Escuela Superior de Ingeniería
Ingeniería Técnica en Informática de Gestión

Gestión de sistema de riego automático con Arduino

Antonio Barcia García

Cádiz, a 11 de julio de 2017

Escuela Superior de Ingeniería
Ingeniería Técnica en Informática de Gestión

Gestión de sistema de riego automático con Arduino

Departamento: Ingeniería en Automática, Electrónica, Arquitectura y Redes de Computadores

Director del proyecto: Juan Antonio Leñero-Bardallo

Autor del proyecto: Antonio Barcia García

Cádiz, a 11 de julio de 2017
Fdo. Antonio Barcia García

Índice general

Índice de figuras

Capítulo 1

Introducción

Esta memoria presenta el desarrollo de un sistema de monitorización y gestión de riego automático, basado en la plataforma de hardware libre *Arduino* [WIKb], al que he nombrado *Jarduino*.

El nombre es una alusión españolizada a *Garduino* [INSa], un pequeño proyecto de control de riego con *Arduino* (colgado en la web *Instructable* [INSb]) que, entre otros del mismo género, inspiró el Proyecto de Fin de Carrera que las siguientes páginas cubren.

1.1. Objetivos

Este Proyecto de Fin de Carrera presenta un demostrador o prototipo de un sistema de gestión riego automatizado con una interfaz amigable y fácil de usar.

El objetivo del prototipo presentado es validar la hipótesis de trabajo [1] de este Proyecto Fin de Carrera: que a través de una interfaz de usuario conectada a un dispositvo de monitorización y riego automatizado, se puede mejorar progresivamente la gestión de un pequeño huerto, y que es posible hacerlo con un prototipo que una persona aficionada al hardware libre — en este caso *Arduino* — pueda montar por sí misma.

He desarrollado este Proyecto de Fin de Carrera partiendo de las ideas que libremente comparten la comunidad de aficionados al *DIY* y el hardware libre (más adelante me referiré a esa comunidad como *makers*, en inglés *hacedores*). Por eso el prototipo implementado no persigue la utilidad inmediata de su uso, sino compartir una aportación a la idea de aunar agricultura y Arduino, que sirva de base para futuros proyectos en la misma

[1] Ver sección 2.1, página 13.

línea.

Al estar *Jarduino* basado en *Arduino* y dispositivos electrónicos compatibles con él, resulta sencillo encontrar los componentes para montar una versión personalizada que extienda o modifique el sistema, como yo mismo he hecho partiendo de ideas previas que me resultaban incompletas en algún aspecto. Además, el código fuente de *Jarduino* se comparte libremente en *github* [git] en esta dirección: https://github.com/ygneo/jarduino

Desde una perspectiva funcional, hay varios objetivos que se pretenden cumplir con el desarrollo de *Jarduino*.

- **Monitorización**. Monitorizar varios factores ambientales relacionados con la agricultura: humedad del sustrato, humedad ambiental y temperatura ambiental.
- **Zonas de riego independientes**. Controlar varias zonas de riego independientes. Cada zona de riego tiene su propia configuración de riego programado, de modo que es posible aplicar esquemas de riego distintos a zonas de riego con cultivos de necesidades hídricas diversas.
- **Reprogramación**. Facilitar la reprogramación de dispositivos de la familia *Arduino* o compatibles. Para reprogramar el dispositivo se parte de de una plantilla de código paremetrizada con la configuración de las zonas de riego definidas en la interfaz de usuario de *Jarduino*.
- **Accesibilidad**. Permitir que *Jarduino* pueda ser utilizado desde varios sistemas operativos. Las herramientas que se han utilizado para su desarrollo son compatibles con los sistemas operativos actualmente más extendidos: Windows, OSX y GNU/Linux.
- **Interoperatividad**. Se han tomado decisiones de diseño que permiten la interoperativad del software y el hardware con otros sistemas. Para la intercomunicación de los componentes de software se han usando formatos estándares para el intercambio de datos como JSON. Para el hardware, el hecho de que el sistema esté diseñado para dispositvos compatibles con *Arduino* abre muchas posibilidades de interconexión con una amplia gama de dispositivos electrónicos de todo tipo.
- **Extensibilidad**. Las decisiones tomadas en torno a la intercomunicación de componentes de software y hardware, y el hecho de que el código se haya diseñado para ser extensible, y que esté disponible libremente para su uso y modificación, permite extender el sistema fácilmente para soportar nuevos requisitos. Es posible añadir interfaces para otras familias de dispositivos, o nuevas zonas de control y monitorización de otros factores que puedan influir en la gestión agrícola de un cultivo.
- **Sencillez de uso**. La interfaz de usuario de *Jarduino*, en adelante *JardUI* (*Jarduino User Interface*), se ha diseñado a partir de la experiencia profesional en el desarrollo

de interfaces web fáciles de usar, siguiendo criterios de diseño de la experiencia de usuario que potencien un uso intuitivo y cómodo.

La síntesis de estos objetivos es contar con un prototipo que permita comprobar si, conjugando la monitorización de los datos obtenidos por los distintos sensores instalados en las zonas de riego, su análisis, y la capacidad de reprogramar el dispositivo que haga de unidad de control, es posible optimizar la gestión de un huerto de tamaño doméstico mediante sucesivas iteraciones de mejora progresiva.

1.2. Alcance

No es el objetivo del desarrollo de *Jarduino* producir un producto final comercializable o listo para ser usado por un usuario sin ningún conocimiento técnico.

Las exigencias de control y monitorización de una explotación agrícola comercial quedan fuera del alcance de este proyecto. No pretende competir con las herramientas profesionales que dan soporte a ese negocio, ni con ninguna otra herramienta comercial profesional. Si acaso es una mejora con respecto a los programadores de riego para uso doméstico, al aportar control y monitorización de varias zonas de riego por separado.

El prototipo o demostrador que se presenta demuestra el funcionamiento de un sistema de monitorización y control de riego para un huerto de dimensiones domésticas, que bien puede estar instalado en recipientes de cultivo en una terraza, o en un terreno de reducidas dimensiones. Su alcance se limita a monitorizar factores ambientales que afecten al cultivo, y a controlar mediante electroválvulas el riego de las distintas zonas en que se divida el huerto.

Al ser un sistema que mantiene una independencia entre el software y el hardware, en teoría el número de zonas de riego a controlar está limitada por las restricciones del esquema de hardware que se quiera utilizar. Un simple dispositivo *Arduino Uno* — el más sencillo de su gama — puede manejar hasta 5 zonas de cultivo, una por cada entrada analógica. Una configuración más compleja, combinando un dispositivo *Arduino* con varios microcontroladores que se comuniquen con él via radio, podría multiplicar la cantidad de zonas de riego, pero requeriría adaptar el código que actualmente se ejecuta en el dispositivo, por lo que también queda fuera del alcance.

El tipo de usuario al que se orienta es un usuario aficionado a la agricultura y la autofabricación de proyectos electrónicos. Alguien que no tenga mayor pretensión que el aprendizaje y la satisfacción del trabajo realizado por sí mismo, y valore la utilidad práctica del producto como un agradable resultado secundario de un ya de por sí satisfactorio proceso de autofabricación.

En este sentido es posible, y probable, que el prototipo resultado de este proyecto no se adapte exactamente a las necesidades de la persona que se disponga a realizarlo. Está diseñado precisamente teniendo eso en cuenta, de modo que deja espacios de extensión e interoperatividad que permiten al proyecto crecer bajo necesidades y requisitos ajenos.

También quedan excluidas por ahora las necesidades especiales de los cultivos hidropónicos, tanto en cuanto a monitorización del estado del agua (oxigenación o nutrientes), como en cuanto a actuadores específicos (inyección de oxígeno y nutrientes, o renovación del agua, por ejemplo).

Por último, aunque *Jarduino* por ahora no contempla zonas de riego con necesidades especiales (como por ejemplo semilleros), este es un tipo de extensión al prototipo sencilla de realizar sobre la base construida.

1.3. Definiciones, acrónimos y abreviaturas

Arduino [WIKb] El término puede referirse a un software de código abierto (en inglés *open source*), una familia de microcontroladores, una compañía de hardware libre y/o a una comunidad tecnológica de usuarios. En este documento se suele usar la palabra en referencia a la plataforma de código abierto y/o a la familia de microcontroladores de hardware libre.

Hay varios tipos de dispositivos de la familia Arduino que se mencionan en este documento:

- *Arduino Uno* [ARDb]. Uno de los modelos más básicos de la familia *Arduino*. Es una placa basada en el microcontrolador ATmega328P [AM3a]. Tiene 14 pines digitales de entrada/salida y 6 entradas analógicas, un oscilador de cristal de cuarzo de 16Mhz, una conexión USB, una entrada de corriente, y un botón de reset. Permite su reprogramación gracías a un *header ISCP* (*In-Circuit Serial Programming*).
- *Arduino Yún* [ARDc]. Un modelo con soporte para interfaz de red (WiFi o Ethernet) basado en el ATmega32u4 [AM3b], un microcontrolador más potente que el de Arduino Uno. También incorpora el procesador Atheros AR9331m que da soporte a una distribución GNU/Linux llamada Linino OS. Además, a diferencia de Arduino Uno, tiene un puerto USB-A, un slot para tarjeta micro-SD y 20 pins de entrada/salida digitales (12 pueden ser usados como entradas analógicas).

OpenGarden es un dispositivo que amplia la funcionalidad de *Arduino* para utilizarlo en aplicaciones relacionadas con la jardineria, puesto que incorpora un circuito

integrado que permite manejar varias electrovávlulas, y sensores de humedad o temperatura, entre otros. A este tipo de dispositivos que amplian la funcionalidad de *Arduino* se les denomina *shield*, del inglés escudo.

DIY [WIKc] (*Do It Yourself*, en castellano *Hágalo Usted Mismo*). Un método de construcción, modificación o reparación de objetos sin la ayuda directa de expertos o profesionales.

Hardware Libre [WIKh] Dispositivos de hardware cuyas especificaciones y diagramas esquemáticos son de acceso público, ya sea bajo algún tipo de pago, o de forma gratuita.

Sustrato [WIKj] Superficie en la que una planta vive.

JSON [WIKg] (*JavaScript Object Notation*) Un formato de texto para el intercambio de datos de amplio uso en el ámbito del desarrollo web, pues es un formato facilmente manipulable usando el lenguaje de programación *JavaScript*. La mayoria de los grandes lenguajes de programación de alto nivel proveen librerias para leer y generar JSON.

IDE [WIKe] (*Integrated Development Environment*) es una aplicación de software que prove un conjuto de facilidades para desarrolladores de sotware.

API [WIKa] (*Application Programming Interface*) es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta libreria de software para ser utilizada por otro software como una capa de abstracción.

HTTP [WIKd] (*Hypertext Transfer Protocol*) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

URL [WIKk] (*Unified Resource Locator*) es un identificador de recursos cuyos recursos referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo. También es conocido como dirección web.

NTP [NTP] (*Network Time Protocol*) es un protocolo de red para sincronizar los relojes de los sistemas informáticos a través del enrutamiento de paquetes en redes con latencia variable.

python [WIKi] Python es un lenguaje de programación interpretado, multiplataforma y multiparadigma.

JavaScript [WIKf] Javascript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,3 basado en prototipos, imperativo, débilmente tipado y dinámico.

1.4. Visión general

Esta memoria de Proyecto de Fin de Carrera se organiza de la siguiente manera:

- El apartado *Descripción general del proyecto* muestra una perspectiva general del sistema a desarrollar, su evolución, sus funciones y restricciones, las características de los usuarios y los requisitos para futuras versiones.
- En el apartado *Desarrollo del proyecto* se explica la hipótesis de trabajo de la que parte este proyecto, la metodología de desarrollo utilizada y los requisitos del sistema. Además se presenta el análisis y diseño del sistema, detallado en las sucesivas iteraciones.
- El apartado *Análisis de mercado y selección de componentes* detalla el proceso que se ha seguido para acabar incorporando los distintos componentes que conforman el prototipo resultante de este proyecto.
- Los dos últimos apartados comprenden un resumen del trabajo realizado, y unas conclusiones finales sobre el desarrollo.
- Para finalizar se incluyen el manual de usuario y de instalación del sistema, y una serie de apéndices que amplian información relacionada con el proyecto.

Capítulo 2

Descripción general del proyecto

2.1. Hipótesis de trabajo

Este Proyecto de Fin de Carrera presenta un prototipo o demostrador que pretende validar una hipótesis de trabajo, que podemos dividir en dos afirmaciones:

1. Es posible implementar un dispositivo basado en *Arduino* que mida periódicamente factores ambientales que afecten a la necesidad hídrica de una planta y, teniéndolos en cuenta, regule el riego automáticamente.

2. El desarrollo de una aplicación con una interfaz de usuario diseñada para monitorizar los datos del dispositivo, y programarlo, permitirá iterar mejoras en la gestión del riego, optimizando el uso de agua sin afectar a la salud y el crecimiento de las plantas.

Para validar el primer punto bastaría con observar la enorme variedad de proyectos relacionados con la monitorización y/o el control de riego que se pueden encontrar en webs de aficionados al hardware y al *DIY* como *Instructable*. El demostrador que se presenta con ese Proyecto de Fin de Carrera es una versión más de estos sistemas — a los que se les denomina en inglés *Garduino*[1] — y que pretende demostrar la válidez de este primer punto de la hipótesis.

En cuanto al segundo punto, partamos de lo que queremos mejorar. *Arduino* proporciona una IDE[2] que permite escribir código, verificar su validez, compilarlo y enviarlo a un

[1]Del inglés *Gardening and Arduino*.
[2]Del inglés *Integrated Development Environment*, Interfaz de Desarrollo Integrada, ver figura 2.1

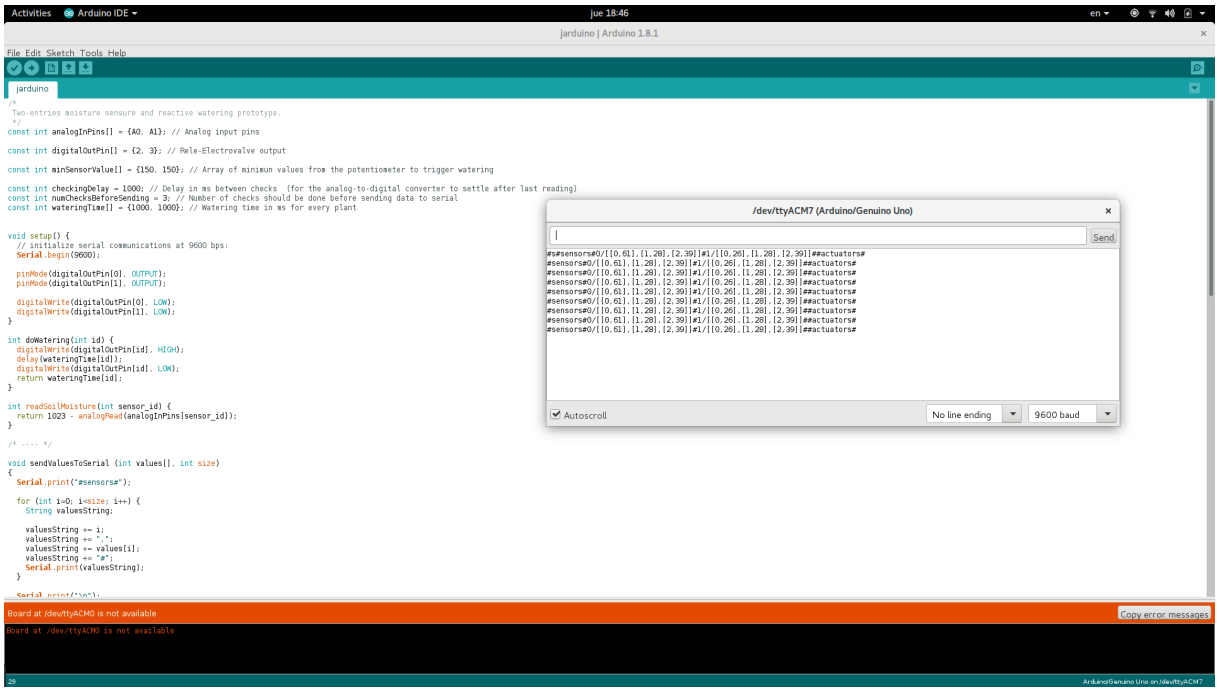


Figura 2.1: Interfaz de Usuario de la IDE de Arduino 1.8.1

dispositivo. Además permite leer lineas de texto que el dispositivo envie, y mostrarlas en una ventana. Está diseñada para programar cualquier dispositivo *Arduino* con cualquier propósito. Pero para interactuar con comodidad y eficiencia con un dispositivo con un propósito concreto resulta demasiado genérica, y acaba siendo necesario desarrollar una interfaz de usuario específica.

A grandes rasgos, para desarrollar una aplicación de gestión de riego automatizado vamos a necesitar escribir código para el dispositivo, validarlo y programar el dispositivo para que ejecute ese código. Mientras se ejecuta, el dispositivo reportará los valores que capte con sus sensores, y los eventos de riego que se hayan producido. Esta información tiene valor cuando se almacena como un histórico del comportamiento del sistema. Para representar esta información de manera que resulte útil para depurar errores, implementar mejoras y monitorizar los datos que proporciona el dispositivo, es necesario diseñar una interfaz de usuario que permita comunicarse con el dispositivo, monitorizar los datos que genere y reprogramarlo fácilmente.

Utilizando esta interfaz de usuario[3] será posible analizar la información sobre el comportamiento del dispositivo para decidir mejoras en los esquemas de riego, basadas en el histórico de datos obtenidos. Estas mejoras se pueden aplicar al dispositivo mediante una reprogramación, que nos permite iniciar un nuevo análisis, y así ir iterando mejoras progresivas.

[3]ver figura 2.2

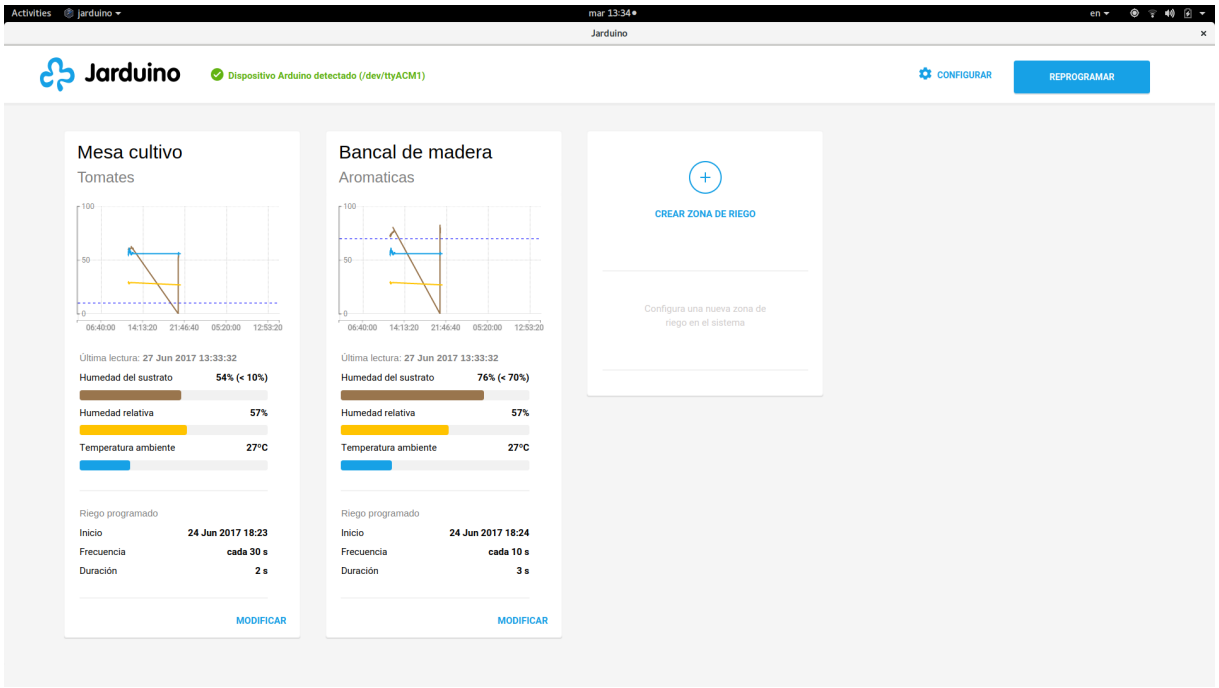


Figura 2.2: Interfaz de Usuario (JardUI)

2.2. Perspectiva del prototipo

El prototipo demostrador desarrollado en este Proyecto de Fin de Carrera es un sistema de riego automático reprogramable que permite manejar varias zonas de riego con necesidades hídricas distintas, al que hemos denominado *Jarduino*.

