Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Comunicación entre módulos para la construcción de jardines inteligentes

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Delgado Sanchis, Alejandro

Tutor: Poza Luján, José Luis

Co-tutor: Posadas Yagüe, Juan Luis

2015-2016

Resumen

En la actualidad, las comunicaciones se han convertido en algo imprescindible para cualquier dispositivo. Sin embargo, este campo todavía no ha sido desarrollado en su totalidad. El objetivo de este proyecto es ampliar la conexión de una mayor cantidad de elementos a la red. Para poder llevar a cabo este proceso, se diseña una arquitectura con diversos protocolos de comunicaciones. Así pues, este trabajo se basa en un entorno de jardines y pretende obtener la autonomía de las macetas a través de su robotización.

Para conseguir que los protocolos sean más fiables y eficientes, se implantan en un ámbito lo más real posible. Se establece de esta forma la necesidad de crear un dispositivo hardware y una aplicación software. El fin último es lograr que estas macetas puedan ser gestionadas o monitorizadas desde diferentes partes del mundo.

Palabras clave: Comunicaciones, Servidor REST, Base de datos, Arduino, Android

Abstract

Nowadays communications have become a must for any device. However, this field has not yet been fully developed. The purpose of this project is to expand the connection of a great number of elements to the network. To carry out this process, an architecture with different communication protocols is designed. Thereby this work is based on an environment of gardens and seeks the autonomy of the pots through its robotization.

To make these protocols more reliable and efficient they are implemented in an environment as real as possible. Thus, the need of a hardware device and a software application is established. The ultimate goal is to ensure that these pots can be managed or monitored from different parts of the world.

Keywords: Communications, Server REST, Database, Arduino, Android

Índice General

Índice de Figuras

Índice de Tablas

Índice de códigos

CAPÍTULO 1

Introducción

En este primer capítulo son expuestos los motivos que han conducido a la elaboración de este proyecto, así como los objetivos que se pretenden alcanzar, y una descripción de los capítulos que forman la memoria del trabajo realizado.

1.1 Motivación

Hoy en día vivimos en un mundo totalmente conectado. Gracias a las comunicaciones, se puede estar informado de lo que ocurre en cualquier parte del mundo con un simple dispositivo que cabe en la palma de la mano. Sin embargo, aún estamos a las puertas de poder comunicarnos con más dispositivos.

Este proyecto logra acercarnos más a un mundo que permita conectar mayor variedad de dispositivos de uso cotidiano a través de Internet para su posible uso desde cualquier parte. En especial este proyecto se centra en los jardines. La automatización de este tipo de entornos se encarga de la optimización de jardines estáticos, sin embargo, en el caso de plantas en macetas, se puede ampliar las posibilidades con la monitorización de parámetros como la temperatura, humedad, luminosidad o incluso niveles de agua en depósitos ubicados en la maceta. Además, también se pueden ofrecer posibilidades de automatización de tareas como la de riego.

Por este motivo se va a construir la infraestructura y los protocolos necesarios para poder encargarse del mantenimiento de un conjunto distribuido de macetas desde cualquier sitio e incluso poder desarrollar en proyectos posteriores inteligencia propia en el jardín.

1.2 Estructura de la memoria

Esta memoria está compuesta de seis capítulos. A continuación se va a realizar una pequeña descripción de cada uno de ellos.

- **Capítulo 1, Introducción**: Introducción al proyecto con los objetivos y la motivación que se ha tenido para realizarlo.

