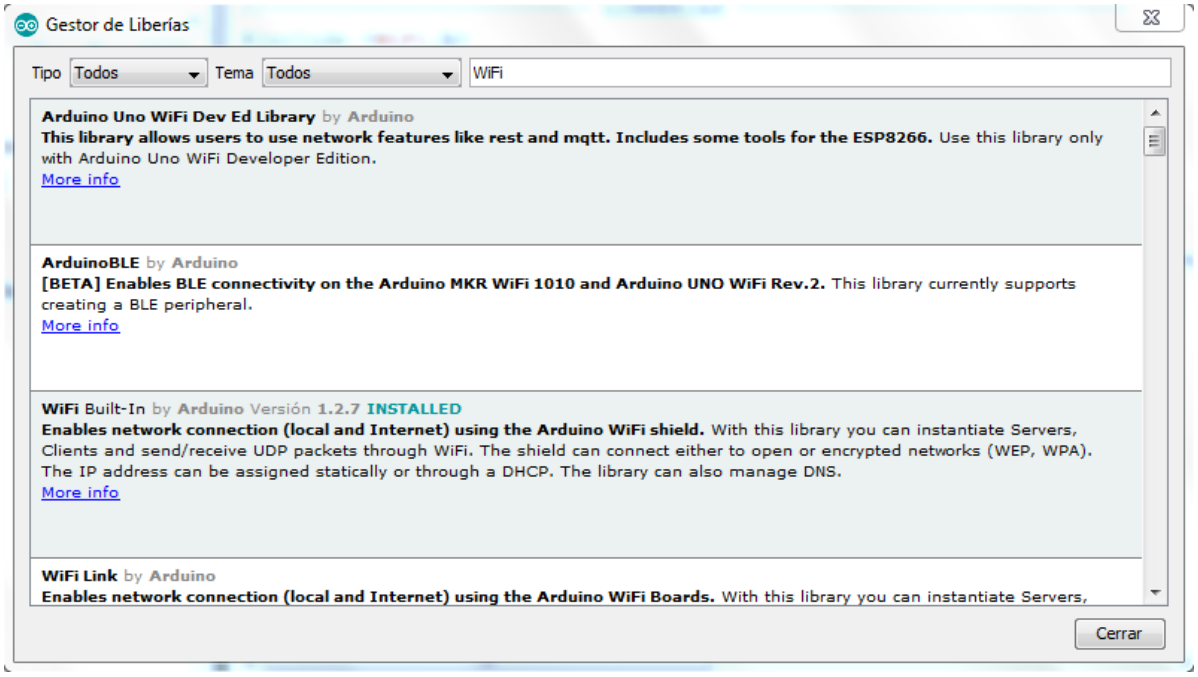


Ilustración 36 - Gestor de Librerías IDE de Arduino.

Para editar cómodamente las credenciales de la red WiFi que se usará, se declaran dos variables globales, ssid y password. Entre comillas, escribiremos el nombre (ssid) y la contraseña (password) de la red WiFi a la que queramos que se conecte el ESP32.

```
// Credenciales WiFi:
const char *ssid = "---";
const char *password = "---";
```

Ilustración 37 - Credenciales WiFi en el código.

Una vez hecho esto, lo siguiente será en la función Setup, escribir unas líneas de código para la conexión del módulo a la red WiFi. Lo hacemos en setup para que la conexión la haga directamente en la inicialización del código, es decir, nada más encender el módulo.

Cabe destacar, que si no se conecta en menos de 25 segundos, lo que se considera un tiempo más que suficiente, mandará un mensaje de error y abortará el intento de conexión.

Por otro lado, aunque ya se conoce en líneas generales qué es el protocolo IP, es necesario hacer un inciso para conocer qué son las direcciones IP, ya que se usará una para la conexxón del dispositivo a la red.

Las **direcciones IP** son números, separados en cuatro grupos numéricos de 8 bits por puntos (de 0.0.0.0 a 255.255.255.255). Un ejemplo es la dirección generalmente asignada al router de casa: 192.168.0.1, otro ejemplo es la dirección 8.8.8.8 de uno de los servidores DNS de Google (**DNS** o *Domain Name System* es, como su nombre indica, un Sistema para el nombre del dominio, donde asignan nombres o dominios, como "***.es***" o "***.com***" a algunas direcciones IP conocidas que los humanos podemos recordar con mayor claridad). Este grupo de cuatro números separados por puntos equivale a un número de 32 bits, y todo dispositivo o aparato conectado a internet tiene su propia dirección IP asignada. Esta puede ser estática o dinámica.[41]

Una **dirección IP dinámica o local** es una dirección IP que el ISP (Proveedor de Servicios de Internet) asigna dinámicamente al dispositivo que pretende tener acceso a Internet. Cada vez que el dispositivo se reinicia, el ISP vuelve a asignar dinámicamente una dirección IP al dispositivo.

Una **dirección IP estática o fija**, es una dirección que se asigna de forma permanente. Esa dirección no cambia aunque se reinicie el router. Estas direcciones IP estáticas suelen asignarse a empresas que compran un paquete servidor-*hosting* a un ISP para asignar a sus clientes de páginas web. También se asignan direcciones IP estáticas para la prestación de servicios de correo electrónico, de bases de datos y para servidores FTP (Protocol Transfer File, protocolo para transferencia de archivos entre sistemas conectados a una red basada en arquitectura cliente-servidor).

Señalar que en el presente proyecto se usa una IP dinámica y no una estática. Si se quisiera usar una IP fija y se conocen los datos de la misma, podríamos descomentar las líneas de código correspondientes, modificar dichos datos y comentar la línea usada para la asignación de una IP dinámica. Para mayor entendimiento, revisar la parte del código correspondiente incluido en el Anexo [6].

5.2. SERVIDOR WEB

5.2.1. PROTOCOLO DE APLICACIÓN MQTT

Una vez visto el esquema general de las distintas capas de abstracción del paquete de protocolos TCP/IP, se puede seleccionar un protocolo de la capa de aplicación para su desarrollo, aunque esto no significa que escojamos la opción más correcta, ya que cada proyecto se puede resolver de diversas formas.

Probablemente el protocolo más empleado para aplicaciones IoT, es HTTP, sin embargo en el presente proyecto de usará el protocolo MQTT del que hablaremos a continuación. Añadir que se escogió este protocolo por dos razones, en el momento de empezar, había bastantes referencias e información al respecto, y por otro lado, se pretendía usar otro entorno de desarrollo que no fuese el más habitual, como lo pretendía ser HTTP. Cabe destacar que actualmente, MQTT está a la altura de HTTP o CoAP.

MQTT son las siglas de Message Queue Telemetry Transport. Se trata de un protocolo de mensajería de publicación/suscripción simple, inventado y desarrollado por el Dr. Andy Stanford-Clark de la industria IBM y Arlen Nipper de Arcom (actual Eurotech) en 1999, y enfocado a la conectividad M2M (Machine to Machine). [42]

El protocolo MQTT se enfoca al envío de datos en aplicaciones donde se requiere muy poco ancho de banda y gracias a sus características, presume de bajo consumo y uso de pocos recursos para su funcionamiento. Es por esto, que se ha convertido en un protocolo muy empleado en la comunicación de sensores y, por tanto, dentro del Internet de las Cosas. Cabe destacar, que las aplicaciones que usan MQTT son lentas, por lo que el tiempo real se mide en segundos.[43]

Para los dispositivos IoT es importante la bidireccionalidad, es decir, la capacidad no sólo de publicar datos en un servidor, sino también de comunicarse con el dispositivo. Este proceso de publicación/subscripción se hace de forma periódica, es decir, el servidor permanecerá activo constantemente a la espera de acciones por parte de los dispositivos conectados.

La arquitectura de MQTT sigue una topología en estrella, con un nodo central que hace de servidor o ”*broker*” y el cual gestiona la red y transmite los mensajes. Hay dos tipos de clientes: los que publican datos (publisher) y los que consumen datos (suscribers). Estos clientes publican o suscriben datos en *“topics”* (también llamados *“feeds”* o simplemente temas). Es decir, existe una plataforma o servidor web (broker) donde diferentes dispositivos o clientes (por ejemplo un móvil, un ordenador, o cualquier otro dispositivo con conexión

WiFi), publican o subscriben mensajes o datos en feeds o temas (como temperatura, hora, lámpara…etc). En este tipo de arquitecturas la comunicación puede ser de uno a uno o de uno a muchos. Un topic o tema se representa mediante una estructura jerárquica en cadena separando cada elemento con una barra "/": **mqtt://broker/topic/message.**

Para afianzar este concepto se pone el siguiente ejemplo:

En un hotel se colocan sensores de temperatura y de movimiento para tener control automático sobre la calefacción y luminosidad. Suponiendo que un hombre entra en el hotel y sube hasta la tercera planta donde se encuentra su habitación, la 305 en el pasillo derecho, camina hacia la misma y el sensor de movimiento detecta su presencia y se enciende la luz del pasillo y cuando entra en su habitación se activa la calefacción y las luces de la misma. Cada habitación del hotel puede gestionar calefacción y luces por un mismo dispositivo, por ejemplo un Arduino. Ahora bien, la subscripción/publicación de información (del ejemplo descrito) se generará por los siguientes topics:

- Para la activación de la luz del pasillo:
  - **hotel/planta3/pasilloderecho/movimiento**
    (Subscripción leer sensor movimiento)
  - **hotel/planta3/pasilloderecho/luz**
    (Publicación de un mensaje puede ser simplemente: **"ON"**). El servidor puede saber si la luz ya está encendida o no suscribiéndose también al tema.
- Para la activación de la calefacción y de las luces de la habitación 305:
  - **hotel/planta3/pasilloderecho/habitacion5/temperatura**
    (Subscripción leer temperatura)
  - **hotel/planta3/pasilloderecho/habitacion5/calefaccion**
    (Publicación mensaje: "ON"). El servidor puede saber si la calefacción ya está encendida o no suscribiéndose también al tema.
  - **hotel/planta3/pasilloderecho/habitacion5/sensor1**
    (Subscripción leer sensor1 de movimiento)
  - **hotel/planta3/pasilloderecho/habitacion5/luzEntrada**
    (Publicación mensaje: "ON"). El servidor puede saber si la luz ya está encendida o no suscribiéndose también al tema.

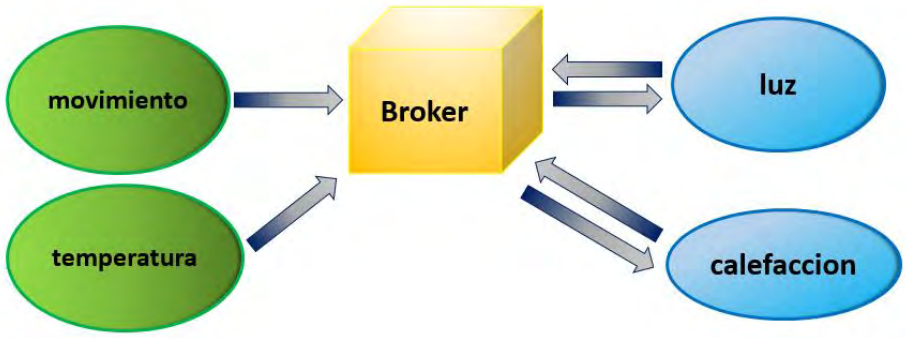


Ilustración 38 - Ejemplo publicación/subscripción a un topic.

El esquema jerárquico del hotel sería algo así:

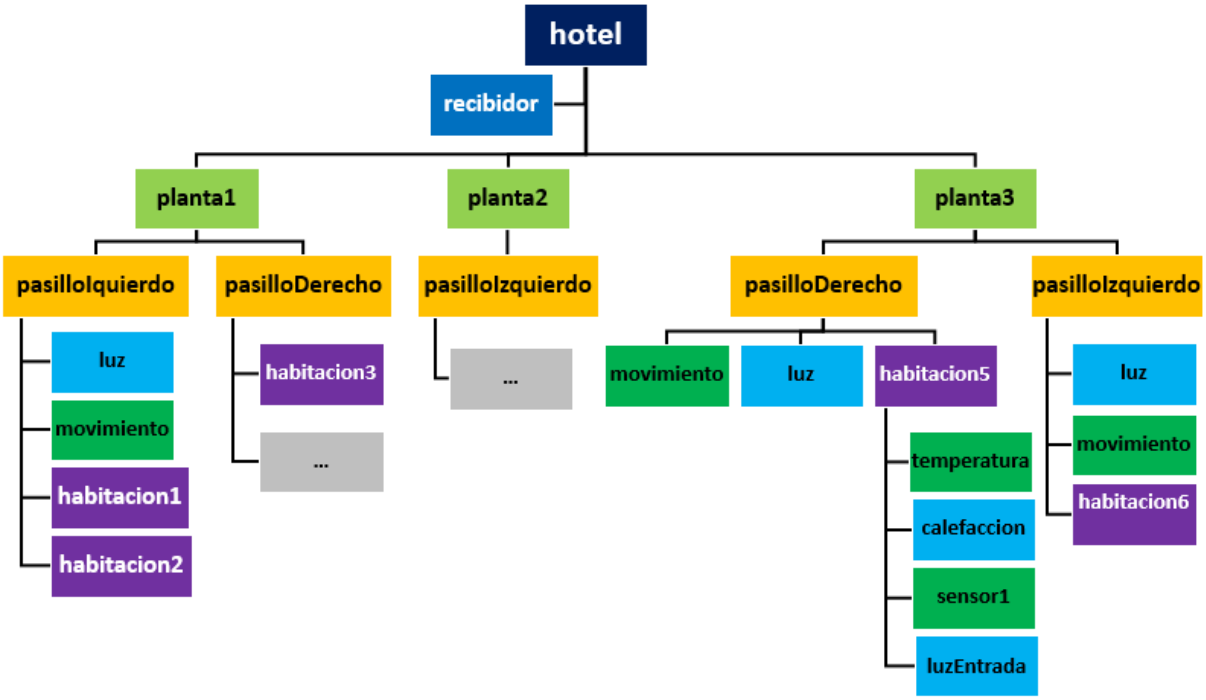


Ilustración 39 - Esquema ejemplo jerarquía MQTT.

Otro ejemplo obtenido de la página web [42], donde también se explica el protocolo MQTT, es el siguiente:

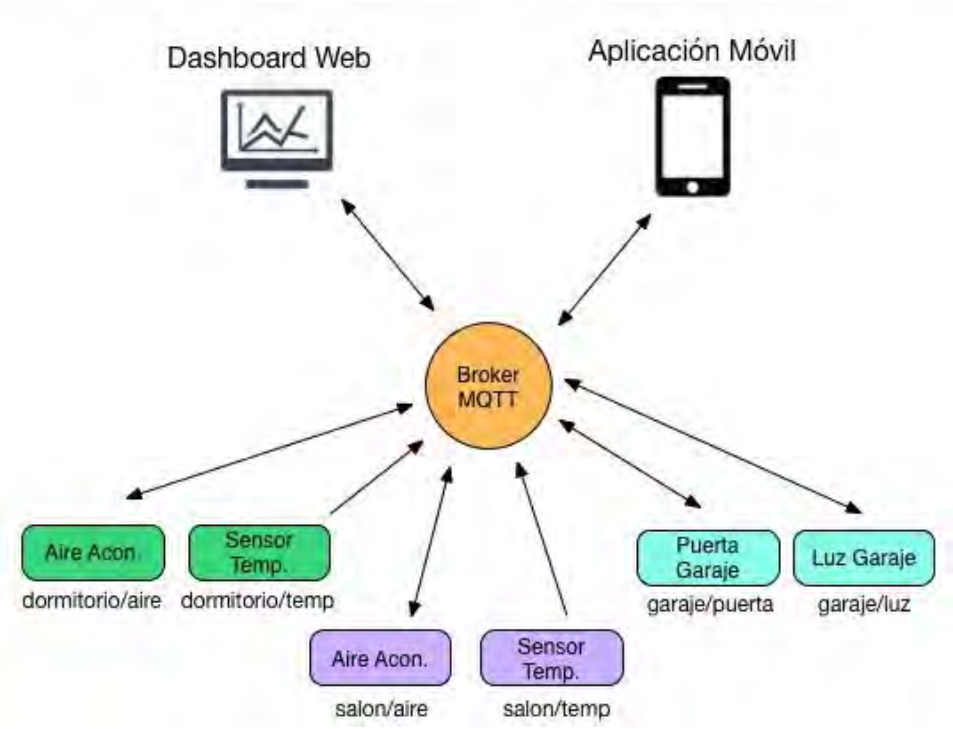


Ilustración 40 - Ejemplo protocolo MQTT.

5.2.2. COMUNICACIÓN MQTT CON SERVIDOR

Una vez elegido el protocolo de comunicación, debemos buscar un servidor Broker que gestione los datos de esos topics [44], [45]. En éste caso, se puede optar por tres tipos:

- Utilizar un servidor propio instalado localmente: Hay corredores/servidores gratuitos y de código abierto pero con su propio hardware y/o software, como ***Mosquitto*** que usa instrucciones Windous y Linux, ***Mosca*** que usa Node.js el cual es necesario instalar, ***Python Test Broker*** (broker de prueba de Python), etc.
- Utilizar un servidor virtual basado en la nube (Online Cloud Base MQTT Broker): usado normalmente para alojar sitios web y aplicaciones web. Les hay con paquetes gratuitos y de pago. Encontramos entre otros: ***test.mosquito.org*** **e** ***iot.eclipse.org*** de Mosquitto, ***test.mosca.io*** de Mosca, ***cloudmqtt*** (se establece un host cuando se crea la primera instancia), ***Adafruit.io***, ***Thingspeak***, etc.
- Utilizar una aplicación de servidor compartido: ***HiveMQ*** es un ejemplo de este tipo de servidores, el gran inconveniente que supone es que cualquiera puede acceder (suscribirse/publicar datos) a todos los temas.

El servidor broker en el que se basa el proyecto es el que proporciona Adafruit IO. Esta plataforma es sencilla y funcional, con una interfaz visual más atractiva que la que ofrecen otros servidores, y totalmente compatible con Arduino. Cabe señalar, que su paquete gratuito es de los más completos (contrastando con otros servidores gratuitos basados en la nube), sin embargo está limitado respecto al número de tableros (5 máximo) y feeds (10 máximo), publicación de datos en tiempo real (30 datos por minuto) y un máximo de 30 días de almacenamiento de datos, lo que limita bastante el proyecto a desarrollar. En el proyecto por dichas limitaciones se establecen prioridades respecto a los temas (feeds) implantados y el tiempo de publicación/subscripción de datos en los mismos.

Paquete gratuito de Adafruit IO:

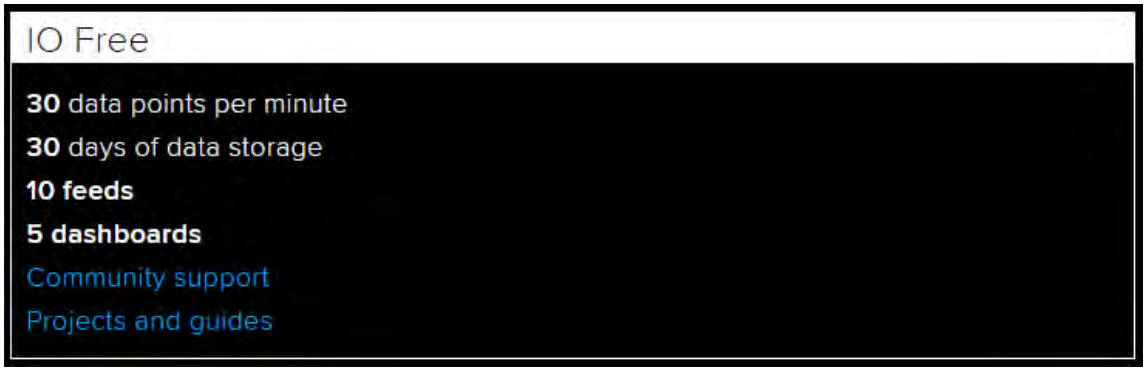


Ilustración 41 - Paquete de uso gratuito del Servidor Adafruit IO.

En el IDE de Arduino hay dos bibliotecas diferentes que se pueden usar para acceder a Adafruit IO. Una biblioteca está basada en el protocolo REST API, y la otra biblioteca se basa en la API MQTT. La diferencia entre estas bibliotecas es que MQTT mantiene una conexión al servicio abierta para que pueda responder rápidamente a los cambios en el feed. La API REST solo se conecta al servicio cuando se realiza una solicitud, por lo que es una opción más adecuada para proyectos que se suspenden durante un período de tiempo (para reducir la potencia) y despertar solo para enviar/recibir datos. En el presente proyecto emplearemos la biblioteca basada en MQTT. Estas bibliotecas se pueden descargar de un repositorio de GitHub.

```
#include <WiFiClient.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
```

Ilustración 42 - Librerias MQTT AIO en el código.

Para conectar un cliente (en este caso el módulo basado en ESP32) es necesario conocer:

- Server: io.adafrit.com
- Server Port: 1883 o 8883 (para conexión cifrada SSL)
- Username: el nombre de usuario de la cuenta de Adafruit
- Key: clave AIO (Adafruit.io)

Nos registramos en AIO (adafruit.io), buscamos en el menú "*View AIO Key*" y apuntamos nombre de usuario clave AIO Key para la configuración en Arduino.

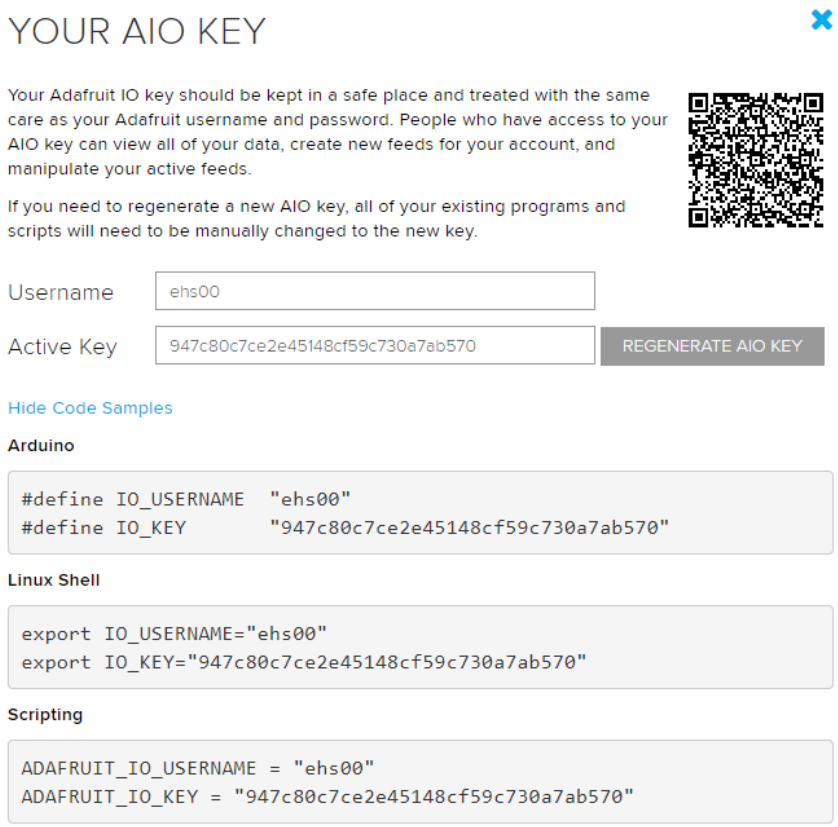


Ilustración 43 - Obtención de clave AIO.

```
// Credenciales Adafruit
#define AIO_SERVER      "io.adafruit.com" // 52.70.203.194
#define AIO_SERVERPORT  1883              // use 8883 for SSL
#define AIO_USERNAME    "ehs00"
#define AIO_KEY         "947c80c7ce2e45148cf59c730a7ab570"
```

Ilustración 44 - Credenciales AIO en el código.

A continuación se crea una clase para el cliente ESP32 llamada *"client"* para conectar con el servidor a través de MQTT. Y se configura la clase del cliente MQTT de Adafruit.io pasando el cliente WiFi (client) y el servidor MQTT y los detalles de inicio de sesión.

>> WiFiClient client;

>> Adafruit_MQTT_Client mqtt (&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_USERNAME, AIO_KEY);

Finalmente para establecer la conexión se crea una función que asegure que se reconecte al servidor siempre que sea necesario. Esto se hará con un bucle, en el que se comparará un estado, si la conexión con el servidor MQTT no se ha realizado en 10 segundos, lo volverá a intentar hasta que se conecte. Esta función se verá en detalle en el código correspondiente del apartado de Software.

Si se precisa de más información, se recomienda acceder a las fuentes citadas como: [46], [47], [48].

5.2.3. CREACIÓN DE UN DASHBOARD Y FEEDS

Creación del **DashBoard** o tablero es muy sencilla**,** una vez registrados en Adafruit.io vamos a la pestaña Dashboards del menú (imagen 43), y creamos uno nuevo con el nombre del tablero a utilizar. En este caso será MInvernadero (imagen 44).

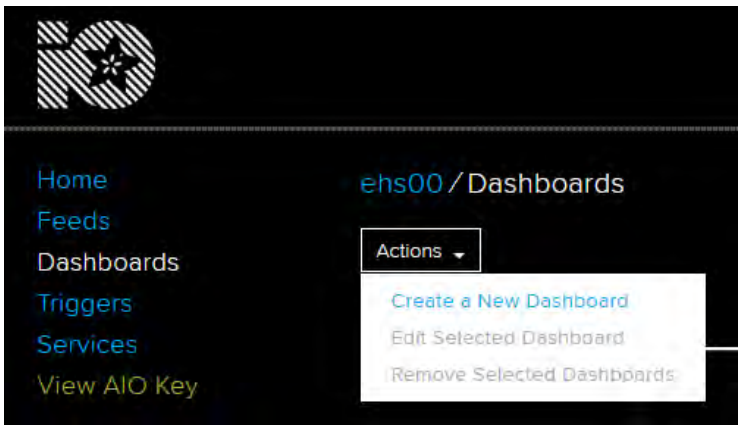


Ilustración 45 - Crear un tablero o Dashboard.

Ilustración 46 - Tablero MInvernadero AIO.

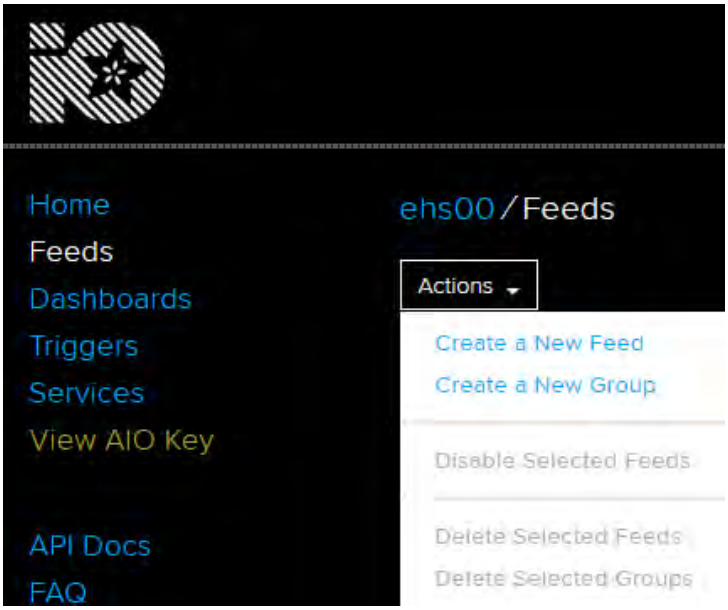


Ilustración 47 - Crear feeds o temas.

Seguidamente pasamos a crear los feeds de forma muy similar (imagen 45). Los feeds o topics que se usarán son los 10 siguientes:

- Para proporcionar información al usuario sobre las medidas realizadas en el invernadero:
  - **humedadAmbiente** – Feed de publicación del porcentaje de humedad relativa ambiente del invernadero
  - **humedadTierra** – Feed de publicación del estado de humedad del suelo: seco, humedo, ideal
  - **luminosidad** – Feed de publicación para mostrar estado de luminosidad: media, alta, baja
  - **nivelAgua** – Feed de publicación del nivel de agua de depósito de suministro en %
  - **pHAgua** – Feed de publicación del pH del agua de riego
  - **temperatura** – Feed de publicación de la temperatura ambiente en ºC del invernadero
- Para mostrar al usuario final la información sobre las alertas generadas con fecha y hora de publicación:
  - **alarmas** – Feed de publicación de alarmas
- Para la configuración remota del sistema, donde el usuario final puede intervenir en la actuación del invernadero autónoma y manualmente:
  - **modoAuto** – Feed de subscripción del botón de selección de modo: A (automático) / M (manual)

- **riego** – Feed de subscripción del botón de activación de riego: ON / OFF
- **ventilación** – Feed de subscripción del botón de activación de ventilación: ON / OFF

En la siguiente captura de imagen se muestran estos feeds o topics junto con el último valor registrado y los días de almacenamiento de datos grabados.

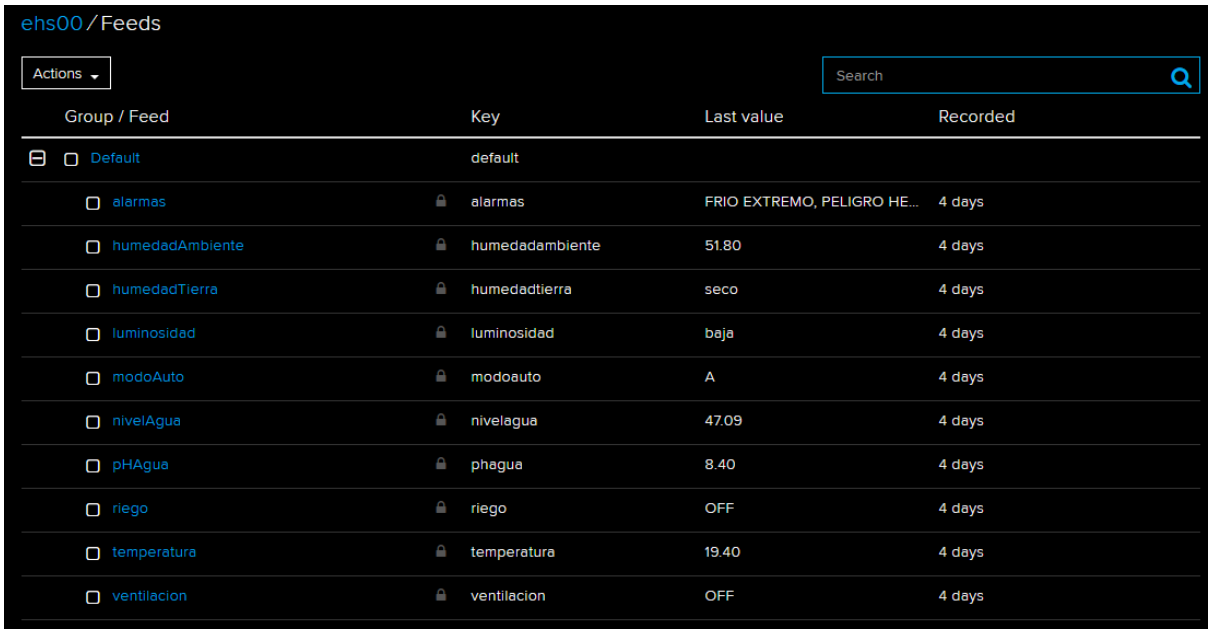


Ilustración 48 - Feeds creados en MInvernadero de AIO.

