UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Sistema de Riego con Arduino

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: García Almiñana, Alberto

Tutor/a: Salido Gregorio, Miguel Angel

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SISTEMA DE RIEGO CON ARDUINO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

ALBERTO GARCIA ALMIÑANA

TUTORIZADO POR

Salido Gregorio, Miguel Ángel

CURSO ACADÉMICO: 2021/2022

Resumen

El presente trabajo fin de grado tiene como objetivo el diseño y desarrollo de un prototipo capaz de controlar el riego, tanto siendo capaz de gestionar el propio la necesidad del riego y de controlarse remotamente por medio de una comunicación inalámbrica.

El Arduino se comunica con el ambiente por medio de unos sensores que le brinda información sobre su alrededor: temperatura, humedad, presión, lluvia. Y en base a estos, decide los tiempos de riego o salvo en caso de lluvia, apagar el riego. Una bomba es gestionada por medio de un relé conectado al Arduino.

Ambos están enlazados entre sí, por medio de una comunicación en la frecuencia 868 MHz utilizando una tecnología llamada LPWAN. Existen diversos tipos en esta tecnología reciente, que en el documento Memoria se explica en más profundidad. El empleado dentro de este es la llamada LoRa o *Long Range* y debe cumplir con la normativa de la ETSI.

Además, las características propias de estos microcontroladores les permite comunicarse por I2C y SPI con los sensores y módulos empleados.

El desarrollo se ha realizado en dos microcontroladores con lenguaje en Arduino, reflejado en el anexo I. El código se ha desarrollado en el propio IDE de Arduino, aprovechando sus sencillas funcionalidades.

El desarrollo de este proyecto deberá ser fácil de transportar, por lo que será de dimensiones reducidas y ligero.

Palabras clave:

Prototipo, Arduino, I^2C, LoRa, código, mkr, bme, presión, temperatura, humedad

Resum

El present treball fi de grau té com a objectiu el disseny i desenvolupament d'un prototip capaç de controlar el reg, tant sent capaç de gestionar el propi la necessitat del reg i de controlar-se remotament per mitjà d'una comunicació sense fil.

El Arduino es comunica amb l'ambient per mitjà d'uns sensors que li brinda informació sobre el seu voltant: temperatura, humitat, pressió, pluja. I sobre la base d'aquests, decideix els temps de reg o excepte en cas de pluja, apagar el reg. Una bomba és gestionada per mitjà d'un relé connectat al *Arduino.

Tots dos estan enllaçats entre si, per mitjà d'una comunicació en la freqüència 868 MHz utilitzant una tecnologia anomenada *LPWAN. Existeixen diversos tipus en aquesta tecnologia recent, que en el document Memòria s'explica en més profunditat. L'empleat dins d'aquest és l'anomenada LoRa o Long Range i ha de complir amb la normativa de la ETSI.

A més, les característiques pròpies d'aquests microcontroladors els permet comunicar-se per I2C i SPI amb els sensors i mòduls emprats.

El desenvolupament s'ha realitzat en dos microcontroladors amb llenguatge en Arduino, reflectit en l'annex I. El codi s'ha desenvolupat en el propi IDE de Arduino, aprofitant les seues senzilles funcionalitats.

El desenvolupament d'aquest projecte haurà de ser fàcil de transportar, per la qual cosa serà de dimensions reduïdes i lleuger.

Paraules clau

Prototip, *Arduino, I2C, *LoRa, codi, *mkr, *bme, pressió, temperatura, humitat

Abstract

The aim of this final degree project is the design and development of a prototype capable of controlling irrigation, both being able to manage the need for irrigation itself and to be controlled remotely by means of wireless communication.

The Arduino communicates with the environment by means of sensors that provide it with information about its surroundings: temperature, humidity, pressure, rain. Based on these, it decides the watering times or, except in case of rain, turns off the watering. A pump is managed by a relay connected to the Arduino.

Both are linked to each other by means of communication on the 868 MHz frequency using a technology called LPWAN. There are several types of this recent technology, which are explained in more detail in the document Memory. The one used within this is called LoRa or Long Range and must comply with ETSI standards.

In addition, the characteristics of these microcontrollers allow them to communicate via I2C and SPI with the sensors and modules used.

The development has been carried out on two microcontrollers with Arduino language, as shown in appendix I. The code has been developed in the Arduino IDE, taking advantage of its simple functionalities.

The development of this project must be easy to transport, so it will be small and light.

Keywords

Prototype, Arduino, I2C, LoRa, code, mkr, bme, pressure, temperature, humidity.

DOCUMENTO Nº1 MEMORIA

DOCUMENTO Nº2 PLANOS

DOCUMENTO Nº3 PLIEGO DE CONDICIONES

DOCUMENTO Nº4 PRESUPUESTO

ANEXO I

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SISTEMA DE RIEGO CON ARDUINO

DOCUMENTO Nº 1. MEMORIA

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

ALBERTO GARCIA ALMIÑANA

TUTORIZADO POR

Salido Gregorio, Miguel Angel

CURSO ACADÉMICO: 2021/2022

ÍNDICE MEMORIA

ÍNDICE FIGURAS

ÍNDICE TABLAS

1. Objeto

El objetivo de este proyecto es el desarrollo de una aplicación de control de riego dotado de una serie de sensores para la obtención de datos del ambiente, y una comunicación inalámbrica y cuyos datos se almacenarán en una base de datos.

2. Justificación del proyecto

El surgimiento de nuevas tecnologías ha permitido la comunicación inalámbrica en muchos aspectos de la vida hasta ahora impensables. El surgimiento de las redes 4G o 5G recientemente, son útiles allí donde hay cobertura. En cambio, en otros terrenos, ya sea por la orografía o lejos de antenas de telefonía, se puede hacer uso de tecnologías en frecuencias para medias o largas distancias. La tecnología LoRa, *Long Range*, permite cubrir grandes distancias, de hasta 8 km fácilmente e incluso a través de software hacerlo llegar hasta los 20 km, más que suficiente para cubrir el objetivo del proyecto.

Pese a la existencia de aparatos programados para el riego, muchos de ellos no son capaces de comunicarse, mas que estando presente. Con tecnologías de comunicación inalámbrica se puede establecer conexiones con estos sin tener que estar presente.

3. Planteamiento de solución alternativas y justificación de la solución adoptada

Existen diferentes tipos de comunicaciones inalámbricas: WiFi, Bluetooth, SigFox, LoRa, redes móviles…

En un principio se planteó la posibilidad de utilizar una red móvil. Pero surge el inconveniente de que todavía en España hay zonas con puntos ciegos de cobertura. No solo es cuestión de cobertura, el consumo es un factor importante, ya que, si el dispositivo debe funcionar con pilas, las redes móvil consumen Y las señales como la WiFi o el bluetooth quedaban totalmente descartadas debido a la distancia de comunicación.

Por tanto, el proyecto se decanta por otras tecnologías, como es la LPWAN, “Low Power Wide Area Networks”, que permiten transmitir datos entre un dispositivo y una estación base/Gateway separados por centenares de metros o quilómetros con un muy bajo consumo energético. Dentro de existen varios estándares como es el SigFox, LoRa o NB-Iot.

3.1 SigFox

Sigfox es una tecnología LPWAN de operador, es decir, existe una compañía llamada Sigfox que se ha encargado de desplegar una

4. Objeto

El objetivo de este proyecto es el desarrollo de una aplicación de control de riego dotado de una serie de sensores para la obtención de datos del ambiente, y una comunicación inalámbrica y cuyos datos se almacenarán en una base de datos.

5. Justificación del proyecto

El surgimiento de nuevas tecnologías ha permitido la comunicación inalámbrica en muchos aspectos de la vida hasta ahora impensables. El surgimiento de las redes 4G o 5G recientemente, son útiles allí donde hay cobertura. En cambio, en otros terrenos, ya sea por la orografía o lejos de antenas de telefonía, se puede hacer uso de tecnologías en frecuencias para medias o largas distancias. La tecnología LoRa, *Long Range*, permite cubrir grandes distancias, de hasta 8 km fácilmente e incluso a través de software hacerlo llegar hasta los 20 km, más que suficiente para cubrir el objetivo del proyecto.

Pese a la existencia de aparatos programados para el riego, muchos de ellos no son capaces de comunicarse, mas que estando presente. Con tecnologías de comunicación inalámbrica se puede establecer conexiones con estos sin tener que estar presente.

6. Planteamiento de solución alternativas y justificación de la solución adoptada

Existen diferentes tipos de comunicaciones inalámbricas: WiFi, Bluetooth, SigFox, LoRa, redes móviles…

En un principio se planteó la posibilidad de utilizar una red móvil. Pero surge el inconveniente de que todavía en España hay zonas con puntos ciegos de cobertura. No solo es cuestión de cobertura, el consumo es un factor importante, ya que, si el dispositivo debe funcionar con pilas, las redes móvil consumen Y las señales como la WiFi o el bluetooth quedaban totalmente descartadas debido a la distancia de comunicación.

Por tanto, el proyecto se decanta por otras tecnologías, como es la LPWAN, “Low Power Wide Area Networks”, que permiten transmitir datos entre un dispositivo y una estación base/Gateway separados por centenares de metros o

quilómetros con un muy bajo consumo energético. Dentro de existen varios estándares como es el SigFox, LoRa o NB-Iot.

6.1 SigFox

Sigfox es una tecnología LPWAN de operador, es decir, existe una compañía llamada Sigfox que se ha encargado de desplegar una infraestructura de grandes antenas por toda España que dar cobertura a todo el territorio. Cualquier dispositivo puede, pagando una suscripción, utilizar esta red para la transmisión de sus datos siempre que se cumplan las normas de uso establecidas por Sigfox. Como operador, ofrece no solo la red y su mantenimiento, sino que también dejan disponibles en su *backend* (accesible vía API) los datos enviados por los nodos. En un sistema que utilice Sigfox, los datos siempre pasan por su infraestructura.

Cabe destacar que tienen el gran mérito de ser la primera gran tecnología LPWAN desplegada en España y la que, actualmente, en más proyectos se ha utilizado. Sus pros funcionales son la cobertura que ofrecen, lo fácil que es subscribirse y su total accesibilidad. Por el contrario, es una tecnología que puede enviar pocos mensajes al día (máximo de 1 cada 10 minutos de subida y solo 4 diarios de bajada) y de pocos bytes cada uno.

6.2 NB-Iot

NB-IoT es una tecnología LPWAN de operador ofrecida por las mismas compañías de telecomunicaciones que ofrecen las redes de telefonía móvil, es decir, es la tecnología que las Telecom ofrecen al ecosistema IoT.

Además de ciertas diferencias técnicas, la principal diferencia funcional con Sigfox es que no están desplegadas ni accesibles grandes redes NB-IoT por todo el territorio. El estado actual de esta tecnología es que, si el proyecto es interesante en cuanto a volumen de equipos, la operadora en cuestión (Movistar, Vodafone, Orange, …) monta una red dedicada a ese proyecto. Ahora bien, si el proyecto es pequeño, no hay una red a la cuál suscribirse. Se espera que esto cambie en los próximos años.

6.3 LoRaWan

LoRaWAN es un estándar desarrollado sobre la modulación radio LoRa con una alta implantación en Europa. Si bien es cierto que puede ser una red de operador ofrecida por varias compañías en un mismo territorio, aquí en España no hay aun operadores que ofrezcan cobertura en todo el territorio. Pero lo interesante de LoRaWAN es que, a diferencia de las demás LPWAN, permite el despliegue de redes propias autogestionadas. En una red autogestionada, los límites son los que marca la ETSI, que son mucho menos restrictivos que los límites que impone un operador privado. Este hecho abre un gran abanico de posibilidades para realizar iniciativas IoT de ámbito local para control de

áreas pequeñas o medianas (un edificio, una fábrica, un puerto, una ciudad, etc.). Evidentemente requieren conocer un poco más técnicamente la tecnología y tener que gestionar la red, pero permite montar redes en cualquier lugar – las redes de operador difícilmente llegarán a todos los interiores de un edificio – y unos costes de mantenimiento menores.

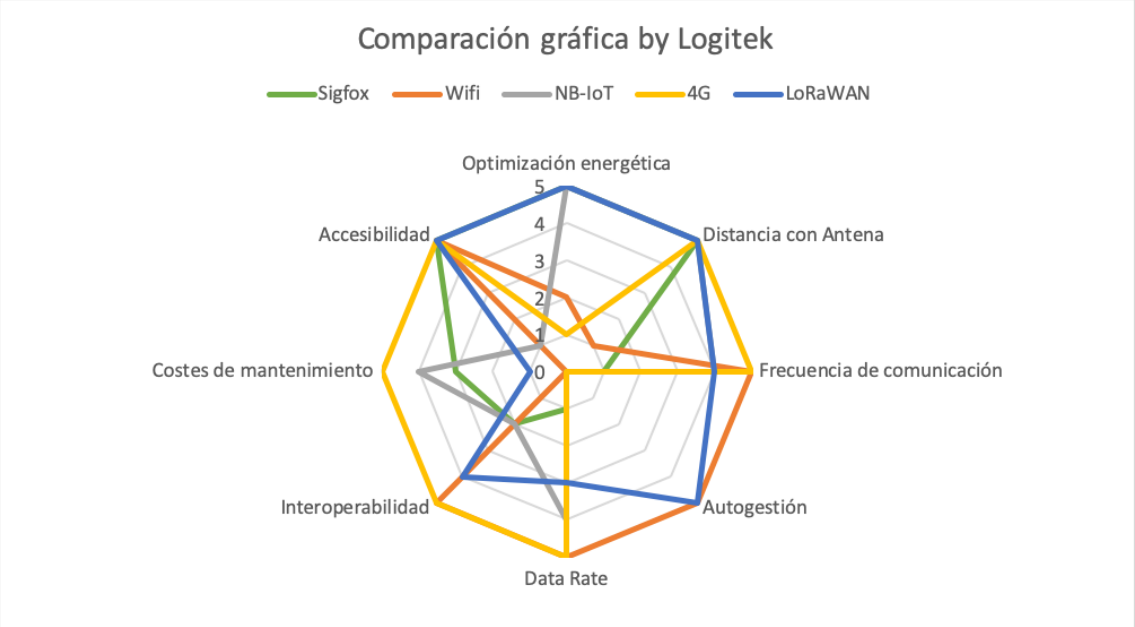


Fig. 1. Comparativa del uso de las redes WAN y móviles.

