UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE REGADO CON FUNCIONES SMART

Jose Manuel Roca Roca

Tutor: Juan Carlos Guerri Cebollada

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2019-20

Valencia, 10 de septiembre de 2020

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV- 000030

Resumen

El objetivo del trabajo es la creación de un sistema de control de riego que ofrezca un amplio rango de funciones con un precio contenido. Para ello, se utilizarán distintos sensores, así como placas Arduino y un servidor alojado en dispositivo Raspberry Pi. Además, se estudiará la integración con APIs de terceros (IFTTT Webhooks) para dar la posibilidad de un alto nivel de automatización. Para ello, se emplearán diversos dispositivos ampliamente disponibles de forma económica, que, sin estar necesariamente orientados a tareas del Internet de las Cosas, permiten adentrarse en dicho territorio sin tener amplia experiencia en este tema. Se ha valorado especialmente la facilidad de uso de la instalación de forma que pueda ser utilizada por personas sin conocimientos informáticos ni telecomunicaciones.

Resum

L'objectiu d'aquest treball és la creació d'un sistema de control de reg que oferisca una amplia varietat de funcionalitats de forma econòmica. Per a aconseguir-ho, s'utilitzaran diferents sensors, així com dispositius Arduino i un servidor allotjat a una Raspberry Pi. A més a més, s'estudiarà la integració amb APIs de tercers (IFTTT Webhooks) per a permetre una automatització encara major. Per a aconseguir-ho s'utilitzaràn distints dispositius disponibles econòmicament, que, encara que no es troben enfocats a l'Internet de les Coses, permeten començar en aquest tema sense tindre una amplia experiència. S'ha valorat especialment la facil·litat d'ús per persones sense coneixements d'informàtica ni telecomunicacions.

Abstract

This project's purpose is to create an irrigation system capable of a wide variety of functions while maintaining a reasonable price. To do that, various sensors will be used alongside Arduino boards and a server located in a Raspberry Pi. Furthermore, integration with third party APIs (IFTTT Webhooks) will be studied to give an extra level of automation. To chieve this, several devices will be used, these devices are cheap and even though they are not focused on offering Internet of Things functionality, they are a good starting point without any required experience. Ease of use for people that lack information technology and telecommunications knowledge will be valued too.

Índice

Capítulo 1. Introducción

1.1 Estado del arte

Actualmente existen múltiples dispositivos en el mercado que permiten el regado automático de nuestras plantas y árboles, algunos de ellos con un funcionamiento muy simple, usando un único temporizador manual; y otros mucho más complejos que permiten la programación de varias fases e incluso ya se encuentran disponibles en el mercado algunas soluciones con conexión a Internet.

Estas últimas son las que nos interesan, ya que son la razón principal por la que se ha realizado este trabajo. De todas las soluciones disponibles, ninguna de ellas está disponible por menos de 100€, y algunos de estos, solo sirven para una sola fase.

1.2 Estado inicial de la instalación

La instalación se encuentra en una casa de campo en el término municipal de Nàquera. La casa cuenta con tres zonas ajardinadas y de cultivo equipadas con tuberías de riego, algunas con terminales de riego por micro goteo y otras mediante aspersores.

Las zonas de riego se encuentran divididas en cinco fases, hecho motivado, por una parte, por la separación física entre ellas, y, por otra, por permitir un nivel de presión del agua aceptable de forma que el regado sea efectivo. El proceso de encender o apagar cada una de las fases era manual, por lo que se hacía necesario encontrar una solución que permitiera realizar el regado de forma autónoma.

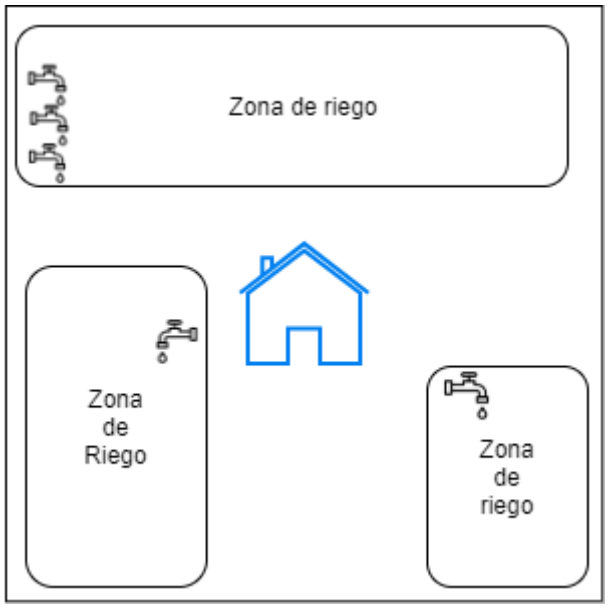

Figura 1: Ubicación de las zonas de riego y las tomas de agua.

1.3 Motivación

Como el estado inicial de la instalación hace inviable su uso en ausencia de usuarios y teniendo en cuenta los productos disponibles que permiten realizar un uso y seguimiento remoto, se ha decidido comprobar si es posible la creación de un sistema de riego con conexión a Internet con un precio contenido y sencillo de utilizar.

Además de la cuestión del precio, se consideran también las dudas sobre la protección que las empresas hacen de nuestros datos e incluso el funcionamiento mismo de los dispositivos en un futuro, ya que, en caso de que las compañías cierren, estos dispositivos pasarían a ser programadores de riego estándar en el mejor de los casos o incluso podrían dejar de funcionar totalmente.

1.4 Objetivos

Se han tenido en cuenta una serie de objetivos por cumplir, el principal y más importante es el de crear un sistema de riego que permita un funcionamiento autónomo. También se han tenido en cuenta los siguientes objetivos:

- Hacer un seguimiento de la humedad del suelo para tener un histórico de datos visualizable y a su vez que sirva como retroalimentación para decidir un tiempo de regado óptimo.

- Permitir un uso tanto local como remoto.

- El sistema debe avisar al usuario en caso de que se produzcan errores.

- Hacer el sistema lo más intuitivo posible.

- Permitir el control del sistema de riego mediante plataformas externas, como IFTTT y asistentes de voz.

1.5 Conocimientos previos

Durante el grado se han adquirido los conocimientos necesarios para programar pequeños scripts y aplicaciones de diversa índole, empleando lenguajes como C, Java, ensamblador o Verilog, entre otros.

Se han adquirido también conocimientos de electrónica que serán de utilidad para el uso de los distintos dispositivos que utilizados durante este proyecto.

En el itinerario de telemática se ha hecho hincapié en distintos protocolos de comunicaciones entre dispositivos que también serán de utilidad durante el transcurso del trabajo.

Se puede afirmar, por lo tanto, que, gracias a los conocimientos adquiridos durante el grado, se pueden llevar a cabo los objetivos definidos para este trabajo, y que cualquier dificultad que pueda surgir será sencilla de aliviar.

Capítulo 2. Metodología

2.1 Gestión del proyecto

Como este proyecto se trata de un trabajo individual, no es necesario utilizar ninguna metodología de desarrollo ágil, por lo que, simplemente, se ha utilizado un diseño evolutivo, en el que, tras desarrollar cada etapa del proyecto, se testea y se construye sobre ella.

Se ha utilizado la herramienta Git para permitir la sincronización del código entre distintos dispositivos, de forma que siempre se tiene acceso a él rápidamente. En caso de haber sido este un proyecto de mayor envergadura o desarrollado entre varias personas, habría sido especialmente útil para poder compartir los últimos cambios de código de forma remota.

2.2 Distribución en tareas

Al inicio del proyecto, se tuvieron en cuenta los requisitos y los objetivos por completar y se dividió el trabajo en varias tareas.

1. Elección de la arquitectura que utilizar.
2. Elección de los dispositivos que utilizar, teniendo en cuenta que han de ser capaces de cumplir los objetivos definidos.
3. Elección de la plataforma de software.
4. Adquisición de los dispositivos.
5. Configuración y programación para que los dispositivos realicen las funciones necesarias.
6. Montaje del hardware.
7. Testeo del sistema en su conjunto.

2.3 Diagrama temporal

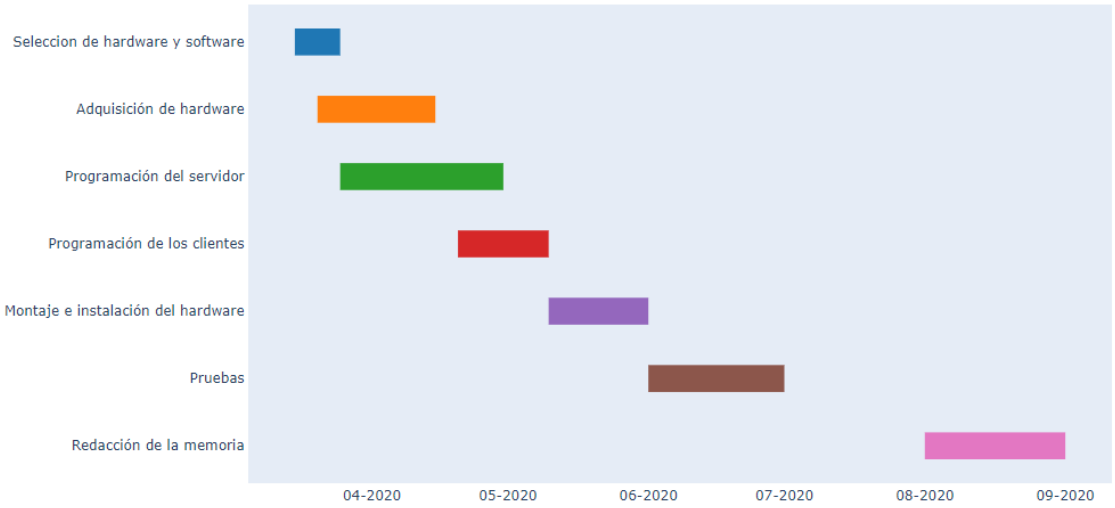

Figura 2

Capítulo 3. Especificaciones del sistema

3.1 Arquitectura elegida

Se han decidido llevar a cabo los objetivos del trabajo mediante el uso de una arquitectura cliente-servidor en estrella, utilizando una Raspberry Pi como servidor y distintas placas Arduino que, comunicándose de forma inalámbrica con el servidor, controlarán las distintas fases de riego y le informarán periódicamente de su estado.

Debido a la disposición física de la casa dentro de la parcela, y estando las salidas de riego ubicadas a lo largo de su perímetro, se ha elegido una arquitectura en estrella, ya que el dispositivo más cercano que los clientes tendrán en todo momento será el servidor.

Como se puede observar en la figura 3, dos de los clientes controlarán una fase cada uno, mientras que el cliente restante controlará tres.

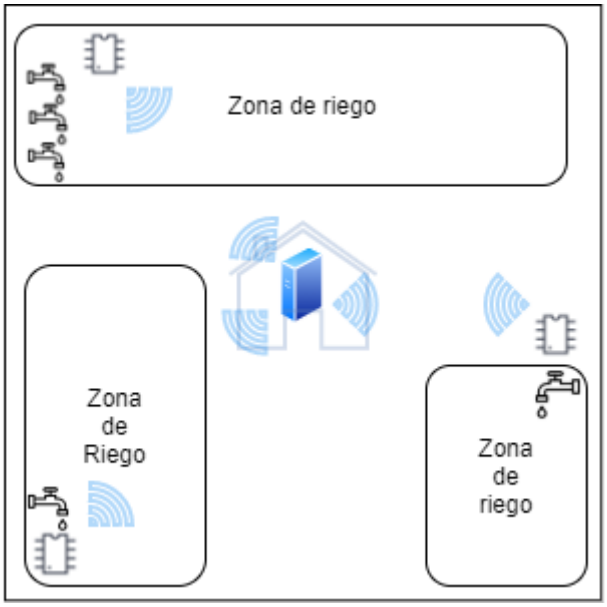

Figura 3: Arquitectura de la red

3.2 Hardware

3.2.1 Raspberry pi

Raspberry pi es una serie de ordenadores sbc (del inglés *single board computer*) de bajo coste, desarrollada por la Raspberry Pi Foundation para ayudar en la enseñanza de informática en los colegios y en países en vías de desarrollo. Desde el lanzamiento de su primer modelo en 2012, se le dieron muchos usos fuera de su propósito inicial, usándose en multitud de proyectos de muy diversa índole como servidores o controladores de robótica.