Cabe destacar, que no es necesario añadir los feeds desde Adafrit IO sino que podemos hacerlo desde el código, y una vez que se conecta el módulo al servidor aparecen los temas generados. Por otro lado, en el código debemos implementar estos topics según sean de publicación o suscripción. Se puede publicar un nuevo valor en un tema o puede suscribirse para recibir una notificación cuando el feed tenga un nuevo valor. Para nombrar y acceder a un tema concreto, se debe escribir la siguiente ruta: ***(nombre de usuario) / feeds / (nombre o clave del feed)*** o bien: ***(nombre de usuario) / f / (nombre o clave del feed).*** Donde ***(nombre de usuario)*** es el nombre de usuario de Adafruit IO (el mismo que se especifica al conectarse al servidor MQTT) y ***(nombre o clave del feed)*** es el nombre o la clave del tema al que queremos acceder.

Ejemplo de feed de publicación:

>> Adafruit_MQTT_Publish temperatura = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/temperatura");

Se crea una variable accesible por código para su modificación de tipo *Publish*, en el que se define como dato mqtt y el nombre del feed de publicación según la ruta especifica.

Ejemplo de feed de subscripción:

>> Adafruit_MQTT_Subscribe modoAuto = Adafruit_MQTT_Subscribe (&mqtt, AIO_USERNAME "/feeds/modoAuto", MQTT_QOS_1);

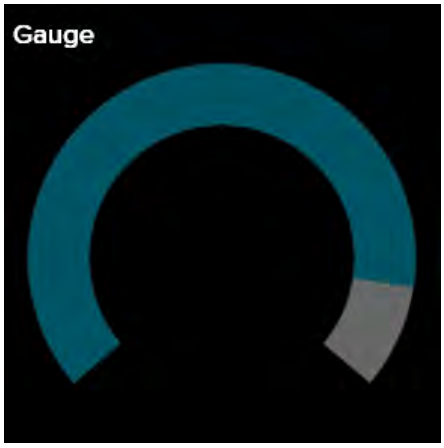
Se crea una variable accesible por código para su modificación de tipo *Subscribe*, en el que se define como dato mqtt y el nombre del feed de subscripción según la ruta especifica. Se incluye la especificación *MQTT_QOS_1*, esto se refiere al nivel de QoS (Calidad del servicio), al publicar datos de feeds. Esto permite confirmar que los datos se han publicado correctamente, a través de tres niveles distintos:

- QoS nivel 0 – Como máximo una vez: el mensaje se envía solo una vez y no se toman medidas para confirmar la entrega (activar y olvidar)
- QoS nivel 1 – Al menos una vez: el mensaje es reintentado por el remitente varias veces hasta que se recibe el acuse de recibo (entrega confirmada)
- QoS nivel 2 – Exactamente una vez: el remitente y el receptor se involucran en un protocolo de dos niveles para garantizar que solo se reciba una copia del mensaje (entrega asegurada). Este nivel no es compatible actualmente con Adafruit IO

5.2.4. CONFIGURACION BLOQUES GRÁFICOS

Una vez generados los feeds, pasamos a configurar los bloques gráficos para cada uno de ellos. Para ello se accede al dashboard generado (“MInvernadero”) y se selecciona “Create a new block”. A continuación se muestran únicamente los bloques usados en el presente proyecto.

Para los datos de temperatura, humedad relativa, nivel de agua del depósito y pH, se ha seleccionado el bloque de la derecha (Gauge).

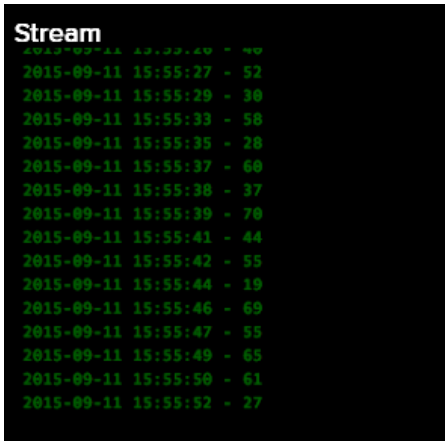


Este bloque no es más que un indicador de sólo lectura que muestra un rango de valores fijos.

Para los datos de humedad de suelo y luminosidad se ha seleccionado el bloque de la derecha (Text).

Se puede utilizar un bloque de texto para enviar y recibir datos. Para publicar, basta con hacer clic en el bloque de texto, escribir cualquier texto y presionar enter para enviar. Se empleará únicamente para recibir datos.

Para mostrar las alarmas se ha seleccionado el siguiente bloque de la derecha (Stream).



Un bloque de flujo (Stream) se puede usar para ver el historial de datos de hasta 5 feeds. Sin embargo, se empleará únicamente para las alarmas generadas por código. Lo más interesante de éste bloque es la posibilidad de mostrar la fecha y hora de la publicación del dato.

Para los botones de selección de modo, riego y ventilación se ha elegido el siguiente bloque de la derecha (Toggle).

Un botón es útil para configurar estados ON/OFF de un tema. Se puede configurar qué valores se envían al presionar y soltar.

Lo siguiente es configurar cada uno de los bloques según los límites de medida de cada parámetro del invernadero:

- **Bloque para temperatura:**

Se escoge el bloque Gauge y el feed “temperatura”. A continuación, se configura el nombre del bloque (Block Title), los niveles mínimo y máximo del medidor de bloque (Gauge Min/Max Value) y el tamaño del mismo (Gauge Width), la unidad de medida(Gauge Label), y los niveles mínimo y máximo admisibles de temperatura (Low/High Warning Value).

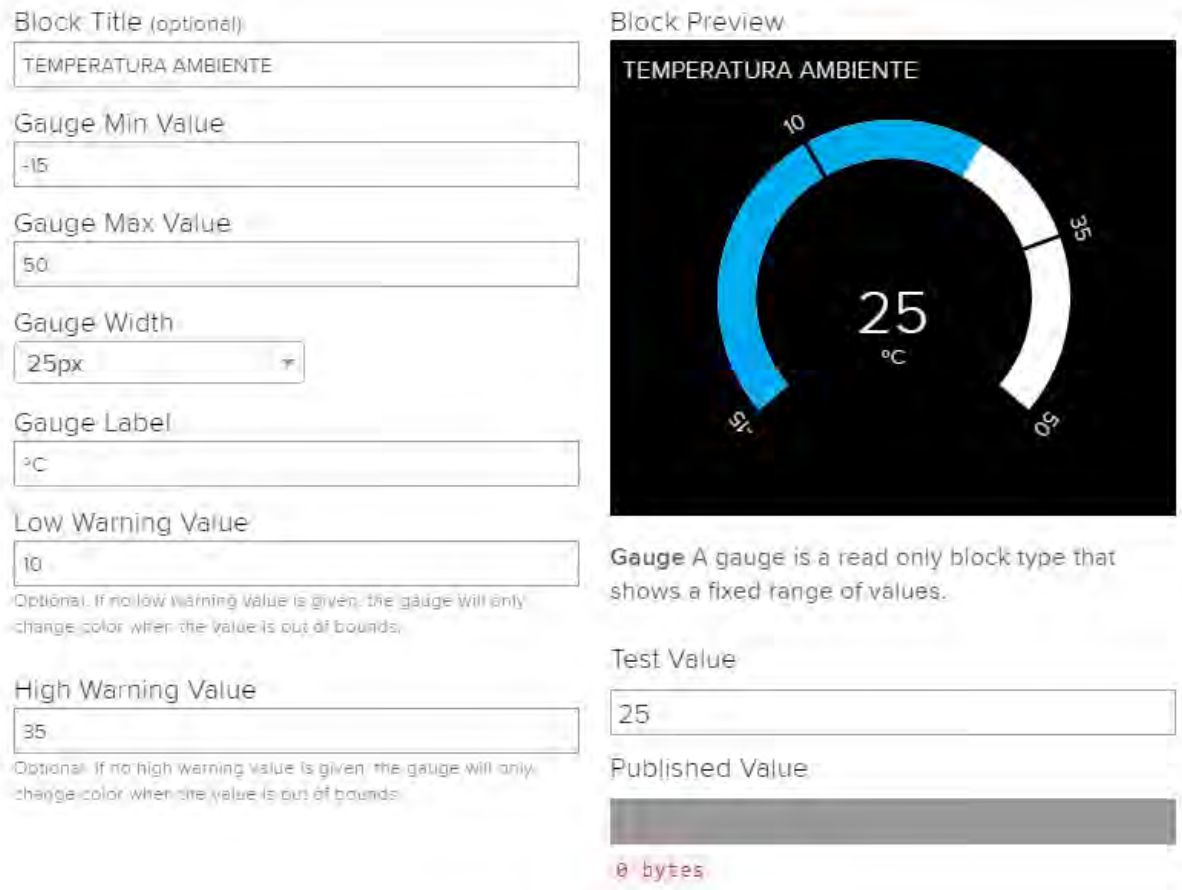


Ilustración 49 - Configuración del bloque para el feed temperatura.

- **Bloque para humedad ambiente:**

Se escoge el bloque Gauge y el feed “humedadAmbiente”. A continuación, se configura el nombre del bloque (Block Title), los niveles mínimo y máximo del medidor de bloque (Gauge Min/Max Value) y el tamaño del mismo (Gauge Width), la unidad de medida(Gauge Label), y los niveles mínimo y máximo admisibles de temperatura (Low/High Warning Value).

Exactamente igual que en el caso de la temperatura pero con los siguientes valores relacionados con la humedad ambiente del invernadero:

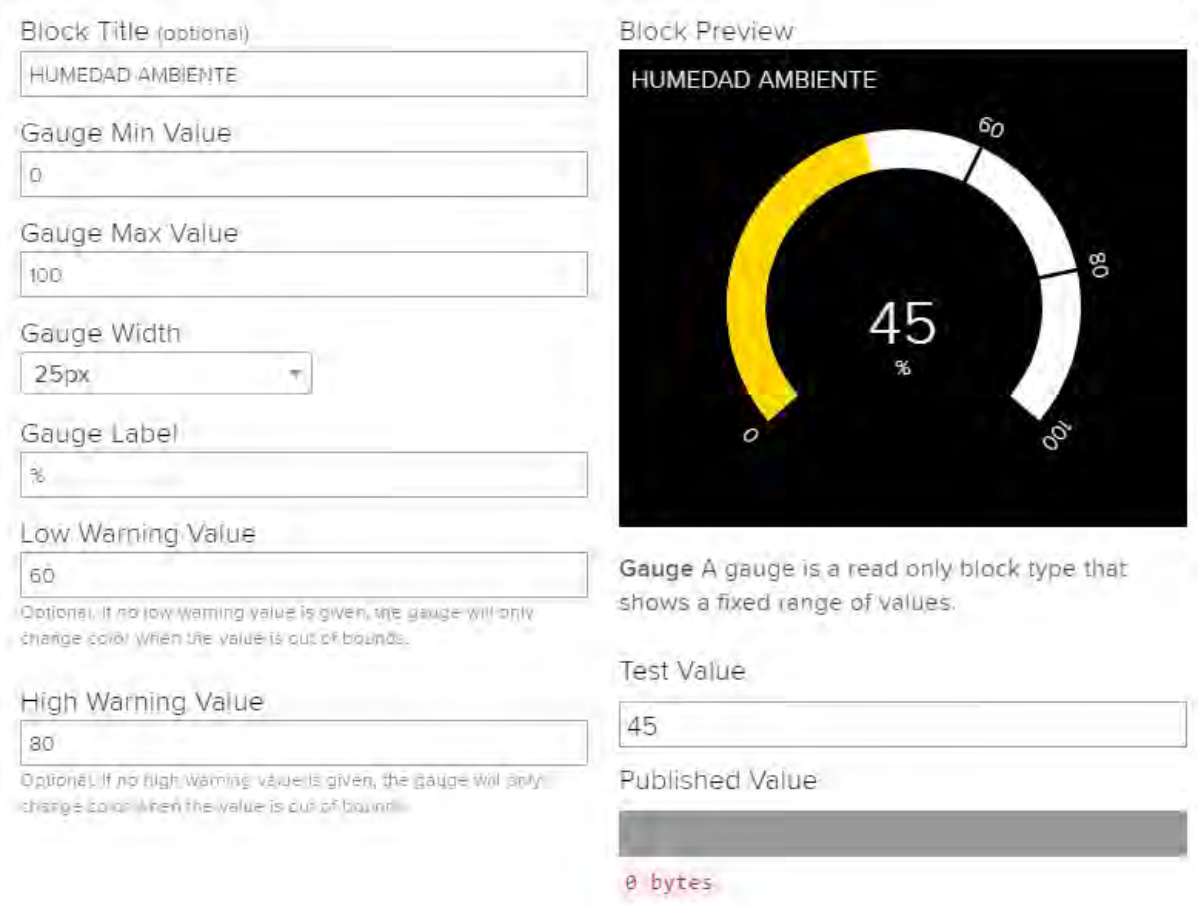


Ilustración 50 - Configuración del bloque para el feed humedadAmbiente.

- **Bloque para humedad suelo:**

Se escoge el bloque Text y el feed “humedadTierra”. A continuación, se configura el nombre del bloque (Block Title) y el tamaño de la fuente o letra (Font Size). Cabe destacar que se puede elegir la opción “Static Text” para cuando se quiera mostrar un texto estático ignorando el valor publicado en el feed. Se ha ignorado este punto ya que lo interesante es ver el estado real de la humedad del suelo del invernadero.

Por ultimo añadir que se ha elegido este bloque por mostrar otro tipo y ver su utilidad. Sin embargo, si se quiere saber directamente el porcentaje de humedad de suelo, como se ve en el caso de la humedad relativa, habría que cambiar en el código el tipo de dato que se publica, para que en lugar de publicar el texto: “seco”, “ideal” o “humedo”, se publique directamente el valor numérico o porcentaje. Además se puede publicar ese valor en este bloque de tipo Texto, sin necesidad de usar un bloque Gauge.

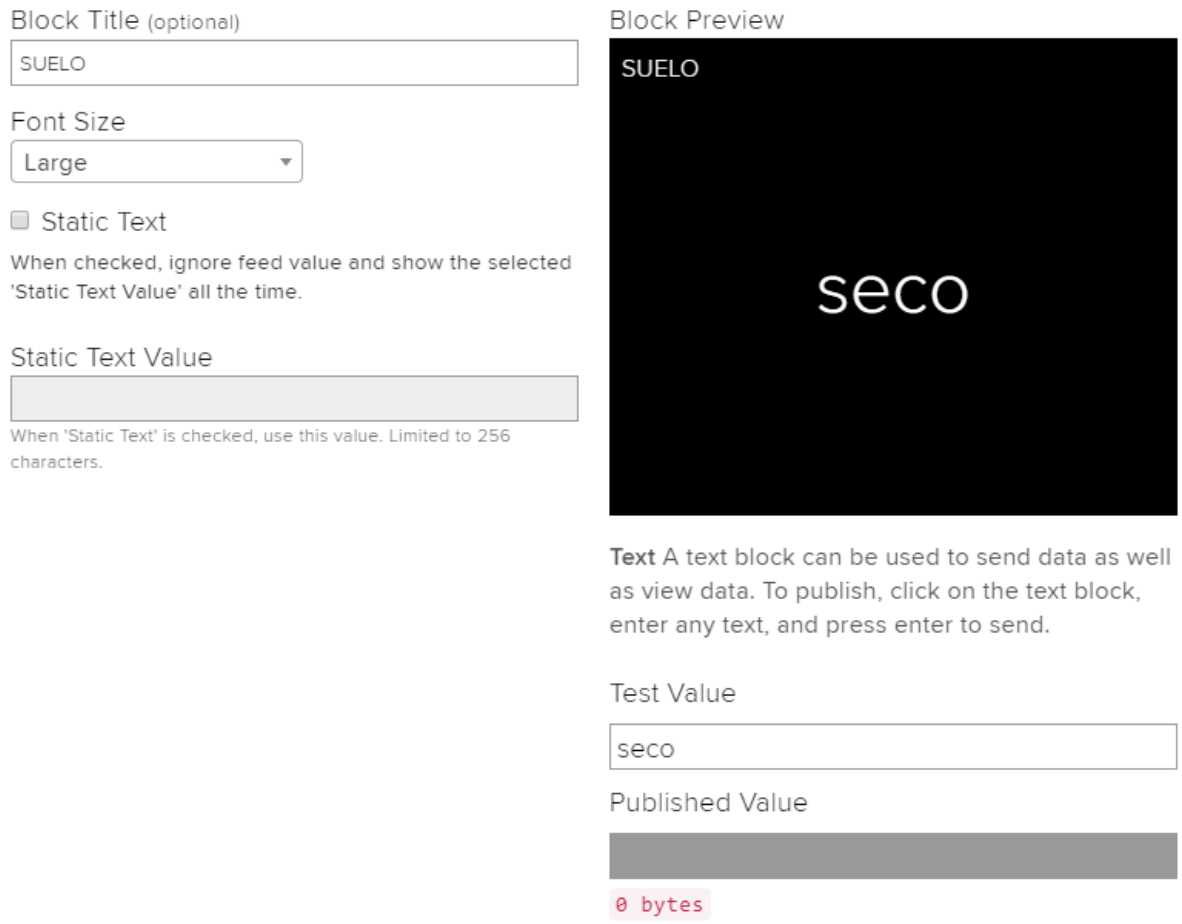


Ilustración 51 - Configuración del bloque para el feed humedadTierra.

- **Bloque para luminosidad:**

Se escoge el bloque Text y el feed “luminosidad”. A continuación, se configura el nombre del bloque (Block Title) y el tamaño de la fuente o letra (Font Size). Ocurre lo mismo que en el caso anterior de humedad de tierra. Se ha optado por el uso de bloque de texto por la configuración del código en el IDE de Arduino que aporta directamente la luminosidad como texto: “baja”, “media” o “alta”, sin embargo también podríamos modificarlo y mandar un valor numérico directamente. Para ver el efecto de esto, podemos poner un valor de ejemplo en “Test Value”.

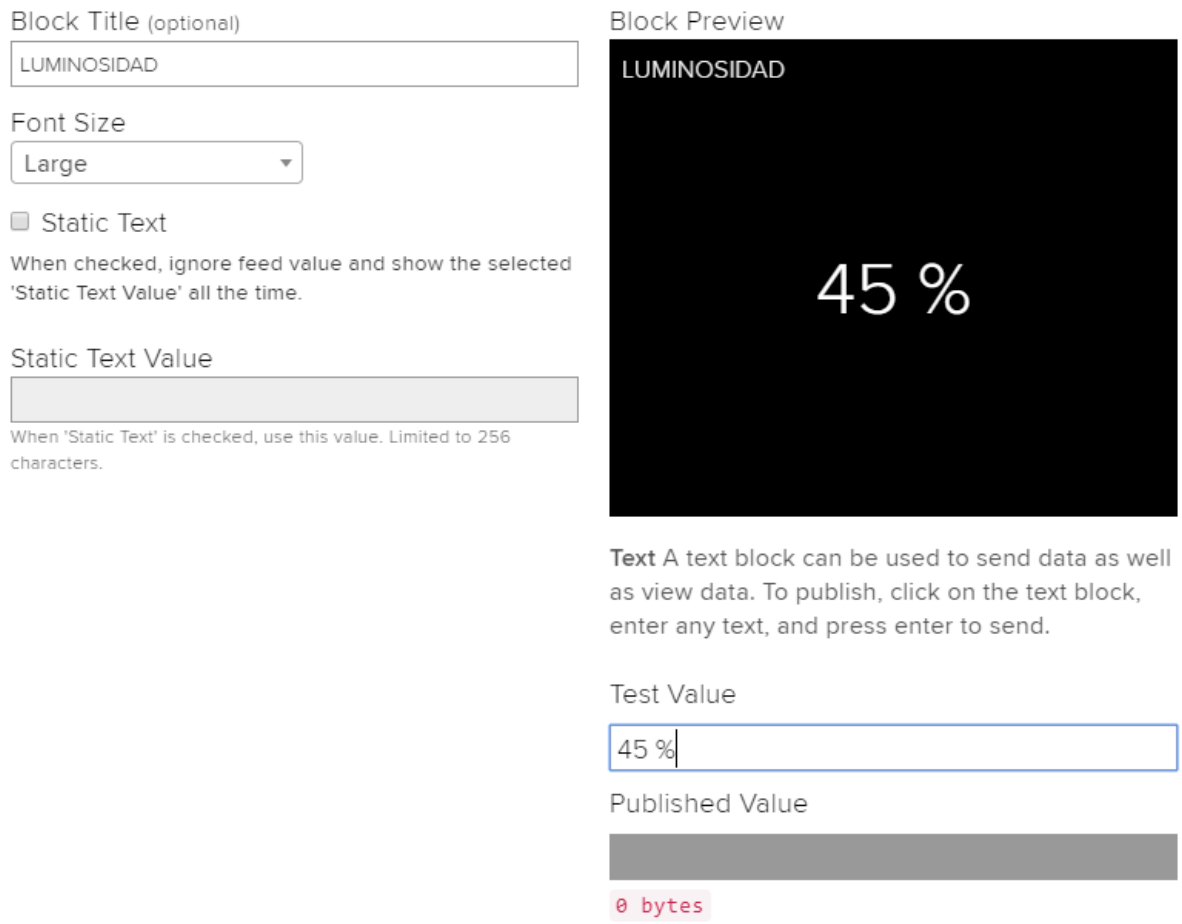


Ilustración 52 - Configuración del bloque para el feed luminosidad.

- **Bloque para pH:**

Se escoge el bloque Gauge y el feed “pHAgua”. A continuación, se configura el nombre del bloque (Block Title), los niveles mínimo y máximo del medidor de bloque (Gauge Min/Max Value) y el tamaño del mismo (Gauge Width), la unidad de medida(Gauge Label), y los niveles mínimo y máximo admisibles de temperatura (Low/High Warning Value). Exactamente igual que en el caso de la temperatura y la humedad ambiente pero con los siguientes valores:

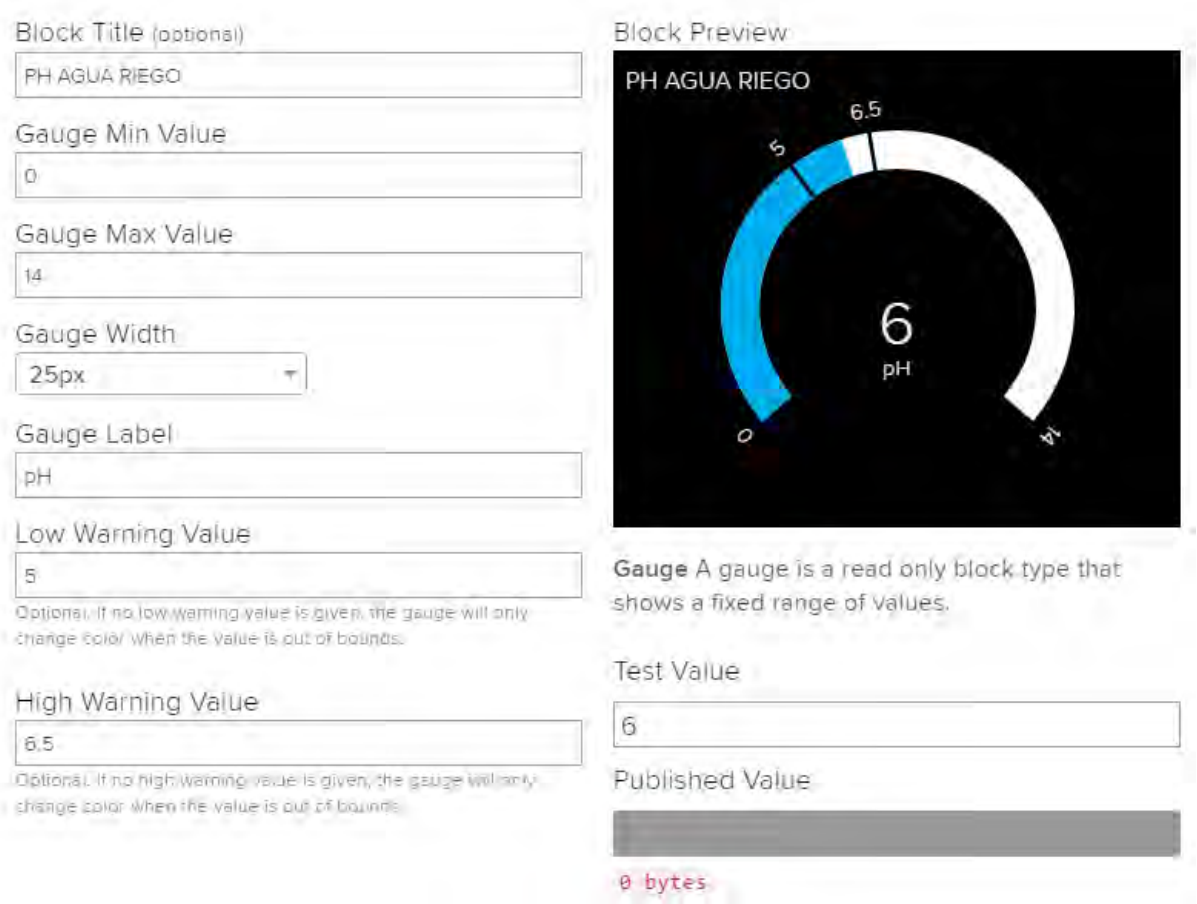


Ilustración 53 - Configuración del bloque para el feed pHAgua.

- **Bloque para nivel de agua en depósito:**

Se escoge el bloque Gauge y el feed “nivelAgua”. A continuación, se configura el nombre del bloque (Block Title), los niveles mínimo y máximo del medidor de bloque (Gauge Min/Max Value) y el tamaño del mismo (Gauge Width), la unidad de medida(Gauge Label), y los niveles mínimo y máximo admisibles de temperatura (Low/High Warning Value). Exactamente igual que en el caso de la temperatura y la humedad ambiente pero con los siguientes valores:

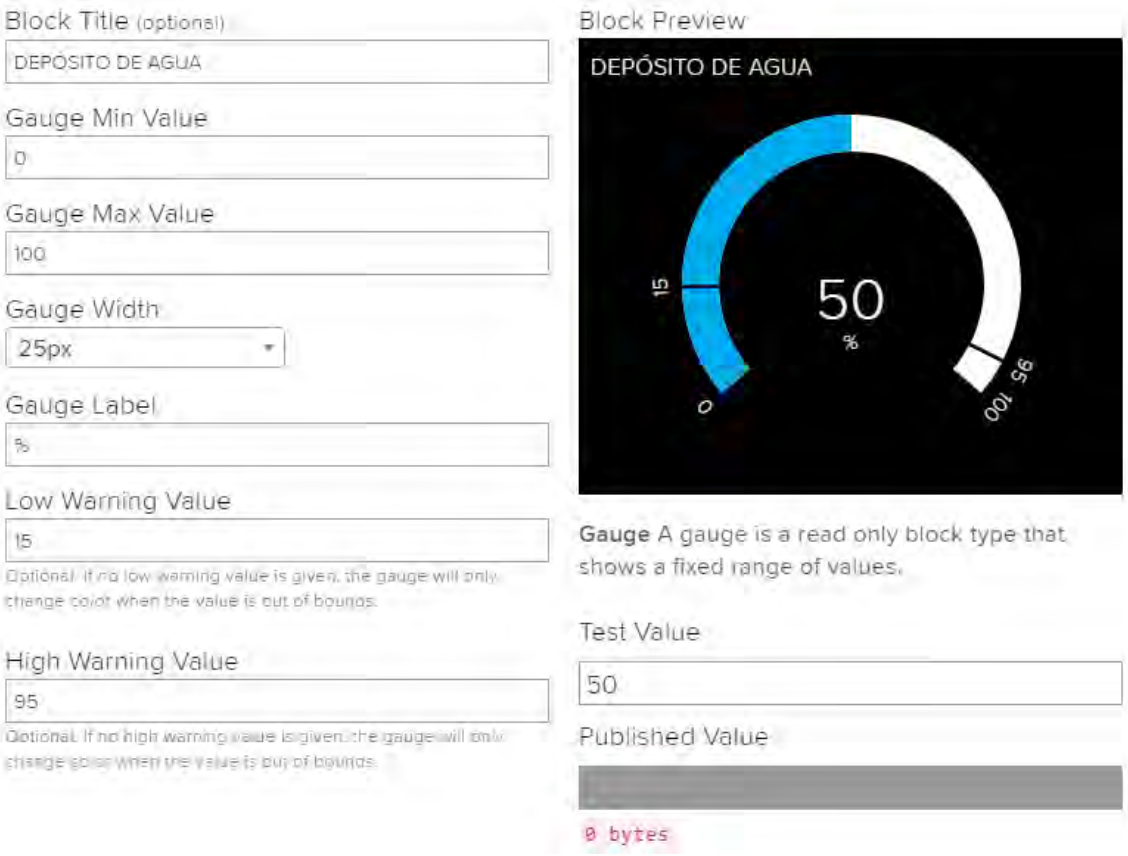


Ilustración 54 - Configuración del bloque para el feed nivelAgua.

- **Bloque para alarmas:**

Se escoge el bloque Stream y el feed “alarmas”. Recordar que se puede escoger más de un feed (hasta 5). Esto puede ser interesante si se quiere ver errores en las publicaciones de un tema.

Group / Feed		Last value	Recorded	
☑ alarmas	🔒	FRIO EXTREMO, P...	5 days	1 of 5
☐ humedadAmbiente	🔒	51.80	5 days	
☐ humedadTierra	🔒	seco	5 days	
☐ luminosidad	🔒	baja	5 days	
☐ modoAuto	🔒	A	5 days	
☐ nivelAgua	🔒	47.09	5 days	
☐ pHAgua	🔒	8.40	5 days	
☐ riego	🔒	OFF	5 days	
☐ temperatura	🔒	19.40	5 days	
☐ ventilacion	🔒	OFF	5 days	

Ilustración 55 - Selección de feeds para configurar el bloque Stream.

Se añade un nombre para el bloque, el tamaño de letra y el color de la fuente. Además se selecciona lo que se quiere mostrar en el bloque. En este caso, se mostrará fecha y hora de publicación del error o alarma y el nombre del feed. No se considera necesario mostrar la localización ni los errores del feed publicado (“alarmas”).