7. Diseño del prototipo

7.1 Arduino MKR WAN 1310

Se basa en el Microchip® SAMD21 de 32 bits, el módulo Lora® Murata CMWX1ZZABZ y el chip criptográfico ECC508. MKR WAN 1310 incluye un cargador de batería, una memoria flash SPI 2 Mbyte y un consumo muy reducido en reposo de aproximadamente 100 µA.

Esta placa de código abierto se puede conectar a: La Cloud IoT de Arduino, su propia red Lora® mediante Arduino Lora® PRO Gateway, la infraestructura LoRaWAN™ existente como The Things Network, o incluso otras placas mediante el modo de conectividad directa.

Puede ser capaz de alimentarse a 5 V, ya que lleva un convertidor de voltaje a 3,3 V. Todos los pines de este son entradas y salidas a 3,3 V.

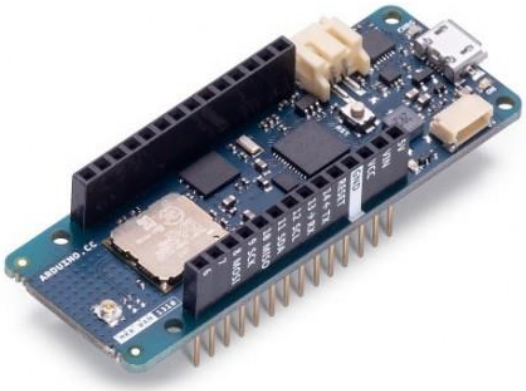


Fig. 2. Arduino MKR WAN 1310.

7.2 Sensor BME280

El sensor BME280 integra en un solo dispositivo sensores de presión atmosférica, temperatura y humedad relativa, con gran precisión, bajo consumo energético y un formato ultra compacto. Basado en tecnología BOSCH piezo-resistiva con gran robustez EMC, alta precisión y linealidad, así como con estabilidad a largo plazo. Se conecta directamente a un microcontrolador a través de I2C o SPI. Algunas de sus características más destacables son:

- Bajo Voltaje de Operación: 3.3 V DC o 5 V
- Rango de Presión: 300 a 1100 hPa (0.3-1.1bar)
- Precisión absoluta: 1 hPa
- Rango de Temperatura: -40°C a 85°C
- Rango de Humedad Relativa: 0-100% RH

Fig. 3. Módulo BME280.

7.3 Detector de lluvia FC-37

Constructivamente son sensores sencillos. Se dispone de dos contactos, unidos a unas pistas conductoras entrelazadas entre sí a una pequeña distancia, sin existir contacto entre ambas. Al depositarse agua sobre la superficie, se pone en contacto eléctrico ambos conductores, lo que puede ser detectado por un sensor. Realiza la medición con el comparador LM393, que permite obtener la lectura tanto como un valor analógico como de forma digital cuando se supera un cierto umbral, que se regula a través de un potenciómetro ubicado en la propia placa.

Fig. 4. Placa resistiva y módulo LM393.

7.4 Reloj externo DS3231

Un reloj de tiempo real (RTC) es un dispositivo electrónico que permite obtener mediciones de tiempo en las unidades temporales que empleamos de forma cotidiana. Está formado por un resonador de cristal integrado con la electrónica necesaria para contabilizar de forma correcta el paso del tiempo. el DS3231 incorpora medición y compensación de la temperatura garantizando una precisión de al menos 2ppm, lo que equivale a un desfase máximo 172 ms/día o un segundo cada 6 días. En el mundo real normalmente consiguen precisiones superiores, equivalente a desfases de 1-2 segundos al mes.

La comunicación se realiza a través del bus I2C. La tensión de alimentación es 2.3 a 5.5 V.

Para las pruebas se ha utilizado el módulo que lleva integrado una pequeña EEPROM AT24C32 y una pequeña batería CR2032 para mantener la hora, aun cuando no tiene alimentación directa por parte del Arduino o externa. Como el de la siguiente figura:

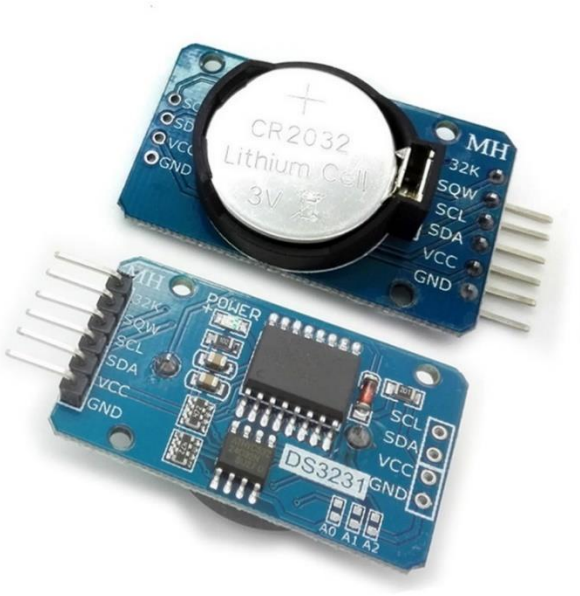


Fig. 5. Módulo DS3231.

7.5 Relé

Un relé es un interruptor, más frecuentemente electromagnético, que utiliza una pequeña corriente para accionar un circuito mayor. Básicamente, se aplica una señal en la entrada que enciende otro circuito conectado en la salida.

Se ha utilizado un relé de 3.3 V de 1 canal de hasta 10 A a 230 V en alterna. Ya que, en cada campo, puede variar la potencia del motor, el único requerimiento es que sea accionable a 3.3 V. Un ejemplo de real y fiable sería el de la marca Songle con un modulo preparado para conexiones de prototipos en Arduino.

Hay que decir que se requiere un adaptador de nivel, que el documento plano, se puede encontrar.

Fig. 6. Módulo relé de la marca Songle.

O también se puede utilizar un modulo ya adaptado a la señal con un elevador. Como el de la marca HeliShun.

Fig. 7. Módulo relé de HeliShun.

7.6 Arduino Wemos D1 Mini ESP8266

Wemos D1 mini ESP8266 es una plataforma de desarrollo similar a Arduino especialmente orientada al Internet de las cosas (IoT). La placa Wemos D1 Mini ESP8266 tiene como núcleo al SoM ESP-12E que a su vez está basado en el SoC Wi-Fi ESP8266, integra además el conversor USB-Serial TTL CH340G y conector micro-USB. Posee un regulador de voltaje de 3.3V en placa.

Integra el Tensilica Xtensa LX3, un potente microcontrolador con arquitectura de 32 bits. El SoM (*System on Module*) ESP-12E fabricado por Ai-Thinker integra en un módulo el SoC ESP8266, memoria FLASH, cristal oscilador y antena WiFi en PCB. Tiene un consumo corriente promedio de 70 mA.

Como dato adicional, la plataforma ESP8266 permite el desarrollo de aplicaciones en diferentes lenguajes como: Arduino, Lua, MicroPython, C/C++, Scratch.

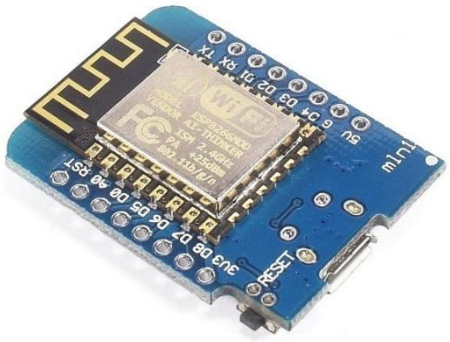


Fig. 8. Arduino Wemos D1 Mini.

7.7 Transceptor LoRa RFM95

Es un modulo inalámbrico basado en el chip LoRa RFM95W que emite a 868 o 915 MHz. Se alimenta tanto a 3.3 como a 5 V, pero la comunicación con el Arduino debe hacerse a 3.3 V.

Tiene una potencia de hasta 100 mW configurable por software de +13 a 20 dBm. Tiene un consumo reducido de entre 50 mA a 150 mA, según la potencia de envío y 30 mA en escucha. Soporta encriptación del tipo AES-128.

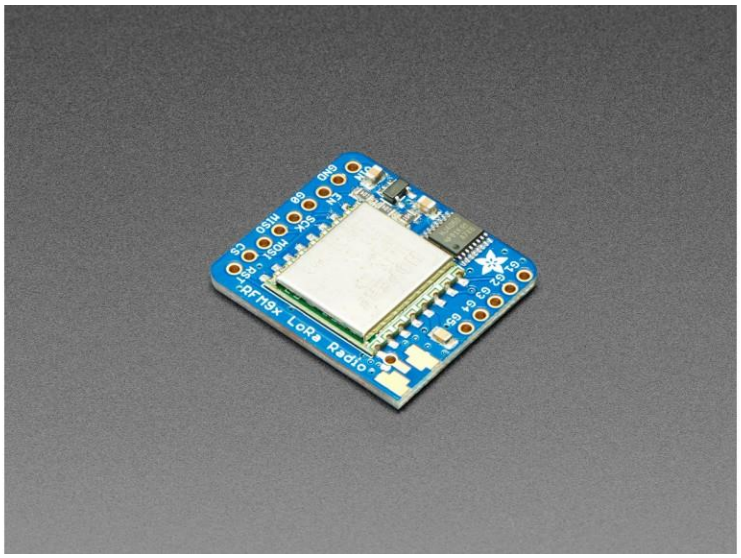


Fig. 9. Módulo RFM95.

8. Software desarrollo

El diseño del código se ha realizado en el IDE de Arduino. El IDE de Arduino nos permite escribir, depurar, editar y grabar nuestro programa (llamados “sketches” en el mundo Arduino) de una manera sumamente sencilla, en gran parte a esto se debe el éxito de Arduino, a su accesibilidad.

Contiene no solo una gran cantidad de librerías, tanto suministradas por compañías de software como por usuarios expertos y aficionados a este mundo.

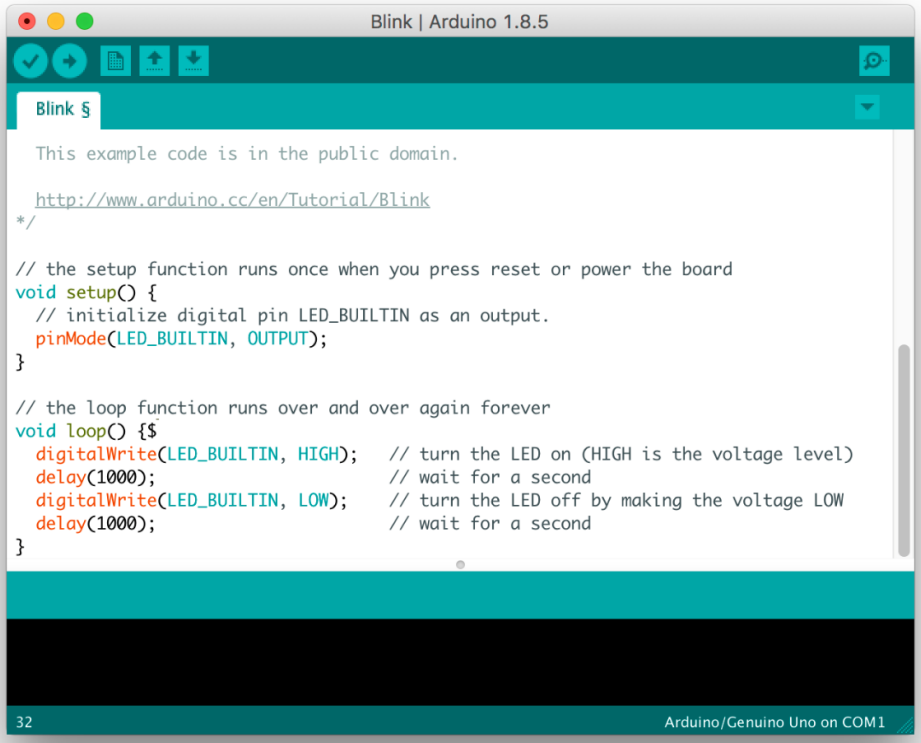


Fig. 10. Ventana del software Arduino 1.8.

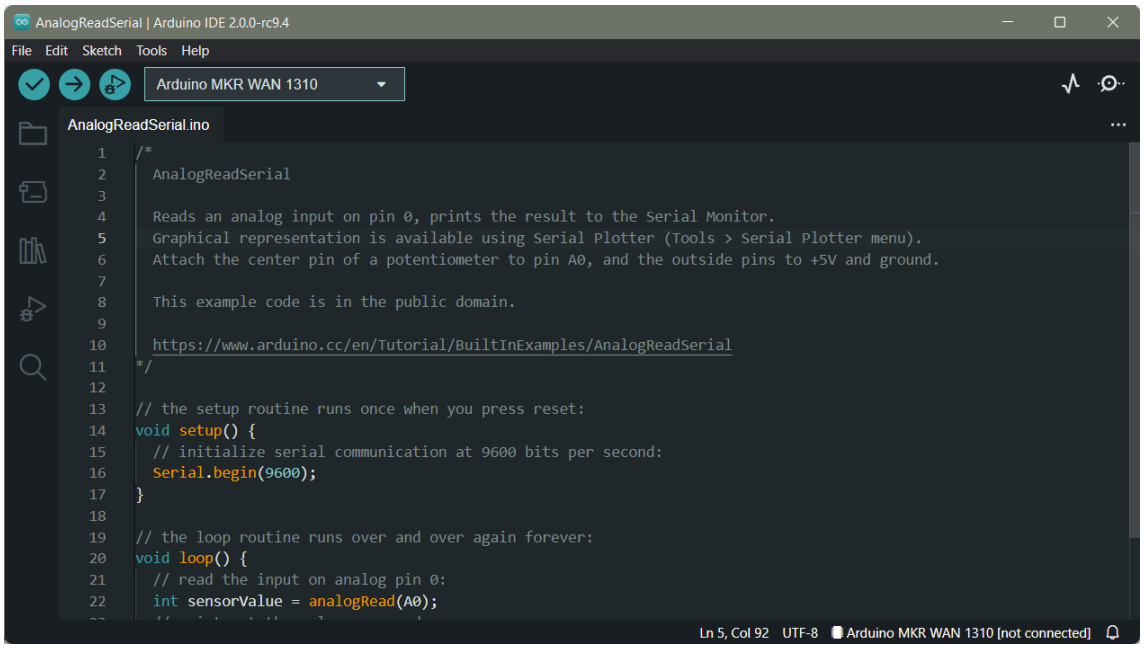


Fig. 11. Software Arduino IDE 2.