Su desarrollo comprende una aplicación software que permite gestionar y monitorizar el sistema de riego, y el montaje de una serie de componentes hardware basados en *Arduino* y *OpenGarden*[4], que toman lecturas de sensores ambientales y pueden abrir o cerrar varias electroválvulas para iniciar o parar el riego. A esta serie de componentes hardware se les denomina en este documento, de forma genérica, *dispositivo* (ver figura 2.3).

[4]Una placa de extensión o *shield* para Arduino, ver https://goo.gl/4ExbFF

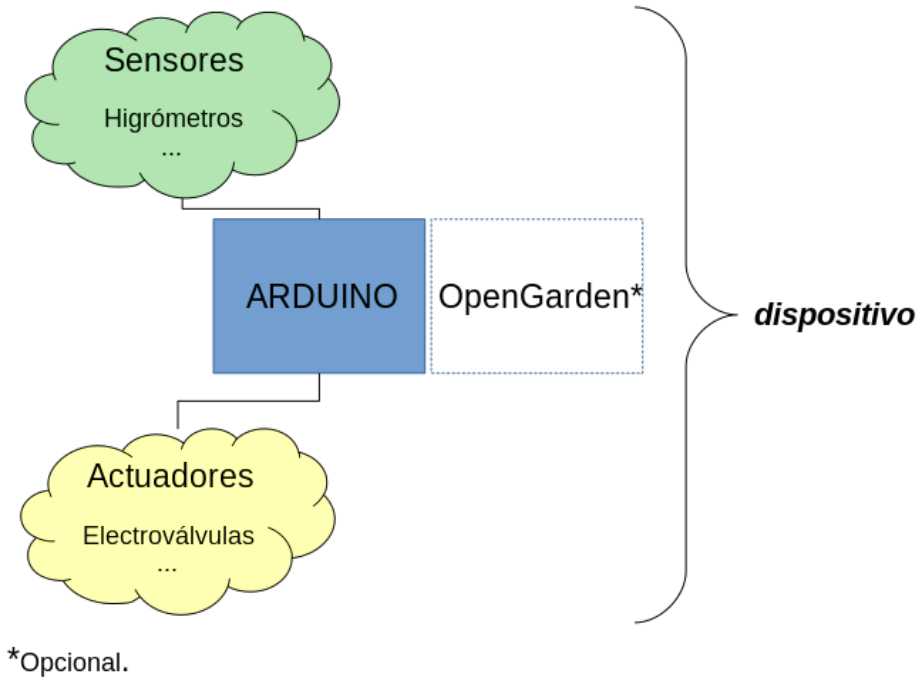
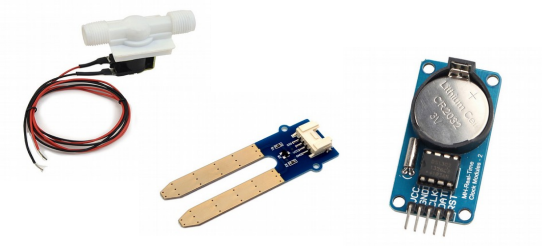


Figura 2.3: Ilustración del concepto de *dispositivo* en este documento.

El hardware del prototipo demostrador es sólo una de las posibles implementaciones hardware con las que es compatible el software de *Jarduino* (ver figura 2.4). Es posible implementar una versión más simple basada en *Arduino Uno*, y una versión con conexión inalámbrica basada en *Arduino Yún*.

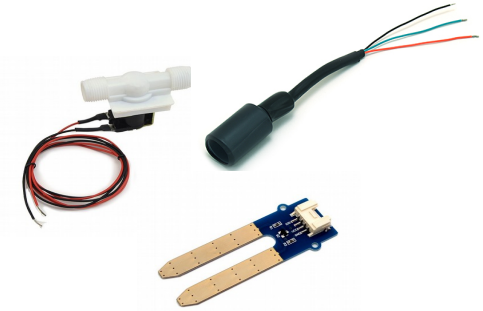
Arduino Uno



Hasta 6 electroválvulas.
Hasta 6 sensores humedad sustrato.
1 módulo Real Time Clock (p.ej. DS1302)

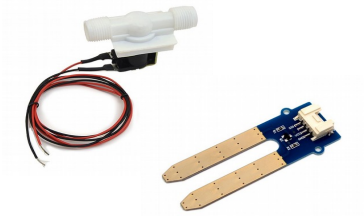
PROTOTIPO DEMOSTRADOR

Arduino Uno
+OpenGarden



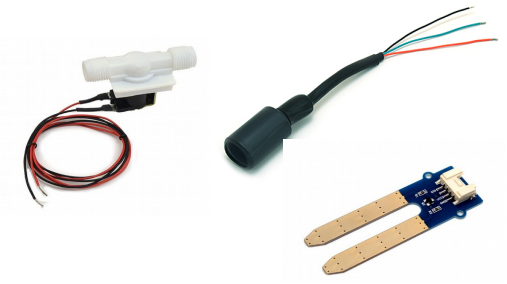
1 sensor temperatura y humedad ambientales.
Hasta 7 electroválvulas.
Hasta 7 higrómetros.

Arduino Yún



Hasta 6 electroválvulas.
Hasta 6 sensores humedad sustrato.

Arduino Yún
+ OpenGarden



1 sensor temperatura y humedad ambientales.
Hasta 7 electroválvulas.
Hasta 7 higrómetros.

Figura 2.4: Posibles implementaciones hardware de Jarduino.

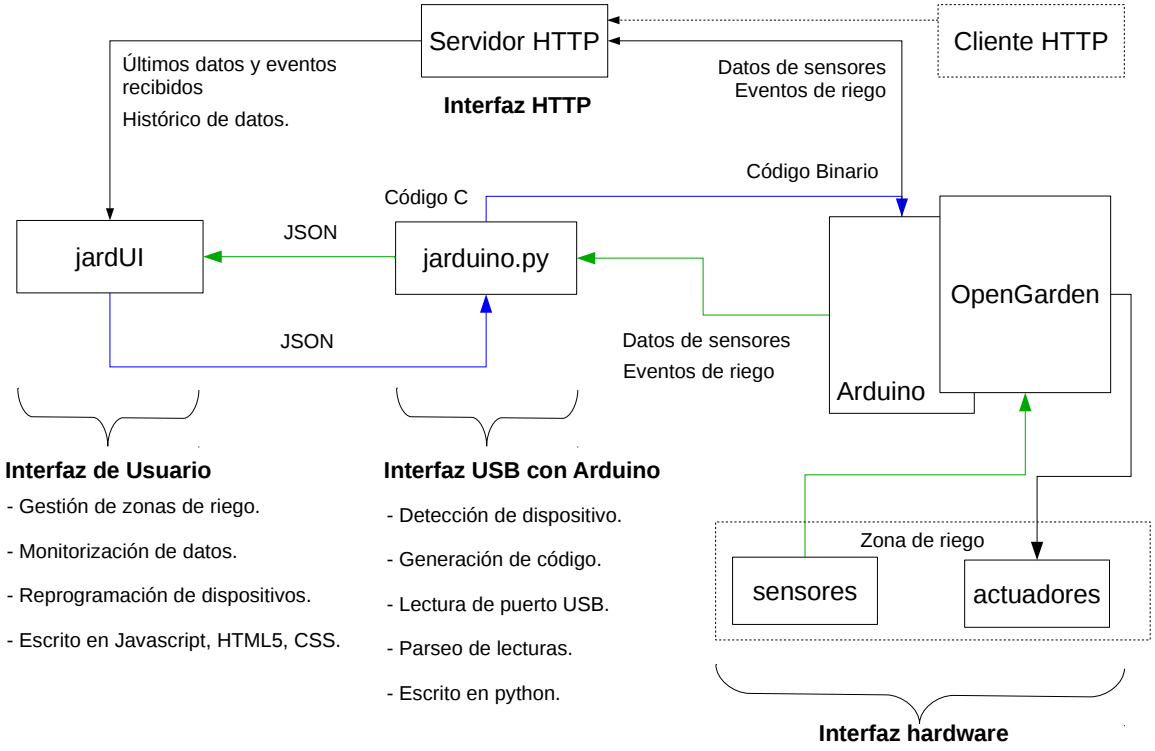


Figura 2.5: Perspectiva global del prototipo demostrador presentado

Este proyecto no es continuación de ningún otro producto o proyecto previo, pero sin duda está influenciado por otros proyectos de la comunidad DIY que proponen prototipos de monitorización y/o automatización de riego basados en Arduino[5].

Se puede tener una perspectiva global (ver figura 2.5) del prototipo presentado entendiénlo como un conjunto de interfaces especializadas que interactúan y comparten información entre sí: un interfaz hardware, con *Arduino* y *OpenGarden*, que lee los sensores ambientales y maneja las electroválvulas, y tres interfaces software: una para interactuar via USB con *Arduino*, tomar lecturas y reprogramarlo; otra que hace de interfaz con el usuario, y un servidor y cliente de HTTP para recibir y almacenar a través de internet los datos que genere el dispositivo.

Interfaz hardware: Arduino y OpenGarden

El software que toma las lecturas de los sensores ambientales (humedad del sustrato, humedad ambiental y temperatura ambiental), controla las electroválvulas, y envia los

[5]Ver sección 2.7, página 28

datos que generan, se ejecuta en la plataforma de hardware libre *Arduino*.

Para facilitar la programación de este software, y la integración del dispostivo *Arduino* con sensores ambientales y electroválvulas—sin que nea necesario montar ninguna electrónica auxiliar para interactuar con ellos—, se ha utilizado además *OpenGarden*, una placa de extensión para *Arduino*.

La entrada de esta interfaz son los datos recogidos directamente por los sensores ambientales conectados a *OpenGarden*, además del historial de riego de cada zona de riego.

La salida puede tomar dos formas: para un dispositivo conectado por USB será una linea de texto formateada de manera que se pueden diferenciar los datos provenientes de los sensores y el historial de riego para cada zona de riego. Si el dispositivo está conectado a una interfaz de red, se envia estos datos en una petición HTTP a un servidor que los lee y almacena.

Interfaz USB con Arduino

Un software de linea de comandos que permite interactuar con *Arduino*, y que es invocado desde la interfaz de usuario para delegar en él las interacciones con el dispostivo.

Proporciona las operaciones de detección del dispostiivo *Arduino*, lectura de los datos que envie el dispositivo via USB y compilación y envio del código al dispositivo (reprogramación).

Su entrada son los datos de los sensores ambientales y el historial de riego proporcionados por el dispositivo a través del puerto USB.

Su salida son esos mismos datos estructurados en un formato JSON, adecuado para su consumo en la interfaz de usuario.

Además, cuando se utiliza para reprogramar el dispositivo, la salida es el código fuente compilado que se envia al dispositivo.

Interfaz HTTP

Esta interfaz consiste en dos piezas de software: un servidor HTTP y un cliente HTTP.

Al servidor HTTP se pueden enviar datos ambientales y de historial de riego desde un dispositivo *Arduino* conectado a un interfaz de red (ethernet o WiFI), de manera que no sea necesario mantener el dispositivo conectado a un ordenador a través de un cable USB para recibirlos. Se encarga de almacenar estos datos en una base de datos.

Por otro lado, la interfaz de usuario puede solicitar al servidor HTTP los datos recibidos a partir de cierta fecha y hora.

El cliente HTTP emula el envio de datos desde un dispositivo *Arduino* conectado a un interfaz de red. Se utiliza para facilitar el desarrollo del prototipo cuando no sea posible contar con un dispositivo conectado, y permite presentar el prototipo demostrador sin depender del estado la interfaz de red disponible durante la presentación.

La entrada de esta interfaz son los envios HTTP que realiza el dispositivo conectado o el cliente que lo emula.

La salida es el conjunto histórico de datos obtenidos a partir de cierta fecha y hora.

Interfaz de usuario

La última capa de interfaz, diseñada para que el usuario interactúe directamente con ella, presentando los datos e interacciones al nivel de abstracción adecuado para trabajar con ellos sin necesidad de conocer detalles de implementación. Por ejemplo, el nivel de humedad del sustrato se presenta en un escala porcentual, en lugar de mostrar el valor exacto (de 0 a 1023) que lee la entrada analógica del dispositivo.

Es una aplicación software que se instala localmente, y que muestra en un gráfico interactivo con los valores históricos de los distintos sensores ambientales, los umbrales que se les aplican para decidir si se riega o no, y los eventos de riego, para cada zona de riego. Permite además configurar distintos esquemas de riego para cada una de estas zonas y reprogramar el dispostivo en base a ellos.

La entrada de esta interfaz puede provenir de los datos enviados directamente por el dispositivo al puerto USB, o del servidor HTTP.

La salida es la representación de esos datos de manera que resulte sencillo y útil para el usuario manejarlos.

Cuando se reprograma el dispositivo, la salida son los parámetros de configuración que necesita la interfaz con *Arduino* para parametrizar y compilar el código que ejecutará el dispositivo.

2.3. Funciones del prototipo

El prototipo desarrollado para este Proyecto de Fin de Carrera sirve como demostrador de un sistema de gestión de riego automatizado. Hemos visto que está compuesto de una serie de interfaces que manejan pequeños conjuntos de operaciones especializadas. Cada una de estas operaciones tiene poco valor funcional por si mísmas, de modo que en esta sección describe las funciones del prototipo desde una perspectiva global, puesto que dividirlas por las que pertenece a cada componente del sistema no tendría mucho valor

ilustrativo.

Se puede observar una visión esquemática de las interacciones entre las interfaces de *Jarduino* para realizar algunas de estas funciones en la figura 2.6.

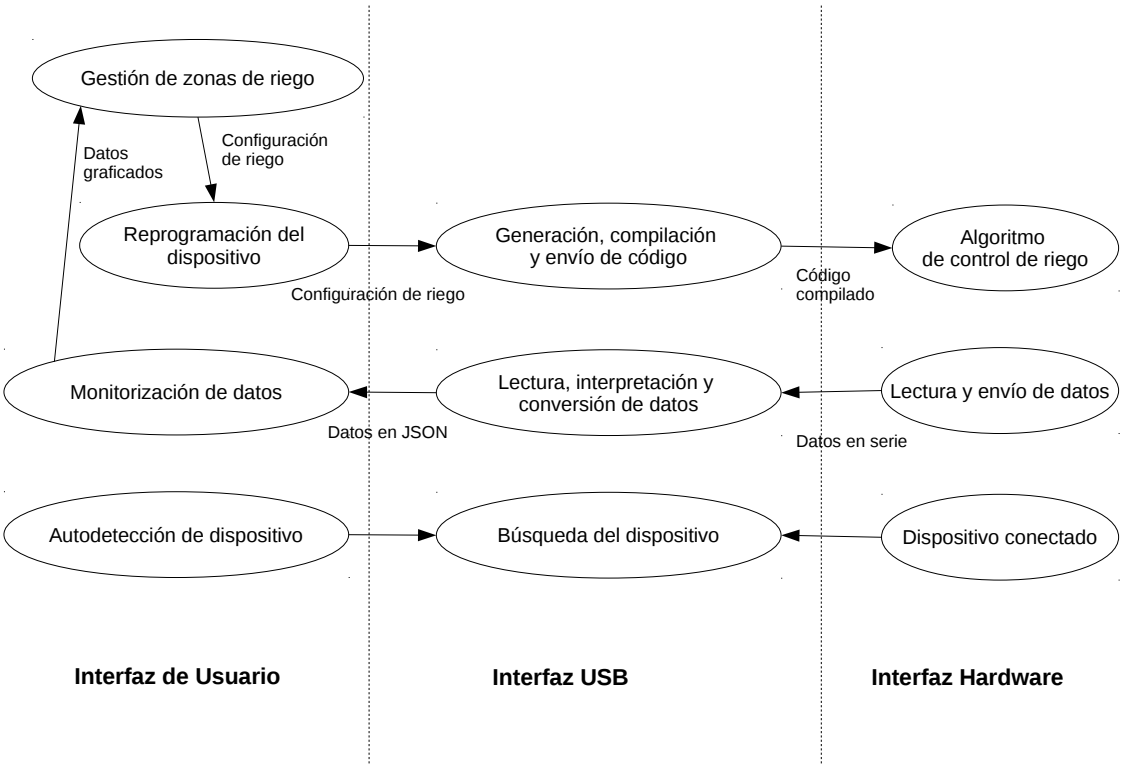


Figura 2.6: Esquema de interacciones del prototipo presentado (Jarduino Uno)

Por lo tanto la lista de funciones que se presenta a continuación no pretende ser exhaustiva ni detallista, sino ilustrativa al nivel de funcionalidad necesario para tener una idea de lo que ofrece el prototipo presentado.

- **Administración de zonas de riego independientes**.

  El prototipo presentado permite manejar de manera independiente hasta 2 zonas de riego diferenciadas en cuanto a las necesidades hídricas de sus cultivos. Pero el software desarrollado no está diseñado con ningún límite en el número de zonas que puede administrar. Permite crear, configurar y eliminar zonas de riego.

  Se podría por lo tanto conectar una tercera electroválvula y un tercer higrómetro al dispositivo del prototipo, y bastaría con crear una nueva zona de riego en la interfaz de usuario, y esta empezaría a recibir datos tan pronto como estuvieran disponibles.

Cada zona de riego tiene su propio nombre y descripción, una configuración de riego programado, y unos umbrales de riego en base a los datos tomados por los sensores ambientales. Por ejemplo, se puede definir un cierto umbral de riego basado en la humedad del sustrato, y el dispositivo solo regará si la humedad del sustrato está por debajo de ese umbral.

- **Riego automatizado reprogramable con umbrales de riego**.

  Para regar adecuadamente no basta con conocer la humedad del sustrato en un momento dado y decidir si se riega o no. Además es necesario saber cuánto y en qué momento se debe regar para optimizar el uso de agua y su máximo aprovechamiento por parte del cultivo. Por ejemplo, será necesario regar durante menos tiempo en invierno que en verano, y en verano mejor regar por la noche que durante el día para evitar evaporación.

  Cada una de las zonas de riego puede tener su propia configuración o esquema de riego programado: fecha y hora de inicio, frecuencia de riego y duración del riego. Además se pueden definir umbrales de riego basados en la humedad del sustrato, la humedad o la temperatura ambientales. Estos umbrales determinan si el riego debe producirse cuando está programado, o no.

  Una vez programado el dispositivo con esta configuración, el sistema comenzará a comprobar si debe regar a la fecha y hora indicadas. Si debe regar, lo hará durante el tiempo indicado, y volverá a hacerlo cuando indique la frecuencia de riego (por ejemplo cada 8 horas).

- **Monitorización de datos ambientales**.

  Los datos tomados por los sensores ambientales del prototipo que se presenta son humedad del sustrato, humedad y temperatura ambientales. Estos datos se pueden leer en tiempo real a través de una conexión USB con el dispositivo, o través de la red local o Internet.

  Cada zona de riego tiene su propio sensor de humedad del sustrato, o higrómetro. En el demostrador que se presenta los sensores de humedad ambiental y temperatura son compartidos por todas las zonas de riego.

  La interfaz de usuario muestra los datos tomados en la última lectura, y los representa gráficamente en un diagrama de lineas interactivo, que se actualiza automáticamente con los últimos datos recibidos.

- **Histórico de datos ambientales y eventos**.

  Los datos ambientales obtenidos por los sensores, junto con su fecha de lectura, y los eventos de riego que se hayan producido, son almacenados en una base de datos que registra el histórico.

  Cada zona de riego tiene su propio conjunto histórico de datos.

Este histórico de datos ambientales y eventos de riego puede ser consultado a través de la interfaz de usuario, mediante un gráfico interactivo que facilita su consulta, filtrado y análisis, con la intención de mejorar la configuración de las distintas zonas de riego.

- **Reprogramación multidispositivo**.

  Una vez configuradas las distintas zonas de riego, la interfaz de usuario permite reprogramar el dispositivo con la configuración definida. Para hacerlo se utiliza una plantilla de código parametrizada, escrita para ser ejecutada en un dispositivo de la familia *Arduino*. Se han desarrollado varias plantillas de código para soportar distintas implementaciones del dispositivo (ver 3.4.1.2.2).