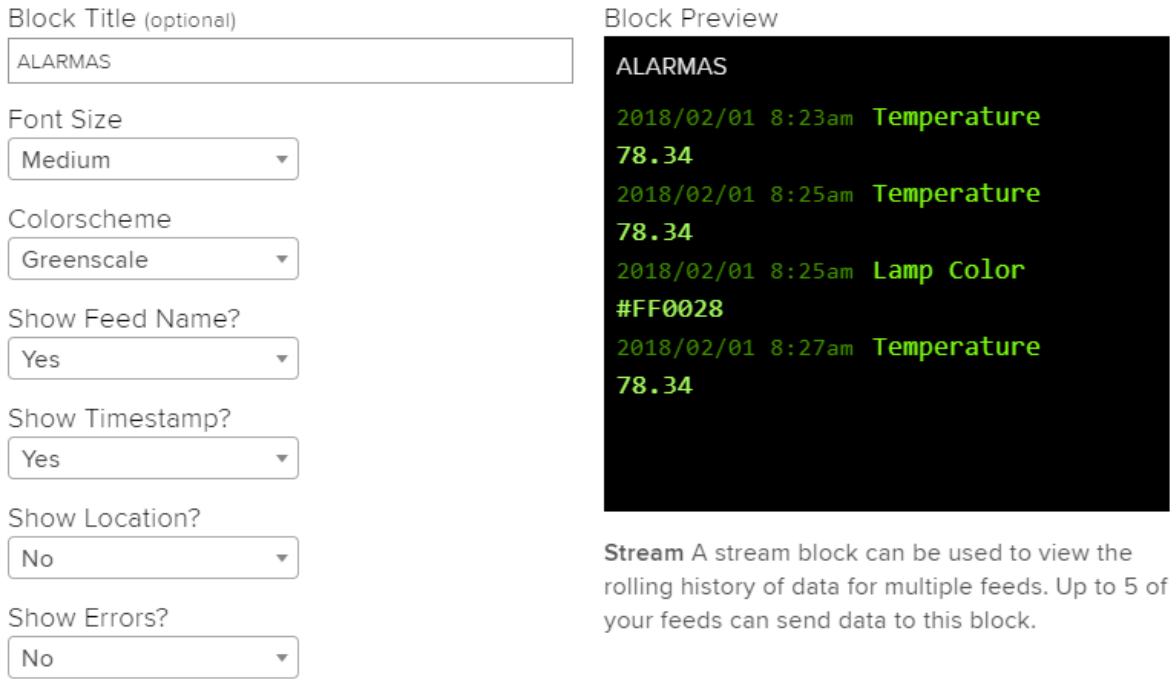


Ilustración 56 - Configuración del bloque para el feed alarmas.

- **Bloque para selección de modo:**

Se escoge el bloque Toggle y el feed “modoAuto”. A continuación damos nombre al bloque y al texto de posición de encendido ON y de apagado OFF. En este caso, se considera que el bloque activa el modo automático (A) y se desactiva para dar paso al modo manual (M).

Ilustración 57 - Configuración del bloque para el feed modoAuto.

- **Bloque para activación riego:**

Se escoge el bloque Toggle y el feed “riego”. A continuación damos nombre al bloque y al texto de posición de encendido ON y de apagado OFF. En este caso, se considera que el bloque activa el riego (ON) o lo desactiva (OFF).

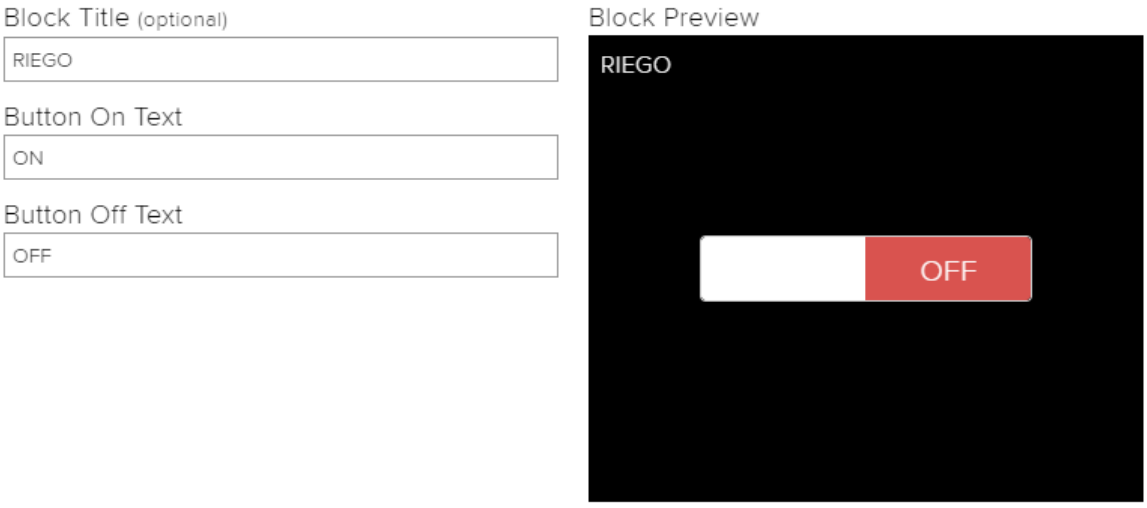


Ilustración 58 - Configuración del bloque para el feed riego.

- **Bloque para activación ventilación:**

Se escoge el bloque Toggle y el feed “ventilacion”. A continuación damos nombre al bloque y al texto de posición de encendido ON y de apagado OFF. En este caso, se considera que el bloque activa la ventilación (ON) o la desactiva (OFF).

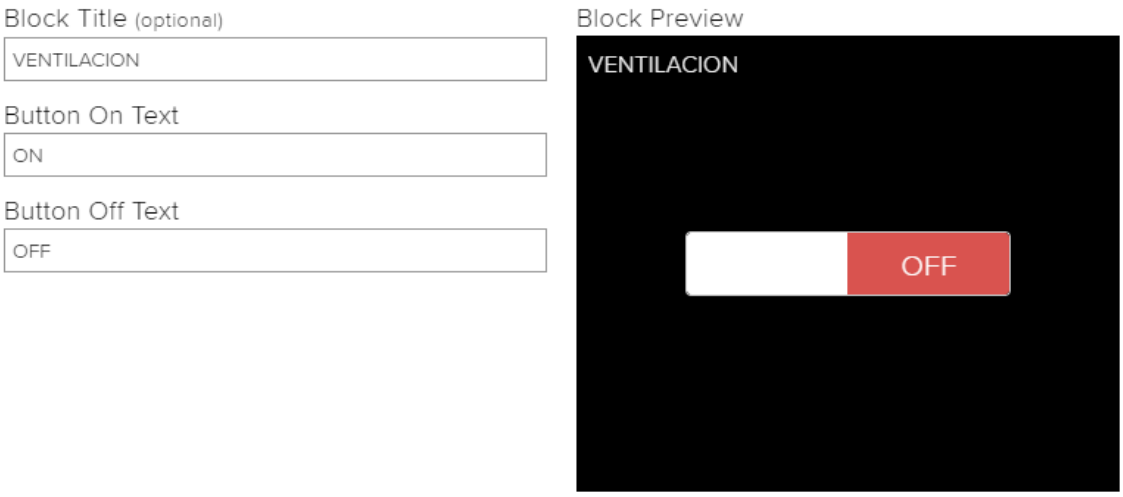


Ilustración 59 - Configuración del bloque para el feed ventilacion.

A continuación se muestra la visión global del Dashboard (tablero) creado. Se puede acceder al servidor web Adafruit IO mediante cualquier dispositivo con conexión a Internet y navegador web, solo debemos buscar la página Adafruit.io y acceder al tablero con nuestras credenciales de Adafruit (usuario y contraseña de la cuenta).

Ilustración 60 - Visión global del Dashboard creado.

5.3. VISUALIZACIÓN EN APP ANDROID

En este apartado se pretende conectar un nuevo cliente al servidor broker Adafruit IO. El cliente pretende ser una aplicación Android desde la cual podamos monitorizar los mismos datos que se mostraban en el servidor web de Adafruit. Es decir, gracias a una aplicación externa (App Android), se podrá acceder a los temas o feeds del servidor broker mediante el uso de un móvil o una Tablet.

La aplicación que se usará para el ejemplo es una App Android de uso gratuito que se puede descargar en Play Store: ***MQTT Dash (IoT, Smart Home)***. Aunque hay otras opciones como *IoT MQTT Panel, IoT MQTT Dashboard o Linear MQTT Dashboard…* se decide emplear MQTT Dash por las características que ofrece, como la usabilidad, manejo y buen rendimiento. Aunque la interfaz de usuario no es aparentemente muy bonita, ofrece algunas soluciones de diseño, como formas y colores.

Ilustración 61 - App MQTT Dash.

Características de la App [49]:

- Se pueden crear paneles para dispositivos, aplicaciones y domótica para el hogar inteligente IoT habilitados para MQTT.
- Compatible con Teléfonos móviles y tablets.
- Se pueden emplear varios bloques para un mismo tema.
- Interfaz de usuario sencilla y fácil de usar con tableros de mandos simples.
- Soporte para JavaScript.
- Protocolo estándar MQTT, lo que hace más sencilla la conexión de distintos dispositivos juntos.
- Soporte para M2M, Sonoff, Electrodragon, esp8266, Arduino, Raspberry Pi, microcontroladores (MCU), sensores, computadoras, bombas, termostatos, control remoto y otras cosas.

Como se vio anteriormente, para conectar un cliente al servidor Adafruit IO se necesita [47]:

- Host: io.adafrit.com
- Port: 1883 o 8883 (para conexión cifrada SSL)
- Username: el nombre de usuario de la cuenta de Adafruit
- Password: clave AIO (Adafruit.io)

Para crear el tablero, se accede a la aplicación y si aún no se a añadido ningún tablero, aparecerá una pantalla negra junto con tres símbolos en la parte superior derecha.

Ilustración 62 - Primera vista de la App.

Para añadir un nuevo tablero o dashboard, se seleccionará el símbolo (+) y aparecerá la siguiente pantalla de configuración:

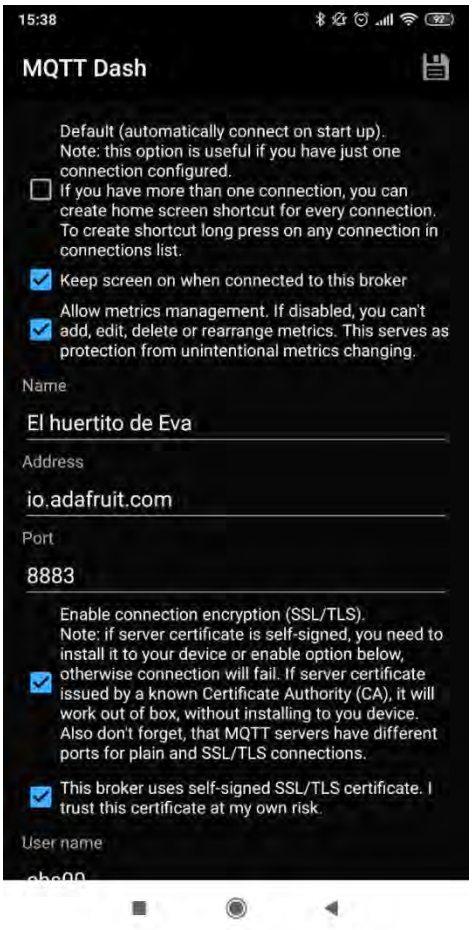


Ilustración 63 - Configuración tablero App. Captura 1

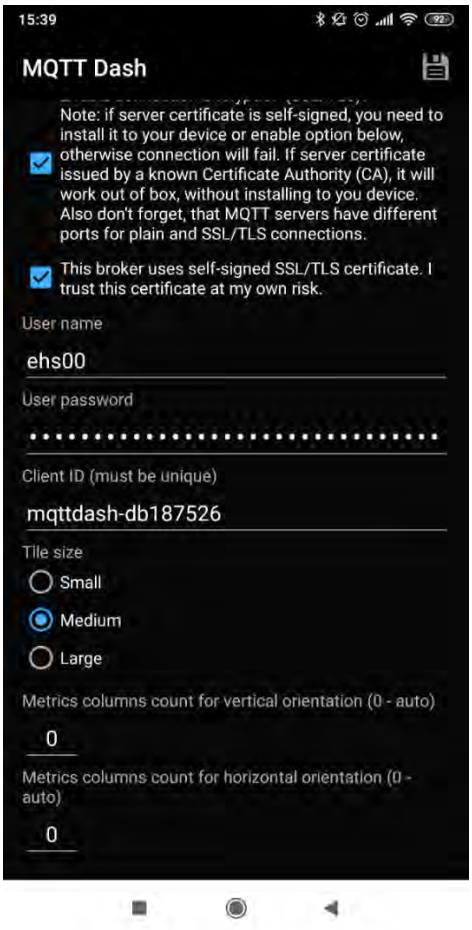


Ilustración 64 - Configuración tablero App. Captura 2

En las imágenes seleccionadas de configuración de tablero, aparece el ejemplo de configuración para el trabajo. Lo primero que vemos son unas casillas de selección:

- ✓ La primera hace referencia a la conexión, se habilitará principalmente si se va a realizar una sola conexión de cliente, para que éste se conecte directamente al servidor. Si hay más de una conexión esta función puede no resultar muy útil.
- ✓ La segunda es para mantener encendida la pantalla mientras el cliente esté conectado.
- ✓ La tercera es para permitir cambios en las métricas del tablero. Si se deshabilita, no se podrán agregar, editar, eliminar o reorganizar. Una vez que acabemos las configuraciones podremos deshabilitar esta función.

Una vez seleccionadas las casillas correspondientes se procede a agregar un nombre de tablero, en este caso “El huertito de Eva”. Luego añadir la dirección del servidor broker utilizado (io.adafruit.com) y el puerto necesario, en este caso se probará la conexión cifrada SSL, por lo que el puerto que debemos poner es el 8883. Habilitaremos también las siguientes casillas:

- ✓ La cuarta función es para habilitar el cifrado de conexión SSL / TLS. Habrá que tener en cuenta que MQTT tiene puertos diferentes para conexiones planas y SSL / TLS, por lo que si habilitamos este punto, deberemos poner el puerto 8883 para la conexión cifrada SSL del cliente al servidor.
- ✓ La quinta y última casilla es para confirmar que el cliente utiliza un certificado SSL / TLS autofirmado.

A continuación, se escribirán las credenciales del servidor broker para la conexión al mismo, es decir, se añade el nombre de usuario y la llave como hicimos para la conexión del cliente ESP32 con Adafruit IO. El resto de datos se pueden dejar tal y como están por defecto en la configuración. Guardamos la configuración del tablero y el Dashboard ya estará generado:

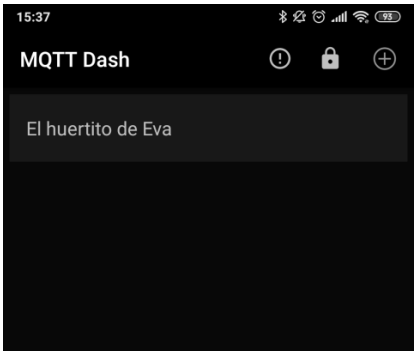


Ilustración 65 - Primera vista del tablero creado en la App

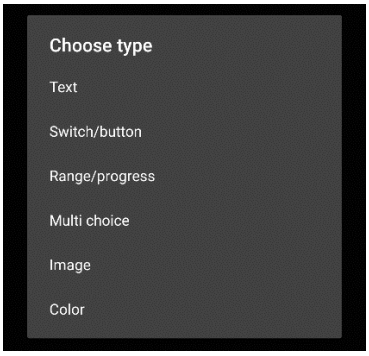


Ilustración 66 - Bloques o Widgets disponibles en la App

Una vez generado el tablero, se pueden empezar a añadir los bloques que conectaremos a los feed del broker. Se selecciona el tablero y una vez dentro, se selecciona el símbolo (+) para elegir el tipo de bloque o *"widget"* a configurar.

Los widgets o bloques disponibles son de tipo texto, botones/interruptores, rango/progreso, multi-elección, imagen y color. Podemos empezar con el tipo *Switch/button* para añadir los botones de selección de modo y activación de los sistemas de riego y ventilación. Estos botones alternan entre estado apagado (OFF) y encendido (ON).

Ilustración 67 - Configuración Widget Selección de Modo.

En la imagen de la izquierda se muestra un ejemplo de configuración. En este caso se añade un botón de selección de modo. Para que el topic de la App esté anclado al feed del Servidor de Adafruit, es necesario incluir la ruta específica del feed. En este caso será: ***ehs00/f/modoAuto.***

Se selecciona la habilitación para publicar en el feed, si no se habilita es como si no se anclasen ambos interruptores, el de la App y el del broker.

Se configuran los mensajes de publicación (A para activar el modo Automático y M para el modo Manual), luego se puede elegir el icono para que aparezca en cada caso. Buscando entre los disponibles, se han encontrado unos que encajan bastante bien con la definición de A=Automático y M=Manual. Por último se puede dejar el resto de la configuración por defecto, ya que no es relevante en este caso. Información QoS en [50].

A continuación se muestran las configuraciones de los botones de activación de riego y de ventilación. Se hace de forma similar al caso anterior de ejemplo.

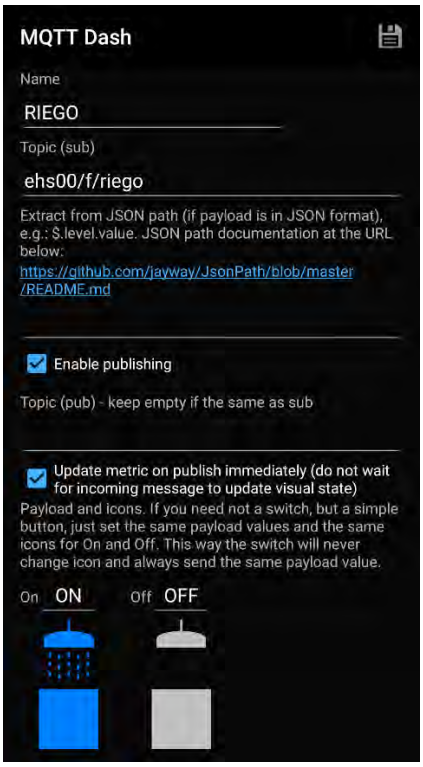


Ilustración 68 - Configuración Widget Riego

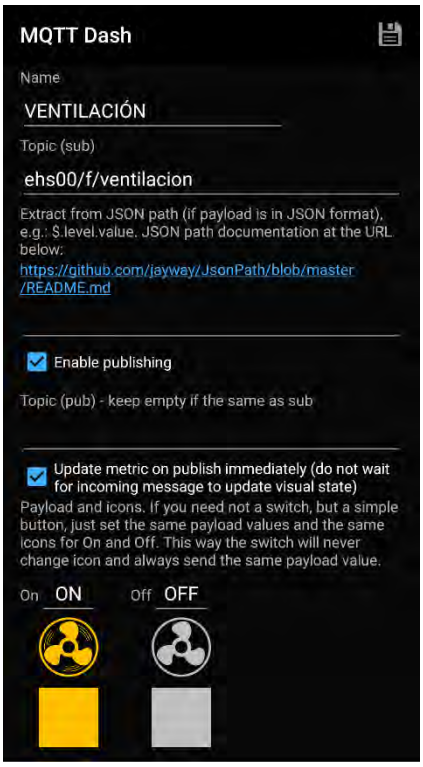


Ilustración 69 - Configuración Widget Ventilación

El siguiente paso es añadir los bloques relativos a los datos de los sensores y de alarmas. Los datos publicados en los siguientes feed del servidor: *temperatura, humedadAmbiente, nivelAgua y pHAgua,* serán monitorizados por los widget de rango/progreso, mientras que los datos de los feeds: *humedadTierra, luminosidad y alarmas*, serán monitorizados por los widgets de texto.

Estos widgets o bloques serán solo de lectura (subscripción), por lo que no se tendrá habilitada la opción de publicar, para que no haya cambios en los datos que se envían al servidor.

A continuación se muestra la configuración de los widgets relativos a los sensores y a las alarmas. Y finalmente la visión general del panel o tablero (*El huertito de Eva*) de la aplicación móvil.

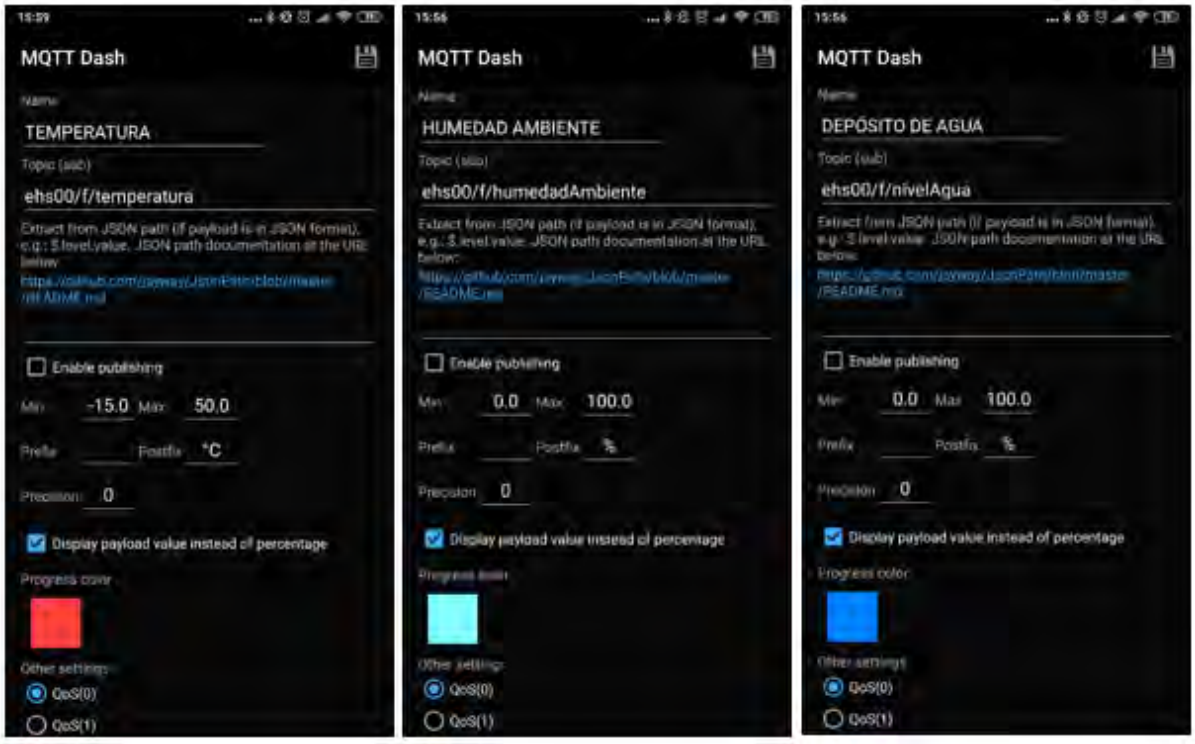


Ilustración 70 - Configuración Widgets 1.

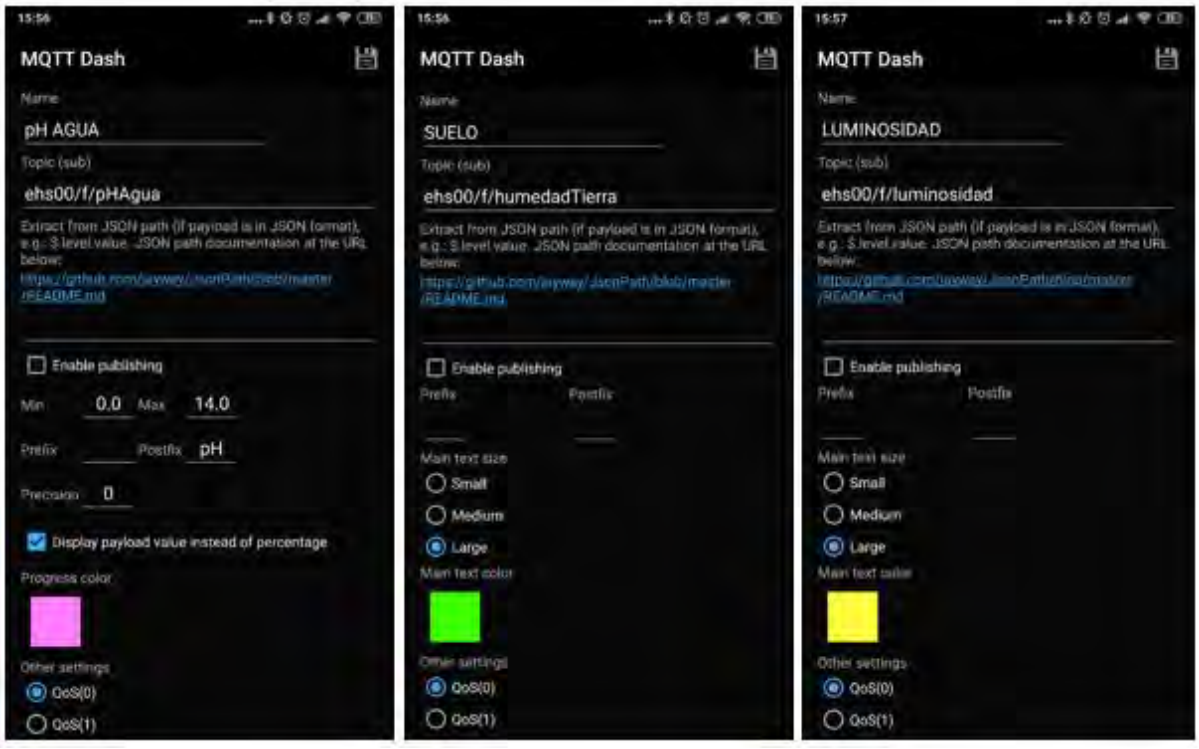


Ilustración 71 - Configuración Widgets 2.

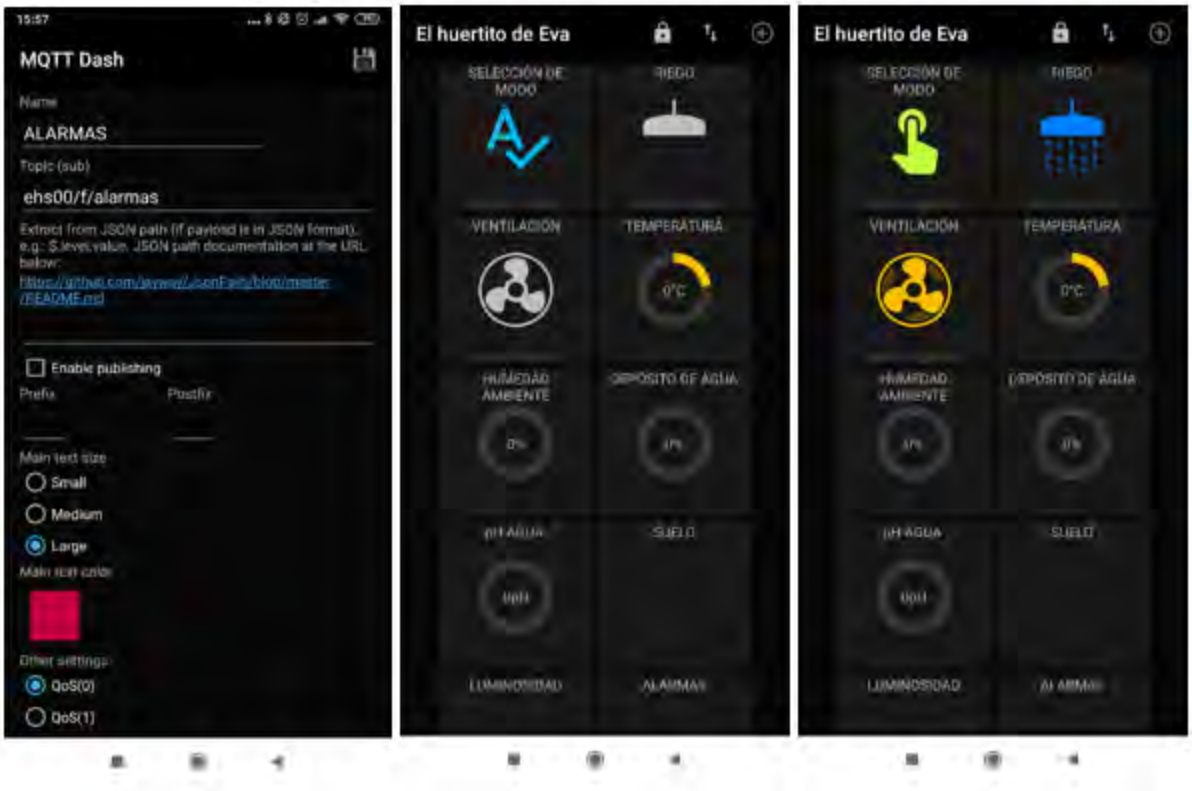


Ilustración 72 - Configuración Widgets 3 y Visión global del tablero.

5.4. ENVÍO ALERTAS POR EMAIL

5.4.1. PLATAFORMA IFTTT

Existe una plataforma que permite crear conexiones entre diversos servicios en línea, servicios de almacenamiento de datos en la nube, correos electrónicos, y multitud de servicios y sitios web y más. Esta plataforma es IFTTT, cuyas siglas corresponden con "IF THIS THEN THAT", que no es más que una estructura condicional usada en programación que permite como su nombre indica, relacionar dos plataformas y establecer una condición y una tarea entre ellas. Es decir, si determinada condición (IF THIS) de la primera plataforma ocurre, entonces (THEN), ejecuta una acción específica (THAT) en la segunda plataforma.

Hay multitud de sitios soportados por IFTTT, como redes sociales entre las que destacan Twitter, Instagram y Facebook, otros servicios de almacenamiento de datos en la nube como Google Drive, OneDrive, Dropbox, etc… servicios email como Gmail y OfficeMail, y otras muchísimas aplicaciones, sitios web, servidores, blogs, servicios de herramientas de negocio, calendarios y agendas… Cada vez se expande a más sitios en línea y es por eso que entre todos ellos se encuentra Adafruit. Más información sobre IFTTT en [51], [52] y [53].

Antes de pensar en qué se va a hacer en el proyecto con IFTTT, es importante hablar del cómo. A continuación se describen unos pasos básicos para la creación de cualquier condicional, sea el que sea.

- Lo primero es **crear una cuenta**, se puede empezar vinculando con la cuenta de Google o Facebook si se dispone de alguna de ellas. Se rellenarán los datos básicos de usuario y se confirmará el correo electrónico que llegará al usuario.
- **Activación de canales**: Se pueden integrar los sitios o plataformas que se quiera, una por una y concediendo los permisos que se piden. Por ejemplo se puede activar ya el canal de Adafruit y el del correo Gmail, ya que son los que usaremos para la generación de alertas.
- **Crear una receta**: Se selecciona la pestaña **New Applet** de la cuenta IFTTT. Y aparecerá el condicional **"If this then that"**. A continuación vamos a ver los pasos a seguir de uno en uno.

Ilustración 73 - IFTTT – "this".

1. Si se hace clic en la palabra **"this",** aparecerán todos los canales disponibles, se selecciona uno de ellos según cual se quiera usar. En el buscador se puede escribir **"Adafruit"** para buscarlo.

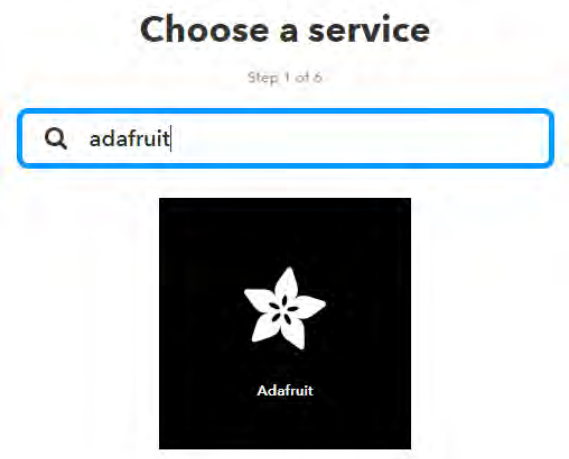


Ilustración 74 - Selección "this".