9. Variables ambientales

En este proyecto, la toma de datos ambientales tiene un papel muy relevante y es, de hecho, uno de los aspectos diferenciadores respecto a otros sistemas de riego automatizados.

Como se detallará en capítulos sucesivos, dicha toma de datos tiene como fin el cálculo del parámetro denominado evapotranspiración, que indica la cantidad de agua que se pierde del conjunto suelo - planta por evaporación y transpiración, con lo que se puede obtener una buena estimación de las necesidades hídricas del cultivo.

A continuación, se expondrán las variables ambientales que han sido consideradas a lo largo del presente proyecto, así como las que han sido descartadas, y las características de los sensores utilizados para tal fin.

9.1 Temperatura

La temperatura es uno de los parámetros más importantes a tener en cuenta en un sistema de control de riego como el que se aborda, e incluso es posible realizar una estimación aceptable de la evapotranspiración a partir únicamente de los valores de temperatura máximo y mínimo.

9.2 Temperatura del suelo

La temperatura del suelo agrícola condiciona los procesos microbianos que tienen lugar en el mismo. La temperatura también influye en la absorción de los nutrientes y en los procesos bióticos y químicos [1].

En un principio, se propuso la temperatura del suelo como otra variable más para ser monitorizada, pero, finalmente, se llegó a la conclusión de que la utilización de estos datos no sería relevante para la determinación de las necesidades de riego del cultivo.

9.3 Humedad relativa

La humedad relativa (RH) es la relación entre la presión parcial del vapor de agua y la presión de vapor de equilibrio del agua a una temperatura dada. La humedad relativa depende de la temperatura y la presión del sistema de interés. La misma cantidad de vapor de agua produce una mayor humedad relativa en el aire frío que en el aire caliente [1].

La humedad relativa es, junto con la temperatura, uno de los parámetros más importantes para llevar a cabo una buena estimación de la evapotranspiración del cultivo.

9.4 Humedad del suelo

Durante las primeras etapas de desarrollo de este proyecto se estudió la posibilidad de utilizar la medida de la humedad del suelo como el parámetro más relevante para el algoritmo de control de riego. Sin embargo, después de consultar con expertos en tierras agraria, la recomendación fue no utilizar este tipo de sensores resistivos, pues el parámetro de la conductividad no depende únicamente de la humedad, sino también de otros factores como el tipo de tierra, las sales disueltas y la cantidad de materia orgánica existente. Esto unido a la poca calidad del sensor, FC-28, y su baja resistencia a la corrosión, han llevado a descartar su uso en el presente proyecto.

9.5 Presión atmosférica

La variación de la presión a lo largo del tiempo permite obtener una información útil que, unida a otros datos meteorológicos (temperatura atmosférica, humedad y velocidad del viento), puede dar una imagen bastante acertada del tiempo atmosférico e incluso un pronóstico a corto plazo de este.

Los datos de la presión atmosférica no se utilizan directamente en el cálculo de la evapotranspiración. En un principio, se intentó aprovechar dicha información para implementar un sistema con el cual obtener una predicción relativamente fiable de la posibilidad de precipitaciones, basado en las variaciones de la presión y acompañado de otros

parámetros ambientales como la temperatura y la humedad. Al igual que en el caso anterior, se acudió al consejo de expertos en el área de observación de la atmósfera y meteorología, los cuales desaconsejaron la realización de dicho sistema debido a la complejidad de los algoritmos de predicción necesarios y la poca exactitud que se podría obtener.

A pesar de ello, se ha decidido mantener este sensor de modo informativo, aunque no se utilice en ninguno de los aspectos del control.

9.6 Detección de lluvia

En este proyecto se optó por hacer uso de un sensor, FC-37, detector de precipitaciones con el fin de utilizar estos datos en el sistema de control, evitando así el riego en esas circunstancias.

10. Alimentación

El sistema necesita energía eléctrica para funcionar, por lo que se analizaron diferentes alternativas.

10.1 Arduino Wemos D1 Mini

Al Arduino conectado en casa, lo más sencillo es conectar a la luz con cualquier cable con micro-usb y un adaptador de corriente AC/DC. Suponiendo que la placa, situándonos en el peor caso de consumo, consume 400 mA y el modulo de radiofrecuencia 150 mA. Cualquier adaptador capaz de ofrecer aproximadamente 550 mA o más.

	Consumo promedio mA	Consumo reposo mA	Consumo máximo mA
Arduino Wemos D1 Mini	70	0,04	400
Módulo RFM95	30	30	150

Tabla 1. Datos de consumo.

Se puede alimentar directamente el Arduino y de ahí el mismo puede alimentar el módulo externo.

10.2 Arduino MKR WAN 1310

Existen dos posibilidades de uso: disponer o no de corriente electrica. Este trabajo se ha realizado suponiendo que existe alguna

alimentacion propia en la zona de riego. Por tanto, se utilizará un adaptador y cable micro-usb similar al anterior.

Si bien, el microcontrolador lleva una salida directa de 3.3 y 5 V. Para evitar que el regulador de voltaje se pueda dañar, se hará uso de una alimentacion externa que dé la energia a todo el sistema. Y sea capaz de dar una corriente superior a 500 mA.

	Consumo promedio mA	Consumo reposo mA	Consumo máximo mA
Arduino MKR WAN 1310	30	0,104	250
Relé	150	0	150
FC-37	2	2	50
BME280	0,0003	0,0003	0,714

Tabla 2. Datos de consumo II.

11. Evapotranspiración

En este apartado se detallará el método utilizado para determinar las necesidades hídricas del cultivo, y con ello poder implementar el algoritmo para el control del riego.

Dicho método se basa en la estimación del parámetro denominado Evapotranspiración, que da cuenta de la cantidad de agua que se pierde a través de la superficie del suelo por evaporación y por transpiración de las plantas.

11.1 Evaporación

La evaporación es el proceso por el cual el agua líquida se convierte en vapor de agua y se retira de la superficie evaporante. Para que ocurra este cambio de fase es necesario un aporte de energía externo, que principalmente es la radiación solar directa, y en menor grado la temperatura del aire **[1]**.

La diferencia entre la presión de vapor del suelo y de la atmosfera, definen la velocidad a la que ocurre este proceso. A medida que ocurre la evaporación, el aire circundante se va saturando y el proceso se ralentiza. Para que continúe es importante que exista una renovación de este aire húmedo por otro más seco, para lo que influye enormemente la velocidad del viento.

11.2 Transpiración

La transpiración se fundamenta en la pérdida mediante vaporización, del agua contenida en los tejidos vegetales. Ésta ocurre principalmente en las hojas de las plantas, y de mismo modo que en la evaporación, depende de la radiación, la temperatura ambiente, la humedad atmosférica y el viento. Otros factores involucrados son el estado de desarrollo de la planta, así como las características del suelo y del agua de riego, que determinan la facilidad con la que ésta será absorbida por las raíces.

Como se ha comprobado, estos dos fenómenos descritos anteriormente ocurren de forma simultánea y se encuentran estrechamente relacionados. En las primeras fases del desarrollo del cultivo, el factor predominante en la pérdida de agua es la evaporación, pero con la evolución de la planta, y el desarrollo de las hojas, la transpiración cobra mayor importancia hasta convertirse en el proceso principal.

En la figura siguiente se detalla está relación entre los dos fenómenos a lo largo del crecimiento del cultivo.

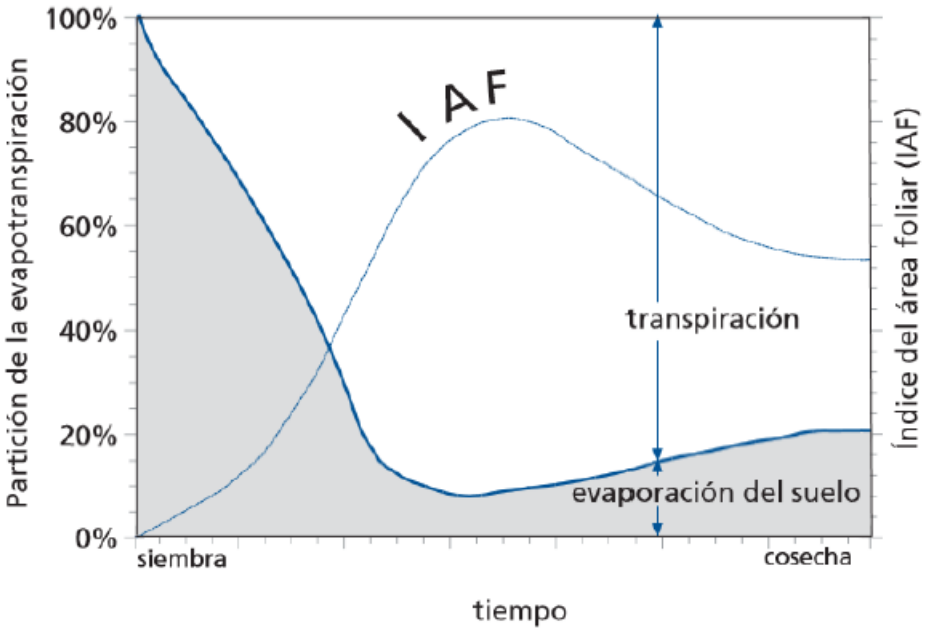


Fig. 12. Relación de la evapotranspiración.

Las unidades más utilizadas para expresar la evapotranspiración son los milímetros (mm) por unidad de tiempo. Representan la altura de agua perdida en un periodo, que puede ser desde una hora hasta varios días. Otra forma de expresarlo es en m^3 por hectárea, de forma que una pérdida de 1 mm de agua equivale a 10 m^3 por hectárea y día ($m3/ha/día$).

La evapotranspiración no depende únicamente de las variables climáticas, también hay que tener en consideración otros aspectos como los factores de cultivo, entre los que se encuentra el tipo de planta, su altura, etapa de desarrollo y el manejo-condiciones ambientales como la salinidad o fertilidad de la tierra, presencia de enfermedades y parásitos, la cubierta del suelo y las prácticas de cultivo y método de riego.

Así pues, en base a los factores comentados, se tienen tres tipos de evapotranspiración: evapotranspiración del cultivo de referencia (ET_O), evapotranspiración del cultivo bajo condiciones estándar (ET_C) y evapotranspiración del cultivo bajo condiciones no estándar (ET_C aj), como se puede apreciar en la figura inmediata.

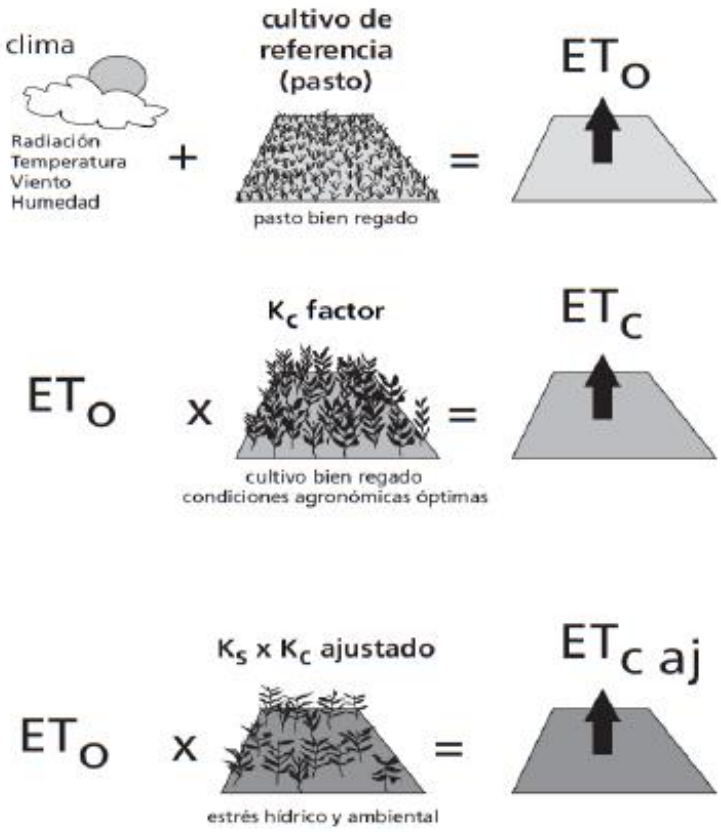


Fig. 13. Diagrama de la evotranspiración.

11.3 Evapotranspiración del cultivo de referencia

La evapotranspiración del cultivo de referencia o ET_O, se emplea para determinar la demanda de evapotranspiración de la atmosfera de forma independiente al tipo de cultivo y condiciones de manejo. Para ello se utiliza como referencia para el cálculo una «superficie extensa de pasto verde, bien regada, de altura uniforme, creciendo activamente y dando sombra totalmente a un suelo moderadamente seco que recibe riego con una frecuencia semanal aproximadamente» [11]. De este modo se

consigue que este parámetro sólo dependa de las variables climáticas, por lo que puede ser obtenido a partir de datos meteorológicos.

11.4 Evapotranspiración del cultivo bajo condiciones estándar

También llamada ET_C, este parámetro refleja la evapotranspiración de un cultivo en óptimas condiciones, libre de enfermedades, bien fertilizado y regado.

Las necesidades de riego de un cultivo es la diferencia entre la cantidad de agua requerida para compensar la evapotranspiración y la precipitación efectiva.

La relación existente entre ET_C y ET_O puede ser determinada experimentalmente y es conocida como Coeficiente del Cultivo (K_C), que es utilizado para relacionar ETC con ET_O de manera que:

$$ETC = KC \cdot ETO$$

11.5 Evapotranspiración del cultivo bajo condición no estándar

La evapotranspiración del cultivo bajo condiciones no estándar da cuenta de la evapotranspiración en cultivos que no se encuentran bajo las condiciones óptimas, como exceso o falta de agua, presencia de plagas, o baja fertilidad del suelo, factores que pueden contribuir en un desarrollo deficiente de las plantas, y así en un distinto valor de ET_C. Por ello, para obtener el valor de la evapotranspiración ajustado se utiliza un coeficiente K_S (estrés hídrico), y el K_C modificado, de modo que reflejen las condiciones reales del cultivo.

