Universidad Zaragoza
1542

Trabajo Fin de Grado

Sistema de control automático y supervisión de riego mediante Arduino.
Automatic control system and supervisión of irrigation through Arduino.

Autor

Diego Ibañez Garfella

Director

Jesús Lázaro Plaza

Codirector

Eduardo Gil Herrando

Escuela Universitaria Politécnica de Teruel.
Grado en Ingeniería Electrónica y Automática.

2020 / 2021

Resumen

El presente trabajo consiste en la propuesta de un sistema de riego automatizado mediante una electroválvula, controlado de manera remota, a través de la red Wifi con una aplicación Android para móvil. Se plantea diseñar un sistema de riego automático, que combine soluciones hardware de bajo coste y software libre.

El microcontrolador, como cerebro de todas las operaciones para asegurar el control, suministro y dosificación del agua para mantener hidratada la planta.

Se añaden a este, los elementos hardware de bajo coste, esenciales para la medición de los valores de temperatura ambiental, humedad de la tierra, humedad del aire y la radiación UV, importantísimos a la hora del desarrollo de las plantas que forman parte del ecosistema en el desarrollo del crecimiento de la planta y la productividad de las cosechas.

Por lo expuesto, esta solución, incluye una aplicación móvil Android que utilizando la tecnología de conectividad al Wifi, establece el canal de comunicación con el microcontrolador, permitiendo la emisión y recepción de las señales de los diferentes sensores logrando gestionar de manera autónoma el riego desde cualquier lugar, a cualquier hora y minimizando el trabajo de las personas.

Abstract

The present work consists of the approach of an automated irrigation system using an electrovalve, controlled remotely, through the Wifi network using an Android mobile application. It is proposed to design an automatic irrigation system that combines low-cost hardware solutions and free software.

The microcontroller, as the brain of all operations to ensure the control, supply and dosage of water to keep the plant hydrated.

Low-cost hardware elements are added to this, essential for measuring the values of environmental temperature, soil humidity, air humidity and UV radiation, which are extremely important when developing the plants that are part of the ecosystem. in developing plant evolution and farming productivity.

Therefore, this solution includes an Android mobile application that using Wi-Fi connectivity technology, establishes the communication channel with the microcontroller, allowing the emission and reception of the signals from the different sensors, managing to manage irrigation autonomously. from anywhere, at any time and minimizing the work of people.

Índice de Contenido

1 Introducción ..... - 1 -
1.1 Suelos y sus componentes primarios ..... - 1 -
1.1.1 El pH ..... - 1 -
1.1.2 La luz ..... - 1 -
1.1.3 Humedad relativa (HR) ..... - 1 -
1.2 Riego ..... - 2 -
1.2.1 Sistemas de riego tradicionales ..... - 2 -
1.2.2 Sistemas de riego modernos ..... - 2 -
1.3 Generalidades del sistema de control ..... - 3 -
1.3.1 Sistema de Control de Lazo Abierto ..... - 3 -
1.3.2 Sistema de Control de Lazo Cerrado ..... - 4 -
1.4 Automatización del riego ..... - 5 -
1.4.1 Sistemas electromecánicos y electrónicos ..... - 5 -
1.4.2 Sistemas de riego inteligentes ..... - 6 -
1.5 Arduino ..... - 6 -
1.5.1 Facultades de Arduino ..... - 7 -
1.6 Plataforma Internet de las cosas (IoT) ..... - 7 -
2 Motivación ..... - 9 -
3 Objetivos ..... - 9 -
4 Materiales ..... - 10 -
4.1 Plataforma IoT para Arduino ..... - 10 -
4.2 Microcontrolador Arduino. ..... - 12 -
4.2.1 Familia Arduino MKR. ..... - 12 -
4.2.2 Arduino Uno Wifi REV2 ..... - 13 -
4.2.3 Familia Arduino Nano ..... - 13 -
4.2.4 Otras placas de desarrollo de IoT ..... - 14 -
4.3 Sensor de humedad de suelo. ..... - 14 -
4.4 Sensor humedad relativo y temperatura. ..... - 15 -
4.5 Sensor Luz LDR. ..... - 15 -
4.6 Válvula Solenoide ..... - 16 -
4.7 Driver de potencia ..... - 17 -
4.8 Escudo Arduino. MKR ENV ..... - 18 -
4.9 Conclusiones ..... - 19 -
5 Desarrollo ..... - 23 -
5.1 Esquema de diseño ..... - 23 -
5.2 Aplicación con Blynk ..... - 24 -
5.3 Software con interfaz de Arduino. ..... - 33 -
5.4 Código ..... - 34 -
5.4.1 Librerías ..... - 34 -
5.4.2 Declaraciones de las variables globales ..... - 35 -
5.4.3 Pines virtuales ..... - 37 -
5.4.4 Gestor de conexión al servidor ..... - 42 -
5.4.5 Funciones ..... - 42 -
6 Resultados ..... - 46 -
7 Conclusiones y líneas de futuro ..... - 48 -
8 Bibliografía ..... - 49 -

Índice de tablas

Tabla 1. Comparativas placas Wifi. [14] .....- 20 -

Índice de figuras

Figura 1. Sistema de control. Lazo abierto. Fuente: Google imágenes.....- 4 -
Figura 2. Sistema de control. Lazo cerrado. Fuente Google imágenes .....- 4 -
Figura 3. Familia Arduino MKR. Fuente https://store.arduino.cc/ .....- 12 -
Figura 4. Arduino Uno Wifi Rev.2 Fuente: https://store.arduino.cc/.....- 13 -
Figura 5. Familia Arduino Nano. Fuente: https://store.arduino.cc/ .....- 13 -
Figura 6. Higrómetro YL-69 .....- 14 -
Figura 7. Sensor humedad y temperatura DHT-11 .....- 15 -
Figura 8. Fotorresistencia. .....- 16 -
Figura 9. Electroválvula 12 V.....- 17 -
Figura 10. Relay 2 canales. .....- 18 -
Figura 11. Escudo MKR ENV. Fuente: store.arduino.cc .....- 18 -
Figura 12. Diagrama de pines. Fuente: store.arduino.cc.....- 21 -
Figura 13. Diagrama de flujo .....- 23 -
Figura 14. Asignación de pines virtuales
Figura 15. Aplicación en funcionamiento.....- 24 -
Figura 16. Esquema de funcionamiento. Fuente: Blynk.....- 24 -
Figura 17. Blynk en Play Store .....- 25 -
Figura 18. Creación de una cuenta nueva.....- 26 -
Figura 19. Datos del proyecto
Figura 20. Blynk envía un email con el token. .....- 26 -
Figura 21. Elementos o widgets disponible .....- 27 -
Figura 22. Propiedades de las notificaciones push .....- 28 -
Figura 23. Configuración reloj.....- 29 -
Figura 24. Configuraciones monitorización de valores de temperatura, humedad de tierra y ambiente.....- 30 -
Figura 25. Configuraciones para los botones de "Auto/Manual" y "Temporizador"..- 30 -
Figura 26. Configuración del programador.....- 31 -
Figura 27. Interfaz programador.....- 31 -
Figura 28. Configuración del terminal de información .....- 32 -
Figura 29. Software de programación Arduino IDE .....- 33 -
Figura 30. Control SAMD Core para placa MKR 1010 Wifi instalado.....- 33 -
Figura 31. Librerías instaladas .....- 34 -
Figura 32. Credenciales de conexión a Wifi y Blynk.....- 35 -
Figura 33. Variable del relé.....- 35 -
Figura 34. Variables para el higrómetro .....- 35 -
Figura 35. Variables para el control de la aplicación móvil.....- 35 -
Figura 36. Variables para el formato día y hora .....- 35 -
Figura 37. Variables para sensores de escudo MKR .....- 36 -
Figura 38. Variables para conexión Wifi.....- 36 -
Figura 39. Variable para conexión al servidor de Blynk .....- 36 -
Figura 40. Configuración de los widgets.....- 36 -
Figura 41. Modo Automático o Manual.....- 37 -
Figura 42. Pin virtual (V1). Umbral para riego automático.....- 38 -
Figura 43. Botón del Temporizador .....- 38 -
Figura 44. Entrada de tiempo (V10). Fecha y hora .....- 39 -
Figura 45. Entrada de tiempo (V10). Ajuste día .....- 39 -
Figura 46. Entrada de tiempo (V10). Error en el temporizador.....- 40 -
Figura 47. Entrada de tiempo (V10). Temporizador no iniciado.....- 40 -
Figura 48. Entrada de tiempo (V10). Temporizador On/Off.....- 41 -
Figura 49. Entrada de tiempo (V10). Temporizador finalizado y próximo día de riego .....- 41 -
Figura 50. Función de conexión al servidor de Blynk.....- 42 -
Figura 51. Función de comprobación y mantenimiento de conexiones.....- 43 -

Figura 52. Función para cálculo de los todos los sensores.....- 43 -
Figura 53. Función gestora del programador .....- 44 -
Figura 54. Función setup().....- 44 -
Figura 55. Función loop().....- 45 -

1 Introducción

Se diseñará un sistema de control de riego automático que dispondrá de distintos tipos de sensores (humedad del aire, de la tierra y temperatura). Los sensores unidos a los parámetros configurados (horario de inicio riego, duración, repetición, etc.) decidirá de forma autónoma si se ha de realizar un riego o no.

1.1 Suelos y sus componentes primarios

El suelo contiene los nutrientes que los vegetales necesitan para desarrollarse. Los nutrientes más importantes son el nitrógeno, el fósforo y potasio, los cuales deben encontrarse en forma asimilable por las plantas.

Además de los nutrientes solubles asimilables por las plantas en la disolución del agua existen unos componentes primarios del suelo:

• Materia inorgánica. Los compuestos inorgánicos en estado mineral no son asimilables por las plantas, pero cumplen la función de retener o almacenar agua; por ejemplo, las moléculas de la arcilla no aportan nutrientes, pero funcionan como un almacén de nutrientes que se van acumulando en ellas.

• Materia orgánica. En los terrenos naturales los sistemas se mantienen en equilibrio gracias a que existe una constante absorción de restos de material vegetal en descomposición, pero en un suelo donde se practica la agricultura este pierde fertilidad.

• Componente líquido. El agua con sustancias en disolución como el oxígeno y dióxido de carbono forman una solución indispensable para que las raíces de las plantas puedan absorber los nutrientes de lo contrario se considera estéril.

• Gases. Las plantas absorben oxígeno para sus procesos metabólicos, el oxígeno también es necesario para que los microorganismos y las bacterias descompongan la materia orgánica y cuyos nutrientes puedan ser asimilables por las raíces de las plantas.

1.1.1 El pH

El nivel de pH del terreno favorecerá o perjudicará el buen desarrollo de las diferentes especies vegetales. La mayoría de nutrientes de las plantas se vuelven más solubles en terrenos ácidos. Las bacterias no pueden descomponer los la materia orgánica con un pH inferior a 4.7.

1.1.2 La luz

La luz, que influye sobre los organismos, proviene directa o indirectamente casi exclusivamente del Sol. La luz provee de la energía necesaria a las plantas para la fotosíntesis, con la cual se produce la materia orgánica para su crecimiento y desarrollo.

1.1.3 Humedad relativa (HR)

La humedad relativa es la relación entre la cantidad de vapor de agua que contiene la masa de aire y la que tendría si estuviera completamente saturada. La humedad se

considera un factor decisivo en la producción del cultivo en cuanto al cuajado. Además, actúa en interacción con las temperaturas.

[1]

1.2 Riego

"El riego consiste en aportar agua a los cultivos por medio del suelo para satisfacer sus necesidades hídricas que no fueron cubiertos mediante la precipitación. Se utiliza en la agricultura y en jardinería" [2].
Dichos sistemas se pueden dividir en dos subfamilias, los riegos tradicionales y los sistemas modernos.

1.2.1 Sistemas de riego tradicionales

Se entiende por riego tradicional aquel que se construyen por canales que transportan el agua hacia las zonas agrícolas. Los canales desembocan normalmente en unas arquetas que disponen de compuertas que al abrir permiten la salida del agua. Estos sistemas de riego, suponen un consumo elevado de agua y un desaprovechamiento de recursos, por lo que la tendencia es a cambiar hacia los sistemas modernos. Entre los sistemas de riego tradicionales se encuentran los siguientes tipos:
- Por arroyamiento o surcos.
- Por inundación o sumersión.
- Por infiltración o canales.
- Por drenaje.

1.2.2 Sistemas de riego modernos

Estos sistemas de riego suponen una reducción en el consumo de agua, y por lo tanto una optimización de recursos; un riego uniforme, obteniendo un caudal regulado mediante la presión del agua. A diferencia de los riegos tradicionales, la instalación de estos sistemas no requiere de construcciones, sino que la conducción de agua se realiza mediante tuberías de PVC hasta los puntos de riego determinado, en función del sistema de instalado. Dichos sistemas se encuentras controlados por unos sistemas programadores que controlan el paso del agua. Dentro de estos sistemas de riego encontramos los siguientes tipos:

- Por aspersión o difusión. En este tipo de sistema los elementos de riego son unas turbinas, aspersores o difusores. El elemento común A destacar Es que se tratan de elementos de riego sectoriales, por lo que cada uno abarca unos determinados metros de radio que regará cuando funcionen. Por ello deben ir sucediéndose a lo largo de la zona que se quiera regar hasta cubrir toda la extensión.
El tiempo de riego de unas zonas a otras con este tipo de riego puede variar, así como las veces que es necesario regar al día según las necesidades del terreno.

- Por goteo o riego localizado. Los elementos de riego en estas instalaciones son unos goteros que son instalados justo en las plantas que queramos que sean regadas. Al ser una acción localizada el consumo de agua se reduce considerablemente, al no ser un riego indiscriminado. Para estos tipos de riego, la programación (tiempo de riego y horarios) varía al igual que en el caso anterior.

- Por hidroponía. Este tipo de riego, cada vez más extendido en la agricultura, es algo especial y diferente a los vistos anteriormente. En este tipo de riego las plantas no se

encuentran plantadas en tierra (no usan como sustrato la tierra), sino que en su lugar se usa fibra de coco, u otros elementos que directamente sustentan a la planta. En el riego por hidroponía la planta toma los nutrientes directamente del agua (solución nutritiva), por lo que tiene características especiales con respecto a los vistos anteriormente. El agua en ningún caso se pierde (excepto por evaporación) y forma parte de un circuito cerrado en el que cada cierto tiempo se tiene que reponer el agua perdida por evaporación y equilibrar los nutrientes disueltos en el agua.

[2, 3, 4, 5, 6]

1.3 Generalidades del sistema de control

En el manejo de un sistema de riego es fundamental determinar el momento más adecuado para regar y la cantidad de agua a aplicar en función, entre otros factores, del estado de humedad del suelo o la planta y de la uniformidad en el reparto de agua del sistema.

A la hora de automatizar un sistema de riego se deben tener en cuenta todos los elementos que integran su sistema de control. Los componentes de los sistemas de control se pueden clasificar en cuatro grandes grupos: sensores y transductores (tensiómetros, manómetros, presostatos, etc.), actuadores (interruptores, electroválvulas, válvulas motorizadas, bombas, variadores de velocidad, etc.), acondicionadores de señal para que la entienda el sistema y unidades de control (ordenadores, programadores, etc.).

Los sistemas para la automatización de una instalación de riego tienen una serie de características similares a cualquier sistema de control de un determinado proceso. Los sistemas de control se han empleado fundamentalmente en el sector industrial, pero con el desarrollo en los últimos años de la informática y la microelectrónica y el abaratamiento de este tipo de dispositivos, se ha introducido en el sector agroalimentario, incluyendo su aplicación a las instalaciones de riego y fertilización.

1.3.1 Sistema de Control de Lazo Abierto

Es el sistema en el que la señal de salida no interviene en el comportamiento del sistema nuevamente, no hay retroalimentación hacia el controlador para que éste pueda ajustar la acción de control. Es decir, la señal de salida no se convierte en señal de entrada para el controlador.

Estos sistemas se caracterizan por:

- Ser sencillos y de fácil concepto.
- Nada asegura su estabilidad ante una perturbación.
- La salida no se compara con la entrada.
- Ser afectado por las perturbaciones. Estas pueden ser tangibles o intangibles.
- La precisión depende de la previa calibración del sistema.

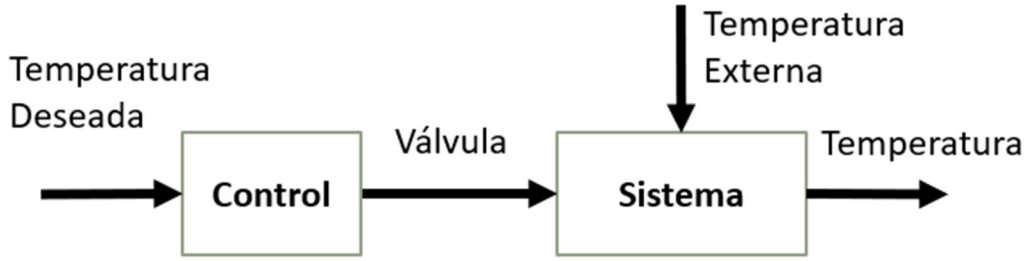

Figura 1. Sistema de control. Lazo abierto. Fuente: Google imágenes

1.3.2 Sistema de Control de Lazo Cerrado

Entendemos por control automático el mantenimiento de un valor deseado dentro de un intervalo, su funcionamiento se basa en medir el valor deseado y compararlo con el intervalo de valores aceptables utilizando la diferencia para proceder a reducirla. Por esto el control automático exige un lazo cerrado de acción y reacción que funcione sin intervención humana. La eliminación de errores y un aumento en la seguridad de los procesos es una importante contribución del uso y aplicación de la técnica de control.
Es importante destacar que anterior a la aplicación masiva de las técnicas de control automático, era el hombre el que aplicaba sus capacidades de cálculo e incluso su fuerza física para la ejecución del control de un proceso o máquina asociada a la producción. En la actualidad, gracias al desarrollo y aplicación de las técnicas modernas de control, un gran número de tareas y cálculos asociados a la manipulación de las variables ha sido delegado a computadoras, controladores y accionamientos especializados para el logro de los requerimientos del sistema.
Esquemáticamente podemos representar cualquier sistema de control por medio de un diagrama de bloques, donde se puede notar la conformación del lazo cerrado.