2. Una vez seleccionado el canal, se pasa a elegir el trigger, es decir, el disparador que se utilizará en el condicional. En este caso hay dos opciones:

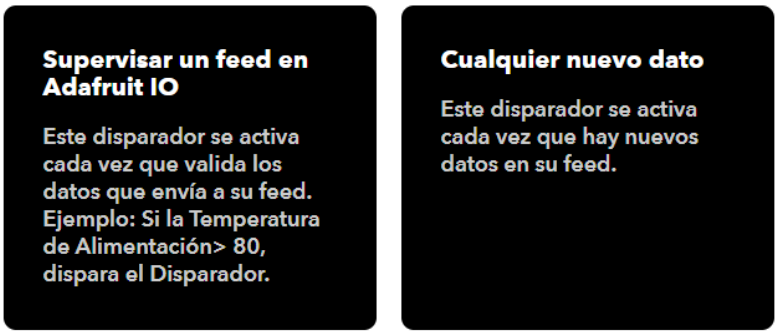


Ilustración 75 - Selección "trigger".

El escoger un trigger u otro dependerá de lo que se quiera hacer. En el siguiente punto se expondrán todos los casos de alertas generadas donde el trigger principal usado es el primero: *Supervisar un feed en Adafruit IO*, ya que nos proporciona un condicional extra donde se podrá comparar el estado del feed con un valor concreto.

Ilustración 76 - Establecer valor del "trigger".

Señalar, que una vez que está conectado a Adafruit (recordar que hay que activar previamente el canal para poder acceder al mismo), en el primer desplegable aparecen todos los feeds disponibles del dashboard *MInvernadero* de Adafruit IO.

Se expone en este ejemplo, cómo generar un email automático para la alerta de heladas. Cuando la temperatura sea menor que 0ºC, enviará un correo electrónico a una cuenta de Gmail avisando de la alerta.

3. Cuando ya se ha generado el trigger, vuelve a aparecer el condicional IFTTT, pero esta vez se seleccionará la palabra **"that"**.

Ilustración 77 - IFTTT – "that".

Se busca Gmail, como se hizo con Adafruit, y se selecciona. Aparecerán tres opciones de acciones, cualquiera de ellas puede ser interesante, sin embargo para el ejemplo se escoge la segunda:

Ilustración 78 - Selección de acción.

4. Aparecerá una pantalla de configuración para determinar el mensaje que se quiere enviar por correo e-mail.

Ilustración 79 - Configurar acción. Mensaje e-mail.

Se tienen opciones para incororar ingredientes, como el nombre del feed y su valor, el valor del trigger o incluso la procedencia del dato. Esto es interesante para generar el mensaje más concreto. En el ejemplo de la derecha se emplea el ingrediente “FeedName” para el título del correo. En este caso aparecerá como Alarma de temperatura. Y en el cuerpo del mensaje se emplea el ingrediente “FeedValue” para que quede registrada la temperatura leída causante de la alerta. Por ejemplo si se lee que hay -2ºC < 0ºC, el mensaje que se generará será: ***La temperatura registrada en el invernadero es -2 ºC! ¡¡ Peligro de helada !!***

5. Se revisa que todo esté correcto, se activa el applet (ON) y se prueba. Los applets desactivados no se ejecutarán, y aparecerán sombreados.

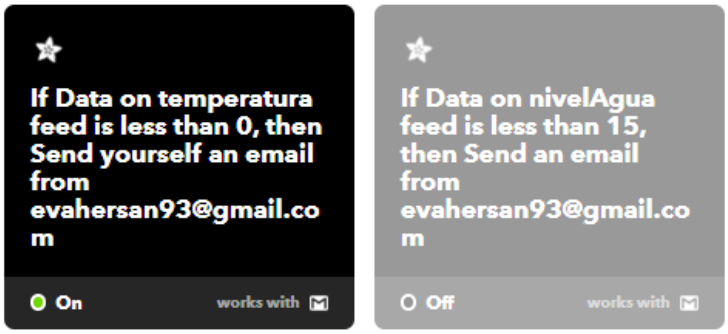


Ilustración 80 - Activar o desactivar acciones.

Para el ejemplo se ha revisado la bandeja de entrada de Gmail y se ha encontrado el siguiente mensaje:

Ilustración 81 - Correo e-mail enviado a Gmail y generado por IFTTT.

5.4.2. GENERACIÓN AUTOMÁTICA DE E-MAILS

En el apartado anterior se ha visto cómo crear una receta en IFTTT, usando el ejemplo de una alerta de temperatura donde se leía un feed del servidor Adafruit IO y se generaba automáticamente un correo e-mail que se enviaba directamente a una dirección Gmail. A continuación lo que se hará será reunir todas las alertas que se vayan a generar en el invernadero, y que se pretenden enviar por correo a la cuenta Gmail del usuario final.

feed	Alerta	Trigger	Mensaje
temperatura (ºC)	⚠ **Helada (0ºC)**	<= 0	Peligro helada, Tª < 0ºC
	⚠ **Tª Baja (10ºC)**	< 10	Alerta frio, Tª < 10 ºC
	⚠ **Tª Alta (35ºC)**	> 35	Alerta calor, Tª > 35 ºC
humedadAmbiente (%)	⚠ **HR Baja (55%HR)**	< 55	Humedad relativa baja, HR<55%
	⚠ **HR Alta (85%HR)**	> 85	Humedad relativa alta, HR>85%
humedadTierra	⚠ **Suelo seco (25%)**	seco	Regar tierra, suelo seco
	⚠ **Suelo humedo (46%)**	humedo	Drenar tierra, suelo húmedo
pHAgua (pH)	⚠ **Agua ácida**	< 5	pH de agua de riego por debajo de 5. Agua ácida
	⚠ **Agua básica**	> 6.5	pH de agua de riego por encima de 6.5. Agua básica
nivelAgua (%)	⚠ **Nivel de agua bajo**	<= 15	Peligro bomba. Llenar depósito.
	⚠ **Nivel de agua alto**	>= 95	Peligro desborde. Revisar depósito.

Tabla 10 – Alertas y mensajes propuestos para generar alertas en IFTTT.

Se pueden gestionar las alertas de dos formas. La primera es como ya se ha descrito en el apartado anterior, se puede acceder a cada uno de los feeds y generar un correo con una alerta determinada, o bien, al disponer de un feed que trasmite directamente los mensajes de alerta hacia el servidor, se puede leer el mismo y mandar el mismo mensaje generado por correo.

Sin embargo, al ser un único feed nos limita mucho, ya que envía la alerta de una en una, generándose retardos innecesarios en la recepción de la información en tiempo real.

Aun así, se expone cómo generar los correos automáticos leyendo la información directamente del feed *alarmas*.

En este caso, se escogerá el trigger "*Cualquier nuevo dato*", de tal forma que generaremos un correo siempre que se publique un nuevo dato en el tema seleccionado:

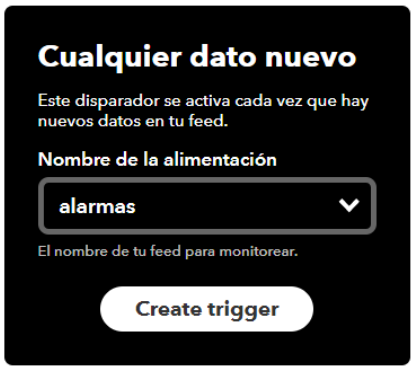


Ilustración 82 - Trigger seleccionado para el feed alarmas.

Lo siguiente será crear la receta para el correo automático, se puede hacer de la siguiente forma:

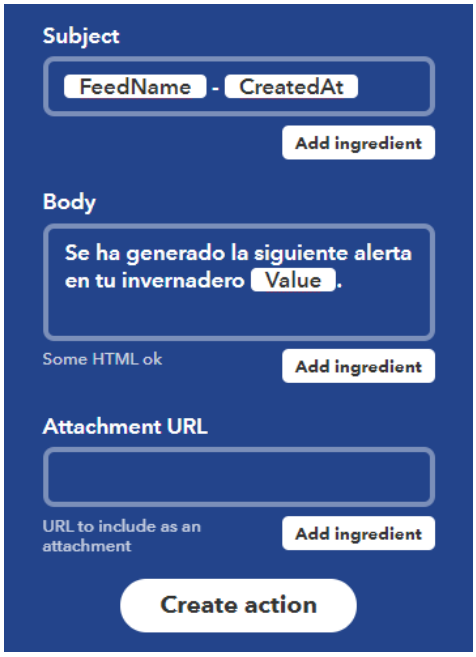


Ilustración 83 - Receta de mensaje para alarmas.

Finalmente, se muestran a continuación los Applets que se usarán para la generación automática de e-mails que se enviarán a la dirección de correo Gmail del usuario final.

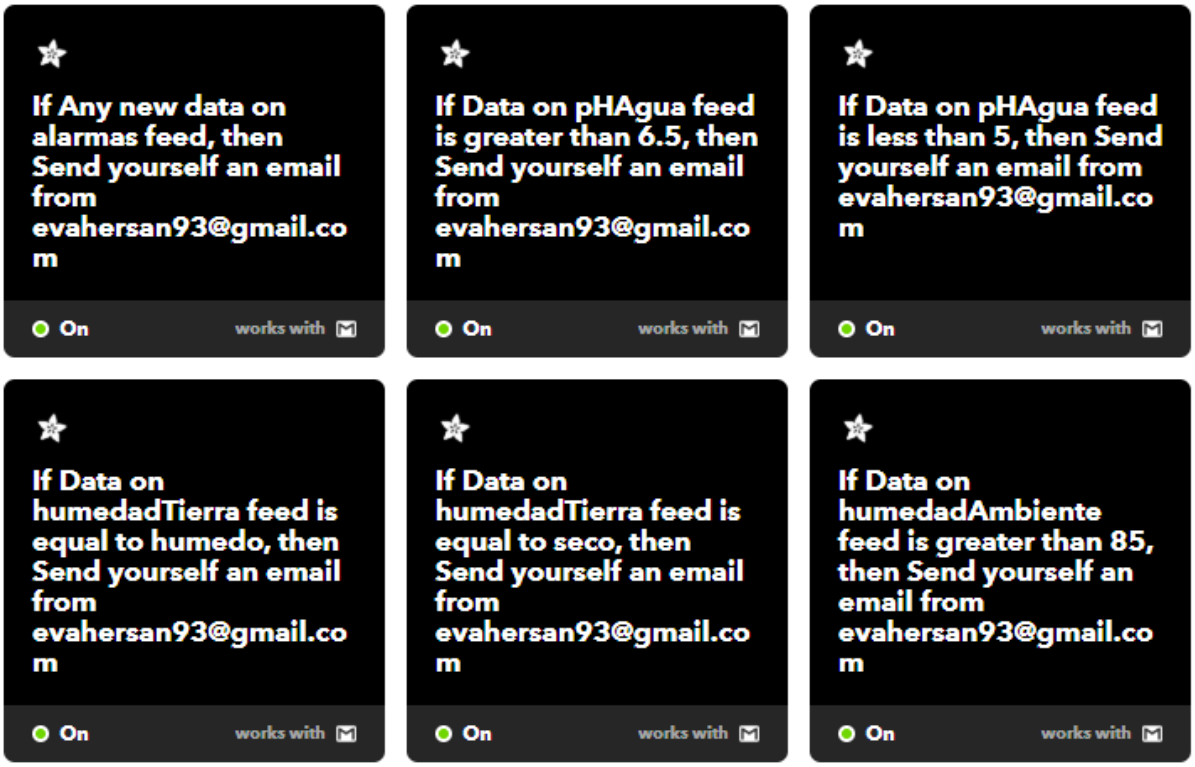


Ilustración 84 - Alertas generadas en IFTT 1

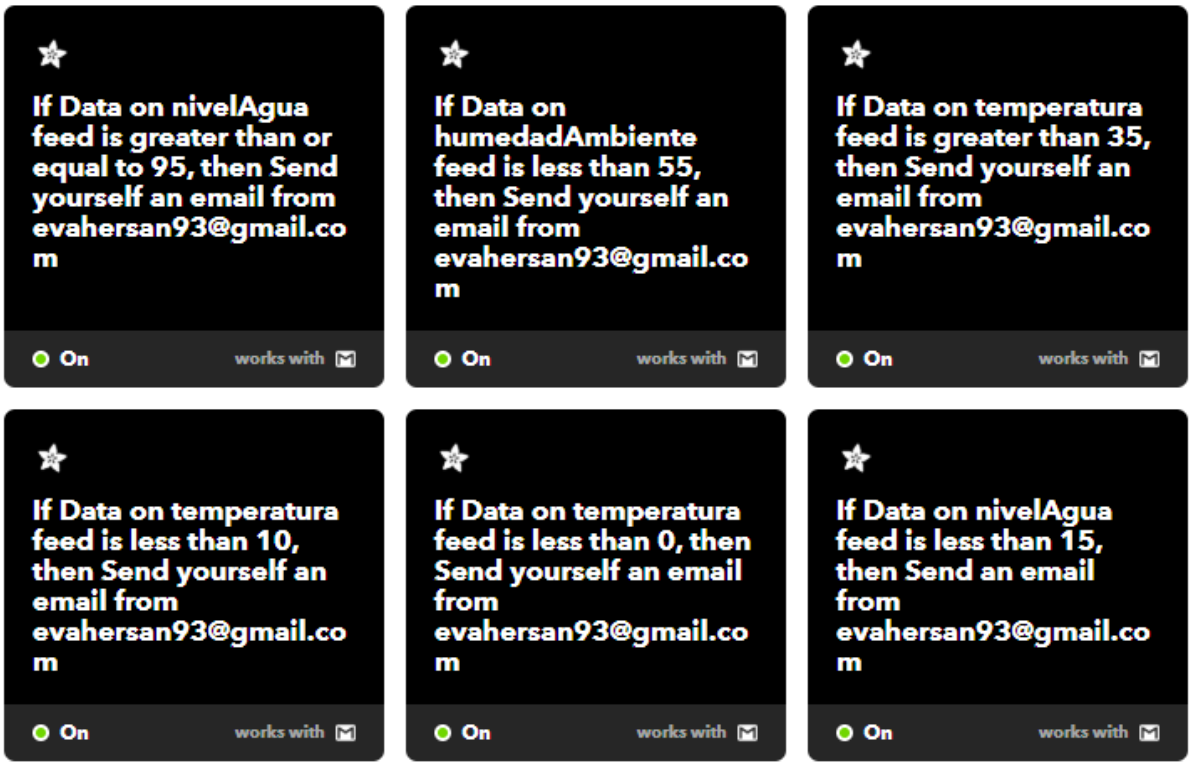


Ilustración 85 - Alertas generadas en IFTTT 2

Ilustración 86 - Bandeja de entrada Gmail.

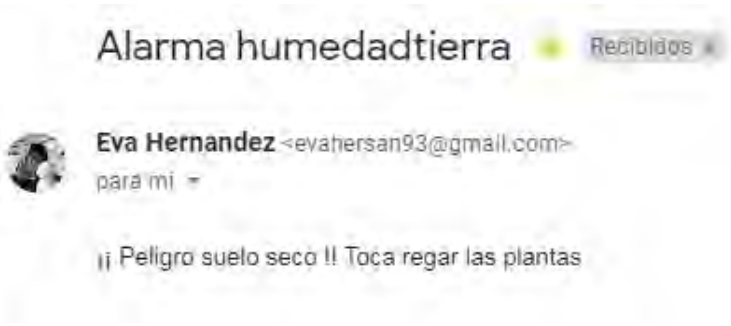


Ilustración 87 - Mensaje generado de ejemplo 1

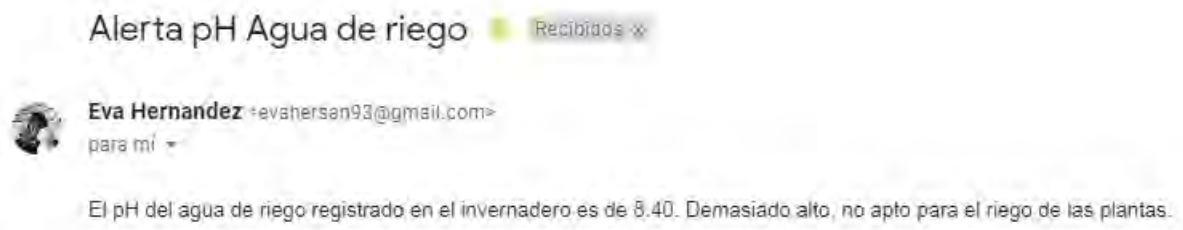


Ilustración 88 - Mensaje generado de ejemplo 2

Ilustración 89 - Mensaje generado de ejemplo 3

Ilustración 90 - Mensaje generado de ejemplo 4

6. SOFTWARE

6.1. VISIÓN GENERAL

El desarrollo del código se puede agrupar en varias secciones, de tal forma que sea entendido el funcionamiento de programa y el proceso general de forma íntegra. De manera más concreta, se pueden intuir los pequeños procesos o funciones que se establecen dentro del código en general. Por ello se presenta el siguiente esquema que define el programa como un conjunto y a su vez en pequeñas secciones:

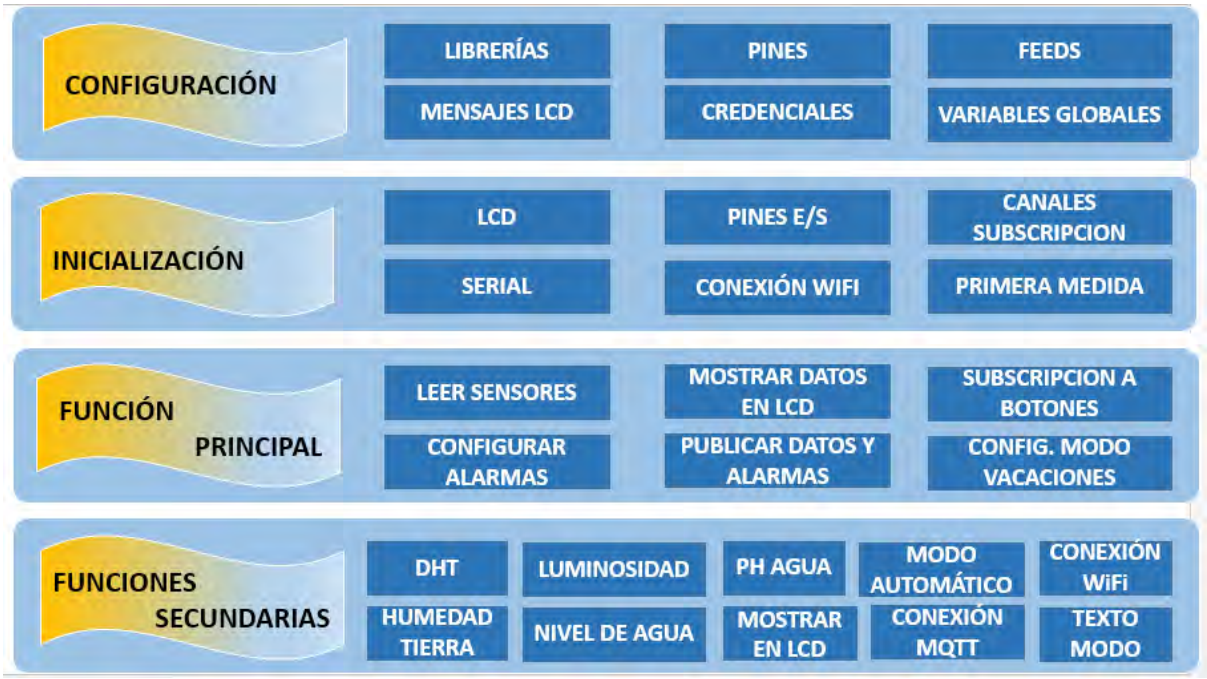


Ilustración 91 - Esquema de programa.

6.2. EL CÓDIGO

6.2.1. CONFIGURACIÓN

En la parte inicial del código se establece la configuración principal del mismo, es decir, se definen y declaran todas las librerías y variables que se usarán y que son accesibles en cualquier parte del programa. Estas variables son símbolos que almacenan valores temporales y en cualquier momento se pueden modificar.

Como en el presente proyecto se emplean multitud de variables, se ha pretendido separar las mismas por bloques para su mejor interpretación. A continuación se detallan estos bloques de variables.

6.2.2. LIBRERIAS

Las librerías o bibliotecas, son un tipo de archivo que podemos importar o incluir en nuestro programa, que contienen especificaciones de funcionalidades ya construidas de algunos sistemas, por ejemplo de la lectura de un teclado o para mostrar algo por una pantalla.

En el presente trabajo se han añadido 6 librerías con las siguientes funcionalidades:

- Establecer conexión Wi-Fi del ESP32 a una red
- Generar un cliente Wi-Fi
- Establecer conexión al Servidor Adafruit por MQTT
- Generar un cliente MQTT
- Definición y lectura del sensor de humedad y temperatura DHT
- Definición y escritura del display I2C LCD.

6.2.3. MENSAJES LCD

Se han creado unas variables para mostrar una serie de mensajes y datos en el display LCD. Como ya se ha visto previamente, el display LCD es de 2x16 (filas x columnas). Es decir, se dispone de dos líneas con 16 espacios donde se puede imprimir un caracter por cada uno de ellos.

Lo que se hace en este punto, es generar variables con un texto que quepa en una sola línea, y la cual se podrá imprimir en la primera línea (fila 0) o en la segunda línea (fila 1). Éstas líneas predefinidas están creadas como variables tipo String, y nombradas para una mejor interpretación como ERROR/INIT/MODO/ALERTA, para distinguir mensajes de error, inicialización (solo se produce una vez en el Setup), selección de modo (Autónomo/Manual) o de alertas respectivamente. En el código se distinguen los siguientes grupos que mostrarán:

- Una línea de guiones si no se muestran nuevos datos, se puede interpretar como una espera. Aparecerá principalmente cuando acaben de mostrarse los datos de los sensores.

Ilustración 92 - Mensajes LCD ejemplo 1

- Fallo en inicialización de la pantalla LCD. Si está todo bien, pasará a mostrar el texto de bienvenida.
- Texto de bienvenida. E inicializando configuración WiFi.

Ilustración 93 - Mensajes LCD ejemplo 2

- Fallo en proceso de Inicialización.
- Inicialización de sensores y actuadores. Si es correcta, en cada uno de ellos, pondrá OK, y si hay algún fallo o error, escribirá KO.

Ilustración 94 - Mensajes LCD ejemplo 3

- Conexión Wi-Fi y Servidor Adafruit. Si se establece conexión, escribirá OK, si ha ocurrido un error y no se ha podido conectar, escribirá KO.

Ilustración 95 - Mensajes LCD ejemplo 4

- Estado sin conexión. Si no hay conexión Wi-Fi, no se podrá acceder al estado Manual, por lo que, para evitar problemas en el invernadero, se activará el Modo Autónomo.
- Datos sensores. Se crea una línea para cada dato a mostrar de los sensores.

Ilustración 96 - Mensajes LCD ejemplo 5

- Datos alertas, se exponen solo 4 alertas de ejemplo: riesgo heladas, calor extremo, nivel agua bajo y tierra seca.

Cabe destacar que en todo momento aparecerá el modo de funcionamiento activo en ese momento Autónomo o Manual. Así como tres caracteres que se identificarán como:

- Si están correctamente las conexiones realizadas, tanto WiFi como MQTT con Adafruit IO, entonces aparecerá el carácter “W”, por el contrario, estará el modo Autonomo activado y en vez de una W pondrá un guión “-“.
- Si está el modo Manual, y se activa el riego, aparecerá el carácter “R” en pantalla, si no, un guión en su lugar.
- Lo mismo ocurre con la ventilación, si está activa aparecerá “V”, por el contrario aparecerá “-“.

Esto es interesante para comprobar que la conexión con el Servidor es correcta, así como si se recibe del mismo, a través de las subscripciones, si se ha activado o no alguno de los actuadores.

Ilustración 97 - Mensajes LCD ejemplo 6

6.2.3.1. PINES

Se definen los pines digitales y analógicos empleados del módulo ESP32, donde irán conectados los sensores y actuadores.

Estos pines son, como ya vimos anteriormente:

- Sensor DHT22 – GPIO 23 / SPI MOSI / Empleado como entrada digital
- Sensor YL69 – GPIO 36 / ADC1-0 / Empleado como entrada analógica
- Fotorresistencia LDR – GPIO 39 / ADC1-3 / Empleado como entrada analógica
- Sensor PH – GPIO 34 / ADC1-6 / Empleado como entrada analógica
- Ultrasonidos – GPIO 17 / TXDu2 / Empleado como salida digital para Trigger
- Ultrasonidos – GPIO 19 / SPI MISO / Empleado como entrada digital para Echo
- Relé 1 – GPIO 27 / Empleado como salida digital para activar ventilador
- Relé 2 – GPIO 14 / Empleado como salida digital para activar bomba de agua
- Buzzer – GPIO 12 / Empleado como salida digital para activar alarma
- Display LCD I2C – GPIO 21 / Wire SDA
- Display LCD I2C – GPIO 22 / Wire SCL

6.2.3.2. CREDENCIALES ADAFRUIT

Se configuran las credenciales del Servidor Adafruit IO. Como se vio anteriormente, se define la dirección del servidor, el puerto, el usuario y la llave para poder acceder a los feeds creados en el tablero a través del cliente (en este caso, el módulo ESP32).

6.2.3.3. CLIENTE Y FEEDS

Se crea el cliente para poder conectar con el servidor, éste cliente intentará acceder al servidor Adafruit IO por el protocolo MQTT, para lo que necesita que se le pase la clase 'client' creada por la librería correspondiente, y las credenciales de Adafruit configuradas antes.

A continuación, se configuran los feed para su publicación y subscripción. Siempre siguiendo el mismo formato de ruta, tal y como se comentó anteriormente en el apartado de MQTT. Aquí es donde se hará uso de las librerías citadas anteriormente para la comunicación MQTT con el servidor web de Adafruit.

6.2.3.4. VARIABLES GLOBALES

Aunque las variables creadas anteriormente también son globales, las distinguimos de las siguientes porque éstas modificaran su valor en algún momento dentro del codigo generado. Sin embargo, hay algunas que se incluyen aquí que no variarán, pero son necesarias para los cálculos dentro de algunas funciones, como por ejemplo la del nivel de agua del depósito**.**

Se distingues tres grupos o bloques de variables:

- **Control tiempo:**

Simplemente se han creado distintas variables que almacenarán los tiempos empleados, siempre en milisegundos, para el muestreo de datos y conteos de tiempo de publicación y subscripción al servidor.

- **Control WiFi**

Variables para almacenar las credenciales WiFi y contadores para tiempo de espera de conexión. Dependiendo de la red a la que se pretende acceder se necesitarán cambiar las credenciales WiFi. Se pueden obtener del módem que suministra nuestro portador de telefonía. O bien, se puede acceder a una conexión WiFi móvil, creando un punto de acceso personal desde dicho móvil y configurando los datos del mismo.

- **Sensores**

Son variables para almacenar los datos de los sensores. Estos datos podrán ser de tipo "float" (coma flotante), tipo "int" (entero) y tipo "bool" (booleano, guarda true o false). A todas ellas se les ha dado un valor inicial, y algunas de estas variables, como *VelSon* o *AlturaTotal,* son variables que se emplean para los cálculos pero que no se verán modificadas en ningún

momento en el programa. Por ejemplo, *VelSon* guarda el valor de la velocidad del sonido, y *AlturaTotal* guarda el valor de la altura total del depósito de agua.

- **Tipo String y char**

Estas variables tipo String (permite almacenar una cadena de caracteres) y tipo char (permite almacenar un caracter, se puede definir también como un vector de caracteres), se emplean para almacenar los datos generados por código en las funciones de humedad de tierra y luminosidad. Se crean también las variables para el modo vacaciones (Autonomo) y Manual, cuyo fin es guardar el dato de la subscripción de los botones de Adafruit y poder gestionar los modos con ellas. Finalmente se crea la variable *notaAlarma,* la cual guardará el mensaje de la Alarma generada en el invernadero que se publicará en el servidor de Adafruit IO.

6.2.4. INICIALIZACIÓN

En este bloque, se realiza la inicialización global del sistema cuando se conecta o se reinicia el mismo. Se implementa todo en la función *Setup* propia del IDE de Arduino para la configuración inicial de pines como entradas y salidas y otras inicializaciones como la del puerto *Serial* de visualización en pantalla.

Cabe destacar, que se han usado instrucciones *try-catch* para el control de excepciones. Estas excepciones servirán para evitar errores o fallos que se puedan producir en la ejecución del código. El bloque try contiene el código protegido que puede causar la excepción. Este bloque se ejecuta hasta que se produce una excepción o hasta completarse satisfactoriamente. La cláusula catch se puede utilizar sin argumentos, en cuyo caso capturará cualquier tipo de excepción general, o también puede aceptar un argumento de objeto (por ejemplo un tipo int), en cuyo caso tratará una excepción específica.

Para mayor entendimiento del código implementado en el *Setup*, se han diferenciado los distintos procesos que vemos a continuación:

- **LCD:** Se iniciará el display LCD y se intentará mandar el mensaje de bienvenida, si se produce algún error, se generará una excepción y se mostrará un error de inicialización y de escritura.
- **SERIAL:** Se iniciará la comunicación Serial.
- **PINES ENTRADA/SALIDA:** Se inicializa el sensor de temperatura y humedad y se definen los puertos de conexión de los sensores y actuadores como entradas o

salidas. En este caso, todos los sensores como entradas, excepto el pin de datos trigger que será de salida, y los actuadores como salidas.

- **CONEXIÓN WiFi:** Se llamará a la función *"configuraConexionesWifi()",* donde se intentará establecer conexión del módulo ESP32 con la red WiFi.
- **CANALES DE SUBSCRIPCIÓN:** En este punto se generan los canales de suscripción a los interruptores de Adafruit IO por MQTT. En este caso, se crean tres, uno para la selección del modo (*modoAuto*), otro para activación del riego (*riego*) y otro para la activación de la ventilación (*ventilacion*). Cada uno de ellos canalizará un paquete de datos. Por ejemplo, *modoAuto* recibirá o "A" o "M", y el *riego* recibirá "ON" o "OFF".
- **PRIMERA MEDIDA DE SENSORES:** Por control de excepciones *try-catch,* se hará la primera llamada a las funciones de los diversos sensores, obteniendo el dato de cada uno de ellos. Si se ha hecho una lectura correcta del sensor, se mostrará por pantalla LCD el mensaje de "OK" correspondiente. De haber algún error en alguno de ellos, se generará la excepción correspondiente y mostrará por pantalla el mensaje contrario "KO". Señalar que, el sensor de ultrasonidos tardará algo más en mostrar el dato por pantalla en esta primera toma de medidas, pues el sensor para obtener un dato más preciso, hace una media de las diez primeras medidas y cuando acaba, lo muestra.

6.2.5. FUNCIÓN PRINCIPAL

La función principal o en el caso de Arduino *"loop",* hay que definirla siempre, pues es la encargada de definir y dirigir el algoritmo que da solución al mismo. Define, como su nombre indica, la función principal del programa.

