ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
INGENIERÍA DE COMPUTADORES

SISTEMA DE RIEGO AUTOMATIZADO Y DISTRIBUIDO CON CONTROL WEB

Realizado por

FRANCISCO JAVIER SOLÍS FRANCO

Dirigido por

ÁNGEL FRANCISCO JIMÉNEZ FERNÁNDEZ

ELENA CEREZUELA ESCUDERO

Departamento

ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

Sevilla, Junio de 2017

Índice General

Índice de Figuras

1. Introducción y Objetivos.

1.1. Introducción.

Hoy día muchos pequeños agricultores no tienen la posibilidad de controlar sus producciones y consumos de agua de forma remota ni la posibilidad de ver las temperatura y humedades de forma gráfica para compararlas con otros días, semanas o años. Además de que los controles de los riegos se realizan de forma manual, es decir, acudiendo al terreno para activar las bombas de agua o activar los temporizadores manualmente, como se observa en la **ilustración 1**.

No llevar el control de los riegos y tener que desplazarse para ello, implica un impacto medio ambiental en el consumo de agua de pozos naturales de los que disponen los agricultores y la emisión de gases en el traslado a sus terrenos. Además de que se puede provocar un exceso de riego en caso de lluvia o haber programado más tiempo del que se necesita o se desea por un mal cálculo.

El no tener el control del riego junto con el cambio climático que en muchos casos provoca sequías, implica que se haga un consumo excesivo de los recursos naturales provocando un problema para la sostenibilidad del medio ambiente, es decir, cada vez más las altas temperaturas y las pocas precipitaciones que se van sucediendo por el cambio climático provoca que pozos naturales, de los cuales los agricultores obtiene el agua, se vayan secando, por lo que si no se tiene el control, podemos consumir más agua de la que necesitamos para el riego, afectando directamente al medio ambiente.

Mientras que si tenemos el control y además el sistema está automatizado, si consideramos que por ejemplo necesitamos 3h de riego y lo programamos así, el sistema controlará la humedad de tal forma que, si a las 2h de riego el suelo ya está lo suficientemente húmedo el sistema cortará el riego evitando el consumo excesivo de los recursos. De esta forma evitamos los traslados innecesarios y la emisión de gases que realizan los vehículos, reduciendo estas emisiones, también contribuimos a reducir el cambio climático y en definitiva el efecto invernadero.

Ilustración 1: Programador de riego automático

Plantear un sistema remoto, nos da la posibilidad de entrar en el IoT (Internet of things, el internet de las cosas), conectando a internet en este caso el sistema de riego para obtener las estadísticas que vamos generando a través de los sensores en el terreno, además de permitirnos identificar el sistema y poder llevar el control, en este caso de consumo de agua a un nivel global, de forma que se puedan tomar mejores medidas para favorecer el crecimiento y explotación de los recursos, como por ejemplo, podríamos saber cuántos litros se consume en España de agua para el regadío y que porcentaje de agricultores usan pozos naturales y cuantos usan depósitos artificiales.

1.2. Objetivos.

El objetivo primario de este Trabajo Fin de Grado es ofrecer un sistema empotrado y distribuido que dé la posibilidad de programar un riego automático y obteniendo datos como la temperatura, humedad, litros que se está consumiendo y periodos de lluvia, mostrando gráficas estadísticas mediante una interfaz web, dando la posibilidad al agricultor de ver cómo puede aumentar su producción haciendo uso de plataformas libres.

Este sistema no es aplicable únicamente a los agricultores, sino también a investigadores que quieran comprobar ritmos de crecimientos o ver evoluciones aplicando diferentes programaciones.

Como objetivos secundarios para cumplir los objetivos primarios tenemos:

- Implementar un controlador de riego en una plataforma de la familia **Raspberry Pi**.

- Desarrollar un **firmware para Arduino** que permita la comunicación con la Raspberry Pi para realizar interfaces con sensores analógicos y aumentar el número de entradas/salidas de propósito general.

- Diseñar una **base de datos** para almacenar las programaciones, así como las estadísticas y circunstancias del sistema.

- Realizar una **página web**, así como una **API**, para el control, configuración y monitorización del sistema, ya sea a través de la propia web o una aplicación para Android.

1.3. Estado del Arte.

En el mercado actual, los sistemas de riegos automáticos más asequibles sólo permiten una programación básica y local, lo cual en caso de fallo se perdería la programación, con el consiguiente trabajo de volver a programar todo. Ese proyecto solventa este problema ya que los datos se almacenan en un servidor externo, lo que nos da la seguridad en caso de fallo de sustituir solamente el hardware sin tener que volverlo a programar.

Como ejemplo tenemos el “IESP4MEEUR” de la empresa RainBird, que permite crear 4 programas individuales con horas de inicio independientes por programa hasta un total de 24 horas de inicio, cuenta con diagnóstico avanzado y detección de cortocircuitos con alerta LED y calculadora de tiempo total de funcionamiento por programa, que tiene un precio aproximado de 140,90€. Incluso hay versiones mucho más baratas que consisten en un único programador horario, el cual solo activa la corriente en las horas indicadas sin control ninguno.

Estos productos básicos y más asequibles solo controlan el tiempo de riego, si admiten la programación por sectores, pero no controlan el riego en caso de lluvia o de que el suelo ya esté lo suficientemente húmedo, ni dan información acerca de la temperatura y humedad para que el usuario sepa las condiciones óptimas.

El uso de plataformas libres para este proyecto, implica un abaratamiento del coste a desembolsar por el usuario, ya que los sensores, actuadores y plataformas usados se encuentran de forma fácil y barata en webs de venta de material electrónico. Además de que el uso de estas plataformas está apoyado por comunidades grandes de usuarios que cuentan con muchos ejemplos y soluciones para los problemas que puedan surgir.

Apostar por el IoT, es apostar por una tecnología emergente que está creciendo y está empezando a formar parte de la vida cotidiana, por lo que incorporarlo al proyecto nos garantiza que el sistema no quede anticuado en poco tiempo, esta tecnología ya la podemos encontrar en neveras, lámparas y/o termostatos de las casas, nos ofrece la posibilidad de saber los productos que tenemos en la nevera para poder realizar la compra o encender las lámparas remotamente desde cualquier parte del mundo, en nuestro caso nos ofrece la posibilidad de controlar el consumo del sector de la agricultura, facilitando la labor de obtener datos para estadísticas.

En resumen:

- Almacenamiento local de la programación frente al almacenamiento distribuido.
- Control único del tiempo de riego frente a la programación del riego con la obtención de datos para la optimización.

- Plataforma cerrada frente plataformas abiertas apoyadas por comunidades.
- Distribuidor único frente a muchos distribuidores y webs.
- Apuesta por tecnología emergente.
- Mayor control de los consumos de recursos.

1.4. Tecnología utilizada.

1.4.1. Arduino.

Se trata de una plataforma de hardware libre (Logo de la plataforma en la ilustración 2) que consiste en una placa de circuito impreso con un **microcontrolador** (usualmente Atmel AVR), puerto digitales y analógicos de **entrada/salida** y un **conversor A/D**, a la cual se le pueden conectar placas de expansión llamadas Shields.

Con respecto al software de la plataforma, cuenta con un entorno de desarrollo (IDE) basado en Processing y su lenguaje de programación está basado en Wiring (en C/C++).

Ilustración 2: Logo Arduino

1.4.1.1. Processing.

Es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. Fue iniciado por Ben Fry y Casey Reas, ambos miembros de Aesthetics and Computation Group del MIT Media Lab dirigido por John Maeda.

El lenguaje de Processing se basa en Java, aunque hace uso de una sintaxis simplificada y de un modelo de programación de gráficos.

1.4.1.2. Wiring.

Es una plataforma de prototipado electrónico de fuente abierta compuesta de un lenguaje de programación, un entorno de desarrollo integrado (IDE), y un microcontrolador. Ha sido desarrollado desde 2003 por Hernando Barragán.

El IDE de Wiring viene con una librería de C/C++ llamada "Wiring", la cual hace operaciones comunes de input/output mucho más fácil. Los programas de Wiring están escritos en C/C++, pese a que sus usuarios sólo necesiten definir dos funciones para hacer un programa ejecutable:

- **setup()** – una función ejecutada sólo una vez en el inicio de un programa la cual puede ser usada para definir los ajustes iniciales de un entorno.

- **loop()** – una función llamada repetidamente hasta que la tarjeta es apagada.

1.4.2. Raspberry Pi 2.

Es un computador de placa simple (Ilustración 3), SBC según sus siglas en inglés (Single Board Computer), de bajo coste desarrollado en Reino unido que cuenta con un System-on-Board-Chip que contiene el procesador ARM (CPU), el procesador gráfico (GPU) y la memoria principal (RAM). No contiene disco duro o disco de estado sólido, ya que el almacenamiento permanente se lleva a cabo con una tarjeta microSD.

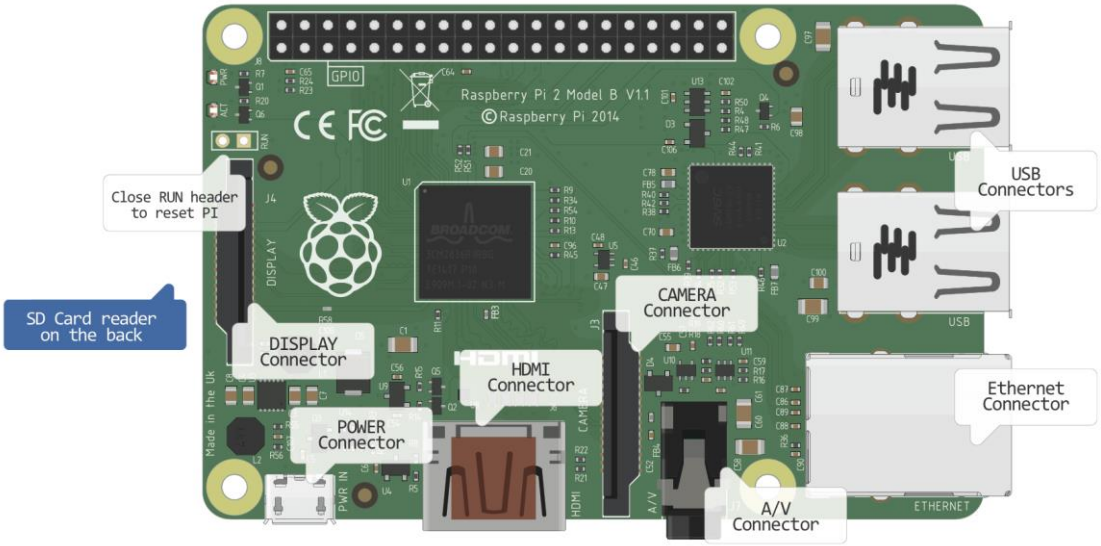

Ilustración 3: Raspberry Pi 2

Como características principales cuenta con:

- **SoC:** Broadcom BCM2836 (CPU + GPU + DSP + SDRAM + Puerto USB).
- **CPU:** 900 MHz quad-core ARM Cortex A7.
- **Juego de Instrucciones:** RISC de 32 bits.
- **GPU:** Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia),1080p30 H.264/MPEG-4 AVC.
- **Memoria (SDRAM):** 1 GB (compartidos con la GPU).
- **Puertos USB 2.0:** 4.
- **Entradas de vídeo:** Conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la RPF.
- **Salidas de vídeo:** Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4), Interfaz DSI para panel LCD.