Para el proyecto, se han supuesto los siguientes datos: uso de cítricos con crecimiento maduro. Aunque, esto se puede modificar o incluir una parte del código donde se puedan modificar estos parámetros según la especie de la planta.

11.6 Cálculos de la evapotranspiración de referencia

Como se ha comentado, la ETO se puede calcular utilizando datos meteorológicos. Para ello según la FAO (Organización de las Naciones Unidas para la Alimentación y la Agricultura), se recomienda utilizar únicamente el método de FAO Penman-Monteith [11], que requiere datos de radiación, temperatura del aire, humedad atmosférica y velocidad del viento.

El método utilizado está basado en una formulación anterior de la ecuación de Penman-Monteith desarrollada a partir de la combinación del balance energético con el método de transferencia de masa, y ampliada posteriormente con unos factores de resistencia que describen la dificultad del flujo de vapor de la planta y suelo hacia la atmosfera.

Dicha ecuación fue revisada por un conjunto de expertos en mayo de 1990, dando lugar a una versión modificada de ésta, que aumenta la precisión respecto al método anterior, dando lugar a la nueva ecuación estándar FAO Penman-Monteith [3]. La ventaja de esta nueva formulación es que es capaz de proporcionar valores aceptables de la ETO, incluso cuando no se dispone de todos los datos necesarios, ya que se puede obtener una estimación de éstos a partir de otros parámetros.

$$ET_O = \frac{0{,}408\Delta(R_n - G) + \gamma \frac{900}{T+273} u_2 (e_s - e_a)}{\Delta + \gamma(1 + 0{,}34 \cdot u_2)}$$

11.6.1 Obtener la latitud en radianes del lugar bajo estudio

En este caso, está situado en los alrededores de Alzira: 39°09'45,4"N 0°27'06,5"W. Este valor se debe pasar a grados y posteriormente a radianes.

11.6.2 Cálculo de la radiación extraterrestre

Se utiliza la siguiente expresión:

$$R_a = \frac{24 \cdot 60}{\pi} G_{sc} \cdot d_r [\omega_S \cdot \sin\varphi \cdot \sin\delta + \cos\varphi \cdot \cos\delta \cdot \sin\omega]$$

Donde:

R_a: radiación extraterrestre [MJ m^{-2} dia^{-1}]

G_{sc}: constante solar = 0,082 [MJ m^{-2} min^{-1}]

d_r: distancia relativa inversa Tierrra-Sol

ω_s: ángulo de radiación a la puesta del sol [rad]

φ: latitud [rad]

δ: declinación solar [rad]

Se deben obtener unos factores previos:

11.6.2.1 Distancia relativa inversa Tierra-Sol y declinación solar

Estos parámetros vienen dados por:

$$d_r = 1 + 0{,}033 \cdot \cos\left(2 \cdot \frac{\pi}{365} \cdot J\right)$$

$$\delta = 0{,}409 \cdot \sin\left(2 \cdot \frac{\pi}{365} \cdot J - 1{,}39\right)$$

Donde J representa el número del día en el total de días del año, es decir, su valor varía entre 1 y 365.

11.6.2.2 Angulo de radiación de la puesta del sol

Se emplea:

$$\omega_s = \cos^{-1}(-\tan\varphi \cdot \tan\delta)$$

Una vez que se tiene Ra, se puede obtener el valor de Rs mediante la ecuación de Radiación de Hargreaves **[12]**.

$$R_S = k_{rs}\sqrt{(T_{max} - T_{min})} \cdot R_a$$

Donde:

R_a: radiación extraterrestre [MJ m^{-2} dia^{-1}]

T_{max}: temperatura máxima [ºC]

T_{min}: temperatura mínima [ºC]

k_{rs}: coeficiente de ajuste (0,16…0,19) [$ºC^{-0.5}$]

11.6.2.3 Obtención de la radiación solar en día despejado

Para se utiliza:

$$R_{SO} = (0{,}75 + 2 \cdot 10^{-5} \cdot z) \cdot R_a$$

Donde z hace referencia a la altura sobre el nivel del mar en metros.

11.6.2.4 Obtención de la radiación de onda corta

Viene dada por:

$$R_{ns} = (1 - \alpha) \cdot R_a$$

11.6.2.5 Cálculo de la radiación neta de onda larga

Finalmente el ultimo parámetro necesario para la obtención de la radiación neta, se halla mediante:

$$R_{nl} = \sigma\left[\frac{T_{max,K}^4 + T_{min,K}^4}{2}\right](0{,}34 - 0{,}14\sqrt{e_a})\left(1.35 \cdot \frac{R_S}{R_{SO}} - 0{,}35\right)$$

Dónde:

R_{nl}: radiación neta de onda larga [MJ m^{-2} dia^{-1}],

σ: constante de Stefan-Boltzmann [4,903 x 10-9 MJ K^{-4} m^{-2} dia^{-1}]

$T_{max,K}$: temperatura máxima absoluta durante un periodo de 24 horas[1]

$T_{min,K}$: temperatura mínima absoluta durante un periodo de 24 horas

e_a: presión de vapor real [kPa],

R_s/R_{so}: radiación relativa de onda corta (valores ≤ 1,0)

11.6.2.6 Presión de vapor real

[1] Temperatura expresada en grados Kelvin. K = ºC + 273,16

El único valor desconocido en la ecuación anterior es la Presión de Vapor Real que se calcula a partir de la fórmula que se presenta seguidamente, en la que se hace uso de los valores de la humedad relativa.

$$e_a = \frac{e_{O,Tmin} \cdot \frac{HR_{max}}{100} + e_{O,Tmax} \cdot \frac{HR_{min}}{100}}{2}$$

Donde:

e_a: presión real de vapor [kPa]

$e_{o,Tmin}$: presión de saturación de vapor a la temperatura mínima diaria [kPa]

$e_{o,Tmax}$: presión de saturación de vapor a la temperatura máxima diaria [kPa]

HR_{max}: humedad relativa máxima [%]

HR_{min}: humedad relativa mínima [%]

El parámetro de la presión de saturación (eo) se obtiene de:

$$e_{O,T} = 0{,}6108 \cdot e^{\left[\frac{17.27 \cdot T}{T+237{,}3}\right]}$$

El fin último de todo este proceso, es la obtención de los valores de la evapotranspiración de referencia ETO, para ello es preciso hallar unos parámetros adicionales.

- Flujo de calor del suelo (G): Para estimaciones diarias del ETO, este valor se puede aproximar a cero.
- Presión de vapor de saturación (es):Se obtiene como la media entre la presión de saturación de vapor a la temperatura máxima y a la temperatura mínima.

$$e_s = \frac{e_{O,Tmin} + e_{O,Tmin}}{2}$$

- Pendiente de la curva de presión de vapor. Para el cálculo se utiliza la temperatura media.

$$\Delta = \frac{4908 \cdot 0{,}6108 \cdot e^{\left(\frac{17.27 \cdot T}{T+237.3}\right)}}{(T + 237{,}3)^2}$$

- Constante psicométrica. Para este parámetro existen valores de γ tabulados en función de la altitud. Para el caso en concreto que nos ocupa, la altitud sobre el nivel del mar es de 15 m.

z m	Γ kPa/ºC
0	0,067
100	0,067
200	0,066
300	0,065

Tabla 3. Valores de la constante psicométrica.

Por último, ya se está en condiciones de aplicar la fórmula de la FAO Penman-Monteith, para el cálculo de la ETO.

La evapotranspiración del cultivo de referencia sirve de marco de comparación para diversos periodos del año y distintos lugares, proporcionando un valor estándar e independiente al tipo de cultivo y a las condiciones de manejo. Por ello se debe ajustar mediante un coeficiente Kc, que da cuenta del tipo de planta y su estado de desarrollo, para finalmente obtener el valor de la evapotranspiración bajo condiciones estándar (ET_C).

A continuación, en la tabla siguiente se presentan algunos de los valores del coeficiente de cultivo K_C utilizados en el presente proyecto. Para una lista más exhaustiva acudir al documento de la FAO [III].

Cultivo	Kc inicio	Kc medio	Kc fin
Manzanas, cerezas, peras	0,6	0,95	0,75
Albaricoque, Melocotón, Nectarinas	0,55	0,9	0,65
Aguacate	0,6	0,85	0,75
Cítricos	0,85	0,85	0,85
Hortalizas pequeñas (Lechugas, zanahorias, etc.)	0,7	1,05	0,95
Raíces y tubérculos	0,5	1,1	0,95
Leguminosas (judías, guisantes, etc.)	0,4	1,15	0,55

Tabla 4. Valores del coeficiente de cultivo Kc

La cantidad de agua requerida para compensar la pérdida por evapotranspiración del cultivo se define como la necesidad de agua del cultivo, por lo que la necesidad hídrica neta (NH_n) básicamente representa la diferencia entre el requerimiento de agua del cultivo (ET_C) y la precipitación efectiva (PE).

$$NH_n = ET_c - PE$$

Este es finalmente el parámetro que será utilizado en el sistema de control implementado.

Dicho valor viene dado en milímetros por día (mm/día), aunque también se puede expresar en litros por metro cuadrado (l/m2), de modo que teniendo en cuenta el marco de plantación (densidad de plantas por metro cuadrado), se puede conocer la cantidad necesaria para cada una.

12. Código

Se ha mencionado anteriormente, el programa sobre el que se ha hecho el diseño es el IDE de Arduino. Se han desarrollado dos códigos distintos para cada Arduino.

Por un lado, el código sobre el Wemos D1 mini que actuará como servidor web y podrá ser colocado en un punto óptimo de la vivienda. Por otro lado, está el Arduino MKR WAN 1310 que estará situado en el campo actuando como controlador del riego.

Además, teniendo en cuenta que el Arduino en casa puede estar conectado permanentemente a internet, se pude abrir un puerto del propio router y conociendo la ip publica, se puede conectar desde cualquier sitio.

12.1 Código del Arduino MKR WAN 1310

El código tiene la capacidad de decidir sobre la hora de riego y la decisión de si regar o no, dependiendo de la lluvia. Además, de la comunicación con el otro Arduino.

Por un lado, se comprueba continuamente, actuando como una interrupción la recepción de mensajes desde el otro Arduino a través de la comunicación LoRa. Y se pueden dar dos casos, en la recepción donde se pida que le envíe los datos actuales de los sensores o donde se le pida que establezca una nueva hora de riego. Ocurre lo mismo para el envío de datos, uno es el ya mencionado de enviar los datos: temperatura, presión, humedad o si hay lluvia, y el otro envío se utiliza para en caso de error con el sensor BME280 o el DS3231, se envíe un código de error.

Por otro lado, se comprueba también, si la hora de riego es la correcta, entonces activa el relé del motor.

Además, el uso del DS3231 no solo permite comprobar la hora, sino que se ha programa una interrupción que le llega al Arduino para actualizar los datos de los sensores. Esto ocurre cada 60 segundos.

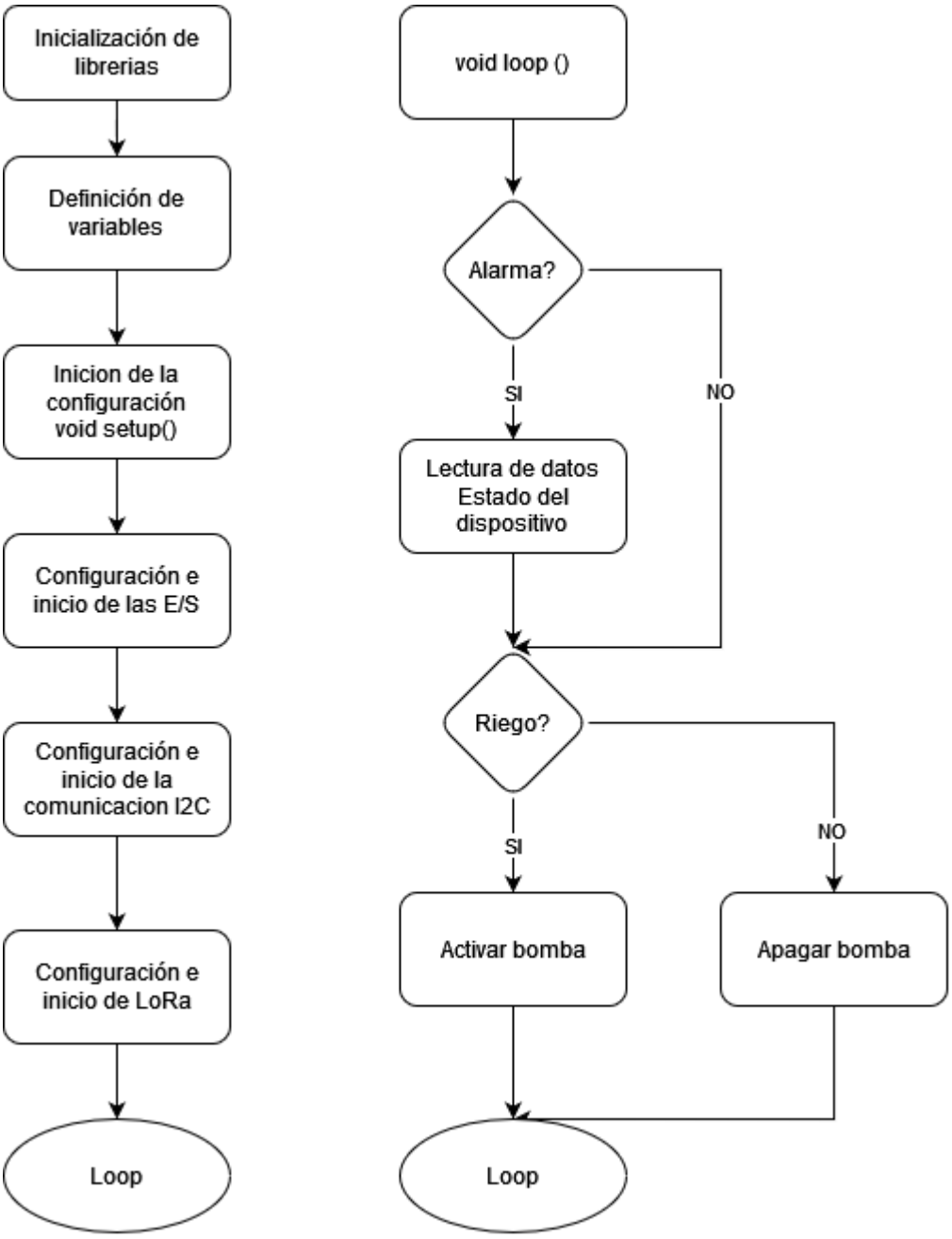


Fig. 14. Diagrama de bloques del código principal.