Para dar solución al sistema, se han implementado una serie de gestiones, cada una encargada de hacer algo específico. Se habla a continuación de cuáles son estas gestiones paso a paso.

6.2.5.1. LEER DATOS SENSORES

Lo primero que gestionará la función principal, es la toma de datos de los sensores incluidos en el proyecto. Es decir, tomará los valores de medida de cada sensor, uno por uno, y almacenará este valor en cada una de las variables globales definidas anteriormente. Cabe señalar, que se sigue empleando el control de excepciones, para controlar que no nos aporte ningún error como dato.

Estas medidas se recogen gracias al valor devuelto por las funciones secundarias creadas. Cada una de estas funciones secundarias tiene como objetivo obtener un valor del sensor al que hacen referencia.

6.2.5.2. CONFIGURACIÓN Y GESTIÓN DE ALARMAS

Lo siguiente será gestionar las alarmas. Como ya se ha comentado, solo dispondremos de un feed para publicar alarmas, por lo que se plantea hacer una gestión de las mismas de forma ordenada por prioridades, es decir, la Alarma 1 tendrá mayor prioridad pues entrará antes en el condicional, mientras que la Alarma 4 tengrá menos prioridad.

En el presente proyecto se mostrarán en el feed *Alarmas* de Adafruit los siguientes niveles de alarmas (colocadas por orden de mayor a menor prioridad):

- Alarma 0: No hay alertas. No se hará nada.
- Alarma 1: Nivel de agua en depósito bajo (<= 15 % de su capacidad).
- Alarma 2: Humedad de tierra, suelo seco. (Solo se activará cuando el modo de funcionamiento esté en Manual). Esto es porque en modo Autónomo ya se produce la gestión del riego de forma automática, mientras que de forma manual no, por ello se ha visto conveniente avisar al usuario para que active el riego.
- Alarma 3: Temperatura ambiente baja, peligro de helada (<= 5 ºC).
- Alarma 4: Temperatura ambiente alta, calor extremo (>= 45 ºC).

6.2.5.3. MOSTRAR ESTADO POR PANTALLA LCD

Lo siguiente será mostrar en el display LCD el modo de funcionamiento actual (Modo Autónomo o Manual) y el estado del invernadero (datos de humedad, temperatura…etc) en caso de no producirse ninguna alarma. Es decir, en la pantalla se podrá ver, en la primera

línea, el modo de funcionamiento, si está conectado a la red WiFi (se mostrará una “W” y si no, un guión “-“), y si el ventilador y/o el riego están activados (Solo para el modo Manual mostrará una “V” y una “R” en cada caso, y si están desactivados o en modo Autónomo, mostrará un guión “-“).

Lo primero que se hace en este punto, es llamar a la función *“getTextoModo()”.* Esta función devolverá el texto que se deberá mostrar en la primera línea del display según el modo en el que se encuentre el sistema en ese momento, así como para mostrar si está el WiFi, la ventilación y/o el riego activados.

A continuación, sólo si no se ha producido ninguna alarma (la variable Alarmas es igual a 0), entonces mostrará los datos obtenidos de los sensores uno a uno en la segunda línea del display. Estos datos se mostrarán secuencialmente y en el mismo orden gracias a la gestión de un menú. Por ejemplo, si el dato anterior mostrado es Humedad ambiente, entrará en el menú, y a continuación mostrará la Temperatura ambiente.

6.2.5.4. PUBLICACION DE DATOS

Llegado a este punto, lo primero antes de poder publicar ningún dato, es comprobar que se ha establecido la conexión WiFi correctamente, y la conexión con el servidor Adafruit IO por MQTT. Como ya hemos hecho anteriormente, todo se implementará dentro de un try-cacht como gestión de excepciones si se producen errores.

Una vez que está todo conectado, se publicarán los datos cada 20 segundos. Este tiempo se puede modificar a través de la variable *tiempoConexionServidor*. Tanto si se ha podido publicar o no algún dato concreto, se podrá comprobar únicamente por el puerto Serial de Arduino.

6.2.5.5. PUBLICACIÓN DE ALARMAS

En esta parte del código se hará la publicación de los mensajes de alarmas en el feed *Alarmas,* mientras se muestran también por la pantalla LCD.

Esta gestión se hará mediante un menú, donde se comparará la variable *Alarma* con los valores numéricos 1, 2, 3… Es decir, si Alarma = 1, entrará en el caso 1 del menú y mostrará

el mensaje de alarma correspondiente a la alarma 1, que en este caso es un mensaje que alertará al usuario de que el nivel de agua del depósito es muy bajo.

En pantalla LCD, ya no aparecerán ni el modo ni ningún otro dato, sino que mostrará la alerta en cuestión. También se podrá comprobar si se ha podido publicar o no la Alarma (como se hacía con las otras publicaciones) por el puerto Serial de Arduino.

6.2.5.6. SUBSCRIPCIONES

En esta parte es donde ocurre lo más interesante del programa, la subscripción a los botones del servidor. Por desgracia, no se ha encontrado la forma de conocer el estado de dichos botones de forma continuada, por lo que es necesario optimizar tiempos y decidir el tiempo en que va a estar activo el canal de subscripción para leer el estado de los interruptores. Este tiempo se podrá modificar en la variable *"tiempoConexionSuscripcion",* que por defecto está en 5 segundos.

Se crea un canal de subscripción que se comparará con cada uno de los canales de subscripción de los interruptores creados. Es decir, se entrará en un condicional, si el paquete de datos recibido ha llegado por la subscripción de uno de los interruptores, por ejemplo *modoAuto,* se convertirá el dato ("A" o "M") a tipo *String,* se mostrará por el puerto Serial y se guardará en la variable *"modo".* Además se mostrará por pantalla LCD el cambio de modo. Lo mismo ocurrirá con la activación de los actuadores. Tal y como se señaló en el apartado 5.2.3.3.

6.2.5.7. CONFIGURACIÓN DEL MODO VACACIONES

Finalmente, mediante condicionales, se configurará el modo de funcionamiento. Si por ejemplo el usuario se va de vacaciones unos días, podrá despreocuparse de su invernadero, ya que dispondrá del modo Automático para la autonomía del sistema. Este modo, como ya se ha visto, se puede configurar mediante el interruptor de selección de modo del Servidor Web de Adafruit IO.

Cabe señalar, que por defecto, el sistema estará funcionando en modo Automático, principalmente para asegurar que el invernadero esté activo a pesar de que falle la comunicación WiFi.

Esta parte del código es muy simple, se ha realizado con dos condicionales anidados. En el primer condicional se establece que, para el modo Automático (modo = “A”) se llame a la función secundaria *“modoAutomatico()”,* mientras que para el modo Manual (modo = “M”) se entrará en los siguientes dos condicionales. Cada uno de estos dos condicionales, evaluará si se ha activado o no el riego y la ventilación, y por tanto se activarán directamente los actuadores según corresponda.

6.2.6. FUNCIONES SECUNDARIAS

Se han creado una serie de funciones, cada una con un conjunto de instrucciones concreto para realizar una tarea específica. Algunas de ellas no devuelven nada (void) y otras devuelven parámetros de tipo *String, float o bool*. Estas funciones son creadas para ser llamadas por la función principal, aunque esto no implica que no se puedan llamar dentro de otras funciones.

A continuación se muestran las funciones secundarias creadas y se explica su función brevemente.

- **DHT – *void getDHT():*** Recoge los datos del sensor de humedad y temperatura DHT22. Si el valor leído no es un valor válido, dará un error de lectura.
- **Humedad tierra – *String HumedadTierra():*** Recoge los datos del higrómetro YL69. Si el valor leído no es un valor válido, dará un error de lectura. Cabe señalar que este valor va de 0 a 4095 puesto que se trata de una lectura analógica y el CAD del ESP32 es de 12 bits. En esta función se convierte dicha escala para proporcionar un % de humedad (valores de 0 a 100) y con ello, en vez de mostrar el porcentaje de humedad, se estable un suelo *“seco”, “ideal” o “húmedo”,* según los parámetros de diseño establecidos anteriormente.
- **Luminosidad – *String NivelLuminosidad():*** Recoge los datos del sensor de luminosidad o LDR. Si el valor leído no es un valor válido, dará un error de lectura. Ocurre lo mismo que con la función de humedad de tierra. Agí es donde se establecen los valores de luminosidad según los parámetros establecidos: luminosidad *“alta”, “media” o “baja”.*
- **Nivel de agua – *float NivelAguaDeposito():*** Recoge la medida efectuada por el sensor de ultrasonidos. Si el valor leído no es un valor válido, dará un error de lectura. En esta función se llamará a la siguiente: *“iniciarTrigger()”* para iniciar la medida del sensor. Con el dato recogido se irán haciendo una serie de cálculos

para obtener finalmente la altura actual de agua del depósito en cm. Y a continuación, se calculará el porcentaje de llenado del depósito, dato que se pasará para su posterior publicación y muestra local.

- **Iniciar trigger – *void iniciarTrigger():*** Método que inicia la secuencia del Trigger del sensor de ultrasonidos para comenzar a medir.
- **ph agua – *float pHAguaRiego():*** Recoge los datos del sensor de pH. Si el valor leído no es un valor válido, dará un error de lectura. En este caso hay que tener en cuenta que el sensor se ha calibrado previamente con dos muestras, por ejemplo PH7 y PH4. En caso de no calibrar el sensor, se obtendrá una medida no realista de pH. Este sensor se prevé que esté colocado en el depósito de agua, pero también se puede tomar una pequeña muestra del agua de riego para hacer la medida.
- **Modo automático – *void modoAutomatico():*** No devuelve nada puesto que simplemente se trata de la actuación del sistema en modo Automático. En esta función se llevan a cabo las tareas de riego y ventilación (actuaciones) según la lógica de control establecida. La lógica de control se establece por medio de condicionales anidados que evalúan los parámetros del sistema de medidas. Por un lado se tiene la actuación sobre el riego y por otro sobre la ventilación.
- **Conexión mqtt – *bool MQTT_connect():*** Función para conectarse y reconectarse según sea necesario al servidor de Adafruit.io a través del protocolo MQTT. Ejecutará hasta tres intentos de conexión que se harán cada 10 segundos. Se llamará en la función principal y tendrá cuidado si se conecta. Si se establece la conexión devuelve *true,* si no, *false*. También se mostrará por pantalla si se ha establecido correctamente o no la conexión.
- **Mostrar en ldc – *bool showInLCD(String lineaA, String lineaB):*** En esta función se pasarán dos datos. Estos datos, son los recogidos en las variables de tipo String *líneaA* (que se imprimirá en la primera línea de pantalla LCD) y *líneaB* (que se imprimirá en la segunda linea).
- **Conexión wifi – *void configuraConexionesWifi():*** Función para conectarse y reconectarse según sea necesario a la red WiFi. Cuenta hasta *ReintentosConexion* si no se puede conectar lo cancela. Si se establece la conexión devuelve *true,* si no, *false*. También se mostrará por pantalla si se ha establecido correctamente o no la conexión.
- **Texto modo – *String getTextoModo():*** Recoge el texto del modo de funcionamiento que se imprimirá por pantalla.

7. PUESTA EN MARCHA Y FUNCIONAMIENTO

En este punto se establecen las conexiones entre los distintos componentes de nuestro sistema, tomando como base la placa de desarrollo ESP32-DevKitC. Para mayor conocimiento de la misma, se muestra a continuación una imagen de la placa con la señalización de cada uno de sus puertos GPIO.

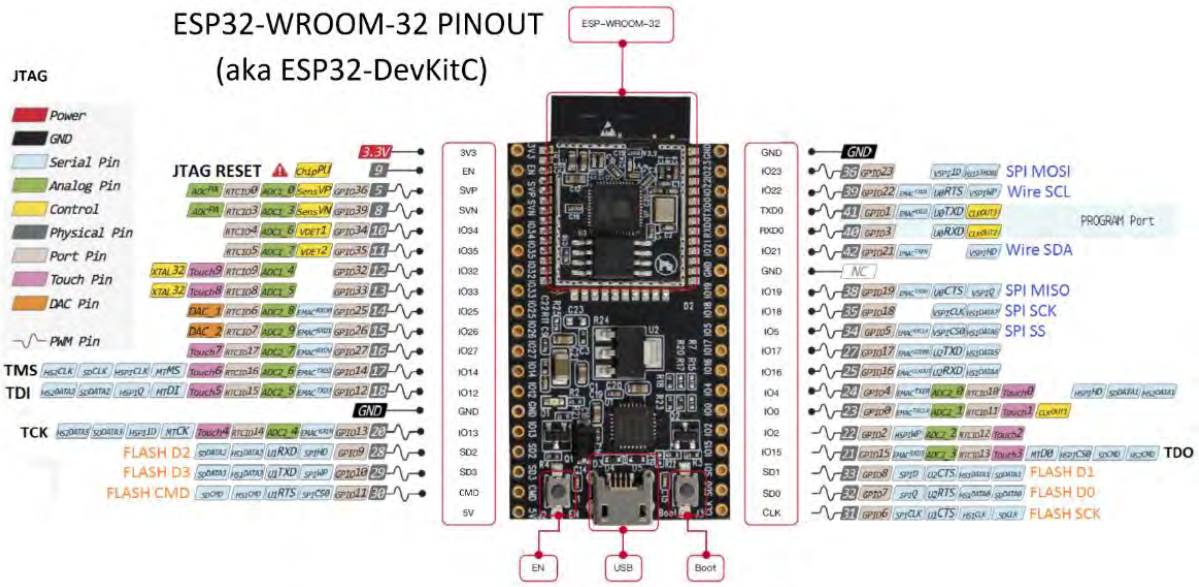


Ilustración 98 - Pines de configuración de la placa de desarrollo ESP32-DevKitC.

Señalar que, para la alimentación del sistema se ha usado un módulo de alimentación que proporciona 5V y 3.3V de corriente continua (DC). Sus características son las siguientes:

Ilustración 99 - Módulo de alimentación para protoboard.

- Interruptor de encendido
- Indicador LED
- Voltaje de entrada: 6.5 – 9 Vcc a través de conector Jack hembra
- Voltaje de salida: 3.3 – 5 V
- Máxima corriente de salida: 700 mA
- Control de rieles de salida independientes: 0, 3.3 o 5 V.

Por otro lado, con ésta tensión de salida no podemos alimentar la bomba de agua, ya que ésta trabaja a 12V de continua. Sin embargo el ventilador sí funciona a 5V, pero lo hace a unas revoluciones por minuto sensiblemente menores que cuando lo conectamos a 12V.

Una posible solución es implementar el siguiente módulo de alimentación:

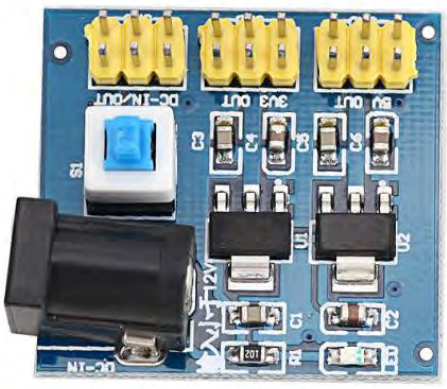


Ilustración 100 - Modulo de alimentación salida de tensión triple.

Este módulo ofrece una tensión de salida triple: los 3.3 V que necesita la placa de desarrollo ESP32-DevKitC, los 5V de funcionamiento que necesitan el resto de sensores y la pantalla LCD, y los 12 V que necesitan los actuadores (bomba de agua y ventilador). La tensión de salida del tercer canal en realidad es la misma que se proporciona a la entrada, es decir, si se alimenta el módulo con 9V, ese tercer canal proporcionará una salida de 9V. Por esta razón se necesitaría un convertidor de tensión AC – DC que proporcione 12 V de continua.

Finalmente, se muestra una lista de conexionado de sensores y actuadores a los puertos concretos usados.

- Sensor DHT22 – GPIO 23 / SPI MOSI / Empleado como entrada digital
- Sensor YL69 – GPIO 36 / ADC1-0 / Empleado como entrada analógica
- Fotorresistencia LDR – GPIO 39 / ADC1-3 / Empleado como entrada analógica
- Sensor PH – GPIO 34 / ADC1-6 / Empleado como entrada analógica
- Ultrasonidos – GPIO 17 / TXDu2 / Empleado como salida digital para Trigger
- Ultrasonidos – GPIO 19 / SPI MISO / Empleado como entrada digital para Echo
- Relé 1 – GPIO 27 / Empleado como salida digital para activar ventilador
- Relé 2 – GPIO 14 / Empleado como salida digital para activar bomba de agua
- Buzzer – GPIO 12 / Empleado como salida digital para activar alarma
- Display LCD I2C – GPIO 21 / Wire SDA
- Display LCD I2C – GPIO 22 / Wire SCL

Una vez conectado todo, se plantea construir un pequeño invernadero o semillero de unos 0.5 m^3, donde plantaremos unas semillas de tomate. Lo siguiente será elegir el depósito de agua, y modificar la altura total de depósito en el código. Luego se colocarán los sensores y actuadores en su lugar correspondiente. De la bomba de agua sacaremos un tubo flexible de PVC de unos 5 o 6 mm de diámetro, que será el encargado de contener el agua impulsada por la bomba y que llegará a la tierra del invernadero/semillero.

Lo siguiente será cargar el programa en la placa ESP32-DevKitC. Para ello hay que tener en cuenta que se necesita presionar el botón “boot” de la misma para que entre en modo programación. Esto se hará siempre que se quiera cargar un nuevo programa en la placa.

Finalmente se alimenta el circuito montado y ya se podrá ver el funcionamiento en vivo. Lo primero que se verá es la inicialización del sistema, seguido de la toma de medidas. Luego podremos ver los datos obtenidos tanto en el Servidor de Adafruit IO como en la aplicación del móvil. Con esto, ya podemos controlar nuestro Invernadero, no solo de forma manual, sino de forma remota, gracias a la tecnología IoT.

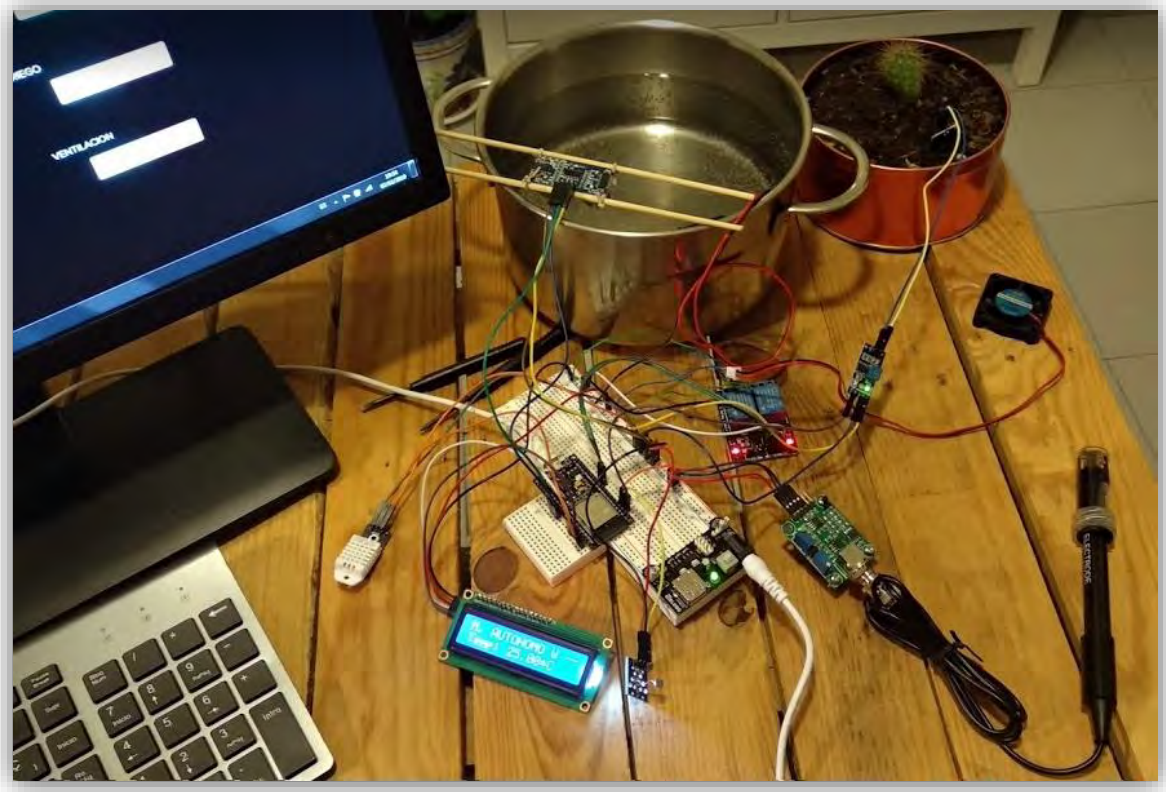


Ilustración 101 - Montaje y puesta en marcha.

8. CONCLUSIONES Y PROXIMOS PASOS

El actual proyecto muestra un prototipo completo, y por tanto se puede concluir que se han alcanzado muchos de los objetivos que se pretendían al comienzo del mismo. Sin embargo, hay aún mucho trabajo por hacer para conseguir un producto que pueda estar en el mercado, empezando por ejemplo, con el diseño de una placa industrial profesional.

Actualmente, cualquier modificación tanto en parámetros y límites de cultivos, como en credenciales WiFi o del Servidor Broker, deben hacerse directamente desde el código, teniendo que cargar de nuevo el programa en la placa ESP32-DevKitC. Esto puede generar nuevos errores y además, algunas de estas modificaciones (como los límites de temperatura), se deben incluir en el Servidor y en la App. Esto se podría mejorar creando, por ejemplo, una aplicación móvil donde se pueda por un lado, aportar las credenciales WiFi y las del Servidor y guardarlas para poder modificar el código externamente. Por otro lado, en dicha aplicación se podría elegir el tipo de cultivo que se desea plantar en el invernadero o semillero, es decir, que el algoritmo de control Automático sea capaz de gestionar diferentes cultivos en base a sus parámetros característicos, ya que actualmente solo se podrían plantar tomates.

Esto último se podría probar generando una base de datos o una biblioteca en la se añadan diferentes tipos de cultivo con sus parámetros y límites característicos. De ésta forma, los parámetros de control del invernadero, independientemente del cultivo que sea, puedan ser modificados de forma externa, sin necesidad de tocar el código del programa.

A continuación, se detallan los puntos considerados como ampliaciones y mejoras, y los cuales por falta de tiempo y/o desconocimiento en la materia no se han llegado a implantar o modificar.

- Mejora del algoritmo automático. Acceso a diferentes parámetros y límites de control según el tipo de cultivo a incluir en el invernadero.
- Servidor dedicado con página web donde se pueda configurar de forma remota el sistema. Configurar credenciales WiFi, credenciales Servidor Broker, selección de parámetros y límites de los cultivos o directamente seleccionar el cultivo deseado, etc.
- Mejora en la gestión de alertas, en vez de establecer prioridades en las alertas, suponer que todas son importantes, y por tanto implementar otro tipo de gestión de las mismas.
- Incorporación de una cámara. Existe la posibilidad de incluir una cámara web, donde se pueda ver el invernadero en tiempo real en una APP o en una página

Web propia. Cabe destacar que, se intentó usar una cámara *OV7670* compatible con Arduino, sin embargo, no se llegó a implementar.

- Diseño de una placa industrial profesional que permita hacer el sistema compacto, de modo que el prototipo pueda presentarse a posibles clientes y analizar la viabilidad de un posible plan de negocio para comercializar el producto final.
- Pasar el proyecto a HTTP de tal forma que se emplee el protocolo REST [54]. Comúnmente utilizado en empresas para la creación de sistemas conectados.

9. BIBLIOGRAFÍA

[1] Adrian McEwwn y Hakim Cassimally, "*Internet de las Cosas*", Edición, Anaya Multimedia, 2014.

[2] Anónimo, Wikipedia (2019, Enero 21). Invernadero. [Online] Disponible en: https://es.wikipedia.org/wiki/Invernadero

[3] Eduardo Barrera Martín, Rafael Victor Herrero Niño y Alejandro Raúl Meraz García, "Invernadero Inteligente" Tesis, Instituto Politécnico Nacional, México D.F., 2014.

[4] Infoagro. Cambios climáticos en invernaderos [Online] Disponible en: http://www.infoagro.com/industria_auxiliar/control_climatico.html

[5] Yuan Li, Wenquan Niu, Miles Dyck, Jingwei Wang y Xiaoyang Zoul (2016, Diciembre 20). Rendimientos y nutrición del tomate de invernadero en respuesta a diferentes volúmenes de aireación del suelo a dos profundidades del riego por goteo subsuperficial. [Artículo científico. Online] Disponible en: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5171848/

[6] Dietmar Schwarz, Andrew J. Thompson y Hans-Peter Kläring (2014, Noviembre 14). Pautas para utilizar tomate en experimentos con un ambiente controlado. [Artículo científico. Online] Disponible en: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4235429/

[7] Guang-cheng Shao, Ming-hui Wang, Na Liu, Min Yuan, Prem Kumar, y Dong-Li She. (2014, Junio 25). Índice de crecimiento y calidad integral del tomate en refugios contra la lluvia en respuesta a diferentes tratamientos de irrigación y drenaje. [Artículo científico. Online] Disponible en: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4098618/

[8] Zhenhua Wei, Taisheng Du, Xiangnan Li, Liang Fang y Fulai Liu (2018, Marzo 27). Efectos interactivos de la fertilización elevada de CO2 y N en el rendimiento y la calidad del tomate cultivado bajo regímenes de riego reducidos. [Artículo científico. Online] Disponible en: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5880949/

[9] Yuling Bai y Pim Lindhout (2017, Agosto 27). La domesticación y la cría de tomates: ¿Qué hemos ganado y qué podemos ganar en el futuro? [Artículo científico. Online] Disponible en: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2759208/

[10] Agroware. Software agrícola (2016, Noviembre 3). Tipos de riego y sus ventajas: ¿cuál es el adecuado? [Online] Disponible en: http://sistemaagricola.com.mx/blog/tipos-de-riego-en-la-agricultura-y-ventajas/

[11] Gruposacsa (2016, Enero 21). Sistemas de riego para invernaderos. [Online] Disponible en: http://www.gruposacsa.com.mx/1050-2/

[12] Siber. Ventilación inteligente. El sistema de ventilación de un invernadero: ventilación natural y mecánica [Online] Disponible en: https://www.siberzone.es/blog-sistemas-ventilacion/el-sistema-de-ventilacion-de-un-invernadero-ventilacion-natural-y-mecanica/

[13] Shimon Valensi (2011, Noviembre 19). Ventilación en invernaderos. [Online] Disponible en: https://azrom.com/technology-highlights/ventilation-in-greenhouses/

[14] Luis Llamas (2018, Abril 1) Esp32, El "Hermano Mayor" Del Esp8266 Con Wifi Y Bluetooth [Online] Disponible en: https://www.luisllamas.es/esp32/

[15] Espressif. ESP32 Una potencia y rendimiento de IoT diferente. [Online] Disponible en: https://www.espressif.com/en/products/hardware/esp32/overview

[16] Oscar González (2017, Febrero 2). Comparativa y análisis completo de los módulos Wifi ESP8266 y ESP32. [Online] Disponible en: https://blog.bricogeek.com/noticias/electronica/comparativa-y-analisis-completo-de-los-modulos-wifi-esp8266-y-esp32/

[17] Juan Manuel Hernández (2018, Julio 24). Sensor de temperature y humedad DHT11 – DHT22. [Online] Disponible en: http://www.omniblug.com/sensor-temperatura-humedad-DHT11-DHT22.html

[18] Alejandro Gálvez, "Desarrollo e implementación de una estación meteorológica mediante la plataforma Hardware/Software Libre Raspberry Pi", Proyecto fin de Carrera, Universidad Miguel Hernández de Elche, Elche, 2015 [Online] Disponible en: https://www.slideshare.net/alejandrogalvez7/

[19] Talos electronics. Sensor de humedad del suelo YL38 y YL69. [Online] Disponible en: https://www.taloselectronics.com/products/sensor-de-humedad-del-suelo-yl38-y-yl69

[20] Lucas Martín (2016, Enero 19). Sensor de humedad de suelo SY-69. [Online] Disponible en: http://www.niplesoft.net/blog/2016/01/19/sensor-de-humedad-de-suelo-yl-69/

[21] Victor Ventura (2016, Abril 17). LRD: Medir la luz con una fotorresistencia [Online] Disponible en: https://polaridad.es/ldr-fotorresistencia-luz-luminosidad-medir-medicion-arduino/

[22] Bricogeek. Sensor análogico de PH Básico. [Online] Disponible en: https://tienda.bricogeek.com/home/581-sensor-analogico-de-ph.html

[23] Elentronilab. Ingeniería y diseño electrónico. Sensor analógico de PH 0-14. [Online] Disponible en: https://electronilab.co/tienda/sensor-analogico-de-ph-de-0-14/

[24] Scidle. Ciencia y tecnología (2017, Marzo 10). Como unar un sensor de ph con arduino. [Online] Disponible en: https://scidle.com/es/como-usar-un-sensor-de-ph-con-arduino/

[25] Infootec. Sensor ultrasonidos HC-SR04. [Online] Disponible en: https://www.infootec.net/sensor-ultrasonidos-hc-sr04/

[26] Ledge, Amazon (2018, Mayo 17). Mini Bomba de Agua Ultra Silencioso 240L/H Bomba Sumergible 4.8W Bomba de Circulación Para Pecera Acuario Jardín, Estanque, Fuente [Online] Disponible en: https://www.amazon.es/gp/product/B07D26TBL7/ref=ox_sc_act_title_2?smid=A2C7VEG0XNNBRK&psc=1

[27] Banggood. Ventilador de 30mm de 12V DC Para Impresora 3D RAMPS Electrónicas / Extrusora - RepRap Prusa [Online] Disponible en: https://www.banggood.com/es/12V-DC-30mm-Cooling-Fan-For-3D-Printer-RAMPS-Electronics-Extruder-RepRap-Prusa-p-1010340.html?cur_warehouse=CN

[28] NayLamp. Tutorial LCD con I2C, controla un LCD con solo dos pines [Online] Disponible en: https://www.naylampmechatronics.com/blog/35_Tutorial--LCD-con-I2C-controla-un-LCD-con-so.html

[29] Greek Factory (2017, Mayo 2017). LCD 16×2 por I2C con Arduino usando solo dos pines [Online] Disponible en: https://www.geekfactory.mx/tutoriales/tutoriales-arduino/lcd-16x2-por-i2c-con-arduino/

[30] Luis Llamas (2016, Mayo 22). Conectar un display LCD Hitachi a arduino por Bus I2c [Online] Disponible en: https://www.luisllamas.es/arduino-lcd-i2c/

[31] Taller Arduino (2010, Octubre 8). Módulo de relés. [Online] Disponible en: https://tallerarduino.com/2012/10/08/modulo-de-reles/

[32] Enrique. Educachip (2014, Septiembre 22) Relé Arduino – Cómo controlar tu casa con arduino. [Online] Disponible en: http://www.educachip.com/arduino-rele-5v/

[33] Luis Llamas (2016, Julio 23). Manejar cargas de más de 220v con arduino y salida por relé. [Online] Disponible en: https://www.luisllamas.es/arduino-salida-rele/

[34] Raúl Vega Marcos, “Estudio comparativo de disitintas tecnologías fotovoltaicas” Proyecto fin de Carrera, Universidad Carlos Tercero, Madrid, 2013. [Online] Disponible en: https://orff.uc3m.es/bitstream/handle/10016/19818/TFG_Raul_Vega_Marcos.pdf?sequence=1&isAllowed=y

[35] Francisco Fernández González (2015, Agosto 24). Cómo calcular una instalación solar fotovoltaica en 5 pasos. [Online] Disponible en: https://clickrenovables.com/blog/como-calcular-una-instalacion-solar-fotovoltaica-en-5-pasos/

[36] Javier María Méndez Muñiz, Rafael Cuervo García y ECA Instituto de tecnología y formación, “*Energía solar fotovoltaica*“, 3ªEdición, Madrid, FC Editorial, 2008.