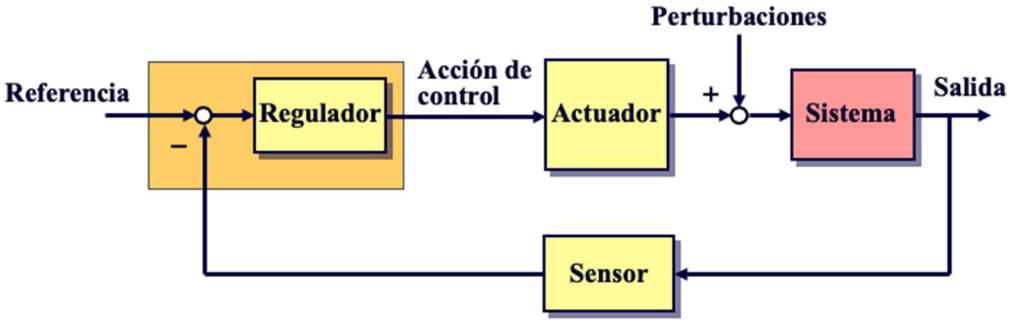

Figura 2. Sistema de control. Lazo cerrado. Fuente Google imágenes

Describiendo el esquema anterior, una referencia dada, que no es más que el valor deseado de la variable de salida, se lo compara con la misma, de lo que da como resultado una señal de error (e= ref – salida).
Esta señal es usada por el controlador para calcular una acción de control, que es enviada al actuador, la variable de salida es medida con un sensor y eventualmente transformada en una señal físicamente compatible con la referencia.
Esta idea de medir la variable y realimentarla para efectuar una comparación y saber que tan alejados estamos del valor deseado, es el concepto fundamental de todo sistema de control. Son los sistemas en los que la acción de control está en función de

la señal de salida. Los sistemas de circuito cerrado usan la retroalimentación desde un resultado final para ajustar la acción de control en consecuencia.
El control en lazo cerrado es imprescindible cuando se da alguna de las siguientes circunstancias:

- Cuando un proceso no es posible de regular por el hombre.
- Una producción a gran escala que exige grandes instalaciones y el hombre no es capaz de manejar.
- Vigilar un proceso es especialmente difícil en algunos casos y requiere una atención que el hombre puede perder fácilmente por cansancio o despiste, con los consiguientes riesgos que ello pueda ocasionar al trabajador y al proceso.

Sus características son:

- Ser complejos, pero amplios en cantidad de parámetros.
- La salida se compara con la entrada y le afecta para el control del sistema.
- Su propiedad de retroalimentación.
- Ser más estable a perturbaciones y variaciones internas.

[7]

1.4 Automatización del riego

En el campo de la electrónica el desarrollo tecnológico conseguido permite realizar de forma automática el riego de jardines, áreas verdes y áreas de cultivos, lo que ha dado lugar a un mayor control y facilidad de manejo de las instalaciones y a una disminución de los costes de mantenimiento.
La automatización de sistemas de riego ofrece numerosas posibilidades que van desde la programación de pequeñas operaciones, como la apertura o cierre de una válvula, hasta la realización de una programación integral del riego, que permite realizar de forma automática distintas operaciones además del riego.

Se entiende por un sistema programador de riego aquel que nos permite definir unos horarios para automatizar el proceso de riego. Dicho sistema, normalmente, está encargado de controlar las válvulas (electroválvulas) que dan servicio a las diferentes zonas (áreas) en las que se suele dividir una parcela, plantación, invernadero, etc.
Para su estudio lo dividiremos en dos tipos, los electromecánicos y electrónicos, y los inteligentes, en función del grado de la tecnología que utilizan en su implementación.

La automatización del riego es básicamente sustituir el control manual por controladores automáticos el cual pretende:

- ✓ Ahorro de mano de obra
- ✓ Ahorro de agua
- ✓ Ahorro de energía
- ✓ Incremento de la eficiencia del riego
- ✓ Control de costes
- ✓ Incremento de productividad de los cultivos

1.4.1 Sistemas electromecánicos y electrónicos

Los sistemas programadores electromecánicos y electrónicos comprenden desde un programador que controla múltiples zonas y con posibilidad de programar diferentes franjas horarias hasta un programador de grifo.

Estos sistemas presentan las siguientes limitaciones:

- Única franja de riego para todas las zonas, donde todas las zonas se regarán el mismo periodo de tiempo y a la misma hora.

- Este tipo de programadores normalmente tienen limitadas la capacidad de poder gestionar la repetición corta de riego en fracciones de minutos, y o bien disponen de unas limitadas franjas horarias.

- Falta de sensorización. Los programadores más avanzados sí cuentan con cierta sensorización, en la que un sensor de lluvia determinará si se ha de regar o no, pero no controlan la humedad del aire, de la tierra en las diferentes zonas o la temperatura.

- Falta de domotización. Esta es la diferencia clave con respecto a lo que describiremos como sistemas de riego inteligentes. Estos sistemas no proveen de una configuración remota vía web, control desde APP, y en algunos casos no disponen ni de pantalla para poder configurarlos.

1.4.2 Sistemas de riego inteligentes

En esta área se incluiría el sistema que se va a desarrollar. En el mercado se pueden encontrar soluciones que permiten la domotización y sensorización del riego, aunque con ciertas limitaciones:

- Los sistemas requieren de una gran inversión. Existen diferentes tipos, desde los que permiten reutilizar la instalación de las electroválvulas tradicionales hasta los que hay de sustituir todas las electroválvulas y cuya inversión es aún mayor.
- El coste de los sistemas es desorbitado, teniendo en cuenta que un sistema como el propuesto podría igualar a uno de los sistemas tradicionales, mientras que estos sistemas disponibles llegan a ser entre cuatro y cinco veces el precio los sistemas tradicionales.

1.5 Arduino

Arduino se inició en el año 2005, como una herramienta fácil para el prototipado rápido, dirigido a estudiantes sin formación en electrónica y programación. Tan pronto como llegó a una comunidad más amplia, la junta de Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos.
Arduino es una plataforma electrónica de código abierto. Sus placas están preparadas para leer entradas (sensores de luz, temperatura, sonido, pulsaciones de un botón, etc.) y convertir esto en una salida (encendiendo un LED, activando un motor, mandando un mensaje por internet, etc.) utilizando diferentes microcontroladores y microprocesadores. En general, la placa Arduino consiste en un microcontrolador sobre una placa de circuito impreso que dispone de múltiples conectores que funcionan como puertos de entrada y salida. En dichos puertos se pueden conectar placas de expansión que incrementan la funcionalidad del modelo empleado.
El software que se utiliza para programar Arduino es un entorno de desarrollo (IDE) basado en entorno de *processing*, y la estructura del lenguaje de programación *Wiring*. Las placas son programadas mediante un ordenador donde se ejecuta el entorno de desarrollo y se carga la información mediante comunicación del puerto serie a la palca Arduino.

1.5.1 Facultades de Arduino

Arduino ofrece gran versatilidad a la hora de poder agregar componentes, tanto para sensorización como para emitir respuestas programadas en la placa. Es por ello que su uso se ha visto muy extendido en los últimos años creando una gran comunidad internacional que cuenta con estudiantes, profesores, aficionados a la tecnología, etc.

Existen miles de proyectos realizados con Arduino, desde objetos de uso diario (conectar una tostadora a la red y accionarla a través del wifi), hasta complejos instrumentos para experimentos y mediciones científicas.

Aunque existen otras placas programables con microcontroladores disponibles en el mercado actualmente; Arduino ofrece varias ventajas:

- ✓ Es económico. El precio es bastante más bajo que sus competidoras.
- ✓ Multiplataforma. El software de desarrollo se puede ejecutar en cualquier sistema operativo para ordenadores (Windows, Macintosh OSX o Linux).
- ✓ Un entorno de desarrollo simple y limpio. Posee una interfaz de uso sencilla para los principiantes.
- ✓ Código abierto y software ampliable. El software de Arduino está publicado como herramientas de código abierto, disponible para que los programadores expertos puedan incrementar las funcionalidades del mismo.
- ✓ Código abierto y hardware ampliable. Los planos de las placas de Arduino se encuentran publicados. Permite crear propias versiones de los módulos, ahorrando así bastante dinero.

[8]

1.6 Plataforma Internet de las cosas (IoT)

El Internet de las cosas (IoT) es una red interconectada de cosas que tiene una función específica que desempeñar. Las cosas pueden ser diferentes dispositivos, como relojes inteligentes, teléfonos móviles, cafeteras, sistemas de calefacción doméstica, etc.

Los dispositivos interconectados le permiten recopilar información útil, automatizar actividades y ahorrar energía o dinero, por ejemplo. En todo esto, definitivamente debe prestar atención a la seguridad para que nadie pueda obtener información personal sobre el usuario o controlar sus dispositivos.

Ventajas:

- ✓ Ahorrando dinero.
- ✓ Ahorro de tiempo (automatización).
- ✓ Información y datos.
- ✓ Mejor planificación de recursos.

Desventajas:

- × Dependencia de la tecnología
- × Seguridad
- × Protección de Datos
- × Complejidad

Cuando creamos proyectos IoT con Arduino, tenemos varias maneras de monitorizar los datos que captamos a través de los sensores. La más básica y que requiere tener el dispositivo conectado a un ordenador es a través del monitor serie. Si tenemos algún display como un LCD o una pantalla TFT, ya podremos desconectar el Arduino del

ordenador y llevarlo a cualquier sito. Pero existe una tercera opción, quizás la más idónea, para poder ver los datos y la información, utilizar una plataforma para proyectos del IoT.

Es interesante hacerse una idea general de lo que sería un sistema basado en plataformas en la nube. En este sistema intervienen tres elementos principales:

- El dispositivo conectado o del IoT. Configurar y crear el circuito con la conectividad (Wifi, Ethernet,..)
- La plataforma en la nube. Configurar plataforma IoT.
- Los dispositivos que consumen la información en la plataforma del IoT. Acceso mediante API a la información en la nube.

Cada uno de estos sistemas se trata por separado y se comunican entre ellos a través de protocolos de comunicación. Estos protocolos deberían ser un estándar para que, independientemente de la plataforma, se puedan comunicar.

En la parte correspondiente se analizarán las diferentes plataformas que hay en el mercado. Me centraré sobre todo en aquellas que nos den un acceso gratuito, aunque sea limitado.

Los protocolos de comunicación más utilizados son HTTP, MQTT y CoAP. Además, para comunicarnos, existen diferentes redes como LoRa o SigFox. Son redes WAN para el IoT y una alternativa a los sistemas tradicionales de comunicación. Más adelante, se nombrarán los diferentes dispositivos Arduino que pueden llegar a utilizar la diversidad de protocolos.

Tener API de acceso de datos, es algo muy importante que hay que valorar a la hora de crear proyectos IoT con Arduino. Tener una API (Application Programming Interface) nos permitirá consultar, modificar y borrar la información desde otros dispositivos. Al final tenemos que entender que es una capa de comunicación estándar para conectarnos a los datos. No hay un lenguaje específico para crear servicios web, existen diferentes APIs, frameworks, librerías y herramientas que nos facilitan esta tarea. El acceso a dicha API dependerá del software desde donde nos conectemos.

Cada día surgen nuevas plataformas para nuestros proyectos IoT y es complicado hacer un análisis de todos ellos dado el volumen tan grande de este tipo de plataformas. Pero se puede hacer una clasificación dependiendo del coste y el sector al que va orientado.

El primer grupo, quizás el más interesante. En este tipo he incluido todas aquellas que nos permiten su uso de una manera gratuita, pero con limitaciones en cuanto al número de mensajes enviados y de dispositivos conectados.
Están enfocadas exclusivamente a dispositivos u objetos conectados. Esta característica las hace ideales para utilizarlas en nuestros proyectos del IoT con Arduino.

El segundo grupo, engloba a plataformas que también ofrecen servicios gratuitos o versiones de prueba. Están más centradas en ofrecer servicios globales a sistemas basados en el IoT. Ya no es solo recibir datos, en estas plataformas nos permiten almacenar webs, API para móviles, bases de datos, etc…

Podemos decir que son el paso intermedio entre las enfocadas claramente al IoT y las plataformas de las grandes corporaciones.

El siguiente grupo, serían las plataformas que ofrecen las grandes empresas y corporaciones como Google, Amazon, Microsoft, IBM, etc… Están orientadas sobre todo al sector industrial y a grandes proyectos del IoT, donde se ven involucrados cientos o miles de dispositivos.

El último grupo englobaría las plataformas de código abierto. Son todas aquellas que nos dan acceso al código sin restricciones. Podemos descargarlas e instalarlas en nuestras máquinas.

[9]

2 Motivación

En los tiempos que vivimos, donde todo el mundo está muy ocupado, y es complicado establecer una rutina de riego, toma una gran importancia el poder automatizar este proceso.

Los sistemas de riego automatizados permiten ajustar el programa de riego basándose en las condiciones climáticas de cada día, con lo que se consigue un mayor ahorro de agua a lo largo del año. Tras un análisis, diseño e implementación hacen de manera fácil un trabajo tan importante como es la tarea de regar en cualquier momento del día, pudiendo conseguir un mejor rendimiento de las plantas al optimizar el riego sin estar físicamente delante de la planta, huerto, etc.

Este tipo de sistemas son utilizados tanto en cultivos pequeños como en cultivos a gran escala, ya sean extensiones macros, huertos, pequeños jardines, patios, balcones hasta interiores de casa.

La elección de este Trabajo Fin de Grado tiene como objeto el diseñar un sistema de riego automatizado en combinación con una interfaz intuitiva y que cualquier persona pueda gestionarla sin tener conocimientos previos.

3 Objetivos

El objetivo principal, es crear un sistema de riego automatizado que permita controlar una electroválvula para realizar el riego de manera remota a través de la red Wifi mediante una aplicación, así como la conexión entre el dispositivo móvil y el microcontrolador.

Se presentan los objetivos pertenecientes al proyecto que se va a desarrollar:

- Desarrollar un sistema empotrado de control de riego automatizado y sensorización de valores esenciales en Arduino.
- Configuración de acceso a la red de comunicaciones mediante Wifi.
- Desarrollar la interfaz de conexión y supervisión entre el microcontrolador y el dispositivo móvil mediante una aplicación móvil libre.
- Demostrar el funcionamiento a través de un prototipo el cual registre las variables de humedad y controle el sistema de riego de manera remota.

4 Materiales

El proyecto tiene como objetivo la utilización de una aplicación móvil para la monitorización de un riego automatizado. Dicha aplicación necesita un hardware para obtener los parámetros del estado de las plantas para así saber sus necesidades en tiempo real, y actuar en función de estas. La obtención de los parámetros se realiza con un sensor de humedad de la tierra y sensores para la obtención de valores ambientales. El microcontrolador Arduino procesa los datos leídos y envía señales a los actuadores.

4.1 Plataforma IoT para Arduino

El IoT constituye uno de los más importantes desarrollos tecnológicos de la última década. Su potencial, permitirá cambiar nuestro estilo de vida. Gracias a las plataformas que están surgiendo en los últimos años, podemos integrar nuestros proyectos con Arduino, dentro de la nube.

No es una tarea sencilla. Requiere de conocimientos en la utilización de APIs y desarrollo web. Algunas nos ofrecen más facilidad que otras.

En cuanto a los criterios de evaluación, antes de decidir por una plataforma u otra, se debe fijar unas ciertas características de las plataformas del IoT. Donde el precio, la dificultad, los protocolos de comunicación y la integración con Arduino.

- El precio es muy importante y en muchos de los proyectos no se necesitan grandes prestaciones. Analizamos las plataformas con una versión gratuita, donde se puede ir escalando según las necesidades.
- La dificultad dependerá sobre todo del tipo de proyecto y de nuestros conocimientos técnicos. Hay plataformas que permiten configurar la adquisición de datos a través de una aplicación visual.
- Los protocolos de comunicación permitirán comunicarnos entre los objetos conectados con dispositivos de terceros.
- La integración con Arduino se puede medir si tenemos una librería a través de la cual conectarnos a la plataforma. Esto nos evita tener que utilizar librerías dependientes de los protocolos de comunicación.

Se presenta, la evaluación de las plataformas para iniciarse en el mundo de los objetos conectados:

Arduino Cloud, quizás se la más sencilla para utilizar en proyectos IoT con Arduino. Nombrar que es compatible totalmente con cualquier placa de Arduino con conectividad. Para configurar un dispositivo es sencillo, tan solo son 4 pasos muy simples. Aunque es muy sencillo conectar un dispositivo, faltaría documentación necesaria para entender cómo funciona por debajo esta plataforma.

Las limitaciones son evidentes, no tiene una API para consultar los datos desde otras aplicaciones, no tiene histórico y no representa la información en tiempo real. Según Arduino, muy pronto irán incorporando estas funcionalidades. De momento se puede usar de forma gratuita, las limitaciones las impondrán las tasas de lecturas de los sensores y los propios dispositivos.