  La interfaz de usuario permite elegir qué plantilla de código utilizar, además una serie de parámetros de configuración que afectan al comportamiento del dispositivo:

  - *Destino de los envíos de datos*

    Los datos se pueden enviar a través de USB, o a través de un interfaz de red conectado al dispositivo. En este segundo caso se ha de introducir la URL hacia la que se enviarán los datos.
  - *Frecuencia de envio de datos.*

    Cada cuanto tiempo se han de enviar datos al destino de datos definido en el punto anterior.
  - *Número de lecturas entre envíos.*

    El número de veces que se pide datos a los sensores entre cada envío de datos. En cada envio, el dispositivo calcula una media de todos los valores tomados.
  - *Plantilla de código.*

    El fichero desde dónde se cargará la plantilla de código. Por defecto se carga la plantilla de código para *Arduino Uno* con *OpenGarden*.
  - *Esquema de conexión de los sensores.*

    El sistema decide automáticamente un esquema de conexión de los sensores al dispositivo, que muestra dónde van conectadas las electroválvulas y los sensores de cada zona de riego. Se puede modificar las conexiones de entrada y salida de ese esquema de conexión, dentro de las posibilidades que permita el dispositivo para el que esté escrita la plantilla de código.

  Además de estos parámetros, se aplican los parámetros de configuración de riego de cada zona de riego, definidos en la interfaz de usuario. Con todos estos parámetros se rellena la plantilla de código que el usuario haya seleccionado. Puede observarse una descripción gráfica de cómo funciona, a grandes rasgos, la función de reprogramación en la figura 2.7.

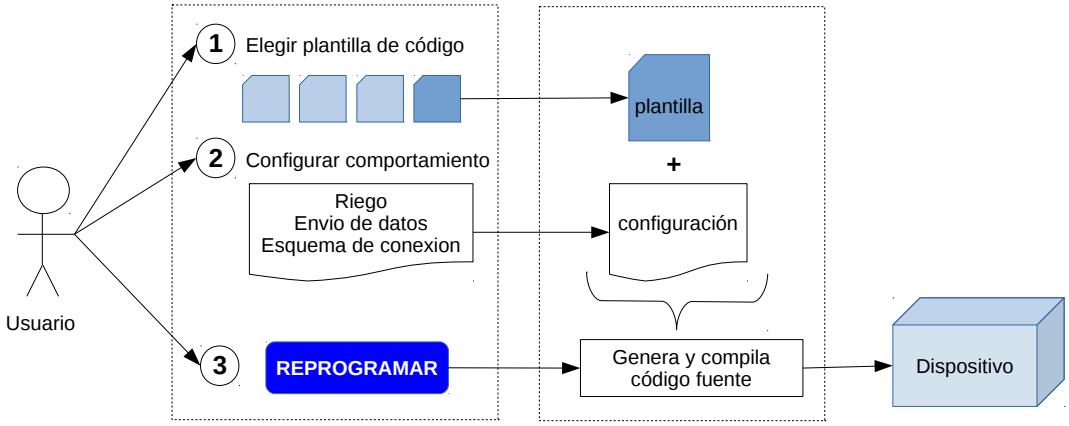


Figura 2.7: Esquema de funcionamiento de la reprogramación multidispositivo

- **Autodetección de dispositivo via USB**.

  La interfaz de usuario muestra si hay un dispositivo *Arduino* conectado al puerto USB del PC dónde se ejecuta la interfaz de usuario.

2.4. Características de los usuarios

El sistema que presento en este Proyecto de Fin de Carrera, *Jarduino*, está diseñado pensando en un tipo de usuario muy concreto, y no tiene ninguna pretensión de uso generalista o comercial.

Este proyecto ha nacido gracias a los diseños de prototipos que la comunidad de *makers*[6] ha compartido, dentro de una difusa categoría que a veces se denomina *Garduino*, por *Gardening* (Jardinería) y *Arduino*.

[6]Término inglés que literalmente significa *hacedores*, muy polisémico. En este texto se toma con el significado de aficionados al DIY (Ver 1.3, página 10)

Jarduino está dirigido a esa misma comunidad de usuarios interesados en alguna faceta de la jardineria y aficionados a divertirse montado sus propios prototipos de monitorización y control de riego con *Arduino*.

Por lo tanto, se trata de personas con los suficientes conocimientos tecnológicos para montar una copia o una versión de *Jarduino*, es decir, deben ser capaces de instalar el software en su sistema, y también de montar el hardware al que lo vayan a conectar, sin ayuda. Ese es precisamente el espíritu del DIY.

El sistema está por lo tanto diseñado para un aficionado doméstico, interesado en ocupar su ocio montando y utilizando un prototipo como el que presento, adaptándolo a sus posibilidades y necesidades.

2.5. Restricciones generales

El prototipo demostrador que se presenta es sólo una de las posibles implementaciones de *Jarduino*. Es posible ampliar y modificar algunas de sus características—por ejemplo, añadir una zona de riego, o modificar el esquema de conexión—e implementar una versión de *Jarduino* que maneje un número distinto de zonas de riego. Y dado que el código fuente es de libre uso, es posible modificarlo para que trate con otro tipo de situaciones para las que no estuviera diseñado inicialmente, pero para las que algún aspecto del sistema, por ejemplo la interfaz de usuario, pueden resultar útiles. En este sentido, el sistema está restringido a la capacidad e imaginación de quien quiera modificarlo o ampliarlo, reutilizando lo que le interese.

El ámbito de uso al que se dirige este prototipo es el doméstico. Pretende ser una versión mejorada de un programador de riego de uso doméstico, de modo que aunque es posible, por ejemplo, ampliar el sistema con dispositivos para controlar o medir el caudal de agua consumido en cada riego, este nivel de control no suele ser necesario para el gasto de agua que suele suponer un cultivo doméstico. En cualquier caso, utilizando *Jarduino* es posible optimizar el uso de agua de riego, sólo que no con la precisión que daría el uso de un sistema de control de caudal.

La elección de *Arduino* como plataforma de desarrollo para el dispositivo hardware responde a varios motivos. Para empezar, se valora la amplia oferta y la facilidad de obtención de componentes hardware como sensores y placas de extensión (*shields*) que permiten realizar todo tipo de tareas. En concreto la existencia de placas de extensión especializadas como *OpenGarden* hace que *Arduino* sea una plataforma que encaje muy bien con las necesidades de este proyecto. El hecho de que además se trate de una plataforma de desarrollo libre y gratuita, con una comunidad de usuarios muy activa, y una gran cantidad de documentación técnica disponible de manera también libre y gratuita—con la

que ya se han desarrollado un buen número de proyectos similares[7]—augura la obtención de buenos resultados.

2.5.1. Restricciones del hardware

Se ha diseñado *Jarduino* para ser compatible con varios dispositivos. El más sencillo de ellos es un *Arduino Uno*, al que hay que añadir un RTC (*Real Time Clock*) que permita programar el horario y enviar las fechas y horas de las lecturas. Además de la electrónica auxiliar que sea necesaria para manejar los higrómetros y las electroválvulas.

El prototipo demostrador que se presenta se ha desarrollado para un dispositivo *Arduino Uno* ampliado con un dispositivo *Open Garden*, que facilita el desarrollo de software y el hardware de un sistema de jardineria basado en *Arduino*. En concreto para este prototipo demostrador han sido útiles estas características de *OpenGarden*:

- Al conectarse sobre un dispositivo *Arduino* amplia su funcionalidad, proporcionando un circuito integrado que permite interactuar con varios sensores para medir la humedad del sustrato, temperatura y humedad ambiente, entre otros.
- Permite manejar hasta 3 electroválvulas o motores, para el manejo de las cuales tiene integrada la electrónica necesaria. Esto permite que el dispositivo sea versátil, y puede utilizarse para abrir o cerrar una o varias electroválvulas, o un aspersor de riego, o incluso usar un servo-motor para, por ejemplo, abrir o cerrar las ventanas de un invernadero para regular su temperatura.
- Proporciona una libreria o API para interactuar con los sensores y las electróválvulas/motoroes, a un nivel de abstracción adecuado para manejarlos con facilidad. Por ejemplo, la libreria para interactuar con el sensor de temperatura devuelve los valores de las lecturas ya convertidos a grados centígrados.
- Incluye un reloj RTC integrado, lo que permite la programación horaria del riego.

También es posible utilizar *Jarduino* con un dispositivo *Arduino Yun*, lo cual permite enviar los datos a través de una interfaz de red. Pero aunque *Yun* permite ser reprogramado remotamente, no se ha desarrollado esa posibilidad para este prototipo, limitando la reprogramación a una conexión USB.

[7] Ver 2.7, página 28

2.5.2. Restricciones del software

El software del dispositivo está escrito para ser ejecutado en las versiones de *Arduino* compatibles con este prototipo (*Uno*, *Uno + OpenGarden* y *Yun*). El código que se ha escrito para ejecutarse en *Arduino Uno* solo puede manejar sensores de humedad del sustrato y electroválvulas. Para manejar el resto de sensores (humedad y temperatura ambientales) se puede ampliar el dispositivo con *OpenGarden* o extender el código para que maneje los sensores adicionales.

La aplicación de usuario se instala localmente y es compatible con Windows, OSX y GNU/Linux. El servidor y el cliente web son compatibles con cualquier sistema operativo capaz de ejecutar un servidor HTTP escrito en python.

2.6. Suposiciones y dependencias

2.6.1. Del hardware

La interfaz de usuario de *Jarduino* depende de una serie de componentes hardware para ser funcional. Es necesario contar con uno de los dispositivos de la familia *Arduino* compatibles (*Uno*, *Uno + OpenGarden* o *Yun*), y con un sensor de humedad o higrómetro por cada zona de riego. Si además se quiere registrar la humedad y temperatura ambientales, es necesario contar con los correspondientes sensores.

Los higrómetros, y el resto de sensores que se conecten al dispositivo se suponen calibrados y aunque su calibración quede fuera del alcance del prototipo presentado, es posible utilizar la monitorización gráfica de los datos para facilitar el calibrado manual.

2.6.2. Del software

El sofware escrito para ser ejecutado en el dispositivo depende de la disponibilidad de las librerias utilizadas para su desarrollo. El sistema desde el que se compile el código para el dispositivo debe contar con las librerias adecuadas. En función de la plantilla de código que se utiliza, las dependencias varian. La plantilla para *Arduino Uno* solo depende de la libreria estándar que provee la plataforma de desarrollo *Arduino*, pero la plantilla de código para *Arduino Uno + OpenGarden* requiere tener instalada en el sistema la libreria *OpenGarden*, libre y gratutimente disponible.

Si se utiliza un dispositivo *OpenGarden*, como se hace para el prototipo demostrador

que se presenta durante la defensa de esta práctica, se cuenta con un API para facilitar la programación del algoritmo de control de riego, pero solo se cuenta con esa API o librería, el código fuente de ese algoritmo se debe desarrollar, puesto que *OpenGarden* no proporciona más que la propia librería y algunos ejemplos de código.

En cuanto a la aplicación de linea de comandos para interaccionar con *Arduino*, sus dependencias son tener instalado python 2.7 en el sistema, e instalar el único paquete de python requerido: *pyserial*. Además de es necesario tener instalado *Arduino Makefile* [ARDa].

La aplicación de interfaz de usuario es un poco más exigente en dependencias. Está escrita en JavaScript, y require instalar una serie de librerias escritas en ese lenguaje programación. Las más importantes son:

- *React*, una libreria para construir interfaces de usuario capaces de soportar actualizaciones de datos frecuentes.
- *python-shell*, una libreria para interaccionar con una aplicación de linea de comandos.
- *Rickshaw*, una libreria para la creación de gráficos basados en conjuntos de datos leidos en tiempo real.

2.7. Comparativa con otros prototipos de control de riego

Se ha hecho mención varias veces en este documento a que *Jarduino* se ha inspirado en otros prototipos que pretenden automatizar el control de riego con *Arduino*. Se trata de proyectos que relacionan la jardinería—entendida de un modo genérico—y *Arduino*, y que explícita o implícitamente se identifican con el concepto *Garduino* (*Gardening and Arduino*).

Una búsqueda en internet del término *Garduino* nos devuelve decenas de resultados que apuntan a proyectos con objetivos similares. Todos tienen un objetivo mínimo en común: utilizar *Arduino* para decidir si se debe regar o no un sustrato, dependiendo del valor de humedad que detecte un higrómetro conectado al dispositivo. Y todos tienen diferencias, a veces mayores, a veces sólo matices, a la hora de abordar esta tarea.

Por supuesto, además de proyectos de este tipo basados en *Arduino*, los hay basados en muchas otras plataformas como *Rasbperry Pi*. Escribir una comparativa exhaustiva de todos estos sistemas se escapa a la intención de esta sección, que no pretende abarcar

todas los desarrollos que se han emprendido relacionados directa o indirectamente con el control de riego.

La comparativa que se presenta compara *Jarduino* con dos ejemplos de proyectos de control de riego basados en *Arduino*:

- *Garduino.* [GARb] Uno de tantos proyectos que se autodenominan *Garduino.* El objetivo es automatizar el riego y la iluminación de las plantas, utilizando sensores de humedad y luminosidad. Como sistema de de riego utiliza una bomba de agua sumergida en un recipiente lleno de agua, que activa desde el *Arduino.* El control de la iluminación escapa al interés de esta comparativa, que se centra en el control de riego.
- *Gard-A-Water* [GARa]. Un proyecto de control de riego con *Arduino* que pretende automatizar el riego de un pequeño terreno, manejando varias electroválvulas a través de un array de relés. Su código fuente abre el riego si el sensor de humedad detecta un valor por debajo de un umbral fijo establecido. Este prototipo permite activar varios relés, pero no definir un umbral de riego distinto para cada zona de riego, de modo que en la práctica sólo maneja una zona de riego.

La comparativa analiza estas características:

- *Zonas de riego.* Cuántas zonas de riego diferenciadas, es decir, con su propio sensor o sensores, su propia electroválvula y su propio umbral de riego.
- *Riego programado.* Indica si se puede programar el inicio, la frecuencia y la duración del riego.
- *Sensores.* Qué sensores maneja.
- *Conexión.* Cómo es posible conectar el dispositivo a un ordenador.
- *Interfaz de Usuario.* Se indica si cuenta con una interfaz de usuario diseñada para la gestión del riego.
- *Reprogramable.* El prototipo provee algún sistema que facilite la reprogramación de los umbrales de riego.

	Zonas de riego	Riego Programado	Sensores	Conexión	Interfaz de usuario	Reprogramable
Garduino	1	✗	Humedad sustrato Temperatura ambiental Luminosidad	USB	✗	✗
Gard-A-Water	Múltiples(*)	✗	Humedad sustrato	USB	✗	✗
Jarduino	Múltiples	✓	Humedad sustrato Humedad ambiental Temperatura ambiental Luminosidad	USB Wifi opcional Ethernet opcional	✓	✓

(*) Su hardware permite varias zonas pero sólo provee un ejemplo de código fuente para manejar una.

2.8. Requisitos para futuras versiones

Las posibilidades de extensión de *Jarduino* son amplias. A medio y largo plazo el desarrollo pueden acabar llevando el prototipo a usos no previstos en esta fase de desarrollo. *Jarduino* puede evolucionar como un sistema personal y doméstico para gestión de huertos y jardines en entornos urbanos, como una aplicación de *machine learning* capaz de adaptar autónomamente esquemas de riego al contexto ambiental, como un asistente para la gestión de cultivos, entre otras muchas posibilidades.

Todas estas posibilidades resultan difusas en el momento actual del desarrollo del prototipo. Es posible concretar más acotando la mirada al desarrollo a corto plazo. En este ámbito, se pueden identificar dos espacios de mejora: en lo que concierne a la operativa con *Arduino*, y en lo que concierne a la interfaz de usuario y la mejora de la gestión de cultivos.

- *Extension de la operativa del dispositivo.*

  - *Añadir a la plantilla de código de Arduino Uno soporte para sensores.*

    Dependiendo del dispositivo con el que se utilice *Jarduino*, se pueden realizar más o menos operaciones. Utilizado con *Arduino Uno* el código del dispositivo es un poco limitado, puesto que solo permite trabajar con sensores de humedad del sustrato.

    Seria necesario extender el código escrito para *Arduino Uno* y dotarlo de capacidad de interacción con sensores de humedad y temperatura ambientales, de modo que sea posible tomar ese tipo de datos sin necesidad de contar con un módulo *Open Garden*.

  - *Añadir nodos comunicados inalámbricamente.*

    Incluso dentro del ámbito de uso doméstico en el que se circunscribe *Jarduino*, puede resultar engorroso, si se manejan más de dos zonas de riego, o estás están muy distanciadas, tener que extender cableado para colocar los higrómetros.

    Para mejorar esta situación, se podría utilizar un dispositivo arduino con soporte para comunicación via radio al que a su vez se conectarán otros microcontroladores menos potentes y más baratos, de la misma familia de controladores AVR [AVR] utilizados por *Arduino*. Estos microcontroladores, con sus correspondiente módulo de radiofrecuencia, pueden enviar datos a *Arduino* sin necesidad de cables, y este a su vez puede enviarlos al destino de datos que se

le haya configurado, por ejemplo a un servidor HTTP a través de la interfaz de red.

- *Extensión de la interfaz de usuario como herramienta de gestión de cultivos.*

  - *Fichas de riego.*

    A medida que se utiliza *Jarduino* para gestionar el riego de un cultivo, se pueden ir deduciendo qué esquemas de riego aplicar en qué circunstancias para obtener buenos resultados.

    Esta información que se deduce del uso del sistema se podría introducir en unas fichas de riego para cada tipo de cultivo, de modo que sea posible utilizarlas como plantillas para aplicar en sucesivas temporadas, cuando se vuelva a sembrar el cultivo en cuestión.

  - *Reprogramación remota.*

    El modelo *Arduino Yun* tiene unas interfaces de red Ethernet y WiFi, que permiten conectar el dispositivo a una red LAN o WAN. Es posible usarlas para reprogramar el dispositivo, pero para el prototipo demostrador que se presenta no se ha desarrollado esta posibilidad, permitiendo únicamente la reprogramación via USB.

    La reprogramación no es un evento que sucede con mucha frecuencia, pero es obvio que resulta molesto no poder realizarla remotamente si resulta necesario.

    Para implementar esta funcionalidad se han de tener en cuenta cuestiones de seguridad que superan el alcance de este Proyecto de Fin de Carrera

    Una vez analizadas e implementadas las precacuiones necesarias, se podría añadir la posibilidad de reprogramar el dispositivo remotamente.

Capítulo 3

Desarrollo del proyecto

3.1. Metodologia de desarrollo

La estrategia que un ingeniero del software debe incorporar a las fases genéricas del proceso de desarrollo del software se conoce como modelo de proceso de desarrollo de software.

El modelo de proceso de desarrollo de software escogido para la realización del este Proyecto de Fin de Carrera es el modelo incremental e iterativo.

En la primera iteración se desarrolla un prototipo que sirve de prueba de concepto para valorar la viabilidad del proyecto. Durante esta iteración se implementa un dispositivo basado en *Arduino* que permite manejar dos higrómetros caseros y dos electroválvulas reutilizadas para controlar el riego en base a dos umbrales de humedad que determinan la apertura o cierre de las elecroválvulas. Al prototipo resultante de esta primera iteración se le denomina *Jarduino Cero*.

La segunda iteración se dedica a desarrollar un prototipo demostrador que, partiendo de los resultados de la iteración anterior, incorpore varias mejoras al funcionamiento. Se añade la posibilidad de manejar más de un par de higrómetros y elecroválvulas, y se incorporan sensores de humedad y temperatura ambientales. Se desarrolla además una interfaz de usuario para la monitorización y configuración de zonas de riego, que permite seguir la evolución de los datos ambientales, definir una programación de riego y varios umbrales de riego, y la reprogramación del dispositivo en base a esta configuración. Al prototipo demostrado de esta segunda iteración, que es el que se presenta en la defensa de esta práctica, se le denomina *Jarduino Uno*.

Para el diseño web se utiliza un modelado UML con extensiones para diseñar apli-

caciones Web [Con03] y para el diseño del software se utiliza un modelado UML con un análisis mediante diagramas de clases.

3.2. Proceso de desarrollo

3.3. Primera iteración: Jarduino Cero

El objetivo de esta primera iteración es desarrollar un prototipo de control de riego basado en *Arduino* que sirva de prueba de concepto para validar la viabilidad del desarrollo de un dispositivo que controle dos zonas de riego independientes.

Una zona de riego puede ser una maceta, una mesa de cultivo, o cualquier otro recipiente o pequeña extensión de terreno dónde se pretenda mantener un cultivo. Esta primera iteración lee la humedad del sustrato de dos zonas de riego, y decide su riego en base a un umbral de humedad para cada una de ellas.