- **Capítulo 2, Entorno de realización:** En este capítulo se lleva a cabo la búsqueda y análisis de sistemas que se asemejen con este proyecto.
- **Capítulo 3, Especificación de requisitos**: Aquí se empieza a describir los requerimientos que va a tener el sistema completo de comunicaciones y los elementos que la componen.
- **Capítulo 4, Diseño**: En esta parte ya se diseñan cada uno de los elementos y protocolos que van a ser empleados en el proyecto y asimismo las herramientas necesarias para llevarlo a cabo.
- **Capítulo 5, Implementación, Implantación y Evaluación**: Este capítulo implementa todos los elementos que constituyen la arquitectura de comunicaciones. También se realiza la descripción de cómo se ha implantado en un sistema real y sus consecuentes pruebas.
- **Capítulo 6, Conclusiones**: En el último capítulo se recogen las principales conclusiones a las que se ha llegado, los problemas encontrados y las posibles ampliaciones futuras.

CAPÍTULO 2

Entorno de realización

Para poder empezar a diseñar el proyecto es conveniente basarse en otros sistemas que contengan parte de la funcionalidad que se pretende realizar. Para ello se buscan y analizan sistemas parecidos de los cuales se extraen ideas y diseños que pueden ser útiles para desarrollar las implementaciones.

2.1 Sistemas Similares

Open Garden

El sistema Open Garden consta de varios nodos, en este caso son sensores que se conectan a una pasarela mediante un módulo de 433Mhz. Esta pasarela es la encargada de realizar la conexión con el router principal por wifi o 3G.

Por otra parte, existe el servidor de tipo Apache, que a su vez utiliza una base de datos MySQL [9]. Esta base de datos es imprescindible para guardar los datos recogidos por los sensores que existen dentro de cada nodo.

Por último, para que el usuario pueda monitorizar el jardín, existen dos aplicaciones. Una vía web y otra en forma de aplicación móvil. Las dos se conectan a la misma red en la que se encuentra el servidor para poder monitorizarlo.

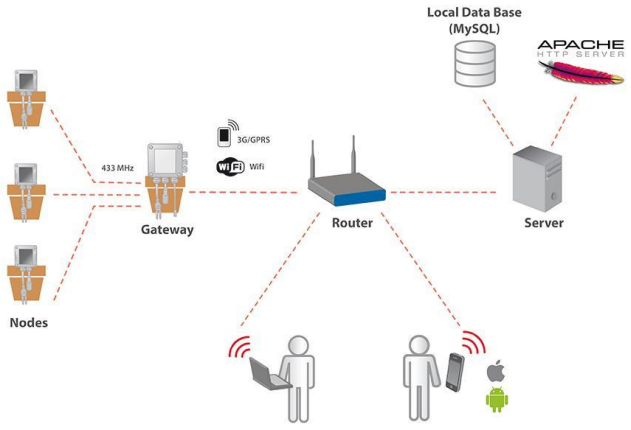


Figura 1 : Arquitectura Open Garden[1]

[1] Fuente :https://www.cooking-hacks.com/documentation/tutorials/open-garden-hydroponics-irrigation-system-sensors-plant-monitoring

PotPet

Este sistema está pensado para gente que le cuesta mantener sus plantas en buen estado. Para ello, cada macetero tiene la posibilidad de desplazarse dependiendo de sus necesidades. Por ejemplo, si la planta que contiene una maceta necesita luz, tiene la posibilidad de desplazarse hasta conseguirla. También tiene la característica de moverse para llamar la atención de las personas y expresar la necesidad de agua.

En cuanto a la tecnología, a nivel de hardware utiliza un controlador llamado Arduino [1]. El medio de comunicación entre las macetas es XBEE, un módulo compatible con ZigBee, habitual entre comunicación de nodos, sobre todo cuando se trata de comunicaciones muy sencillas. Estos módulos envían la información a un servidor que se encarga de gestionar las macetas.