- **Salidas de audio:** Conector de 3.5 mm, HDMI.
- **Almacenamiento:** MicroSD.
- **Conectividad de Red:** 10/100 Ethernet (RJ-45) via hub USB.
- **Fuente de alimentación:** 5 V vía Micro USB o GPIO header.
- **Consumo energético:** 800mA (4.0 W).

1.4.3. Actuadores y sensores.

Son elementos electrónicos que recogen o realizan una acción de acuerdo a unos estímulos del entorno que los rodea o a través de una señal, como relés, displays, higrómetros, sensor de temperatura, etc… (Se definirán en detalle en el apartado de diseño hardware).

1.4.4. Python.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa, y en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma, su logo puede verse en la ilustración 4.

La filosofía de este lenguaje hace hincapié en una sintaxis de código legible. Posee una licencia de código abierto que es compatible con GNU.

Ilustración 4: Logo Python

Python es un lenguaje de programación **multiparadigma**. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Python usa **tipado dinámico** y **conteo de referencias** para la administración de memoria.

Una característica importante de Python es la **resolución dinámica de nombres**; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en **C o C++**. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

1.4.5. MySQL.

Es un sistema de gestión de base de datos relacional open source desarrollado por Oracle y está considerada la base de datos más popular del mundo y su Lenguaje se basa en SQL, su logo es muy conocido y puede verse en la ilustración 5.

Ilustración 5: Logo MySQL

La versión usada en el proyecto es la 5.7 que cuenta como características principales:

- Amplio subconjunto del **lenguaje SQL**. Algunas extensiones son incluidas igualmente.
- **Disponibilidad** en gran cantidad de plataformas y sistemas.
- Posibilidad de **selección de mecanismos de almacenamiento** que ofrecen diferentes velocidades de operación, soporte físico, capacidad, distribución geográfica, transacciones...
- **Transacciones y claves foráneas.**
- **Conectividad segura.**
- **Replicación.**
- **Búsqueda e indexación de campos de texto.**

1.4.6. CodeIgniter.

CodeIgniter es un framework para el desarrollo de aplicaciones en **PHP** (véase el logo en la ilustración 6) que utiliza el MVC (Modelo Vista Controlador) que permite a los desarrolladores Web mejorar la forma de trabajar y hacerlo con mayor velocidad.

Al tratarse de un framework para desarrollo Web admite el uso de **HTML**, **CSS** (**Boostrap** en el caso de este proyecto, que se trata de una librería CSS ya implementada) y **JavaScript**.

Ilustración 6: Logo CodeIgniter

Sus características son:

- **Versatilidad:** Quizás la característica principal de CodeIgniter, en comparación con otros frameworks PHP. CodeIgniter es capaz de trabajar la mayoría de los entornos o servidores, incluso en sistemas de alojamiento compartido, donde sólo tenemos un acceso por FTP para enviar los archivos al servidor y donde no tenemos acceso a su configuración.

- **Facilidad de instalación:** No es necesario más que una cuenta de FTP para subir CodeIgniter al servidor y su configuración se realiza con apenas la edición de un archivo, donde debemos escribir cosas como el acceso a la base de datos. Durante la configuración no necesitaremos acceso a herramientas como la línea de comandos, que no suelen estar disponibles en todos los alojamientos.

- **Flexibilidad:** CodeIgniter es bastante menos rígido que otros frameworks. Define una manera de trabajar específica, pero en muchos de los casos podemos seguirla o no y sus reglas de codificación muchas veces nos las podemos saltar para trabajar como más a gusto nos encontremos. Algunos módulos como el uso de plantillas son totalmente opcionales. Esto ayuda muchas veces también a que la curva de aprendizaje sea más sencilla al principio.

- **Ligereza:** El núcleo de CodeIgniter es bastante ligero, lo que permite que el servidor no se sobrecargue interpretando o ejecutando grandes porciones de código. La mayoría de los módulos o clases que ofrece se pueden cargar de manera opcional, sólo cuando se van a utilizar realmente.

- **Documentación tutorializada:** La documentación de CodeIgniter es fácil de seguir y de asimilar, porque está escrita en modo de tutorial. Esto nos facilita mucho la referencia rápida, cuando ya sabemos acerca del framework y queremos consultar sobre una función o un método en concreto, pero para iniciarnos sin duda se agradece mucho.

1.4.7. Apache.

Es un servidor web HTTP de código abierto para plataformas Unix, Windows, Macintosh y otras. Es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation (logo en la ilustración 7) dentro del proyecto HTTP Server (httpd).

Ilustración 7: Logo Apache

Sus ventajas como servidor web son:

- **Modular.**

- **Código abierto.**

- **Multi-plataforma.**

- **Extensible.**

- **Popular** (fácil conseguir ayuda/soporte).

1.4.8. Raspbian.

Es una distribución del sistema operativo GNU/Linux y por tanto libre, basada en Debian (ilustración 8), y optimizada para la SBC Raspberry Pi.

Actualmente la versión de esta distribución se basa en Debian 8.0 (Debian Jessie) y cuenta con el entorno de escritorio PIXEL, además de incorporar VNC y SSH para el control remoto.

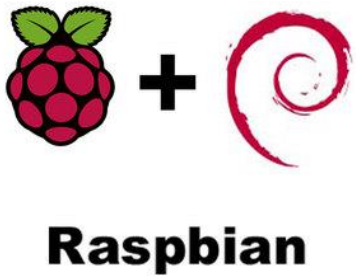

Ilustración 8: Logo Raspbian

Sus características:

- **Núcleo:** Linux.
- **Kernel:** 4.4
- **Tipo de núcleo:** monolítico.
- **Interfaz:** PIXEL.
- **Plataforma soportada:** ARMv6.
- **Gestión de paquetes:** dpkg.
- **Licencia:** GPL y otras licencias libres.

1.4.9. Eclipse.

Es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma al que se le añaden plugins para programas en java por ejemplo (Java Development Tools, JDT), cuyo logo es reconocible como puede verse en la ilustración 9, en el caso de este proyecto se añadió el plugin de Python. En concreto la versión usada de eclipse es la NEON 2.

Ilustración 9: Logo de Eclipse

1.4.10. MySQL WorkBench.

Es una herramienta visual de diseño de base de datos que integra desarrollo software, Administración de base de datos, diseño de base de datos, creación y mantenimiento para el sistema de base de datos MySQL, su logo se trata del mostrado en la ilustración 10.

Ilustración 10: Logo MySQL WorkBench

2. Arquitectura del Sistema.

La arquitectura del sistema (ilustración 11) se basa en un modelo cliente-servidor, y como puede verse en la siguiente figura, Raspberry Pi, Arduino y el usuario constituyen los clientes, mientras que la web junto a la API y la base de datos estarán alojados en un servidor, al cual se accederá mediante llamadas http e inserciones y consultas a la base de datos.

Ilustración 11: Diagrama de la Arquitectura

En la parte de cliente, solo se encuentran los controles de los sensores y llamadas a los datos de configuración e inserción de los datos recogidos. En la parte servidor, nos encontramos con la base de datos y la API junto a la interfaz web, que únicamente realizan el almacenamiento de los datos, su procesamiento y visualización, podemos ver un diagrama completo de la arquitectura en la ilustración 12.

La ventaja de usar este modelo es que, al no tener la base de datos local, pudiendo esta ser distribuida, en caso de fallo del sistema empotrado los datos y en definitiva las estadísticas, estarían a salvo. Básicamente la comunicación se establece a través de la base de datos.

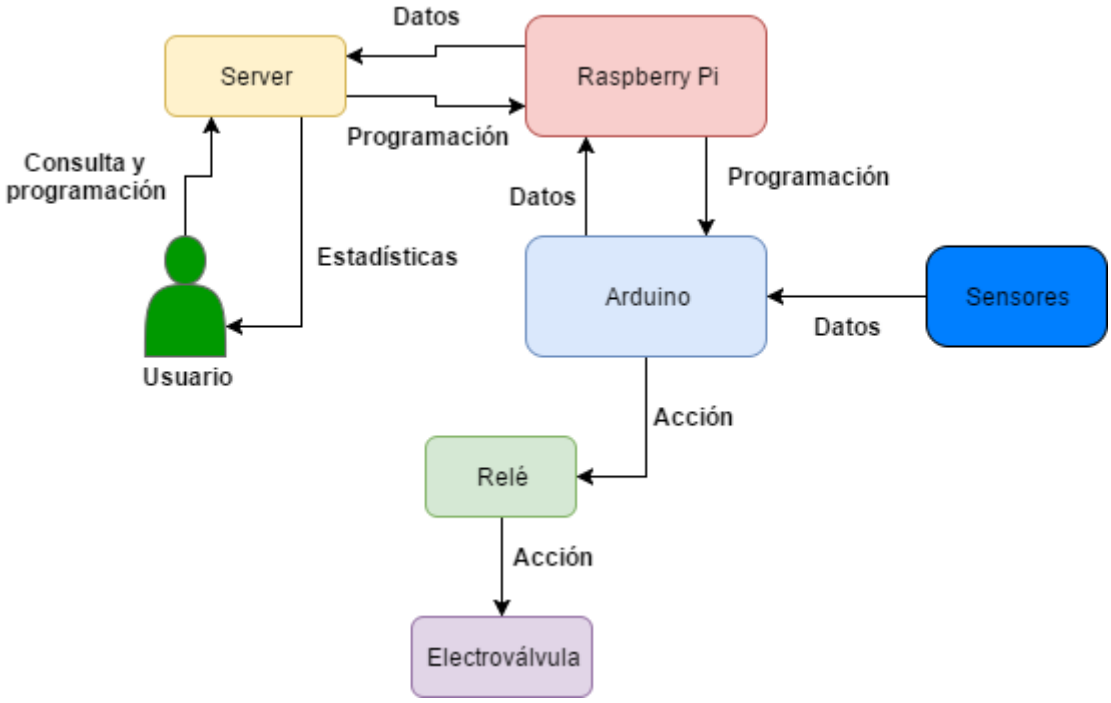

Ilustración 12: Diagrama completo

3. Diseño Hardware.

El esquema hardware está compuesto por dos plataformas principales, Raspberry y Arduino, ambas conectadas por comunicación serie USB.