El desarrollo de esta primera iteracion se ha basado en dos premisas:

1. *Ahorro de recursos*. Tender a la reutilización y autofabricación de todos los componentes que sea posible, para optimizar el uso de recursos en esta primera fase, en la que estamos valorando la viabilidad del prototipo con la mínima inversión posible.
2. *Desarrollo de la funcionalidad mínima necesaria*. Se prioriza el desarrollo de un prototipo que riegue de manera independiente dos recipientes en base al nivel de humedad del sustrato que contengan, dejando por el momento aparcadas la parametrización del código que ejecuta el dispositivo o el diseño e implementación de una interfaz de usuario.

En la figura 3.1 se muestra una perspectiva global del prototipo desarrollado en esta primera iteración. Se indican ciertas restricciones que aplican a este primer prototipo, cómo el hecho de que la conexión a los sensores y las electroválvulas sea fija, pues no se cuenta aún con un código parametrizable con el que se puede modificar automáticamente el esquema de conexión.

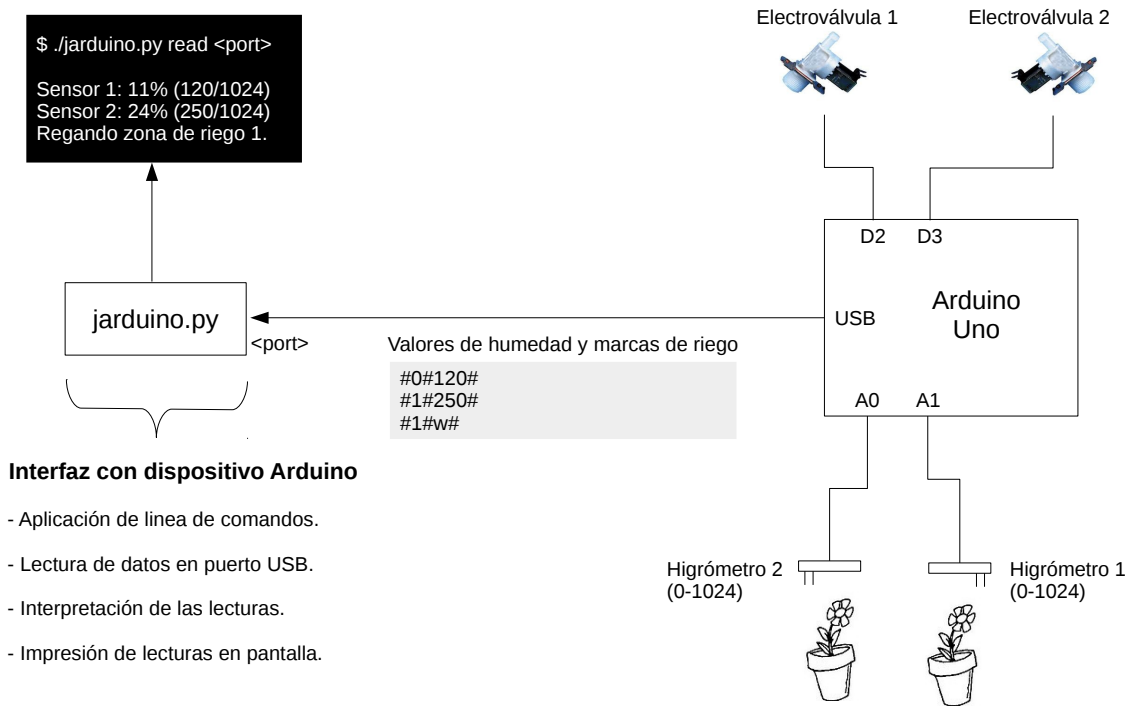


Figura 3.1: Perspectiva global del prototipo de la primera iteración (Jarduino Cero)

3.3.1. Especificación de los requisitos del sistema

3.3.1.1. Requisitos del hardware

El hardware de esta primera iteración, al que hemos denominado *Jarduino Cero*, tiene como objetivo construir un prototipo que sea barato de construir, cómodo de utilizar durante su desarrollo, que se pueda transportar y que pueda utilizarse en un entorno en el que no se disponga de una entrada de agua.

Para ejecutar el código del algoritmo de control de riego se utilizará un dispositivo *Arduino*, que debe leer los valores de humedad de dos sensores, y controlar la apertura y cierre de dos elecroválvulas.

El dispositivo *Arduino* se conectará al PC a través del puerto USB, por el cual enviará datos sobre el estado de la ejecución del algoritmo de control de riego.

3.3.1.2. Requisitos del software

El sofware de *Jarduino Cero* debe implementar un *algoritmo de control de riego* que se instala y se ejecuta en el dispositivo *Arduino* y se encarga del control de riego y de enviar datos al puerto USB; y *una interfaz de linea de comandos* que se instala y ejecuta en el PC al que va conectado el dispositivo, y se encarga de leer los datos de entrada en el puerto USB y mostrarlos, interpretados, en pantalla.

Algoritmo control de riego.

El algoritmo de control de riego debe implementarse para ser ejecutado en *Arduino*, y por lo tanto correrá en un bucle infinito hasta que el dispositivo se apague o sea reprogramado.

Las entradas analógicas del dispositivo *Arduino* que van conectadas a los sensores de humedad y las salidas digitales que manejan las electroválvulas tienen una correspondecia uno a uno.

El algoritmo ha de cumplir varias funciones:

- **Lectura periódica de valores de humedad**. Cada cierto tiempo, definido como una variable, ha de leer los valores de humedad que devuelvan los sensores de humedad, y almacenar un sumatorio para cada sensor.
- **Apertura y cierre de electroválvulas**. Se definen los umbrales de humedad de cada zona de riego como dos variables que almacenan un número entero. Este

número determina el valor de humedad que debe devolver un sensor para que se abra su correspondiente electroválvula.

Además se definen dos variables que indican la cantidad de tiempo que deben mantenerse abierta cada electroválvula. Una vez pasado ese tiempo, la electroválvula se debe cerrar.

- **Envio de valores humedad y marcas de riego al puerto USB**. Cuando se hayan realizado un cierto número de lecturas, ha de enviarse por el puerto USB dos lineas de texto indicando el valor de humedad de cada sensor, y una tercera línea indicando si se ha abierto una de las electroválvulas.

Interfaz de línea de comandos.

La interfaz de linea de comandos se instala y ejecuta en el PC al que conectado, a través del puerto USB, el dispositivo *Arduino*.

La función de la interfaz de línea de comandos es leer la entrada serie del puerto USB que se le indique, y esperar a recibir datos que pueda interpretar. Ha de tener en cuenta que la salida serie del dispositivo puede llegar truncada.

Una vez reciba datos interpretables, los muestra en pantalla, convirtiendo los valores de humedad en valores porcentuales.

La interfaz se ha mantener abierta, recibiendo e interpretando datos, mientras sea posible.

3.3.2. Desarrollo del hardware

Para cumplir los requisitos definidos para el desarrollo del hardware, se opta por adquirir un dispositivo *Arduino Uno* para correr el código de control de riego. Es un dispositivo barato, y con suficiente potencia para la tarea a realizar, que no es nada exigente en recursos.

Se elige una placa microcontroladora como *Arduino Uno* en lugar de un microordenador, como por ejemplo algún modelo de *Rasbpery Pi*. A pesar de que la diferencia de precio puede llegar a ser poca entre ambos aparatos, se utiliza *Arduino Uno* por dos razones. Por una parte, utilizar un microordenador resultaría sobredimensionado con respecto al problema a resolver. La capacidad de procesamiento de *Rasbpery Pi* estaría infrautilizada ejecutando un algoritmo que debe leer datos de unos sensores, enviar señales digitales, y ejecutar unas sencillas comprobaciones. Por otra parte, ha resultado más sencillo encontrar proyectos de control de riego implementados en *Arduino*, por lo que ha parecido más sensato tomar ese camino, ya que se contaba con referencias previas.

El prototipo desarrollado se compone de tres partes. Para empezar tenemos un recipiente de agua con una bomba eléctrica de agua a la cual van conectadas, en T, un par de electroválvulas. Éstas van conectadas a un circuito eléctrico que las conecta a las salidas digitales de un dispositivo *Arduino Uno*. Por otro lado tenemos dos sensores de humedad, colocados sobre dos recipientes (normalmente macetas) que representan dos zonas de riego. Los dos sensores de humedad van conectados a otro circuito eléctrico que acaba en dos entradas analógicas del dispositivo.

Se puede observar un esquema general de estos componentes en la figura 3.2. A los circuitos eléctricos necesarios para el funcionamiento de este prototipo le hemos denominado electrónica auxiliar, y su esquema eléctrico se puede observar en la figura 3.3.

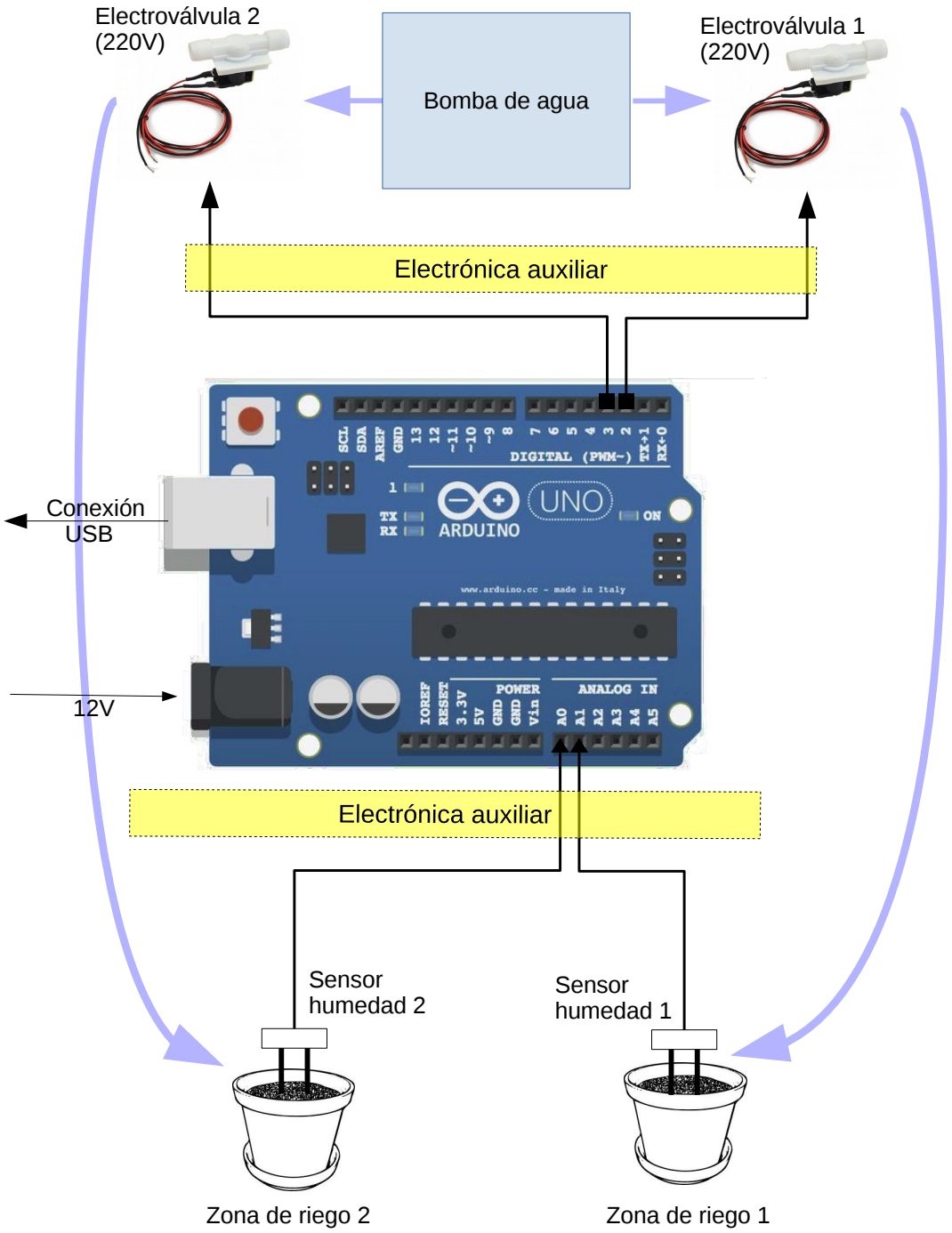


Figura 3.2: Esquema general del hardware del primer prototipo (Jarduino Cero)

Sensores de humedad del sustrato

Los sensores de humedad del sustrato (o higrómetros) se han implementado utilizando un par de tornillos, enterrados en el sustrato, y conectados a una tensión de 12V y a un circuito eléctrico compuesto por un transistor y un par de resistencias. La humedad se mide como la diferencia de potencial entre los dos tornillos, y se envia a las entradas analógica del dispositivo *Arduino Uno*. A mayor humedad en el sustrato en el que se coloquen el par de tornillos, mayor conductividad eléctrica entre ellos, y por lo tanto mayor diferencia de potencial.

A cada una de las dos zonas de riego le corresponde su propio sensor de humedad.

Electroválvulas

Para este primer prototipo se han utilizado dos electroválvulas de 220V reutilizadas de una lavadora. Las electroválvulas se manejan a través de un circuito eléctrico, que consiste en un par de transistores conectados a dos salidas digitales del *Arduino Uno*, y a sendos relés que las abren y cierran a merced de si se recibe una señal digital o no.

3.3.3. Desarrollo del software

Para cumplir los requisitos definidos para el desarrollo del sofware, se desarrollan dos componentes de software interrelacionados pero independientes entre sí: un *algoritmo de control de riego* y una *interfaz de linea de comandos*.

Algoritmo de control de riego

El primer componente que se desarrolla es el código fuente del algoritmo de control de riego que se ejecuta en el dispositivo *Arduino Uno*. Se desarrolla en AVR C, el subconjunto del lenguaje programación C que se ejecuta en la famila de controladores AVR, al que pertenece el microcontrolador ATmega328P que posee *Arduino Uno*.

Los dispositivos de la familia *Arduino* se pueden programar en C o C++. Se ha optado por escribir el código utilizando C, pues las características de programación avanzada—como orientación a objetos—que proporciona C++ no resultan necesarias para el algoritmo desarrollado.

La figura 3.4 representa el algoritimo en un diagrama de flujo.

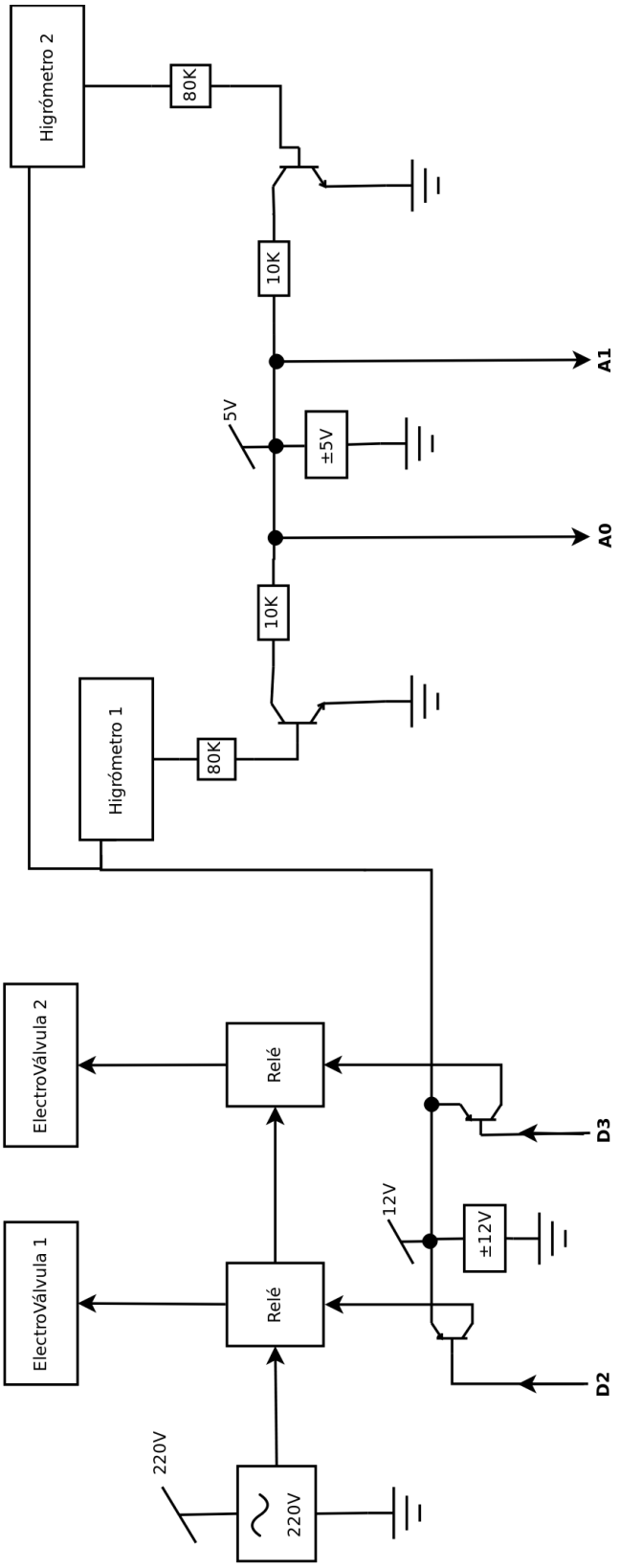


Figura 3.3: Electrónica auxiliar del prototipo de la primera iteración (Jarduino Cero)

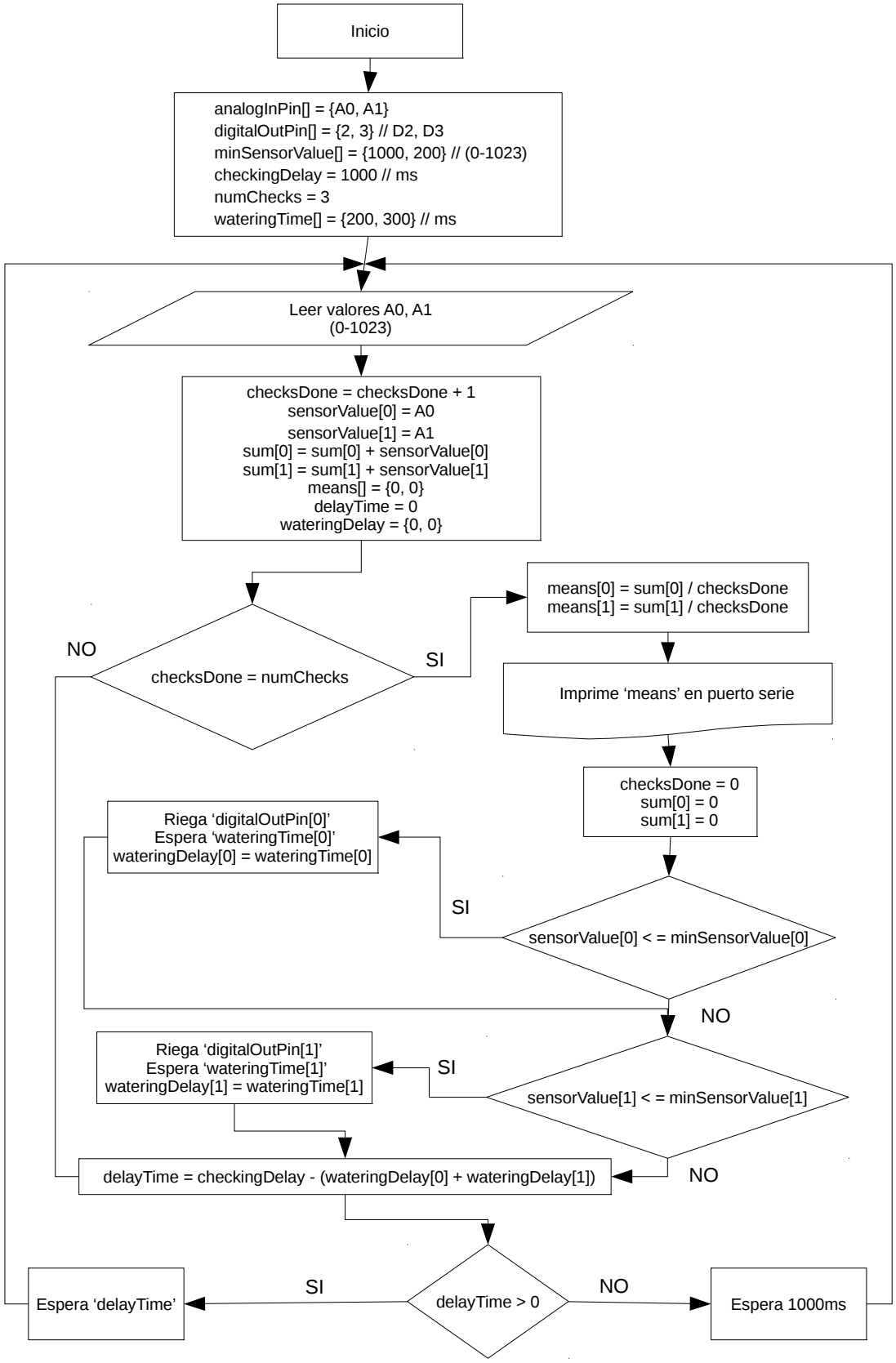


Figura 3.4: Algoritmo de control de riego del primer prototipo (Jarduino Cero)

Interfaz de línea de comandos

El segundo componente que se desarrolla es la interfaz de línea de comandos. Se trata de una aplicación que se instala en el PC al que vaya a conectarse el dispositivo para monitorizar su salida de datos, y que se ejecuta utilizando el intérprete de comandos del sistema.