[10]

Cayenne My Devices y **Blynk**, dos de las plataformas más sencillas de usar. Ambas a base de un gestor visual, con una configuración del dispositivo muy sencilla. Centrado en Arduino, disponen de una librería que la encontramos en el repositorio oficial.

La plataforma Cayenne presento problemas de comunicación entre el hardware y el servidor debido a un error interno de la propia aplicación. Presentando fallos de conexión en las pruebas se realizaron con el sensor de temperatura y humedad.
Una de las limitaciones es que sólo podemos enviar 10 valores por segundo, más que suficiente para monitorizar nuestros sensores.
Solo ofrecen una versión gratuita así que pude empezar a probarla. La descarté por completo, y seguí en busca de la plataforma IoT que me diera robustez y seguridad de operación.

[11]

Blynk es una plataforma de IoT independiente del hardware y basada en la nube que permite a los usuarios conectar más de 400 modelos de hardware con la nube Blynk segura y de código abierto a través de wifi, 2G-4G, LTE o Ethernet.

Esta plataforma de código abierto se puede implementar en servidores locales, en entornos de nube como AWS (Amazon Web Service) o alojada por Blynk.

Los usuarios pueden crear aplicaciones nativas de IoT para Android e iOS usando una selección de widgets prediseñados para supervisar datos de sensores, controlar la electrónica, recibir notificaciones... Los módulos de aplicaciones prediseñados permiten a los usuarios crear prototipos y probar e implementar rápidamente sus aplicaciones sin necesidad de contar con un equipo de diseño o ingeniería. Las funciones de la aplicación incluyen visualización de datos del sensor, control remoto del equipo, administración de dispositivos, controles de reglas, notificaciones.

Con la solución de Blynk, los usuarios pueden personalizar las aplicaciones de IoT con sus logotipos, fuentes, colores e iconos para publicarlas en las tiendas como propias. El aprovisionamiento está incluido, lo que permite a los clientes crear cuentas y conectar dispositivos a redes wifi o móviles en pocos pasos. Las aplicaciones se pueden conectar a cualquier número, tipo y combinación de productos fabricados. Los usuarios también pueden crear paneles personalizados para cada cliente y actualizarlos en cualquier momento.

Blynk dispone de una versión gratuita y ofrece una prueba gratis. La versión de pago de Blynk está disponible a partir de USD 199.00/mes.

[12]

ThingSpeak es la apuesta de MathWorks, los creadores de MathLabs, para el Internet de las Cosas. Es una plataforma IoT muy reconocida en el mundo “Maker”. Está enfocado exclusivamente a la construcción de aplicaciones del IoT. Permite almacenar datos, visualizarlos y exponerlos a otras APIs.
La base de esta plataforma son los canales, en ellos se almacenan los datos que le enviamos y se compone de 3 elementos:

- 8 campos para almacenar datos de cualquier tipo.
- 3 campos para almacenar la ubicación, latitud, longitud y elevación. Por supuesto que necesitaríamos un componente que nos diera esta información.
- 1 campo para almacenar el estado.

Cada uno de estos campos puede ser actualizado cada 15 segundos.
Podemos encontrar esta librería para utilizarla en proyectos del IoT con Arduino, dentro del entorno de desarrollo oficial.
Tiene una documentación muy extensa con ejemplos y es totalmente gratuita.

[13]

4.2 *Microcontrolador Arduino.*

A continuación, se van a analizar las diferentes familias de Arduino para seleccionar el modelo más adecuado para llevar a cabo la automatización del riego y la comunicación con la aplicación de control.

4.2.1 Familia Arduino MKR.

Figura 3. Familia Arduino MKR. Fuente https://store.arduino.cc/

Cada dispositivo que crea tiene su propia forma de conexión.

El Arduino MKR GSM 1400 utiliza comunicaciones móviles, permite la conexión al flujo de comunicación móvil. Es adecuado para conectarse a una red cuando no es posible realizar otras conexiones. Necesita una tarjeta SIM para conectarse a una red móvil.

Arduino MKR1000 Wifi y MKR Wifi 1010 utilizan tecnología WiFi popular. Ambas placas de desarrollo de IoT son ejecutadas por el microcontrolador SAMD21 y están equipadas con la opción de agregar una batería (Li-Po 3.7V, 700mAh mínimo). Es importante que los sensores no usen más de 3.3V.

Arduino MKR NB 1500 (LTE / banda estrecha) utiliza tecnología NB-IoT y MKR WAN 1310 LoRa. Todos ellos son pequeños, asequibles, seguros y están especialmente diseñados para la creación de prototipos.
El MKR NB 1500 utiliza el nuevo módulo de radio u-blox SARA-R410M-02B, que conecta la banda estrecha a la red de Internet de las cosas (*NB - IoT*) . Esto debería proporcionar una mejor conexión de red en lugares donde otras conexiones no se extienden. Puede cubrir hasta 25 km desde el mástil.
El corazón de la MKR WAN 1310 es el procesador SAMD21. Lo que lo hace especial es el módulo de radio CMWX1ZZABZ, que habilita la tecnología LoRa WAN. Esto significa que la transmisión de datos puede tener lugar con un consumo de energía extremadamente bajo (04 µ A). La velocidad de transferencia de datos no es muy alta, pero, se extiende más allá de Wifi o GSM.

Y MKR no es el único producto que Arduino permite conectarse a Internet. Son varias las oportunidades que disponemos.

4.2.2 Arduino Uno Wifi REV2

Para que Ardino Uno fuera adecuado para proyectos de IoT, se conectó el U-blox NINA-W102. Esto permite que el dispositivo esté conectado de forma segura a Wi-Fi (chip criptográfico ECC608) y agrega capacidad Bluetooth (BLE). La placa de desarrollo está alimentada por un microcontrolador ATmega4809 y se añade a la placa un sensor de inercia IMU (giroscopio y acelerómetro). Esta placa de desarrollo se puede utilizar en todo tipo de proyectos de IoT y robótica.

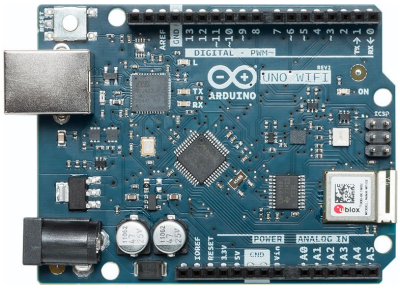

Figura 4. Arduino Uno Wifi Rev.2 Fuente: https://store.arduino.cc/

4.2.3 Familia Arduino Nano

Internet de las cosas también ha complementado al tradicional Arduino Nano y ha desarrollado cuatro versiones diferentes.

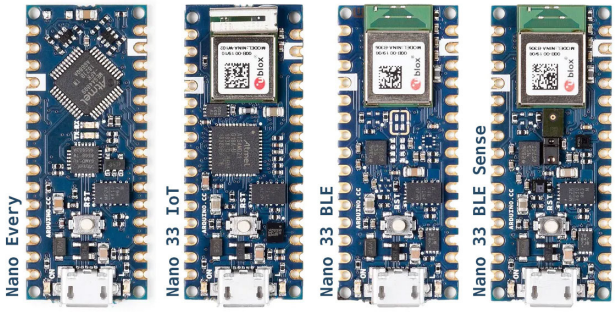

Figura 5. Familia Arduino Nano. Fuente: https://store.arduino.cc/

- **Nano Every**: reemplaza el llamado microcontrolador Nano estándar. En su corazón está el microchip ATMega4809, que es muchas veces más rápido y tiene más memoria. Adecuado para dispositivos que no necesitan estar conectados a una red.

- **Nano 33 IoT**: el corazón es el procesador ATSAMD21 y gracias al módulo U-blox (NINA-W10) puede conectarlo de forma segura (chip criptográfico ECC608) a WiFi. Además, permite la conexión a través de Bluetooth (BLE).
  Esta placa es adecuada para proyectos de IoT, alarmas de vibración, podómetros y robots. Los sensores conectados nunca deben devolver más de 3.3V de voltaje

- **El Nano 33 BLE**: BLE (Bluetooth Low Energy) significa que se utiliza una versión especialmente energéticamente eficiente. En su corazón está el procesador nRF52840 y está equipado con un giroscopio, un acelerómetro y un magnetómetro con sensor inercial (IMU de 9 ejes).

Adecuado para proyectos en un área más pequeña e ideal para el desarrollo de robots que se pueden controlar, por ejemplo, con un teléfono inteligente.

- **Nano 33 BLE Sense**: similar al microcontrolador anterior, con aún más sensores agregados.

[14]

4.2.4 Otras placas de desarrollo de IoT

Los dispositivos Arduino no son los únicos microcontroladores programables. Realmente hay muchos de ellos y los llamados "chinos" también son adecuados para aprender. Se debe de tener en cuenta que tengan ESP8266 o ESP32. En 2014, la empresa china *Espressif Systems* presentó un microcontrolador ESP8266 de 32 bits, habilitado para WiFi y muy económico.

4.3 Sensor de humedad de suelo.

Este sensor tiene la capacidad de medir la humedad del suelo. Aplicando una pequeña tensión entre los terminales del módulo hace pasar una corriente que depende básicamente de la resistencia que se genera en el suelo y ésta depende mucho de la humedad. Por lo tanto, al aumentar la humedad la corriente crece y al bajar la corriente disminuye. Existen dos modelos diferentes, donde tan solo cambia el nombre del dispositivo. Siendo, el sensor FC-28, HL-69 y el sensor YL-69, de características y precios semejantes, este último como elegido para el desarrollo.

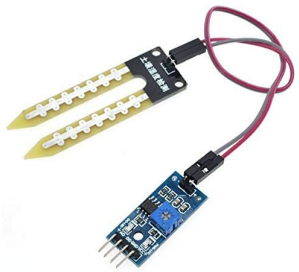

Figura 6. Higrómetro YL-69

Consiste en una sonda YL-69 con dos terminales separados adecuadamente y un módulo YL-38 que contiene un circuito comparador LM393 SMD, muy estable, un led de encendido y otro de activación de salida digital. Este último presenta 2 pines de conexión hacia el módulo YL-69, 2 pines para la alimentación y 2 pines de datos. VCC, GND, D0, A0.
Especificaciones técnicas:

- Voltaje de entrada: 3.3 - 5 V.
- Voltaje de salida: 0 ~ 4.2 V.
- Corriente: 35 mA.
- VCC: Tensión de alimentación.
- GND: Tierra.
- A0: Salida analógica que entrega una tensión proporcional a la humedad. Puede ser medida directamente desde un puerto analógico en un microcontrolador.
- D0: Salida digital; este módulo permite ajustar cuándo el nivel lógico en esta salida pasa de bajo a alto mediante el potenciómetro.
- Dimensiones YL-38: 30 x 16 mm.
- Dimensiones YL-69: 60 x 30 mm.

[15]

4.4 Sensor humedad relativo y temperatura.

Es un sensor digital de temperatura y humedad relativa de bajo costo y fácil uso. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Utilizado en aplicaciones académicas relacionadas al control automático de temperatura, aire acondicionado, monitoreo ambiental en agricultura y más.

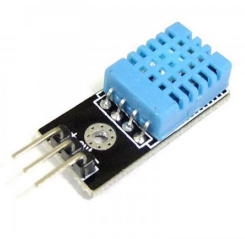

Figura 7. Sensor humedad y temperatura DHT-11

Utilizar el sensor DHT11 es muy sencillo tanto a nivel de software como hardware. A nivel de software se dispone de librerías para Arduino con soporte para el protocolo "Single bus".

Especificaciones técnicas:

- Voltaje de Operación: 3V - 5V DC.
- Rango de medición de temperatura: 0 a 50 °C.
- Precisión de medición de temperatura: ±2.0 °C.
- Resolución Temperatura: 0.1°C.
- Rango de medición de humedad: 20% a 90% RH.
- Precisión de medición de humedad: 5% RH.
- Resolución Humedad: 1% RH.
- La única desventaja es que sólo se puede obtener nuevos datos cada 2 segundos, asegurando alta estabilidad y fiabilidad a lo largo del tiempo.
- La distancia máxima recomendable de longitud de cable es de 20m., de preferencia utilizar cable apantallado.
- Proteger el sensor de la luz directa del sol (radiación UV).

En comparación con el DHT22 y DHT21, este sensor es menos preciso, menos exacto y funciona en un rango más pequeño de temperatura / humedad, pero su empaque es más pequeño y de menor costo.

[16]

4.5 Sensor Luz LDR.

Una fotorresistencia, también llamados LDR lineal o fotodiodo, es un componente eléctrico, el cual posee una resistencia capaz de variar su magnitud al estar en contacto con distintas magnitudes de intensidad lumínica. Está conformado por una célula fotorreceptora y dos pastillas.

Figura 8. Fotorresistencia.

La base del funcionamiento de una fotorresistencia radica en su componente principal, el sulfuro de cadmio (CdS). Este componente químico es un semiconductor que tiene la capacidad de variar su resistencia según la cantidad de luz que en él incida.
Cuanto mayor intensidad es la luz que incide sobre el sulfuro de cadmio, más baja es la resistencia, es decir mayor facilidad de los electrones para moverse.

Características de las fotorresistencias

- Conformados por un semiconductor de alta resistencia.
- La resistencia varía entre 1MΩ, cuando hay mucha oscuridad y 100Ω con altas intensidades de luz.
- Tiempo de respuesta de 1 décima de segundo cuando varia la intensidad lumínica.
- De muy bajo costo y tecnología sencilla.

Ventajas de las fotorresistencias:

- Puede abarcar superficies grandes.
- Es fácil de usar. Solo con conectar las terminales funciona.
- El costo de adquirir una fotorresistencia es bajo y permite abaratar costos de personal.
- Tiene una gran relación luz, oscuridad, electricidad.

Desventajas de las fotorresistencias:

- Se debe elegir la fotorresistencia en función del tipo de luz que se quiere medir (inflarrojo, luz visible o ultravioleta).
- Se produce lo que se conoce como efecto histéresis. Lo que hace es quedar guardado en lo que sería “la memoria” del metal, ciertas condiciones, que al tener que cambiar por cambios de luz, retrasan el funcionamiento.
- No pueden ser usados en lugares donde la luz varia rápidamente, ya que tiene una velocidad de respuesta lenta.
- La variación de la resistencia en función de la intensidad lumínica no es lineal.

[17]

4.6 Válvula Solenoide

La principal elección de la electroválvula es por la tensión de funcionamiento, ya que disponía de un transformador de 220v a 12v. El segundo motivo para la selección es el bajo costo de la herramienta para el control del flujo de agua en una tubería con la ayuda de esta válvula solenoide. Las válvulas solenoides son un tipo de electroválvula todo/nada o abierto/cerrado.

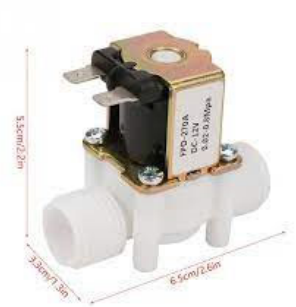

Figura 9. Electroválvula 12 V

Tienen dos partes: el solenoide y el cuerpo de plástico. El solenoide es un electroimán que al ser energizado se desplaza junto con el diafragma de la válvula y permite el paso del fluido. La válvula se mantiene abierta mientras el solenoide está energizado. Cuando no está alimentado un resorte se encarga de regresar la válvula a su posición de reposo, que en este caso es del tipo normalmente cerrada (NC).

El cuerpo de la válvula está fabricado en plástico con roscados machos a ambos lados de 1/2" tipo NPS (recta).

Especificaciones técnicas:

- Voltaje de operación: 12V DC.
- Corriente de operación: 0.6A.
- Potencia consumo: 8W.
- Temperatura de funcionamiento: 5ºC a 100ºC.
- Presión de funcionamiento mínima: 0.02 MPa (0.2 Bar).
- Presión de funcionamiento máximo: 0.8 MPa (8 Bar).
- Tiempo de respuesta (apertura): ≤ 0.15 s.
- Tiempo de respuesta (cerrado): ≤ 0.3 s.
- Conector tubería: Rosca externa 1/2" NPS Macho.
- Reposo: Normalmente cerrado.
- Tipo de válvula: Diafragma.
- Adecuado para agua y fluidos de baja viscosidad.
- No se recomienda para aplicaciones que usan solo la gravedad, por la presión mínima de funcionamiento.
- Material cuerpo: Plástico ABS.

Controlar la válvula es muy sencillo con la ayuda de un microcontrolador y un driver de potencia. El uso del driver entre el microcontrolador y la válvula es necesario pues la corriente y voltaje de la válvula son mayores a los usados por el microcontrolador. Si conectamos directamente el microcontrolador a la válvula es seguro que dañaremos nuestro circuito.

[Ficha técnica]

4.7 Driver de potencia

Muchas veces es necesario manejar cargas de más de 2A de corriente. Puede manejar cargas como: válvulas solenoide 24V DC, motores DC, Luces LED de alta potencia.
El microcontrolador Arduino y, en cuanto, al driver de potencia recomendamos utilizar transistores mosfet como:

- Módulo Mosfet IRF520

Este módulo es capaz de manejar una carga de hasta 9A. Soporta control por PWM.

Los transistores de tipo Mosfet presentan mejores características que los BJT en aplicaciones de Encendido/Apagado de cargas de alto amperaje. Para activar el mosfet se debe de enviar 5 Voltios al Gate del Mosfet, esto permitirá que la corriente fluya a través de la carga y esta se active.
Cuando se utiliza este módulo en aplicaciones de alto consumo puede ser necesario utilizar un disipador de calor (para corrientes menores a 6 A no es necesario ningún disipador).

- Módulo Mosfet IRF540

Este módulo, semejante al anterior, este es capaz de manejar una carga de hasta 30A.