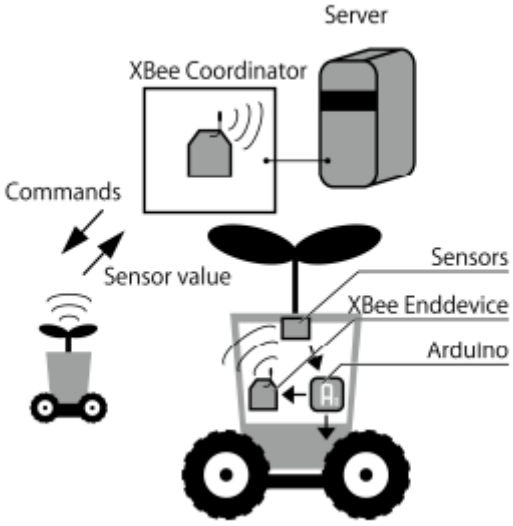


Figura 2 : Arquitectura PotPet [2]

RSEM (Red de Sensores para Experimentos Medioambientales)

Se trata de un diseño de la universidad de Arizona a gran escala, que utiliza estaciones meteorológicas para recoger información sobre los sensores y enviarlos a un centro de datos. La finalidad de este proyecto es poder estudiar las consecuencias sobre el cambio climático.

Su arquitectura es bastante compleja. Utiliza estaciones meteorológicas compuestas por sensores, actuadores y un pequeño procesador para realizar cálculos a menor escala. También tiene nodos que se conectan con las estaciones meteorológicas usando para ello una red privada de la que forma

[2] Fuente :http://sappari.org/pdf/potpet_tei2011.pdf

parte un servidor principal, que tiene por tarea enviar la información recogida al centro de datos. A continuación, se muestra el esquema de dicha arquitectura.

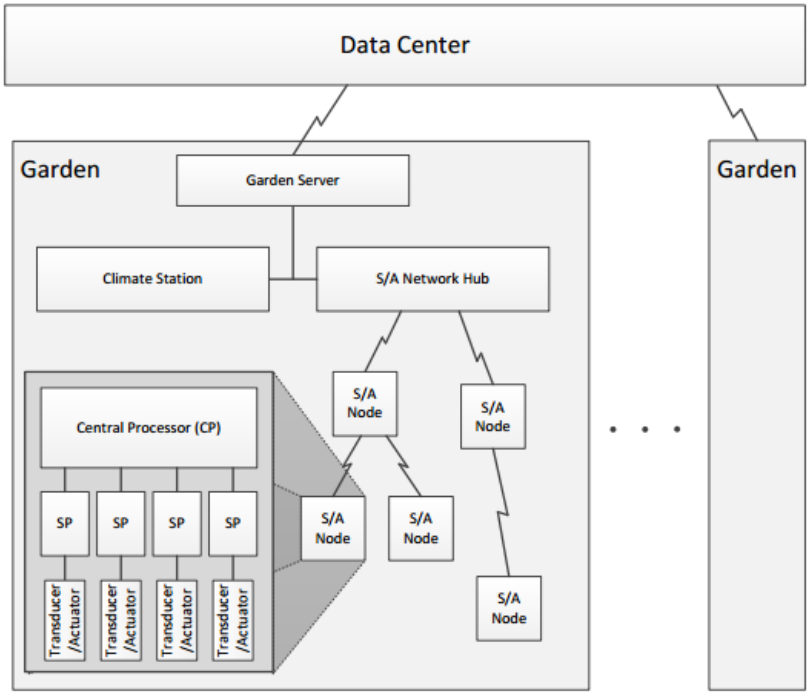


Figura 3 : Arquitectura RSEM [3]

La parte que más puede servir de ayuda para la realización del presente proyecto es la de Garden Server, es decir, la del servidor. Utiliza unos pequeños computadores llamados Moxa V2101-T-LX. En cada uno de ellos está instalado un Linux Debian 5.0 y un servidor Apache. Las comunicaciones se realizan mediante Wifi.