3.1. Rapsberry Pi.

La Raspberry Pi 2 es la encargada de ejecutar los scripts y obtener la programación de la base de datos, al igual que es la encargada de recibir los datos de Arduino, como se puede ver en la siguiente figura (ilustración 13).

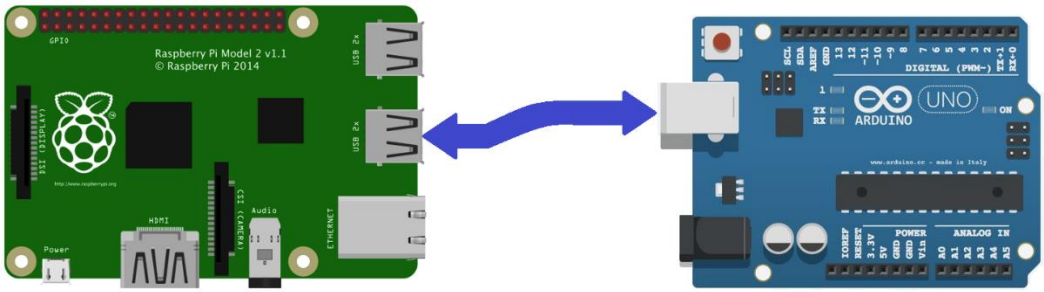

Ilustración 13: Esquema conexión Rpi-Arduino

Ejecuta 2 scripts, uno cada 24h y otro cada hora. El primer script que se ejecuta una vez cada 24h es el encargado de poner en hora cada arduino y realizar las consultas de la programación de las arduino para mandar los comandos necesarios, mientras que el otro script que se ejecuta cada hora recibe los datos de las arduino y los inserta en base de datos (Ambos scripts se detallan en el apartado de software).

3.2. Arduino.

La Arduino (En Arduino Uno el código ocupa aproximadamente el 50% de la memoria, en Arduino Mega ocupa aproximadamente el 6%) será la encargada del control de los relés y la obtención de los datos mediante los sensores conectados a sus pines de I/O digitales y analógicos. El esquema de conexión es el siguiente (ilustración 14):

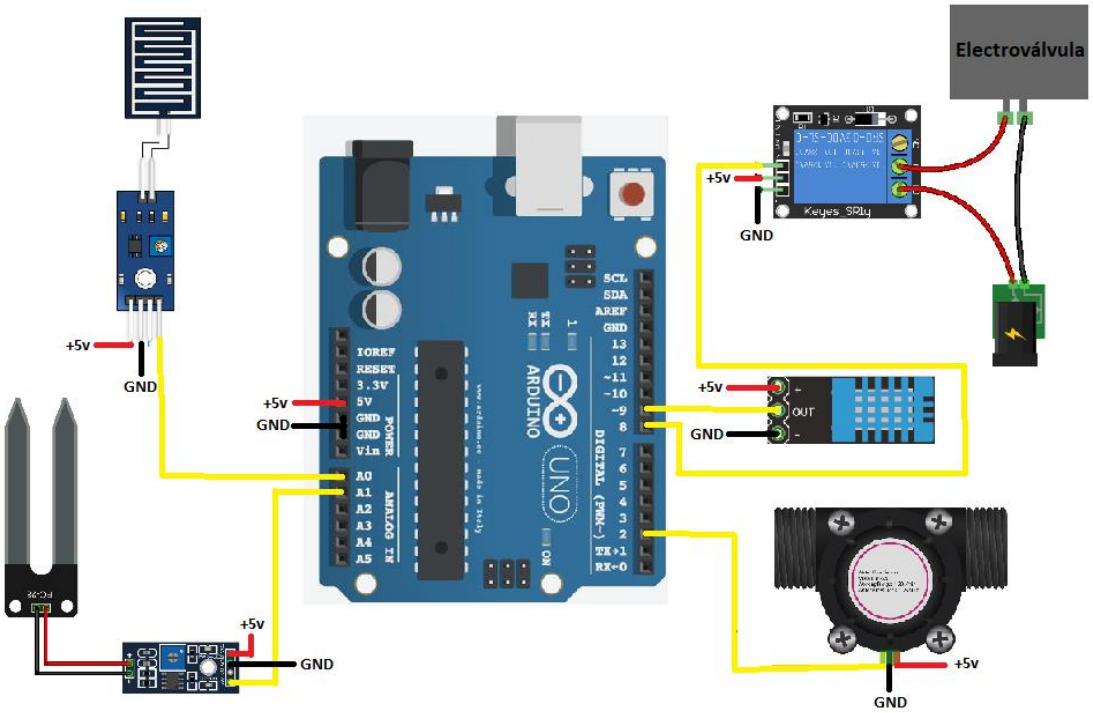

Ilustración 14: esquema conexión Arduino

Las entradas son:

- Sensor de lluvia (**FC-37**): **Pin A0**. (Analógico)
- Sensor de humedad de tierra o Higrómetro(**FC-28**): **Pin A1**. (Analógico)
- Caudalímetro (**YF-S201**): **Pin 2**. (Digital, por PWM)
- Sensor Temperatura y humedad de Ambiente(**DHT-11**): **Pin 9**. (Digital)
- Pines del 4 al 7 se usan como entrada para establecer en tiempo de ejecución el identificador. (Digital)

Las salidas son:

Relé: **Pin 8**. (Digital)

3.3. Actuadores y Sensores.

3.3.1. FC-37.

Es el sensor encargado de detectar la lluvia (ilustración 15), su esquema eléctrico es sencillo, únicamente compara la diferencia de resistencia y no tiene polaridad.

Ilustración 15: Sensor FC-37

La alimentación se hace a través de los pines Vcc y GND, y los datos se obtiene de dos maneras, como digital o como analógico, como puede verse en la ilustración 16, si usamos la salida digital obtenemos un valor alto o bajo dependiendo de si llueve o no, este valor se puede calibrar con el potenciómetro, mientras que si usamos la salida analógica obtenemos un valor entre 0 y 1023, el cual el umbral se define en código, estas comparaciones y calibraciones se llevan a cabo en el comparador LM393 (ilustración 17).

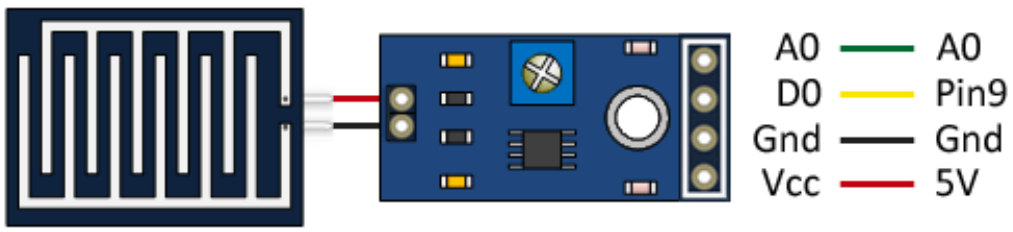

Ilustración 16: Sensor de lluvia FC-37 (conexiones)

La calibración es sencilla, metemos el sensor en agua o echamos agua sobre él y cambiamos el potenciómetro hasta que se dispare el umbral de salida que queremos. Su voltaje de trabajo es de 5v.

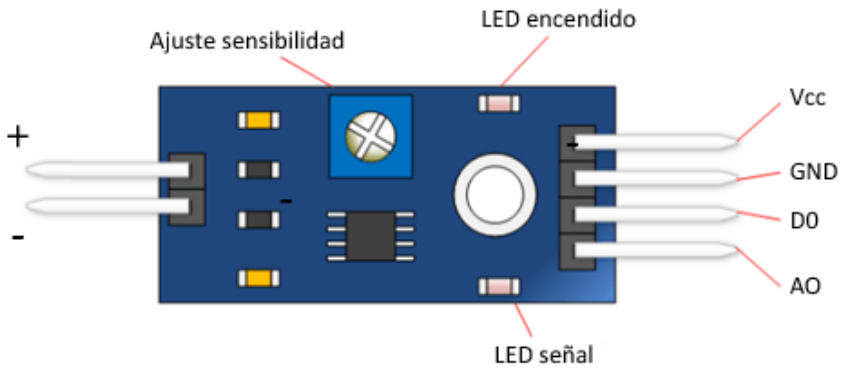

Ilustración 17: Comparador LM393

3.3.2. FC-28.

Se trata de un sensor para medir el grado de humedad del suelo (ilustración 18). El esquema eléctrico es sencillo. Alimentamos el módulo conectando GND y 5V a los pines correspondientes de Arduino.

Ilustración 18: Sensor FC-28

Si queremos usar la lectura analógica (ilustración 19), conectamos la salida A0 a una de las entradas analógicas de Arduino.

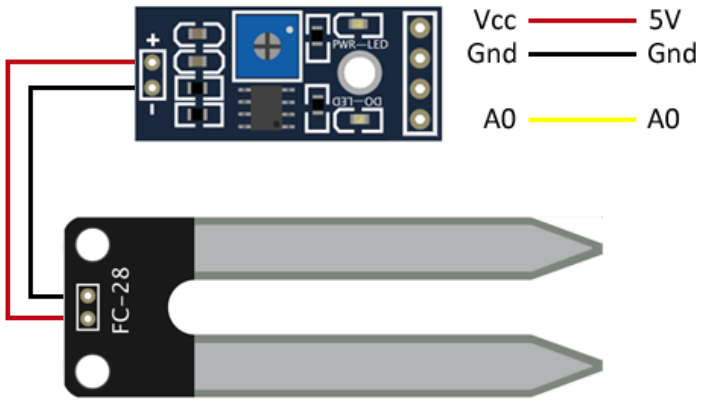

Ilustración 19: Sensor FC-28 (conexiones)

Si quisiéramos emplear el valor digital, se ajusta con el potenciómetro de la placa, en su lugar conectaríamos la salida D0 del sensor a una entrada digital de Arduino.

Cuenta con el comparador LM393 (ilustración 20) al igual que el sensor FC-38 y su alimentación es de 3.3v a 5v y su configuración se realiza igual, ajustando el potenciómetro para la salida digital.

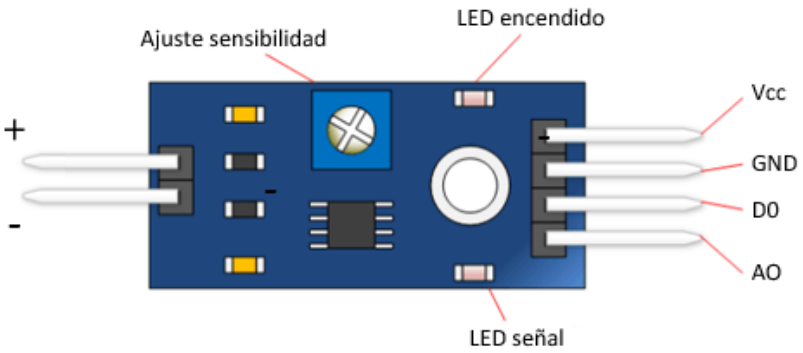

Ilustración 20: Comparador LM393

El funcionamiento es el siguiente:

- Salida alta cuando no se llega al umbral establecido.
- Salida baja cuando se llega al umbral.