A lo largo de los años, la Raspberry pi foundation ha lanzado distintas nuevas versiones de Raspberry Pi que varían principalmente en su soc (*system on a chip*), ram y factor de forma.

Ya que todas las versiones de Raspberry Pi cuentan con procesadores ARM, tienen un consumo eléctrico muy reducido, convirtiéndolas en dispositivos ideales tanto para el público *maker* como para otras tareas que no requieran una amplia capacidad de computación.

Para este trabajo se ha utilizado una placa Raspberry Pi 2 ya que ya se encontraba disponible en las premisas de la instalación. Este modelo de Raspberry Pi cuenta con un procesador de cuatro núcleos a 900Mhz fabricado por Broadcomm, e integra un Gigabyte de memoria RAM.

Respecto a la conectividad, cuenta con un puerto Fast Ethernet, cuatro puertos USB, 40 puertos GPIO, interfaces CSI y DSI, así como un lector de tarjetas microSD para almacenar el sistema operativo. [1]

Como todos los modelos de Raspberry Pi utilizan la misma arquitectura, el software escrito para un modelo es, en la mayoría de los casos, totalmente compatible con el resto.

Figura 4: Raspberry Pi 2 [2]

3.2.2 Arduino

Arduino es una plataforma de electrónica open-source creada con el propósito de facilitar el aprendizaje y desarrollo de aplicaciones de electrónica sencillas. El proyecto comenzó en 2005 en Italia, con el objetivo de ser una herramienta de enseñanza en el instituto IVREA. Sus características diferenciadoras respecto a otras plataformas de desarrollo son su bajo coste y la gran comunidad con la que cuenta.

A lo largo de los años se han lanzado múltiples modelos de placas Arduino, al contrario que Raspberry Pi, las diferencias entre los diversos modelos de Arduino van más allá de simples mejoras en el rendimiento, incluyendo algunos de los modelos superiores, incluso procesadores ARM multinúcleo.

En su mayor parte, las placas Arduino utilizan microcontroladores Atmel AVR, que, integrados en una placa de circuito impreso, utilizan puertos de entrada y salida para comunicarse con dispositivos externos. Al aumentar su popularidad, comenzaron a comercializarse pequeñas placas adaptadoras conocidas como shields. Estos shields permiten la conexión de dispositivos de forma sencilla y suelen ofrecer una capa de abstracción adicional frente al dispositivo conectado para facilitar el desarrollo.

La mayoría de las placas Arduino pueden ser alimentadas mediante un puerto USB, existen variantes que, además, cuentan con un puerto tipo barrel; también pueden ser alimentadas mediante pines en la placa. Además de alimentación, la interfaz USB de las placas suele incorporar una interfaz serie para permitir la programación del microcontrolador.

Las placas cuentan con un cargador de arranque (bootloader) que permite el inicio del código programado por el usuario.

Para este proyecto se han usado varias placas Arduino, concretamente una placa Arduino Uno durante el desarrollo y tres Arduino Nano para la instalación final. Cada Arduino lleva conectado un transmisor de radiofrecuencia y relés para controlar una o varias fases de riego.

Figura 5: Arduino Uno [3]

3.2.3 Transmisores NRF24L01

Para la transmisión inalámbrica de datos se han empleado transmisores NRF24L01, fabricados por Nordic Semiconductor para aplicaciones de bajo consumo. Disponen de librerías para Raspberry Pi así como Arduino, por lo que son una herramienta ideal para la realización de proyectos que los utilicen.

El NRF24L01 integra un transceptor RF de 2400 a 2525 MHz, lo que permite utilizar un total de 125 canales de 1Mhz. [4]

El chip dispone de una velocidad de transmisión seleccionable de 250 Kbps, 1Mbps, o 2 Mbps. Se ha elegido la velocidad de 250 Kbps para la transmisión para reducir la tasa de errores y el alcance. [5]

El NRF24L01 permite la conexión inalámbrica entre dispositivos usando un transceptor en la banda sin licencia de los 2.4 Ghz, al igual que los protocolos WiFi, Bluetooth, etc. Permite el uso de 125 canales de 1 Mhz. Dispone de tres velocidades de transmisión entre las que elegir, 250 Kbps, 1Mbps, y 2 Mbps. En el caso de elegir 2Mbps se recomienda usar una separación de canales de 2Mhz, ya que de lo contrario se produciría un solapamiento entre ellos.

Normalmente el chip se puede encontrar montado en un módulo para facilitar la conexión con los dispositivos que vayan a hacer uso de él.

El módulo se conecta a los dispositivos a través de un bus SPI (Serial Peripherial Interface), por lo tanto, es compatible tanto con Arduino como con Raspberry Pi, así como con otras placas similares que integren dicho bus de datos.

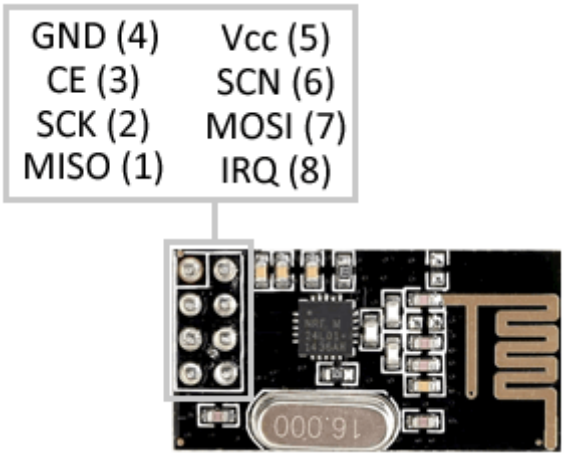

Figura 6: Modulo NRF24L01 con sus conexiones. [6]

Como se puede ver en la figura 6, los dos primeros pines son Vcc y GND, que proveen de corriente y tierra al transmisor, que funciona a 1.9-3.6 voltios[7], por lo que se puede conectar directamente a los puertos de 3,3 voltios de las placas que se usan en el proyecto. El consumo del transmisor durante la transmisión alcanza unos 13.5 mA [8], por lo que pueden llegar a saturar el pequeño regulador que utilizan las placas, así que se ha decidido soldar condensadores de 110 uF entre Vcc y GND así como usar transformadores capaces de entregar la potencia suficiente. Los pines de datos se pueden conectar directamente a los puertos de datos de las placas.

Los pines CE (*Chip Enable*) y CS (*Chip Select)* controlan los modos de funcionamiento del transmisor, el pin CE sirve para alternar entre los modos de transmisión y recepción; mientras que el pin CS (a veces llamado CSN ya que se activa a nivel bajo) se usa para habilitar la transmisión de datos a través del bus SPI.

El pin SCK transporta los pulsos de reloj del bus SPI para que, tanto el dispositivo maestro, como el esclavo, se encuentren sincronizados.

Los pines MISO (*Master In Slave Out*) y MOSI (*Master Out Slave In*) transportan la información del bus SPI. Como sus nombres indican, MISO transporta datos del maestro (la placa) al esclavo (el transmisor), mientras que MOSI realiza la transmisión inversa, es decir, del esclavo al maestro.

IRQ (*Interrupt Request*) se activa a nivel bajo cuando se produce una interrupción en el transmisor. Este pin no se usará en este proyecto.

El NRF24L01 permite una comunicación robusta mediante control de errores y reenvío de mensajes.

El módulo utiliza SPI como protocolo de comunicación con el host, por lo que es sencillo controlarlo desde cualquier dispositivo que implemente dicho bus, como Arduino y Raspberry Pi.

Existen dos tipos de módulo dependiendo de la antena que incorporen, existen aquellos con antena integrada y que tienen un alcance de unos 30 metros, así como otros que poseen antena y amplificador de alta ganancia y que permiten comunicaciones a mayor distancia.

3.2.4 Módulos YL-38 y sensores de humedad YL-69

Para poder decidir el tiempo de riego se hace necesario tener un conocimiento sobre las condiciones a las que se encuentra el suelo, para ello se utilizan los módulos YL-38, que en combinación con el sensor de humedad YL-69 permite realizar una medición continua de la humedad del suelo. Su funcionamiento se basa en el hecho de que la tierra varía su resistencia de forma inversa respecto al nivel de humedad. Por lo tanto, medir la humedad de la tierra es una tarea tan sencilla como medir la resistencia entre los dos terminales del sensor YL-69. El módulo YL-38 mide esta resistencia y proporciona dos salidas, una analógica y una digital. La analógica permite una recogida de datos en un rango de 0 a 360.

La salida digital únicamente se activa si la humedad baja por debajo de un cierto nivel, pudiéndolo ajustar usando el potenciómetro incorporado al módulo.

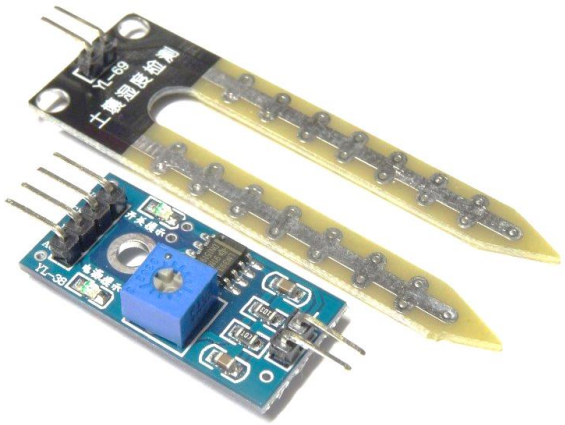

Figura 7: Módulo YL-38 junto al sensor de humedad YL-69

Se ha decidido utilizar el modo analógico ya que permitirá un control más fino sobre el tiempo de riego, además gracias a este modo podemos almacenar los datos proporcionados de forma que se pueda visualizar la información y valorar la efectividad de la instalación.

Así, para tomar medidas únicamente han de insertarse los terminales del sensor en el suelo de forma que se pueda aplicar una pequeña corriente y medir así la resistencia, y, por lo tanto, la humedad de la tierra.

Figura 8: Sensor YL-69 una vez insertado en la tierra.

3.2.5 Electroválvulas de riego

Es necesario en este proyecto contar con un dispositivo que abra o cierre el caudal de agua de forma electrónica y fiable, para ello se han adquirido cinco electroválvulas y se han instalado en lugares cercanos a las distintas fases de riego y a tomas de corriente eléctrica, su funcionamiento se basa en el uso de un electroimán. Dicho electroimán se encuentra normalmente obstruyendo el paso de agua a través de la válvula, pero si se le aplica una corriente eléctrica el imán se desplazará permitiendo así que el agua fluya. Adicionalmente, las electroválvulas permiten un control manual mediante una rosca integrada, en caso de ser necesario.

Figura 9. Electroválvula

Las electroválvulas utilizadas utilizan una tensión de 24 voltios de corriente alterna para accionar su mecanismo de funcionamiento, puesto que las placas Arduino no son capaces de entregar ese voltaje, se han usado transformadores, que en combinación con relés electromagnéticos permitirán el accionado del electroimán.

3.2.6 Relés electromagnéticos

El concepto de funcionamiento de un relé electromagnético es muy similar al de las electroválvulas, ya que se basan en un imán que acciona un mecanismo cuando recibe energía eléctrica, en este caso se acciona un interruptor al recibir una señal a nivel bajo por su puerto de entrada.

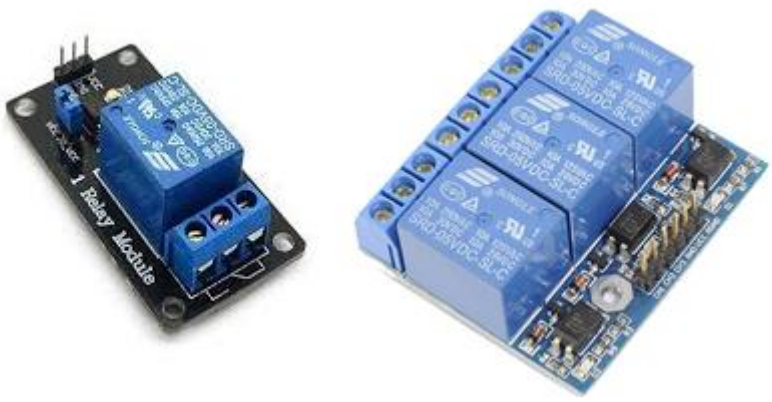

Figura 10: Relés utilizados

3.3 Software

3.3.1 Node-RED

Node-RED es una herramienta de programación y automatización por flujos con una intuitiva interfaz visual. Su primera versión estable fue lanzada el 9 de abril de 2020 y previamente ya había cosechado una fama favorable debido a su facilidad de uso y sus posibilidades de extensión.