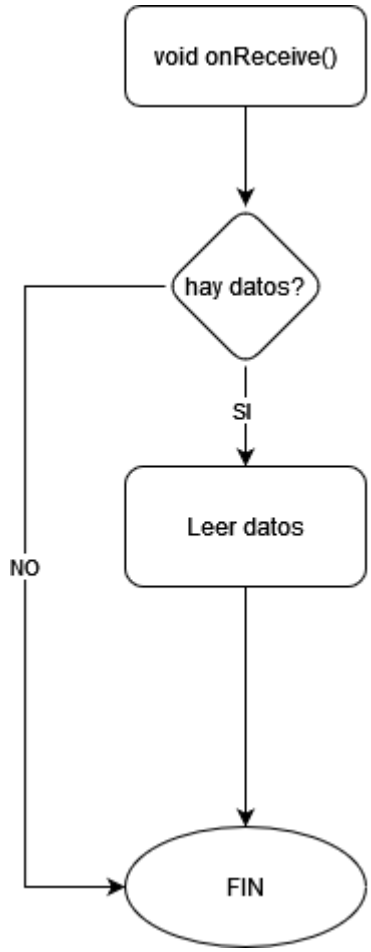


Fig. 15. Diagrama de bloques de la interrupción.

La siguiente figura muestra el circuito planteado para las pruebas.

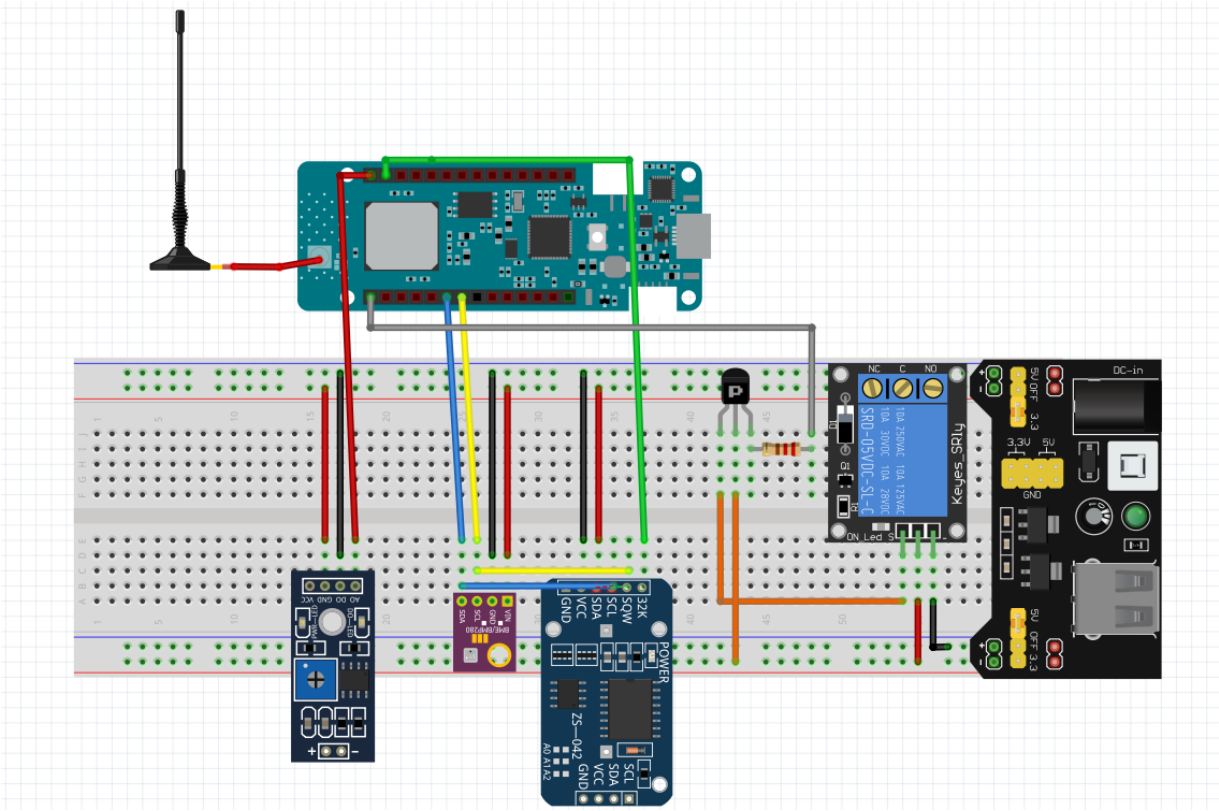


Fig. 16. Circuito planteado.

12.2 Código sobre el Arduino Wemos D1 Mini

El código está formado en dos bloques, una parte la implementación de la web que se muestra y la otra parte la recepción y envío de datos al otro Arduino.

12.2.1 Web Server

La página web se ha escrito en el código HTML con el uso de secuencias en JavaScript para mejorar su utilidad. Se ha fijado la dirección IP local del Arduino para tener un acceso más directo a este, siendo esta la 192.168.1.184 con el puerto 80 del router.

Al conectarse y acceder a la web, lo primero que pide es un usuario y contraseña, se ha apuntado en la parte de arriba al tratarse de un prototipo. La siguiente figura muestra una captura de cómo es el inicio.

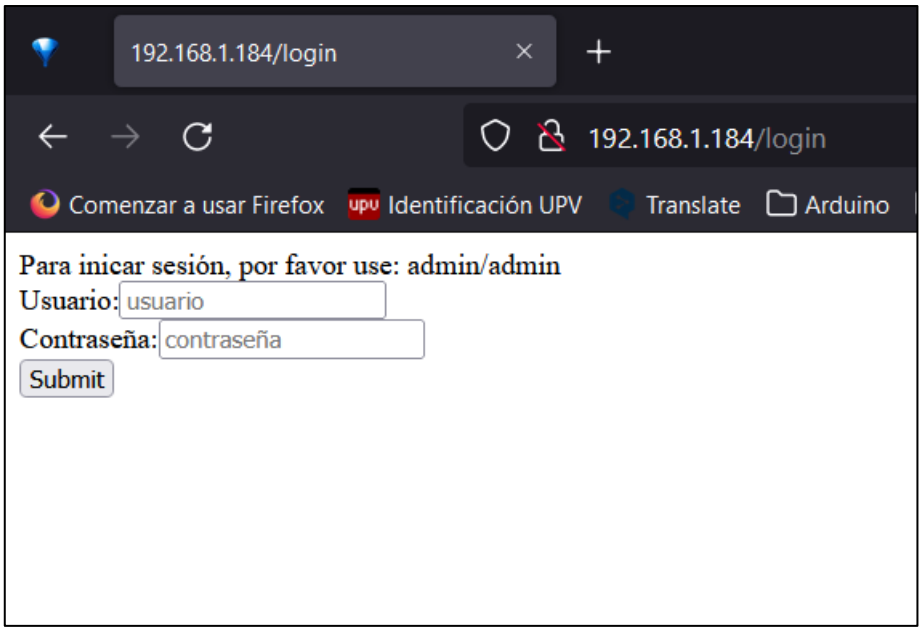


Fig. 17. Web de inicio.

En caso de error de inicio de sesión, se muestra lo siguiente:

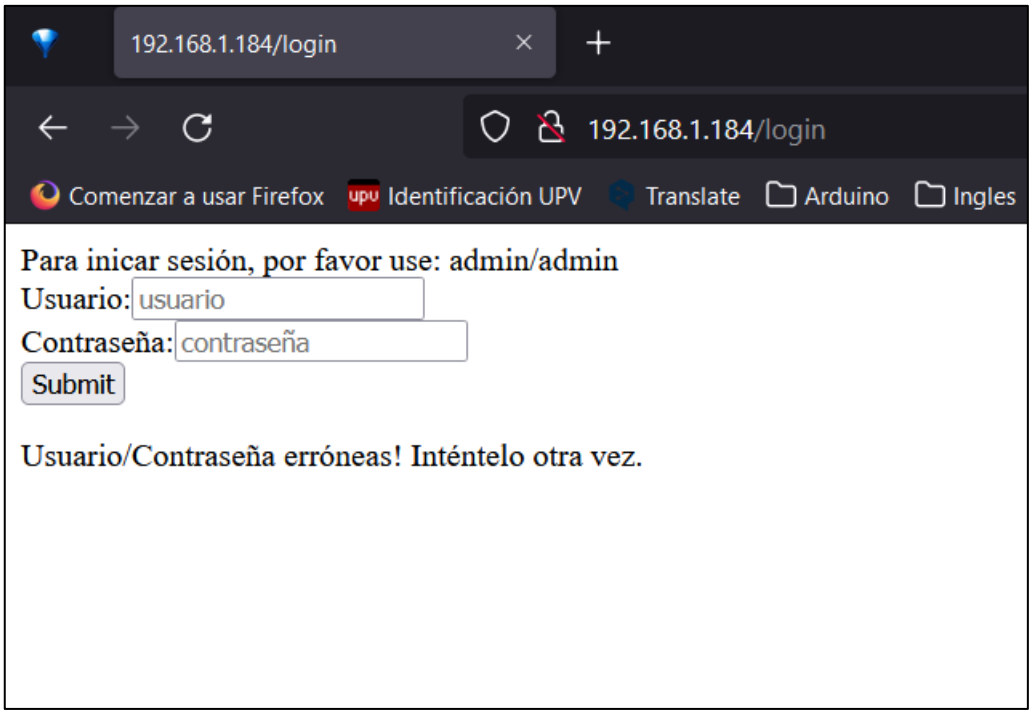


Fig. 18. Web de inicio con error de contraseña.

Una vez se ha iniciado sesión, se redirige a la siguiente página. Durante esta transición el Arduino le pide los datos de los sensores y el estado de este. En la figura de más abajo, se puede observar lo que muestra. Una

región del estado, mostrando una de las tres opciones: online, offline, error. Más abajo se muestran los datos de los sensores y en la parte final se puede ver la hora actual de riego establecida por defecto y un formulario donde se puede rellenar para cambiar esta. Además, existen dos botones con los que configurar, el riego en automático o en manual. Y por último, un pequeño enlace para cerrar sesión.

Fig. 19. Web del servidor con datos.

Aunque no ha quedado como parte del proyecto, se puede adicionar al Arduino una tarjeta SD, donde se registren las horas de riego y los datos de los sensores cada cierto tiempo. Haciendo, así como una base de datos.

Finalmente, el circuito planteado es el siguiente:

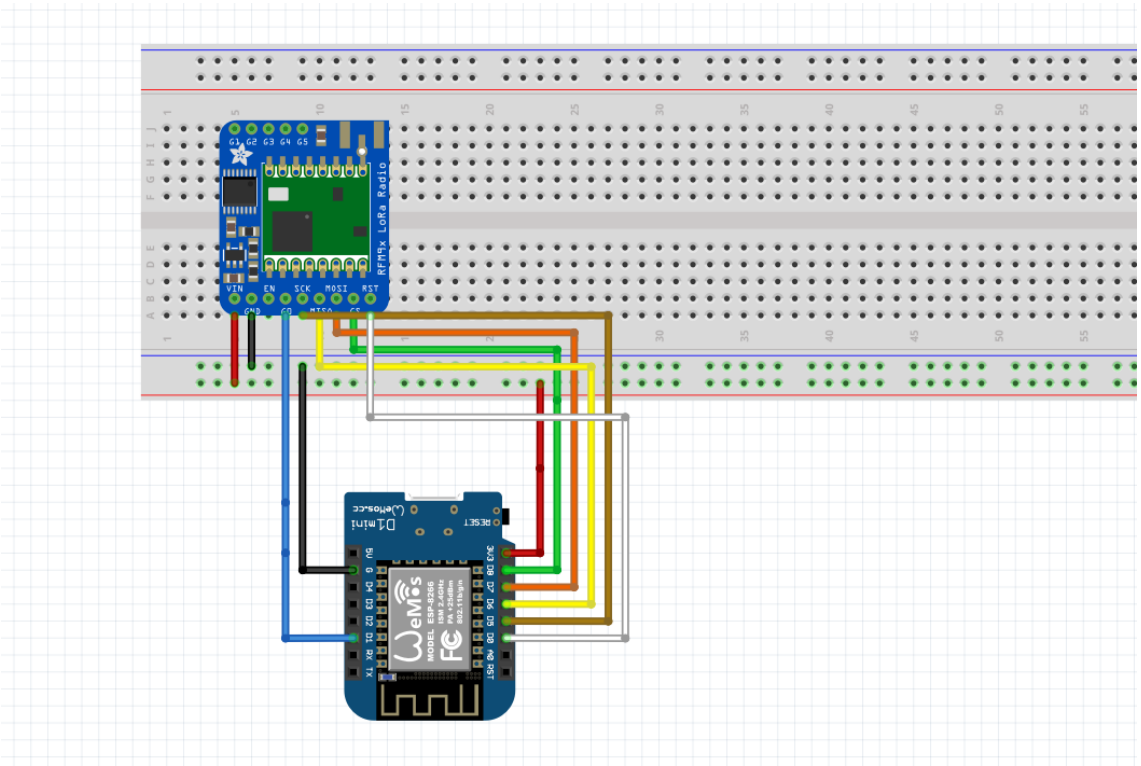


Fig. 20. Conexión del módulo RFM95.

13. Conclusiones

El proyecto cumple con las expectativas planteadas: comunicarse y gestionar el riego. Se establece una comunicación estable entre ambos capaz de transmitir toda la información programada. Todos los sensores obtienen la información sin errores con medidas puntuales en el tiempo. Y el reloj externo a pesar de cortarle la alimentación, sigue contando gracias a la pequeña pila que lo mantiene. Y al volver a conectar el sistema, todo sigue correcto.

A pesar del correcto funcionamiento, de madrugada, especialmente en épocas de frio, existe la posibilidad de que el rocío precipite y moje el sensor de lluvia, dando a entender al programa que está lloviendo cuando no es así. Por tanto, este es un punto mejorable.