[37] JRC Comisión internacional. Sistema de Información Geográfica Fotovoltaica - Mapas Interactivos. [Online] Disponible en: http://re.jrc.ec.europa.eu/pvgis/apps4/pvest.php

[38] Electrónica embajadores. Solarpower. Series Xunzel. [Datasheet. Online] Disponible en: https://www.electronicaembajadores.com/Datos/pdf1/sa/sa41/sa41010-14.pdf

[39] Iván Uriarte (2018, Diciembre). Instalando el ESP32. El nuevo chip de Espressif [Online] Disponible en: https://www.prometec.net/instalando-esp32/

[40] Informática para tu negocio. Diferencia entre una IP estática y una IP dinámica [Online] Disponible en: https://www.informaticaparatunegocio.com/blog/diferencia-entre-una-ip-estatica-y-una-ip-dinamica/

[41] MQTT. Preguntas frecuentes. [Online] Disponible en: http://mqtt.org/faq

[42] Aprendiendo Arduino (2018, Noviembre 19). MQTT. [Online] Disponible en: https://aprendiendoarduino.wordpress.com/tag/protocolos/

[43] Aprendiendo Arduino (2017, Marzo 31). Plataformas IoT. [Online] Disponible en: https://aprendiendoarduino.wordpress.com/2017/03/31/plataformas-iot/

[44] Steve. (2019, Enero 29). MQTT Brokers / Servidores y Guía de Alojamiento. [Online] Disponible en: http://www.steves-internet-guide.com/mqtt-hosting-brokers-and-servers/

[45] Justin Cooper (2015, Enero 22). Arduino. [Online] Disponible en: https://learn.adafruit.com/adafruit-io/arduino

[46] Justin Cooper (2015, Enero 22). MQTT Api. [Online] Disponible en: https://learn.adafruit.com/adafruit-io/mqtt-api

[47] Adafruit IO REST API (v2.0.0). [Online] Disponible en: https://io.adafruit.com/api/docs/#!/v2

[48] Routix Software (2017, Marzo 2). MQTT Dash (IoT, Smart Home) [Aplicación móvil. Online] Disponible en: https://play.google.com/store/apps/details?id=net.routix.mqttdash&hl=es_419

[49] Anónimo, Wikipedia (2019, Enero 31). MQTT. [Online] Disponible en: https://en.wikipedia.org/wiki/MQTT

[50] Elias Notario (2013, Octubre 1). Guía IFTTT: qué es, cómo funciona y 15 recetas útiles. [Online] Disponible en: https://www.ticbeat.com/tecnologias/guia-ifttt-que-es-como-funciona-15-recetas-utiles/

[51] IFTTT. [Online] Disponible en: https://ifttt.com/services

[52] Gloria Sedano. ¿Qué es IFTTT y cómo usarlo? [Online] Disponible en: https://www.webespacio.com/ifttt/

[53] Kevin Sharp (2015, Septiembre 16). IoT 101: Networks [Online] Disponible en: https://www.artik.io/blog/2015/09/iot-101-networks/

DOCUMENTO Nº 2:

ANEXOS

ANEXO 1: PROCESO DE DISEÑO DE LA PCB

1. Base del proyecto

El proyecto nace de la idea de construir un invernadero domótico, el cual recibirá y procesará datos de diferentes sensores y actuará según la lógica de control ajustada. Además, se pretende comunicar por wifi con los dispositivos colocados en el invernadero, a través de una aplicación móvil o por página web. La idea original suponía emplear un Arduino y un módulo ESP8266 para las comunicaciones, por ello, la PCB diseñada incluye la placa base de Arduino Mega 2560, conectores para los sensores exteriores del invernadero, buzzer para alarma, leds de indicación de estado de los sensores y un zócalo para colocar el módulo wifi basado en ESP8266 (NodeMCU).

Los sensores exteriores empleados son:

- DHT22: Sensor de temperatura y humedad ambiente.
- YL69: Sensor de humedad de tierra.
- LDR: Sensor de luminosidad.
- Ultrasonidos: Para medir el nivel de agua de un depósito.

Para indicar el estado del invernadero, según los sensores empleados, se tienen:

- 4 leds para DHT22: Dos led azules y dos led rojos (Los led azules indican temperatura y humedad por debajo de los niveles ideales establecidos y los led rojos indican temperatura y humedad por encima de los niveles ideales establecidos). Puede estar el led azul de temperatura encendido y el resto de leds apagados, lo que indicaría que la temperatura es demasiado baja pero los niveles de humedad son ideales.
- 2 leds para YL69: Led verde para indicar que el suelo está encharcado (exceso de agua) y por tanto se debe drenar la tierra. Led rojo para indicar que el suelo está seco y se debe regar. Si ambos están apagados, indica que la humedad de tierra está dentro de los niveles ideales.
- 3 leds para LDR: Se encenderán según el nivel de luminosidad. Un led amarillo encendido indica baja luminosidad y los 3 encendidos luminosidad alta.

También se dispondrá de una alarma para alertar si tenemos que llenar el tanque de agua para el riego, para ello se añade un buzzer a la PCB.

Aunque lo ideal habría sido añadir el diseño del módulo ESP8266 a la PCB, en lugar de eso se añade un zócalo para colocar la placa NodeMCU en la PCB. Creamos eso sí, las rutas de conexión al puerto serial del Arduino con el puerto serial del módulo wifi.

2. Librería de componentes

El programa que se emplea para crear la placa es '*Design Spark PCB'*, el cual es bastante intuitivo y fácil de usar. Lo primero que se ha hecho es crear la librería de componentes, centrándose principalmente en buscar los componentes de la librería de Arduino Mega 2560.

Los componentes están formados por dos partes, '*Schematic Symbol*' y '*PCB Symbol (footprint)*'. Algunos de los componentes como resistencias y condensadores smd se han modificado de la librería de Desing Spark, escogiéndoles por el tamaño. Hemos intentado que fuesen todos del mismo tamaño (smd 1206) pero no siempre se encontraba el componente original y se buscaba uno con características similares de otro tamaño.

Para buscar los componentes hay que fijarse en el '*RS Part Number*' y en la página oficial de RS (distribuidor RS) buscar el componente con ese Part Number. Si no se encuentra el componente se puede buscar otro de características similares para implementarle en el proyecto.

Aunque se puede diseñar desde cero y modificar la librería de Design Spark para obtener los componentes del proyecto, se puede emplear la herramienta de búsqueda de componentes '*Library Loader*' para buscar (con su MPN, *Manufacturer Part Number*) y descargar el componente necesario ya con su símbolo de esquemático y su símbolo footprint.

Una vez que tenemos los componentes necesarios de la placa de arduino, se busca en RS los conectores para los sensores, basándose en el numero de pines de conexión de dichos sensores. El sensor DHT necesitará un conector de 3 terminales, el ultrasonido un conector de 4 terminales, el YL69 un conector de 2 terminales (cabe destacar, que este sensor tiene una pequeña placa pcb que se añade directamente a la diseñada, y la LDR y el DHT necesitan una resistencia de pull up que también se añadirá), la LDR precisa un conector de 2 terminales.

Además de los conectores, se busca un buzzer, los componentes de la placa para el sensor YL69, dos conectores hembra, de 15 pines para formar el zócalo del módulo wifi y finalmente los taladros de sujeción de la placa.

3. Esquemáticos

Una vez que se tiene la librería de componentes completa, se pasa a realizar el esquemático de la PCB. Se separan en dos hojas el circuito esquemático del arduino y en la tercera hoja el esquemático relativo a los sensores.

Se nombran las net que van conectadas al microcontrolador, de ésta forma son fácilmente localizables, y se sabrá en todo momento, qué conecta cada una de ellas.

Antes de pasar el diseño a la PCB, es importante comprobar que está bien, fijándose en que no falta nada. Analizando las nets y las conexiones.

Para ver el esquemático completo se han creado 3 PDFs, uno por cada hoja. Con el nombre *'proyecto1-HojaX'*. Siendo X el número de hoja.

4. Distribución de componentes en la PCB

Una vez que se tiene el esquemático completo y comprobado, se pasa a la PCB. Se comprueba que el esquemático y la PCB son iguales con la herramienta '*Schematic/PCB Check…*', es decir, se analiza que tenemos los mismos componentes del esquemático que se va a utilizar en la PCB. Se da un tamaño provisional a la placa, y se colocan los componentes de la mejor forma posible, de manera que queden bien distribuidos con espacio para las rutas y vías, y de forma que los componentes que estén conectados, estén lo más cerca posible, para que las rutas no sean demasiado largas.

Hay que cumplir con el espaciado mínimo entre componentes y la placa. Si hay errores, se pueden ir corrigiendo poco a poco gracias al DRC online, que canta los errores según se está modificando la placa.

La distribución propuesta en la placa es la siguiente:

- Centrado en la placa, y cerca por tanto de todos los componentes asociados a él con nets, está el microcontrolador.
- A la izquierda de la placa, los conectores USB y POWER del arduino.
- En la parte superior y hacia la derecha, los conectores de los sensores.
- A la derecha del arduino el modulo del sensor YL69.
- En la parte inferior los leds que indican el estado de los sensores.
- A la derecha del todo el zócalo del módulo NodeMCU, separado de pin a pin la distancia de 28 mm.

Una vez colocados los componentes en la placa de forma ordenada, se pasa a diseñar las pistas que se usaran para enrutar.

5. Ruteado de pistas

Para las pistas de tensión se ha dimensionado un ancho nominal de pista de 0.6 mm y un mínimo de 0.3mm. Este ancho mínimo le define principalmente componente que tiene los pad más restrictivos (más pequeños). Para el dimensionado de las pistas se usa la herramienta Design Calculators, donde podemos definir la corriente máxima que atravesará las pistas de tensión y ésta herramienta nos proporcionará el ancho mínimo de pista que necesitamos.

Para las pistas de señal, se han usado rutas de 0.2 mm de ancho. Por otro lado se tiene en cuenta que es mejor que las esquinas de las rutas sean a 45º y no a 90º, para evitar los giros bruscos y pérdidas de señal.

Se pasa a generar las rutas de las pistas de señal, cambiando de capa cuando es necesario y respetando las distancias mínimas entre pistas, pads y componentes. Se comprueba a menudo que no tenemos errores pasando el DRC, y modificando el diseño cuando sea necesario en caso de que les hubiera.

Para los pad y pistas que quedan de tensión Vcc, lo que se hace es conectar con una vía directamente a la capa de tensión Vcc.

Para los pad y pistas que nos quedan de tierra GND, o 0V, lo que se hace es dibujar un área de cobre en la capa superior TOP y conectada a la capa de tensión 0V. De esta forma las conexiones a 0V se unen a éste área. También se pueden hacer las conexiones con vías a la capa GND, igual que se hizo con la capa Vcc.

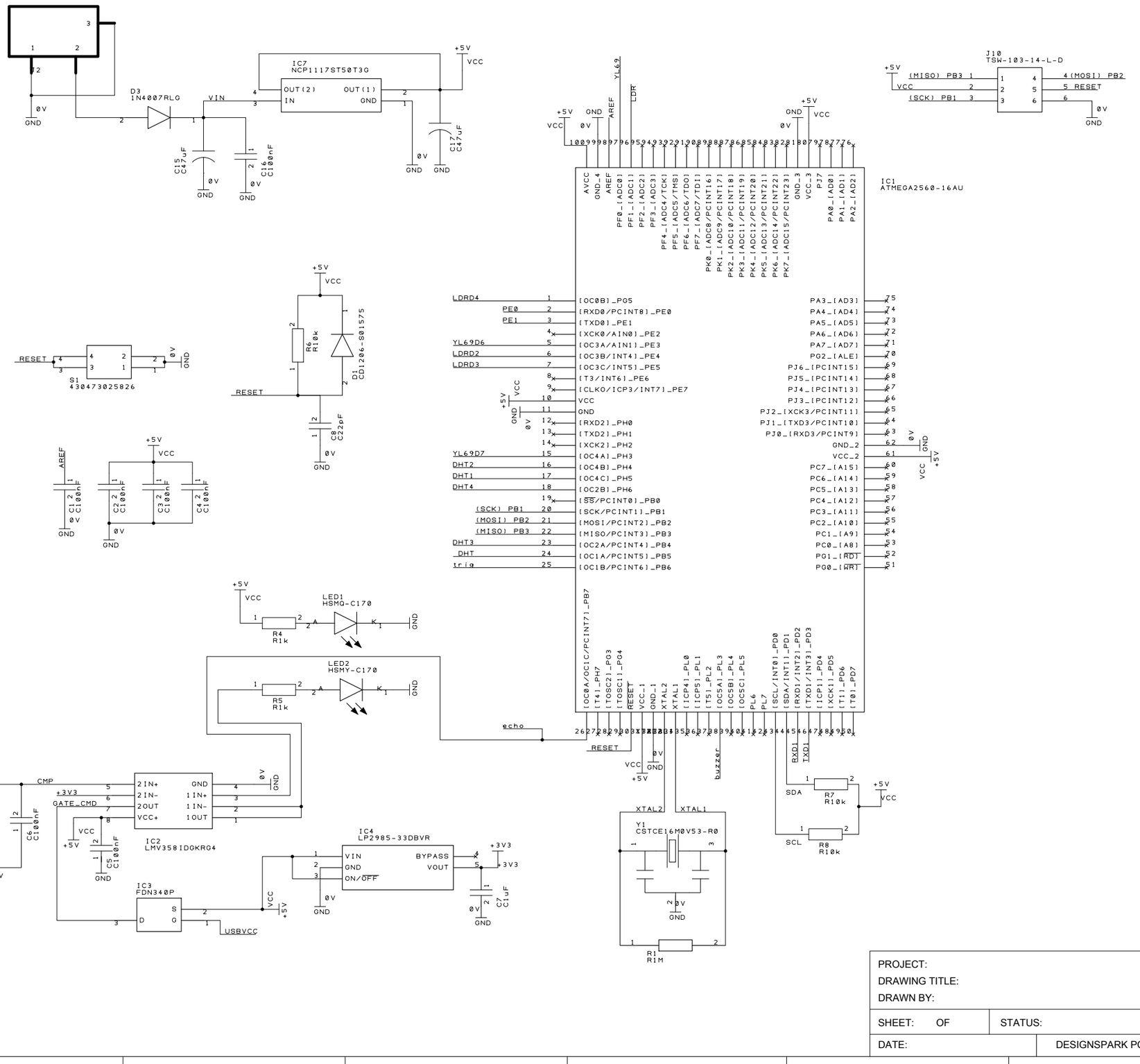
Finalmente se vuelve a comprobar que no hay errores DRC, y si los hay, se resolverán uno a uno. Luego se genera la lista de materiales, colocados los componentes por nombre y en la primera columna, y en las columnas adyacentes, la cantidad, el valor del componente, el distribuidor, el MPN (Manufactured Part Number) y el RS Part Number entre otras descripciones, aunque las nombradas son las más importantes que deben aparecer en la lista de materiales.

6. Generación de ficheros Gerber

Una vez acabada la PCB se generan los ficheros Gerber que se enviarán al fabricante. Los ficheros que a generar siempre son: planos superior e inferior (TOP y BOTTOM), planos de tensión y tierra si les hubiera, como es el caso (VCC y GND), capa de serigrafía, de la capa superior Top (SILKSCREEN-TOP), máscara de soldadura de las capas Top y Bottom (SOLDERMASK-TOP,BOTTOM) y los taladros (DRILL).

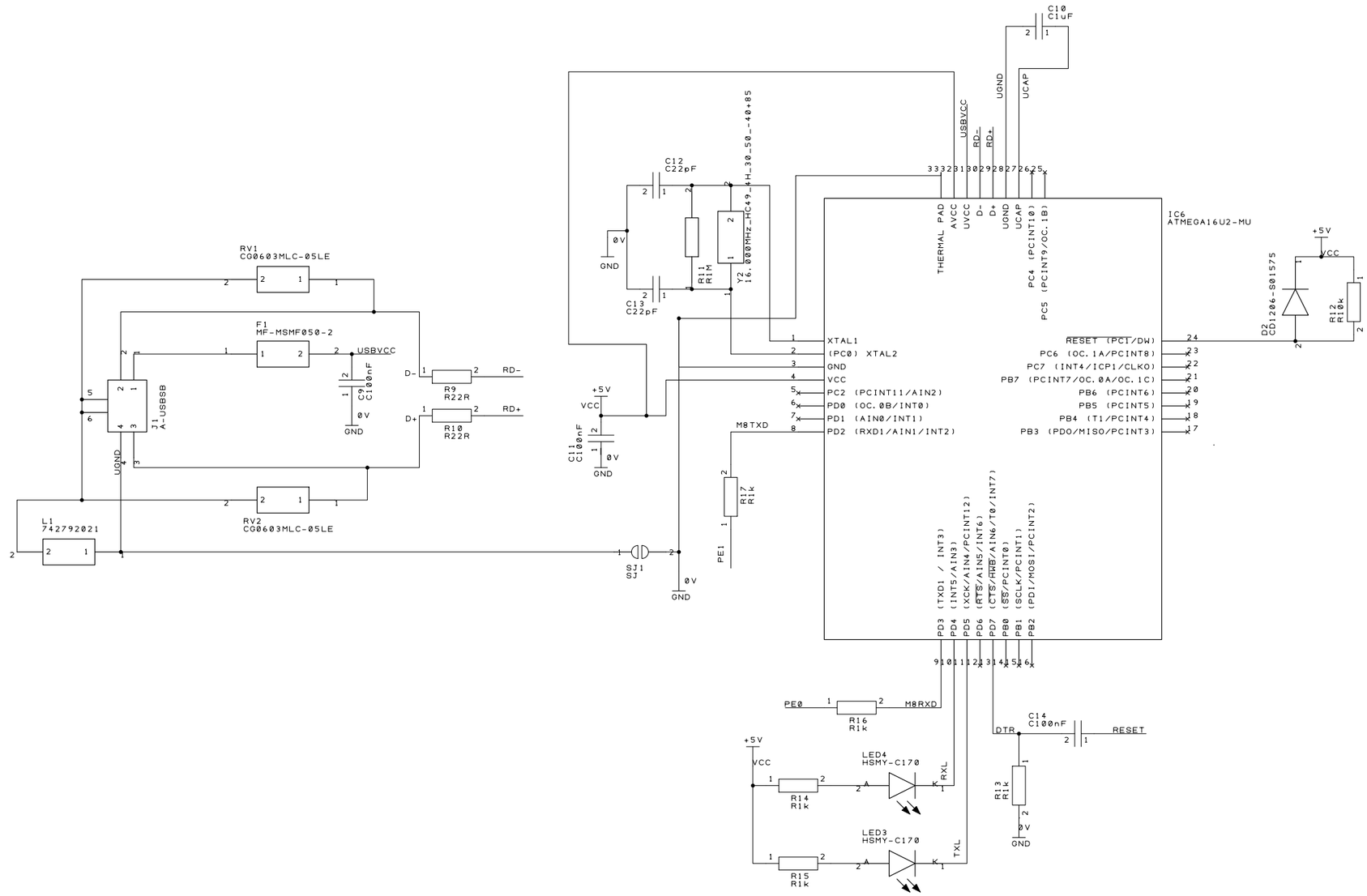
Se generan todos los ficheros, y se comprueba que están bien gracias al visor de Gerbers online, 'www.gerber-viewer.com'. Finalmente se pueden generar estos ficheros de documentación en PDF.

ANEXO 2: ESQUEMÁTICO PCB Estudio Previo – HOJA 1



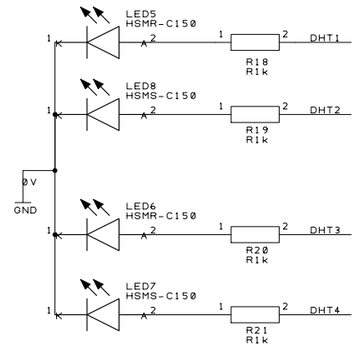
PROJECT:		
DRAWING TITLE:		
DRAWN BY:	REVISION:	
SHEET: OF	STATUS:	
DATE:	DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB	

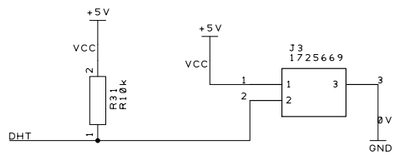
ANEXO 3: ESQUEMÁTICO PCB Estudio Previo – HOJA 2

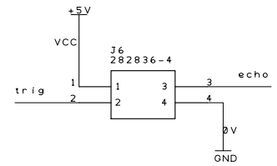


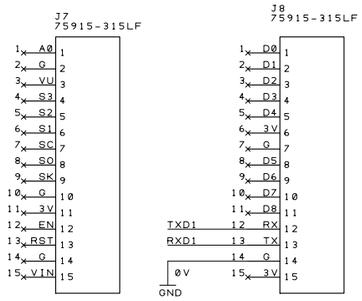
PROJECT:		
DRAWING TITLE:		
DRAWN BY:	REVISION:	
SHEET: OF	STATUS:	
DATE:	DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB	

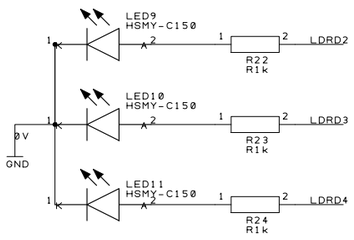
ANEXO 4: ESQUEMÁTICO PCB Estudio Previo – HOJA 3

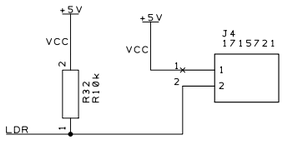


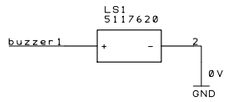


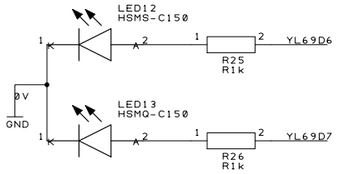


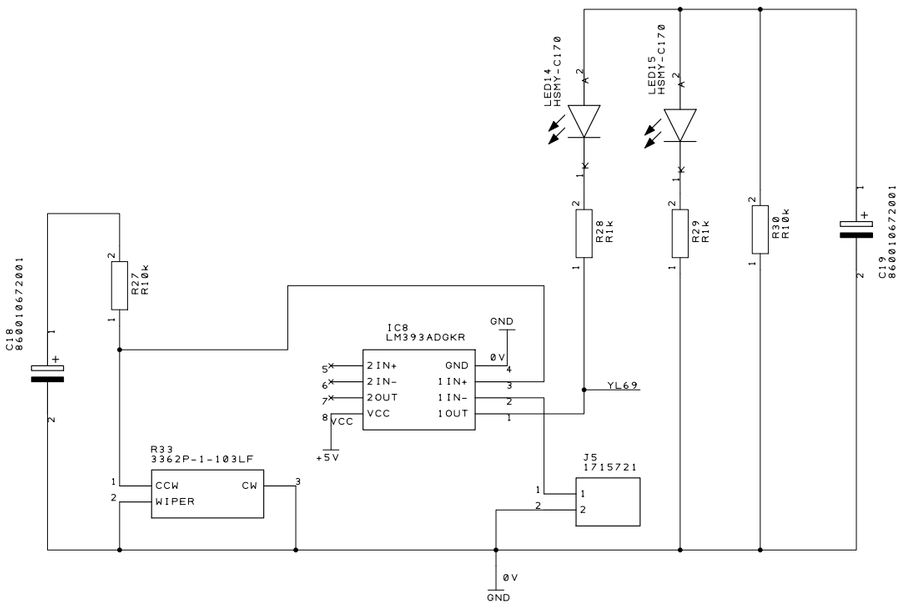


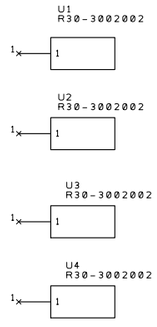






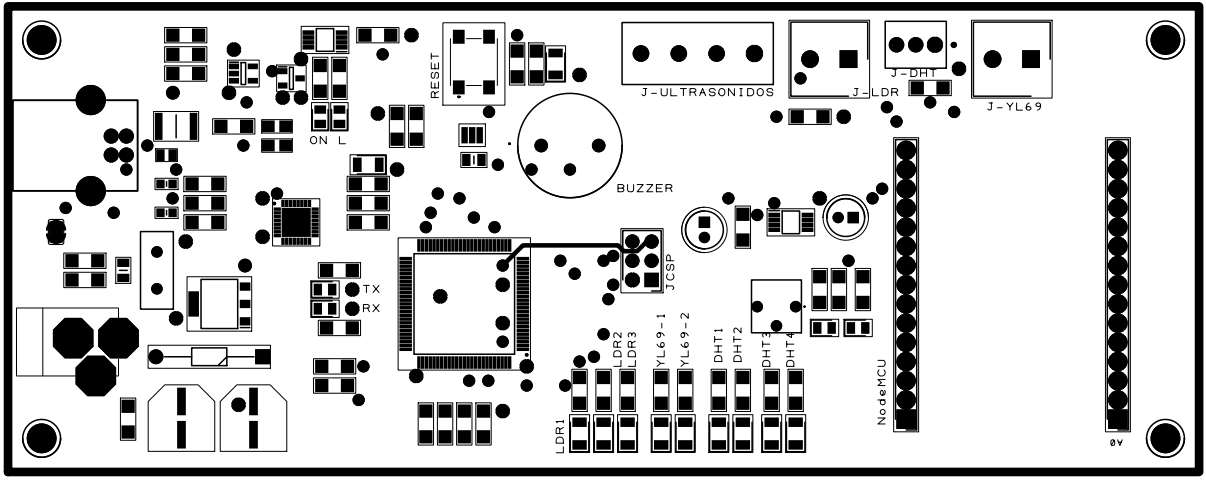


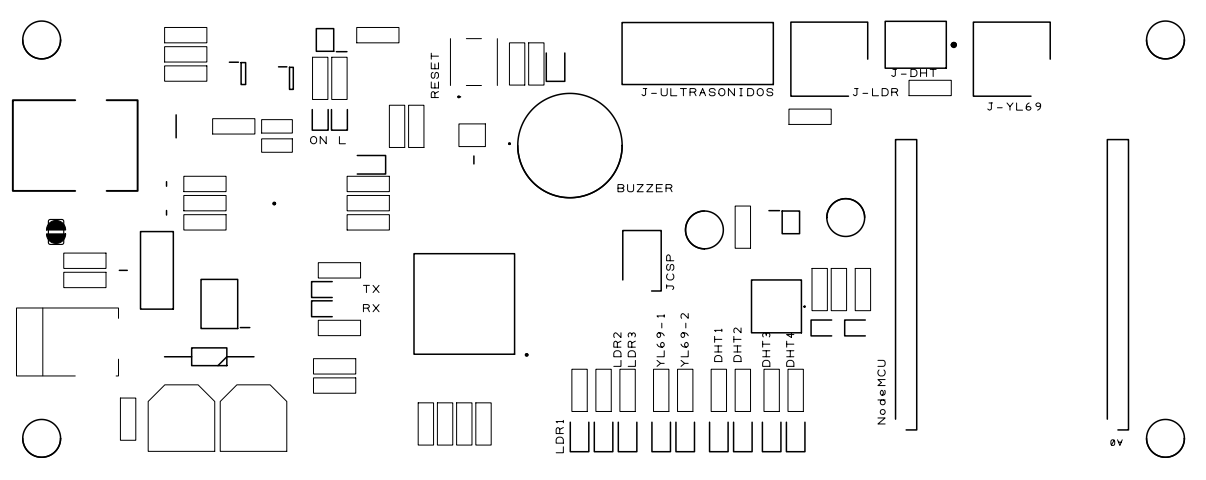


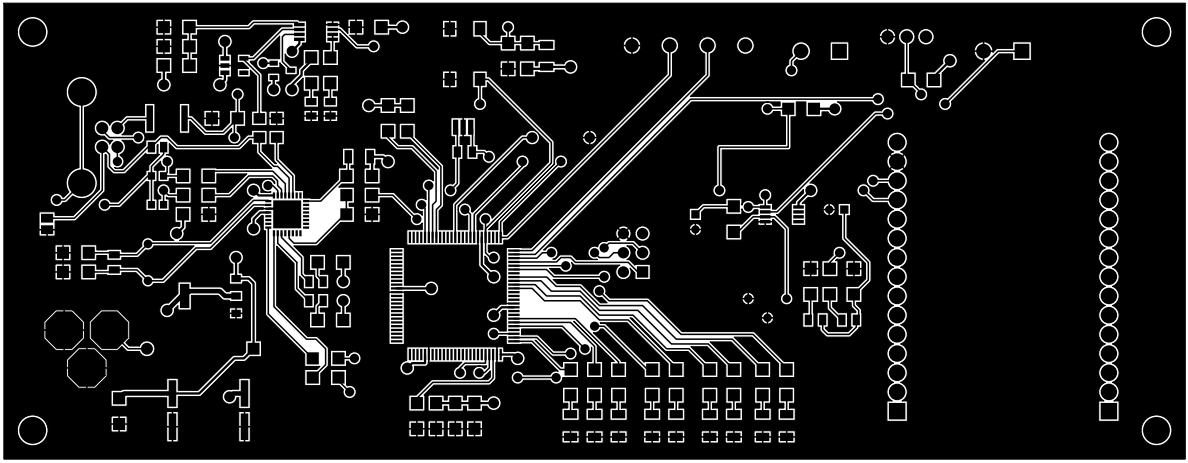


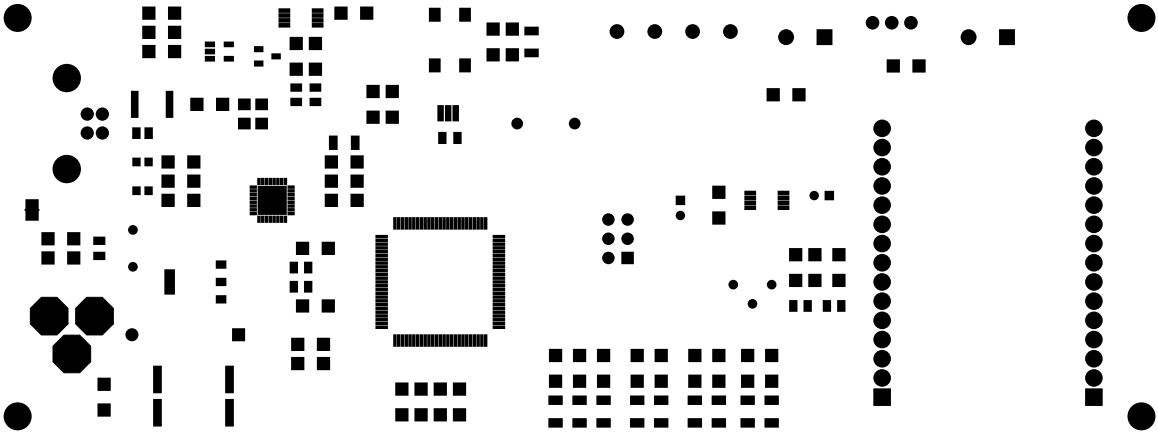
PROJECT:		
DRAWING TITLE:		
DRAWN BY:	REVISION:	
SHEET: OF	STATUS:	
DATE:	DESIGNSPARK PCB available for FREE at www.DesignSpark.com/PCB	