Por su soporte para escribir aplicaciones de linea de comandos, y por su portabilidad, se ha optado por *python* como lenguaje de programación para escribir el código fuente de la aplicación.

Del dispositivo se espera que devuelva:

- Los valores de humedad leídos por los dos sensores de humedad.
- Si se ha abierto una de las electroválvulas, y en el caso de que se haya abierto, cuál de ellas.

La interfaz se encarga de leer los datos que envia el dispositivo a través de un identificador de puerto serie USB que se le haya indicado, interpretarlos, formatearlos e imprimirlos en pantalla.

Las siguientes tres líneas son un ejemplo de datos tal y cómo los envia el dispositivo al puerto serie USB.

```
#0#102#
#1#204#
#0#w#
```

En el siguiente ejemplo se ejecuta la interfaz de línea de comandos indicándole que lea en el puerto serie '/dev/tty0/', y suponiendo que lee esas mismas tres líneas.

```
$ ./jarduino.py read /dev/tty0

Sensor 1: 10% (102/1024)
Sensor 2: 20% (204/1024)
Regando en zona de riego 1
```

La interfaz de linea de comandos seguirá ejecutándose en primer plano, e imprimiendo, formateados, los datos que siga recibiendo, hasta que el usuario la interrumpa, o se detenga la entrada de datos porque el dispositivo se haya desconectado del puerto USB, o se haya apagado.

3.4. Segunda iteración: Jarduino Uno

En esta segunda se desarrolla el prototipo demostrador que permitirá mostrar el funcionamiento de *Jarduino* durante la defensa de este proyecto de Fin de Carrera. A este prototipo se le ha llamado *Jarduino Uno*. Sus objetivos coinciden con los objetivos generales que se pretenden cumplir con este Proyecto de Fin de Carrera, y que se pueden leer en la sección 1.1, página 7.

Se parte de la iteración anterior para mejorar y extender la funcionalidad desarrollada en el primer prototipo. En cuanto al hardware, resulta útil aprovechar el hecho de que en la iteración anterior se haya desarrollado un prototipo pensado para su transporte y demostración en ausencia de una entrada de agua. En cuanto al software, se han extendido, modificado y mejorado tanto la interfaz de línea de comandos, como el algoritmo de control de riego, además de añadirse una aplicación de interfaz de usuario (*JardUI*) para interactuar con el sistema y administrarlo.

A grandes rasgos, se pretende:

- Añadir la posibilidad de manejar más de dos zonas de riego independientes.
- Tomar datos de nuevos sensores ambientales que puedan estar relacionados con la gestion de riego: humedad y temperatura ambientales.
- Dotar al sistema de una aplicación de interfaz de usuario que permita:
  - Monitorizar las lecturas de datos ambientales que se reciban del dispositivo.
  - Gestionar la configuración de riego de cada zona, y configurar y reprogramar el dispositivo *Arduino* que ejecuta el algoritmo de control de riego.
- Poder utilizar distintos tipos de dispositivos de la familia *Arduino*.
- Comunicación inalámbrica con dipositivos *Arduino* que lo permitan.
- Almacenar un histórico de datos ambientales recogidos por los sensores, y de eventos de riego (cuándo y durante cuánto tiempo se ha regado).

3.4.1. Especificación de los requisitos del sistema

3.4.1.1. Requisitos del hardware

El hardware de esta primera iteración, al que hemos denominado *Jarduino Uno*, no continúa con el objetivo de reutilización y ahorro de recursos de la primera iteración, si no

que tiene como objetivo construir un prototipo que añada funcionalidades al sistema, pero no requiera necesariamente la autofabricación de sensores, y también se pretende evitar la necesidad de implementar circuitos eléctricos para manejar tanto lo sensores como las electroválvulas, siempre que sea posible.

El prototipo resultante de esta segunda iteración es el demostrador que se utilizará durante la defensa de este Proyecto de Fin de Carrera, y constituye solo una de las posibles implementaciones[1] hardware de Jarduino, diseñada para manejar dos zonas de riego.

Se ha de tener en cuenta que quien desarrolla este proyecto de Fin de Carrera carece de formación en electricidad y electrónica, y dado que es posible encontrar placas controladoras para todo tipo de dispositivos, diseñadas para ser compatibles con *Arduino*, resultará más conveniente utilizar una de estas placas en lugar de tratar de fabricar uno mismo la electrónica auxiliar necesaria, lo que supone un esfuerzo mayor para alguien sin formación, además de probablemente innecesario si se puede disponer de un producto que ahorre dicho esfuerzo.

Durante el desarrollo de este segundo prototipo, se pretende cumplir los siguientes requisitos:

- Sustituir los sensores de humedad del sustrato que se habían autofabricado en la primera iteración por sensores de humedad electrónicos que pueden conectarse directamente a las entradas analógicas del dispositivo *Arduino*.
- Sustituir las elecroválvulas reutilizadas de 220V utilizadas en el prototipo resultante de la anterior iteracion, por electroválvulas de 12V, que se puedan manejar directamente con las salidas digitales del dispositivo *Arduino*, conectado a una toma de corriente de 12V.
- Siempre que sea posible, evitar la necesidad de implementar circuitos eléctricos para manejar sensores y electroválvulas, con la intención de simplificar el hardware y reducir el número de componentes a implementar y mantener.
- Añadir sensores de humedad ambiental y temperatura ambiental, para incoporar sus datos al control de riego.

A pesar de que se haga hincapié en el hecho de evitar en la medida de lo posible la necesidad de implementar circuitos eléctricos para contar con un hardware funcional, esto implica la necesidad de adquirir la placa controladora que se adapte a las necesidades del proyecto, lo cual genera un gasto y una dependencia.

[1] Ver 2.4, página 17

De modo que el hardware no debe imponer una restricción en este sentido, y debe ser posible, si se quiere, construir una versión del hardware que implemente los circuitos eléctricos necesarios para interactuar con sensores y electroválvulas, tal y como se hizo en la primera iteración. Es responsabilidad del diseño del software que se desarolle ser capaz de interactuar un dispositivo construido de una manera o de otra.

El prototipo demostrador que se presenta con este Proyecto de Fin de Carrera no tiene la intención de tener una utilidad final para el riego de zonas de riego, aunque se podría utilizar para un cultivo de interior. La intención es servir de soporte de validación y pruebas, y de prototipo demostrador de una de las posibles implementaciones hardware compatibles con Jarduino.

3.4.1.2. Requisitos del software

El software a desarrollar en esta segunda y ultima iteración parte del software desarrollado en la primera iteración. Ha de ampliar la funcionalidades de la interfaz de línea de comandos, extender, parametriza e implementar varias versiones del algoritmo de control de riego (una por cada implementación de hardware soportada, ver 2.4), y añadir una interfaz de usuario que interáctue con la interfaz de línea de comandos para monitorizar y manejar el dispositivo.

Además se ha de desarrollar un servidor HTTP, al que pueden enviarse y del que puedan obtenerse los datos del sistema (valores tomados por los sensores y eventos de riego), y un cliente HTTP que simule un dispositivo enviando datos al servidor HTTP, para facilitar la depuración y el desarrollo del servidor HTTP.

3.4.1.2.1 Algoritmo control de riego

El algoritmo de control de riego desarrollado en la anterior interacción se ha de modificar para:

1. Manejar n sensores y n electroválvulas.

   Cada par de sensor/electroválvula pertenece a una de las n posibles zonas de riego. Se ha de reescribir parte del código para que deje de soportar sólo 2 sensores y 2 electroválvulas, y pueda funcionar con un número arbitrario de sensores y electroválvulas, que quedará solo restringido por la implementación hardware.

2. Implementar riego programado.

   El algoritmo resultante de la anterior iteración no permitía programar el riego, es decir, indicar una hora de inicio, una frecuencia y una duración del riego. Por

ejemplo, comenzar a regar las 00:00 de mañana, regar cada 8 horas, en caso de que el riego sea necesario, y si lo es, regar durante 2 minutos.

Es necesario que el nuevo algoritmo implemente este funcionalidad. La decisión sobre si se debe regar o no depende de cómo se hayan configurado los umbrales de riego, según se explica un poco más adelante en esta misma lista de requisitos.

3. Añadir compatibilidad con la libreria de *OpenGarden*.

   En esta iteración se desarrollan tres versiones del algoritmo de control de riego, cada una dando soporte a una implementación hardware del dispositivo de riego distinta (ver 3.4.1.2.2).

   Para dar soporte a la implementación hardware que incluye *OpenGarden*, se debe implementar una versión de este algoritmo que utilice las funciones de la librería de código que proporciona *OpenGarden* para facilitar la programación del dispositivo.

   En concreto se deben utilizar las funciones proporcionadas para:

   - Leer los valores de la entrada de sensor de humedad del sustrato (conectado a la entrada MOIST de *OpenGarden*) y del sensor de humedad y temperatura ambiental (conectado a la entrada DHT22 de *OpenGarden*).
   - Abrir y cerrar las electroválvulas conectadas a las salidas de electroválvula/motores de *OpenGarden*.

   Se pueden observar las entradas y salidas específicas de *OpenGarden*, a las que se han hecho referencia en la lista anterior, en las figuras 3.6, 3.7 y 3.8.

4. Convertir los valores que se reciben de los higrómetros a las escalas adecuadas.

   Los sensores de humedad del sustrato o higrómetros devuelven los valores de humedad que detectan en una escala de 0 a 1023. Se ha de convertir estos valores a una escala que vaya de 0 a 1023.

5. Permitir utilizar los valores de humedad del sustrato, humedad ambiental y temperatura como umbrales de riego.

   La aplicación de interfaz de usuario permite definir umbrales de riego basados en la humedad del sustrato, la humedad ambiental y la temperatura ambiental, para cada zona de riego.

6. Enviar valores de los sensores y eventos de riego, junto con la fecha y hora en que se tomaron.

   Cada cierto intervalo de tiempo se han de tomar un cierto número de lecturas de los sensores conectados al dispositivo. Una vez tomadas estas lecturas, se calcula su valor medio, y se envia al dispositivo, bien a través de la conexión USB, o bien a través de la interfaz de red del dipositivo. Para cada zona de riego se envia:

- Un identificador de la zona de riego.
- La fecha y hora en la que se envian los datos.
- Los valores medios de los sensores conectados al dispositivo: humedad del sustrato y humedad ambiental en valores porcentuales, temperatura ambiental en grados centígrados.
- Si se ha abierto alguna de las electroválvulas correspondientes a la zonas de riego, se indica cuándo se abrió y durante cuánto tiempo permaneció abierta.

La frecuencia de envío de datos y el número de lecturas que se deben tomar entre envíos se definen en la sección de configuración de la interfaz de usuario (ver 2.2).

3.4.1.2.2 Plantillas de código parametrizadas.

A partir del algoritmo de control de riego definido en la sección anterior, se han de desarrollar 4 plantillas de código parametrizadas, una para cada implementación hardware que soporta *Jarduino*, representadas en la figura 2.4. La siguiente tabla muestra sus diferencias:

Plantilla de código	**Sensores soportados**	**Conexión**	**Fecha y hora**
Arduino Uno + RTC	Humedad sustrato	USB	RTC
Arduino Uno + OpenGarden	Humedad sustrato Humedad ambiental Temperatura ambiental	USB	RTC
Arduino Yún	Humedad Sustrato	USB Wifi Ethernet	NTP
Arduino Yún + OpenGarden	Humedad sustrato Humedad ambiental Temperatura ambiental	USB Wifi Ethernet	RTC

El algoritmo de control de riego define una serie de variables (ver 3.4) que determinaban su comportamiento, a saber:

- A qué salidas digitales se conectan las electroválvulas, y a qué entradas analógicas se conectan los sensores de humedad del sustrato.
- Frecuencia de envío datos.
- Número de lecturas que se deben pedir a los sensores entre envíos.

- Para cada zona de riego:
  - Fecha y hora de inicio del riego.
  - Frecuencia de riego.
  - Duración del riego.
  - Umbrales de riego: humedad del sustrato, humedad ambiental y temperatura ambiental.

El valor de estas variables debe provenir de la configuración de riego de cada zona de riego y de la configuración general que se establezcan en la interfaz usuario, según se específica en la sección 3.4.1.2.3.

El componente encargardo de recibir estas configuraciones, aplicarlas a la plantilla de código que el usuario haya seleccionado, y generar una versión del código fuente con los valores de configuración que correspondan, es la aplicación de línea de comandos que hace de interfaz con el dispositivo *Arduino*, cómo se especifica en 3.4.1.2.4.

3.4.1.2.3 Interfaz de usuario (JardUI)

Se ha de desarrollar una interfaz de usuario para la monitorización y la gestión del riego, a la que denominaremos *JardUI*. La interfaz debe ser una aplicación que se instala en el PC al que se conectará el dispositivo vía USB, o desde el que *JardUI* pedirá información al servidor HTTP.

La aplicación de interfaz de usuario *JardUI* tiene que realizar las siguientes funciones:

1. **Administración de zonas de riego independientes**.

   La interfaz de usuario debe permitir crear, editar y eliminar zonas de riego. Una zona de riego representa un espacio o recipiente dónde se pretende mantener un cultivo con unas necesidades hídricas concretas, y una configuración de riego propia.

   Cada zona de riego tiene los siguientes atributos:

   - Nombre y descripción de la zona.
   - Configuración de riego programado.
     - Fecha y hora de inicio.
       A partir de la fecha y hora indicadas, se comenzará a regar, en el caso de que sea necesario, lo cual viene determinado por la configuración de los umbrales de riego.

- Frecuencia de riego.
  Cada cuánto tiempo se ha de regar, en caso de que sea necesario.
- Duración del riego.
  Durante cuánto tiempo se ha de regar, en caso de que sea necesario.

- Umbrales de riego.

  Los umbrales de riego determinan si se debe regar o no una zona de riego. En última instancia determinan si el dispositivo debe abrir o no la electroválvula correspondiente a una zona de riego, cuando corresponda, según la configuración de riego programado.

  Se definen 3 umbrales de riego. Cada umbral de riego es un valor numérico, y es posible definir uno por cada tipo de dato que obtendremos de los sensores:

  - Humedad del sustrato.
  - Humedad ambiental.
  - Temperatura ambiental.

  El usuario podrá elegir cuál o cuáles umbrales de riego decide aplicar, pudiendo no aplicar ninguno, en cuyo caso sólo se utilizará la configuración de riego programado, regándose con la frecuencia indicada por ella.

Cada vez que se modifique la configuración de riego programado o los umbrales de riego de una zona de riego, la interfaz de usuario debe mostrar un mensaje indicando: *Para que los cambios tenga efecto es necesario reprogramar el dispositivo Arduino.*

2. **Monitorización de datos ambientales**

   Cada zona de riego recibe datos de los sensores conectados al dispositivo. Estos datos se organizan en una secuencia temporal, es decir, llegan acompañados de la fecha y hora en la que se se leyeron. Suponiendo que se haya implementado un dispositivo hardware que lea los valores de humedad del sustrato, humedad ambiental y temperatura ambiental, el dispositivo enviará los valores tomados por los sensores, bien a través del puerto USB o bien los enviará al servidor HTTP a través de una interfaz de red.

   En cualquiera de los dos casos, la aplicación de interfaz de usuario recibirá, por la vía que corresponda (USB o HTTP), los siguientes datos:

   - Fecha y hora de lectura de los datos.
   - Valores de los sensores para cada zona de riego.

     Los valores de humedad del sustrato y humedad relativa se recibirán en valores porcentuales. Los valores de temperatura se recibirán en grados centígrados.

     En el prototipo demostrador presentado, los sensores de humedad relativa y temperatura son compartidos por todas las zonas de riego, pero el formato y la estructura de los datos y su proceso de lectura por la interfaz de usuario debe permitir enviarlos por separado para cada zona de riego.

- Eventos de riego de cada zona de riego.

  En el caso de que se haya iniciado el riego en alguna de las zonas de riego—como consecuencia de que lo exija el riego programado, y los umbrales de riego se cumplan—se recibirá un evento de riego, que consistirá en un identificador de la zona de riego, la fecha y hora en la que se ha producido el riego, y un valor númerico que indique la duración del riego en milisegundos.

Dependiendo de la implementación hardware que se haya realizado, y por lo tanto de los sensores que se hayan conectado al dispositivo, llegarán unos datos u otros, y la interfaz de usuario deberá ser capaz de recoger los datos que lleguen e ignorar las ausencias, y mostrar los que corresponda a cada zona de riego.

Para monitorizar estos datos obtenidos de los sensores ambientales, se representarán gráficamente en un diagrama de líneas para cada zona de riego, que se actualizará en tiempo real con los datos de cada nueva lectura.

En el eje horizontal o de abcisas del diagrama se mostrarán las horas en la que se ha tomado los valores, y en el eje vertical o de ordenadas, se mostrarán los valores tomados. Sobre el diagrama se representarán, como ejes horizontales, él o los umbrales de riego que se hayan establecido.

Además de representar los datos gráficamente en un diagrama de líneas, se mostraran los los últimos valores recogidos, y la fecha y hora de la última lectura.

3. **Configuración general**

   La aplicación de interfaz de usuario debe contar con una sección de configuración general, que permita al usuario establecer:

   a) Conexión para el envío y recepción de datos.

   Si se selecciona una conexión USB, el dispositivo enviará datos a través de esa conexión, y la interfaz de usuario los recibirá a través de la misma.

   En el caso de que se seleccione una conexión HTTP, el usuario tendrá que indicar qué URL quiere que el dispositivo utilice para enviar datos, y qué URL utilizará la interfaz de usuario para recibirlos. Por defecto las URLs se referirán a un servidor instalado localmente (*localhost*).

   b) Frecuencia de envío de datos.

   Determinará cada cuánto tiempo enviará datos desde el dispositivo.

   c) Número de lecturas entre envío de datos.

   Indica el número de lecturas que el dispositivo debe solicitar a los sensores entre envíos de datos. El dispositivo enviará el valor medio de los valores tomados en esas lecturas.

   d) Plantilla de código.

Qué plantilla de código (ver 3.4.1.2.2) se utilizará para reprogramar el dispositivo. El usuario puede seleccionar una de las plantillas que se proveen, o una propia, pues se le permitirá seleccionar cualquier fichero de su sistema.

e) Esquema de conexión.

La correspondencia entre zonas de riego y sensores/electroválvulas se decidirá automáticamente. Por ejemplo, a la primera zona de riego creada le corresponderán la entrada A0 para el sensor de humedad del sustrato, y la salida D2 para la electroválvula, a la segunda la entrada A1 y la salida D3 y así sucesivamente.

Este esquema de conexión puede ser modificado por el usuario, permitiéndole indicar, para cada zona de riego, dónde están conectados los sensores de humedad del sustrato y las electroválvulas.

4. **Histórico de datos ambientales y eventos de riego**

   La interfaz de usuario debe almacenar el histórico de datos ambientales y eventos de riego. Si el origen de datos es el puerto USB, los datos se almacenan localmente en una base de datos alojada la máquina en la que esté instalada la aplicación de usuario. Si llegan a través de una petición al servidor HTTP, los datos se descargarán en un almacenamiento local que sirve de caché, que será actualizado cada vez que se haga una nueva consulta al servidor HTTP.

   De este modo, la interfaz de usario deberá permitir consultar los datos históricos dentro de un periodo de tiempo determinado, sin necesidad de acceder al servidor remoto ni contar con un dispositivo conectado al puerto USB.