3.3.3. Relay (Relé).

Es un dispositivo electromagnético que funciona como un interruptor controlado por un circuito eléctrico (ilustración 21) que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes (ilustración 22).

Ilustración 21: Relé

Dado que el relé es capaz de controlar un circuito de salida de mayor potencia que el de entrada, puede considerarse, en un amplio sentido, como un amplificador eléctrico o como separador de fase de control y potencia.

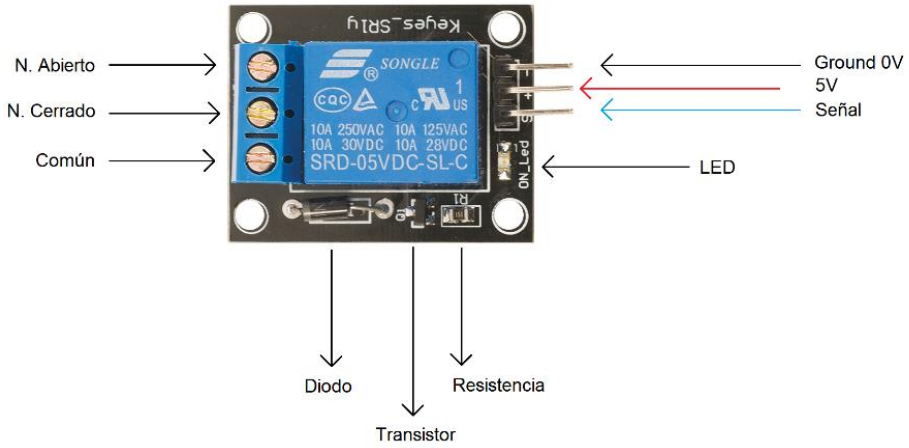

Ilustración 22: Conexiones relé

El relé usado en el prototipo se trata del songle srd-05vdc-sl-c que cuenta con las siguientes características:

- **Capacidad de conmutación** para 10A en un diseño de tamaño pequeño para técnicas de montaje en PCB's de alta densidad.
- **Selección de material plástico** para alta temperatura y un mejor desempeño térmico.
- **Completamente sellado.**
- **Circuito de relay magnético** simple para bajar el coste de producción en masa.
- **Voltaje nominal:** 5V.
- **Corriente nominal:** 70 a 90mA.
- **Resistencia:** 55 a 70 ohm.
- **Potencia:** 0.45W.
- **Capacidad para carga resistiva:** 28VDC@7A, 125VAC@10A, 240VAC@7A.
- **Capacidad para carga inductiva:** 120VAC@5A, 28VDC@5A.
- **Máximo voltaje:** 250VAC.

3.3.4. YF-S201.

Se trata del sensor de flujo o caudalímetro, viene con tres cables: rojo (energía 5-24VDC), negro (tierra) y amarillo (salida de pulsos de efecto Hall) como puede verse en la ilustración 23. Al contar los pulsos de la salida del sensor, se puede calcular fácilmente el flujo de agua. Cada pulso es de aproximadamente 2.25 mililitros. Hay que tener en cuenta que esto no es un sensor de precisión, y la frecuencia del pulso varía un poco dependiendo de la velocidad de flujo, presión del fluido y la orientación del sensor. Si se necesita más del 10% de precisión se tendrá que hacer la calibración adecuada. Sin embargo, es muy adecuado para las tareas diarias cotidianas.

Ilustración 23: Sensor YF-S201 (Caudalímetro)

La señal de pulso es una simple onda cuadrada así que es bastante fácil de registrar y convertir en litros por minuto utilizando la siguiente fórmula:

Frecuencia de pulsos (Hz) / 7.5 = caudal en L / min.

Sus características son:

- **Modelo:** YF-S201.
- **Tipo de Sensor:** Efecto Hall.
- **Voltaje Nominal:** 5 a 18V DC (Voltaje Mínimo Requerido 4.5V).
- **Máxima Corriente de operación:** 15mA @ 5V.
- **Voltaje de Salida:** 5V TTL.
- **Velocidad de Flujo:** 1 a 30 Litros/Minuto.
- **Rango de Temperatura:** -25 a +80℃.
- **Rango de Humedad:** 35%-80% RH.

- **Precisión:** ±10%.
- **Máxima Presión de Agua:** 2.0 MPa.
- **Salida del Ciclo de Trabajo:** 50% +-10%.
- **Pulsos por Litro:** 450.
- **Durabilidad:** Mínimo 300,000 ciclos.
- **Largo del Cable:** 15cm.
- **Conectores nominales** tubería de 1/2″.
- **Diámetro externo** de 0.78″.
- **Rosca** de 1/2″.
- **Tamaño:** 2.5″ x 1.4″ x 1.4″.

3.3.5. DHT-11.

Es el sensor de temperatura y humedad del ambiente (ilustración 24), es barato y fácil de conseguir, por lo que hay muchos ejemplos de uso y configuración.

Para su uso se añade una librería de este sensor para poder obtener de forma sencilla ambos datos, temperatura y humedad.

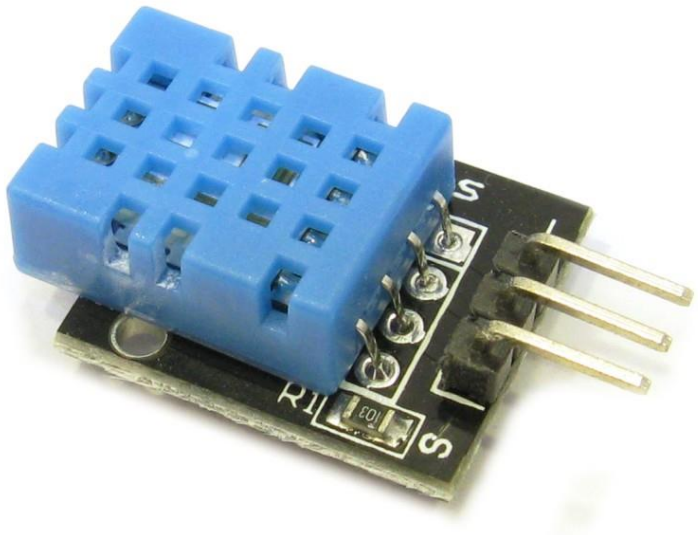

Ilustración 24: Sensor de temperatura y humedad DHT-11

Sus características son:

- **Modelo:** DHT-11.
- **Fuente de alimentación:** 3-5.5v.
- **Señal de salida:** Señal digital a través de bus único.
- **Elemento de detección:** Resistencia polimérica.
- **Rango de medición:** Humedad 20-90% RH, Temperatura +- 2.0 Celsius.
- **Resolución o sensibilidad:** Humedad 1% RH, Temperatura 0.1 Celsius.
- **Repetitividad:** Humedad +-1% RH, Temperatura +-1 Celsius.
- **Histéresis de humedad:** +-1% RH.
- **Estabilidad a largo plazo:** +-0.5% RH/Año.
- **Periodo de detección:** Promedio de 2s.

4. Implementación Firmware.

4.1. Arduino.

El protocolo de funcionamiento de Arduino se basa en la comprobación de un periodo, es decir, si la hora actual se encuentra en el periodo indicado se devuelve TRUE, en caso contrario FALSE, como puede verse en la ilustración 25.

Las comunicaciones se realizan a través de comando, se detecta la habilitación del puerto serie y se almacena en un buffer el comando recibido. Aquí hay que distinguir 2 tipos de comandos, el tipo 1 es el de configuración, en el que únicamente se lee el comando y se configuran las horas, mientras que el tipo 2 es de envío de datos.

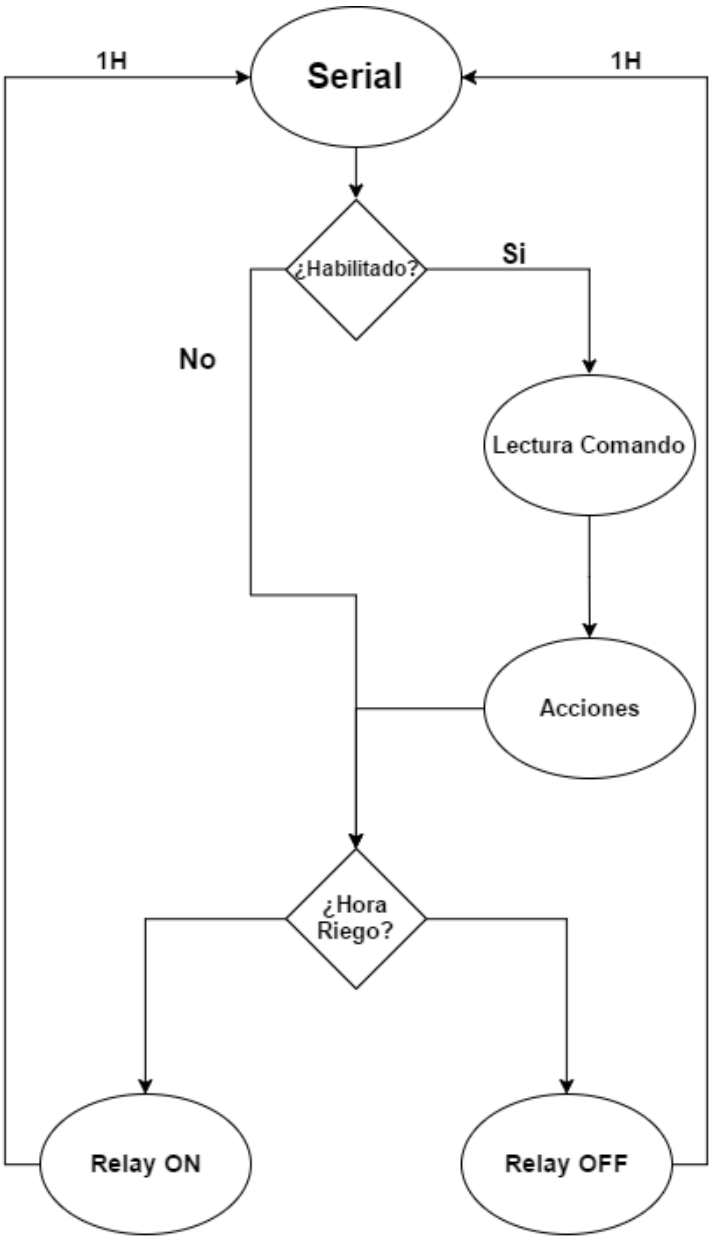

Ilustración 25: Diagrama de funcionamiento general

El diagrama es sencillo, si se recibe un comando se procesa dicho comando y se actúa en consecuencia, si se trata de configuración, se modifican las variables necesarias y se sigue, si se trata de lectura de datos, se envían los datos pedidos de acuerdo al comando.