O los clásicos relay como el Relay de 2CH.

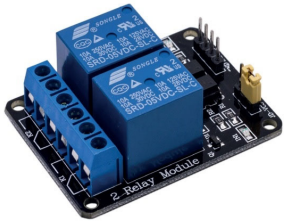

Figura 10. Relay 2 canales.

Relays o Reles, estos dispositivos permiten controlar cargas de alto voltaje con una señal pequeña. Este módulo Relay activa la salida normalmente abierta (NO: Normally Open) al recibir un "0" lógico (0 Voltios) y desactiva la salida con un "1" lógico (5 voltios).

Especificaciones técnicas:

- Voltaje de Alimentación: 12V DC.
- Señal de Control: TTL (3.3V o 5V).
- Para activar salida NO: 0 Voltios.
- Nº de Relays (canales): 2 CH.
- Capacidad máx: 10A/250VAC, 10A/30VDC.
- Corriente máx: 10A (NO), 5A (NC).
- Tiempo de acción: 10 ms / 5 ms.
- Entradas Optoacopladas.
- Indicadores LED de activación.

[Ficha técnica]

4.8 Escudo Arduino. MKR ENV

MKR ENV shield es una placa de expansión MKR WIFI 1010.

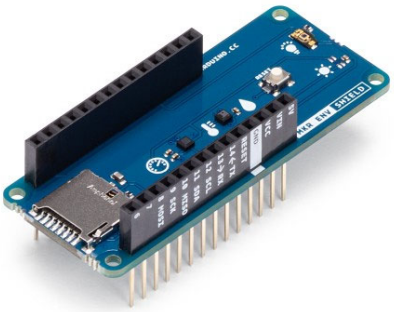

Figura 11. Escudo MKR ENV. Fuente: store.arduino.cc

Permite que adquiera datos ambientales con varios sensores que te permiten medir:

- Presión de aire (260 - 1260 hPa, LPS22HB).
- Temperatura (± 1 ° C, -40 a +60 ° C, HTS221).
- Humedad (± 3,5% rH, 20 a + 80% rH, HTS221).
- Intensidad UV (VEML6075).
- Intensidad de radiación UVb (VEML6075).
- Índice UV (calculado).
- Intensidad de luz (LUX, TEMT6000).

Posee la peculiaridad de poder almacenar directamente en la tarjeta microSD.

[18]

4.9 Conclusiones

Finalmente, después de realizar el análisis de los diferentes módulos con tecnología Wifi y todas las herramientas necesarias para la gestión del riego automatizado. La selección de la placa fue el Arduino MKR Wifi 1010 con la plataforma gestora de la aplicación de Blynk.

El Arduino MKR Wifi 1010 fue elegido por su gran versatilidad a la hora de poder agregarle componentes para aumentar su conectividad y pudiendo añadirle escudos a la propia placa. El interés principal del proyecto es el de realizar un sistema de riego que pudiera ser controlado de forma telemática. Esta placa, a diferencia de otras placas Arduino permite poder optar por diferentes soluciones.

- MKR 1010 simplifica la creación de prototipos de aplicaciones IoT basadas en Wifi, fácil de conectar y configurar a otros productos de Arduino.
- El módulo híbrido de conectividad Wifi / BLE tiene un modo de bajo consumo de energía, mejorando la vida útil de las baterías.
- La placa puede conectarse a cualquier tipo de red Wifi existente o puedes usarla para crear tu propio Punto de Acceso.
- Fácil de alimentar con un cable USB o una batería Li-Po externa de 3.7V.
- Diferencia entre ESP8266 y Arduino MKR Wifi 1010. Desde el lanzamiento del módulo ESP8266, que forma parte de la serie Node MCU, se ha utilizado cada vez más en la mayoría de los proyectos IoT.

La popularidad del dispositivo se debe a su bajo precio y a su capacidad de ser programado usando Arduino IDE.
Comparando las especificaciones y la funcionalidad de ambas tarjetas, es obvio que el MKR tiene más pines digitales, opción de circuito de carga de batería Li-Po.
Por destacar alguno de los dos inconvenientes considerables son que el MKR es al menos 8 veces más costoso que el ESP8266 y no cuenta con un fuerte apoyo de la comunidad. Por lo tanto, depende puramente del diseñador para considerar los pros y los contras y seleccionar hacer su elección ideal.

Tabla 1. Comparativas placas Wifi. [14]

	ARDUINO UNO WIFI REV2	ARDUINO MKR 1000 WIFI	ARDUINO MKR WIFI 1010	ARDUINO NANO 33 IOT
Microcontrolador	ATmega4809	SAMD21 Cortex®-M0 + MCU ARM de bajo consumo de 32 bits		
Módulo de radio	u-blox NINA-W102	WINC1500 low power	u-blox NINA-W102	
Fuente de alimentación de la placa (USB / VIN)	7 - 12V	5V		
Tensión de funcionamiento	5 V	3,3 V		
Velocidad de reloj	16 MHz	32,768 kHz (RTC), 48 MHz		48 MHz
Memoria flash de la CPU	48 KB	256 KB		
SRAM	6 KB	32 KB		
Pines de entrada / salida digita	14	8		14
Pines PWM	5	12	13	11
I2C	NO	1		
Batería compatible	NO	Li-Po de celda única, 3,7 V, 700 mAh mínimo	Li-Po de celda única, 3,7 V, 1024 mAh como mínimo	NO
Precio	45 €	36 €	28 €	20 €

La elección frente a la Arduino MKR1000 Wifi, porque es la evolución mejorada del desarrollo que utiliza el módulo U-blox NINA-W102 para conectarse a la red, equipando con el módulo ESP32 proporcionando hasta un máximo de 400 metros en rango de Wifi y su bajo consumo de energía. Además de Wifi, proporciona capacidad Bluetooth y una conexión segura a Internet (chip criptográfico ECC508). Como mejora, la placa tiene un LED RGB y un puerto I2C adicional para los módulos.

Si bien otras placas de similares características, ya incluyen el módulo Wifi y *ethernet* integrados (Arduino Uno Wifi REV2), el coste de la misma es muy elevado y sin tener opción a estar alimentado a una batería Li-Po externa, para realizar futuras líneas de mejora del proyecto.

También de similares características al Arduino MKR Wifi 1010, el Nano 33 IoT, tiene el mismo procesador y chipset inalámbrico, con la única deficiencia que tampoco se puede conectar a una batería Li-Po externa.

Por otro lado, las características que se exponen (Tabla 1), son superiores al resto de la gama de precios similares, y por la cantidad de pines disponibles para la conexión de sensores y relés futuros, nos permitía en un futuro incrementar el número de zonas que controlaría, sin que esto conllevara un cambio de la placa programable.

La selección de la placa también se basó en el desconocimiento inicial de cuánta memoria requeriría el programa a desarrollar en cuestión, tanto flash como SDRAM. Por ello, también fue un motivo, a parte del precio de la placa, el descartar el módulo Arduino Uno Wifi REV2. Esto, ante la necesidad de tener pines de conexiones analógicos y digitales en cantidad para posibles ampliaciones y con la posibilidad de versiones posteriores del proyecto, dejo como posibles opciones la placa Arduino MKR1000 Wifi y Arduino MKR Wifi 1010. Siendo, este último, el elegido por ser el sucesor y teniendo la mejora del módulo de radio de Wifi y Bluetooth.

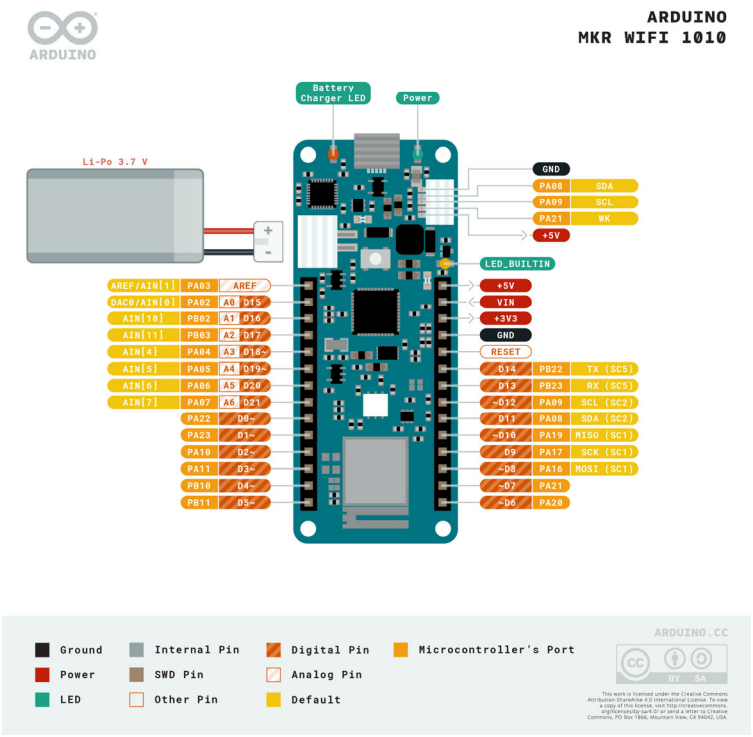

Figura 12. Diagrama de pines. Fuente: store.arduino.cc

Para la toma de datos de los sensores, el escudo MKR ENV, es la herramienta elegida. Esta elección se basó en gran medida por el potencial de almacenar los datos recopilados localmente, este escudo tiene una ranura para una tarjeta microSD. En cuanto al precio del escudo y todos los sensores comprados individualmente, el precio total es similar.

La placa permite que una placa MKR adquiera datos ambientales recopilados por una serie de sensores.

Estos sensores son de última generación y miden:

- Rango de presión absoluta: 260 a 1260 hPa.
- Rango de humedad: 0 - 100%.
- Rango de temperatura -40 +120 °C.
- Rango de lux de 10 a 100.000 lux.
- Resolución UVA / UVB.

Cuando se utiliza una placa de expansión determinada, estos sensores ocupan determinadas salidas. El sensor de luz es A2 y la ranura para tarjetas SD usa D4 (SD CS), D8 (MOSI), D9 (SCK) y D10 (MISO). El sensor de temperatura y humedad utiliza el D6 y el sensor de presión absoluta el pin digital D7.

El sensor de humedad del suelo, el YL-69 es ampliamente empleado en sistemas automáticos de riego para detectar cuando es necesario activar el sistema de bombeo, apertura de electroválvulas, accionamiento de aspersores y así como diferentes dispositivos para el riego. Permite obtener la medición como valor analógico o como una salida digital, activada cuando la humedad supera un cierto umbral. Utilizando la entrada analógica (A1). Los valores obtenidos van desde 0 sumergido en agua, a 1023 en el aire (o en un suelo muy seco). Un suelo ligeramente húmero daría valores típicos de 600-700. Un suelo seco tendrá valores de 800-1023.

En cuanto a la plataforma Blynk, mediante el control del hardware de forma remota mostrando los datos de sensores, almacenar datos y visualizarlos.

Los tres principales componentes de la plataforma son:

1. La APP de Blynk, donde se permite crear interfaces utilizando widgets que la propia plataforma muestra.
2. El Blynk server, el principal gestor de todas las comunicaciones entre el dispositivo móvil inteligente.
3. El hardware, pudiéndose utilizar con propio Blynk Cloud o crear el servidor local privado Blynk.

Aspectos importantes en la elección y ejecución en la plataforma:

- Blynk es gratuito para un uso básico y no comercial. El uso comercial o funciones más avanzadas como crear una app independiente a partir de un proyecto están disponibles sólo en planes de pago.
- API y UI similares para todos los dispositivos y hardware compatibles
- Conexión a la nube mediante: Wifi, Bluetooth y BLE, Ethernet, USB (serie), GSM...
- Conjunto de widgets fáciles de usar.
- Manipulación directa de pines sin escritura de código.
- Fácil de integrar y agregar nuevas funcionalidades usando pines virtuales.
- Envío de correos electrónicos, tweets, notificaciones push, etc.
- Monitoreo de datos históricos a través del widget SuperChart.
- Comunicación de dispositivo a dispositivo mediante Bridge Widget.
- APP en constante movimiento, continuamente se añaden nuevas funciones.

5 Desarrollo

Comenzar a diseñar aplicaciones para la Internet de las cosas (IoT) puede ser difícil debido a la multitud de opciones y posibilidades, tanto a nivel de hardware como de software, así como en lo que respecta a protocolos de comunicación y aplicaciones.

5.1 Esquema de diseño

Antes de entrar en el desarrollo detallado de la aplicación de Blynk y en el código del sistema de riego, analizamos el diagrama de flujo del programa para comprender mejor su funcionamiento.

En el diagrama se pueden observar dos zonas, correspondiente a la función *setup()*, donde se realiza la iniciación de las funciones necesarias y las configuraciones iniciales oportunas. Por otra parte, la función *loop()*, la parte del programa que se itera permanentemente. El bucle debe de estar lo más limpio posible, trasladándose todas las demás rutinas a temporizadores y funciones independientes. Por eso tan solo debe aparecer: *Blynk.run()*, comando esencial que mantiene la librería en un segundo plano, siendo necesario para administrar el envío de datos, conexiones de Blynk, y en segunda posición el *timer.run(),* necesario para la ejecución en unos determinados intervalos de tiempos de las funciones nombradas en el *setup()*.

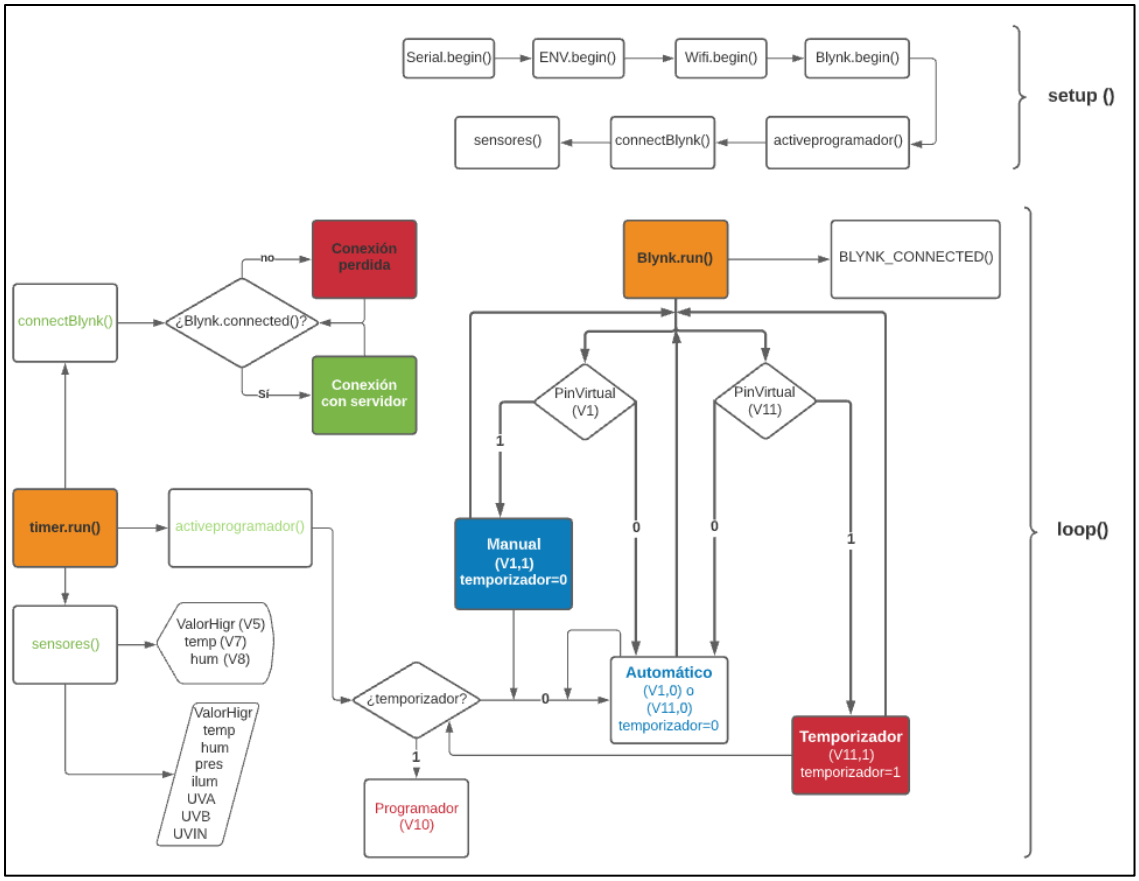

Figura 13. Diagrama de flujo

En cualquier momento el usuario puede realizar un cambio, modificación o habilitar el modo de riego, a través de la activación del pin virtual correspondiente.

En la primera de las imágenes (Figura 15), la aplicación se encuentra en modo de configuración de los diferentes elementos que la complementan. En ella, se puede observar en número del pin virtual al que está conectado el widget perteneciente.
En la segunda, (Figura 15), la aplicación se encuentra en funcionamiento, en comunicación con el servidor de Blynk y a su vez a la placa.

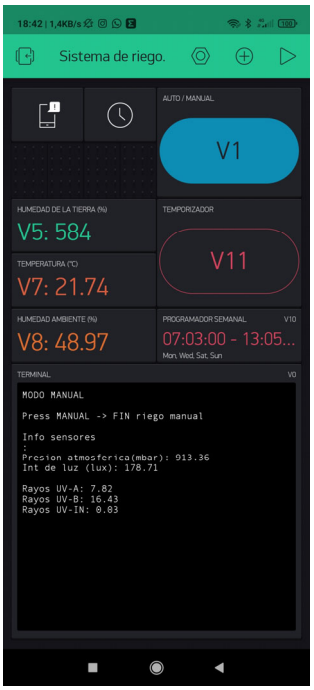

Figura 14. Asignación de pines virtuales

Figura 15. Aplicación en funcionamiento

5.2 Aplicación con Blynk

En el desarrollo de aplicación para gestionar el riego y control automatizado, está Blynk, que nos facilita enormemente la tarea y nos permite introducirnos en esta tecnología de una forma simple y sencilla.