Si bien tiene unas funcionalidades limitadas, se le puede añadir más. Una idea posterior, seria incluir indicadores leds para visualizar el estado, e incluso añadir una pantalla y teclado a la placa del campo, para hacer una comunicación más sencilla, en caso concretos de señales débiles.

13.1 Dificultades en la realización del trabajo

El principal problema era la sincronización de ambos con el menor tiempo posible. Al principio se planteó, un enlace a través de una palabra

o clave que se enviaban ambos Arduino para reconocerse. A pesar de esto, había un retardo muy notable en las comunicaciones y a veces incluso ni funcionaba la comunicación. Luego se planteó, hacerlo por otro tipo de sincronización a través de un canal, así el que actúa como receptor ignora las señales o interferencias que le llegan y deja la via libre para recibir sin ningún problema.

14. Referencias

[1] Córdova Aguilar, Hildegardo (2002). Naturaleza y sociedad.

[2] S. Laserna, <<AgroES,>> [Internet]. Disponible en: https://www.agroes.es/agricultura/el-suelo/143-temperatura-del-suelo-agricultura

[3] FAO, [En línea]. Disponible en: http://www.fao.org/3/a-x0490s.pdf

[4] G. Hargreaves y Z. Samani, «Estimation of potential evapotranspiration,» *Journal of Irrigation and Drainage Division,* 1982.

[5] Hoja de características Cortex-M0 32-bit SAMD21 – Arduino [Internet]. 2022. Disponible en: https://content.arduino.cc/assets/mkr-microchip_samd21_family_full_datasheet-ds40001882d.pdf

[6] Hoja de características Murata CMWX1ZZABZ – Arduino [Internet]. 2022. Disponible en: https://content.arduino.cc/assets/mkrwan1310-murata_lora_module-type_abz.pdf

[7] Hoja de características FC-37 Rain Detector – ueloelectronics [Internet] 2022. Disponible en: https://uelectronics.com/wp-content/uploads/2018/04/yl-83-rain-detector-datasheet_low.pdf

[8] Hoja de características BME280 – mouser [Internet]. 2022. Disponible en: https://www.mouser.com/datasheet/2/783/BST-BME280-DS002-1509607.pdf

[9] Hoja de características ESP8266 – espressif [Internet] 2022. Disponible en: https://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[10] Hoja de características RFM95 – hoperf [Internet] 2022. Disponible en: https://www.hoperf.com/data/upload/portal/20190801/RFM95W-V2.0.pdf

[11] Hoja de características RC DS3231 – Maxim Integrated [Internet]. 2022. Disponible en: https://datasheets.maximintegrated.com/en/ds/DS3231.pdf

[12] Normativa europea en relación a comunicaciones LoRa – etsi [Internet]. 2022. Disponible en: https://www.etsi.org/deliver/etsi_en/300200_300299/30022001/03.01.01_60/en_30022001v030101p.pdf

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SISTEMA DE RIEGO CON ARDUINO

DOCUMENTO Nº 2. PLANOS

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

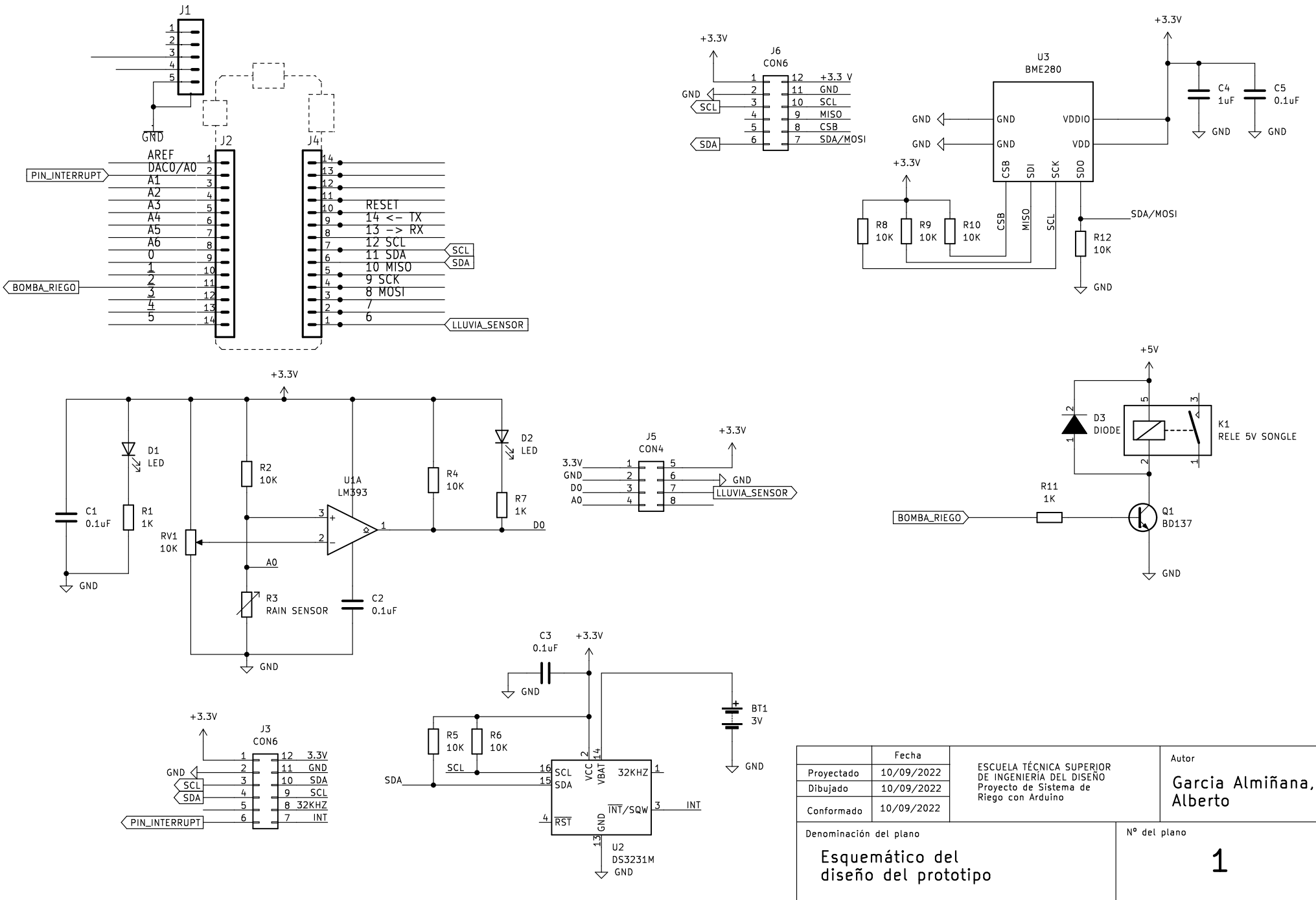
REALIZADO POR

ALBERTO GARCIA ALMIÑANA

TUTORIZADO POR

Salido Gregorio, Miguel Angel

CURSO ACADÉMICO: 2021/2022



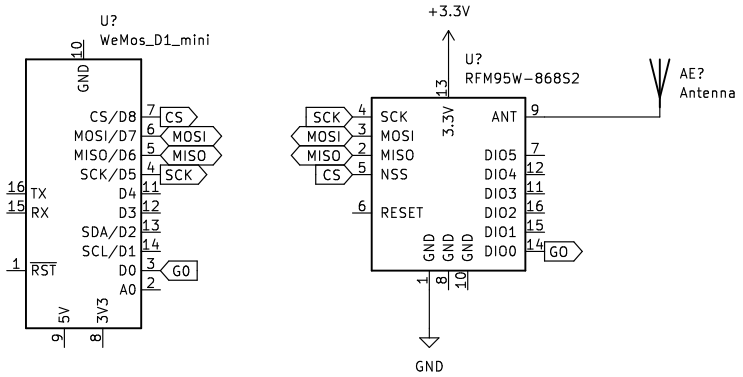
	Fecha		
Proyectado	10/09/2022	ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO Proyecto de Sistema de Riego con Arduino	Autor Garcia Almiñana, Alberto
Dibujado	10/09/2022		
Conformado	10/09/2022		

Denominación del plano

Esquemático del diseño del prototipo

Nº del plano

1



	Fecha	ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO Proyecto de Sistema de Riego con Arduino	Autor
Proyectado	10/09/2022		Garcia Almiñana, Alberto
Dibujado	10/09/2022		
Conformado	10/09/2022		

Denominación del plano

Esquemático del diseño del prototipo

Nº del plano

2

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SISTEMA DE RIEGO CON ARDUINO

DOCUMENTO Nº 3. PLIEGO DE CONDICIONES

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

ALBERTO GARCIA ALMIÑANA

TUTORIZADO POR

Salido Gregorio, Miguel Angel

CURSO ACADÉMICO: 2021/2022

1. Definición y alcance del proyecto

La presente especificación técnica se refiere a la producción de un medidor de ganancia y fase. Se definen las obligaciones y condiciones, estás serán aplicadas al Promotor, al Ingeniero, al Contratista y a todos los relacionados con este.

Se podrán adoptar soluciones alternativas a las exigidas en este documento siempre que se justifique debidamente su necesidad y siempre y cuando no disminuyan la calidad de este.

2. Condiciones y normas de carácter general

Durante la ejecución del proyecto se deberá atender a la siguiente normativa:

- Reglamento electrotécnico de baja tensión RD 842/2002, de 2 de agosto de 2002.
- UNE-EN 61439. Conjuntos de aparamenta de baja tensión. Parte 1: Reglas generales
- Real Decreto 110/2015, de 20 de febrero sobre residuos de aparatos eléctricos y electrónicos.
- Real Decreto 187/2016, 6 de mayo por el que se regulan las exigencias de seguridad del material eléctrico destinado a ser utilizado en determinados límites de tensión. Esto implica la necesidad de un marcado CE en el proyecto.
- Regulación LoRaWAN Regional Parameters.
- Norma ETSI EN 300 220-1 de la ETSI*, European Telecommunications Standars Institut*. Marca la normativa del uso de ondas de radio de 25 MHz hasta 1000 MHz.

3. Condiciones técnicas

Las condiciones especificadas siguientes se refieren a los requerimientos, ya especificados en el documento nº1 de la memoria.

3.1 Condiciones de la ejecución

- Microcontrolador:
  Las características son las siguientes: microcontrolador ESP8266 y SAMD21 tal y como se ha especificado en el documento 1 “Memoria”. El respectivo control de calidad a realizar es: transferir el código de programación especificado en el Anexo I para verificar la comunicación y se comprobará que los resultados son los esperados.

- Sensor de parámetros ambientales:
  El sensor debe ser capaz de medir los 3 parámetros ambientales necesarios: temperatura, presión y humedad. Tal y como se ha especificado en el documento nº1.
- Sensor de lluvia:

  Tal y como se ha especificado en el documento de Memoria.

- Modulo de radiofrecuencia
  Las características son las siguientes: RFM95 o similar.
- Modulo externo de reloj:
  Las características son las siguientes: RTC DS3231. O similar. Capaz de medir con precisión y sin una desviación superior al año de 10 segundos.

3.2 Condiciones de la ejecución

El montaje del prototipo se realiza sobre una protoboard o similar. Pudiendo soldarse los componentes.

4. Condiciones facultativas

Las siguientes condiciones facultativas van dirigidas a la parte contratista y a la

dirección facultativa.

Correspondientes a la parte contratista:

- Conocer la normativa actual.

- Presencia o localización de los responsables o sus representantes durante la ejecución del proyecto.

- Facilitar al Ingeniero Técnico los materiales necesarios, asegurándose que estos son de una calidad suficiente y si no lo son cambiarlos por los oportunos.

- Conocer el proyecto en todas sus partes además de la notificación de la iniciación, finalización, pruebas, controles y recepciones del proyecto o de alguna de sus partes.

- Obligación de seguir en todo momento las indicaciones del proyecto y de la dirección facultativa.

- Notificación previa a la iniciación, finalización, realización de pruebas, controles, recepciones o certificaciones del proyecto o de algunas de sus partes.

- Custodiar los libros de órdenes y seguimiento.

- Reconocer al director del proyecto como la máxima autoridad técnica del proyecto.

- Derecho a recibir los pagos comprometidos en las fechas pactadas.

Correspondientes a la parte de la dirección facultativa:

- Cumplir con la legalidad y las condiciones del contratista.

- Redactar justificadamente las rectificaciones, modificaciones o adicciones que se realicen al proyecto.

- Asumir la responsabilidad de ser la máxima autoridad técnica.

- Supervisar los aspectos del proyecto que puedan afectar a la fiabilidad, calidad y seguridad durante su ejecución.

- Informar periódicamente al cliente de la marcha de los trabajos y de cuantas contingencias surjan y puedan afectar al coste y prestaciones del sistema.

- Encontrarse presente en los momentos del desarrollo del proyecto que se convenga.

- Realizar el seguimiento necesario para intentar obligar a la contrata cumplir los plazos pactados.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SISTEMA DE RIEGO CON ARDUINO

DOCUMENTO Nº 4. PRESUPUESTO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

ALBERTO GARCIA ALMIÑANA

TUTORIZADO POR

Salido Gregorio, Miguel Angel

CURSO ACADÉMICO: 2021/2022

En este apartado se desarrolla el Presupuesto del proyecto, donde se muestra la estimación de los costes de ejecución del proyecto.

1. Precios de los componentes

Componente	Precio/Unidad	Unidades	Subtotal
Arduino MKR WAN 1310	39,60	1	39,60
Arduino Wemos D1 Mini	4,95	1	4,95
BME280	14,95	1	14,95
FC-37 Rain Sensor	4,91	1	4,91
RFM95	7,87	1	7,87
DS3231	6,99	1	6,99
Relé	5,60	1	5,60
Protoboard	4,95	2	9,90
Cable micro-usb	2,99	2	5,98
Adaptador red	9,99	2	19,98
		Subtotal	120,73
Medios auxiliares sobre costes directos	5%		6,04
		Total	**126,77 €**

Tabla 5. Precios de los componentes.

2. Equipo de alquiler

Concepto	Precio/Unidad	Tiempo	Subtotal
Alquiler PC	10	15	150,00
		Subtotal	150,00
Medios auxiliares sobre costes directos	5%		7,50
		Total	**157,50 €**

Tabla 6. Equipo de alquiler.