Se continúa comprobando la hora de riego, si es TRUE se activa el relé, en caso contrario, se desactiva y como último paso se comprueban los litros consumidos cuya función está siempre trabajando, ya que se registra el consumo siempre, aunque el relé esté OFF.

4.2. Raspberry pi.

En el caso de Raspberry hay que distinguir 2 protocolos de comunicaciones basados en 2 script que se detallan en la parte de Python del apartado de Software.

El primer protocolo, que puede verse en la ilustración 26, consiste en el envío de la configuración al puerto serie. Por lo que hay que identificar las arduino para realizar la consulta correcta a la base de datos y obtener su configuración, una vez que la tenemos solo hay que construir los comandos y enviarlos, este protocolo se activa 1 vez cada 24h, los comandos son:

- D, envía la orden para recibir el identificador de la arduino para realizar la consulta.
- SHH:MM:SS-DD/mm/YYYY, este comando pone en hora a arduino, se envía cada 24h por seguridad en caso de que arduino pierda alimentación y pierda la hora actual.
- IHH:mm, envía la hora de inicio de riego.
- FHH:mm, envía la hora de fin de riego.

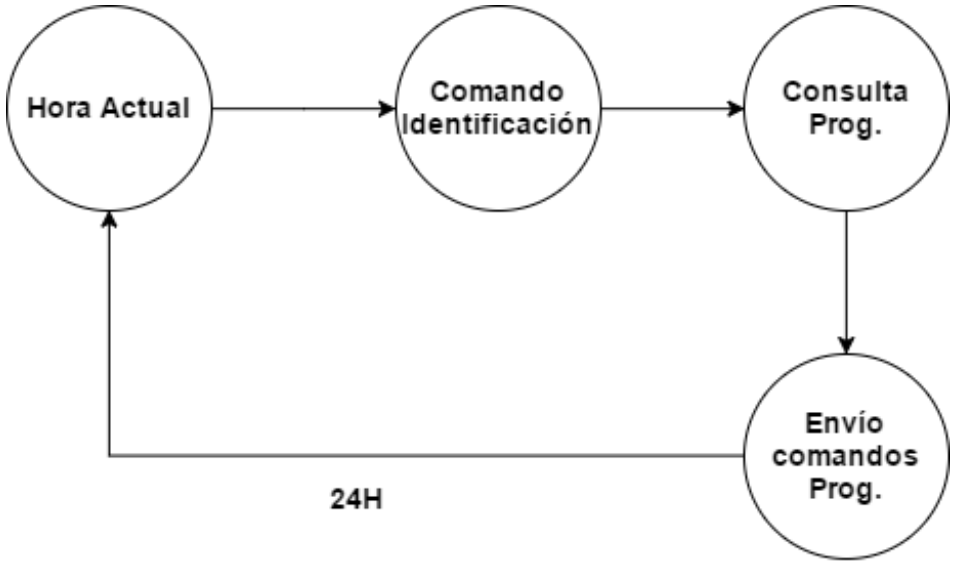

Ilustración 26: Diagrama de estados de la configuración

El segundo protocolo, que puede verse en la ilustración 27, consiste en la lectura de los datos y su inserción en la base de datos. En este caso no hay que identificar a las arduino ya que esa identificación se envía en los datos que se inserta, el comando es ‘L’.

Una vez recibido los datos, se insertan en la base de datos, en caso de fallo de hace un RollBack para que la base de datos no sufra modificaciones o entradas erróneas.

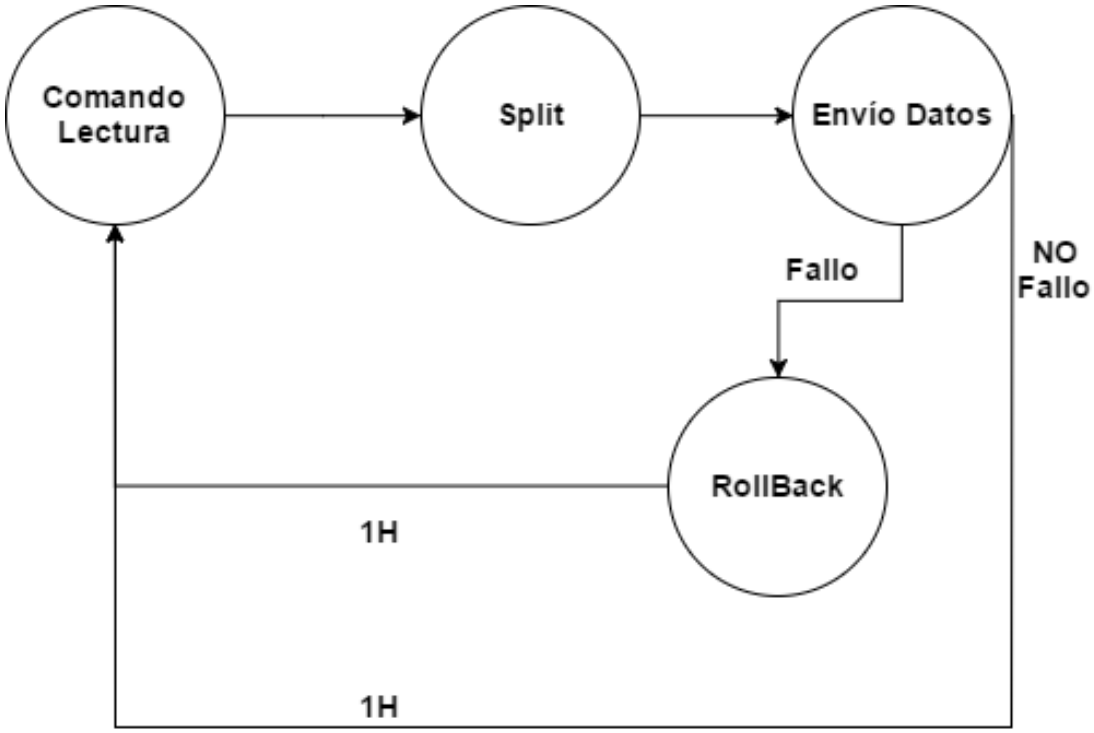

Ilustración 27: Diagrama de estados del envío de datos

5. Software.

5.1. Python.

En Python se ha realizado los protocolos de configuración y lectura de datos de arduino, para ello se necesitan 2 script como se ha mencionado en el apartado 4.2. Raspberry Pi. Por lo que en ambos scripts al hacer uso de la base de datos y el puerto serie y tiempo, es necesario añadir las librerías de MySQLdb, Serial y Time de Python, que se añaden con import e instalando las librerías en la Rpi.

El script de configuración comienza con la obtención del identificador de las arduino, para ello abre el primer puerto serie que se trata del '/dev/ttyACM0', '…/ACM1' el siguiente y así sucesivamente:

```
PuertoSerie = serial.Serial(port='/dev/ttyACM0',
baudrate=9600,timeout=5.0)
```

Antes que nada, se pone en hora la arduino enviando el comando 'SHH:MM:SS-dd/mm/YYYY', para formar este comando se concatena 'S' con la instrucción **time.strftime("%H:%M:%S-%d/%m/%Y")**, para enviar el comando es así:

```
var = 'D'
PuertoSerie.write(var)
time.sleep(0.1)
#Lectura de datos
byteToRead=PuertoSerie.inWaiting();
sArduino = PuertoSerie.read(byteToRead).strip()
```

Una vez abierto el puerto se envía el comando 'D' para recibir por el puerto serie el identificador de la arduino conectada a ese puerto. Una vez que tenemos el identificador, se construye la consulta SQL que se envía a la base de datos para obtener las horas de inicio y fin de esa arduino, por lo que para enviar la consulta hay abrir la conexión con la base de datos.

```
bd =
MySQLdb.connect(host='127.0.0.1',user='root',passwd='Admin',db='ardu
reg')
cursor = bd.cursor()
sql="SELECT P.Hora_ini, P.Hora_fin FROM arduino AS A,
arduino_programacion AS AP, programacion AS P WHERE
A.IDArduino='"+sArduino+"' AND A.IDArduino=AP.Arduino_IDArduino AND
AP.Programacion_IDProg=P.IDProg;"

try:
      # Ejecutamos el comando
      cursor.execute(sql)
```

```
        resultado=cursor.fetchall()
        print resultado
        for registro in resultado:
                inicio=registro[0]
                fin=registro[1]
                print inicio
                print fin
        print "exito!"
except:
        print "Error"

cursor.close()
```

Una vez que tenemos las horas, se preparan los dos comandos siguientes que son los de inicio y fin. Para el comando de inicio hay que concatenar el comando 'I' con la hora de inicio y para el comando de fin de riego se concatena el comando 'F' con la hora de fin, las horas se reciben en forma de array por lo que se accede a la hora de inicio con el índice 0 y a la hora de fin con el índice 1.

El script de lectura de datos no necesita del identificador ya que se pasa junto a todos los datos con el comando de lectura, que para ello solo hay que enviar el comando 'L'. Por lo que solamente se reciben los datos, se construye el SQL y se envía a la base de datos:

```
sArduino = PuertoSerie.readline().strip()
#Separa la cadena en valores, cada valor hasta la coma es almacenado
en una variable(modificar)
sTempAmbiente,sHumAmbiente,sLitros,sId=sArduino.split(',')
ta = float(sTempAmbiente)
ha = float(sHumAmbiente)
li = float(sLitros)
zona = sId
PuertoSerie.close()
# Establecemos la conexi□n con la base de datos
#bd = MySQLdb.connect("host","user","pass","db" )
bd =
MySQLdb.connect(host='127.0.0.1',user='root',passwd='Admin',db='ardu
reg')
# Preparamos el cursor que nos va a ayudar a realizar las
operaciones con la base de datos
cursor = bd.cursor()
#Almacenamos los valores en tabla datos de la base
sql='INSERT INTO
estadisticas(Fecha,Temperatura,Hum_Aire,Litros,Arduino_IDArduino)
VALUES (now(),"%f","%f","%f","%s");' % (ta,ha,li,zona)
print (sql)
try:
        # Ejecutamos el comando
        cursor.execute(sql)
        bd.commit()
        print "exito!"
except:
        print "Error"
        bd.rollback()
# Nos desconectamos de la base de datos
bd.close()
```

5.2. API.

Para la API se hace uso del framework CodeIgniter, el cual nos proporcionas la estructura de directorios para el desarrollo de la propia API y la web.