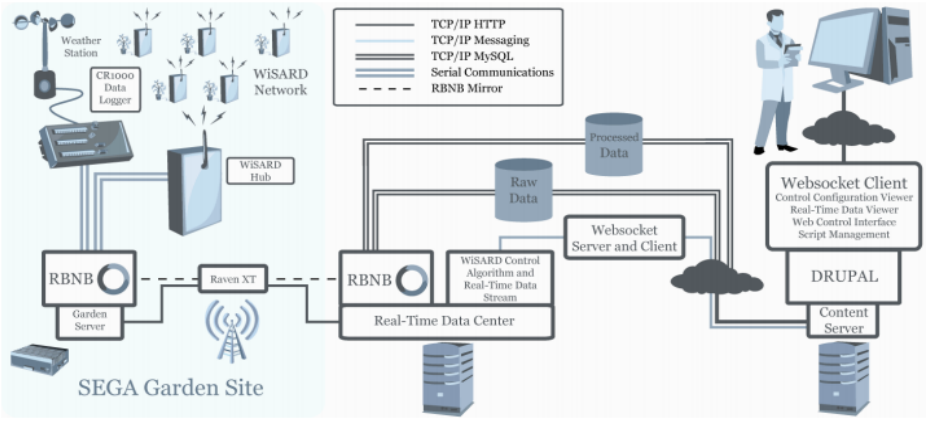


Figura 4 : Esquema General de RSEM[3]

[3]Fuente:http://disalw3.epfl.ch/miscellaneous/events_news/docs/Flikkema_seminar_19112012.pdf

En la figura 4 se puede observar todo el sistema de procesado de la información procedente de los sensores.

MQTT

Este entorno es uno de los más innovadores ya que posee algunas características no vistas hasta ahora. Los sensores están gobernados por un Arduino, que a su vez se conecta a una Raspberry [2] y ésta, envía la lectura de los datos mediante un protocolo llamado MQTT, publicándolas más tarde a internet. Para poder acceder a estos datos mediante web existe un servidor en NodeJS que se conecta al servicio de MQTT y los adapta para mostrarlos a un cliente por medio de un navegador.

A continuación, se muestra un esquema de los principales elementos de la arquitectura.

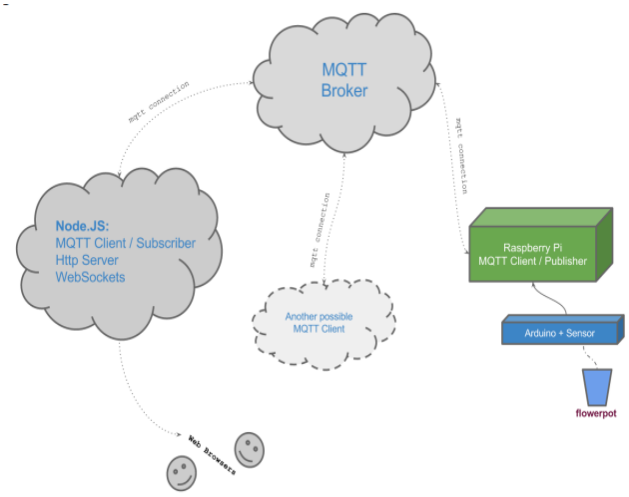


Figura 5 : Esquema MQTT[4]

2.3 Análisis de sistemas similares

Primero se realiza un análisis de las características cuantificables de los sistemas anteriormente mostrados. En la siguiente tabla se expone este análisis con el siguiente significado.

- **Sistema**: Nombre del dispositivo que se va a analizar.
- **Servidor**: Tipo de servidor que tiene el sistema.
- **Medio de comunicación**: Indica el tipo de tecnología que utiliza para realizar las comunicaciones.
- **Modelo de controlador**: Se emplea para saber el tipo de controlador que utiliza.

[4] Fuente : https://adriandubiel.com/2015/04/09/mqtt-iot-protocol-test-flower-pot-moisture-web-monitoring/

- **Persistencia**: Indica con que tecnología se guardan los datos del sistema para hacerlos persistentes.
- **Comercial**: Significa si el sistema está en venta o no.