La programación por flujos es una forma de definir el comportamiento de una aplicación, mediante el uso de "nodos", cada nodo tiene un objetivo, y si se le proporcionan datos por la entrada, el nodo realizará una acción sobre dichos datos y los retransmitirá por su salida.

En la actualidad existen múltiples plataformas de programación por flujos, principalmente en el sector empresarial, como el ESB (*Enterprise Service Bus*) Mule.

Por definición, resulta muy cómodo representar visualmente un programa creado con programación por flujos, cosa que la hace muy atractiva de cara a nuevos usuarios ya que se puede observar de un vistazo los distintos pasos que se llevan a cabo durante la ejecución del programa.

Node-RED utiliza un entorno de ejecución basado en Node.js al que se accede mediante cualquier navegador web, el cual permite construir programas sin escribir grandes cantidades de código. El entorno dispone de un editor de flujos basado en tecnología web, donde se pueden visualizar y modificar utilizando nodos con el fin de hacer que dispositivos y servicios de todos los tipos y arquitecturas se comuniquen entre ellos.

En Node-RED, se dispone de una "paleta" de nodos inicial muy completa que permite llevar a cabo una amplia variedad de proyectos, siendo posible añadir nuevos nodos mediante la herramienta npm.

Gracias a estas características, Node-RED permite emplear menos tiempo en la planificación, diseño, y puesta a punto de los distintos conectores que se tendrían que utilizar utilizando un entorno de desarrollo al uso.

Al estar basado en Node.js, permite ser ejecutado en prácticamente cualquier dispositivo que soporte javascript, así como instancias en la nube (Azure, AWS…). Además, permite un acceso a las funciones específicas de cada uno de estos dispositivos.

Relativo a compatibilidad con sbcs, cuenta con soporte oficial para placas Raspberry Pi, así como BeagleBoard[9]; además, hay versiones desarrolladas por la comunidad que permiten su uso en dispositivos que no cuentan con soporte oficial, como Orange Pi, por ejemplo.

Cuenta además con gran cantidad de nodos desarrollados por la comunidad, por lo que se facilita mucho la conexión con dispositivos y servicios adicionales.

Relevante para este proyecto es que tiene acceso a los pines gpio (*general purpose input/output*) integrados en las placas Raspberry Pi.

3.3.2 Arduino

Arduino no es solo una plataforma de hardware, sino que es también el software que la hace funcionar. El entorno de software Arduino se distribuye con la librería Arduino.h, y al importarse permite el acceso a todos los pines, bancos de memoria y funciones del microcontrolador disponibles.

3.3.3 IFTTT

IFTTT (*If This Then That*), es una plataforma online que permite la creación de sencillos comandos condicionales llamados applets.

El funcionamiento de un applet se basa en la activación de un trigger (this), que provoca la ejecución de una acción (that).

Un usuario puede crear una cuenta gratuita y, usando la web o la aplicación disponible para Android y iOS, asociarla a sus servicios y dispositivos de forma que puede activar y crear recetas con solo unos pocos clicks. Prácticamente todos los dispositivos de domótica cuentan con integración con IFTTT, así como muchos de los servicios que los usuarios utilizan en su día a día, como email, calendario, el tiempo o las noticias.

IFTTT dispone de dos planes para entidades que deseen crear nuevos servicios: un plan gratuito que permite la creación de applets y otro de pago enfocado a empresas que además proporciona datos analíticos, integración de IFTTT en dispositivos y servicio técnico.

Para el público maker, IFTTT dispone de una funcionalidad llamada Webhooks, que mediante llamadas http, permite a los usuarios integrarlos con sus dispositivos de iot que no disponen de integración nativa. El uso de Webhooks requiere de una clave para la api que se encuentra en el panel de usuario de IFTTT.

Capítulo 4. Análisis y diseño

Durante la fase de diseño de la solución de software se tuvieron en cuenta los objetivos y se definieron los siguientes procesos por realizar por cada una de las partes del proyecto:

4.1 Servidor

4.1.1 Modo automático

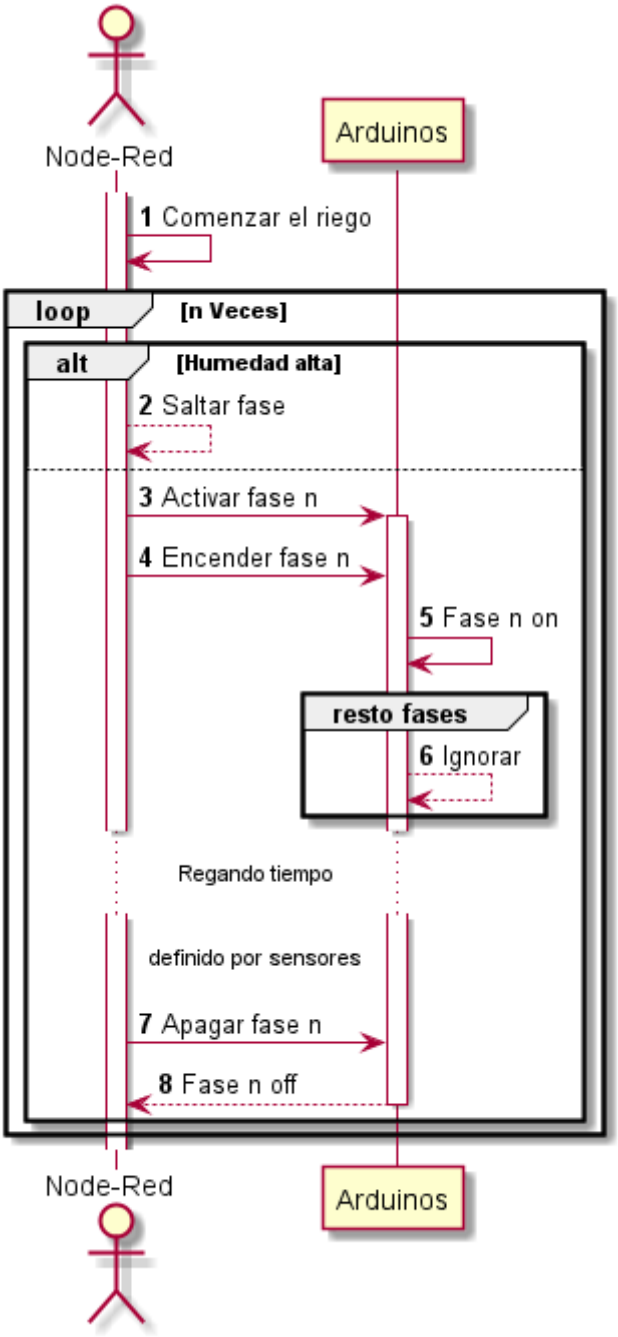

Figura 11: Funcionamiento del modo automático

En el modo automático, en función de la información proporcionada por los sensores, se decide un tiempo de riego.

De forma empírica se ha comprobado que, en un día de verano caluroso, los sensores de humedad alcanzan unos valores alrededor de los 300 Ω, mientras que un día con lluvia puede bajar hasta los 100Ω e incluso menos.

Con esta información se puede deducir que mantener la humedad del suelo de forma que ofrezca una resistencia de unos 200 Ω permitiría tener un suelo húmedo sin incurrir en importantes gastos de agua.

De esta forma, se ha definido una ventana por la que, en función de la resistencia, se modificará el tiempo de riego. La fórmula que se empleará es la siguiente:

$$t = 7 * h - 150 \quad \textbf{(4.1)}$$

Siendo h la humedad detectada y t el tiempo que se regará, en segundos.

Por lo tanto, logramos tener un tiempo de regado reducido si la resistencia del suelo es de unos 150 Ω, mientras que alcanzaremos el tiempo máximo de riego (media hora aproximadamente) cuando se encuentre alrededor de los 300 Ω. Si los valores de resistencia se encuentran fuera del rango 150-300 Ω no se usará la fórmula, sino que simplemente se bloqueará el riego en caso de ser inferior. Se regará el tiempo máximo de media hora y se avisará al usuario mediante correo electrónico si se supera el límite.

4.1.2 Funciones manuales y conexiones a servicios de terceros

Gracias al uso de Node-RED, IFTTT y Alexa, el usuario dispone de funciones casi ilimitadas de configuración. A continuación, se detallan aquellas que se dejarán preconfiguradas y preparadas para su uso.

Para recibir peticiones de IFTTT Webhooks se ha de habilitar un canal de comunicación HTTP mediante que use objetos JSON. Esta arquitectura no solo permite la recepción de mensajes de IFTTT, sino que cualquier petición con el formato correcto será atendida, de forma que siempre se tiene la opción de usar otro servicio en caso de que sea necesario.

Activación y desactivación por voz: a través del asistente virtual Alexa de Amazon, de los cuales el usuario posee varios dispositivos, es posible encender o detener el regado de forma manual. Esto se consigue mediante el uso de un nodo específico que permite que el sistema sea detectado por Alexa como un dispositivo genérico con función de apagado y encendido.

Tras definir los nodos y la configuración de puertos, bastará con decir “Alexa, detecta dispositivos” para que el dispositivo creado quede asociado a la cuenta del usuario.

Finalmente, el usuario podrá decir “Alexa, enciende el riego” o “Alexa, apaga el riego” para así iniciar o detener el proceso de regado. Si se desea un control por fases, también se permitirá el control individual diciendo “Alexa, enciende fase”, seguido del número de la fase en cuestión.