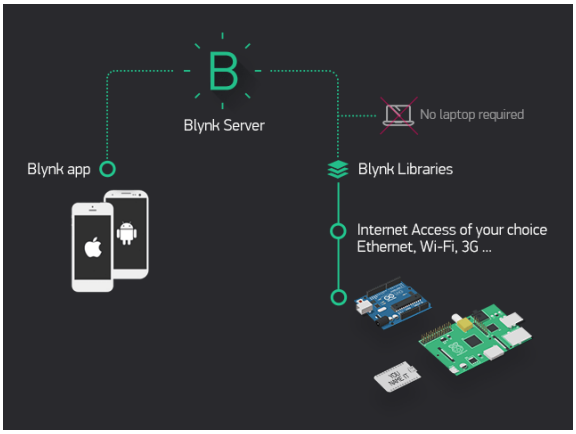

Figura 16. Esquema de funcionamiento. Fuente: Blynk

En resumen. Blynk es una plataforma que provee soluciones para el desarrollo de aplicaciones de IoT. Su funcionamiento se basa en una app, que puede utilizarse tanto en teléfonos Android como iOS, que se comunica con el hardware a través de los servidores de Blynk. La app permite diseñar un “panel de control” con una amplia

variedad de elementos que posibilitan enviar y recibir información hacia y desde el hardware, el que debe incluir alguna de las numerosas librerías especiales provistas por Blynk, disponibles para múltiples dispositivos tales como Arduino, Raspberry o placas basadas en ESP8266 y ESP32. Estas librerías resuelven la mayor parte de los detalles de la comunicación con la app, permitiendo que el desarrollador se enfoque en los detalles de su proyecto.

Para crear una aplicación IoT con Blynk es simple.

En el proyecto, aparecen tres estados claramente diferenciados: el riego programado por días de la semana y hora, el riego en modo automático y el riego manual.

- El riego programado, permite controlar la apertura y cierre de la electroválvula de manera automática fijando el o los días de la semana y el periodo de riego a considerar, mediante hora de inicio y hora de finalización del riego. Dicho riego puede estar fraccionado hasta en segundos de riego si se desease.
- El riego automatizado unido a la lectura de la variable procedente del higrómetro, hace de consigna para el riego se haga efectivo.
- El riego de modo manual. Este modo sujeto a la única condición de la apertura y cierre del relé consecuente del accionamiento de la electroválvula.

Los tres estados son controlados y monitoreados en el teléfono a través de Internet. Esto podría servirnos para controlar el riego y saber las condiciones meteorológicas, mientras estamos fuera, en el trabajo o en cualquier otra ubicación.

El circuito para realizar toda esta gestión, contiene la placa Arduino MKR Wifi 1010, que contiene un ESP8266 que nos provee de conectividad WiFi, el escudo de expansión MKR ENV, el higrómetro YL-69 y un módulo de relé para poder controlar la electroválvula que funciona con 12 Voltios.

Paso 1: Instalar la app Blynk.

Para ello debemos ir a la tienda de apps correspondiente a nuestro teléfono (Android o iOS), descargar e instalar la app.

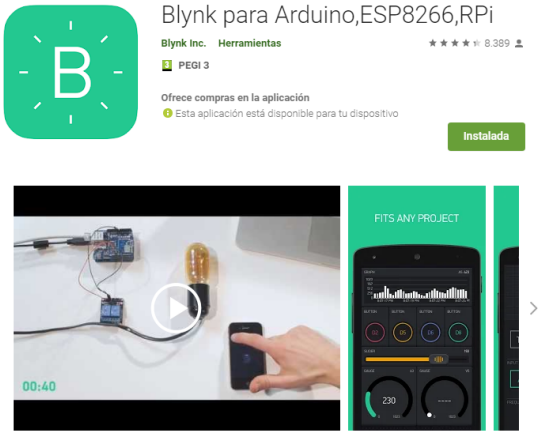

Figura 17. Blynk en Play Store

Paso 2: Crear la cuenta en Blynk.

Una vez descargada e instalada la app, la abrimos y creamos una cuenta, con una dirección de email a la que podamos acceder y una contraseña (Figura 18).

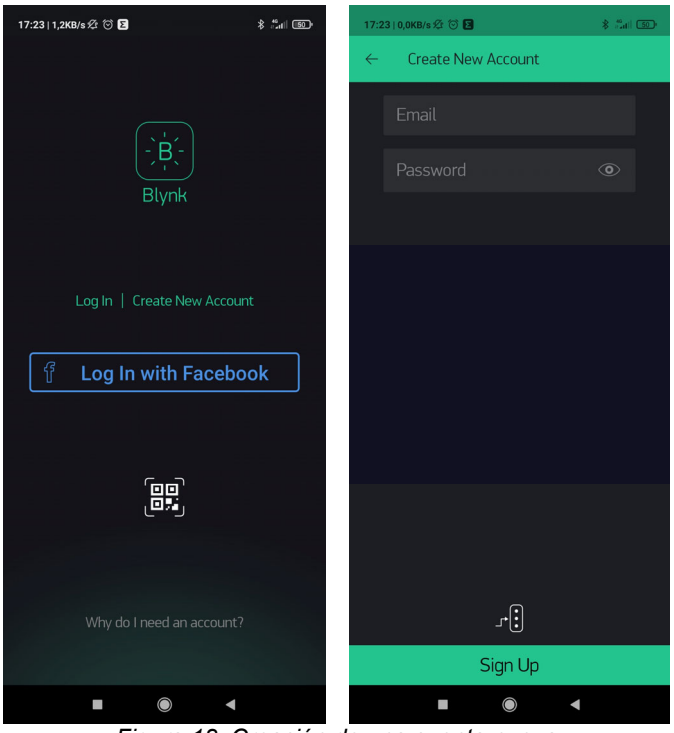

Figura 18. Creación de una cuenta nueva

Paso 3: Crear el proyecto.

Vamos ahora a crear un proyecto que contenga los elementos necesarios para controlar nuestro hardware. Debemos darle un nombre, “Sistema de Riego” y configurarlo según los detalles del hardware que utilicemos. En este caso en particular usamos la placa Arduino MKR Wifi 1010, en Blynk hay que seleccionar la Arduino MKR1000 de similares funciones y características, al tratarse de la que se conecta a WiFi, así que definimos esos valores y pulsamos “Create” (Figura 19).

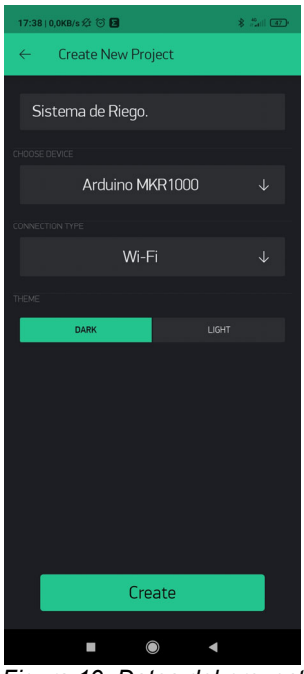

Figura 19. Datos del proyecto

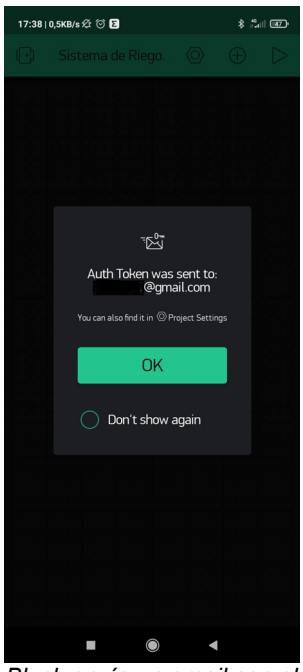

Figura 20. Blynk envía un email con el token.

Una vez hecho esto, Blynk nos enviará un mail a la cuenta ingresada al principio con un "token", una clave que vinculará la app con el hardware y que deberemos copiar luego en nuestro programa (Figura 20).

Paso 4: Diseñar la interfaz.

Debemos ahora construir la interfaz de la app, agregando cada uno de los widgets necesarios para realizar el control y riego del sistema automatizado.
Pulsando el símbolo "+" podremos ver todos los elementos que podemos utilizar. Algo importante es que cada uno tiene un costo de "energía". Al comienzo tenemos 2000 unidades que se van descontando a medida que agregamos elementos. Si borramos un elemento recuperaremos la energía correspondiente. En general el crédito inicial alcanza para hacer aplicaciones medianamente complejas, de todos modos, podemos comprar energía adicional si fuera necesario (Figura 21).

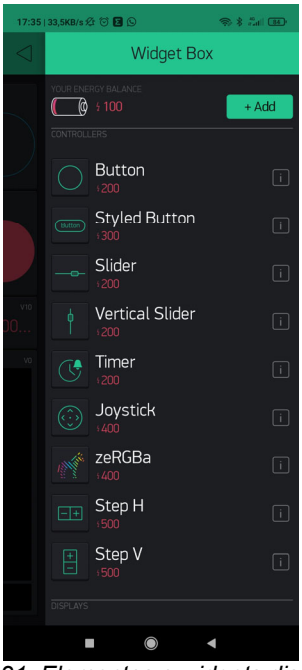

Figura 21. Elementos o widgets disponible

Los widgets son módulos de interfaz. Cada uno de ellos realiza una función de entrada / salida específica cuando se comunica con el hardware. Cada widget tiene su propia configuración. Algunos de los widgets (por ejemplo, Bridge) solo habilitan la funcionalidad y no tienen ninguna configuración.

Hay 4 tipos de widgets:

1. Controladores: se utilizan para enviar comandos que controlan su hardware como pueden ser el "Button", "Joystick", "Step H", …
2. Pantallas: se utilizan para la visualización de datos de sensores y otras fuentes. Por ejemplo, el comando "Terminal", "Video Streaming", …
3. Notificaciones: envíe mensajes y notificaciones; "Email", "Twitter" y "Notification".
4. Interfaces: widgets para realizar ciertas funciones de GUI (Graphical User Interface); "Text input", "Menu", "Time Input", …
5. Otros: widgets que no pertenecen a ninguna categoría; "Real-time clock", "Reports", …

La configuración común de widgets, se realiza mediante el “Selector de pines”. La selección del pin es uno de los principales parámetros que debe configurar. Define desde qué pin controlar o leer. En los que se encuentran estas diferentes opciones:

1. Pines digitales: representan pines de E/S digitales físicos en su hardware.
2. Pines analógicos: representan pines de E/S analógicos físicos en su hardware. Los pines habilitados para PWM están marcados con el ~ símbolo.
3. Pines virtuales: es un concepto inventado por Blynk Inc. para proporcionar intercambio de cualquier dato entre el hardware y la aplicación móvil Blynk. No tienen representación física. Blynk puede controlar los pines de E/S digitales y analógicas en su hardware directamente. Ni siquiera necesita escribir código para ello.

Siguiendo con el proyecto, para el diseño de la aplicación del “Sistema de Riego” son necesarios los siguientes widgets para la creación de la misma. Necesitaremos agregar y configurar para el correcto funcionamiento:

- El widget de notificaciones, “Notification”. Elemento de notificaciones push permite enviar mensajes desde su hardware a su dispositivo (Figura 22).

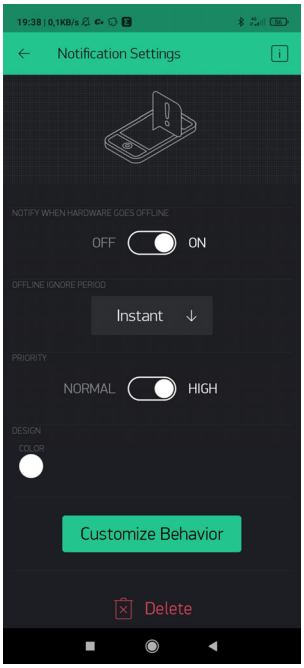

Figura 22. Propiedades de las notificaciones push

Actualmente también contiene 3 opciones adicionales:

- ✓ Notifica cuando el hardware esté fuera de línea, tanto cuando pierda la conexión Wifi como la conexión eléctrica del hardware. Se recibe una notificación automática en caso de que su hardware se desconecte.
- ✓ Período de ignorar sin conexión. Se trata de definir cuánto tiempo el hardware puede estar fuera de línea (después de estar fuera de línea) antes de enviar la notificación. En caso de que se exceda el período, se enviará una notificación de "hardware fuera de línea". No recibirá ninguna notificación en caso de que el hardware se vuelva a conectar dentro del período especificado.
- ✓ Prioridad alta, brinda más posibilidades de que su mensaje se entregue sin demoras.

Por otro lado, posee algunas limitaciones que no afectan en gran medida en el funcionamiento de la aplicación para el riego, como son, la longitud máxima permitida de la notificación push no puede exceder los 120 caracteres.
Otra de las limitaciones, cada dispositivo solo puede enviar una notificación push cada cinco segundos.

- El widget, “Real-time clock”. Widget encargado de la función del reloj en tiempo real que permite obtener la hora del servidor para obtener el tiempo en el hardware utilizado. Tan solo necesita la preseleccionar la zona horaria en la interfaz. No se requiere nada más que arrastrar y soltar el widget RTC en la pantalla de diseño (Figura 23).

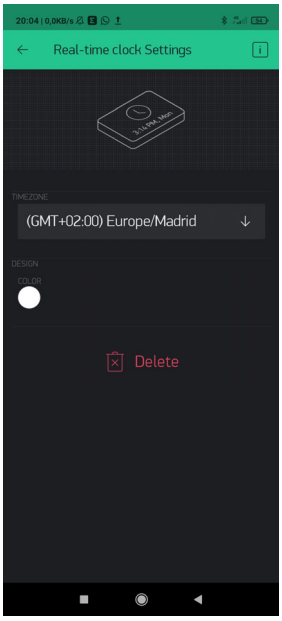

Figura 23. Configuración reloj

- Widgets tipo de pantalla, “Value Display”. Herramienta usada para tres sensores para la visualización del valor. Llamados en el diseño de la aplicación “Humedad de la tierra”, “Temperatura” y “Humedad Ambiente” (Figura 24).

Hay dos formas de enviar datos desde su hardware a los widgets de la aplicación: mediante los pines virtuales usando la frecuencia de lectura incorporada de Blynk mientras la aplicación está activa configurando el parámetro 'Frecuencia de lectura' en el intervalo de 5 segundos; seleccionando el modo PUSH que permitiría obtener o mandar los datos desde el hardware.
Los sensores de humedad de la tierra, humedad ambiente y temperatura tienen seleccionado una salida mediante un pin virtual en la configuración del widget correspondiente.

En los sensores, debemos asignar el rango específico de lectura de cada uno de ellos. Conseguimos mediante el diseño adecuado (opción de texto), el color del dato en el widget correspondiente al valor del sensor sea representativo del nivel alto o bajo de esa variable. Dicho de otra manera, cuando el valor de la humedad del suelo sea alto, la visualización del valor sea en rojo. De la misma manera, si el valor de la temperatura es alto.

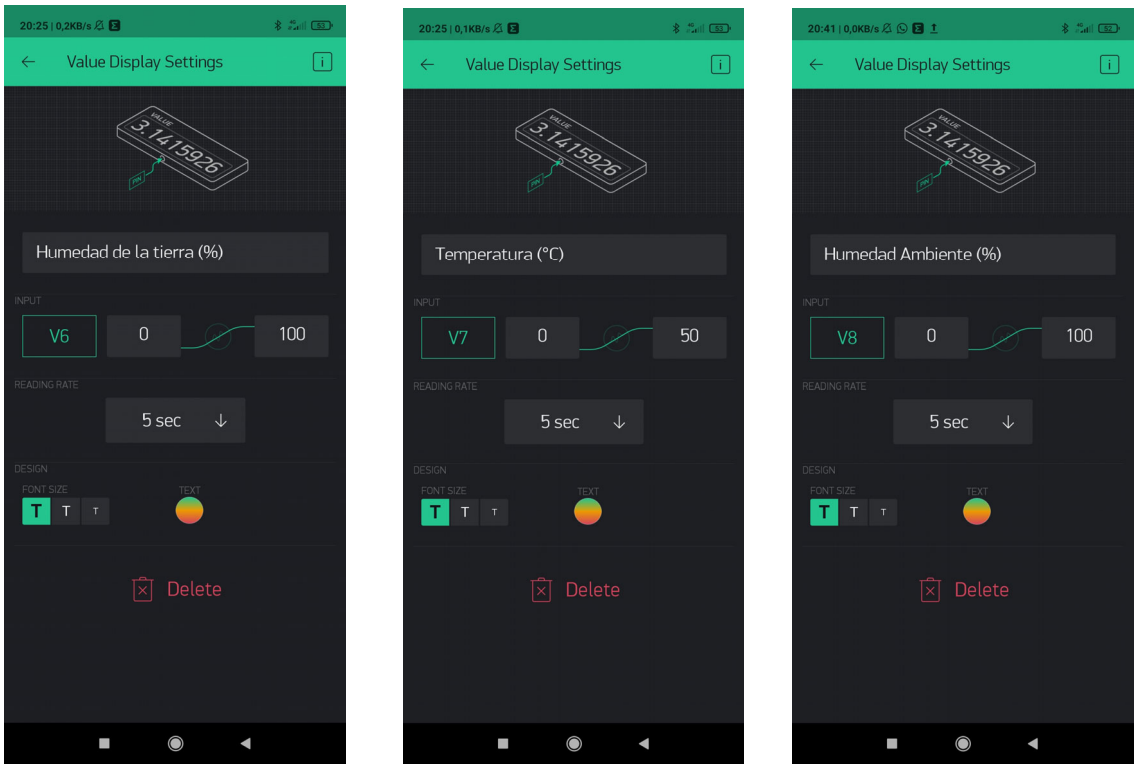

Figura 24. Configuraciones monitorización de valores de temperatura, humedad de tierra y ambiente

- El widget de “Button”, del tipo controladores. Permite enviar valores ON y OFF (BAJO / ALTO). El botón envía 1 (ALTO) al presionar y envía 0 (BAJO) al soltarlo.