Sistema	Servidor	Medio de comunicación	Modelo de controlador	Persistencia	Comercial
Open Garden	Apache	Wifi o 3G	Arduino	MySql	Si
PotPet	(1)	Xbee	Arduino	(1)	No
MQTT	Node JS	Cable	Arduino y Raspberry	(1)	No
RSEM	Apache	Wifi	(1)	MySql	No

(1) No especificado

Tabla 1 : Análisis cuantitativo

En estos sistemas no existe mucha información sobre el tipo de arquitectura que utilizan o sus protocolos implementados, uno de los motivos por los cuales resulta interesante este proyecto.

Lo siguiente que se va realizar es un análisis cualitativo de los sistemas anteriores, teniendo en cuenta las siguientes características:

- **Diseño**: Mide el grado de aspecto y diseño que tiene.
- **Complejidad**: Mide el nivel de complejidad que tiene la arquitectura.
- **Fiabilidad**: Mide la fiabilidad que tiene todo el sistema referente a las comunicaciones.
- **Facilidad de instalación**: Representa el nivel de complejidad que tendrá el sistema a la hora de instalarlo.

Todo esto se mide con la siguiente escala:

1 – No adecuado

2 – Poco adecuado

3 – Adecuado

4 – Muy adecuado

Sistema	Diseño	Complejidad	Fiabilidad	Facilidad de instalación
Open Garden	4	4	3	3
PotPet	3	3	2	2
MQTT	3	2	3	2
RSEM	4	2	4	1

Tabla 2 : Análisis cualitativo

2.4 Síntesis

Después de analizar sistemas anteriores, se procede a extraer alguna de las características existentes en el presente proyecto. Para nombrarlas es necesario un código que sirva para diferenciarlas.

CA1: Comunicar cada parte de la arquitectura proporcionando una gestión de nodos internos con las peticiones externas.

CA2: Realizar persistencia de los datos que lo requieran en un SGBD relacional.

CA3: Conexiones robustas con confirmaciones de recepción para los mensajes.

CA4: Realizar una gestión integral de detección, tratamiento de errores y almacenamiento de registro temporal de los mismos.

2.5 Tecnología a emplear

Teniendo en mente una idea de lo que se está usando en este ámbito de sistemas, resultan más evidentes las características que debe tener el proyecto. Partiendo de este punto se puede empezar a hablar de las tecnologías necesarias.

En la parte central de las comunicaciones, es decir el servidor, hay que definir los protocolos y el lenguaje que va seguir. En los sistemas anteriores no se habla de ningún protocolo, solo de qué tipo de software tiene cada servidor. El más utilizado es Apache pero para este proyecto resulta más interesante una de sus variantes. Se trata de Tomcat [5], que permite la integración con el entorno de desarrollo eclipse [6]. Este entorno es el que se utiliza para desarrollar los servicios en el servidor.

En la parte referente a la persistencia se observa que los sistemas que la utilizan siguen la tecnología MySQL, la más recurrida referente a bases de datos relacionales. Por lo tanto, también se va a centrar en ella el desarrollo de la persistencia. Más tarde se verá el porqué de esta decisión.

Por último queda por elegir el modelo de controlador hardware, que servirá para realizar pruebas en todo lo referente a las comunicaciones. Se usará finalmente Arduino. Este dispositivo es de código abierto y muy fácil de utilizar para realizar proyectos con sensores y actuadores.

2.6 Conclusiones

En este capítulo se han elegido diferentes sistemas que tienes parecido con el que se va a desarrollar. Se han analizado tanto cuantificablemente como cualitativamente. Una vez hecho esto se han extraído las posibles características que se van a mantener en el proyecto. Después de dichas características, se ha mencionado en el apartado técnico las tecnologías que van a servir de ayuda con las futuras implementaciones.