Para poder usarla hay que establecer las configuraciones como los datos de la base de datos y los módulos php y html que se van usar, en este caso se añadió los datos de la base de datos junto a los módulos de formularios con sus validaciones, url base, lenguaje, etc… de la siguiente forma.

Se añaden o modifican estas líneas en el archivo '**config.php'** que se encuentra en el directorio **'config'** de codeigniter:

```
$config['base_url'] = 'http://localhost/tfg';//url o ip del server
apache
$config['language']     = 'spanish';
$config['charset'] = 'UTF-8';
```

Para añadir la carga de los módulos a usar se modifican las siguientes líneas del archivo '**autoload.php'** que también se encuentra en el directorio 'config':

```
$autoload['libraries'] = array('database','form_validation');
$autoload['helper'] = array('url','form');
```

Para la conexión con la base de datos se modifican los datos del archivo '**database.php'** del directorio 'config':

```
$db['default'] = array(
      'dsn' => '',
      'hostname' => 'localhost',
      'username' => 'root',
      'password' => 'Admin',
      'database' => 'ardureg',
      'dbdriver' => 'mysqli',
      'dbprefix' => '',
      'pconnect' => FALSE,
      'db_debug' => (ENVIRONMENT !== 'production'),
      'cache_on' => FALSE,
      'cachedir' => '',
      'char_set' => 'utf8',
      'dbcollat' => 'utf8_general_ci',
      'swap_pre' => '',
      'encrypt' => FALSE,
      'compress' => FALSE,
      'stricton' => FALSE,
      'failover' => array(),
      'save_queries' => TRUE
);
```

A partir de aquí ya se puede construir la API la cual tendrá las funciones que siguen el siguiente formato:

- **Nombre_llamada_get():** para llamadas GET.
- **Nombre_llamada_post():** para llamadas POST.

La construcción de la API se realiza en el directorio **'controllers/api'** en el archivo **v0.php**, que si no existe se crea de forma manual. El cual se comienza creando la clase del mismo nombre que el archivo y como extensión se añade **REST_controller**.

Para hacer las funciones de la API que son las que serán llamadas se hace siguiendo la siguiente estructura:

```
public function estadistica_get(){
            //comprobar entrada de dia mes o año, en cada caso es una
consulta
        $dia = $this->get('dia');
        $mes = $this->get('mes');
        $anno = $this->get('anno');
        $id = $this->get('id');
        if($dia != NULL and $mes != NULL and $anno != NULL){
            $query = $this-> db ->query("SELECT * FROM estadisticas
WHERE DAY(Fecha) = ".$dia." AND MONTH(Fecha) = ".$mes." AND
YEAR(Fecha) = ".$anno);
        }else if($dia == NULL and $mes != NULL and $anno == NULL){
            $query = $this-> db ->query("SELECT * FROM estadisticas
WHERE DAY(Fecha) = ".$dia." AND MONTH(Fecha) = MONTH(now()) AND
YEAR(Fecha) = YEAR(now())");
        }else{
            $query = $this-> db ->query("SELECT Fecha, Temperatura,
Hum_Aire, Litros FROM estadisticas WHERE YEAR(Fecha)=YEAR(now()) AND
Arduino_IDArduino= '".$id."'");
        }
        $resultado = $query->result();
        $this->response($resultado, REST_Controller::HTTP_OK);
    }
```

Al igual que para POST:

```
public function prog_post()
    {

        $idProg = $this->input->get("id");
        $ini = $this->input->get("ini");
        $fin = $this->input->get("fin");
        $query_update = $this-> db->query("UPDATE programacion SET
Hora_ini=".$ini.", Hora_fin=".$fin."WHERE IDProg=".$idProg);


        if($query_update){
            $this->
response($query_update,REST_Controler::HTTP_CREATED);//201
        }else{
```

```
            $this->
response($query_update,REST_Controler::HTTP_INTERNAL_SERVER_ERROR);//
/500
        }
    }
```

5.3. Base de Datos.

La base de datos, que puede verse en la ilustración 28, sigue la 3FN (tercera forma normal) para cumplir con el estándar y tiene la siguiente estructura:

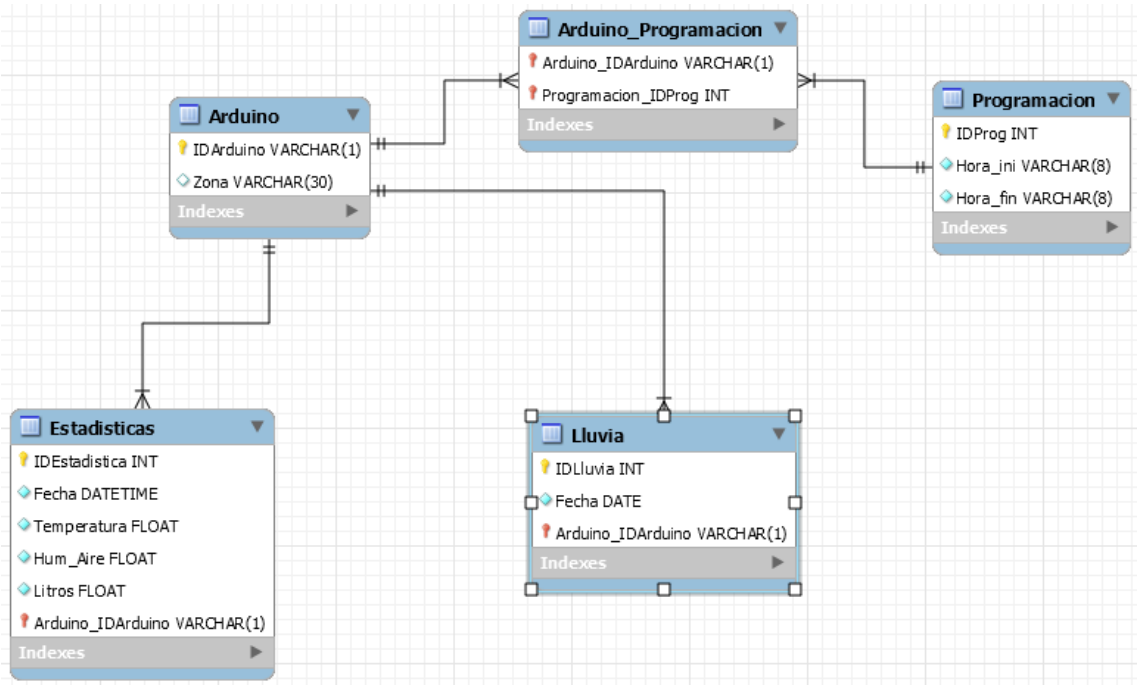

Ilustración 28: UML base de datos

Las tablas están relacionadas de la siguiente manera, por un lado, tenemos la identificación de las arduino con su sector que se relaciona de FK con las estadísticas, es decir, que una arduino tendrá muchas entradas en las estadísticas que almacena los datos recibido de esa arduino (relación 1:N), la tabla de programación permite a través de la tabla intermedia una relación N:M para poder tener programación en intervalos de cada una las arduino. Para el registro de lluvias se almacena únicamente fecha y hora.

5.4. Interfaz de Usuario.

Está basada en HTML, javascript y bootstrap, que es un CSS ya creado para su uso que permite que los estilos sean adaptados a dispositivos móviles y diferentes resoluciones.

Su programación se inserta en vistas en la propia API que se llaman con **'load->view(…)'** dentro de funciones php, básicamente es un modelo de programación vista-controlador.

Para la generación de las gráficas se hace uso de Highcharts (ilustración 29), que se trata de un grupo de funciones javascript ya creadas que permiten crear las gráficas a partir del json obtenido de las consultas realizadas a la API.

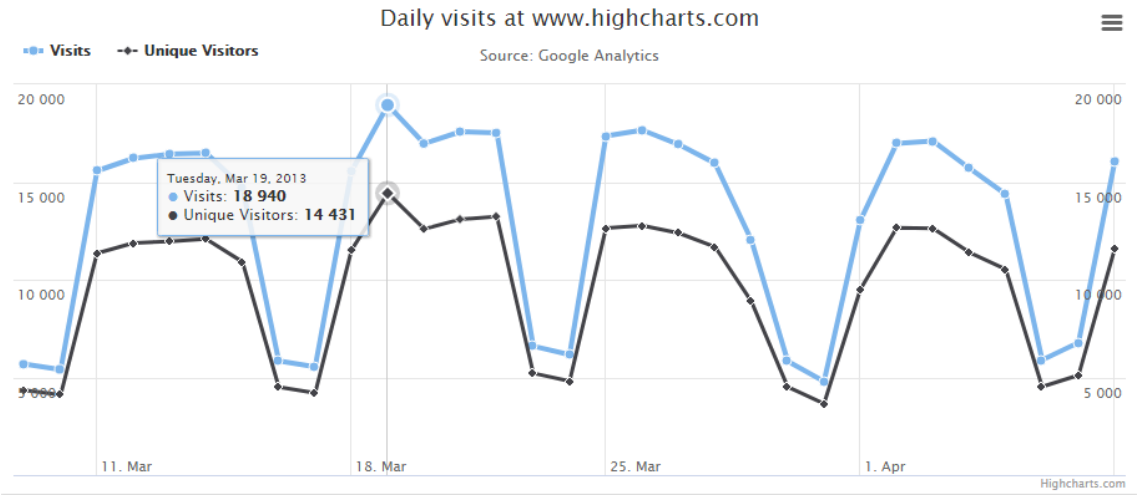

Ilustración 29: Vista de la gráfica con Hightcharts

5.5. Apache.

Tanto la API como la web se almacenan en el directorio **'httpd'** de Apache, bajo el subdirectorio 'TFG' en este caso, puede ser llamado como uno quiera, pero habría que modificar la línea de la API del localhost. El servidor tiene configurado el puerto 80 para el servicio web, como puede verse en la ilustración 30.

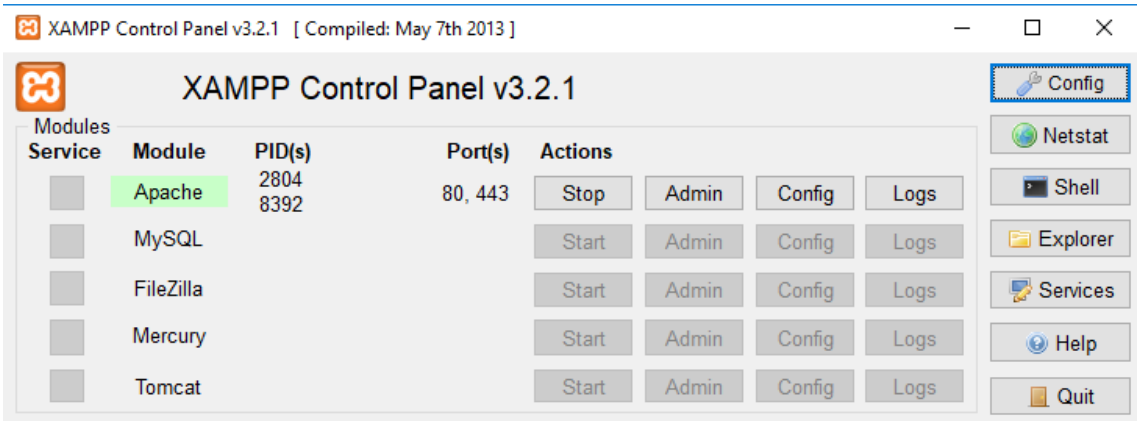

Ilustración 30: Panel de control con puerto Apache(XAMPP)

5.6. Arduino.

En la programación de Arduino, básicamente se realiza una configuración en tiempo de ejecución y luego se comprueba constantemente la llegada de comandos, la hora de riego y el consumo de litros.