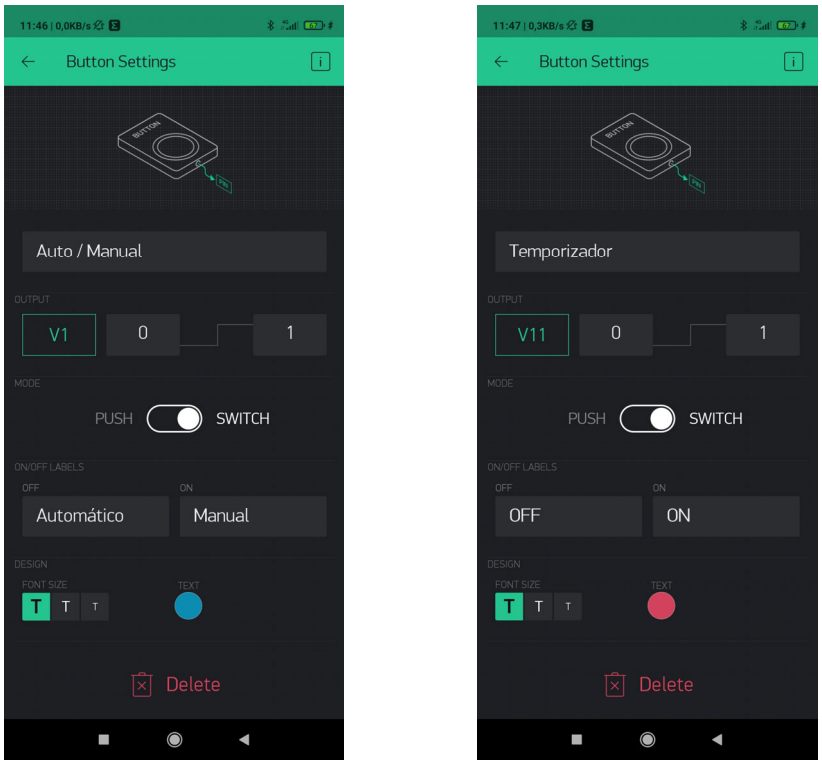

Figura 25. Configuraciones para los botones de "Auto/Manual" y "Temporizador"

En el proyecto, el widget es utilizado en dos ocasiones.
Botón llamado “Auto/Manual”, dependiendo el nivel bajo o alto, actuará en modo automático o modo manual correspondientemente. El primero de ellos realiza la función del riego de manera manual (nivel ALTO) o, por lo contrario (nivel BAJO), el riego de manera automática, preformando un nivel mínimo de humedad para el riego.
Por otro lado, el botón “Temporizador”, simplemente dispone de la conmutación encendido y apagado. La selección ON tiene la capacidad de activar el riego de manera programada. En el caso contrario, deshabilita el temporizador y pasa al modo automático.

Al tocar el botón podemos acceder a sus propiedades: en primer lugar, el nombre indica la función del botón correspondiente; la siguiente función, los dos modos se gestionan a través de la salida del pin virtual y los valores posibles (Output), en nuestro caso 0 y 1; el modo de funcionamiento es mediante conmutador (Mode); las ultimas propiedades pertenecen a las etiquetas que muestra el botón (On/Off labels) y el tamaño y color del texto (Figura 25).

- El widget de tipo interfaz, “Time Input”, es la herramienta de entrada de tiempo que permite seleccionar la hora de inicio / finalización, el día de la semana, la zona horaria, los valores prefijados de amanecer / atardecer y enviarlos al hardware.

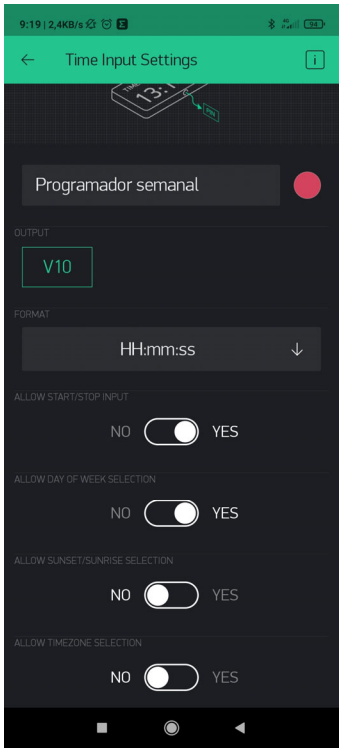

Figura 26. Configuración del programador

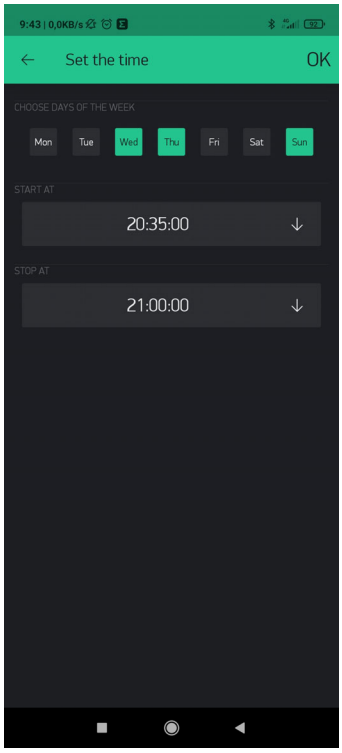

Figura 27. Interfaz programador

En proyecto, el widget es llamado PROGRAMADOR SEMANAL. El formato elegido es HH:MM:SS, el cual posibilita realizar una programación del riego en segundos a través de la aplicación. El widget será controlado mediante el pin virtual 10, el programador se configura con la función de hora de inicio y hora de finalización, y eligiendo el día de la semana. Marcando la función “YES” en la aplicación. Las otras dos funciones, no han sido configuradas para el desarrollo de la aplicación controladora (Figura 26).

La función del widget de entrada de tiempo es enviar los parámetros asociados del tiempo de inicio y finalización en segundos y los días de la semana asignados al pin virtual 10, en nuestro caso (Figura 27).

Tiene diferentes aspectos importantes, los cuales hay que tener en cuenta a la hora de programar el temporizador y el programador semanal.

- ✓ El hardware debe establecer el horario con la interfaz de la aplicación en segundos del día. Eso quiere decir, las horas y los minutos deben ser pasados a segundos, (3600*horas + 60*minutos + segundos).
- ✓ Los índices de los siete días de la semana, tienen un ciclo que comienza en lunes y termina en domingo. (1- lunes, 2- martes, 7- domingo)

  - ➢ El widget de “Terminal”, es la herramienta semejante al monitor serie del ordenador, donde visualiza los mensajes programados en el código del hardware. Este permite enviar la información e interacción entre la aplicación de Blynk y el hardware conectado (Figura 28).

    Aspectos importantes que el widget terminal nos blinda y serán de gran uso:
    - ✓ Siempre almacena los últimos 25 mensajes que el hardware mando a la nube de Blynk.
    - ✓ Posee comandos característicos para visualización de la información en la pantalla correspondiente.

```
terminal.print();   // Imprime valores, igual que Serial.print()
terminal.println(); // Imprime valores, igual que Serial.println()
terminal.write();   // Escribe datos en el buffer
terminal.flush();   // Envio de los datos.
terminal.clear();   // Borra todos las líneas del terminal.
```

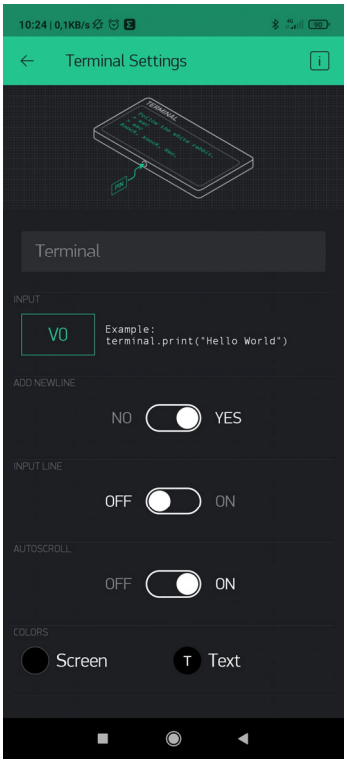

Figura 28. Configuración del terminal de información

En la configuración del widget de terminal de control, visualización e información se realiza mediante la salida del pin virtual número cero (V0). Deshabilitamos la opción de la línea de entrada en el terminal, en el proyecto no necesitamos interactuar con hardware.

Con esto quedaría terminado el proyecto del lado de la app.

5.3 Software con interfaz de Arduino.

Paso 1: Instalar el software de programación Arduino (IDE)

El otro eslabón fundamental de esta cadena es el programa utilizado para la creación del código. El software Arduino (IDE), se trata del entorno de desarrollo integrado común a todas las placas y que se ejecuta tanto en línea como fuera de línea.
Al tratarse de un software libre y código abierto, mediante la descarga de la IDE Arduino fuera de línea en el sistema operativo “Windows 10” desde la página oficial (Arduino.cc/en/software). Se debe instalar el software Arduino Destop IDE y agregarle Arduino SAMD Core.
La instalación se realiza de manera intuitiva y es de fácil seguimiento hasta disponer de la herramienta de escritura del código y la carga en la placa (Figura 29).

Figura 29. Software de programación Arduino IDE

Paso 2. Instalar el controlador para la placa Arduino MKR Wifi 1010

Este sencillo procedimiento se realiza seleccionando el menú *Herramientas*, luego *Placa* y el último *Gestor de placas*, como se documenta en la página Arduino *Boards Manager* . Finalmente, en la línea del buscador, escribir *SAMD MKR Wifi 1010* (Figura 30).

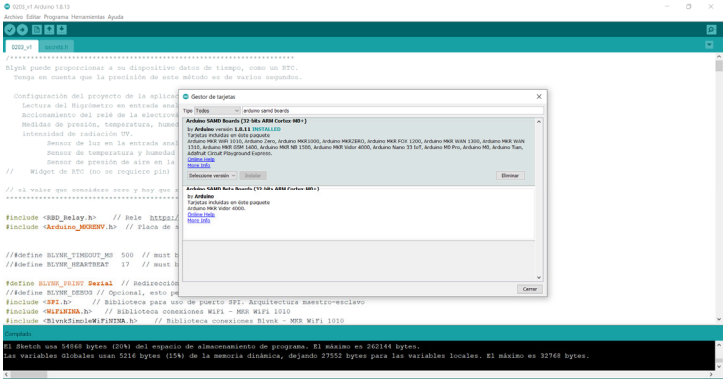

Figura 30. Control SAMD Core para placa MKR 1010 Wifi instalado.

5.4 Código

El último paso sería escribir el código en la placa Arduino, el detalle de la codificación se explica en los siguientes apartados.

5.4.1 Librerías

Comenzamos con la inclusión de las librerías imprescindibles para el correcto desarrollo del código:

```
//LIBRERIAS
#include <Arduino_MKRENV.h>  // Placa de sensores temperatura, humedad...
#include <SPI.h>     // Biblioteca para uso de puerto SPI. Arquitectura maestro-esclavo
#include <WiFiNINA.h>   // Biblioteca conexiones WiFi - MKR WiFi 1010
#include <BlynkSimpleWiFiNINA.h>    // Biblioteca conexiones Blynk - MKR WiFi 1010
#include <WidgetRTC.h>    // Biblioteca de Widget RTC en Blynk
#include "secrets.h"    // Biblioteca credenciales del WiFi
#include <TimeLib.h>    // Bilbioteca para gestion de tiempos https://github.com/PaulStoffregen/Time
```

Figura 31. Librerías instaladas

- Placa de sensores del escudo MKR. Se ha utilizado la librería *Arduino_MKRENV.h* versión 1.2.0, desarrollada por Arduino. La cual, se encarga de los cálculos necesarios. Mediante la sintaxis adecuada pemite devolver los valores en las unidades solicitadas de temperatura, humedad, presión, luz y UV. (https://www.arduino.cc/en/Reference/Arduino_MKRENV)

- Comunicación con dispositivos mediante SPI. *SPI.h,* (https://www.arduino.cc/en/reference/SPI). Esta librería es la gestora de la comunicación con otros dispositivos, siendo Arduino, el dispositivo maestro. Se utiliza para inicializar el bus SPI con los diferentes esclavos como es el escudo de sensores, la aplicación Blynk, el reloj…
  Conexión Wifi a la red doméstica, mediante la librería *WifiNINA.h,* version 1.8.10, desarrollada por Arduino (https://www.arduino.cc/en/Reference/WiFiNINA). Esta permite la utilización de las capacidades de la placa MKR Wifi 1010, tanto como, conexión entrante como servidor o como cliente en conexiones salientes. En nuestro caso, utilizamos como servidor lo que permite configuración a la red doméstica para el control del sistema de riego.

- Conexión de la placa con la red doméstica Wifi, se utiliza *BlynkSimpleWiFiNINA.h,* versión 1.0.0, creada por Blynk, *(https://github.com/blynkkk/blynk-library/blob/master/src/BlynkSimpleWiFiNINA.h)* Función indispensable para la configuración, conexión o reconexión automática a los puntos de acceso Wifi en el tiempo de ejecución en las placas que ejecuten escudos WifiNINA.

- Sincronización de la hora mediante el servidor de Blynk con nuestra placa, se realiza mediante la libreria *WidgetRTC.h.* Librería creada por Blynk en marzo de 2016. (https://github.com/blynkkk/blynk-library/blob/master/src/WidgetRTC.h). Asociada a esta librería, también, la librería *TimeLib.h,* versión 1.6.0 *(https://github.com/PaulStoffregen/Time)* Creada por "*Michael Margolis"* con la finalidad principal de habilitar la funcionalidad de cronometraje para Arduino. Funciones de fecha y hora con variedad de sintaxis, con disposiciones para sincronizar con fuentes externas.

- Credenciales de la conexión de Wifi y autentificación para la conexión al servidor de Blynk, *secrets.h*. Librería de creación propia para guardar y se define, el nombre y la contraseña de las distintas disposiciones de conexión a la red Wifi en las que he realizado las diferentes pruebas de la aplicación de riego. Además, se encuentra la

contraseña para el acceso al servidor de Blynk para el control de la aplicación (Figura 32).

Figura 32. Credenciales de conexión a Wifi y Blynk

5.4.2 Declaraciones de las variables globales

A continuación, la declaración de todas las variables necesarias para la ejecución del proyecto.

```
//RELE
int PinRele = 5;          // pin digital 5
```

Figura 33. Variable del relé

La variable con la que se nombra el uso del relé, el cual se ha conectado al pin digital número 5 del escudo MKR, nombrado a partir de ahora en el proyecto como *PinRele* (Figura 33).

```
//HIGROMETRO
int PinHumedad = A1; // dato análogico
int ValorHigr;     // Lectura del sensor
int SensHigr = 700;    // Valor de ajuste del umbral, para la sensibilidad del sensor
```

Figura 34. Variables para el higrómetro

En cuanto a las variables del sensor de humedad de la tierra, se conecta a través del pin analógico número 1, llamado en el proyecto *PinHumedad*. La variable con nombre *ValorHigr,* utilizada para la lectura del valor de sensor y, *SensHigr,* es el valor que hace de umbral para diferenciar entre la tierra este húmeda o seca. Esta última variable, el valor elegido es de 700 puesto que el sensor de humedad de tierra posee lecturas entre 0 y 1023. (Figura 34).

```
//VARIABLES CONTROL APP
int temporizador;   //Valor para la iniciación del temporizador. (1=ON. 0=OFF)
```

Figura 35. Variables para el control de la aplicación móvil

La variable *temporizador* declarada para la entrada en funcionamiento del modo **Temporizador**. Cuando el valor es igual a 1, habilita el envío de los parámetros característicos de entrada de tiempos hacia el hardware, para comenzar el riego programado. Si el valor de la variable es 0, solo puede estar en modo **Manual** o **Automático** (Figura 35).

```
//VARIABLES HORA Y DIA
long startseconds;    //Hora de inicio del TimeInput en segundos
long stopseconds;   // Hora de fin del TimeInput en segundos
long timeseconds;   // Hora en segundos
char Date[16];    // Caracteres para la fecha
char Time[16];    // Caracteres para la hora
```

Figura 36. Variables para el formato día y hora

Para las variables pertenecientes a los parámetros de la entrada de tiempos del widget son las siguientes: la variable *startseconds*, guarda la hora de inicio del programador a la cual se inicia el riego; la variable *stopseconds*, guarda la hora de fin del programador; la variable *timeseconds*, guarda la hora actualizada.
Las variables *Date* y *Time*, encargados de mostrar la cadena de caracteres correspondientes a la fecha y la hora. (Figura 37)

```
//SENSORES
float temp;    //Valor temperatura
float hum;     //Valor humedad
float pres;    //Valor presión
float ilum;    //Valor iluminancia
float UVA;     //Valor UVA
float UVB;     //Valor UVB
float UVIN;    //Valor UVIN
```

Figura 37. Variables para sensores de escudo MKR

La declaración de las variables, tipo real, que gestionaran los valores de los sensores del escudo MKR, son los siguientes: *temp, hum, pres, ilum, UVA, UVB, UVIN*. Pertenecientes a los valores de la temperatura, humedad, presión, iluminancia, rayos UVA, rayos UVB y el índice UV, respectivamente (Figura 37).

```
//CREDENCIALES PARA CONEXION WIFI
char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;
int status = WL_IDLE_STATUS;  //estado temporal de Wifi es desconectado
bool isFirstConnect = true;   // Primera conexión
```

Figura 38. Variables para conexión Wifi

Los datos necesarios para la conexión a la red Wifi. Para guardar las cadenas de caracteres con el nombre de la red Wifi y la contraseña, mediante *ssd* y *pass,* estando disponibles en la librería *secrets.h.*
Inicialmente, se define *status* como un estado temporal de la conexión Wifi, y donde la *isFirstConnect* se considera verdadera o correcta (Figura 38).
Las credenciales para la conexión al servidor de Blynk son guardados de la misma manera que el nombre y la contraseña de la red Wifi (Figura 39).

```
//CREDENCIAL PARA CONEXION A Blynk App.
char auth[] = AuthToken;
```

Figura 39. Variable para conexión al servidor de Blynk

```
//BLYNK
BlynkTimer timer;   // BlynkTimer proporciona la funcionalidad SimpleTimer

//configuracion de los widget usados en Blynk en el movil
WidgetRTC rtc;    //  Reloj para Blynk
WidgetTerminal terminal(V0);    // Terminal en Blynk == monitor serie en PC
```

Figura 40. Configuración de los widgets

La declaración de un nuevo objeto timer, *BlynkTimer,* incluida en librería de Blynk. Para el envío de datos periódicamente determinados en intervalos, manteniendo la función *loop()* lo más limpia posible.
Se define el *rtc* para la sincronización del reloj en tiempo real, obteniendo la hora del servidor. Es necesario el *WidgetRTC,* llamado (*Real-time clock),* de la aplicación de Blynk (Figura 23). Dentro de la librería TimeLib.h, la función time_t solicita al WidgetRTC la sincronización de tiempo, que anteriormente la WidgetRTC ha realizado la actualización de la hora y día desde el servidor.
Por último, el *WidgetTerminal* permite la conexión del terminal serie virtual al pin virtual (V0), (Figura 28).