Capítulo 3

Especificación de requisitos

Llegado a este capítulo, ya se ha podido ver qué tipos de arquitecturas y que tecnologías se están usando actualmente en el mercado. A partir de aquí ya se empieza a pensar que deberá hacer el sistema y con quien se tendrá que comunicar.

3.1 Definiciones, acrónimos y abreviaturas

Sistema: Se usa este término para referirse globalmente a todas las partes que componen el proyecto, es decir, la parte de macetas, comunicaciones y aplicación.

Proyecto: Este término se utiliza para concretar sobre la parte de dentro del sistema que se está desarrollando. En concreto toda la parte de comunicación donde intervenga el servidor.

Arduino: Aunque más tarde se habla con más detenimiento sobre este término, solo cabe destacar que se trata de una placa con un circuito impreso y un controlador que permite recibir o escribir de las entradas y salidas conectadas a él.

Servidor: Para este proyecto el servidor será una ordenador con un software y unas herramientas que se encargan de gestionar y procesar peticiones.

Robot-maceta: Se trata de un pequeño dispositivo hardware que será el encargado de cumplir las órdenes enviadas desde el servidor. Por ejemplo avanzar o enviar valor de un sensor. También se le llama maceta porque en el sistema general deberá llevar una planta sobre él y de ahí jardines inteligentes.

REST [3] [4]: Son las iniciales de (Representational State Transfer) que en español significa Transferencia de Estado Representacional. Es el protocolo que se usa para las peticiones de la aplicación o del robot-maceta en este proyecto.

APP: Es un acrónimo referido a la palabra aplicación, al que se recurre para referirse a la aplicación móvil.

3.2 Descripción general

Este proyecto forma parte de un sistema completo que se va a llevar a cabo entre varios alumnos de esta universidad. El sistema está constituido en el nivel más bajo de macetas equipadas con diferentes actuadores y sensores conectados a una placa Arduino. La función de cada macetero es vigilar y satisfacer las necesidades de la planta que contiene. Para ello se controla que tenga agua, luz y otras necesidades.

El siguiente nivel que se va a llevar a cabo es la comunicación de los robots-maceta con otras y estas a su vez con el servidor. En el último nivel está el servidor, que atenderá las peticiones tanto de la aplicación móvil como de los robots-maceta. Con dichas peticiones se podrá controlar factores como el nivel de agua, luz, humedad, movimiento de macetas etc.