3. Resumen de presupuesto

Concepto	Importe
Componentes	126,77
Equipo	157,50
Presupuesto de ejecución de material	284,27
13% gastos generales	36,95
6% beneficio empresa	17,06
Subtotal	54,01
21% IVA	11,34
Presupuesto de ejecución por contrata	**295,61 €**

Tabla 7. Resumen de presupuesto.

4. Honorarios de la empresa

Concepto		Importe
Proyecto	8% sobre PEM	22,74
IVA	21% sobre Proyecto	4,78
Total Honorarios		27,52
Dirección de Proyecto	10% sobre PEM	28,43
IVA	21% sobre Dirección de Proyecto	5,97
Total Honorarios Dirección de Proyecto		34,40
Total Honorarios de la empresa		**61,91 €**

Tabla 8. Honorarios de la empresa.

5. Mano de obra

Concepto	**Precio/Unidad**	**Tiempo**	**Importe**
Diseño del sistema por un Ingeniero	18,00	160	2880,00
		Subtotal	2880,00
Medios auxiliares sobre costes directos	5%		144,00
			3.024,00 €

Tabla 9. Mano de obra.

6. Presupuesto general

Concepto	Importe
Presupuesto de ejecución contratada	295,61 €
Honorarios de empresa	61,91 €
Mano de obra	3.024,00 €
Total presupuesto general	**3.381,52 €**

Tabla 10. Presupuesto general

El presupuesto general asciende a la cantidad de TRES MIL TRESCIENTOS OCHENTA Y UNO CON CINCUENTA Y DOS EUROS (3.381,52 €).

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SISTEMA DE RIEGO CON ARDUINO

ANEXO I. CÓDIGO DEL MICROCONTROLADOR

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

ALBERTO GARCIA ALMIÑANA

TUTORIZADO POR

Salido Gregorio, Miguel Angel

CURSO ACADÉMICO: 2021/2022

CODIGO DEL ARDUINO MKR WAN 1310

```
/* TFG RIEGO
   Alberto Garcia Almiñana 2021/2022
   Ingeniería Electrónica Industrial y Automática

*/

//----DECLARACION DE LIBRERIAS----
#include <Wire.h>
#include <SPI.h>
#include <LoRa.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <RTClib.h>
#include "ArduinoLowPower.h"

//----INICIALIZACION DE CONEXIONES I2C----
RTC_DS3231 RTC_RELOJ;
Adafruit_BME280 BME_SENSOR;

//DEFINICIONES
#define BOMBA_RIEGO 2
#define PIN_INTERRUPT 4
#define LLUVIA_SENSOR 6

#define PRESION_NIVEL_MAR 1013.25
#define TX_POWER 17  //Potencia de la transmisión, 2...20
/*DIRECCIONES FÍSICAS*/
#define BME280_ADDRESS 0x76  //Dirección del BME280
#define DS3231_ADDRESS 0x68  //Dirección del DS3231
#define DEVUELVE_DATOS 0x00
#define ESTABLECE_DATOS 0x01
#define ESTADO 0xB0
```

```
#define ENVIO_DATOS 0xB1
#define ENVIO_ERRORES 0xB2
/*VALORES MAXIMOS Y MINIMOS*/
#define PRESION_MIN 990
#define PRESION_MAX 1050
#define HUMEDAD_MIN 10
#define HUMEDAD_MAX 100
#define TEMPERATURA_MIN -10
#define TEMPERATURA_MAX 60

//----INICIALIZACION DE VARIABLES PROPIAS----
struct Datos {
  float humedad;
  float temperatura;
  float presion;
  bool lluvia;
};

struct Tiempo {
  int anyo;
  uint8_t mes;
  uint8_t dia;
  uint8_t hora;
  uint8_t minutos;
  bool riegoAuto;
};

//----INICIALIZACION DE VARIABLES GLOBALES----
bool estadoBME = false;    //Estado de la conexión I2c con el BME280
bool estadoRTC = false;    //Estado de la conexión I2c con el DS3231
bool estadoBomba = false;  //Estado de la hora de riego
bool estadoLora = false;    //Estado de la conexión de la comunicacion LoRa (Long Range)
byte estadoDispositivo = true;
```

```
volatile bool estadoAlarma = true;  //Estado de la interrupción
float superficie;
float Kc, caudal;
Datos datosLectura;
Datos datosValoresMax;
Datos datosValoresMin;
Tiempo horaRiego;

//----INICIALIZACIÓN DE FUNCIONES----
float tiempoRiego();
bool comprobarLecturasDatos(float dato, int valor_min, int valor_max);
void lecturaDatos();
void loraDatos(byte tipoDato);
bool horaDeRiego(DateTime fecha);
float actualizarValorMax();
float actualizarValorMin();
void reiniciarValores();
uint16_t diaDelAnyo(int anyo, int mes, int diaDelMes);
byte estadoActual();

void setup() {
  //DECLARACION DE ENTRADAS Y SALIDAS
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(PIN_INTERRUPT, INPUT_PULLUP);
  pinMode(LLUVIA_SENSOR, INPUT);
  pinMode(BOMBA_RIEGO, OUTPUT);

  delayMicroseconds(100);

  //______SALIDAS POR DEFECTO EN BAJO_______
  digitalWrite(BOMBA_RIEGO, LOW);
  digitalWrite(LED_BUILTIN, LOW);  //LED de la placa: apagado

  //_____CONFIGURACION DEL PIN DE INTERRUPCION______
```

```
 attachInterrupt(digitalPinToInterrupt(PIN_INTERRUPT), onAlarm, CHANGE);

 delayMicroseconds(100);

 Wire.begin();
 /*______COMUNICACIONES I2C-BME280_____*/
 if (BME_SENSOR.begin(BME280_ADDRESS)) {
   //Variable sensor encontrado
   estadoBME = true;
   //....CONFIGURACION.....
   BME_SENSOR.setSampling(Adafruit_BME280::MODE_FORCED,
                       Adafruit_BME280::SAMPLING_X1,  // temperature
                       Adafruit_BME280::SAMPLING_X1,  // pressure
                       Adafruit_BME280::SAMPLING_X1,  // humidity
                       Adafruit_BME280::FILTER_OFF);
 }

 delayMicroseconds(100);

 /*______COMUNICACIONES I2C-DS3231 RTC_____*/
 if (RTC_RELOJ.begin()) {
   estadoRTC = true;
   RTC_RELOJ.disable32K();
   RTC_RELOJ.clearAlarm(1);
   RTC_RELOJ.clearAlarm(2);
   RTC_RELOJ.writeSqwPinMode(DS3231_OFF);

   RTC_RELOJ.disableAlarm(2);

   //Alarma cada 60 segundos
         if    (!RTC_RELOJ.setAlarm1(RTC_RELOJ.now()    +    TimeSpan(60),
DS3231_A1_Second)) {
     estadoReloj = false;
   } else {
```

```
      estadoReloj = true;
    }
  }

  /*______CONFIGURACIÓN DE LoRa_______*/
  if (LoRa.begin(868E6)) {
    estadoLora = true;

    LoRa.setSpreadingFactor(8);
    //Potencia del transmisor
    LoRa.setTxPower(TX_POWER);
    LoRa.onReceive(onReceive);
    LoRa.receive();
  }

  reiniciarValores();

  //Estado Dispositivo
  estadoDispositivo = estadoActual();
}

void loop() {
  if (RTC_RELOJ.alarmFired(1))
    RTC_RELOJ.clearAlarm(1);
  if (estadoAlarma) {
    lecturaDatos();
    estadoDispositivo = estadoActual();
    estadoAlarma = false;
  }

  //Obtiene la hora actual y llama a la función
  DateTime fechaActual = RTC_RELOJ.now();
  if (estadoBomba == false && horaDeRiego(fechaActual) && datosLectura.lluvia
== false) {
```

```
    digitalWrite(BOMBA_RIEGO, HIGH);
    estadoBomba = true;
    } else if (estadoBomba == true && horaDeRiego(fechaActual) || datosLectura.lluvia == true) {
    digitalWrite(BOMBA_RIEGO, LOW);
    estadoBomba = false;
  }
}

//FUNCION DEVUELVE EL ESTADO DEL DISPOSITIVO
byte estadoActual() {
  if (estadoBME && estadoRTC)
    return 0x01;
  else if (estadoBME == false || estadoRTC == false)
    return 0x02;
}

//RECEPCIÓN DE PAQUETE DE DATOS
void onReceive(int packetSize) {
  if (packetSize == 0) return;

  DateTime fechaActual = RTC_RELOJ.now();
  uint8_t direccionReceptor, direccionTransmisor, tipoDatos, datosFinales;

  tipoDatos = LoRa.read();
  if (tipoDatos == DEVUELVE_DATOS) {
    loraEnvio(ENVIO_DATOS);
    return;
  } else if (tipoDatos == ESTABLECE_DATOS) {
    horaRiego.hora = LoRa.read();
    horaRiego.minutos = LoRa.read();
    horaRiego.riegoAuto = LoRa.read();
    return;
  }
```

```
}

//ENVIO DE DATOS
void loraEnvio(byte tipoDato) {
  estadoDispositivo = estadoActual();
  lecturaDatos();
  LoRa.beginPacket();

  if (tipoDato == ENVIO_DATOS) {
        String salida = String(datosLectura.temperatura) + "t" +
String(datosLectura.humedad) + "h" + String(datosLectura.presion) + "p";
    LoRa.write(ENVIO_DATOS);
    LoRa.write(estadoDispositivo);
    LoRa.write(datosLectura.lluvia);
    LoRa.write(salida.length());
    LoRa.print(salida);
  } else {
    LoRa.write(ENVIO_ERRORES);
    LoRa.write(estadoDispositivo);
  }
  LoRa.endPacket();
  LoRa.receive();
}

//ACTUALIZAR VALORES MAXIMOS
float actualizarValorMax(float valor, float nuevoValor) {
  if (nuevoValor > valor)
    return nuevoValor;
  return valor;
}

//ACTUALIZAR VALORES MINIMOS
float actualizarValorMin(float valor, float nuevoValor) {
  if (nuevoValor < valor)
```

```
    return nuevoValor;
  return valor;
}

//REINICIAR VALORES MINIMOS Y MAXIMOS
void reiniciarValores() {
  datosValoresMin.temperatura = TEMPERATURA_MAX;
  datosValoresMin.presion = PRESION_MAX;
  datosValoresMin.humedad = HUMEDAD_MAX;

  datosValoresMax.temperatura = TEMPERATURA_MIN;
  datosValoresMax.presion = PRESION_MIN;
  datosValoresMax.humedad = HUMEDAD_MIN;
}

uint16_t diaDelAnyo(int anyo, int mes, int diaDelMes) {
  const uint8_t diasMeses[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
  bool bisiesto;
  uint16_t dias = 0;

  if ((anyo % 400 == 0) || ((anyo % 4 == 0) && (anyo % 100 != 0)))
    bisiesto = true;
  else
    bisiesto = false;

  for (int contadorMes = 0; contadorMes < (mes - 1); contadorMes++) {
    dias += diasMeses[contadorMes];
    if (contadorMes == 1 && bisiesto) {
      dias++;
    }
  }
  return dias + diaDelMes;
}
```

```
//FUNCION TIEMPO DE RIEGO
float tiempoRiego() {
  DateTime now = RTC_RELOJ.now();
  float Tmed = datosValoresMax.temperatura + datosValoresMin.temperatura / 2;
  int z = 15;          //altura sobre el nivel del mar
  int u2 = 10;          //viento
  float gamma = 0.067;  //cte psicometrica
  int G = 0;
  float alpha = 0.23;
  float kRs = 0.19;   //coeficiente de ajuste
  float Gsc = 0.082;  //cte solar
  float J = diaDelAnyo(now.year(), now.month(), now.day());
  //float J = ((7 * 275 / 9) - 30 + 8) - 2;
  float phi = 0.49452;                                  //latitud
  float sigma = 0.000000004903;                         //constante de Stefan-Boltzmann
  float delta = 0.409 * (sin((2 * 3.14 * J / 365) - (1.39)));  //declinacion solar
  float dr = 1 + (0.033 * cos(2 * 3.14 * J / 365));          //distancia relativa inversa al sol
  float ws = acos(-tan(phi) * tan(delta));                   //angulo de radiacion a la puesta del sol
  float Ra = (24 * 60 / 3.14) * Gsc * dr * ((ws * sin(phi) * sin(delta)) + (cos(phi) * cos(delta) * sin(ws)));
    float Rs = kRs * Ra * sqrt((datosValoresMax.temperatura - datosValoresMin.temperatura));  //radiacion hargreaves
  float Rso = Ra * (0.75 + (0.00002) * z);                                  //rdiacion en un dia despejado
  float eoTmax = 0.6108 * (pow(2.718, ((17.27 * datosValoresMax.temperatura) / (datosValoresMax.temperatura + 237.3))));
  float eoTmin = 0.6108 * (pow(2.718, ((17.27 * datosValoresMin.temperatura) / (datosValoresMin.temperatura + 237.3))));
  float ea = (((eoTmin * datosValoresMax.humedad / 100) + (eoTmax * datosValoresMin.humedad / 100)) / 2);                                  //presion de vapor real
        float es = (eoTmax + eoTmin) / 2;
                    //presion de vapor de saturacion
```

```
        float Rns = (1 - alpha) * Rs;
                        //radiacion neta de onda corta
  float Rnl = sigma * ((pow((datosValoresMax.temperatura + 273.15), 4) + pow((datosValoresMin.temperatura + 273.15), 4)) / 2) * (0.34 - (0.14 * sqrt(ea))) * ((1.35 * Rs / Rso) - 0.35);  //radiacion neta de onda larga
  float Rn = Rns - Rnl;
  float D = (4098 * (0.6108 * (pow(2.718, ((17.27 * Tmed) / (Tmed + 237.3)))))) / (pow((Tmed + 237.3), 2));  //pendiente de la curva de presion de vapor
  float ETo = ((0.408 * D * (Rn - G)) + (gamma * 900 * u2 * (es - ea) / (Tmed + 273))) / (D + (gamma * (1 + 0.34 * u2)));
  float ETc = Kc * ETo;
  float NHn = ETc * superficie;
  float tiempo = NHn / caudal;
  return tiempo;
}

//COMPROBAR VALORES LECTURAS
bool comprobarLecturasDatos(float dato, int valor_min, int valor_max) {
  if (dato < valor_min || dato > valor_max)
    return false;
  return true;
}

//LECTURA DE DATOS DE LOS SENSORES
void lecturaDatos() {
  bool lecturaCorrecta_TEMP = false, lecturaCorrecta_HUM = false,
       lecturaCorrecta_PRES = false;
  float valorTemp, valorHum, valorPres;
  estadoBME = BME_SENSOR.takeForcedMeasurement();

  delayMicroseconds(10);

  valorTemp = BME_SENSOR.readTemperature();
  valorHum = BME_SENSOR.readHumidity();
  valorPres = BME_SENSOR.readPressure() / 100.0;
```

```
    if (comprobarLecturasDatos(valorTemp, TEMPERATURA_MIN, TEMPERATURA_MAX)) {
      datosLectura.temperatura = valorTemp;
                                        datosValoresMax.temperatura = actualizarValorMax(datosValoresMax.temperatura, valorTemp);
                                        datosValoresMin.temperatura = actualizarValorMin(datosValoresMin.temperatura, valorTemp);
     } else
      estadoBME = false;
     if (comprobarLecturasDatos(valorPres, PRESION_MIN, PRESION_MAX)) {
      datosLectura.presion = valorPres;
       datosValoresMax.presion = actualizarValorMax(datosValoresMax.presion, valorPres);
        datosValoresMin.presion = actualizarValorMin(datosValoresMin.presion, valorPres);
     } else
      estadoBME = false;
     if (comprobarLecturasDatos(valorHum, HUMEDAD_MIN, HUMEDAD_MAX)) {
      datosLectura.humedad = valorHum;
                                        datosValoresMax.humedad = actualizarValorMax(datosValoresMax.humedad, valorHum);
       datosValoresMin.humedad = actualizarValorMin(datosValoresMin.humedad, valorHum);
     } else
      estadoBME = false;


     datosLectura.lluvia = digitalRead(LLUVIA_SENSOR);


     estadoAlarma = false;


     if (estadoBME != true)
      loraEnvio(ENVIO_ERRORES);
}


//Comprueba la hora para el riego
```

```
bool horaDeRiego(DateTime fecha) {
  int diaSemana;
  float hora;
  bool horaCondicion, diaCondicion;

  diaSemana = fecha.dayOfTheWeek();
  hora = fecha.hour() + fecha.minute() / 60.0;

  horaCondicion = (hora > horaRiego.hora) && (hora < (hora + tiempoRiego() *
24));

  if (horaCondicion)
    return true;

  return false;
}

void onAlarm() {
  estadoAlarma = true;
}
```