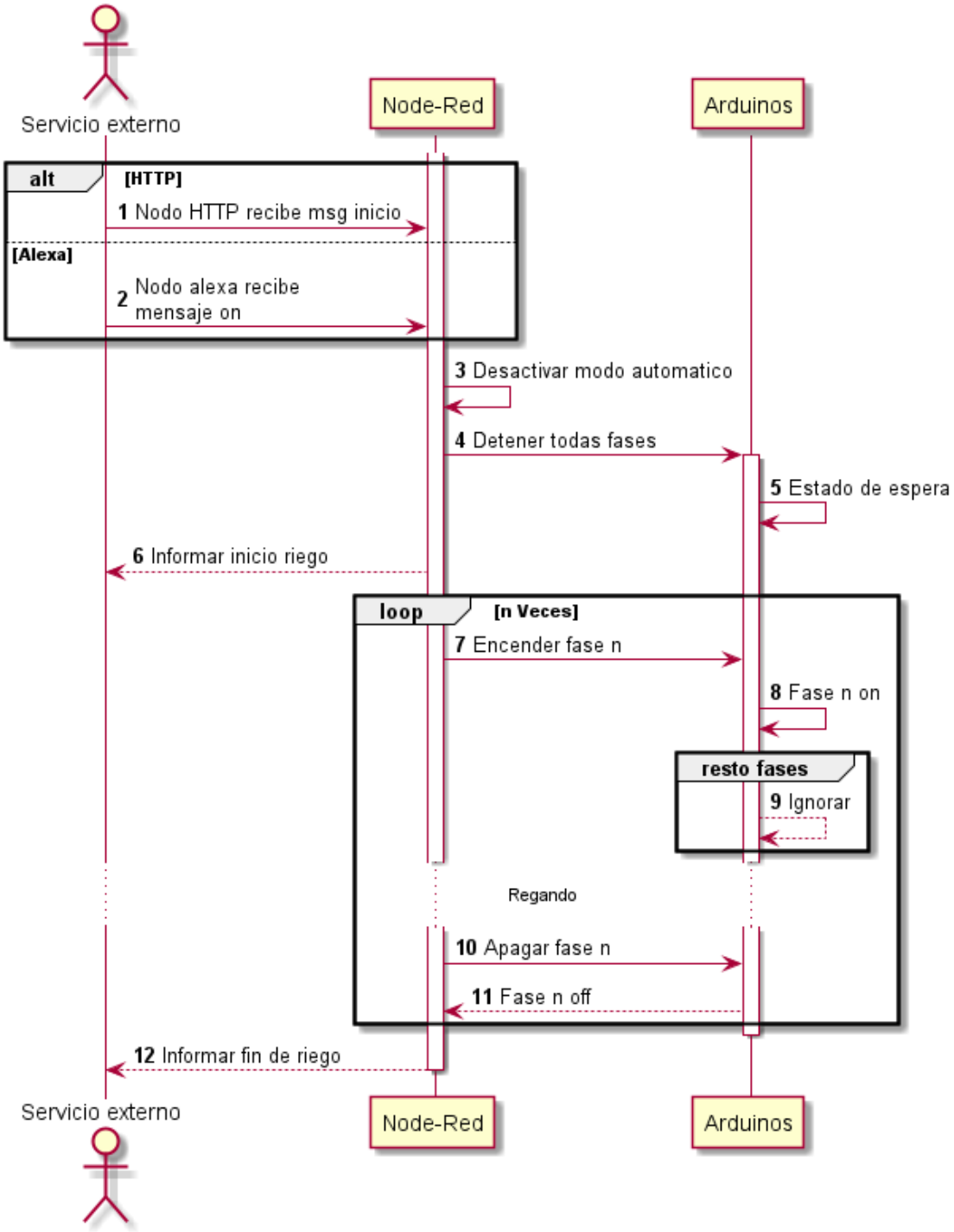

Figura 12: Inicio manual por servicios externos

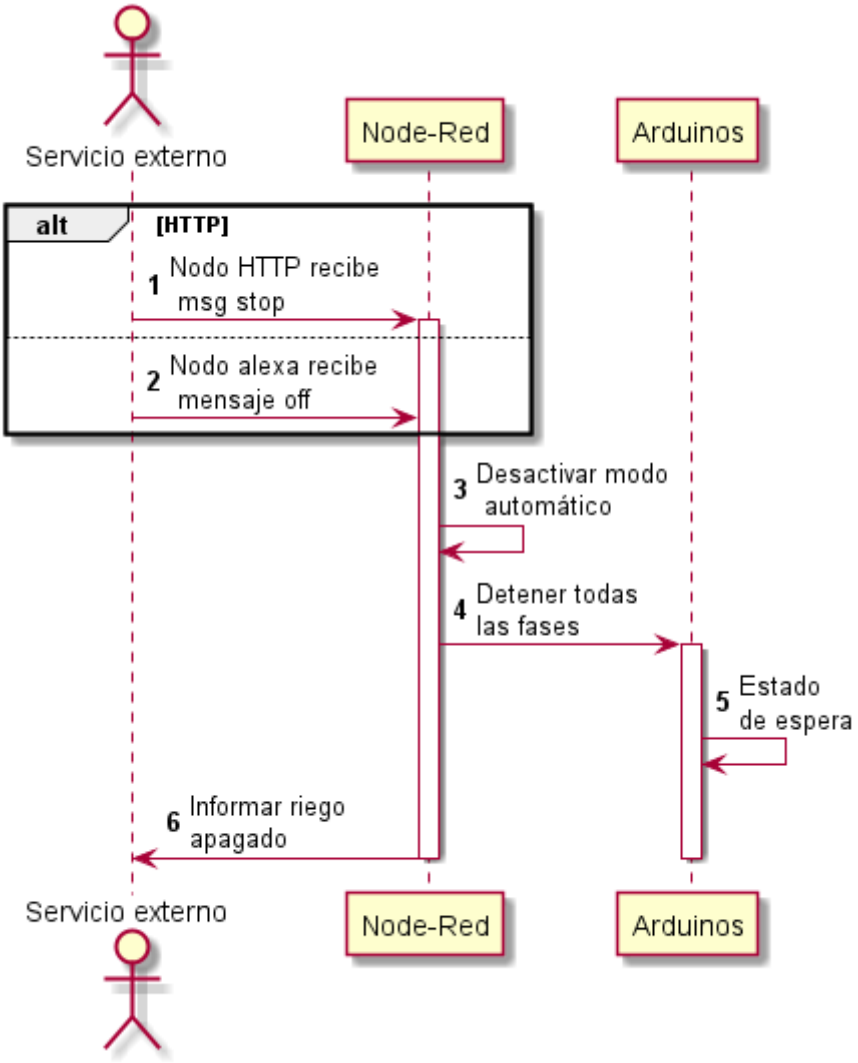

Figura 13: Parada manual por servicios externos

4.2 Clientes

Las placas Arduino, que actúan de clientes, permitirán la recepción de mensajes por parte del servidor, de forma que puedan ser encendidas y apagadas de forma remota.

Sin embargo, las placas han de contar con un cierto grado de autonomía, ya que, en caso de no recibir mensajes de apagado, deben apagarse de forma que no se malgaste agua, este temporizador se ha definido en media hora, ya que es el tiempo máximo de riego de cada fase en condiciones normales.

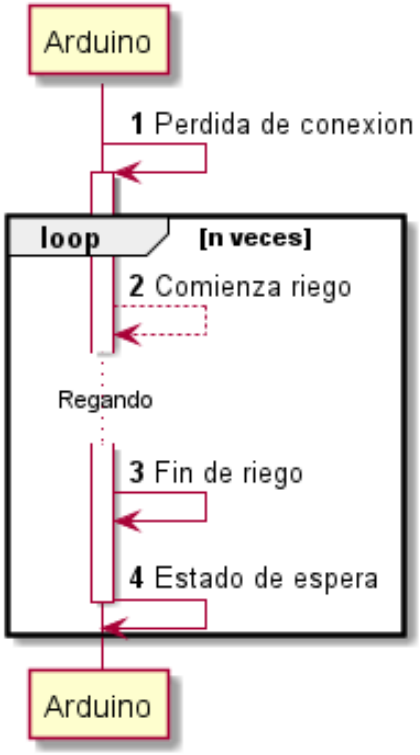

Figura 14: Modo stand alone.

Además, los clientes han de ser capaces de tomar el control en caso de que se pierda la conexión con el servidor durante más de 12 horas.

Capítulo 5. Desarrollo

5.1 Montaje e instalación del hardware

5.1.1 Raspberry Pi

5.1.1.1 Módulo NRF24L01

Como la Raspberry Pi está equipada con un bus SPI, la conexión con los módulos es sencilla, tan solo se deben conectar los pines siguiendo la siguiente tabla.

Pin de la antena	Pin de la Raspberry Pi
GND	6
Vcc	1
CE	11
CSN	24
SCK	23
MOSI	19
MISO	21

Tabla 1

Figura 15: Raspberry Pi con módulo NRF24L01

5.1.2 Placas Arduino

5.1.2.1 Electroválvulas

Las electroválvulas se han instalado en las distintas tomas de agua de la zona, para ello ha bastado con desacoplar los tubos de riego existentes y usar adaptadores para permitir que cada electroválvula pueda ser conectada tanto a la toma de agua como a los tubos de riego.

De cada electroválvula salen dos cables, en caso de usar un programador de riego estándar, un cable se conectaría a una de las clavijas del programador, mientras que el otro se conectaría a la clavija común del programador. Como en este proyecto es necesario conectarlas a un relé electromagnético, se ha conectado uno de los cables a una de las tomas de los transformadores de 24V, y la otra al puerto "normalmente abierto" de un relé.

5.1.2.2 Relés electromagnéticos

Los relés electromagnéticos tienen tres puertos a los que se pueden conectar los dispositivos al ser accionados. Ellos son el puerto NC (normalmente cerrado), el puerto NA (normalmente abierto) y el común. Cuando se conecta un dispositivo hay que elegir entre conectarlo al puerto NC o al NA. Como nos interesa que el relé solo active la electroválvula al recibir una señal del Arduino, se ha decidido conectar uno de los cables de la electroválvula al puerto NA mientras que el cable restante del transformador se conectará al puerto común.

La conexión de los relés con las placas Arduino se realiza mediante tres pines, el pin vcc, ground y el pin de entrada de datos. Como los relés funcionan con un voltaje de 5 V, el pin vcc se conecta al conector de 5V de la placa Arduino. El pin de tierra se conecta al pin GND y el pin de entrada de datos, a cualquiera de los pines digitales disponibles en la placa, en este caso al pin digital número 2.

5.1.2.3 Módulos YL-38

Los módulos se pueden conectar tanto al puerto de 3.3 voltios como al de 5 voltios, por lo que se ha decidido conectarlo al de 5V para reducir lo máximo posible las interferencias con el módulo de radio. Como se ha decidido usar el modo analógico de los detectores de humedad, se ha conectado el pin AO a uno de los pines analógicos de la placa Arduino, en este caso, el pin A2.

El sensor de humedad únicamente tiene que conectarse a los pines negativo y positivo del módulo para funcionar.

5.1.2.4 Módulos nrf24l01

Pin de la antena	Pin de Arduino Nano
GND	GND
Vcc	3V3
CE	D9
CSN	D10
SCK	D13
MOSI	D11
MISO	D12

Tabla 2

Como los pines entre los distintos modelos de Arduino son distintos, se muestran las conexiones tanto para Arduino uno como para Arduino nano. Finalmente, los pines CE y CS del módulo pueden conectarse a cualquier par de pines digitales del Arduino.

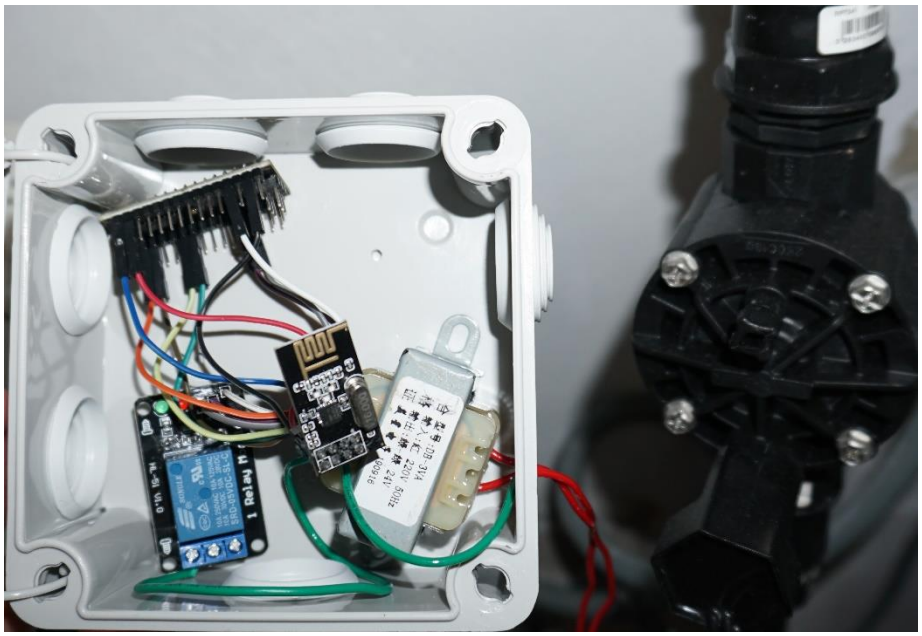

Figura 16: Arduino Nano conectado a los distintos dispositivos de la instalación

5.2 Software

5.2.1 Node-RED

Para la instalación de Node-RED en una Raspberry Pi con Raspbian lite, basta con introducir el comando “npm install -g node-red”, aunque se recomienda usar el script disponible en su página web [10]. Con Raspbian full viene instalado por defecto, aunque también se recomienda actualizarlo usando el mismo script, ya que también se ocupa de comprobar dependencias como node.js y su gestor de paquetes, npm.