5.4.3 Pines virtuales

Los pines virtuales son necesarios para intercambiar las acciones de los botones de los diferentes modos entre la aplicación y el código generado.

Botón modo Automático o Manual. Pin virtual (V1).

El pin virtual (V1), seleccionado para el control de los modos de funcionamiento del riego en modo **Manual** o, por lo contrario, en modo **Automático**.

Modo **Manual**, el usuario es el gestor de la puesta en marcha del riego y de la finalización del mismo. Esta condición de riego está en funcionamiento mientras el valor del pin virtual (V1) sea igual a 1, en caso contrario estará en modo **Automático**.
El primero de los modos, además, parametriza los siguientes sensores del escudo MKR ENV como son la presión atmosférica, la iluminancia, rayos UVA o el índice de UV.
Estos tan solo serán visibles mientras se realiza el riego manual. Dichos valores, son enviados y visualizados en el terminal virtual serie (Figura 41).

```
//BOTON BLYNK V1 --> MANUAL = 1 // AUTOMATICO = 0
BLYNK_WRITE(V1){
  if(param.asInt() == 1){     //esta en modo manual
    terminal.clear();
    terminal.println("--------------");
    terminal.println("MODO MANUAL");
    terminal.println();
    terminal.println("Press MANUAL -> FIN riego manual");
    terminal.println();
    terminal.println("Info sensores: ");
    terminal.print("Presion atmosferica(mbar): ");
    terminal.println(pres);
    terminal.print("Int de lus (lux): ");
    terminal.println(ilum);
    terminal.print("Rayos UV-A: ");
    terminal.println(UVA);
    terminal.print("Rayos UV-B: ");
    terminal.println(UVB);
    terminal.print("Rayos UV-IN: ");
    terminal.println(UVIN);
    terminal.flush();

    if(manual == 0){      //está en modo automático
      temporizador=0;
      Blynk.virtualWrite(V1, 1);  // Widget ManuaL --> ON
      Blynk.virtualWrite(V11, 0);  // Widget Temp --> OFF
      digitalWrite(PinRele, HIGH); // RELE ON
    }
    else{    //está en modo manual
      temporizador=0;
      Blynk.virtualWrite(V1, 1);  // selector ON manual Widget
      Blynk.virtualWrite(V11, 0);  // selector tiempos OFF
      digitalWrite(PinRele, HIGH); // RELE ON
    }
  }
  else{      //boton manual = 0
    temporizador = 0;
    terminal.clear();
    terminal.println("--------------");
    terminal.print("MODO AUTOMATICO");
    terminal.println();
    terminal.println();
    terminal.println();
    terminal.flush();
    Blynk.virtualWrite(V1, 0);
    Blynk.virtualWrite(V11, 0);
```

Figura 41. Modo Automático o Manual

```
    // riego automatico predefinido en código
    if(ValorHigr >= SensHigr && digitalRead(PinRele) == LOW){  //con notificacion push
      terminal.println();
      terminal.println("La tierra está seca");
      terminal.println();
      terminal.println("RIEGO AUTOMATICO ON");
      terminal.flush();
      digitalWrite(PinRele, HIGH);
      Blynk.notify("RIEGO AUTOMÁTICO ON");
    }
    else if(ValorHigr >= SensHigr && digitalRead(PinRele) == HIGH){  //sin notificacion push
      terminal.println();
      terminal.println("La tierra está seca");
      terminal.println();
      terminal.println("RIEGO AUTOMATICO ON");
      terminal.flush();
      digitalWrite(PinRele, HIGH);  // set RELE ON
    }

    if(ValorHigr <= SensHigr && digitalRead(PinRele) == HIGH){
      terminal.println();
      terminal.println("La tierra está húmeda");
      terminal.println();
      terminal.println("RIEGO AUTOMATICO OFF");
      terminal.flush();
      digitalWrite(PinRele, LOW); // set RELE OFF
      Blynk.notify("Riego automático OFF");
    }
    else if(ValorHigr <= SensHigr && digitalRead(PinRele) == LOW){ //sin notificacion push
      terminal.println();
      terminal.println("La tierra está húmeda");
      terminal.println();
      terminal.println("RIEGO AUTOMATICO OFF");
      terminal.flush();
      digitalWrite(PinRele, LOW); // set RELE OFF
    }
  }
}
```

Figura 42. Pin virtual (V1). Umbral para riego automático

El modo automático tiene la facultad de riego predefinido por codificación por lo que cuando la humedad de la tierra proporcionada por el higrómetro, desciende de un cierto valor, o sobrepasa un cierto umbral, el riego se comporta de manera autónoma. El riego automatizado conlleva cambios en el estado del riego dependiendo de la humedad de la tierra. De manera instantánea, el sistema manda una notificación al teléfono móvil, sin tener abierta la propia aplicación de *"Sistema de riego"* (Figura 42)

Botón modo Temporizador. Pin virtual (V11).

Botón para la gestión del modo **Temporizador**, definido en el pin virtual (V11), es el encargado de la activación de la entrada de tiempo.
La activación de este modo, solo se realiza cuando la selección del botón es igual a 1. Produciendo que la variable temporizador cambie a nivel alto, ocasionando la puesta en funcionamiento del widget para seleccionar el inicio y finalización de la hora, y el día de la semana. (Figura 43).

```
//BOTON BLYNK V11 --> INICIA MODO TEMPORIZADOR
BLYNK_WRITE(V11){
  if(param.asInt() == 1){
    Blynk.virtualWrite(V1, 0);
    Blynk.virtualWrite(V11, 1);
    temporizador = 1;
    terminal.clear();
  }
  else{
    temporizador = 0;
  }
}
```

Figura 43. Botón del Temporizador

Entrada de tiempo. Pin virtual (V10).

El widget de entrada de tiempo permite seleccionar la hora de inicio y finalización, y el día de la semana, la salida elegida es el pin virtual (V10). Este pin virtual se sincroniza cuando el modo **Temporizador** está activo (V11), donde produce el cambio de la

variable *temporizador* sea igual a 1 (Figura 43) y, por lo tanto, se inicia la sincronización de los parámetros para gestionar el día de la semana y la hora.

Se definen las variables de la entrada de tiempo con la función *TimeInputParam t(param).* La fecha se codifica mediante una cadena de caracteres ligada a la variable *Date* para el número del día, mes y año. La hora, minutos y segundo se generan de la misma manera asociada a la variable de tipo carácter *Time.* Ambas, visualizadas a través del terminal de la aplicación (Figura 44)

```
//WIDGET DE ENTRADA DE TIEMPO
BLYNK_WRITE(V10){
  if(temporizador == 1){
    TimeInputParam t(param);

    sprintf(Date, "%02d/%02d/%04d", day(), month(), year());
    sprintf(Time, "%02d:%02d:%02d", hour(), minute(), second());

    terminal.clear();
    terminal.println("MODO TEMPORIZADOR");
    terminal.println();
    terminal.print("Fecha: ");
    terminal.print(Date);
    terminal.print(" Hora: ");
    terminal.print(Time);
    terminal.flush();
```

Figura 44. Entrada de tiempo (V10). Fecha y hora

Siguiendo con la entrada de tiempo, es necesario realizar un ajuste con el número de día de la semana. Blynk tiene establecido el día 7 de la semana, el domingo. Pero para la librería utilizada en el proyecto, *TimeLib.h*, el primer día de la semana es el domingo.

```
  //Ajuste dias de la semana. TimeLib el dia 1 de la semana es Domingo y en Blynk es el dia 7
  int ajusteDia = -1;   //Blynk hay que restar un dia a los dias en TimeLib.
  if( weekday() == 1 ){    // Domingo es dia 1 y en Blynk es dia 7.
    ajusteDia = 6;
  }
  if(t.isWeekdaySelected( weekday() + ajusteDia)){   // si coincide con el dia de la seleccion
    if(t.hasStartTime()){ // Hora de inicio del riego
      terminal.println();
      terminal.println(String("RIEGO START: ") + t.getStartHour() + ":" + t.getStartMinute());
      terminal.flush();
    }
    if(t.hasStopTime()){ // Hora de fin del riego
      terminal.println(String("RIEGGO STOP: ") + t.getStopHour() + ":" + t.getStopMinute());
      terminal.flush();
    }
    terminal.println();
    terminal.println("TEMPORIZADOR ACTIVO EL: ");
    terminal.flush();

    for(int i = 1; i <= 7; i++){   // Dias de la semana
      if (t.isWeekdaySelected(i)){
        switch (i){
          case 1:terminal.print("- Lunes");break;
          case 2:terminal.print("- Martes");break;
          case 3:terminal.print("- Miércoles");break;
          case 4:terminal.print("- Jueves");break;
          case 5:terminal.print("- Viernes");break;
          case 6:terminal.print("- Sábado");break;
          case 7:terminal.print("- Domingo");break;
        }
        terminal.flush();
      }
    }
```

Figura 45. Entrada de tiempo (V10). Ajuste día

Por tanto, es necesario realizar una pequeña función para adecuar el funcionamiento del código.

Si el día seleccionado del programador y el día de la semana, coinciden, la aplicación informa de la hora de inicio y fin del riego, además de informar de los días que el riego está programado para ese intervalo horario (Figura 45).

```
timeseconds = ((hour() * 3600) + (minute() * 60) + second());   //  hora actual en segundos
startseconds = (t.getStartHour() * 3600) + (t.getStartMinute() * 60); // hora inicio del TimerInput en segundos
stopseconds = (t.getStopHour() * 3600) + (t.getStopMinute() * 60);    // hora fin del TimerInput

// error al introduccir el temporizador
if(stopseconds < startseconds){
  digitalWrite(PinRele, LOW); // set RELE OFF
  terminal.clear();
  terminal.println("ERROR EN EL HORARIO DEL TEMPORIZADOR");
  terminal.println();
  terminal.println("Hora STOP < Hora START");
  Blynk.notify("ERROR SELECCION HORARIO TEMPORIZADOR");
  terminal.flush();
 }
```

Figura 46. Entrada de tiempo (V10). Error en el temporizador

El programador de riego gestiona la hora de inicio y la finalización de riego. Las variables necesarias para el correcto funcionamiento son la hora, la hora de inicio y de finalización del riego, todas ellas en segundos del día. Las variables definidas son *timeseconds*, *startseconds* y *stopseconds,* respectivamente.

El caso de que al introducir la hora de finalización sea anterior a la hora de inicio del riego (*stopseconds* < *starseconds*), la aplicación muestra un mensaje de error en el terminal virtual, además, manda una notificación repetitiva de aviso de error. La notificación no deja de emerger hasta que se subsane el error (Figura 46).

El programa prosigue de manera exitosa cuando la hora de inicio y hora de finalización del riego son seleccionados correctamente, siendo la hora de inicio anterior a la hora de finalización (*stopseconds > starseconds)*.
En el desarrollo de la aplicación puede producirse dos situaciones:

- La hora en tiempo real, no esté englobada entre la hora de inicio y la hora de fin del riego. El terminal de la aplicación informa mediante un mensaje que el temporizador no se ha iniciado. (Figura 47)

```
else{    // ( stopseconds > startseconds )
 if(timeseconds < startseconds){
   terminal.println();
   terminal.println("Temporizador no iniciado.");
   terminal.print("Riego comienza a las: ");
   terminal.println(String(" ") + t.getStartHour() + ":" + t.getStartMinute());
   terminal.flush();
 }
```

Figura 47. Entrada de tiempo (V10). Temporizador no iniciado

- Producir la iniciación del temporizador, al encontrarse la hora en tiempo real comprendida entre la hora de inicio (*timeseconds > startseconds)* y la hora de finalización del riego (*timeseconds <= stopseconds)*. En ese caso, se efectúa el riego hasta la hora para finalizar el riego. El inicio y fin del riego es efectivo mediante el accionamiento de salida del relé. En consecuencia, el estado del relé indica, por las notificaciones correspondientes el aviso de riego temporizado. También, el estado del relé, se informa que el temporizador se activará el próximo día de riego (Figura 48).

```
      else{    // (timeseconds > startseconds)
       if(timeseconds <= stopseconds){
         if(digitalRead(PinRele) == LOW){
           digitalWrite(PinRele, HIGH); // set LED ON
           Blynk.notify("Riego temporizado ON");
           terminal.println("Riego ON");
           terminal.flush();
         }
         if(digitalRead(PinRele) == HIGH){
           digitalWrite(PinRele, HIGH); // set LED ON
           terminal.println("Riego ON");
           terminal.flush();
         }
        }
        else{    //  timeseconds >= stopseconds
         if(digitalRead(PinRele) == HIGH){
           digitalWrite(PinRele, LOW); // set LED OFF
           Blynk.notify("Riego temporizado OFF ");
           terminal.println();
           terminal.print("Riego finalizado ya a las:");
           terminal.println(String(" ") + t.getStopHour() + ":" + t.getStopMinute());
           terminal.println();
           terminal.flush();
         }
         if(digitalRead(PinRele) == LOW){
           digitalWrite(PinRele, LOW); // set LED OFF
           terminal.println();
           terminal.print("Riego finalizado ya a las:");
           terminal.println(String(" ") + t.getStopHour() + ":" + t.getStopMinute());
           terminal.println();
           terminal.println("Temporizador se activa el próximo dia seleccionado");
           terminal.flush();
         }
        }
       }
     }
   }
```

Figura 48. Entrada de tiempo (V10). Temporizador On/Off

Por el contrario, si el día seleccionado no coincide con el día de la semana (Figura 49), el terminal virtual de la aplicación de riego informa que el temporizador esta inactivo y no hay programador riego durante el día de hoy. Informando, también del mismo modo, los días que están programados hasta ese momento.

```
    else{
      terminal.println("HOY. TEMPORIZADOR INACTIVO");
      terminal.println();
      terminal.println("Riego programado para el: ");
      terminal.flush();

      for(int i = 1; i <= 7; i++){   // Dias de la semana
        if(t.isWeekdaySelected(i)){
          switch (i){
            case 1:terminal.print("- Lunes");break;
            case 2:terminal.print("- Martes");break;
            case 3:terminal.print("- Miércoles");break;
            case 4:terminal.print("- Jueves");break;
            case 5:terminal.print("- Viernes");break;
            case 6:terminal.print("- Sábado");break;
            case 7:terminal.print("- Domingo");break;
          }
          terminal.flush();
        }
      }
    }
    terminal.println();
  }
}
```

Figura 49. Entrada de tiempo (V10). Temporizador finalizado y próximo día de riego

5.4.4 Gestor de conexión al servidor

El siguiente comando gestiona la configuración para la conexión con el servidor de Blynk. Se trata de la función encargada de realizar la rutina de conexión entre el hardware y el servidor de Blynk. Esta función se ejecuta cada vez que se establezca la nueva conexión. La sincronización de esta función es necesaria siempre que la conexión se interrumpe debiéndose ejecutar para poder establecerla de nuevo.

```
//CONEXION A SERVIDOR BLYNK
BLYNK_CONNECTED(){
  if(isFirstConnect == true){
    rtc.begin();
    Blynk.syncAll();
    Blynk.notify("Conexión al servidor Blynk");
    isFirstConnect = false;
  }
  else{
    rtc.begin();
    Blynk.syncAll();
    Blynk.notify("Reconexión al servidor Blynk");
  }
}
```

Figura 50. Función de conexión al servidor de Blynk

Si se trata de la primera conexión, es decir cuando el hardware es desconectado de la alimentación. Se inicia el RTC interno, el cual esta referenciado al widget de Blynk para disponer del reloj en tiempo real del servidor. A continuación, se solicita todos los valores recientes almacenados en el servidor. Los valores y estados de los pines se establecen en el último valor almacenado. Cuando la conexión se haya establecido con éxito se envía una notificación push, y la variable que indica la primera conexión toma el valor de falso, para establecer las posibles reconexiones al servidor por pérdida de conexión a la red Wifi o, pérdida de comunicación con el servidor de Blynk (Figura 50).