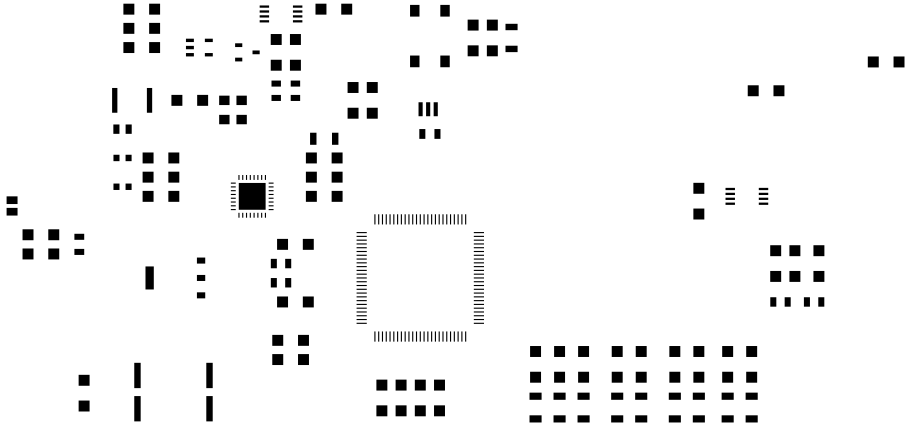
ANEXO 5: FICHEROS GERBER PCB Estudio Previo

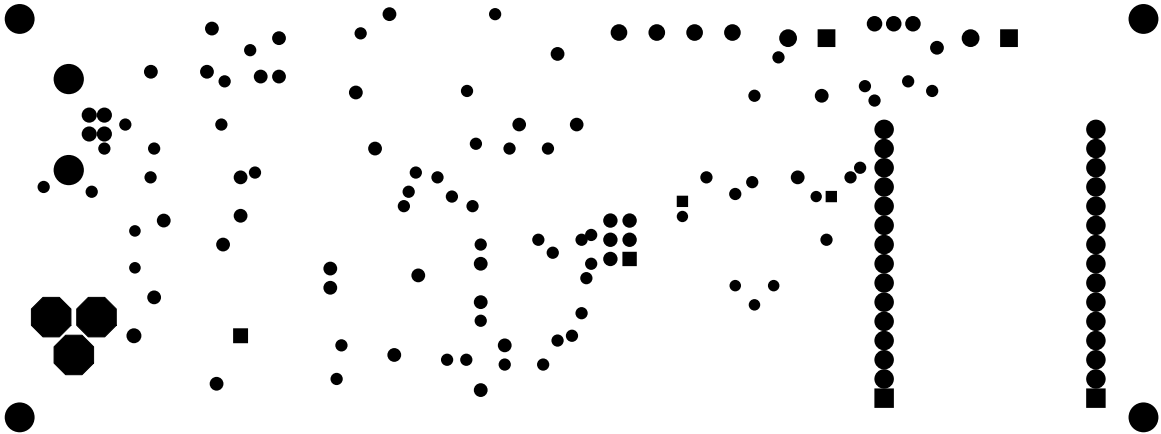

RESET
ON L
J-ULTRASONIDOS
J-LDR
J-DHT
J-YL69
BUZZER
JCSP
TX
RX
LDR1
LDR2
LDR3
YL69-1
YL69-2
DHT1
DHT2
DHT3
DHT4
NodeMCU
0V

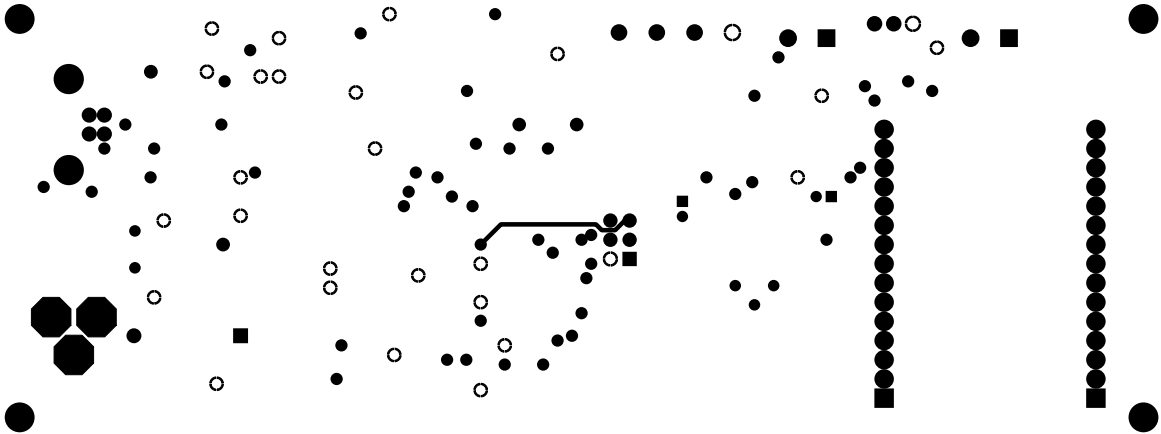

RESET
ON L
J-ULTRASONIDOS
J-LDR
J-DHT
J-YL69
BUZZER
JCSP
TX
RX
LDR1
LDR2
LDR3
YL69-1
YL69-2
DHT1
DHT2
DHT3
DHT4
NodeMCU
0V