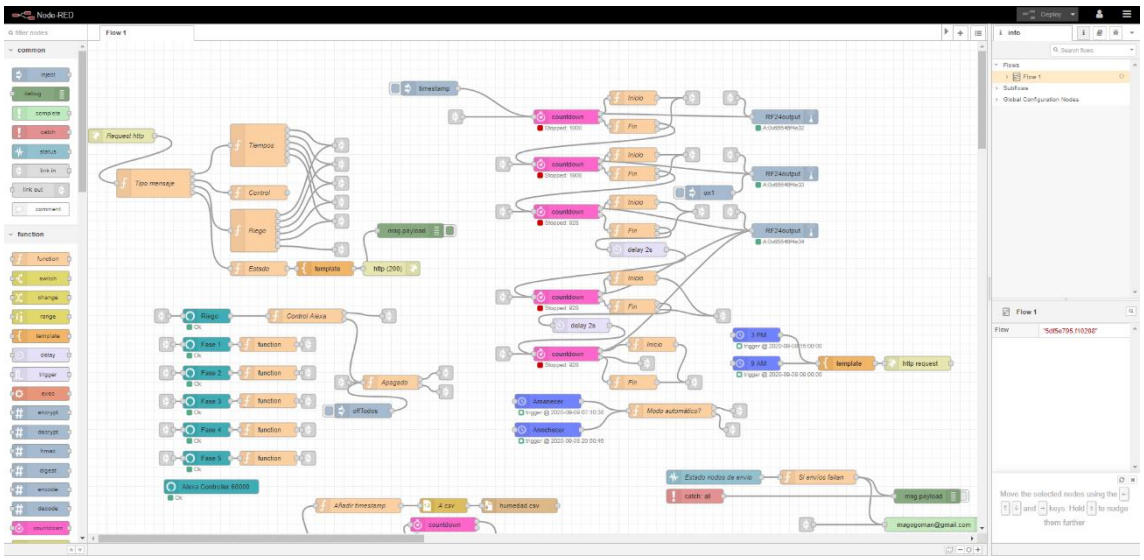

Figura 17: Editor de Flows de Node-RED

5.2.2 Visual Studio Code con Platformio

Pese a que Arduino pone a nuestra disposición un entorno de desarrollo básico, se ha optado por usar un entorno de desarrollo más potente como Visual Studio Code.

Visual Studio Code no es un entorno de desarrollo en sí mismo, pero permite la instalación de plugins que lo complementan en muchos aspectos. En este trabajo se ha utilizado junto al plugin Platformio, ya que integra herramientas para ser usado fácilmente con multitud de placas de desarrollo y microcontroladores de todas las marcas y fabricantes.

5.3 Programación del servidor con Node-RED

5.3.1.1 Editor de flujos

Para la programación del servidor se ha de utilizar la herramienta que Node-RED nos proporciona, el editor de flujos integrado.

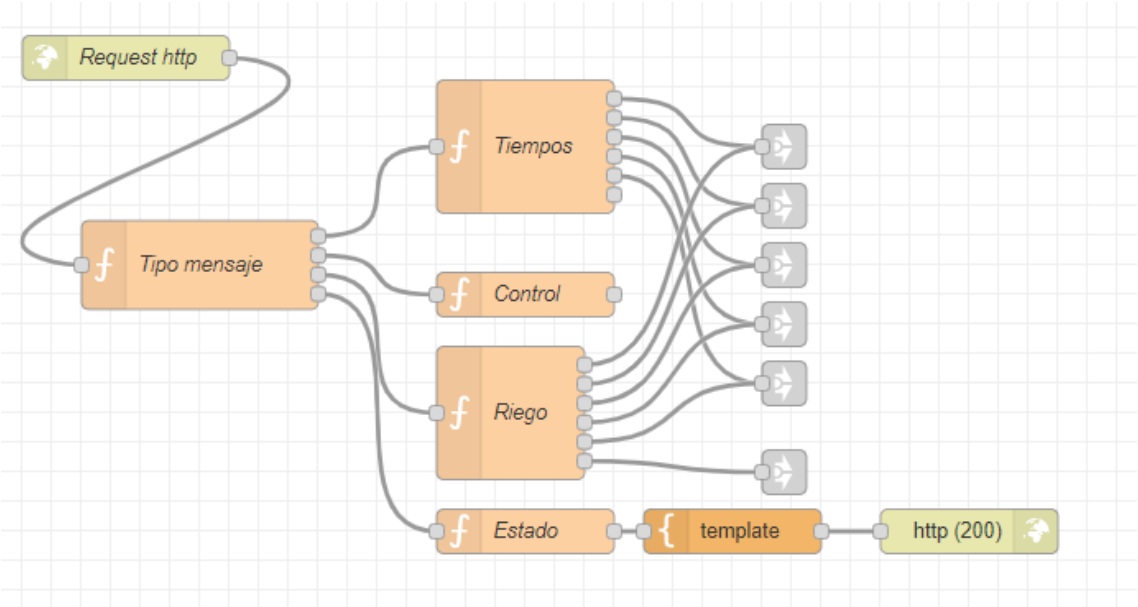

Figura 18: Entrada http del proyecto

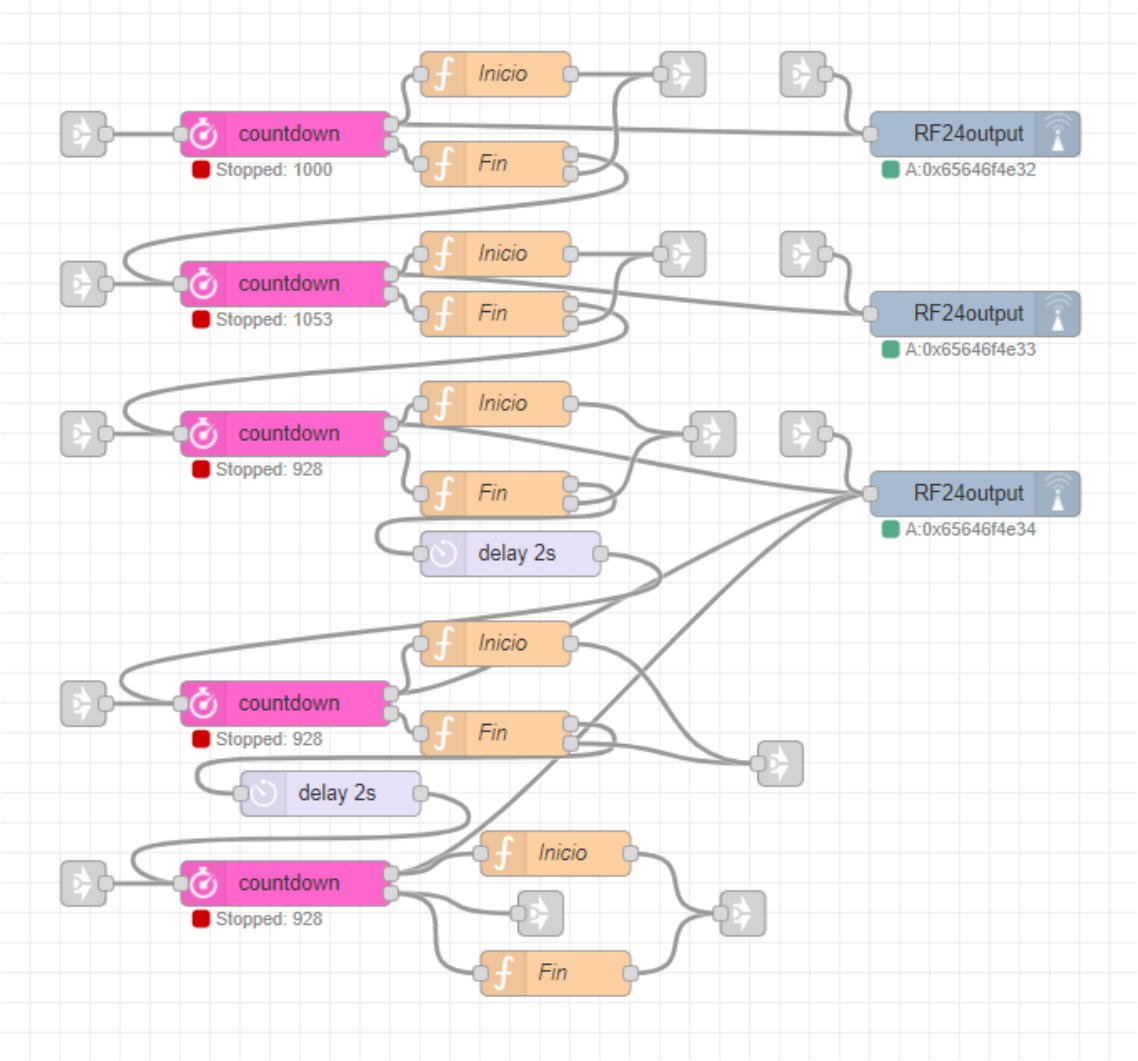

Figura 19: Flujo de regado

Al ser Node-RED una herramienta de programación visual, es muy sencillo trasladar los requisitos al editor de flujos, de forma que tenemos un esbozo de las diversas acciones que va a llevar a cabo la aplicación, así como sus componentes, sus entradas y sus salidas. En las figuras que muestran el editor de flujos, solo ha sido necesario programar los nodos de función, por lo que se hace obvia esta comodidad que Node-RED nos proporciona.

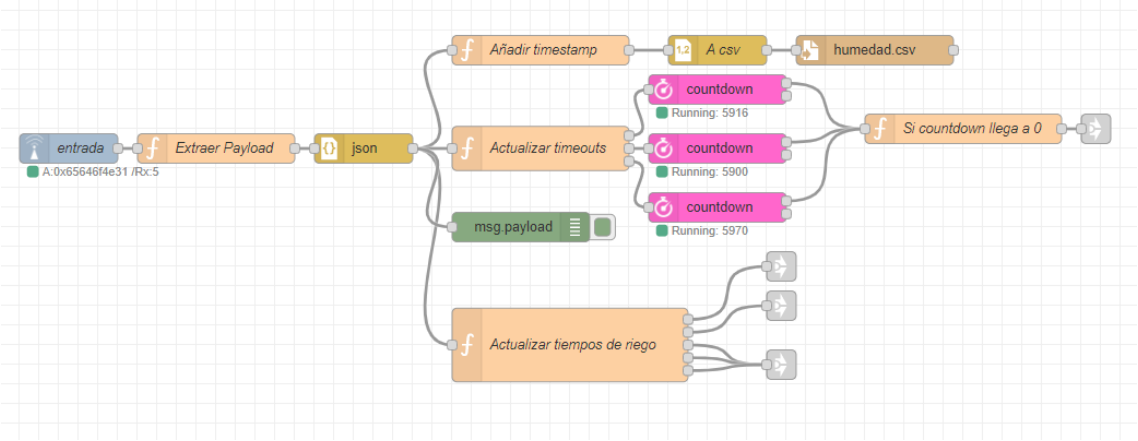

Figura 20: Recepción de mensajes por la antena

Otra ventaja a veces omitida, pero que sin duda tiene una gran importancia, es que gracias al estar basado en node.js, y al ser este un lenguaje interpretado, permite un desarrollo muy ágil ya que el retardo entre la pulsación del botón deploy a que se hayan aplicado los cambios es de meros milisegundos.

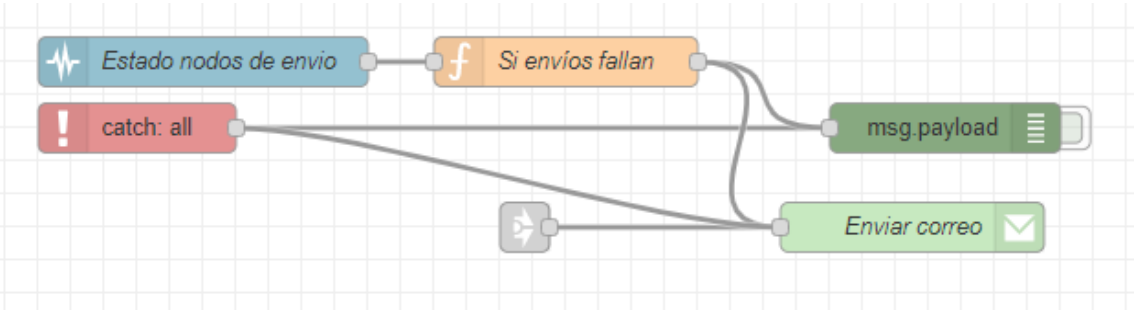

Figura 21: Aviso de fallos

Como se puede ver en las figuras, las líneas que unen los nodos permiten el flujo de datos de un nodo al siguiente, además, para proporcionar orden y limpieza visual, se cuenta con nodos de enlace, que permiten enlazar nodos de forma oculta, aun así, se puede mostrar dichos enlaces si se hace click sobre ellos.