5.4.5 Funciones

Función de comprobación de conexiones.

La función realiza la comprobación y mantenimiento de la conexión con el servidor y la conexión a la red Wifi correspondiente. La comprobación se realiza mediante la escritura detallada de los pasos que se van generando, a través la escritura en el monitor serie en el PC. La función centrada en informar de la conexión al servidor de Blynk. Función principal basada en la rutina de conexión entre el servidor y el hardware.

En el caso que exista la conexión con Blynk, imprime por el monitor serie el mensaje *"Conectado a servidor Blynk"*.
En caso contrario, cuando la conexión con el servidor se interrumpe, imprime el mensaje *"Conexión perdida"*. Esta interrupción se puede ver afectada por la caída de la conexión de la red Wifi o, la falta de comunicación con el servidor.
La primera, solucionada inicializando con los parámetros de la conexión propia a la red Wifi, y esperando 5 segundos para asegurar la conexión. Si no fuera posible la conexión con el servidor, se realizaría de nuevo el proceso.
Seguidamente, si la conexión ha sido satisfactoria o no, se muestra el mensaje apropiado. Siendo esta conexión realizada con éxito se pasa a una segunda interrupción de comunicación, que puede ser derivada de la desconexión de la comunicación Wifi o, tan solo, por la caída de la comunicación con el servidor.

```
//COMPROBACION DE FALLOS DE CONEXION Y/O COMUNICACION
void connectBlynk(){
  if(!Blynk.connected()){
    Serial.println("Conexión perdida");

    if(WiFi.status() != WL_CONNECTED){
      Serial.println("No conectado a WiFi. Conectando...");
      WiFi.begin(ssid, pass);
      delay(5000); //give it some time to connect
      if(WiFi.status() != WL_CONNECTED){
        Serial.println("No conectado a WiFi");
      }
      else{
        Serial.println("Conectado a WiFi");
      }
    }
    if(WiFi.status() == WL_CONNECTED && !Blynk.connected()){
      Serial.println("No conectado al Servidor Blynk. Conectando...");
      Blynk.begin(auth, ssid, pass);
      delay(5000); //give it some time to connect
      if(!Blynk.connected()){
        Serial.println("Conexion fallida al Servidor Blynk");
      }
    }
  }
  else{
    Serial.println("Conectado a servidor Blynk");
  }
}
```

Figura 51. Función de comprobación y mantenimiento de conexiones

La segunda de las interrupciones, si la conexión con la red wifi existe, pero no existe comunicación con el servidor de Blynk. Mediante la llamada a inicializar la sintaxis de Blynk con los parámetros correspondientes a la red Wifi y la autentificación para la nube de Blynk, esperar 5 segundos para validar la conexión. Si no fuera posible la conexión con el servidor, comienda de nuevo el condicional de comprobación (Figura 51).

Función de los valores de los sensores.

Se compone de varias funciones pertenecientes a la lectura de todos los sensores utilizados en el proyecto, y a la visualización a través de la aplicación mediante los pines virtuales correspondientes.

```
//TODOS LOS VALORES DE SENSORES
void sensores(){
  ValorHigr = analogRead(PinHumedad);
  temp = (ENV.readTemperature(CELSIUS)-7);
  hum = ENV.readHumidity();
  pres = ENV.readPressure(MILLIBAR);
  ilum = ENV.readIlluminance();
  UVA = ENV.readUVA();
  UVB = ENV.readUVB();
  UVIN = ENV.readUVIndex();

  Blynk.virtualWrite(V5, ValorHigr);  // Send it to the server
  Blynk.virtualWrite(V7, temp);
  Blynk.virtualWrite(V8, hum);
}
```

Figura 52. Función para cálculo de los todos los sensores

Los datos de los diferentes sensores empleados son declarados mediante la entrada analógica de la placa y con las respectivas funciones que nos proporciona el escudo MKR (Figura 52). El valor del higrómetro se trata de una variable de lectura analógica en el pin correspondiente a la conexión del mismo. Un segundo tipo de funciones, requiere de una inicialización de la placa escudo MKR ENV para obtener los valores de establecidos. Por último, la función necesaria para el envío de valores al servidor y visualización a través de la aplicación Blynk.

Función para el temporizador y modo automático.

La siguiente función se desarrolla a expensas de la selección sobre el botón del modo **Temporizador** (Figura 53).
Si se activa el **Temporizador**, quiere decir, la variable *temporizador* es igual a 1. Se procede a la sincronización de todos los parámetros pertenecientes al pin virtual (V10), el perteneciente al widget de entrada de tiempos (Figura 41).

Por el contrario, si la variable *temporizador* es igual a 0, provocada por la desactivación del botón en modo **Temporizador**, la entrada en el modo **Automático** (Figura 42).

```
//GESTOR DEL PROGRAMADOR Y MODO AUTOMATICO
void activeprogramador(){
  if(temporizador == 1){
    Blynk.syncVirtual(V10);   // sincronice con el timeInput
  }
  if(temporizador == 0){
    Blynk.syncVirtual(V1,0);    // MODO AUTOMATICO
  }
}
```

Figura 53. Función gestora del programador

Función *setup().*

Es la primera función en ejecutarse dentro del programa. Función que establece la configuración de un programa y los diferentes componentes físicos para que funcione correctamente en el proyecto, de ejecución única.

```
void setup(){
  //HIGROMETRO
  pinMode(PinHumedad, INPUT);
  //RELE
  pinMode(PinRele, OUTPUT);
  //SERIAL
  Serial.begin(9600);

   //// FUNCIONES CONOCEDORAS DE ERROR DE CONEXION. SOLO VISTAS EN EL MONITOR SERIE ////
  //CHECK AL MODULO WIFI.
  if(WiFi.status() == WL_NO_MODULE){
    Serial.println("Comunicación con el módulo Wifi fallida!");
    // no continua
    while (true);
  }
  //CHECK AL MODULO DE SENSORES.
  if(!ENV.begin()){
    Serial.println("Problema con la placa de expansión MKR ENV");
     // no continua
     while (1);
  }
  //CONEXION A WIFI
  while(status != WL_CONNECTED){
    Serial.print("Intentando conectar a WPA SSID: ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass);    // Credenciales de la red WIFI
    delay(3000);
  }
  Serial.println("Conexión realizada a la red Wifi.");
  //// FIN FUNCIONES CONOCEDORAS DE ERROR DE CONEXION ////

  //CONEXION DIRECTA AL SERVIDOR BLYNK Y AL WIFI
  Blynk.begin(auth, ssid, pass);  // Configurar Blynk

  //INTERVALOS DE EJECUCION DE LAS FUNCIONES
  timer.setInterval(5000L, activeprogramador);  // Verifique cada 5 seg si el temporizador debe ejecutarse
  timer.setInterval(1000L, sensores);   // Cálculo del valores de los sensores.
  timer.setInterval(60000L, connectBlynk);  // Comprueba cada minuto si todavía está conectado al servidor
}
```

Figura 54. Función setup()

El *setup() (*Figura 54), en las primeras líneas, proporciona a la placa de Arduino que determinados pines funcionan como entrada, en el caso del *PinHumedad*, y como salida en del *PinRele.*
La declaración de la comunicación serial, mediante el comando *Serial.begin().* Necesario para poder visualizar la información del hardware, en caso de fallo de comunicación con el módulo MKR ENV y error de conexión a la red Wifi.
Blynk.conneted(), gestiona la comunicación de forma directa al servidor Blynk mediante los comandos de autentificación, nombre de la red y contraseña.
En la última parte de la función *setup(),* se encuentran las definiciones de las funciones cronometradas, establecen los diferentes intervalos para la ejecución de las distintas funciones. El *activeprogramador,* temporiza cada 5 segundos la función del programador. La función de los sensores, *sensores*, ejecutada cada segundo y la función de comprobación de comunicación al servidor de Blynk y a la red Wifi, *connectBlynk.* Esto quiere decir, el programador actualiza la información en el terminal de la aplicación, la fecha y la hora en intervalos de 5 segundos. Los valores de los sensores tienen un intervalo de 1 segundo para la lectura y la comprobación de la comunicación con el servidor, realizada en intervalos de 60 segundos. (Figura 55).

Función *loop().*

Por último, la función bucle del programa *loop()* contiene la función que se ejecutará continuamente. En el bucle tan solo debe aparecer, *Blynk.run(),* al tratarse de un gestor de conexión, la rutina debe llamarse con frecuencia para procesar los comandos entrantes, realiza el mantenimiento de la conexión y la comunicación con el servidor de Blynk. Es una rutina principal de Blynk responsable de mantener activa la conexión, enviar datos, recibir datos, etc. Del mismo modo, *time.run(),* que ejecuta los tres temporizadores definidos en el *setup()* (Figura 55).

```
void loop(){
    Blynk.run();
    timer.run();
}
```

Figura 55. Función loop()

6 Resultados

Una vez el sistema está montado, la codificación instaurada en la placa y la aplicación configurada para el funcionamiento y control del sistema de riego desde una aplicación móvil, vamos a presentar los principales resultados.

Este sistema de riego se puede comportar de tres modos *Manual*, *Automático* y *Temporizado*, maneras totalmente diferentes a la hora de realizar el accionamiento de activación del relé para la apertura de la electroválvula, en este caso.

Además, dispone de los valores de la humedad de la tierra, temperatura, y humedad ambiente en tiempo real, en intervalos de 5 segundos.
El sensor de humedad de la tierra posee un valor de lectura de 0 a 1023, en consecuencia, cuanto más alto es el valor, menos húmeda esta la tierra. Se define una variable que límite el umbral para decidir si la tierra está seca, se activa el riego o, la tierra está húmeda, no se activa o se desactiva el riego. El valor umbral es 700, se hace uso en el modo *Automático*.
En relación a la variable de la temperatura, se hizo una calibración aproximada con un termohigrómetro analógico. Estando el hardware a pleno rendimiento, se comparó el valor de la variable de temperatura del sensor y el valor del termohigrómetro. Observándose una diferencia entre 7 grados, concluyendo que el calor generado del microcontrolador podía alterar la variable. Se decidió hace el ajuste pertinente en el sensor de temperatura.
En cuanto al valor de la humedad ambiente, el sensor recoge valores semejantes a los comprobados con el termohigrómetro.

Para finalizar la descripción de la aplicación, esta dispone de una interfaz de visualización de los mensajes, la cual informa en todo momento del modo que está activo, visualización de otros sensores (presión atmosférica, iluminancia, UV), informa de hora de inicio y paro del riego… y, la aplicación posee una gestión de notificaciones al móvil de activación o desactivación de los diferentes modos de riego, la desconexión/conexión/reconexión al servidor de Blynk, error a la hora de introducir la hora de riego.

La activación de la selección para el modo de riego puede realizarse de dos formas distintas. El usuario selecciona el modo de funcionamiento a través de los botones que gestionan los tipos de riego configurados, pero, puede dar la ocasión que a través del servidor de Blynk, cuando se inicie la conexión o haya una reconexión al servidor Blynk. (tras la recuperación de la conexión, el servidor envía a nuestro sistema la última información guardada en el).
Resumiendo, el sistema de riego volverá al mismo modo de la última conexión hecha con el servidor de Blynk.

El accionamiento del botón a modo *Manual* es por consigna manual o por devolución de último valor guardado, en este caso el pin virtual 1 sea igual a 1 (V1,1).
Este modo implica la apertura del relé sin ningún tipo de condición de cierre, salvo el accionamiento al mismo botón (vuelve a modo *Automático*) o, accionar el botón a modo *Temporizado*. En modo *Manual*, el terminal virtual de la aplicación muestra los valores de los sensores de presión atmosférica, iluminancia, rayos UV y índice UV.

El modo *Automático*, puede ser activado mediante estas posibilidades: el primero, por devolución de último valor guardado en servidor de Blynk, valor del pin virtual 1 igual a 0 (V1,0); el segundo, por la finalización del modo *Manual*; y, por último, por la finalización, estado en OFF del modo *Temporizado*.

Siendo así, el modo *Automático,* un modo automatizado predefinido en el código, a través del sensor de humedad de la tierra y el valor umbral, toma la decisión de regar, en caso de la tierra seca o, no realizar el riego, por estar la tierra húmeda. Este modo, informa a través del terminal virtual, y en caso de no tener la aplicación abierta, informa mediante una notificación emergente el cambio en el estado de riego.

Para la utilización del modo *Temporizador*, el usuario debe activa el botón (V11,1) mediante la aplicación o, por última conexión al servidor de Blynk.
A continuación de la activación del botón, la aplicación se sincroniza con el servidor registrando todas las variables pertenecientes al estado de tiempos (V10) para la gestión del programador horario y semanal de la aplicación. Se puede seleccionar de manera individualiza el día y elegir la franja horaria para la semana. Cabe destacar, si se desea realizar el riego diario durante la misma franja horaria, la aplicación actuará de la misma manera seleccionando todos los días o bien, no seleccionando ninguno.
Toda la información perteneciente al modo *Temporizador* es enviada cada 4 segundos al terminal virtual de la aplicación para visualizar la fecha; la hora, minutos y segundos en tiempo real; la hora de inicio de riego; la hora de finalización de riego; los días que esta activado el riego; el riego del día ha finalizado; informa de error al introducir las horas de riego.

7 Conclusiones y líneas de futuro

A modo de conclusión, vamos a retomar los objetivos pertenecientes a este proyecto.

Para empezar, el proyecto se ha desarrollado sobre un sistema empotrado de la marca Arduino. En la elección del hardware hubo que realizar un análisis de los diferentes modelos que hay en la actualidad en el mercado. Además, la valoración de diferentes sensores necesarios para el sistema de control de riego automatizado.

El proyecto ha presentado nuevos conocimientos de algunos puntos de los objetivos marcados. En relación al mundo de las comunicaciones inalámbricas con un sistema hardware, se analizan y describen las distintas comunicaciones porque a la hora de elegir el modelo de Arduino, se tenían que conocer. La placa elegida lleva el módulo Wifi integrado en la misma. Por consiguiente, las configuraciones de comunicación entre hardware y la red Wifi.

Otro de los nuevos conocimientos y objetivos del proyecto ha sido el analizar los diferentes interfaces de comunicación para conseguir la comunicación entre el hadware y la aplicación móvil y para desarrollar el control y visualización de sensores en el sistema de riego a través de la propia aplicación. Demostrando el funcionamiento el registro de las variables necesarias y el control de manera remota desde cualquier lugar del mundo, donde exista conexión a internet.

Como conclusión final, se ha realizado un proyecto de un sistema de riego automatizado simplificado con la sensorización necesaria para el conocimiento del entorno, produciéndose una comunicación estable con una red Wifi conocida y creando una aplicación que gestiona los diferentes modos de riego, atendiendo a las necesidades del momento. Todo ello con elementos hardware de bajo coste y una aplicación totalmente gratis.

En referencia a las líneas de futuro, se podrían tener en cuenta los siguientes aspectos a la hora de mejorar el diseño.
En primer lugar, el proyecto se ha realizado con un solo sensor de humedad de tierra, pudiendo abarcar en un futuro, la introducción de más sensores de medición de humedad que permitan parametrizar zonas de riego con un poco más de extensión.

Otra posible mejora, el escudo MKR ENV proporciona una ranura SD donde se puede construir una base de datos, registrando los sensores procedentes del propio escudo, valores de humedad de la tierra, numero de riegos realizados o incluso añadiendo otros, caudalímetros para medición de cantidades de agua.

Finalmente, la más atractiva en mi opinión, tratándose de un sistema de riego automatizado a pequeña escala, o “doméstico”. La placa MKR 1010 Wifi, permite que funcione a 5 voltios con energía a través del puerto USB o, mediante una batería Li-Po. Además, permite cargar la batería mientras está alimentada por el puerto USB. La conexión mediante la batería externa Li-Po permitiría la alimentación de la placa de control sin necesidad de cable para la alimentación, mientras no se pierda la conexión con la red WiFi de comunicación.

8 Bibliografía

[1] https://elblogverde.com/tipos-riego/
[2] https://es.wikipedia.org/wiki/Riego
[3] https://es.wikipedia.org/wiki/Riego_por_aspersi%C3%B3n
[4] https://es.wikipedia.org/wiki/Riego_por_goteo
[5] https://es.wikipedia.org/wiki/Hidropon%C3%ADa
[6] https://www.ambientum.com/enciclopedia_medioambiental/suelos
[7] https://es.wikipedia.org/wiki/Sistema_de_control
[8] https://www.arduino.cc/en/Guide/Introduction
[9] https://es.wikipedia.org/wiki/Internet_de_las_cosas
[10] https://www.arduino.cc/en/IoT/HomePage
[11] https://developers.mydevices.com/cayenne/features
[12] https://blynk.io/
[13] https://thingspeak.com/pages/learn_more
[14] https://www.arduino.cc/en/Products/Compare
[15] https://www.amazon.es/gp/product/B07RG27CFC
[16] https://tienda.bricogeek.com/sensores-temperatura/986-sensor-de-humedad-y-temperatura-dht11.html
[17] https://es.wikipedia.org/wiki/Fotorresistor
[18] https://store.arduino.cc/mkr-env-shield-r2