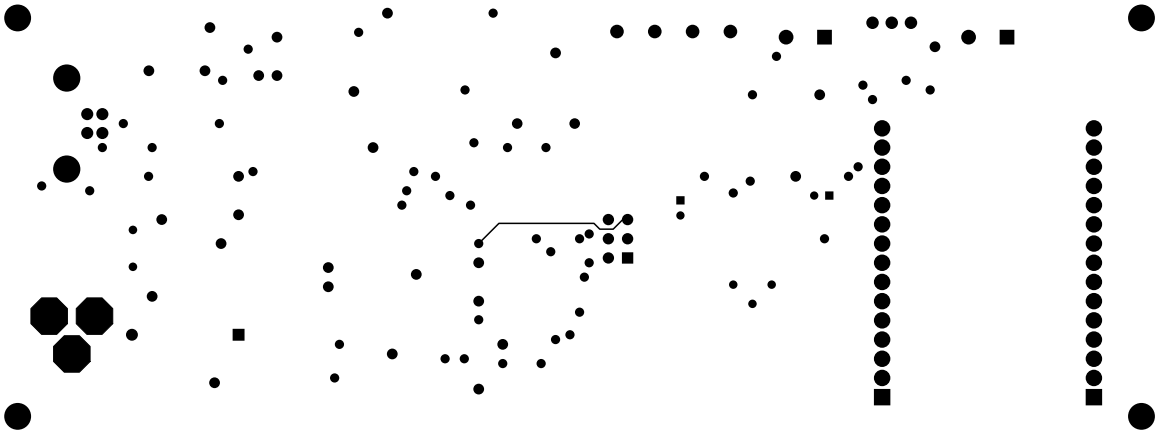


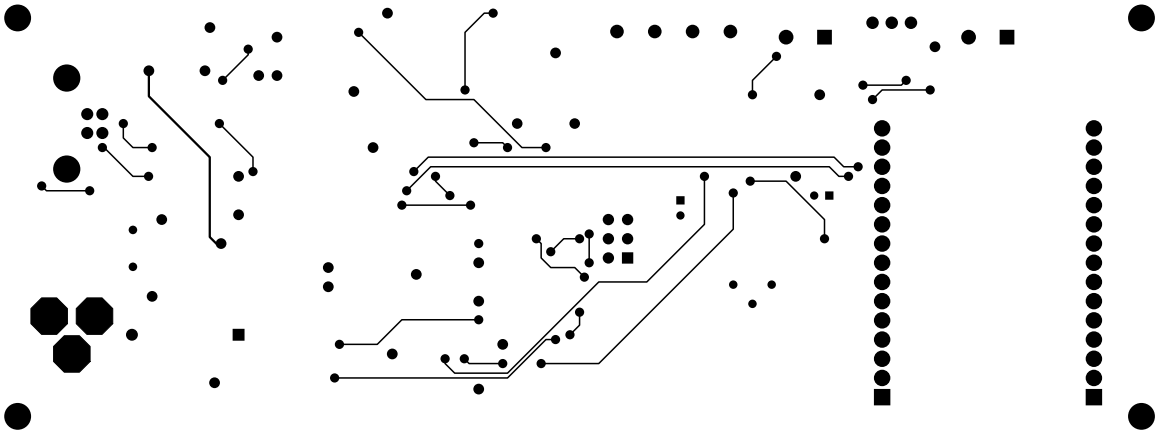


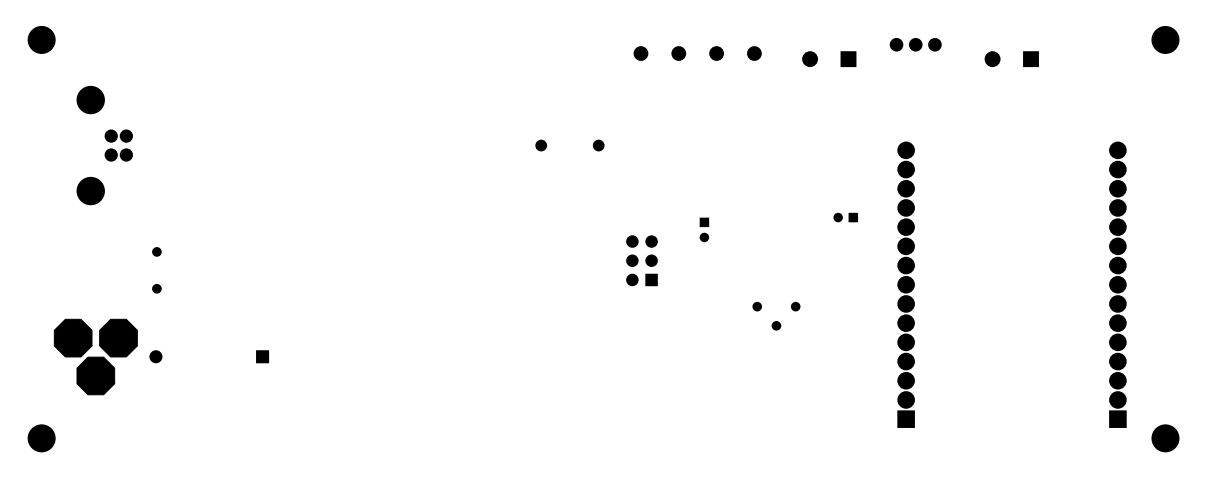


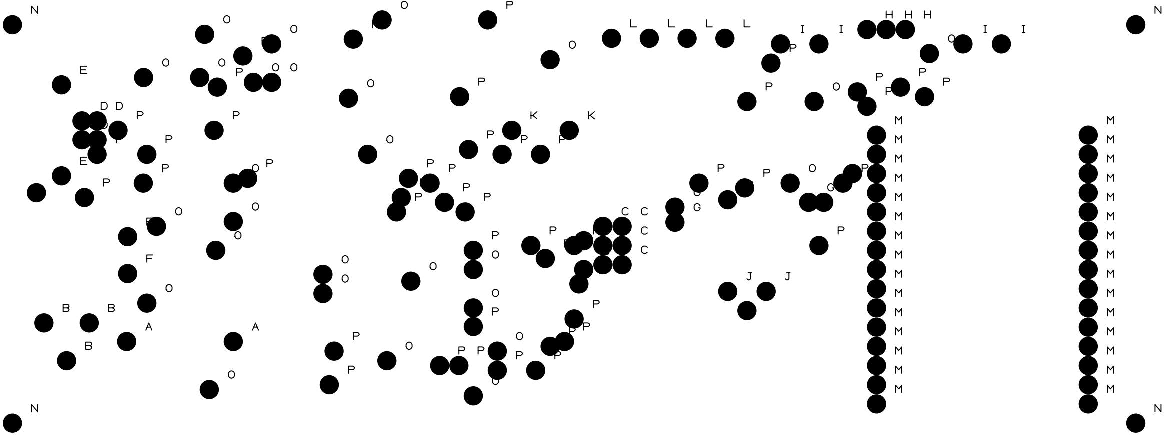












ANEXO 6: CÓDIGO

```
//------------------------ LIBRERIAS ----------------------------
#include <WiFi.h>
#include <WiFiClient.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <DHT.h>
#include <LiquidCrystal_I2C.h>

/************************ MENSAJES LCD *********************************/
// Mensajes en LCD (A es la primera línea, B la segunda)

const String  ESCRITURA_CORRECTA_LCD = "ESCRITURA CORRECTA EN LCD:\n";
const String  ERROR_ESCRITURA_LCD = "ERROR ESCRITURA EN LCD:\n";

#define ERROR_ESCRITURA_LCD 1
#define ERROR_LECTURA_SENSOR 2

const String  LINEA = "----------------";
const String  VACIO = "";
const String  ERROR_ESCRITURA_A = " ERR. ESCRITURA ";
const String  ERROR_ESCRITURA_B = "  PANTALLA LCD";
const String  ERROR_INICIALIZACION_A = "ERROR EN PROCESO";
const String  ERROR_INICIALIZACION_B = " INICIALIZACION";

const String  BIENVENIDA_A = "  TFG. - GIEIA";
const String BIENVENIDA_B = "EVA M. HERNANDEZ";

const String  INIT_A = " INICIALIZACION";
const String  INIT_B_HUMEDAD_OK = " HUMEDAD OK";
const String  INIT_B_HUMEDAD_KO = " HUMEDAD KO";
```

```
const String  INIT_B_TEMPERATURA_OK = " TEMPERATURA OK";
const String  INIT_B_TEMPERATURA_KO = " TEMPERATURA KO";
const String  INIT_B_TIERRA_OK = " HUM. TIERRA OK";
const String  INIT_B_TIERRA_KO = " HUM. TIERRA KO";
const String INIT_B_LUM_OK = " LUMINOSIDAD OK";
const String INIT_B_LUM_KO = " LUMINOSIDAD KO";
const String INIT_B_AGUA_OK = " NIVEL AGUA OK";
const String INIT_B_AGUA_KO = " NIVEL AGUA KO";
const String  INIT_B_PH_OK = " PH AGUA OK";
const String  INIT_B_PH_KO = " PH AGUA KO";
const String  INIT_B_VENTILADOR_OK = " VENTILADOR OK";
const String  INIT_B_VENTILADOR_KO = " VENTILADOR KO";
const String INIT_B_RIEGO_OK = " RIEGO OK";
const String  INIT_B_RIEGO_KO = " RIEGO KO";
const String INIT_B_ALARMA_OK = " ALARMA OK";
const String  INIT_B_ALARMA_KO = " ALARMA KO";

const String  INIT_CONEXIONWIFI = "  CONFIG. WIFI";
const String  INIT_B_CONEXIONWIFI_OK = "CONEXION WIFI OK";
const String INIT_B_CONEXIONADAF_OK = "CONEXION ADAF OK";

const String INIT_A_CONEXIONWIFI_KO = "     ERROR";
const String  INIT_B_CONEXIONWIFI_KO = "EN CONEXION WIFI";

const String INIT_A_CONEXIONADAF_KO = "     ERROR";
const String INIT_B_CONEXIONADAF_KO = "EN CONEXION ADAF";

const String  INIT_A_MODO_SINCONEXION = "ENTRANDO EN MODO";
const String  INIT_B_MODO_SINCONEXION = "  SIN CONEXION";

const String MODO_AUTONOMO = "M. AUTONOMO ";
const String  MODO_MANUAL = "M. MANUAL   ";
```

```
// M. AUTÓNOMO W VR
// M. MANUAL   W VR

const String  HUMEDAD = "Humedad: ";
const String  TEMPERATURA = "Temp: ";
const String  SUELO = "Suelo: ";
const String  NIVEL_AGUA = "Niv.Agua: ";
const String PH = "PH Agua: ";
const String  LUZ_BAJA = "Luz: Baja";
const String  LUZ_MEDIA = "Luz: Media";
const String LUZ_ALTA = "Luz: Alta";
const String  HUMEDAD_TIERRA = "H.Tierra: ";
const String  HUMEDAD_TIERRA_SECA = "H.Tierra: Seca";
const String  HUMEDAD_TIERRA_IDEAL = "H.Tierra: Ideal";
const String HUMEDAD_TIERRA_HUMEDA = "H.Tierra: Húmeda";

const int  LINEA_INT = 0;
const int  HUMEDAD_INT = 1;
const int  TEMPERATURA_INT = 2;
const int  HUMEDAD_TIERRA_INT = 3;
const int  NIVEL_AGUA_INT = 4;
const int  PH_INT = 5;

const String  ALERTA_A = "     ALERTA";
const String  ALERTA_B_HELADA = "RIESGO DE HELADA";
const String  ALERTA_B_CALOR = " CALOR EXTREMO";
const String  ALERTA_B_AGUA = "NIVEL AGUA BAJO";
const String  ALERTA_B_TIERRA_SECA = "  TIERRA SECA";


//------------------------- PINES ------------------------------
```

```
// Definimos el pin digital donde se conecta el sensor de humedad y
temperatura
#define DHTPIN 23
// Dependiendo del tipo de sensor
#define DHTTYPE DHT22
// Pin analogico de entrada para el sensor de Humedad de Tierra (YL69)
#define PinDatosHT 36
// Pin analogico de entrada para el LDR (Luminosidad)
#define PinLDR 39
// Pin analogico de entrada para el sensor de PH
#define PinPH 34
// Configuramos los pines del sensor de Ultrasonidos - Trigger y Echo
#define PinTrig 17
#define PinEcho 19

// Pines relés:
const int releVentilador = 27;
const int releBombaAgua = 14;
// Pin buzzer para alarmas
const int buzzer = 12;
// Seleccionamos el numero de filas y columnas de la pantalla LCD 16x2
int lcdColumnas = 16;
int lcdFilas = 2;



/************************ Adafruit.io Setup
*********************************/
// Credenciales Adafruit
#define AIO_SERVER      "io.adafruit.com" // 52.70.203.194
#define AIO_SERVERPORT  1883              // use 8883 for SSL
#define AIO_USERNAME    "ehs00"
#define AIO_KEY         "9a9423dd048e4e08ae1950f9805f34ed"
```

```
/************ Global State (you don't need to change this!)
*****************/


// establecemos dirección de LCD, número de columnas y filas

// para conocer la dirección I2C de la pantalla LCD ejecutar el sketch
"scanI2C"

LiquidCrystal_I2C lcd(0x27, lcdColumnas, lcdFilas);

// Inicializamos Sensor DHT22

DHT dht(DHTPIN, DHTTYPE);


// Create an ESP32 WiFiClient class to connect to the MQTT server.

WiFiClient client;


// Setup the MQTT client class by passing in the WiFi client and MQTT
server and login details.

Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT,
AIO_USERNAME, AIO_USERNAME, AIO_KEY);


/****************************** Feeds
***************************************/


// Configuramos los feed para su publicación. Las rutas MQTT para AIO
siguen la forma: <username>/feeds/<feedname>

Adafruit_MQTT_Publish temperatura = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/temperatura");

Adafruit_MQTT_Publish humedadAmbiente = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/humedadAmbiente");

Adafruit_MQTT_Publish luminosidad = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/luminosidad");

Adafruit_MQTT_Publish humedadTierra = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/humedadTierra");

Adafruit_MQTT_Publish nivelAgua = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/nivelAgua");

Adafruit_MQTT_Publish pHAgua = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/pHAgua");

Adafruit_MQTT_Publish alarmas = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/alarmas");
```

```
// Configuramos los feed para suscribirnos a cambios en los botones

Adafruit_MQTT_Subscribe modoAuto = Adafruit_MQTT_Subscribe (&mqtt,
AIO_USERNAME "/feeds/modoAuto", MQTT_QOS_1);

Adafruit_MQTT_Subscribe riego = Adafruit_MQTT_Subscribe (&mqtt,
AIO_USERNAME "/feeds/riego", MQTT_QOS_1);

Adafruit_MQTT_Subscribe ventilacion = Adafruit_MQTT_Subscribe (&mqtt,
AIO_USERNAME "/feeds/ventilacion", MQTT_QOS_1);


//------------------- VARIABLES GLOBALES -------------------------


// CONTROL DEL TIEMPO


// Variable para contar segundos

unsigned long previousMillis = 0;

unsigned long previousMillisLCD = 0;

unsigned long previousMillisWifi = 0;

// Variable tiempo de muestreo (cada 5 seg)

int tiempoConexionServidor = 20000;

int tiempoConexionSuscripcion = 5000;

int delayInicio = 1500;

int delayConexion=500;

int delayLCD=500;

int delayReconexionWifi = 40000;


// CONTROL WIFI


// Credenciales WiFi:

//const char *ssid = "vodafone0120";

//const char *password = "7TSCLLQZS7GRLW";

//const char *ssid = "MiFibra-58D7";

//const char *password = "Z2JXZuXD";

const char *ssid = "MiFibra-6E64-24G";
```

```
const char *password = "on9Gi3a5";


int contconexion = 0;
int reintentosConexion = 10;
bool CONECTADO = false;



// Variables Sensor Humedad y Temperatura DHT22
float humAmb = 50;
float tempAmb = 25;
// Variables Sensor Humedad Tierra YL69
int valHT = 0;
int finalvalHT = 0;
// Variables Sensor de Luminosidad LDR
int valLDR = 0;
int finalvalLDR = 0;
// Variable Sensor pH
float pH = 0;
// Variables Sensor de Ultrasonidos - nivel de agua depósito (altura)
const float VelSon = 34000.0;   // Constante velocidad sonido en cm/s
const int numLecturas = 10;     // Número de muestras
float lecturas[numLecturas];    // Array para almacenar lecturas
int lecturaActual = 0;          // Lectura por la que vamos
float total = 0;                // Total de las que llevamos
float media = 0;                // Media de las medidas
bool primeraMedia = false;      // Para saber que ya hemos calculado por
lo menos una
float AlturaTotal = 19;         // Altura Total del depósito en cm (A
modificar según depósito, medida del sensor cuando está vacío el depósito)
float AlturaActual = 0;         // Altura Actual en cm
float NivelAgua = 50;           // Nivel de agua en tanto %
// Variable para alarmas. Tipo entero para definir un menu de alarmas
int Alarma = 0;
```

```
// ------- VARIABLES DE ESTADO TIPO String Y CHAR-------
String HumTierra = "ideal";
char charHumTierra [25];
String Luminosidad = "media";
char charLuminosidad [25];
String notaAlarma = "";
char charAlarma [50];
// Definimos variables globales para el modo vacaciones, y para el sistema
manual
String modo = "A";
String regar = "OFF";
String ventilador = "OFF";
String texto_modo = MODO_AUTONOMO;
int datoMostradoLCD = 0;

//-------------------------------- SETUP ---------------------------------
------------------------------

void setup() {
  try{

    // Iniciamos LCD y escribimos mensaje de bienvenida
    try {
      lcd.init();
      lcd.backlight();
      showInLCD(BIENVENIDA_A,BIENVENIDA_B);
        delay(4000);
    }
    catch(int e1) {
      showInLCD(ERROR_INICIALIZACION_A,ERROR_INICIALIZACION_B);
      throw ERROR_ESCRITURA_LCD;
```

```
}

// Inicialización comunicación Serial
Serial.begin(115200);
delay(1000);
showInLCD(INIT_A,LINEA);

// iniciamos sensor DHT
dht.begin();

pinMode ( PinDatosHT, INPUT );
pinMode ( PinLDR, INPUT );
pinMode ( PinTrig, OUTPUT );
pinMode ( PinEcho, INPUT );
pinMode ( PinPH, INPUT );

for (int i = 0; i < numLecturas; i++){
lecturas[i] = 0;
}

// Conexión WIFI y ADAF
configuraConexionesWifi();

  mqtt.subscribe(&modoAuto);
  mqtt.subscribe(&riego);
  mqtt.subscribe(&ventilacion);

// PRIMERA MEDIDA Y PRUEBA DE SENSORES

// TIERRA
try{
  HumTierra = HumedadTierra();
```

```
      delay(delayInicio);
      showInLCD(INIT_A,INIT_B_TIERRA_OK);
    }
    catch (int e){
      delay(delayInicio);
      showInLCD(INIT_A,INIT_B_TIERRA_KO);
    }

    // LUMINOSIDAD
    try{
      Luminosidad = NivelLuminosidad();
      delay(delayInicio);
      showInLCD(INIT_A,INIT_B_LUM_OK);
    }
    catch (int e){
      delay(delayInicio);
      showInLCD(INIT_A,INIT_B_LUM_KO);
    }

    // AGUA
    try{
        for (int j = 0; j < numLecturas; j++){
              NivelAgua = NivelAguaDeposito();
        }
        delay(delayInicio);
        showInLCD(INIT_A,INIT_B_AGUA_OK);
    }
    catch (int e){
      delay(delayInicio);
      showInLCD(INIT_A,INIT_B_AGUA_KO);
    }
```

```
// PH
try{
  pH = pHAguaRiego();
  delay(delayInicio);
  showInLCD(INIT_A,INIT_B_PH_OK);
    }
catch (int e){
  delay(delayInicio);
  showInLCD(INIT_A,INIT_B_PH_KO);
}

  // HUMEDAD Y TEMPERATURA
try{
  getDHT();
  showInLCD(INIT_A,INIT_B_HUMEDAD_OK);
  delay(delayInicio);
  showInLCD(INIT_A,INIT_B_TEMPERATURA_OK);
}
catch (int e){
showInLCD(INIT_A,INIT_B_HUMEDAD_KO);
delay(delayInicio);
showInLCD(INIT_A,INIT_B_TEMPERATURA_KO);
}

// VENTILADOR
try {
  pinMode(releVentilador,OUTPUT);
  digitalWrite(releVentilador,LOW);
  delay(delayInicio);
  showInLCD(INIT_A,INIT_B_VENTILADOR_OK);
    }
catch (int e){
```

```
      delay(delayInicio);
      showInLCD(INIT_A,INIT_B_VENTILADOR_KO);
    }

    // BOMBA DE AGUA
    try{
      pinMode(releBombaAgua,OUTPUT);
      digitalWrite(releBombaAgua,LOW);
      delay(delayInicio);
      showInLCD(INIT_A,INIT_B_RIEGO_OK);
        }
    catch (int e){
      delay(delayInicio);
      showInLCD(INIT_A,INIT_B_RIEGO_KO);
    }

    // ALARMA
    try{
      pinMode(buzzer,OUTPUT);
      delay(delayInicio);
      showInLCD(INIT_A,INIT_B_ALARMA_OK);
        }
    catch(int e){
      delay(delayInicio);
      showInLCD(INIT_A,INIT_B_ALARMA_KO);
    }
  }
  catch(int e) {
    Serial.println(ERROR_INICIALIZACION_A + ERROR_INICIALIZACION_B+" Error
Code: "+String(e));
  }
}
```

```
//------------------------------- LOOP ----------------------------------
------------------------------

void loop() {
  try{

      // ---------------------------------
    // LEEMOS LOS DATOS EN LOS SENSORES
    // ---------------------------------

    try{getDHT();} catch (int e){}
    try{HumTierra = HumedadTierra();} catch (int e){}
    try{Luminosidad = NivelLuminosidad();} catch (int e){}
    try{NivelAgua = NivelAguaDeposito();} catch (int e){}
    try{pH = pHAguaRiego();} catch (int e){}

    // ---------------------------------
    // CONFIGURACIÓN Y GESTIÓN DE ALARMAS
    // ---------------------------------

    if (NivelAgua <= 15){
    Alarma = 1;
    digitalWrite(buzzer, HIGH);
    }else if ((HumTierra == "seco")&&(modo == "M")){
    Alarma = 2;
    digitalWrite(buzzer, HIGH);
    }else if (tempAmb <= 5){
      Alarma = 3;
      digitalWrite(buzzer, HIGH);
    }else if (tempAmb >= 45){
    Alarma = 4;
```

```
      digitalWrite(buzzer, HIGH);
    }else{
    Alarma = 0;
    digitalWrite(buzzer, LOW);
    }

    // ------------------------------------------------
    // REGISTRAMOS CUANTOS MILISEGUNDOS HAN PASADO
    // ------------------------------------------------

    unsigned long currentMillis = millis();

    // ------------------------------------------------------------------
-------------------------
    // MOSTRAMOS EN EL LCD EL MODO DE FUNCIONAMIENTO ACTUAL Y EL ESTADO EN
CASO DE NO HABER ALARMAS
    // ------------------------------------------------------------------
-------------------------
    // M. AUTÓNOMO W VR - Se indica además si Wifi, Ventilador o Riego
están activos
    // M. MANUAL   W VR

      texto_modo = getTextoModo();

    if(Alarma == 0)
    {
      if(datoMostradoLCD == LINEA_INT){
        showInLCD(texto_modo,LINEA);
        datoMostradoLCD = LINEA_INT;
      }

      if (currentMillis - previousMillisLCD >= delayLCD)
      {
```

```
    previousMillisLCD = currentMillis;
    switch (datoMostradoLCD) {
    case LINEA_INT:
      datoMostradoLCD = HUMEDAD_INT;
      showInLCD(texto_modo,HUMEDAD+ String(humAmb) + "% ");
      break;
    case HUMEDAD_INT:
      datoMostradoLCD = TEMPERATURA_INT;
      showInLCD(texto_modo,TEMPERATURA+ String(tempAmb) + "*C ");
      break;
    case TEMPERATURA_INT:
      datoMostradoLCD = HUMEDAD_TIERRA_INT;
      showInLCD(texto_modo,HUMEDAD_TIERRA+ HumTierra);
      break;
    case HUMEDAD_TIERRA_INT:
      datoMostradoLCD = NIVEL_AGUA_INT;
      showInLCD(texto_modo,NIVEL_AGUA+ String(NivelAgua) + "% ");
      break;
    case NIVEL_AGUA_INT:
      datoMostradoLCD = PH_INT;
      showInLCD(texto_modo,PH+ String(pH));
      break;
    case PH_INT:
      showInLCD(texto_modo,LINEA);
      datoMostradoLCD = LINEA_INT;
      break;
    default:
      showInLCD(texto_modo,LINEA);
      datoMostradoLCD = LINEA_INT;
      break;
        }
  }
```

```
    }


    // ------------------------------------------------------------------
-------
    // CONEXIÓN CON EL SERVIDOR DE ADAFRUIT CADA tiempoConexionServidor
SEGUNDOS
    // ------------------------------------------------------------------
-------


    if(CONECTADO)
    {
            try{
              if(!mqtt.ping())
              {mqtt.disconnect();}
              // Conectamos por MQTT al servidor
            }catch(int e) {}


        if(MQTT_connect())
        {
              if (currentMillis - previousMillis >=
tiempoConexionServidor) {
                  // PUBLICAMOS LOS DATOS DE LOS SENSORES
                  previousMillis = currentMillis;
                  Serial.print(F("\nEnviando temperatura: "));
                  Serial.print(tempAmb);
                  Serial.print("...");
                  if (! temperatura.publish(tempAmb)) {
                    Serial.println(F("Error"));
                  } else {
                    Serial.println(F("OK!"));
                  }
                  Serial.print(F("\nEnviando humedadAmbiente: "));
```

```
Serial.print(humAmb);
Serial.print("...");
if (! humedadAmbiente.publish(humAmb)) {
  Serial.println(F("Error"));
} else {
  Serial.println(F("OK!"));
}
HumTierra.toCharArray(charHumTierra, 15);
Serial.print(F("\nEnviando humedadTierra: "));
Serial.print(HumTierra);
Serial.print("...");
if (! humedadTierra.publish(charHumTierra)) {
  Serial.println(F("Error"));
} else {
  Serial.println(F("OK!"));
}
Luminosidad.toCharArray(charLuminosidad, 15);
Serial.print(F("\nEnviando luminosidad: "));
Serial.print(Luminosidad);
Serial.print("...");
if (! luminosidad.publish(charLuminosidad)) {
  Serial.println(F("Error"));
} else {
  Serial.println(F("OK!"));
}
Serial.print(F("\nEnviando nivel de agua: "));
Serial.print(NivelAgua);
Serial.print("...");
if (! nivelAgua.publish(NivelAgua)) {
  Serial.println(F("Error"));
} else {
  Serial.println(F("OK!"));
```

```
                }
                Serial.print(F("\nEnviando pH Agua de Riego: "));
                Serial.print(pH);
                Serial.print("...");
                if (! pHAgua.publish(pH)) {
                  Serial.println(F("Error"));
                } else {
                  Serial.println(F("OK!"));
                }

                // PUBLICAMOS LAS ALARMAS
                switch (Alarma) {
                Serial.print(F("Valor analizado para la variable Alarma:
"));
                Serial.println(Alarma);
                case 1:
                  // Enviamos al servidor la alerta: Alarma AGUA
DEPOSITO, NIVEL MUY BAJO! (<=15%)
                  notaAlarma = "AGUA DEPOSITO, NIVEL MUY BAJO!";
                  notaAlarma.toCharArray(charAlarma, 50);
                  Serial.print(F("\nEnviando Alarma 1 ..."));
                  if (! alarmas.publish(charAlarma)) {
                  Serial.println(F("Error"));
                  } else {
                  Serial.println(F("OK!"));
                  }
                  showInLCD(ALERTA_A,ALERTA_B_AGUA);
                  break;
                case 2:
                  // Enviamos al servidor la alerta: SUELO MUY SECO Y
MODO MANUAL ACTIVADO, REGAR!
                  notaAlarma = "SUELO MUY SECO Y MODO MANUAL ACTIVADO,
REGAR!";
```

```
                    notaAlarma.toCharArray(charAlarma, 50);
                    Serial.print(F("\nEnviando Alarma 2 ..."));
                    if (! alarmas.publish(charAlarma)) {
                    Serial.println(F("Error"));
                    } else {
                    Serial.println(F("OK!"));
                    }
                    showInLCD(ALERTA_A,ALERTA_B_TIERRA_SECA);
                    break;
                  case 3:
                    // Enviamos al servidor la alerta: Alarma FRIO EXTREMO,
PELIGRO HELADA! (<=5ºC)
                    notaAlarma = "FRIO EXTREMO, PELIGRO HELADA!";
                    notaAlarma.toCharArray(charAlarma, 50);
                    Serial.print(F("\nEnviando Alarma 3 ..."));
                    if (! alarmas.publish(charAlarma)) {
                    Serial.println(F("Error"));
                    } else {
                    Serial.println(F("OK!"));
                    }
                    showInLCD(ALERTA_A,ALERTA_B_HELADA);
                    break;
                  case 4:
                    // Enviamos al servidor la alerta: Alarma CALOR
EXTREMO, PELIGRO FUEGO! (>=45ºC)
                    notaAlarma = "CALOR EXTREMO, PELIGRO FUEGO!";
                    notaAlarma.toCharArray(charAlarma, 50);
                    Serial.print(F("\nEnviando Alarma 4 ..."));
                    if (! alarmas.publish(charAlarma)) {
                    Serial.println(F("Error"));
                    } else {
                    Serial.println(F("OK!"));
                    }
```

```
                    showInLCD(ALERTA_A,ALERTA_B_CALOR);
                    break;
                  default:
                    // En cualquier otro caso no haremos nada. Aseguramos
buzzer apagado.
                    Alarma = 0;
                    digitalWrite(buzzer, LOW);
                    break;
                  }

              }

              Serial.println("Inicio Suscripcion.");
              Adafruit_MQTT_Subscribe *subscription;
              // este es el 'subbloqueo ocupado de espera de paquetes de
suscripción entrantes'
              while(subscription =
mqtt.readSubscription(tiempoConexionSuscripcion))
              {
                    Serial.println("Bucle Suscripción.");
                    // solo nos preocupamos por los eventos del boton modo
                    if (subscription == &modoAuto) {
                        // convertir mqtt ascii payload a int
                        char *valueModo = (char*)modoAuto.lastread;
                        Serial.print(F("Received: "));
                        Serial.println(valueModo);
                        // Aplicar mensaje al modo
                        modo = String(valueModo);
                        showInLCD(getTextoModo(),LINEA);
                    }
                    if (subscription == &riego) {
                        // convertir mqtt ascii payload a int
                        char *valueRiego = (char *)riego.lastread;
```

```
                        Serial.print(F("Received: "));
                        Serial.println(valueRiego);
                        // Aplicar mensaje al riego
                        regar = String(valueRiego);
                        showInLCD(getTextoModo(),LINEA);
                    }
                    if (subscription == &ventilacion) {
                        // convertir mqtt ascii payload a int
                        char *valueVentilacion = (char
*)ventilacion.lastread;
                        Serial.print(F("Received: "));
                        Serial.println(valueVentilacion);
                        // Aplicar mensaje al ventilador
                        ventilador = String(valueVentilacion);
                        showInLCD(getTextoModo(),LINEA);
                    }

                    // Condicionales para configurar el modo
(A=Automatico,M=Manual)
                    if (modo == "A") {
                        // Paso a modo automatico
                        modoAutomatico();
                    }else if (modo == "M") {
                        //  Paso a modo manual
                        if (regar == "ON") {
                         // Se activa riego
                         digitalWrite(releBombaAgua, HIGH);
                        }else if (regar == "OFF") {
                         // Se desactiva riego
                         digitalWrite(releBombaAgua, LOW);
                        }
                        if (ventilador == "ON") {
                         // Se activa ventilacion
```

```
                        digitalWrite(releVentilador, HIGH);
                       }else if (ventilador == "OFF") {
                        // Se desactiva ventilacion
                        digitalWrite(releVentilador, LOW);
                       }
                    }
                }
      }
    }
      else
      {
            //Reintentar conexión Wifi

            if (currentMillis - previousMillisWifi >= delayReconexionWifi)
{
                  // Publicamos los datos de los sensores cada
tiempoConexionServidor segundos
                  previousMillisWifi = currentMillis;
                  configuraConexionesWifi();
            }
      }

  }
  catch(int e) {
    Serial.println(ERROR_INICIALIZACION_A + ERROR_INICIALIZACION_B+" Error
Code: "+String(e));
  }
}

//-------------------------------- FUNCIONES ----------------------------
------------------------------

void getDHT(){
```

```
  try {

    Serial.println("SENSOR DHT: Inicio de Medida.");

    float tempIni = tempAmb;
    float humIni = humAmb;
    // Leemos la temperatura
    tempAmb = dht.readTemperature();
    // Leemos la humedad relativa
    humAmb = dht.readHumidity();
    // Comprobamos si ha habido algún error en la lectura
    if (isnan(tempAmb) || isnan(humAmb)) {
    Serial.println("SENSOR DHT:  Error obteniendo los datos del sensor
DHT.");
    // Dejamos los valores anteriores
    tempAmb = tempIni;
    humAmb = humIni;
    throw ERROR_LECTURA_SENSOR;
    return;
    }

    Serial.println("SENSOR DHT: Lectura correcta. Temp Ambiente: "+
String(tempAmb)+" ,Humedad Ambiente: "+ String(humAmb));
    return;
  }
  catch (int e){
    Serial.println("SENSOR DHT: Error Lectura.");
    throw ERROR_LECTURA_SENSOR;
  }
}

String HumedadTierra(){
  try {
```

```
    Serial.println("SENSOR HUMEDAD TIERRA: Inicio de Medida.");


    // Mapeo de lectura analógica del sensor de humedad de tierra:
(conversor ADC de 12 bits => 4096 estados (de 0 a 4095))
    valHT = map(analogRead(PinDatosHT), 0, 4095, 100, 0);
    // Guardamos el valor final de humedad de tierra,
    // definimos sus valores extremos (limites,rango) en 0% y 100% de
humedad relativa
    finalvalHT = constrain (valHT, 0, 100);


    // Actuaciones:
    if ((finalvalHT >= 0) and (finalvalHT <= 25)) {
      HumTierra = "seco";
    } else if ((finalvalHT >= 26) and (finalvalHT <= 45)) {
      HumTierra = "ideal";
    } else if ((finalvalHT >= 46) and (finalvalHT <= 100)) {
      HumTierra = "humedo";
    }


    Serial.println("SENSOR HUMEDAD TIERRA: Lectura correcta: "+ HumTierra+
" ValHT: "+ String(valHT) + " ,FinalValHT: "+ String(finalvalHT));
    return HumTierra;
  }
  catch (int e){
    Serial.println("SENSOR HUMEDAD TIERRA: Error Lectura.");
    throw ERROR_LECTURA_SENSOR;
  }
}


String NivelLuminosidad(){


  try {
```

```
    Serial.println("SENSOR LUMINOSIDAD: Inicio de Medida.");
    // El valor leido por el ADC (voltaje) aumenta de manera directamente
proporcional
    // con respecto a la luz percibida por el LDR, por ello definiremos un
% de oscuridad (100% oscuridad total y 0% luminosidad total)
    // Mapeo de lectura analógica del sensor LDR: (conversor ADC de 12
bits => 4096 estados (de 0 a 4095))
    valLDR = map(analogRead(PinLDR), 0, 4095, 100, 0);
    // Guardamos el valor final de luminosidad,
    // definimos sus valores extremos (limites,rango) en 0% y 100% de
oscuridad
    finalvalLDR = constrain (valLDR, 0, 100);

    if ((finalvalLDR >= 0) and (finalvalLDR <= 30)) {
      Luminosidad = "alta";
    } else if ((finalvalLDR >= 31) and (finalvalLDR <= 60)) {
      Luminosidad = "media";
    } else if ((finalvalLDR >= 61) and (finalvalLDR <= 100)) {
      Luminosidad = "baja";
    }

    Serial.println("SENSOR LUMINOSIDAD: Lectura correcta. Luminosidad:
"+Luminosidad+" ,ValLDR: "+ String(valLDR) + " ,FinalValLDR: "+
String(finalvalLDR));
    return Luminosidad;
    }
  catch (int e){
    Serial.println("SENSOR LUMINOSIDAD: Error Lectura.");
    throw ERROR_LECTURA_SENSOR;
  }
}

float NivelAguaDeposito(){
```

```
  try {


    Serial.println("SENSOR NIVEL DE AGUA: Inicio de Medida. Lecturas
Guardadas: "+String(lecturaActual));

    // Eliminamos la última medida

    total = total - lecturas[lecturaActual];

    iniciarTrigger();

    // La función pulseIn obtiene el tiempo que tarda en cambiar entre
estados, en este caso a HIGH

    unsigned long tiempo = pulseIn(PinEcho, HIGH);

    // Obtenemos la distancia en cm, hay que convertir el tiempo en
segundos (ya que está en microsegundos por eso se multiplica por 0.000001)

    float distancia = tiempo * 0.000001 * VelSon / 2.0;

    // Almacenamos la distancia en el array

    lecturas[lecturaActual] = distancia;

    // Añadimos la lectura al total

    total = total + lecturas[lecturaActual];

    // Avanzamos a la siguiente posición del array

    lecturaActual = lecturaActual + 1;

    // Comprobamos si hemos llegado al final del array

    if (lecturaActual >= numLecturas){

    primeraMedia = true;

    lecturaActual = 0;

    }

    // Calculamos la media

    media = total / numLecturas;

    // Solo mostramos si hemos calculado por lo menos una media

    if (primeraMedia){

    // Mostramos medida por puerto serie:

    Serial.print(media);

    Serial.println(" cm media");

    // Calculamos la altura en cm de agua desde la base del deposito
```

```
    AlturaActual = AlturaTotal - media;
    // Calculamos el porcentaje de llenado:
    NivelAgua = (AlturaActual*100)/AlturaTotal;
    // Mostramos Nivel de agua en cm y en tanto % por puerto serie:
    Serial.println("SENSOR NIVEL DE AGUA: Lectura correcta. Altura (cm):
"+ String(AlturaActual)+" ,Nivel Agua: "+ String(NivelAgua) + "%");
    }
    else{
    Serial.println("SENSOR NIVEL DE AGUA: Calculando primera media de
lecturas, Valor Actual: " + String(lecturaActual));
    }
    return NivelAgua;
    }
  catch (int e){
    Serial.println("SENSOR NIVEL DE AGUA: Error Lectura.");
    throw ERROR_LECTURA_SENSOR;
  }
}

void iniciarTrigger(){
  // Método que inicia la secuencia del Trigger para comenzar a medir
  // Ponemos el Triiger en estado bajo y esperamos 2 ms
  digitalWrite(PinTrig, LOW);
  delayMicroseconds(2);
  // Ponemos el pin Trigger a estado alto y esperamos 10 ms
  digitalWrite(PinTrig, HIGH);
  delayMicroseconds(10);
  // Comenzamos poniendo el pin Trigger en estado bajo
  digitalWrite(PinTrig, LOW);
}

float pHAguaRiego(){
  try{
```

```
    Serial.println("SENSOR PH: Inicio de Medida.");

    int mvpH = analogRead(PinPH);
    double vpH = (5*mvpH) / 4095; //conversion de mv a V (adc de 12 bits)
    // PH_step = (voltage@PH7 - voltage@PH4) / (PH7 - PH4)
    // PH_probe = PH7 - ((voltage@PH7 - voltage@probe) / PH_step)
    float pH = 7 + ((2.5 - vpH) / 0.357);

    Serial.println("SENSOR PH: Lectura correcta. mvpH: "+String(mvpH)+"\t
vpH: "+String(vpH)+ "\t PH: "+String(pH));
    return pH;
    }
  catch (int e){
    Serial.println("SENSOR PH: Error Lectura.");
    throw ERROR_LECTURA_SENSOR;
  }
}

void modoAutomatico(){
  // ACTUACIONES - Riego y alarma:
  // Si hay suficiente agua en el depósito (más del 15%) podremos regar
  if (NivelAgua>15){
    // Desactivamos Alarma (buzzer) - No hay peligro de que se estropee la
bomba
    digitalWrite(Alarma,LOW);
    // Solo regaremos al atardecer para que las plantas retengan la mayor
cantidad de agua posible sin que se evapore
    if(Luminosidad == "baja"){
      // Y regaremos si hace falta, es decir, si la tierra está seca
      if(HumTierra=="seco"){
        // Activamos riego
        digitalWrite(releBombaAgua,HIGH);
```

```
      }else{
        // Desactivamos riego
        digitalWrite(releBombaAgua,LOW);
      }
    }else{
      // Desactivamos riego
      digitalWrite(releBombaAgua,LOW);
    }
  }else{
    // Activamos alarma cuando el nivel de agua sea inferior al 15% ya que
hay peligro de que se estropee la bomba
    digitalWrite(Alarma,HIGH);
    // Como la bomba no debe trabajar en vacío, aseguramos que el riego
está desconectado.
    digitalWrite(releBombaAgua,LOW);
  }


  // ACTUACIONES - Ventilación:
  if(Luminosidad != "baja"){
    if(tempAmb>30){
      // Si la luminosidad es media o alta, y ademas la temperatura es
mayor de 30*C
      // activaremos la ventilación (Condiciones DIURNAS)
      digitalWrite(releVentilador,HIGH);
    }else{
      // En otro caso desactivamos la ventilación
      digitalWrite(releVentilador,LOW);
    }
  }else if(Luminosidad=="baja"){
    if(tempAmb>20){
      // Si la luminosidad baja, y ademas la temperatura es mayor de 20*C
      // activaremos la ventilación (Condiciones NOCTURNAS)
      digitalWrite(releVentilador,HIGH);
```

```
    }else{
      // En otro caso desactivamos la ventilación
      digitalWrite(releVentilador,LOW);
    }
  }
}

bool MQTT_connect() {

  try{
    // Conectarse a adafruit.io a través de MQTT
    // Función para conectarse y reconectarse según sea necesario al
servidor MQTT.
    // Debería llamarse en la función de bucle y tendrá cuidado si se
conecta.
    int8_t ret;

    // Stop if already connected.
    if (mqtt.connected()) {
            return true;
    }

    Serial.print("Connecting to MQTT... ");

    uint8_t retries = 3;
    while ((ret = mqtt.connect()) != 0) { // connect will return 0 for
connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying MQTT connection in 10 seconds...");
       mqtt.disconnect();
       delay(tiempoConexionServidor);  // wait delayConexion seconds
       retries--;
       if (retries == 0) {
```

```
            CONECTADO = false;
            showInLCD(INIT_A_CONEXIONADAF_KO,INIT_B_CONEXIONADAF_KO);
            delay(delayInicio);
            showInLCD(INIT_A_MODO_SINCONEXION,INIT_B_MODO_SINCONEXION);
            return false;
       }
    }
    Serial.println("MQTT Connected!");
      return true;
  }
  catch (int e) {
    return false;
  }
}

bool showInLCD(String lineaA, String lineaB){
  try{
    // limpiamos display para escribir nuevo mensaje
    lcd.clear();
    // Mostramos informacion inicial en el display
    lcd.setCursor(0,0);
    lcd.print(lineaA);
    lcd.setCursor(0,1);
    lcd.print(lineaB);
    Serial.println(ESCRITURA_CORRECTA_LCD + "  LineaA: <" + lineaA +">\n
LineaB: <" + lineaB+">");
    return true;
  }
  catch(int e)
  {
    Serial.println(ERROR_ESCRITURA_LCD + "  LineaA: <" + lineaA +">\n
LineaB: <" + lineaB+">");
    return false;
```

```
  }
}


void configuraConexionesWifi()
{
      contconexion = 0;
      // Conexión WIFI y ADAF
      showInLCD(INIT_A,INIT_CONEXIONWIFI);
      WiFi.begin(ssid, password);
      while (WiFi.status() != WL_CONNECTED and contconexion <
reintentosConexion) { //Cuenta hasta REINTENTOS_CONEXION si no se puede
conectar lo cancela
            ++contconexion;
            delay(delayConexion);
            Serial.print(".");
      }

      if (contconexion < reintentosConexion)
      {
        CONECTADO = true;
        Serial.println(WiFi.localIP());
        showInLCD(INIT_A,INIT_B_CONEXIONWIFI_OK);
        delay(delayInicio);

      }else
      {
       CONECTADO = false;
       showInLCD(INIT_A_CONEXIONWIFI_KO,INIT_B_CONEXIONWIFI_KO);
       delay(delayInicio);
       showInLCD(INIT_A_MODO_SINCONEXION,INIT_B_MODO_SINCONEXION);
       delay(delayInicio);
      }
}
```

```
String getTextoModo()
{
    if (modo == "M") {texto_modo = MODO_MANUAL;}
    else {texto_modo = MODO_AUTONOMO;}

    if(CONECTADO) {texto_modo = texto_modo+"W ";}
    else{texto_modo = texto_modo+"- ";}

    if (ventilador == "ON") {texto_modo = texto_modo+"V";}
    else{texto_modo = texto_modo+"-";}

    if (regar == "ON") {texto_modo = texto_modo+"R";}
    else{texto_modo = texto_modo+"-";}

      return texto_modo;
}
```

DOCUMENTO Nº 3:

PLIEGO DE CONDICIONES

Índice de Pliego de Condiciones

1. DISPOSICIONES GENERALES

1.1. Resumen del proyecto

El proyecto se basa en el diseño de un modelo experimental de un sistema de monitorización y control sobre un invernadero de uso doméstico conectado a Internet empleando la tecnología IoT.

El sistema se compone de un módulo con conectividad WiFi basado en un SoC ESP32, sensórica básica y actuadores de baja potencia de funcionamiento, como son un ventilador y una bomba de agua.

Las conexiones de los sensores con el módulo principal se realizan con cables de cobre. La conexión del módulo con el ordenador mediante cable Serial (USB – micro USB). La conexión del sistema con alimentación se realiza a través de un módulo de fuente de alimentación de entrada 9V y salida a 3.3V, 5V y 12V. Éste módulo de fuente de alimentación va conectado directamente a la red eléctrica mediante un transformador AC/DC.

1.2. Alcance y aplicabilidad del pliego de condiciones

El objetivo del siguiente pliego de condiciones es la ordenación de las condiciones técnicas y facultativas que han de estar presentes en la ejecución de este proyecto. Este proyecto va a ser supervisado por un ingeniero Industrial, que en este caso va a ser el profesor encargado de dirigir el proyecto. El pliego de prescripciones técnicas establece la definición del montaje en cuanto a su naturaleza intrínseca. Las condiciones incluidas se entienden aplicables al ámbito del diseño del prototipo.

2. CONDICIONES TECNICAS

2.1. Características del prototipo

El sistema está diseñado para ser alimentado directamente a 3.3V, 5V y 12V gracias a un módulo de alimentación que recibe 9V de un transformador AC/DC conectado a la red y que proporciona dichas tensiones en tres salidas diferenciadas. También se puede realizar a alimentación mediante otros métodos, como baterías portátiles.

Los materiales empleados están sujetos a pruebas y/o ensayos para asegurar su funcionamiento. De ser sustituidos por otros similares deberán tener las mismas características que los originales. Para ello se deberán revisar las especificaciones propias del componente a sustituir en su respectivo Datasheet.

2.2. Condiciones de ejecución

El prototipo deberá realizarse siempre siguiendo las especificaciones de diseño. Cualquier modificación parcial puede inducir en un mal funcionamiento del sistema o prototipo.

2.3. Condiciones de montaje

El individuo o empresa realizadora del presente proyecto deberá estar capacitada para la interpretación del mismo con carácter global. Todo el proceso se ejecutará con estricta sujeción al pliego de condiciones y conforme a las directrices dadas en el proyecto. En caso de ser necesaria alguna modificación para adaptar el prototipo, se debe realizar bajo revisión de un ingeniero técnico o en su defecto de una persona cualificada para dicho fin.

2.4. Precauciones de uso

Cualquiera de los componentes empleados no podrá conectarse a una tensión superior a la especificada como máxima tanto en la memoria del proyecto como en los documentos técnicos (Datasheets) que establece el fabricante de dicho componente. De no

cumplirse este requisito, se puede causar el deterioro total o parcial del sistema o alguno de sus componentes. La vida útil de éstos últimos estará condicionada a la dada por el fabricante, siempre y cuando se sigan las instrucciones de mantenimiento de los mismos.

El montaje y puesta en marcha del prototipo diseñado del sistema debe seguir las indicaciones, conexiones y recomendaciones expuestas a lo largo de los diferentes apartados del presente documento técnico.

Debido al propio envejecimiento de los materiales frente al tiempo, éstos pueden verse deteriorados o incluso quedar inutilizables, por lo que la recomendación será sustituir aquellos por otros iguales, a ser posible del mismo fabricante.

3. CONDICIONES LEGALES

3.1. Usos permitidos

El proyecto define el diseño de un sistema de monitorización y control apoyado en un prototipo, por lo que no dispone de los permisos pertinentes para ser aplicado como un proyecto completo, y por tanto, implica que no se podrá comercializar con el mismo.

Un prototipo se implementa con el fin de apoyar un estudio realizado previamente y para la realización de pruebas y ensayos experimentales en un ámbito seguro y controlado, realizado por usuarios con conocimientos de la materia y bajo su propia responsabilidad. En cualquier caso, se deben respetar las características técnicas y cumplir las precauciones de aplicación impuestas anteriormente. El uso inadecuado del prototipo es responsabilidad directa y única del usuario, careciendo de ningún tipo de responsabilidad civil o penal por parte del diseñador.

3.2. Propiedad intelectual

La propiedad intelectual reside tanto en el diseñador como en su institución. Asimismo, los componentes involucrados en el diseño forman parte de trabajos externos, por lo que la propiedad intelectual corresponde a su autor original.

3.3. Seguridad y Salud

Las disposiciones mínimas de seguridad y salud para el uso de los diferentes prototipos por parte de los usuarios se establecen en el Real Decreto 1801/2003, el cual forma parte de la Ley General de Sanidad y la Ley General para la Defensa de los Consumidores y Usuarios.

4. CONDICIONES ECONÓMICAS

El sistema resultante del presente proyecto posee carácter de prototipo, por lo que no se encuentra disponible para su comercialización.

Por otra parte, se valora una futura explotación del mismo, siempre que se encuentre enmarcada en la investigación o en la mejora del sistema diseñado. Las posibles partidas presupuestarias fruto de esa posible explotación se definirán en el marco de esa futura investigación.

DOCUMENTO Nº 4:

PRESUPUESTO

Índice de presupuesto

1. COSTES DIRECTOS

1.1. MANO DE OBRA DIRECTA

Al tratarse de un proyecto basado en el diseño de un prototipo, solo es necesaria la contratación de un ingeniero. Este se encarga del estudio, diseño, desarrollo y testeo del prototipo y la elaboración de la documentación.

En la siguiente tabla se establece los tiempos aproximados y generales que se han invertido en cada actividad con su precio por hora y total.

Ingeniero	Nº de horas	Sueldo Bruto / hora	Total
Estudio y documentación	40 h	16.00 € / hora	640 €
Diseño de los circuitos y programación	45 h		720 €
Implementación del circuito final (Programación + montaje)	35 h		560 €
Pruebas del circuito final y test del dispositivo	15 h		240 €
Elaboración de la memoria	40 h		640 €
Coste total			**2800 €**

Tabla 1 - Mano de Obra Directa

1.2. MATERIAS PRIMAS

Las materias primas que se han consumido para la investigación y para el desarrollo del prototipo se recogen en la siguiente tabla de manera detallada. Cabe destacar que los precios unitarios pueden variar según el proveedor y el número de unidades.

	Precio unitario	Cantidad	Precio Total
ESP32 - DevKitC	11 €	1	11 €
Protoboard	2.99 €	1	2.99 €
DHT22	3.83 €	1	3.83 €
LDR	0.87 €	1	0.87 €
HC-SR04	2.94 €	1	2.94 €
YL69,YL38	1.95 €	1	1.95 €
PH Sensor	32 €	1	32 €
Display LCD I2C	7.89 €	1	7.89 €
Módulo 2 relés	2.03 €	1	2.03 €
Conversor de potencia salida a (3.3V, 5V, 12V)	8.23 €	1	8.23 €
Adaptador AC/DC de 12 V, 2A	11.99 €	1	11.99 €
Cables	2.78 € / Pack 40	1	2.78 €
Bomba de Agua 12V	9.99 €	1	9.99 €
Ventilador 12V	5.37 €	1	5.37 €
Coste total			**103.86 €**

Tabla 2 - Materias Primas

1.3. PUESTO DE TRABAJO

El cálculo del gasto en el puesto de trabajo se realiza teniendo en cuenta los gastos de amortización de las herramientas, así como el software, las licencias y el gasto energético. Una necesidad básica en el diseño y la investigación es el uso de herramientas e instrumentos de laboratorio que ocasionan gastos energéticos debido a su alimentación. Además, hay un gasto en la vida útil de los componentes.

Los equipos informáticos utilizados han sido de carácter personal y es el material con el que se ha realizado todo el trabajo, desde la investigación, hasta la programación final del dispositivo empleado. Para calcular el gasto en el puesto de trabajo, se tiene en cuenta el tiempo invertido en la realización del proyecto, ya que en todo momento se ha necesitado el apoyo de estas herramientas.

El consumo medio de energía en España viene a ser de 0.14 € el kW. A continuación se calcula el coste energético según la fórmula:

$$Coste\ Energético = Consumo\ Medio\ Energía * Potencia\ (kWh) * horas\ Trabajadas$$

$$Coste\ Energético = 1kWh * 0.14 \frac{€}{kWh} * 175h = 24.5\ €$$

Ecuación 1 - Coste energético

A continuación se calcula el coste de amortización de la instrumentación ya que, como se ha explicado previamente, se desgastan y deterioran por el uso y paso del tiempo. Para poder calcular esto, tendremos en cuenta el equipamiento utilizado siguiente:

Útil	Coste
Ordenador	855 €
Multímetro	14 €
TOTAL	**869 €**

Tabla 3 - Equipamiento puesto de trabajo

Estimando un ciclo de vida útil de 5 años y un tiempo de uso de 8 meses para el proyecto por lo que el coste de amortización instrumental calculado es:

$$Coste\ de\ Amortización = \frac{Tiempo\ de\ Uso}{Ciclo\ de\ Vida\ Útil}\ x\ Inversión\ Total\ Equipo$$

$$Coste\ de\ Amortización = \frac{8\ meses}{5\ años * 12\frac{meses}{año}}\ x\ 869\ € = 115.87\ €$$

Ecuación 2 - Coste de Amortización

Por otro lado se debe de tener en cuenta los gastos generados por el software utilizado en el proyecto y la programación del mismo. La mayor parte de los programas utilizados son de software libre, en caso contrario sería necesario solicitar licencias. En éste caso se podrían solicitar licencias de estudiante ya que el proyecto actual tiene exclusivamente fines académicos. El coste del puesto de trabajo se observa en la siguiente tabla:

Útil	Coste
Coste energético	24.5 €
Coste de amortización	115.87 €
Software y licencias	0 €
	140.37 €

Tabla 4 - Coste total del puesto de trabajo

Existen unos costes indirectos, basados en la Mano de Obra Indirecta, Gastos Generales y Gastos Sociales. Los gastos de Mano de Obra Indirecta, corresponden a gastos de personal administrativo que no participa activamente, los Gastos Generales son aquellos donde se incluyen el pago y mantenimiento de las instalaciones, y por último, en los Gastos Sociales, vendrían reflejados la formación de los trabajadores, la seguridad social y la responsabilidad civil.

Todos estos Costes Indirectos no se incluirán, ya que al tratarse de un proyecto basado en un prototipo, y al haber realizado el trabajo fuera de un entorno de laboratorio y/o empresarial, no nos permite estimar los gastos indirectos de una manera realista.

2. COSTE TOTAL DEL PROYECTO

A continuación se muestra una tabla resumen con los costes totales del presente proyecto:

	Coste
Mano de Obra Directa	**2800 €**
Materia Prima	**103.86 €**
Puesto de Trabajo	**140.37 €**
COSTE TOTAL DEL PROYECTO	**3044.23 €**

El Coste Total del Proyecto asciende a **tres mil cuarenta y cuatro euros con veintitrés céntimos.**