   Por lo tanto serán necesarias dos bases de datos: una que almacene los datos remótamente, que será la que utilice el servidor HTTP para realizar las consultas e inserciones, y una local que sirva de caché.

   Los requisitos de los datos a almacenar en ambas base de datos son los mismos:

   - Fecha y hora en la que se enviaron los datos desde el dispositivo.
   - Para cada zona de riego:
     - Valores medios leídos por los sensores conectados al dispositivo.
     - Si se ha producido un riego, fecha y hora en la que se produjo, y duración del mismo.

5. **Reprogramación multidispositivo**

   La aplicación de interfaz de usuario debe permitir reprogramar el dispositivo para que ejecute el algoritmo de control de riego con los parámetros de configuración de cada zona de riego, y los de la configuración general, detallada en el apartado anterior. La tarea de reprogramación sólo se puede realizar si el dispositivo está conectado al PC por el puerto USB, en caso contrario la reprogramación no estará disponible.

Cada zona de riego tiene uno o varios umbrales de riego, que determinan si se debe regar o no con respecto al valor que se lea en uno de los sensores conectados al dispositivo. Por ejemplo, puede tener un umbral de riego basado en la humedad del sustrato, que sea del 10 %. Si la lectura de humedad del sustrato está por debajo del 10 %, el dispositivo regará. En caso contrario, no. Para cada zona de riego hay además una configuración de riego programado, que indica cuando se debe comenzar a regar, si los umbrales de riego lo permiten, con qué frecuencia y durante cuánto tiempo. Podemos refirirnos a la configuración de riego programado y a los umbrales de riego de forma global como *configuración de riego*.

Además de esta configuración de riego para cada zona, se aplica la configuración general, que determina cuándo y cómo se envían desde el dispositivo los valores de los sensores, el esquema de conexión de sensores y electroválvulas, y la plantilla de código que se utilizará.

Hay una plantilla de código para cada implementación del dispositivo soportada por *Jarduino*, cómo se especifica en 3.4.1.2.2. Todas las plantillas tienen una serie de parámetros que se pueden sustituir por valores concretos para generar una versión del algoritmo de control de riego. La tarea de generar este código fuente le pertenece a la interfaz de línea de comandos.

6. **Autodetección de dispositivo via USB**

   La aplicación de interfaz de usuario, al iniciarse, debe tratar de detectar si hay un dispositivo de la familia *Arduino* conectado al PC por el puerto USB.

   Si lo encuentra, debe mostrar el mensaje: *Dispositivo Arduno detectado*. Si no lo encuentra, debe mostrar el mensaje *No se ha encontrado ningún dispositivo Arduno*.

3.4.1.2.4 Interfaz de línea de comandos.

La aplicación de interfaz de línea de comandos mantiene la misma responsabilidad para la que se desarrolló en la iteración anterior: servir de interfaz entre el usuario o *JardUI* y el dispositivo *Arduino* conectado al puerto USB del PC.

En esta segunda iteración se ha de partir de la aplicación de interfaz de comandos desarrollada en la primera iteración, y añadir las siguientes mejoras y funcionalidades:

1. Autodetectar el dispositivo *Arduino* conectado.

   La aplicación dearrollada en la iteración anterior necesitaba que se le indicara en qué puerto está conectado el dispositivo con el que debía interactuar.

   Durante esta iteración se debe modificar la aplicación para que detecte si hay un dispositivo *Arduino* conectado a algún puerto USB, y en caso de que lo haya, usar ese puerto USB para comunicarse con él.

2. Generar el código fuente a partir de una plantilla de código.

   El usuario puede seleccionar en la configuración general de *JardUI* qué plantilla de código quiere utilizar para programar el dispositivo.

   La aplicación de interfaz de línea de comandos debe poder recibir como parámetro la ruta del fichero de plantilla de código que el usuario haya seleccionado, además de la configuración de riego de cada zona y la configuración general definida en la interfaz de usuario. Usando esta información debe rellenar la plantilla de código y generar el código fuente que utilizará el dispositivo para implementar el algoritmo de control de riego.

3. Compilar el código fuente y reprogramar el dispositivo con él.

   Una vez generado el código fuente a partir de la plantilla de código seleccionada por el usuario, se debe compilar para que se pueda ejecutar en el dispositivo *Arduino* que se pretende reprogramar, y reprogramar dicho dispositivo con el código compilado.

4. Leer e interpretrar la salida del dispositivo a través del puerto USB.

   La aplicación de interfaz de usuario debe interpretar y convertir la salida que genere el dispositivo durante su ejecución, a un formato legible por*JardUI*, de manera que la interfaz de usuario puede a su vez interpretar y manejar los datos generados por el dispositivo.

3.4.1.2.5 Servidor HTTP

Para que sea posible la comunicación entre la aplicación de interfaz de usuario (*JardUI*) y el dispositivo (sin necesidad de que éste último esté conectado por el puerto USB al PC dónde se ejecuta la interfaz de usuario) se ha de desarrollar una aplicación que actúe cómo servidor HTTP, a la cuál pueda el dispositivo enviar datos a través de su interfaz de red, y de la cuál pueda *JardUI* recibir el histórico de datos generados a partir de cierta fecha, o dentro de cierto intervalo de fechas.

La aplicación a desarrollar, por lo tanto, deberá ser capaz de comunicarse con un cliente usando el protocolo HTTP, e implementar un par de URLs que realicen la funcionalidad deseada, a saber:

1. URL para la recepción de datos desde el dispositivo.

   Debe aceptar peticiones de tipo POST y almacenar los datos recibidos en una base de datos donde se mantendrá el histórico de datos.

2. URL para el envío de datos hacia *JardUI*.

Debe aceptar peticiones de tipo GET, con un parámetro obligatorio que indique a partir de qué fecha se quieren recibir los datos, y un parámetro opcional que indique hasta qué fecha. Devolverá los datos históricos correspondientes al intervalo de tiempo indicado, obteniéndolos de la base de datos que almacena dicho histórico.

3.4.1.2.6 Cliente HTTP

Con el propósito de facilitar la depuración y el desarrollo del servidor HTTP, sin necesidad de contar con un dispositivo conectado a la misma red que el PC dónde se desarrolla el servidor HTTP, se debe desarrollar un cliente HTTP que emule el funcionamiento de un dispositivo *Arduino* que envíe datos, via petición POST, al servidor HTTP.

Se debe desarrollar como una aplicación de línea de comandos, y se ejecutará en paralelo al servidor HTTP para poder realizar las peticiones contra él. Generará datos aleatorios (valores de los sensores y de los eventos de riego) en el mismo formato en el que se enviiarían desde el dispositivo, de manera que para el servidor HTTP sea imposible diferenciar si los datos le llegan de un dispositivo real o del cliente HTTP.

3.4.2. Desarrollo del hardware

Jarduino puede ser utilizado con varias posibles implementaciones del hardware del dispositivo de riego. La figura 2.4 muestra las que son soportadas por las plantillas de código que se han desarrollado (ver sección 3.4.1.2.2). Nada impide al usuario realizar su propia implementación, y de hecho el espíritu detrás de este proyecto pretende lo contrario: conminar al usuario de *Jarduino* a implementar su propia versión del hardware del dispositivo, junto con su propia plantilla de código (si es necesario) y utilizar la aplicación de interfaz de usuario de *Jarduino* para interactuar con ella.

En esta sección se pretende describir el proceso del prototipo demostrador que se presenta para la defensa de este Proyecto de Fin de Carrera, que es solo una de las posibles implementaciones compatibles con *Jarduino*.

Uno de los requisitos definidos para el desarrollo del hardware es, en la medidad de lo posible, evitar la necesidad de desarrollar la electrónica auxiliar necesaria para manejar sensores y electroválvulas. Para facilitarlo, se opta por adquirir un dispositivo *OpenGarden* para desarrollar el prototipo demostrador que se presenta. Hay que recordar que el prototipo presentado es sólo una de las posibles implementaciones hardware del dispositivo de control de riego. No es necesario por lo tanto adquirir *OpenGarden* para utilizar *Jarduino*, pero se ha considerado necesario utilizarlo para facilitar la implementación del hardware y la programación del software que interactúa con sensores y electroválvulas (el algoritmo de control de riego).

OpenGarden [OG] es una placa extensión o *shield* para *Arduino* que se vende cómo parte de un kit, o por separado. Los kits de *OpenGarden* están orientados a la monitorización de cultivos (de exterior, interior o hidropónicos). Proporcionan todos los componentes hardware necesarios, y una libreria en AVR C (el subconjunto de C que ejecuta *Arduino*) para interactuar con ellos. Con esta libreria se proporcionan ejemplos de uso, pero no un algoritmo de control o monitorización de riego completo. El usuario de *OpenGarden* debe desarrollar el código que quiere que ejecute el dispositivo. *OpenGarden* no es más que una plataforma de trabajo, no proporciona la solución a ningun problema de control de riego concreto.

El desarrollo de este algoritmo de control de riego que se ejecutará sobre la plataforma *OpenGarden* se describe en la siguiente sección.

Para el desarrollo del protipo demostrador que se presenta se han utilizado únicamente aquellos componentes de *OpenGarden* que se han considerado necesarios para el control de riego: sensores de humedad del sustrato, sensor de humedad ambiental, sensores de temperatura y electroválvulas. Otros sensores, como el de luminosidad, incluido en la placa, no están relacionados con el control de riego, y no se han utilizado. La figura 3.5 muesta todas las posibilidades de conexión de *OpenGarden*.

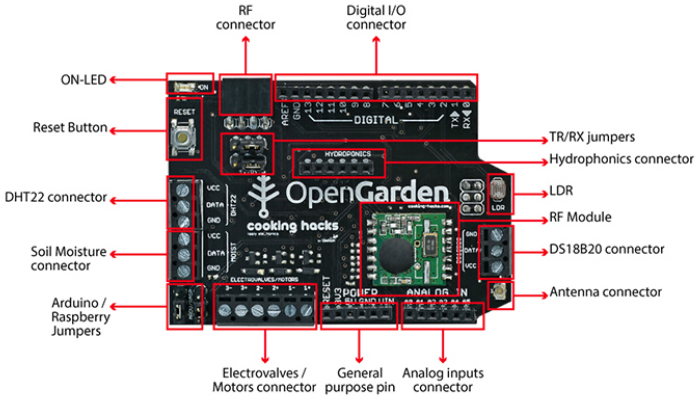


Figura 3.5: Posibilidades de conexión de la placa de extensión OpenGarden

El prototipo demostrador presentado controla dos zonas de riego, cada una con su sensor de humedad del sustrato y su electroválvula correspondiente. El sensor de humedad y temperatura ambiental es compartido por ambas zonas. Por lo tanto el dispositivo cuenta con dos sensores de humedad del sustrato, dos electroválvulas y un sensor de humedad y temperatura ambientales. Las siguientes figuras muestran las entradas de conexión de estos componentes en la placa *OpenGarden*.

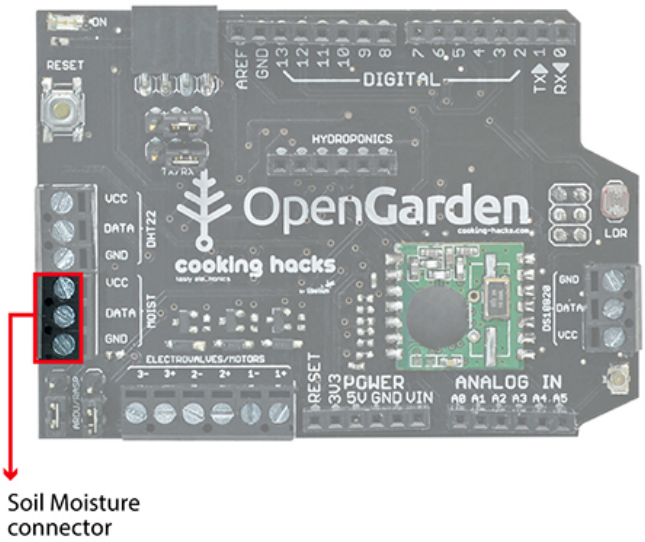


Figura 3.6: Entrada para la conexión con sensor de humedad del sustrato

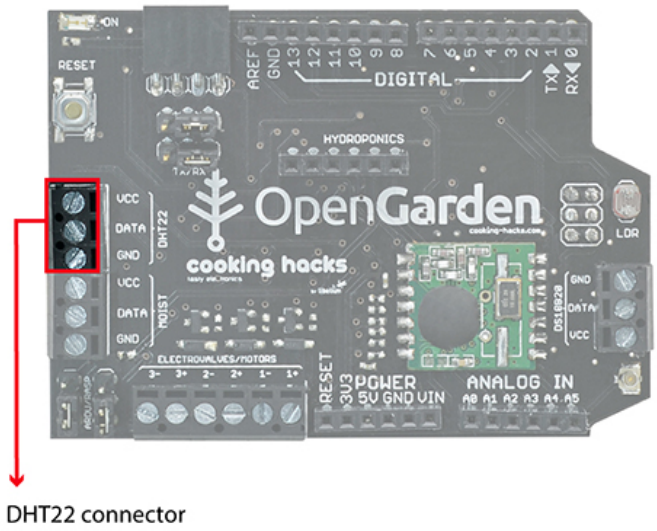


Figura 3.7: Entrada para la conexión con sensor de humedad y temperatura ambiental

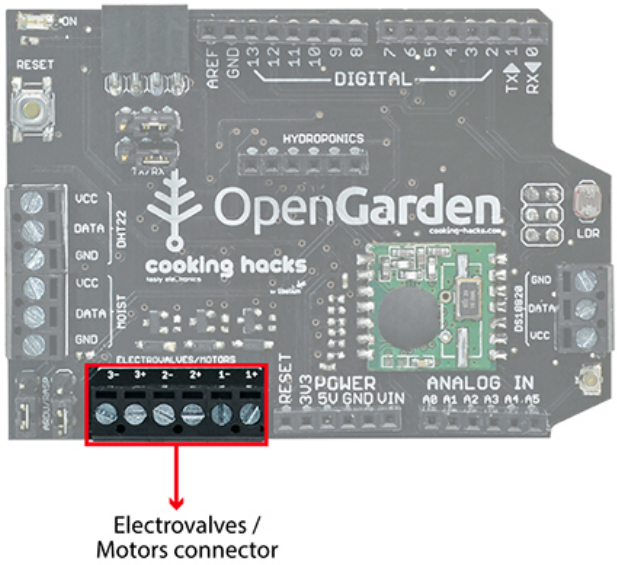


Figura 3.8: Salida para la conexión con electroválvulas o motores

Puede observarse un esquema del montaje de los componentes hardware del protoipo demostrador presentado en la figura 3.9.

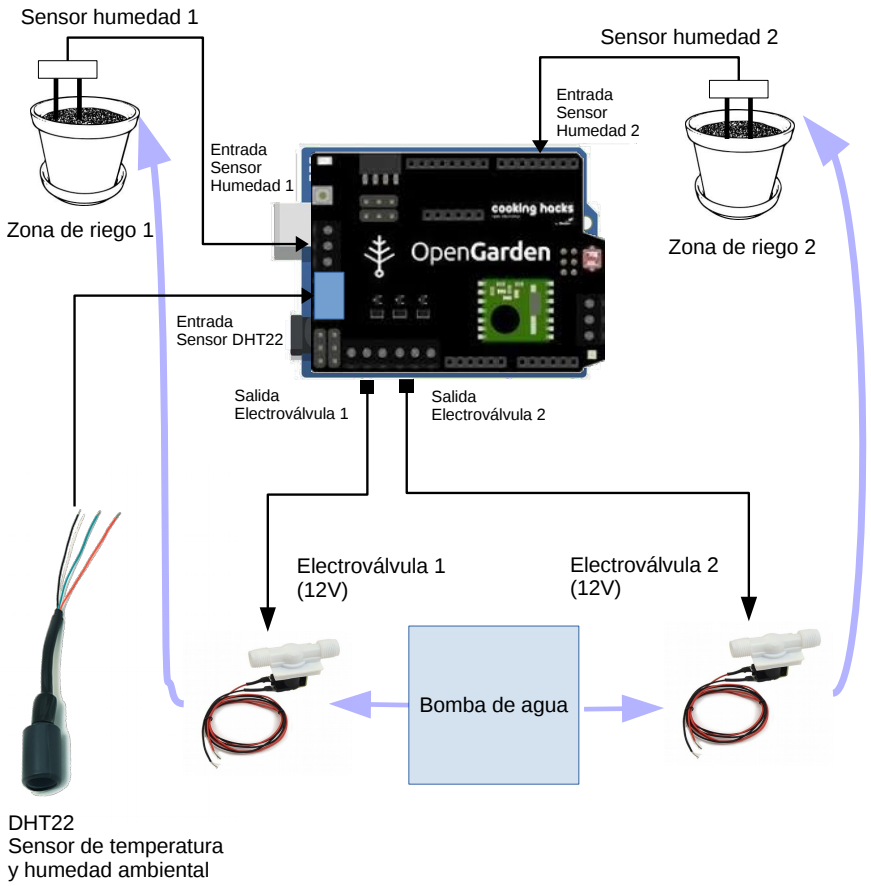


Figura 3.9: Esquema general del hardware del prototipo presentado (Jarduino Uno)

Por último, se ha reutilizado el recipiente de agua, la bomba de agua y el conector en T para las electroválvulas utilizadas en la anterior iteración, para facilitar el transporte y la demostración del prototipo sin necesidad de contar con una entrada de agua.

3.4.3. Desarrollo del software

Para cumplir los requisitos definidos para el desarrollo del software, se parte de los dos componentes de software desarrollados en la anterior iteración: el algoritmo de control de riego y la interfaz de linea de comandos, cuyas funcionalidades se amplían. Además, se añaden tres nuevos componentes: la interfaz de usuario, a la que hemos denominado *JardUI*, el servidor HTTP y el cliente HTTP.

Para describir el desarrollo de los componentes de software más importantes que conforman *Jarduino*, vamos a analizar y diseñar cada componente por separado. Para cada componente tendremos por una parte el análisis, ilustrado con diagramas de casos de uso, y por otra parte una descripción del diseño del componente, ilustrado con los diagramas que correspondan según el tipo de componente.

Interfaz de usuario (JardUI)

La interfaz de usuario, a la que se ha denominado *JardUI*, es lo que se conoce como una aplicación de escritorio. Es decir, es una aplicación que se instala en el ordenador desde el que se quiere interactuar con el dispositivo hardware de control de riego. Puede interactuar con el dispositivo hardware de control de riego, que ejecuta el algoritmo de control de riego, bien a través del puerto USB o bien, con la mediación del servidor HTTP, remótamente a través de la red local o de Internet. La siguiente figura representa, con un alto nivel de abstracción, las interacciones de la interfaz de usuario con el resto de componentes software desarrollados.

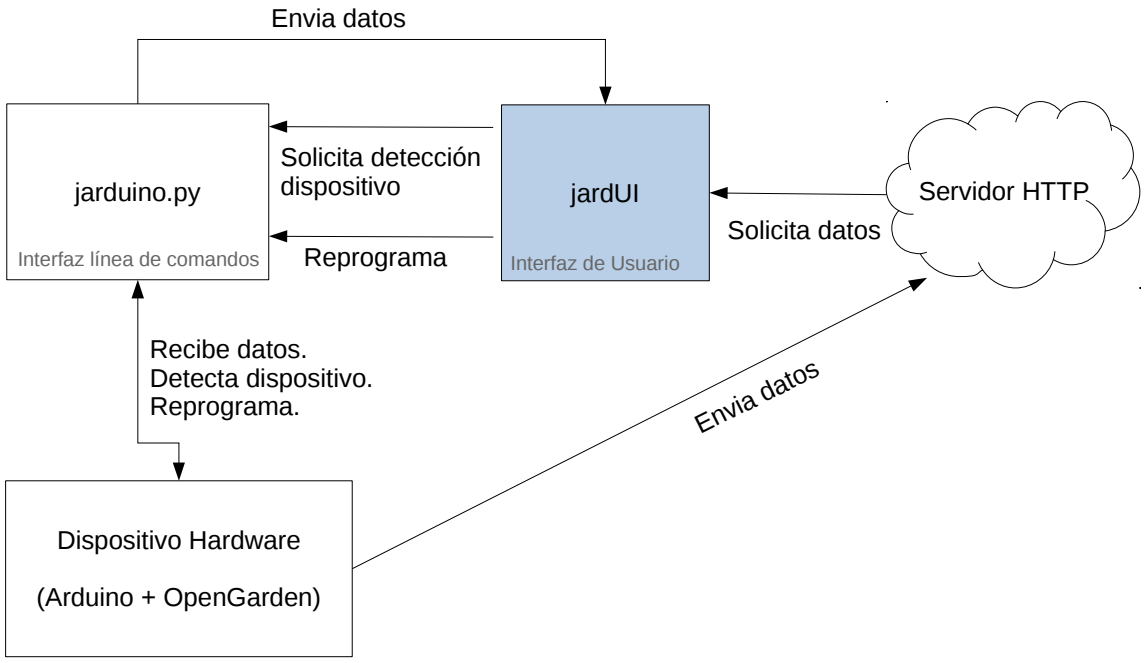


Figura 3.10: Interacciones de la interfaz de usuario con el resto de componentes software.