La configuración del identificador se realiza a través de un bucle en setup(), el cual comprueba los pines desde el 4 al 7, el que esté en alto indicará el ID de la arduino que va desde la A hasta la D, por defecto, si no hay ningún de esos 4 pines en alto asigna el ID 'E'.

```
for(int i=4;i<8;i++ ){
    pinMode(i, INPUT);
    if(digitalRead(i)== HIGH){
      switch(i){
        case 4:
           id='A';
           break;
        case 5:
           id='B';
           break;
        case 6:
           id='C';
           break;
        case 7:
           id='D';
           break;
        default:
           id='E';
           break;
      }
    }
  }
```

La comprobación de comandos se realiza con una condición IF en el puerto serie, si es TRUE, se procesa el mensaje almacenándolo en un buffer, en caso contrario se sigue ejecutando normalmente.

```
if(Serial.available()>0){
    memset(cmdLeido,0,sizeof(cmdLeido));
    while(Serial.available()>0){
    cmdLeido[posicion]=Serial.read();
    posicion++;
    }
    posicion = 0;
    processMsg();

  }
```

La comprobación de la hora de riego se realiza de la misma forma, un condicionante IF de una función que comprueba el intervalo de riego, si la hora actual se encuentra en ese intervalo se activa el relé, en caso contrario se desactiva.

La comprobación del consumo de litros se realiza siempre, esté el relé activo o no. El cálculo del consumo se realiza contando los pulsos que llegan al pin 2 por PWM. Dentro de esta función se realiza la comprobación con **millis()**, para realizar el cálculo cada segundo y obtener los pulsos acumulados durante ese segundo, la función para calcularlo es la siguiente:

```
void caudalFunc()
{
  if(millis() - tiempoAnterior > 1000)
    {
      // Realizo los cálculos
      tiempoAnterior = millis(); // Actualizo el nuevo tiempo
      pulsos_Acumulados += pulsos; // Número de pulsos acumulados
      litros = pulsos_Acumulados*1.0/450; // Cada 450 pulsos son un
litro
      pulsos = 0; // Pongo nuevamente el número de pulsos a cero
    }
}
```

6. Planificación y Costes.

6.1. Planificación.

6.1.1. Tareas.

Definición del sistema: Se establecen los requisitos y objetivos a desarrollar, además de empezar obtener información de proyectos relacionados.

Tecnología a usar: Elección de la tecnología comprobando pros y contras de ellas y compra de del hardware necesario.

UML: Diseño del UML de la base de datos.

Programación BBDD: Creación de la base de datos.

Testeo y errores (BBDD): Comprobación del funcionamiento de la BBDD.

Programación y testeo individual (Hardware): Programación y testeo individual de cada uno de los componentes hardware escogidos para el proyecto.

Programación conjunta: Unión de la programación individual de los sensores y plataformas en un solo módulo.

Testeo programación conjunta: Comprobación del módulo o sistema final.

Web: Diseño de la web de control para el usuario y API.

Comunicaciones: Programación de los script y protocolos de comunicación entre plataformas.

Integración con Hardware: Unión de la parte software con el hardware para formar el sistema completo.

Pruebas unitarias: Pruebas de cada una de las partes de forma individual dentro del sistema completo junto al software.

Pruebas de campo: Pruebas simuladas de entornos reales.

Corrección de errores: Corrección de los posible errores y fallos del sistema.

Documentación: Realización de la documentación necesaria.

Prototipo: Diseño del prototipo.

6.1.2. Definición de las actividades.

Requisitos funcionales con respecto a las distintas tareas en su composición:

Requisito funcional 1: Definición del proyecto con respecto a las tecnologías existentes.

- Realización de estudio de mercado sobre las tecnologías utilizadas y emergentes.
- Análisis del impacto sobre el uso de la tecnología sobre el sistema.
- Decisión de implementación respecto a valoración académica, personal y empresarial.

Requisito funcional 2: Definición de la BBDD

- Estudio de la relación de los datos recogidos por los sensores con información contrastada en Internet.
- Implementación de BBDD en MySql con posibles mejoras con la integración de base de datos distribuidas.
- Integración de conexiones persistentes y no persistentes en la sincronización de datos.
- Búsqueda de una integración segura de la tecnología de la BBDD.
- Integración de sistemas de debugger online.

Requisito funcional 3: Hardware

- Uso de Hardware abierto.
- Búsqueda de códigos realizados para su posible reutilización.
- Estudio de ámbito de funcionamiento de los sensores.
- Testeo de los sensores y equipos.

Requisito funcional 4: Software

- Uso de nuevas interfaces web limpias y sencillas en su integración al usuario.
- Investigación de la comunicación software con tecnología cifrada.
- Uso de la nueva tecnología de posibles incorporaciones de Bot en Telegram.
- Integración fiable con Hardware.

Requisito funcional 5: Testeo y corrección del sistema

- Equipos de prueba al hardware con simulación de agentes externos.
- Pruebas en entornos reales.

6.1.3. Diagrama de Gantt.

El diagrama de Gantt se encuentra en el anexo A (ilustración 35), ya que por formato y espacio no es posible insertarlo aquí.

En él mostramos todas las tareas junto a los periodos de inicio y duración planeados y reales, de forma que podemos observar una desviación respecto a la planificación estipulada que se ha producido por algún retraso.

Los periodos se contemplan en meses, es decir cada periodo corresponde a 1 mes, ya que el proyecto se ha ido realizando a lo largo del curso en paralelo con las clases y exámenes de las asignaturas de cuarto y tercer año de la carrera, se ha dedicado un total de 300h, por lo que cada periodo corresponde a 30h/mes.

En la ilustración 31 podemos ver un gráfico donde nos muestra las tareas y su parte proporcional completada dentro del proyecto.

Ilustración 31: Gráfico Tareas

6.2. Costes.

El coste de desarrollo se ha calculado usando un solo elemento de cada pieza hardware, mientras que el de producto se calcula atendiendo a un diseño final aprovechando los 4 puerto USB de Raspberri Pi 2, por lo que los elementos hardware se multiplican por 4.

6.2.1. Coste Desarrollo.

El presupuesto se divide en 2 partes, una parte material, que comprende el presupuesto en los elementos hardware (ilustración 32), y otra parte software, correspondiente al desarrollo del software.

Con respecto al material:

- **Raspberry Pi 2:** 40€.
- **Arduino Uno R3:** 23,95€.
- **Relé:** 1,74€.
- **FC-37 (Lluvia):** 2,45€.
- **FC-28 (Humedad suelo):** 2,33€.
- **DHT-11 (Humedad y temperatura ambiente):** 4,70€.
- **Protoboard:** 6,81€.

- **YF-S201 (Caudalímetro):** 6,90€.

En la parte **Hardware**, el presupuesto es de: **88,88€**.

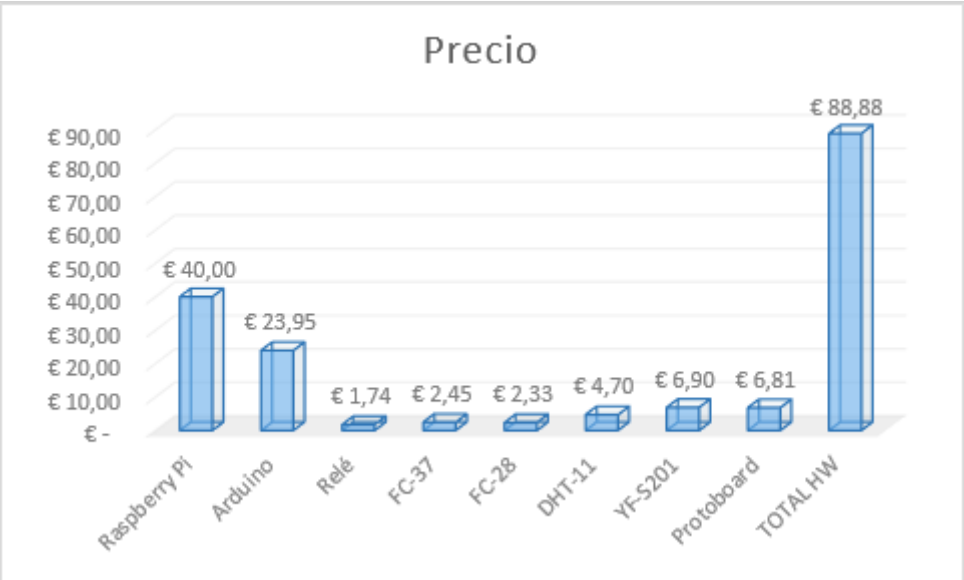

Ilustración 32: Gráfico Coste Hardware

Si nos referimos a la parte software, las horas del proyecto son de **300h**, si suponemos que el desarrollador cobra **10€** la hora, nos hace un total de: 3.000€.

El total del presupuesto del desarrollo del prototipo, que se muestra en la ilustración 33, es de **3.000€ + 88,88€ = 3.088,88€**.

Ilustración 33: Gráfico Coste Hardware y Software

6.2.2. Coste Producto.

Partimos de que arduino, los sensores y relés, habría que multiplicarlo por 4, ya que se haría uso de todos los puertos de Raspberry Pi 2 como sistema final, a lo que habría que añadir el **beneficio** a obtener que sería de entre un **15-20%** y la **amortización** que sería del **5%** (visible en la ilustración 34), por lo que el presupuesto quedaría así:

- **Raspberry Pi 2:** 40€.
- **Arduino Uno R3 x4:** 23,95€ x 4 = 95,80€.
- **Relé x4:** 1,74€ x 4 = 6,96€.
- **FC-37 (Lluvia) x4:** 2,45€ x 4 = 9,80€.
- **FC-28 (Humedad suelo) x4:** 2,33€ x 4 = 9,32€.
- **DHT-11 (Humedad y temperatura ambiente) x4:** 4,70€ x 4 = 18,80€.
- **YF-S201 (Caudalímetro) x4:** 6,90€ x 4 = 27,60€.