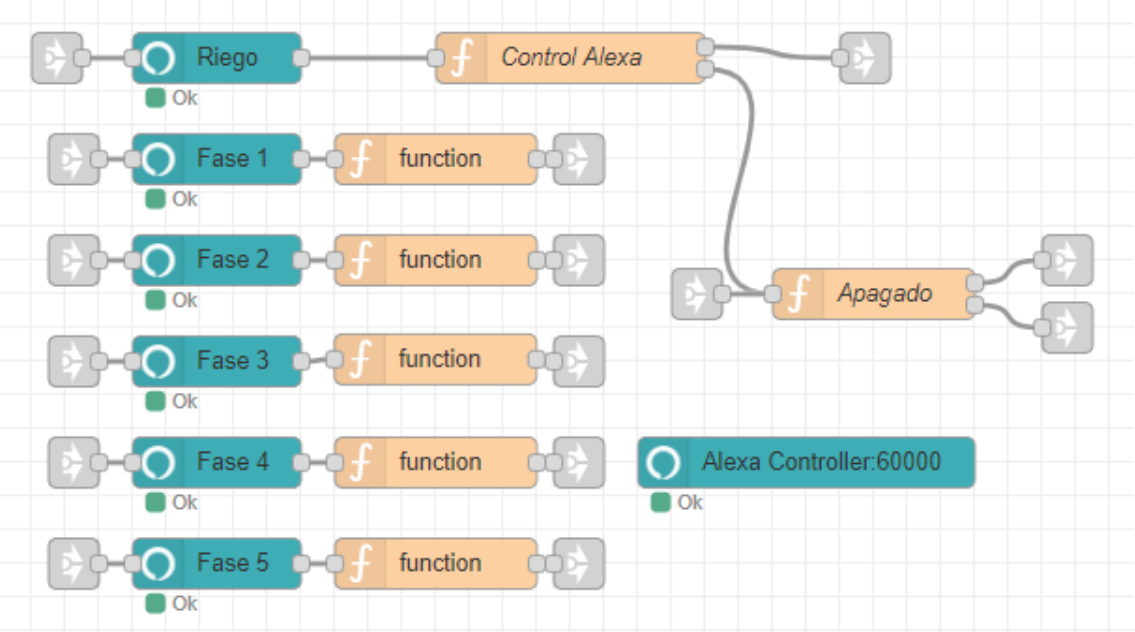

Figura 22: Control por Alexa

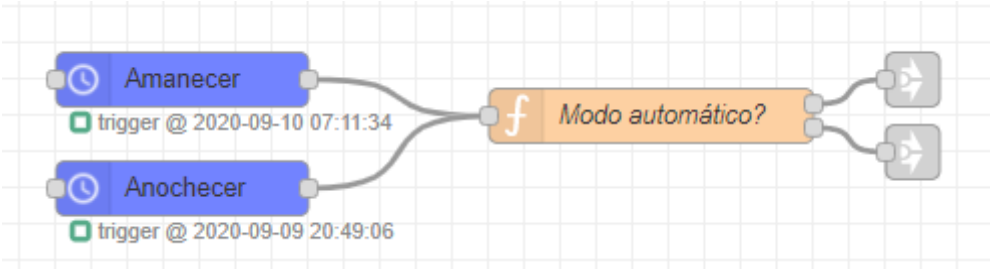

Figura 23: Iniciadores del modo automático

5.3.1.2 Nodos utilizados

Para la realización del trabajo se han utilizado algunos nodos proporcionados por Node-RED, así como nodos de función, a continuación, se analizan los nodos utilizados.

5.3.1.2.1 Nodos de función

Los nodos de función son la herramienta principal para el control de los flujos en Node-RED, se tratan de nodos con una entrada y una o múltiples salidas. A lo largo del trabajo se han utilizado varios nodos de función distintos, a continuación, se describe el funcionamiento de aquellos más importantes.

-Nodo “Tipo Mensaje”: este nodo se encuentra situado a la salida del endpoint http y tiene cuatro salidas que llevan a los siguientes nodos de función dependiendo de la cadena “Tipo” definida en el JSON.

-Nodo ”Tiempos”: Si el mensaje es del tipo “Tiempos”, se modificarán los temporizadores de cada fase de forma que coincidan con los de la solicitud. La recepción de un mensaje de este tipo indica el apagado del modo automático, esto se consigue modificando una variable global.

```
{
    "tipo": "Tiempos",
    "valores" :
    [
        {
            "fase": 1,
            "tiempo": 1000
        },
        {
            "fase": 2,
            "tiempo": 900
        },
        {
            "fase": 3,
            "tiempo": 700
        },
        {
            "fase": 4,
            "tiempo": 900
        },
        {
            "fase": 5,
            "tiempo": 900
        }
    ]
}
```

Figura 24: Ejemplo de mensaje de tiempos

```
var payload = msg.payload;
var returnMsg = [];
for (var elem of payload) {
    var fase = elem.fase -1;
    returnMsg[fase] = {topic:"control", payload:elem.tiempo};
}
globsal.set("auto", false);
return returnMsg;
```

Figura 25: Manejo de mensajes de tiempos

-Nodo “Control”: gracias a este nodo se puede activar o desactivar el modo automático dependiendo de la variable “Auto”. Si se apaga el modo automático, dejará de usarse el nivel de humedad del suelo para calcular el tiempo de riego y los temporizadores quedarán fijos hasta que lleguen mensajes tipo ”Tiempos” que los actualicen, o se reestablezca el modo automático. Su funcionamiento es muy simple, ya que simplemente asigna el valor del campo “auto” a una variable global del mismo nombre.

```
{
      "tipo": "Control",
      "auto": false
}
```

Figura 26: Mensaje de control

-Nodo “Riego”: Un mensaje de tipo “Riego” permite activar una o varias de las fases de riego. Adicionalmente se pueden definir los tiempos deseados para cada fase o dejarlos en blanco para usar los actuales. Esta llamada también desactiva el modo automático.

```
var payload = msg.payload;
var returnMsg = [];
var encendido = payload.encendido;
var primeraFase = 0;

if (encendido)
{
    for (var elem of payload)
    {
        var fase = elem.fase -1;
        returnMsg[fase] = {topic:"control",
payload:elem.tiempo};
        if (fase <= primeraFase)
        {
            primeraFase = fase;
        }
    }
    returnMsg[5] = {topic:"Fase"+(primeraFase+1),
payloadl:"on1"};
}
else
{
    returnMsg[5] = {payload:"offAll"};
}

globsal.set("auto", false);

return returnMsg;
```

Figura 27: Manejo de mensajes de riego

-Nodo “Estado”: Los mensajes de tipo “Estado” devuelven un objeto JSON indicando el estado actual del sistema de regado, indicando el tiempo de riego configurado para cada fase, así como la humedad detectada. Esto se consigue mediante un nodo template, que usando la sintaxis Mustache, permite generar objetos manualmente, como el JSON que nos interesa.

```
{
 "auto": {{ global.auto }},
 "hFase1": {{ global.hFase1 }},
 "hFase2": {{ global.hFase2 }},
 "hFase3": {{ global.hFase4 }},
 "hFase4": {{ global.hFase4 }},
 "hFase5": {{ global.hFase4 }},
 "tFase1": {{ global.tFase1 }},
 "tFase2": {{ global.tFase2 }},
 "tFase3": {{ global.tFase4 }},
 "tFase4": {{ global.tFase4 }},
 "tFase5": {{ global.tFase4 }},
}
```

Figura 28: Mensaje de respuesta con los estados

-Nodos “Añadir timestamp”, “A csv” y humedad.csv: El uso de estos tres nodos de forma conjunta permite guardar en un archivo csv los valores de humedad, junto con el numero de fase y marca de tiempo, de forma que pueden ser consultados con posterioridad e incluso crear una gráfica de seguimiento. Para no crear archivos demasiado grandes, y por tanto, difíciles de procesar, se ha agregado una tarea diaría de Cron, que permite obtener un fichero csv por día.

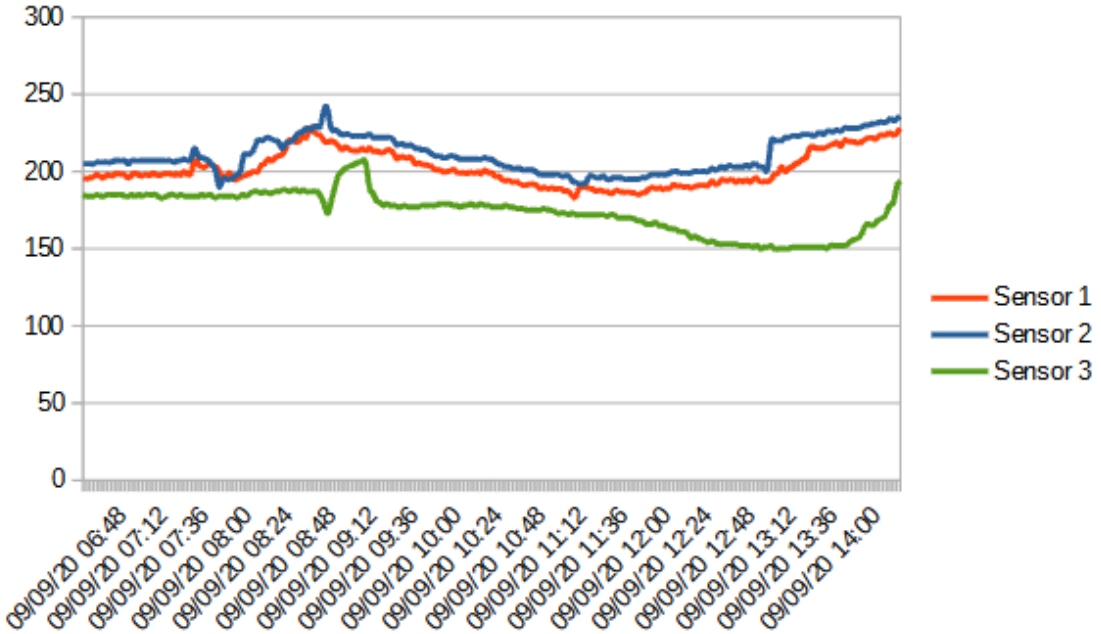

Figura 29: Gráfica con los datos de humedad durante 8 horas.

-Nodo “Actualizar temporizadores”: Este nodo se encuentra tras la entrada de la señal de radio. A su entrada recibe los estadosque cada Arduino manda con la humedad del suelo detectada, y los redirigea cada uno de los contadores de timeout para que sean reiniciados a 15 minutos cada vez que se recibe un mensaje de estado.

5.3.1.2.2 NODOS NRF24L01

Estos son los nodos que permiten la comunicación con los distintos Arduinos. Para que puedan funcionar primero es necesaria la creación de un nodo de configuración, que sirve para informar al hardware de la radio de los parámetros que se van a utilizar para la conexión.

```
[
    {
        "id": "577fa8ec.e4d398",
        "type": "RF24radio",
        "z": "",
        "name": "Radio",
        "ce": "17",
        "cs": "0",
        "palevel": "3",
        "datarate": "2",
        "channel": "0x76",
        "crclength": "2",
        "retriesdelay": "15",
        "retriescount": "15",
        "payloadsize": "32"
    }
]
```


Figura 30: Definición JSON del nodo de configuración de la antena

Los parámetros más importantes en el nodo de configuración son los siguientes:

- CE: Indica el pin de gpio de la raspberry pi que se encuentra conectado al pin ce (chip enable) del módulo nrf24l01.
- CS: Como la raspberry pi puede tener más de un módulo nrf24l01 conectado, se utiliza para seleccionar el módulo adecuado. Como en este proyecto solo se ha utilizado un único módulo, se deja a cero.
- Palevel: El nivel de potencia de la antena, siendo 0 el mínimo y 3 el máximo. Como la instalación se encuentra en un lugar con obstáculos, se ha seleccionado el nivel de transmisión máximo.
- Datarate: Indica la velocidad de transmisión de la antena. Se ha seleccionado la velocidad mínima ya que permite una mayor sensibilidad y por lo tanto, mayor distancia de transmisión.
- Channel: El canal utilizado para la transmisión, es conveniente seleccionar uno de forma aleatoria para impedir interferencias con otros usuarios.