Análisis

Partiendo de los requisitos definidos en la sección 3.4.1.2.3 (página 15), se comienza con su análisis, utilizando casos de uso. La figura 3.11 muestra los casos de uso de la interfaz de usuario. Se identifican 4 actores: el usuario, el propio sistema, la interfaz USB o interfaz de línea de comandos y la interfaz HTTP.

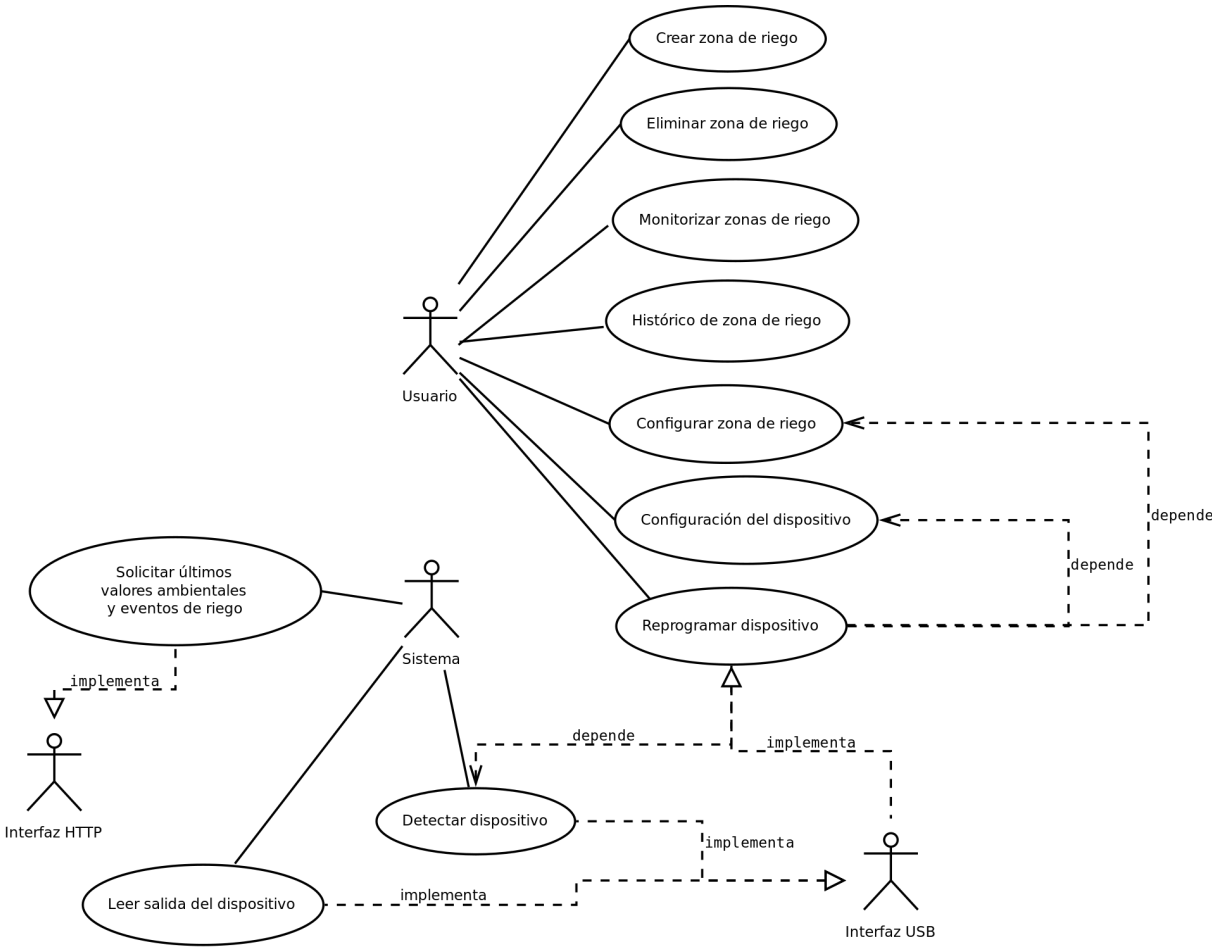


Figura 3.11: Casos de uso de la interfaz de usuario.

A continuación se describen los casos de uso extraídos de los requisitos de desarrollo de la aplicación de interfaz de usuario.

Caso de Uso: Crear zona de riego

- *Descripción*: El usuario puede crear una zona de riego. El sistema mostrará un formulario con los campos descritos en la tabla 3.1
- *Actores*: Usuario, Sistema.
- *Precondiciones*: Ninguna.

  Postcondiciones: Se crea una zona de riego con los datos introducidos por el usuario y se muestra un mensaje indicando que ha de reprogramarse el dispositivo para que los cambios realizados tenga efecto.

Cuadro 3.1: Campos de la zona de riego

Nombre del campo	**Tipo**	**Valores**	**Valor por defecto**	**Obligatorio**
Nombre	Texto			Sí
Descripción	Texto			No
Fecha y hora de inicio de riego	Fecha/Hora	Cualquier fecha y hora	Fecha y hora actual	Sí
Frecuencia de riego	Número		1	Sí
Unidad de frecuencia de riego	Opciones	Horas Minutos Segundos	Horas	Sí
Duración del riego	Número		2	Sí
Unidad de duración del riego	Opciones	Horas Minutos Segundos	Horas	Sí
Umbral de riego (Humedad del sustrato)	Número	1-100 %	50 %	No
Umbral de riego (Humedad ambiental)	Número	1-100 %		No
Umbral de riego (Temperatura ambiental)	Número	1-100ºC		No

Caso de Uso: Configurar zona de riego

- *Descripción*: El usuario puede cambiar la configuración de una zona de riego. El sistema mostrará un formulario con los campos descritos en la tabla 3.1 y permitirá modificarlos con las restricciones descritas en la tabla.
- *Actores*: Usuario, Sistema.
- *Precondiciones*: Se ha creado al menos una zona de riego.
- *Postcondiciones*: Se actualiza la zona de riego con los datos introducidos por el usuario y se muestra un mensaje indicando que ha de reprogramarse el dispositivo para que los cambios realizados tenga efecto.

Caso de Uso: Eliminar zona de riego

- *Descripción*: El usuario puede eliminar una zona de riego. El sistema le pedirá confirmación.
- *Actores*: Usuario, Sistema.

- *Precondiciones*: Se ha creado al menos una zona de riego.
- *Postcondiciones*: Si el usuario confirma, se elimina la zona de riego. Se muestra un mensaje indicando que ha de reprogramarse el dispositivo para que los cambios realizados tenga efecto.

Caso de Uso: Gráfico de valores ambientales y eventos de riego

- *Descripción*: El usuario puede monitorizar el estado de una zona de riego. El sistema debe mostrar para cada zona de riego un gráfico
- *Actores*: Usuario, Sistema.
- *Precondiciones*: Se ha creado al menos una zona de riego.
- *Postcondiciones*: Si el usuario confirma, se elimina la zona de riego. Se muestra un mensaje indicando que ha de reprogramarse el dispositivo para que los cambios realizados tenga efecto.

Caso de Uso: Monitorizar zonas de riego

- *Descripción*: El usuario puede monitorizar el estado de las zonas de riego. El sistema debe mostrar para cada zona de riego:
  - Su nombre y descripción.
  - Si no se han recibido datos aún para la zona de riego, se indicar al usuario que se están esperando los datos.
  - Un gráfico en forma de diagrama de líneas que muestre:
    - Una línea que interpole los últimos valores de humedad del sustrato que se han recibido, si los hay.
    - Una línea que interpole los últimos valores de humedad ambiental que se han recibido, si los hay.
    - Una línea que interpole los últimos valores de temperatura ambiental que se han recibido, si los hay.
    - Opcionalmente, una línea que represente el umbral de humedad del sustrato, si está definido para la zona de riego.
    - Opcionalmente, una línea que represente el umbral de humedad ambiental, si está definido para la zona de riego.
    - Opcionalmente, Una línea que represente el umbral de temperatura, si está definido para la zona de riego.

    - Los eventos de riego deben aparecer representados en el gráfico, teniendo en cuenta su duración.
  - La fecha y hora en la que se ha recibido la última lectura.
  - Los últimos valores de humedad del sustrato, humedad ambiental y temperatura ambiental que se hayan recibido.
  - La configuración de riego: fecha y hora de inicio, frecuencia y duración.
  - Los umbrales de riego para humedad del sustrato, temperatura ambiental y humedad ambiental.

- *Actores*: Usuario, Sistema.
- *Precondiciones*: Se ha creado al menos una zona de riego.
- *Postcondiciones*: Ninguna.

Caso de Uso: Histórico de zona de riego

- *Descripción*: El usuario puede visualizar, en un gráfico de diagrama de líneas como el descrito en el caso de uso anterior, el histórico de valores ambientales y eventos de riego recibidos dentro de un periodo de tiempo. El sistema permite:
  - Filtrar el periodo de tiempo.
  - Ampliar y reducir la resolución del gráfico.
- *Actores*: Usuario, Sistema.
- *Precondiciones*: Se ha creado al menos una zona de riego.
- *Postcondiciones*: Ninguna.

Caso de Uso: Configuración del dispositivo

- *Descripción*: El usuario puede configurar el comportamiento general del dispositivo y el origen de los datos. El sistema muestra un formulario al usuario en el que éste puede establecer los valores de los campos de la tabla 3.2.

  Todos los datos son obligatorios excepto las URLs, que sólo lo son cuando se haya seleccionado la opción HTTP.

  Además de estos datos, el usuario podrá modificar el esquema de conexión que el sistema habrá generado automáticamente. Por ejemplo, a la primera zona de riego creada le corresponderán la entrada A0 para el sensor de humedad del sustrato, y la salida D2 para la electroválvula, a la segunda la entrada A1 y la salida D3 y así sucesivamente. El usuario podrá modificar estas correspondencias.

Cuadro 3.2: Campos de configuración del dispositivo

Nombre del campo	**Tipo**	**Valores**	**Valor por defecto**
Tipo de conexión	Opciones	USB HTTP	USB
URL envío de datos (POST)	Texto	Cualquier URL válida	http://localhost/jarduino
URL envío de datos (GET)	Texto	Cualquier URL válida	http://localhost/jarduino
Frecuencia de envío de datos	Número		1
Unidad de frecuencia de envío de datos	Opciones	Horas Minutos Segundos	Horas
Plantilla de código	Fichero		Ruta de la plantilla de código Arduino+OpenGarden

- *Actores*: Usuario, Sistema.
- *Precondiciones*: Ninguna.
- *Postcondiciones*: Se almacena la configuración y se muestra un mensaje indicando que el dispositivo debe ser reprogramado para que los cambios tengan efecto.

  Si se selecciona el tipo de conexión HTTP, el dispositivo utilizará la URL indicada envío de datos, y la interfaz HTTP utilizará la URL indicada para la recepción de datos desde el servidor.

  Si se selecciona el tipo de conexión USB, el sistema leerá la salida del puerto USB al que esté conectado el dispositivo.

Caso de Uso: Deteccción del dispositivo

- *Descripción*: El sistema, cada 3 segundos, solicitará a la interfaz USB que trate de detectar un dispositivo *Arduino* conectado a la máquina por el puerto USB.
- *Actores*: Usuario, Sistema, Interfaz USB.
- *Precondiciones*: Se ha seleccionado tipo de conexión USB en la configuración.
- *Postcondiciones*: Se muestra al usuario un mensaje indicando que el dispositivo ha sido detecteado.

Caso de Uso: Reprogramación del dispositivo

- *Descripción*: El usuario puede reprogramar el dispositivo pulsando un botón.
- *Actores*: Usuario, Sistema, Interfaz USB.
- *Precondiciones*: Dispositivo conectado.
- *Postcondiciones*:
  1. Se genera un fichero de configuración que incluye:
     - Para cada zona de riego:
       - Un identificador.
       - Configuración de riego programado. (ver caso de uso *Crear zona de riego*)
       - Configuración de umbrales de riego. (ver caso de uso *Crear zona de riego*)
     - Configuración del dispositivo. (ver caso de uso *Configuración del dispositivo*)
  2. Se envía al interfaz USB la solicitud de reprogramar el dispositivo usando el fichero de configuración generado.

Caso de Uso: Leer salida de datos

- *Descripción*: El sistema, cada vez que lo indique la frecuencia de envío de datos, solicita a la interfaz USB que lea la salida del dispositivo. (ver **??**)
- *Actores*: Usuario, Sistema, Interfaz USB.
- *Precondiciones*: Dispositivo conectado y tipo de conexión USB configurada.
- *Postcondiciones*: Se actualizan el gráfico de datos ambientales y eventos de riego (ver caso de uso *Monitorizar zona de riego*).

Caso de Uso: Solicitar últimos valores ambientales y eventos de riego

- *Descripción*: El sistema, cada vez que lo indique la frecuencia de envío de datos, solicita a la interfaz HTTP los últimos valores ambientales y eventos de riego.
- *Actores*: Usuario, Sistema, Interfaz HTTP.
- *Precondiciones*: Tipo de conexión HTTP seleccionada.
- *Postcondiciones*: Se actualizan el gráfico de datos ambientales y eventos de riego (ver caso de uso *Monitorizar zona de riego*).

Diseño

JardUI se ejecuta como una aplicación de escritorio, pero se desarrolla como una aplicación web escrita en *HTML5*, *CSS3* y *JavaScript*, con la intención de que sea fácilmente portable, y puede acabar ejecutándose como un sitio web accesible remótamente, o como una aplicación que se ejecute en móviles o tabletas.

JavaScript (abreviado *JS*) es un lenguaje fácilmente portable a una gran variedad de plataformas, siempre que la plataforma cuente con un navegador capaz de ejecutarlo. Es un lenguaje relativamente sencillo, con el que sin embargo resulta complicado realizar aplicaciones con cierta complejidad si no se cuentan con el apoyo de librerias y marcos de trabajo que permitan desarrollar aplicaciones complejas en un tiempo y con un esfuerzo razonable. Así cómo no tiene sentido práctico desarrollar una aplicación de escritorio para, por ejemplo, Windows, escrita íntegramente desde cero en C o C++, sin contar con la ayuda de un IDE, un marco de trabajo y/o un cierto número de librerias, algo parecido ocurre con *JavaScript*. Por lo que resulta necesario apoyarse en una serie de librerías y herramientas:

- *NodeJS* Un intérprete de *JavaScript* multiplataforma diseñado para escribir aplicaciones del lado del servidor. *Jarduino* utiliza una librería llamada *python-shell* para interactuar con la interfaz de línea de comandos y otra librería llamada *node-sqlite3* como API para manejar el motor de base datos *SQLite*.
- *NW.js* Una libreria que permite generar aplicaciones de escritorio para varias plataformas (Windows, OSX, GNU/Linux) a partir de páginas web, y que es compatible con las librerías escritas en NodeJS. Genera una aplicación con un navegador incrustado que puede renderizar páginas y ejecutar *JavaScript*.
- *ReactJS* una librería de *JavaScript* para la construcción de interfaces de usuario.
- *Rickshaw* una librería de *JavaScript* para la creación de gráficos interactivos que representen datos en series temporales.

La figura 3.12 muestra los componentes de *JavaScript* que se han utilizado.

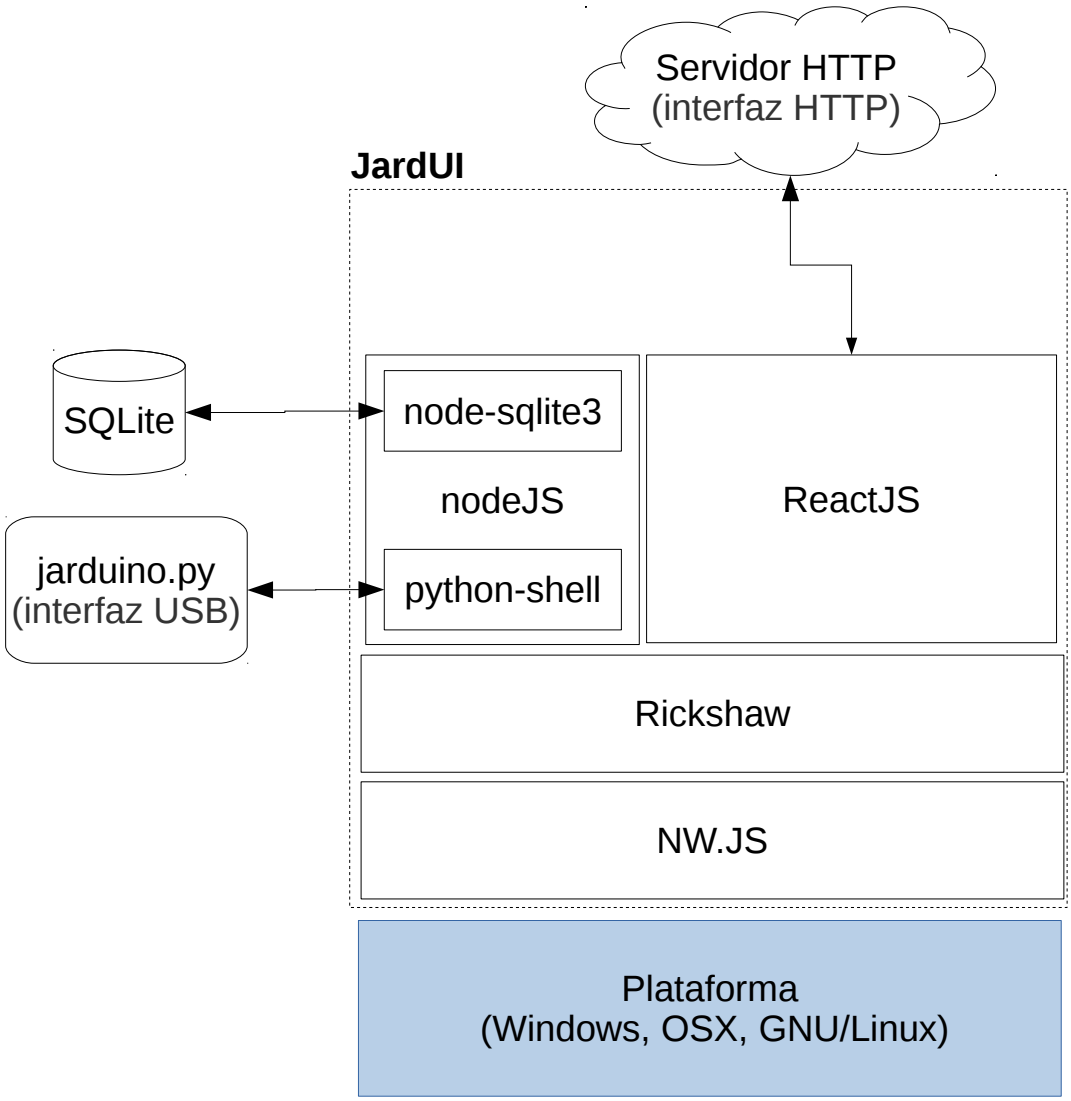


Figura 3.12: Componentes JavaScript utilizados para desarrollar JardUI

Diseño de la base de datos

Para el desarollo de la base de datos de la interfaz de usuario se decide utilizar el motor de base de datos relacional SQLite, por su portabilidad y la facilidad de interactuar con él utilizando *JavaScript*. Es un motor de base datos modesto, pero suficiente para las características de los datos a almacenar.

En la figura 3.13 se muestra el diagrama de Entidad-Relación de la base de datos diseñada.