Lo que hace un total en material de: **208,28€** por unidad.

Sumando el beneficio y la amortización, suponen un incremento de entre el **20-25%**, que da un total de: 208,28 + 15-20% = **249,94€- 260,35€.**

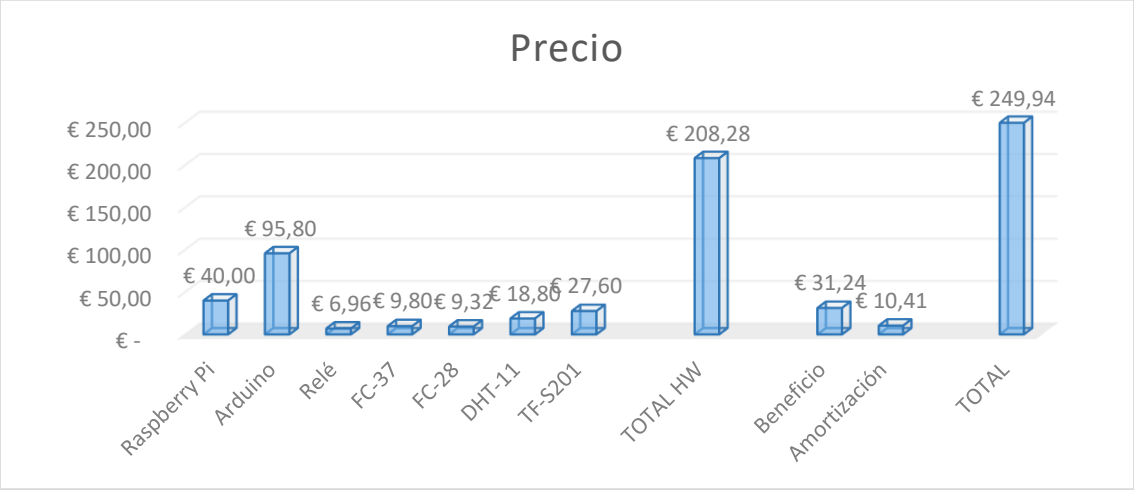

Ilustración 34: Coste producto con 20%

7. Conclusiones.

Nos encontramos frente a un proyecto que pretende suplir las carencias de programadores de riego automáticos baratos frente a las necesidades del ahorro de agua en el caso de fallos en la programación del riego por haber calculado mal las horas de riego y excederse, como la necesidad del paro de riego de forma automática en caso de lluvia, además de ofrecer la recogida de datos para mejorar la productividad o realizar estudios.

Todo ello remoto, de forma que podemos programar los riegos por sectores y programarlos de forma independiente sin la necesidad de desplazarnos al lugar.

Para poder llevar a cabo el proyecto se han usado tecnologías y plataformas de código abierto y libres como han sido Raspberry Pi, Arduino, PHP, JavaScript, Python, MySQL, BootStrap y Apache, ya que existe mucha información y ejemplos con esta tecnología y tienen a comunidades grandes tras ellas, por lo que la solución de problemas y obtener ejemplos es más fácil, además del ahorro que supone el uso de plataformas y software libre.

Referente a los objetivos planteados, se han conseguido superar todos a excepción de realizar una implementación para el control de los periodos de lluvia y mantenerlos en un registro, de forma que se pudiera consultar cuando y durante que tiempos exactos estuvo lloviendo en el cultivo, en la base de datos esto se ha tenido en cuenta.

Tampoco se ha podido llevar a cabo la incorporación de tecnología inalámbrica y establecer una red de nodos muchos más efectiva ni la implementación de la programación por intervalos para cada arduino, únicamente se permite una única programación por cada arduino.

En lo que se refiere a planificación, los retrasos se produjeron por la espera de los materiales obtenidos a través de distribuidores asiáticos, ya que podían retrasarse de acuerdo a la estimación de llegada de los paquetes, como lo que ocurrió, que se estimó la llegada en un mes o mes y medio y algunos paquetes tardaron tres meses en llegar.

8. Trabajo futuro.

Como trabajo futuro queda realizar la implementación del control mediante registros de los periodos de lluvia que se produzcan en el cultivo, actualmente esto se ha contemplado en la base de datos, pero no se ha desarrollado los controles y protocolos necesarios en la API o en Python para poder realizarlo.

Se quiere realizar esto como futuro ya que de esta forma podríamos ver y comparar para su estudio el ritmo de crecimiento de los cultivos o la producción, además de dotar al sistema de elementos inalámbricos más eficientes y establecer una red de nodos, ya que sería menos engorroso que tener el sistema cableado.

Además, se pretende realizar una PCB que contenga esta tecnología de forma que todo el sistema estuviera englobado en una única placa que producida en masa sería más barata frente a su posible comercialización como producto.

Referente a la base de datos, sería posible la incorporación a ésta de usuarios, de forma que se pudieran tener todos los sistemas identificados para un mayor control global, de forma que desde el punto de vista lógico veríamos una base de datos centralizada, aunque ésta realmente es distribuida físicamente, ningún dato estaría localmente, evitando así la perdida de estos en caso de algún accidente en el Hardware.

En el apartado de la interfaz de control, se pretender hacer una APP para Smartphone, de forma que no se necesite entrar al servidor web, si no únicamente usar la APP, ofreciendo el mismo control y estadísticas que la web, lo que implica el uso de usuarios. Además, se podría hacer uso de la API de Telegram para realizar un BOT que nos permitiera conocer mediante comandos simples como ‘Temperatura’ o ‘Litros’ para obtener ese dato concreto en ese mismo instante y no tener que mirar las estadísticas que se actualizan cada hora.

En el apartado estadístico, actualmente se ofrecen gráficos simples, por lo que podría mejorarse y realizar estadísticas más complejas para adaptarlo más a una perspectiva de estudio y experimentación como pueden ser huertos inteligentes o huertos hidropónicos, de esta forma podemos dotar a biólogos y/o botánicos de más herramientas que faciliten sus estudios y experimentos.

Para resolver la conectividad se podría hacer uso de un módulo GSM, tecnología conocida y que nos daría la cobertura necesaria para enviar y recibir los datos y programación, con lo que también resolveríamos la parte de IoT, ya que cada módulo estaría identificado.

9. Bibliografía.

Estudio de mercado:

- https://www.rainbird.es/
- http://www.novedades-agricolas.com/es/riego/sistemas-de-riego/riego-automatico

Precedentes:

- https://medium.com/@mantgambl/sistema-de-riego-automatizado-con-raspberry-pi-arduino-parte-2-89be06249fa8
- http://upcommons.upc.edu/bitstream/handle/2099.1/25074/memoria.pdf?sequence=4
- https://www.youtube.com/watch?v=SNKneLTf3Kk
- http://androidelectronic.esy.es/index.php/proyectos/83-arduinoriego
- https://www.xatakahome.com/trucos-y-bricolaje-smart/sistema-de-riego-inteligente-con-tecnologia-arduino-perfecto-para-mantener-tus-plantas-saludables

Sensores:

- http://www.prometec.net/sensores-dht11/ (DHT-11)
- http://miarduinounotieneunblog.blogspot.com.es/2016/04/caudalimetro-con-sensor-de-flujo-yf.html (YF-S201)
- https://www.youtube.com/watch?v=YeAv5CQYayE (FC-37)
- http://www.microjpm.com/products/relay-srd-05vdc-sl-c-spdt-5vdc/ (Relé)

Librerías Arduino:

- http://playground.arduino.cc/Code/time (Time)
- http://www.prometec.net/time-arduino/ (Time)
- http://forum.arduino.cc/index.php?topic=325071.0 (Time)

Software:

- http://www.maestrosdelweb.com/guia-python-bases-de-datos-mysql/ (Python + mysql)

- https://geekytheory.com/php-mysql-highcharts-mostrar-grafica-dinamica-en-funcion-del-tiempo/ (php+mysql+highcharts)
- https://code.tutsplus.com/es/tutorials/working-with-restful-services-in-codeigniter--net-8814 (codeigniter)
- https://www.codeigniter.com/user_guide/ (codeigniter)
- http://getbootstrap.com/getting-started/ (BootStrap)

Diagramas:

- https://www.draw.io/ (Herramienta online gratuita para realizar diagramas)

Definiciones:

- https://es.wikipedia.org/wiki/Wikipedia:Portada

10. Anexos.

10.1. Anexo A.

Planificador del proyecto

Resaltado del períod 1 | Plan

ACTIVIDAD	PLAN INICIO	PLAN DURACIÓN	REAL INICIO	REAL DURACIÓN	PORCENTAJE COMPLETADO
Definición del sist.	1	2	1	2	100%
Tecnología a usar	2	2	2	3	100%
UML	2	2	2	1	100%
Prog. BBDD	4	2	3	2	100%
Testeo y errores BBDD	4	2	3	2	100%
Prog. Y Testeo ind. (HW)	4	2	5	1	100%
Prog. Conjunta	5	3	6	3	100%
Testeo prog. Conjunta	5	3	6	3	100%
Web	5	2	5	4	90%
Comunicaciones	5	1	5	2	100%
Integración con Hardw.	6	1	7	1	100%
Pruebas unitarias	6	2	7	2	100%
Pruebas de campo	6	2	7	2	100%
Corrección de errores	6	3	7	3	100%
Documentación	3	7	3	8	100%
Prototipo	4	6	5	6	90%

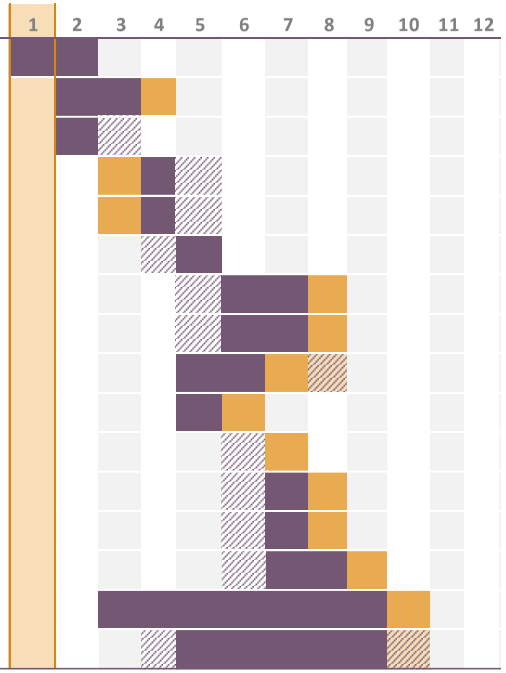

Ilustración 35: Diagrama de Gantt

Leyenda:

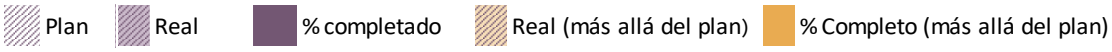

Ilustración 36: Leyenda