CODIGO DEL ARDUINO WEMOS D1 MINI

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <SPI.h>
#include <LoRa.h>

//Pines de conexión SPI con el módulo LoRa Rfm95
#define ss 15
#define rst 16
#define dio0 2
//Definición de variables
#define DEVUELVE_DATOS 0x00
#define ESTABLECE_DATOS 0x01
#define ESTADO 0xB0
#define ENVIO_DATOS 0xB1
#define ENVIO_ERRORES 0xB2
#define FIN_DATOS 0xFF
#define TX_POWER 17  //Potencia de la transmisión, 2...20
#define LED_BUILTIN 2

struct Datos {
  String humedad;
  String temperatura;
  String presion;
  bool lluvia;
};

struct Tiempo {
  int anyo;
  uint8_t mes;
  uint8_t dia;
  uint8_t hora;
  uint8_t minutos;
};
```

```
//Variables globales
const char* ssid = "MIWIFI_2G_QF6h";
const char* password = "JfhkgtQX";

byte estadoComunicacion = false;
bool riegoAuto = true;
uint8_t hora = 22, minutos = 30;

Datos datosRec;

//Puesto del servidor
ESP8266WebServer server(80);
//Ip estática
IPAddress local_IP(192, 168, 1, 184);
//Puesta de acceso IP al router
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);
IPAddress primaryDNS(8, 8, 8, 8);
IPAddress secondaryDNS(8, 8, 4, 4);

String deIntString(uint8_t num) {
  String numVuelta;
  if (num == 0)
    numVuelta = "00";
  else if (num > 0 && num < 10)
    numVuelta = "0" + String(num);
  else
    numVuelta = String(num);
  return numVuelta;
}

bool is_authenticated() {
  Serial.println("Enter is_authenticated");
```

```
  if (server.hasHeader("Cookie")) {
    Serial.print("Found cookie: ");
    String cookie = server.header("Cookie");
    Serial.println(cookie);
    if (cookie.indexOf("ESPSESSIONID=1") != -1) {
      Serial.println("Authentication Successful");
      return true;
    }
  }
  Serial.println("Authentication Failed");
  return false;
}

//Código en HTML de la web principal
String WebHtml() {
  String header;
  header = "<html><head><title>RIEGO</title>";
                    header += "<link href=\"http://fonts.googleapis.com/css?family=Roboto:300|Playfair+Display:400\" rel=\"stylesheet\" type=\"text/css\"/>";
          header += "<link rel=\"stylesheet\" href=\"http://static.tumblr.com/pjglohe/2qinf00ga/estilos.min.css\"></head>";
  header += "<body><div class=\"page-wrap\">";
  header += "<script type=\"text/javascript\">";
  header += "function myFunction(elmnt,clr) {elmnt.style.color = clr;</script>";
  //Cambio de color
  header += "<script type=\"text/javascript\">";
  header += "function myFunction2(elmnt,clr) {";
  header += "elmnt.style.color = clr;</script>";
  //Titulo 1
  header += "<div align=\"center\" style=\"background: #eeeeee; border: 1px solid black;\">";
  header += "<meta http-equiv='Content-Type'";
  header += "content='text/html; charset=utf-8'/>";
```

```
header += "<h1>CONFIGURACIÓN RIEGO  <input type='button'
      value='Recargar página' onClick='location.reload();'";

 header += "align='center'style='width: 200px; height: 30px; background:
      #6699FF; color: #ffffff; cursor: pointer; border: opx;'/></h1></div>";

header += "<div align=\"center\" style=\"background: #eeeeee; border: 1px solid
      black;\">";

if (estadoComunicacion == 0x01)

     header += "<h2><font size = 5>Estado del dispositivo: <font
      color=green>ONLINE</font></h2></div>";

else if (estadoComunicacion == 0x00)

     header += "<h2><font size = 5>Estado del dispositivo: <font
      color=blue>OFFLINE</font></h2></div>";

else

     header += "<h2><font size = 5>Estado del dispositivo: <font
      color=red>ERROR</font></h2></div>";

 header += "<div align=\"center\" class=\"datos-tiempo\"><h3><font size =
      5><u>Datos metereológicos en tiempo real</u></font> </h3>";

   header += "<ul><li><font size = 4>Temperatura: " +
      datosRec.temperatura + "&ordm;C</font></li>";

 header += "<li><font size = 4>Humedad: " + datosRec.humedad +
      "%</font></li>";

 header += "<li><font size = 4>Presión: " + datosRec.presion +
      " hPa</font></li>";

if (datosRec.lluvia)

  header += "<li><font size = 4>Está lloviendo</li></font>";

else

  header += "<li><font size = 4>No está lloviendo</li></font>";

header += "</ul></p></div>";

 header += "<div align=\"center\" class=\"configuracion\"><h4><font size =
      5><u>Configuración de las horas de riego</u></font></h4>";

 header += "<font size = 4>Hora de riego actualmente establecida a
      las: <b>" + deIntString(hora) + ":" + deIntString(minutos) +
      " h</b></font>";

header += "<p><form action='/hora'><font size = 4>Establecer nueva hora de
      riego:</font><input type='number' name='HORA' placeholder='hora'
      min='0' max='23'> ";

 header += "<input type='number' name='MINUTOS' placeholder='minutos'
      min='0' max='59'>  ";

header += "<input type='submit' name='ENVIO' value='Enviar'></form>";
```

```
  header += "</p>";
  header += "<p><font size = 4>Riego en modo: ";
  if (riegoAuto)
    header += "<font color = green>AUTOMÁTICO";
  else
    header += "<font color = red>MANUAL";
  header += "</font>    ";
   header += "<form method='POST' action='/AutoOn'><button type='submit' name='AUTOON'>Automático</button></form>";
   header += "<form method='POST' action='/AutoOff'><button type='submit' name='AUTOOFF'>Manual</button>";
  header += "</form>";
   header += "<div align='rigth'><a href=\"/login?DISCONNECT=YES\"><font size=3>Desconexión</font></a></div></body></html>";

  return header;
}

void handleCicloAuto() {
  riegoAuto = true;
  envioDatos(ESTABLECE_DATOS);
  Serial.println(riegoAuto);
  server.send(200, "text/html", WebHtml());
}

void handleCicloMan() {
  riegoAuto = false;
  envioDatos(ESTABLECE_DATOS);
  Serial.println(riegoAuto);
  server.send(200, "text/html", WebHtml());
}

void handleRequest() {
  String horaEntrada, minEntrada;
  horaEntrada = server.arg("HORA");
```

```
  minEntrada = server.arg("MINUTOS");
  hora = horaEntrada.toInt();
  minutos = minEntrada.toInt();
  server.send(200, "text/html", WebHtml());
  envioDatos(ESTABLECE_DATOS);
}

//web para el inicio de sesión, también cuando cierras sesión
void handleLogin() {

  String msg;
  if (server.hasHeader("Cookie")) {
    Serial.print("Found cookie: ");
    String cookie = server.header("Cookie");
    Serial.println(cookie);
  }
  if (server.hasArg("DISCONNECT")) {
    Serial.println("Desconectado");
    server.sendHeader("Location", "/login");
    server.sendHeader("Cache-Control", "no-cache");
    server.sendHeader("Set-Cookie", "ESPSESSIONID=0");
    server.send(301);
    return;
  }
  if (server.hasArg("USERNAME") && server.hasArg("PASSWORD")) {
     if (server.arg("USERNAME") == "admin" && server.arg("PASSWORD") ==
         "admin") {
      server.sendHeader("Location", "/");
      server.sendHeader("Cache-Control", "no-cache");
      server.sendHeader("Set-Cookie", "ESPSESSIONID=1");
      server.send(301);
      Serial.println("Inicio correcto");
      return;
    }
```

```
    msg = "Usuario/Contraseña erróneas! Inténtelo otra vez.";
    Serial.println("Inicio de sesión incorrecto");
  }

      String content = "<html><body><meta http-equiv='Content-Type'content='text/html; charset=utf-8'/><form action='/login' method='POST'>Para inicar sesión, por favor use: admin/admin<br>";
      content += "Usuario:<input type='text' name='USERNAME' placeholder='usuario'><br>";
    content += "Contraseña:<input type='password' name='PASSWORD' placeholder='contraseña'><br>";
  content += "<input type='submit' name='SUBMIT' value='Submit'></form>" + msg + "<br>";
  content += "</body></html>";
  server.send(200, "text/html", content);
}

void handleRoot() {
  envioDatos(DEVUELVE_DATOS);
  Serial.println("Enter handleRoot");
  String header;
  if (!is_authenticated()) {
    server.sendHeader("Location", "/login");
    server.sendHeader("Cache-Control", "no-cache");
    server.send(301);
    return;
  }

  server.send(200, "text/html", WebHtml());
}

void handleNotFound() {
  String message = "Archivo no encontrado\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
```

```
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) {
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
}

void envioDatos(uint8_t datos) {
  Serial.println("Enviado");

  LoRa.beginPacket();

  if (datos == DEVUELVE_DATOS) {
    LoRa.write(DEVUELVE_DATOS);
  } else if (datos == ESTABLECE_DATOS) {
    LoRa.write(ESTABLECE_DATOS);
    LoRa.write(hora);
    LoRa.write(minutos);
    LoRa.write(riegoAuto);
  }

  LoRa.endPacket();
  LoRa.receive();
}

void onReceive(int packetSize) {
  if (packetSize == 0) return;

  String cadenaRec = "";
  int direccionRec, direccionTrans, tipoMensaje, longitudCadena;
  bool estadoLluvia;
```

```
  tipoMensaje = LoRa.read();

  if (tipoMensaje == ENVIO_DATOS) {
    estadoComunicacion = LoRa.read();
    estadoLluvia = LoRa.read();
    longitudCadena = LoRa.read();
    while (LoRa.available())
      cadenaRec += (char)LoRa.read();
  } else if (tipoMensaje == ENVIO_ERRORES) {
    estadoComunicacion = 0x02;
  }

  if (tipoMensaje == ENVIO_DATOS) {
    if (longitudCadena != cadenaRec.length())
      return;
    else {
      int posT, posH, posP;
      posT = cadenaRec.indexOf("t");
      posH = cadenaRec.indexOf("h");
      posP = cadenaRec.indexOf("p");
      datosRec.temperatura = cadenaRec.substring(0, posT - 1);
      datosRec.humedad = cadenaRec.substring(posT + 1, posH - 1);
      datosRec.presion = cadenaRec.substring(posH + 1, posP - 1);
      datosRec.lluvia = estadoLluvia;
    }
  }
}

void setup() {
  Serial.begin(115200);

  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);
```

```
WiFi.mode(WIFI_STA);
if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
  Serial.println("Fallo configuracion");
}

//Conectar a la SSID con la contraseña
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
//Imprime por pantalla la direccion
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());

server.on("/", handleRoot);
server.on("/login", handleLogin);
server.on("/inline", []() {
  server.send(200, "text/plain", "this works without need of authentication");
});
server.on("/hora", handleRequest);  //funcion para los datos introducido
server.on("/AutoOn", handleCicloAuto);
server.on("/AutoOff", handleCicloMan);

server.onNotFound(handleNotFound);

server.collectHeaders("User-Agent", "Cookie");
server.begin();
Serial.println("HTTP server started");

LoRa.setPins(ss, rst, dio0);
```

```
  if (!LoRa.begin(868E6)) {
    Serial.print("Error en Modulo LoRa");
    delay(500);
  } else {
    LoRa.setSpreadingFactor(8);
    LoRa.setTxPower(TX_POWER);
    LoRa.onReceive(onReceive);
    LoRa.receive();
  }
}

void loop() {
  server.handleClient();
}
```