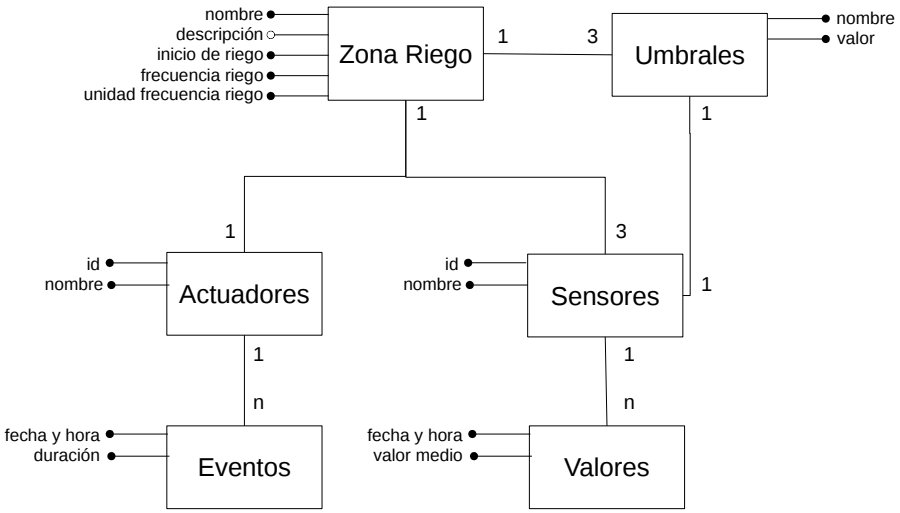


Figura 3.13: Esquema entidad-relación de la base de datos

Diseño de la interfaz de usuario

La aplicación de interfaz de usuario se ha diseñado con el objetivo principal de facilitar:

- La administración de zonas de riego.
- La monitorización en tiempo real de los valores ambientales que afectan a todas las zonas de riego.

- La reprogramación del dispostivo.

La siguiente figura muestra el diseño la pantalla principal, sin ningún dispositivo *Arduino* conectado, y con dos zonas de riego creadas.

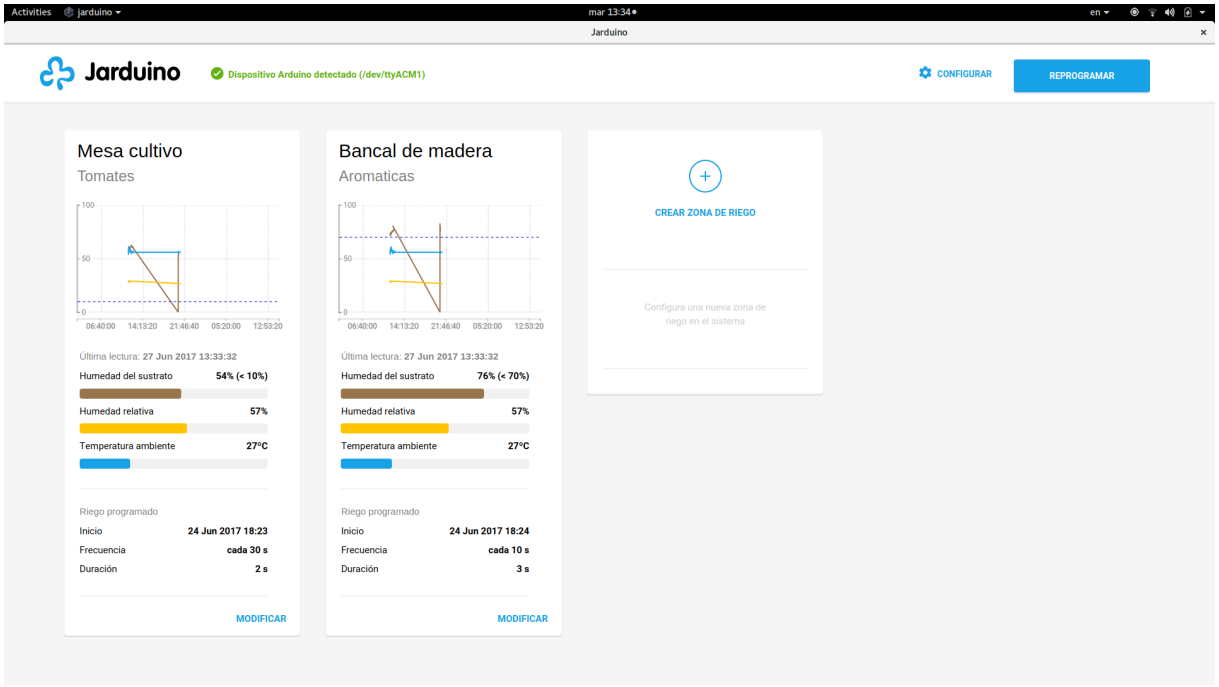


Figura 3.14: Pantalla principal de la interfaz de usuario con 2 zonas de riego

A continuación se muestran sendos ejemplos de creación y modificación de una zona de riego.

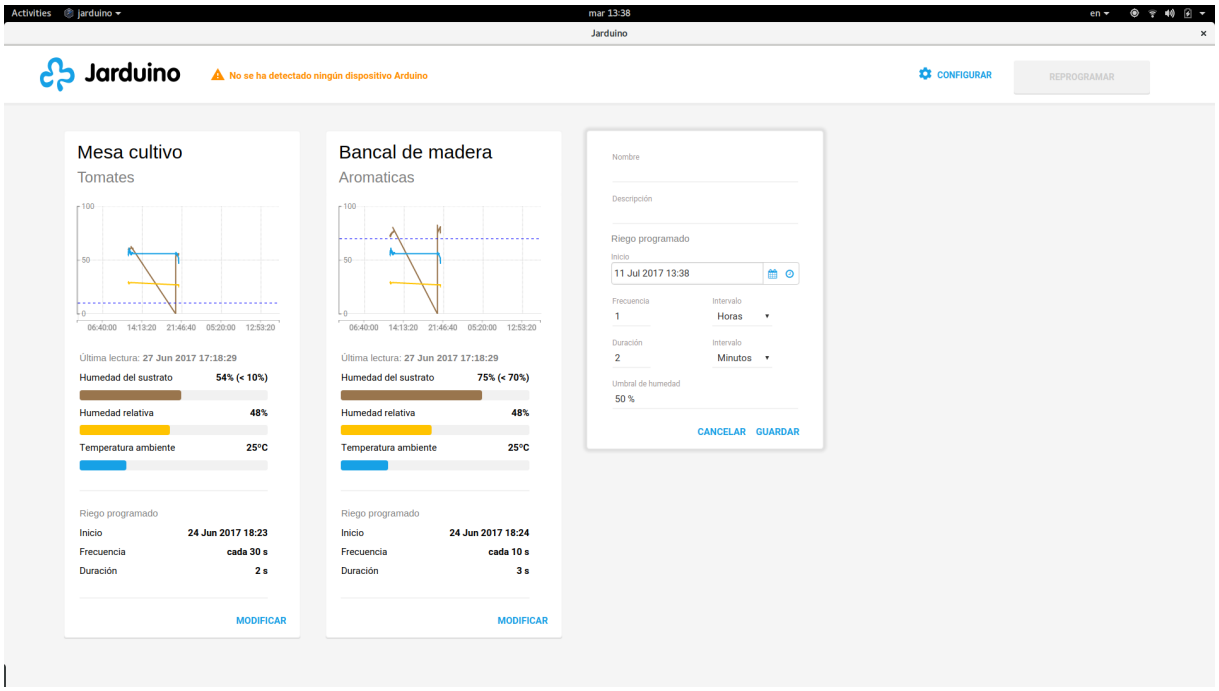


Figura 3.15: Crear una zona de riego

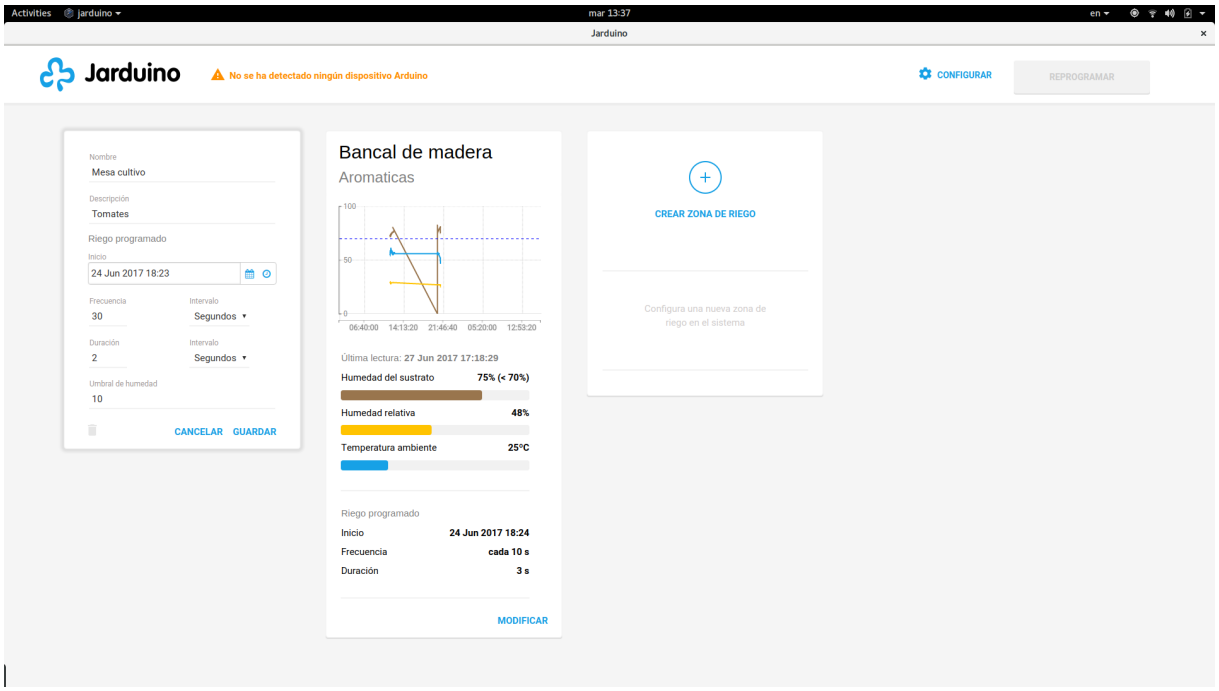


Figura 3.16: Modificar una zona de riego

Capítulo 4

Resumen

La presente memoria de Proyecto de Fin de Carrera describe el desarrollo de un sistema de monitorización y control de riego, autónomo y reprogramable, basado en la plataforma de hardware libre *Arduino*. El sistema se maneja a través de una aplicación de interfaz de usuario que facilita la gestión del control de riego y la reprogramación del dispositivo de riego (implementando con *Arduino*). A este sistema se le ha denominado *Jarduino*[1].

El objetivo de *Jarduino* es aportar una aproximación al problema del control de riego autónomo basado en dispositivos *Arduino*, que introduce una serie de características de las que carecen otros prototipos de este tipo, a saber:

- Gestión de varias zonas de riego con necesidades hídricas diversas.

  Se pueden definir un programa de riego (fecha de inicio y frecuencia de riego) y unos umbrales de riego (qué determinan si se debe regar o no, en base, por ejemplo, a la humeda del sustrato) distintos para cada zona de riego, lo que permite adaptar el riego a las necesidades hídricas de las variedades vegetales que se hayan plantado en cada zona de riego.

- Umbrales de riego múltiples.

  Permite decidir si se riega o no en base a varios factores ambientales: humedad del sustrato, humedad y temperatura ambientales.

- Riego programado.

  Permite programar el inicio de riego, su frecuenca y duración.

- Interfaz de usuario diseñada específicamente, que permite:

[1] En alusión a otros prototipos de monitorización y/o control de riego, que se autodenominan *Garduino*, del inglés *Gardening* + *Arduino*

- Configurar zonas de riego.
- Monitorizar el estado de las zonas de riego: valores de los sensores ambientales y efecto del riego en la humedad del sustrato.
- Reprogramar el dispositivo de riego, aplicando el esquema de riego que se haya configurado para cada zona de riego.

La aproximación al control de riego que representa *Jarduino* facilita la adopción de un esquema de riego que se puede ir mejorando progresivamente, mediante sucesivas reprogramaciones, para adaptarse a distintas condiciones ambientales y necesidades del cultivo, permitiendo ahorrar agua y reduciendo el impacto en la cosecha o la salud de las plantas.

Los componentes del sistema *Jarduino* son:

- Algoritmo de control de riego.

  Implementa la lógica de control de riego (riego programado y umbrales de riego de cada zona de riego) definida en la interfaz de usuario. Se han escrito varias versiones del algoritmo para diferentes implementaciones hardware (ver 2.4), compatibles con diferentes versiones de *Arduino*. Desarrollado usando el subconjunto del lenguaje de programación C compatible con *Arduino*

- Interfaz USB o aplicación de línea de comandos.

  Permite a la aplicación de interfaz de usuario, o a un usuario directamente, interactúar con el dispositivo *Arduino* a través del puerto USB. Escrito en el lenguaje de programación *python*.

- Aplicación de interfaz de usuario.

  Gestiona la configuración de riego de las zonas de riego, monitoriza su estado y permite reprogramar, a través de la interfaz USB, el dispositivo. Escrito en los lenguajes *JavaScript*, *HTML5* y *CSS3*.

- Servidor HTTP.

  Proporciona acceso remoto a los datos generados por un dispositivo *Arduino* que tenga acceso a un interfaz de red. Escrito en el lenguaje de programación *python*.

- Cliente HTTP.

  Emula un el envio de datos desde el dispositivo de riego, para facilitar el desarrollo y la depuración. Escrito en el lenguaje de programación *python*.

Jarduino es un proyecto multidisciplinar, multitecnológico, multiplataforma y multidispositivo.

Multidisciplinar porque aúna conocimientos sobre desarrollo de software, sobre la plataforma de hardware libre *Arduino* y todos su *ecosistema* de componentes hardware y software, sobre diseño de interfaces de usuario, cultivo urbano y algo de electrónica.

Multitecnológico porque para su desarrollo se han empleado diversas técnicas y tecnologías, que cubren todo el espectro de desarrollo de software que va desde la implementación del código fuente del dispositivo de control de riego—basado en Arduino, y desarrollado en C— hasta el último detalle de la interfaz de usuario, escrita en *JavaScript*,*HTML5* y *CSS3*, pasando por una aplicación de línea de comandos escrita en *python* que permite interactuar con el dispositivo *Arduino* a través del puerto USB, además de un servidor y cliente HTTP escrito también en *python*.

Multiplataforma porque todas las tecnologías, lenguajes de programación y herramientas utilizadas para el desarrollo, y por lo tanto todas las piezas de software implementadas, además de la propia plataforma *Arduino*, se pueden utilizar en las plataformas Windows, OSX y GNU/Linux.

Y multidispositivo porque está diseñado para generar código de control de riego para distintas implementaciones hardware del dispositivo de control de riego.

Capítulo 5

Conclusiones

A modo de conclusión, vamos a retomar los objetivos generales de este Proyecto de Fin de Carrerar—definidos en capítulo introductorio, en la sección 1.1—y vamos a valorar hasta que punto se han cumplido con el desarrollo del prototipo demostrador que se presenta.

Para empezar, se fundamentó una hipótesis de trabajo en la sección 2.1, que, resumida, sería que a través de una interfaz de usuario conectada a un dispositivo de monitorización y riego automatizado, se puede mejorar progresivamente la gestión del riego de un pequeño huerto, y que es posible hacerlo con un prototipo que una persona aficionada al hardware libre — en este caso *Arduino* — pueda montar por sí mismo.

La materialización de la segunda parte de esta hipótesis es el prototipo demostrador que se presenta, que ha sido montado por mí mismo, con escasos conocimientos de electrónica y un conocimiento básico de la plataforma *Arduino*.

En cuanto a la posibilidad de mejora de la gestión del riego, la capacidad de monitorización del estado de las zonas de riego, y de reprogramación del dispositivo en base a la configuración de éstas, parecen apuntar en esa dirección, pero la realidad es que cuando se escribe este documento no se ha podido demostrar fehacientemente este punto, pues requeriría un uso prolongado del prototipo en un entorno real, conectado a una salida de agua y aplicado sobre al menos un par de zonas de riego.

El resto de objetivos se valoran a continuación:

- **Monitorización**. Se pretendían monitorizar varios factores ambientales relacionados con la agricultura: humedad del sustrato, humedad ambiental y temperatura ambiental. El prototipo demostrador presentado permite realizar esta función.

- **Zonas de riego independientes**. Se debian controlar varias zonas de riego independientes, con necesidades hídricas distintas. La aplicación de interfaz de usuario presentada realiza esta función.
- **Reprogramación**. Había que facilitar la reprogramación de dispositivos de la familia *Arduino* o compatibles, en base a la configuación de riego definida para cada zona de riego. La aplicación de interfaz de usuario presentada realiza esta función.
- **Accesibilidad**. Era necesario que *Jarduino* pueda ser utilizado desde varios sistemas operativos. Las herramientas que se han utilizado para su desarrollo lo permiten, pero sólo se presenta en su versión para GNU/Linux.
- **Interoperatividad**. Se debían tomar decisiones de diseño que permiten la interoperativad del software y el hardware con otros sistemas. El sistema permite esta interoperativad, pero la explota principalmente en el soporte a varias implementaciones hardware del dispositivo de riego, puesto que el prototipo presentado no expone ninguna API pública que permita la interacción con otras piezas de software.
- **Extensibilidad**. Debería ser posible añadir interfaces para otras familias de dispositivos, o nuevas zonas de control y monitorización de otros factores que puedan influir en la gestión agrícola de un cultivo. Dado que el código fuente del prototipo presentado es de acceso libre y gratuito a través de la comunidad de desarrollo *github* esto es posible, pero en este estadio del desarrollo implica tocar código, no es posible extender la funcionalidad sin conocer el lenguaje de programación del componente que se quiera modificar.
- **Sencillez de uso**. La aplicación de interfaz de usuario permite configurar las zonas de riego y reprogramar el dispositivo con muy poco esfuerzo.

Estos objetivos sintetizados, consisten en contar con un prototipo demostrar que permita comprobar si, conjugando la monitorización de los datos obtenidos por los distintos sensores instalados en las zonas de riego, su análisis, y la capacidad de reprogramar el dispositivo que haga de unidad de control, es posible optimizar la gestión de un huerto de tamaño doméstico mediante sucesivas iteraciones de mejora progresiva.

Por lo tanto, y en síntesis, se puede afirmar el prototipo demostrador presentado permite comprobar si, conjugando la monitorización de los datos obtenidos por los distintos sensores instalados en las zonas de riego, su análisis, y la capacidad de reprogramar el dispositivo de control de riego, es posible optimizar la gestión de un huerto de tamaño doméstico mediante sucesivas iteraciones de mejora progresiva.

Referencias

[AM3a] *ATmega328P datasheet*. https://goo.gl/m6DvSz.

[AM3b] *ATmega32U4 datasheet*. https://goo.gl/PCDvcA.

[ARDa] *Arduino Makefile*. http://hardwarefun.com/tutorials/compiling-arduino-sketches-using-makefile.

[ARDb] *Arduino Uno*. https://store.arduino.cc/arduino-uno-rev3.

[ARDc] *Arduino Yún*. https://store.arduino.cc/arduino-yun.

[AVR] *AVR*. https://goo.gl/bZh4Qs.

[Con03] Jim Conallen. *Building Web Applications with UML. Second Edition.* Addison-Wesley, 2003.

[GARa] *Gard-A-Water*. http://www.instructables.com/id/Watering-Garden-with-GARD-A-WATER-Arduino-Project/.

[GARb] *Garduino*. http://makezine.com/projects/make-18/garduino-geek-gardening/.

[git] *Github*. https://github.com.

[INSa] *Garduino: Gardening + Arduino*. http://www.instructables.com/id/Garduino-Gardening-Arduino/.

[INSb] *Instructables*. http://www.instructables.com/.

[NTP] *NTP*. https://goo.gl/QsuymG.

[OG] *OpenGarden*. https://goo.gl/nkmXHu.

[WIKa] *API*. https://goo.gl/sgB6JV.

[WIKb] *Arduino*. https://en.wikipedia.org/wiki/Arduino.

[WIKc] *Do It Yourself*. https://goo.gl/Rf8ycq.

[WIKd] *HTTP*. https://goo.gl/R3bHqY.

[WIKe] *IDE*. https://goo.gl/FFo9hi.

[WIKf] *javascript*. https://es.wikipedia.org/wiki/JavaScript.

[WIKg] *JavaScript Object Notation*. https://en.wikipedia.org/wiki/JSON.

[WIKh] *Open Source Hardware*. https://goo.gl/e5OcmX.

[WIKi] *python*. https://es.wikipedia.org/wiki/Python.

[WIKj] *Sustrato biológico*. https://goo.gl/GmOkYd.

[WIKk] *URL*. https://en.wikipedia.org/wiki/URL.