Hay dos tipos de nodo nrf24l01 utilizables:

-Nodo de salida: Permiten el envío de datos hacia los clientes

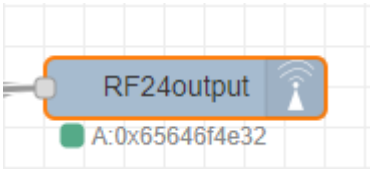

Figura 31

```
[
    {
        "id": "17430be0.4fde54",
        "type": "RF24input",
        "z": "60fd86d0.0c2ae",
        "name": "entrada",
        "topic": "nrf24",
        "radio": "577fa8ec.e4d398",
        "outputstring": true,
        "pipeaddress": "0x65646f4e31",
        "autoack": true,
        "x": 110,
        "y": 960,
        "wires": [
            [
                "ad1000ed.0f5c7"
            ]
        ]
    },
    {
        "id": "577fa8ec.e4d398",
        "type": "RF24radio",
        "z": "",
        "name": "Radio",
        "ce": "17",
        "cs": "0",
        "palevel": "3",
        "datarate": "2",
        "channel": "0x76",
        "crclength": "2",
        "retriesdelay": "15",
        "retriescount": "15",
        "payloadsize": "32"
    }
```

Figura 32: Definición JSON de un nodo de salida de la antena

-Nodo de entrada: Permiten la recepción de datos desde los clientes

Figura 33

```
[
    {
        "id": "17430be0.4fde54",
        "type": "RF24input",
        "z": "60fd86d0.0c2ae",
        "name": "entrada",
        "topic": "nrf24",
        "radio": "577fa8ec.e4d398",
        "outputstring": true,
        "pipeaddress":
"0x65646f4e31",
        "autoack": true,
        "x": 110,
        "y": 960,
        "wires": [
            [
                "ad1000ed.0f5c7"
            ]
        ]
    },
    {
        "id": "577fa8ec.e4d398",
        "type": "RF24radio",
        "z": "",
        "name": "Radio",
        "ce": "17",
        "cs": "0",
        "palevel": "3",
        "datarate": "2",
        "channel": "0x76",
        "crclength": "2",
        "retriesdelay": "15",
        "retriescount": "15",
        "payloadsize": "32"
    }
]
```

Figura 34: Definición JSON del nodo de entrada de la radio

5.4 Acceso externo

Como se puede deducir, para que el servidor sea capaz de recibir solicitudes HTTP a través de Internet, es necesario que sea accesible desde el puerto 1880. Esto supone un problema, ya que el único ISP (Proveedor de Servicios de Internet) disponible en la zona, utiliza CG-NAT (Carrier Grade Network Address Translation) en su red, lo que quiere decir que una IP pública se comparte entre varios clientes, por lo que el acceso remoto sería imposible sin una solución adicional.

Para subsanar este problema, se ha decidido configurar el servidor de forma que conecte a una VPN (Virtual Private Network) que dispone de una IP pública individual asignada por el ISP.

Una vez conectado el servidor a la VPN, se ha redirigido el puerto 1880 hacia él, de forma que es posible contactar con él desde el exterior, adicionalmente, esto hace posible la modificación de los flows a través de Internet.

5.5 Programación de las placas Arduino

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include <printf.h>
#include <ArduinoJson.h>

/****************** Configuracion ****************************/
/***       NumArduino debe ser un número del 1 al 5        ***/
int numArduino = 1;
uint64_t dirServidor = 0x65646f4e31;

/* Definicion del modulo nrf24l01 */
RF24 radio(9,10);
/*************************************************************/

long inicioTimeout;
long timeoutHumedad;
long timeoutModoStandAlone;
bool modoStandAlone;
bool regando;
int ultimaFaseRegada;
char mensajeRecibido[32];
char respuesta[32];
```

Figura 35

Se comienza definiendo las librerías a importar, así como algunas variables. Para que el programa funcione, es vital editar los valores en el apartado de configuración con los que se vayan a utilizar. De esta forma, se debe dar un valor a numArduino distinto para cada uno de los clientes de la instalación, mientras que la variable dirServidor ha de ser la dirección definida en el nodo de entrada de radio en Node-RED. La inicialización del módulo de radio permite configurar los puertos CE y CS a los que se ha conectado, en este caso el 9 y 10 respectivamente.

```
void setup()
{
  Serial.begin(9600);
  printf_begin();

  radio.begin();

  // Potencia maxima
  radio.setPALevel(RF24_PA_MAX);

  // Velocidad minima para tener mejor cobertura
  radio.setDataRate(RF24_250KBPS);

  //Seleccion de canal de comunicaciones
  radio.setChannel(0x76);

  // Definir las direcciones de envio y recepcion
  radio.openWritingPipe(dirServidor);
  //Las direcciones de los Arduinos son contiguas a la del
servidor
  radio.openReadingPipe(1, dirServidor + numArduino);

  // Se inicia la radio
  radio.startListening();
  radio.printDetails();

  //Inicializacion de los pines conectados a reles
  //Observar que una salida a nivel alto apaga el rele,
mientras que el nivel bajo lo enciende
  pinMode(2, OUTPUT);
  digitalWrite(2, HIGH);
  pinMode(3, OUTPUT);
  digitalWrite(3, HIGH);
  pinMode(4, OUTPUT);
  digitalWrite(4, HIGH);

  //Inicializacion de otras variables
  inicioTimeout = micros();
  regando = false;
  timeoutHumedad = micros();

  timeoutModoStandAlone = micros();
  ultimaFaseRegada = 0;
}
```

Figura 36

En el setup de las placas Arduino se inicializan las variables y el módulo de radio, hay que destacar que es necesario utilizar el mismo canal y velocidad que los definidos en Node-RED, de lo contrario no se podrá establecer la comunicación.

```
void loop()
{

  if (radio.available())// Si se recibe un mensaje
  {

    //Leer mensaje recibido
    radio.read(&amp;mensajeRecibido, sizeof(mensajeRecibido));
    //Reiniciar timeout modo stand alone
    timeoutModoStandAlone = micros();
    modoStandAlone = false;


    Serial.print(mensajeRecibido);

    //Dejar de escuchar para responder
    radio.stopListening();
    //Escribir respuesta
    radio.write(mensajeRecibido, sizeof(mensajeRecibido));
    //Reiniciar la escucha
    radio.startListening();
    char on1[] = "on1";
    char on2[] = "on2";
    char on3[] = "on3";
    char off[] = "off";
```

Figura 37

El loop se ejecuta en bucle una vez se ha completado la parte de setup, aquí es donde se procesarán los mensajes recibidos y enviados para que el riego se lleve a cabo. Se comienza comprobando si se ha recibido algún mensaje con radio.available(), en caso positivo se entra a la parte del código que procesa un mensaje recibido.

Si el mensaje recibido coincide con mensaje de encendido o de apagado, se actuará en consecuencia.

```
    if (strcmp(mensajeRecibido, on1) == 0 ||
strcmp(mensajeRecibido, on2) == 0 || strcmp(mensajeRecibido,
on3) == 0)
    {

      inicioTimeout = micros();
      regando = 1;
      if (strcmp(mensajeRecibido, on1) == 0)
      {
        controlFases(true,false,false);
      }
      if (strcmp(mensajeRecibido, on2) == 0)
      {
        controlFases(false,true,false);
      }
      if (strcmp(mensajeRecibido, on3) == 0)
      {
        controlFases(false,false,true);
      }
    }
    else if (strcmp(mensajeRecibido, off) == 0)
    {
      regando = 0;
      controlFases(false,false,false);
    }
  } //Fin if (radio.available())
```

Figura 38

Cada dos minutos se reporta la humedad detectada por el sensor conectado a esta placa Arduino, de forma que se observa un temporizador para determinar cuándo se enviará el mensaje.

```
  //Si han pasado dos minutos desde que se reporto la humedad
por ultima vez
  if (!radio.available() && (micros() - timeoutHumedad) >=
120000000)
  {
    //Reiniciar timeout
    timeoutHumedad = micros();
    radio.stopListening();
    int humedad;

    //Definir capacidad de json para 4 objetos
    const size_t capacity = JSON_OBJECT_SIZE(4);
    DynamicJsonDocument doc(capacity);
    String estado;
    humedad = analogRead(A2);

    doc["t"] = 0; //Tipo de mensaje
    doc["e"] = regando; //Estado regando/espera
    doc["f"] = numArduino; //Numero del Arduino
    doc["h"] = humedad; // Humedad detectada
    serializeJson(doc, estado);
    char array[estado.length()];
    strcpy(array, estado.c_str());
    bool ack = radio.write(array, sizeof(array));
    radio.startListening();
    Serial.print(ack);

    if (ack)
    {

      timeoutModoStandAlone = micros();
      modoStandAlone = false;
    }
  }
```

Figura 39

Si ha pasado más de media hora desde que se recibió el mensaje de inicio, se entiende que se ha perdido la conexión con el servidor, por lo que se detendrá el riego.

```
  //Si ha pasado mas de media hora desde el inicio, cortar
el riego
  if (micros() - inicioTimeout >= 1800000000 &amp;amp;&amp;amp;
regando)
  {
    regando = false;
    controlFases(false,false,false);
  }

  //Si pasa un tiempo aleatorio entre 11 y 14 horas, activar
el modo standalone
  if (micros() - timeoutModoStandAlone >= (rand() % (50400 -
39600 + 1)) + 39600)
  {
    modoStandAlone = true;
  }
```

Figura 40

Se ha programado también una rutina que permite que las placas Arduino tomen el control del riego cuando no han podido establecer conexión con el servidor durante más de 12 horas.

```
if (modoStandAlone && ((micros() -
inicioTimeout >= 1800000000) || ultimaFaseRegada
== 0))
  {
    switch (ultimaFaseRegada)
    {
    case 0:
      controlFases(true,false,false);
      inicioTimeout = micros();
      regando = true;
      ultimaFaseRegada = 1;
      break;
    case 1:
      controlFases(false,true,false);
      inicioTimeout = micros();
      regando = true;
      ultimaFaseRegada = 2;
      break;
    case 2:
      controlFases(false,false,true);
      inicioTimeout = micros();
      regando = true;
      ultimaFaseRegada = 3;
      break;
    case 3:
      controlFases(false,false,false);
      inicioTimeout = micros();
      regando = false;
      ultimaFaseRegada = 0;
      modoStandAlone = false;
      break;
    default:
      break;
    }
  }
} // Fin Loop
```

Figura 41

Pequeña función utilizada para facilitar el control de cada fase

```
void controlFases (boolean fase1, boolean fase2, boolean
fase3)
{
      digitalWrite(2, fase1?LOW:HIGH);
      digitalWrite(3, fase2?LOW:HIGH);
      digitalWrite(4, fase3?LOW:HIGH);
}
```

Figura 42

Capítulo 6. Resultados

6.1 Costes

Uno de los objetivos principales de este trabajo era que la instalación tuviera un coste económico, a continuación, se proporciona un desglose de precios para comprobar si se ha conseguido:

- Raspberry Pi - Desde 10 €.
- Arduino Nano – Desde 1 €, tres unidades 3 €.
- Módulos NRF24L01 – Desde 1 €, cuatro unidades 4 €.
- Módulos YL-69 – Desde 2€. Tres unidades 6 €
- Relés electromagnéticos individuales – Desde 2 €, dos unidades 4 €.
- Relé electromagnético triple – Desde 2 €.
- Electroválvulas – Desde 10€, cinco unidades 50€.
- Transformadores 24V – Desde 5€, tres unidades 15€.

Como se puede observar, el precio se encuentra por debajo de los 100€, resultando más económico que muchas de las soluciones disponibles para comprar. Además, los elementos de mayor coste han sido las electroválvulas, que habrían sido necesarias igualmente incluso en el caso de adquirir un sistema comercial, por lo que el ahorro se hace evidente.

6.1.1 Funcionalidades

Como ya se ha explicado, mediante el modo automático, la instalación permite un uso desatendido durante largos periodos de tiempo, así como hacer un seguimiento del estado de la instalación.

A su vez, gracias a la conexión con servicios externos, se ha conseguido un altísimo grado de personalización. Con Alexa, se consigue un control manual de la instalación por voz, además, Alexa también dispone de un sistema de rutinas, por lo que es posible añadir nuevas funcionalidades.

A continuación, se muestran ejemplos de algunas de las funciones que se pueden implementar gracias a IFTTT webhooks, tanto como trigger, así como acción.

6.1.1.1 Bloquear riego en caso de previsión de lluvia

Entre los múltiples servicios disponibles en IFTTT, se encuentra Weather Underground, que permite el aviso de las condiciones metereológicas, por lo que será ideal para este ejemplo.

Crear un nuevo applet es tan sencillo como pulsar el botón “Create” en la página principal de IFTTT. Seleccionando weather underground como “this”, podemos llegar a la siguiente ventana de configuración.

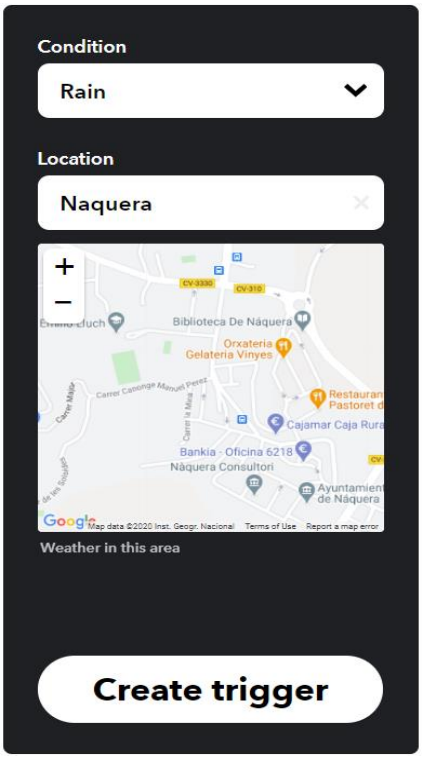

Figura 43

Cuando se confirmen los datos introducidos, se vuelve a la pantalla principal y se elige como "That" la opción "Webhooks".

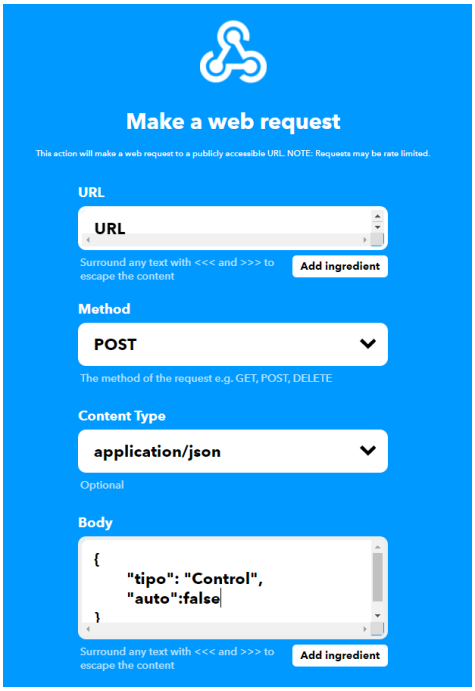

Figura 44

Gracias a este sencillo applet, se consigue que cuando exista previsión de lluvia, se pase la instalación al modo manual, por lo que el riego se detendrá hasta que se inicie por algún método, o se restaure el modo automático.

6.1.1.2 Notificación con la información de los sensores

Para poder recibir automáticamente notificaciones sobre el estado de los sensores, se ha definido un applet que recibe una solicitud http con el estado de los sensores en el cuerpo, que al ser recibido por IFTTT, se convierte en una notificación de Android.

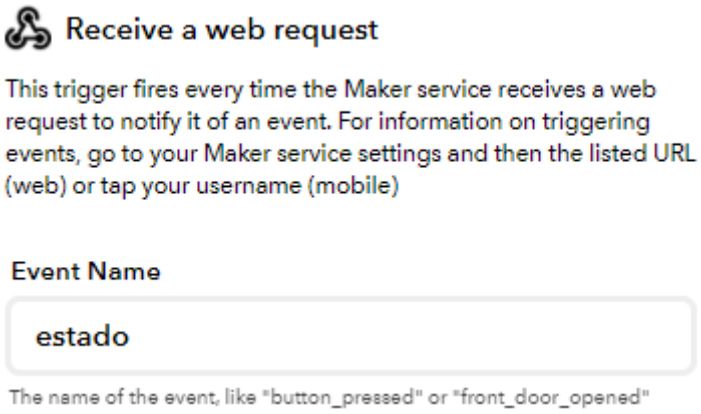

Figura 45

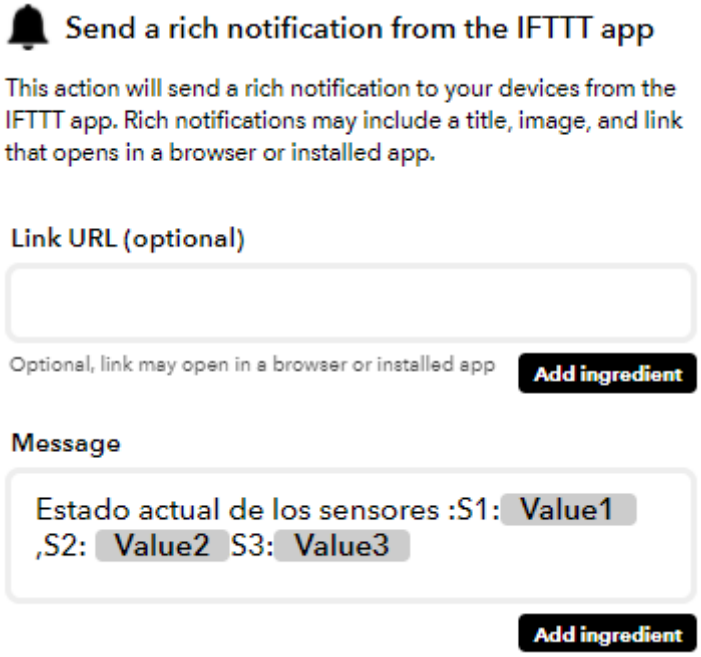

Figura 46

A continuación, se crea un pequeño flujo en Node-RED que permite la creación de la solicitud.

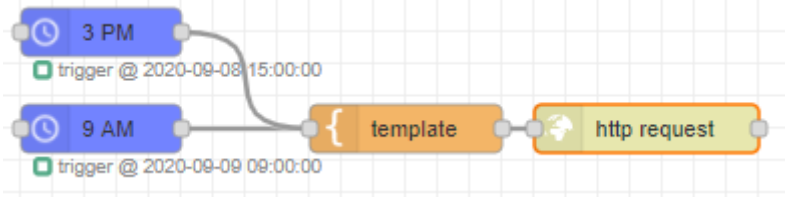

Figura 47

En el nodo template tan solo se ha tenido que definir un JSON con los valores a transmitir y en el nodo http se ha definido la siguiente URL para hacer una solicitud POST:

https://maker.ifttt.com/trigger/estado/with/key/*key_api_IFTTT*

6.1.2 Seguridad

Node-red cuenta con permite asegurar los flujos y las entradas http de forma que un intruso no pueda tomar control de la instalación del usuario.

Para la creación del hash de la contraseña, basta con ejecutar el comando

node-red admin hash-pw

Este script solicitará una contraseña y calculará su hash mediante bcrypt.

El siguiente paso es definir dicho hash en el archivo settings.json, para ello es necesario descomentar el apartado adminAuth, y editar los campos de forma que reflejen el nombre de usuario y hash de contraseña que el usuario desee.

De forma similar, se pueden asegurar los nodos http descomentando y editando la línea con el parámetro httpNodeAuth en el archivo settings.json.

De esta forma cuando se deseen usar los nodos http se deberá proveer a la request de los campos de autenticación básica para que la solicitud pueda ser atendida.

En IFTTT webhooks, habrá que proporcionar dichos parámetros de autenticación básica de la siguiente forma:

https://usuario:contraseña@url

Para que los módulos NRF24L01 admitan un mensaje, es necesario que el emisor esté usando el mismo canal, la misma velocidad y que se haya introducido una dirección correcta, además, como ninguno de estos campos se transmiten en texto plano en los mensajes, se considera que no es necesario aplicar seguridad adicional sobre los mensajes enviados por las radios NRF24L01.

6.1.3 Seguridad física

La Raspberry Pi que actúa de servidor se encuentra en interior y bajo llave, por lo que no es necesario realizar ningún trabajo adicional sobre ella.

Ya que algunas de las placas Arduino se encuentran al exterior, se han instalado dentro de cajas estancas de forma que se encuentren protegidas de los elementos, además, se ha usado resina epoxi en algunas partes que puedan tener componentes electrónicos expuestos al agua de riego, como los detectores de humedad. La resina epoxi también ha sido utilizada para asegurar correctamente algunos cables y así impedir que puedan ser desconectados accidentalmente.

Capítulo 7. Conclusiones y desarrollo futuro

7.1.1 Objetivos alcanzados

A lo largo de este trabajo se han detallado una serie de requisitos motivados por la necesidad de crear un sistema de regado con conexión a Internet. Estos requisitos se han visto cubiertos gracias al uso del hardware y software elegidos, pudiendo realizar todas las funciones que eran necesarias.

Se ha aprendido a manejar node-red así como el gestor de paquetes npm, Visual Studio Code, y se ha profundizado sobre los conocimientos de Arduino y JavaScript, todas estas herramientas han demostrado ser de gran utilidad a la hora de realizar el proyecto.

Durante el desarrollo se han encontrado algunas dificultades, sobre todo en el lado de node-red, ya que, al tratarse de una plataforma tan nueva, no se encuentra tan documentada como otras más longevas. Otros problemas han sido causados por su propia naturaleza, y es que, siendo una plataforma de programación basada en flujos, hace difícil la creación de programas genéricos que permitan la reutilización de código, por ejemplo, es necesario utilizar un nodo de salida de antena para cada dispositivo, ya que no es posible cambiar la dirección a la que se envían en tiempo de ejecución.

7.2 Trabajo futuro

Pese a que se han logrado los objetivos marcados, este proyecto es ampliamente extensible, sobre todo en la parte referida a la comunicación entre dispositivos.

Existe una versión de la librería de NRF24L01 llamada nrf24l01mesh, que como su nombre indica, permite el uso de comunicación en malla a los dispositivos. Puede ser muy interesante desarrollarlo en este sentido ya que permitiría un nivel de robustez aún mayor en caso de una caída del servidor en la Raspberry Pi. Por desgracia, esta versión de la librería no se encuentra disponible aún para node-red, por lo que habría que estudiar como solventar ese problema.

Se podría estudiar el uso de ZigBee, que pese a tener un precio muy superior, admite una gran cantidad de dispositivos simultáneos conectados en malla, así como una señal muy robusta.

Otro apartado por el que se podría extender es la creación de una aplicación para Android que permita la configuración de la instalación de forma visual, y sin tener que depender únicamente de servicios de terceros.

Finalmente, se podría estudiar su uso con otros asistentes de voz, como Google Assistant, Siri de Apple u otras soluciones libres como Home Assistant.

Capítulo 8. Bibliografía

[1][2] Raspberry Pi 2 Model B – Raspberry Pi https://www.raspberrypi.org/products/raspberry-pi-2-model-b [Online].

[3] Arduino Uno Rev3 https://store.arduino.cc/arduino-uno-rev3 [Online].

[4][5] NRF24L01 Datasheet

https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf [Online]. p. 24

[6] Comunicación inalámbrica a 2.4Ghz con Arduino y NRF24L01

https://www.luisllamas.es/comunicacion-inalambrica-a-2-4ghz-con-arduino-y-nrf24l01/ [Online]

[7] NRF24L01 Datasheet

https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf [Online]. p. 8

[8] NRF24L01 Datasheet

https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf [Online]. p. 14

[9] Getting Started : Node-RED https://nodered.org/docs/getting-started/ [Online]

[10] Running on Raspberry Pi : Node-RED https://nodered.org/docs/getting-started/raspberrypi [Online]