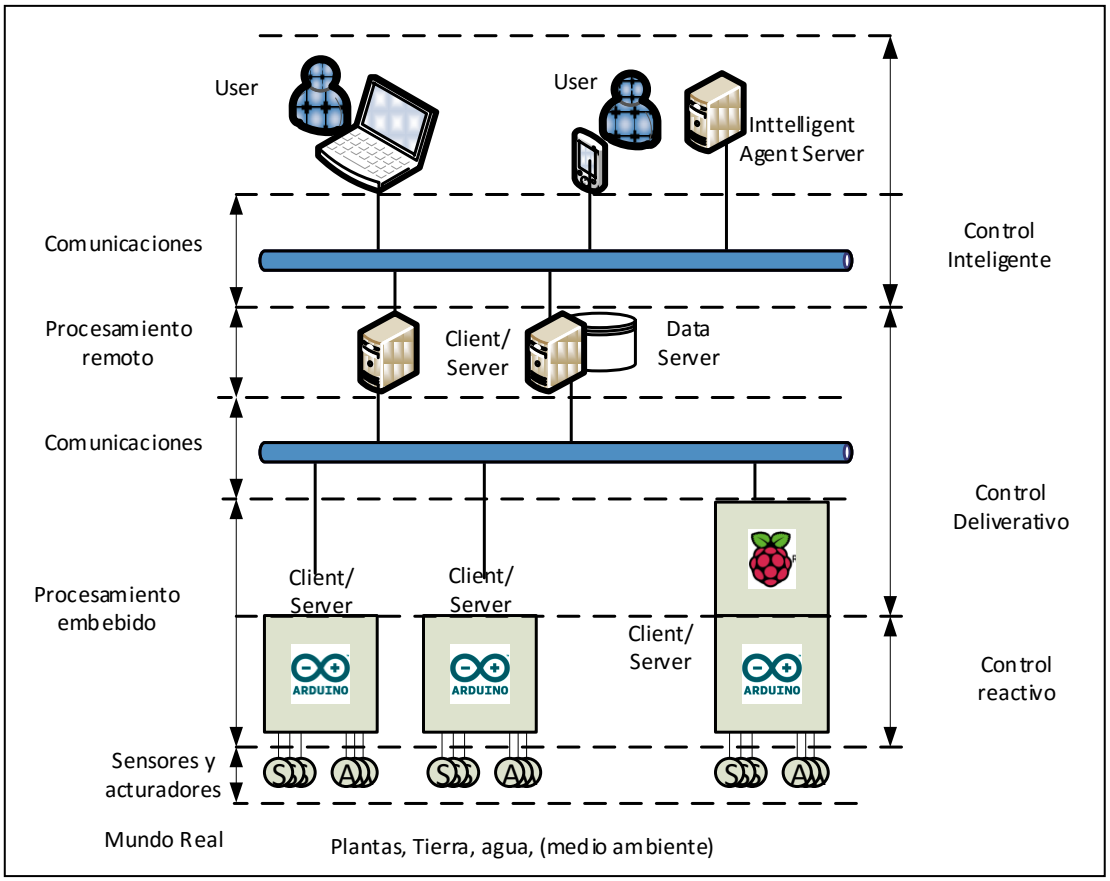


Figura 6 : Diagrama del sistema completo

En el diagrama superior se puede observar los diferentes niveles de los que está compuesto el sistema general.

3.3 Personas Involucradas

Aquí se describen todas las personas que componen el proyecto y el rol que desempeña cada uno.

Nombre	Alejandro Delgado
Rol	Desarrollador comunicaciones
Categoría profesional	Estudiante
Responsabilidades	Implementación comunicaciones módulo-servidor y servidor-aplicación
Información de contacto	aldelsa1@inf.upv.es
Aprobación	-

Tabla 3 : Componente Alejandro Delgado

Nombre	Axel Guzmán Godia
Rol	Desarrollador automatización.
Categoría profesional	Estudiante
Responsabilidades	Desarrollar el código Arduino y montar el hardware
Información de contacto	axguzgo@inf.upv.es
Aprobación	-

Tabla 4 : Componente Axel Guzmán

Nombre	Laia Ferrando Ferragut
Rol	Desarrollador Aplicación móvil.
Categoría profesional	Estudiante
Responsabilidades	Diseño aplicación móvil
Información de contacto	laferfe@inf.upv.es
Aprobación	-

Tabla 5 : Componente Laia Ferrando

Nombre	Israel Beltrán Arias
Rol	Desarrollador comunicaciones
Categoría profesional	Estudiante
Responsabilidades	Implementación comunicaciones entre módulos.
Información de contacto	isbelar@inf.upv.es
Aprobación	-

Tabla 6 : Componente Israel Beltrán

Nombre	José Luis Poza Luján
Rol	Supervisor.
Categoría profesional	Profesor Contratado Doctor
Responsabilidades	Supervisar proyecto.
Información de contacto	joplu@disca.upv.es

Tabla 7 : Componente José Luis Poza

Nombre	Juan Luis Posadas Yagüe
Rol	Supervisor.
Categoría profesional	Profesor Contratado Doctor
Responsabilidades	Supervisar proyecto
Información de contacto	jposadas@disca.upv.es

Tabla 8 : Componente Juan Luis Posadas

3.4 Casos de Uso

Una vez que ya se ha descrito y explicado cada parte del sistema se procede a realizar los casos de uso.

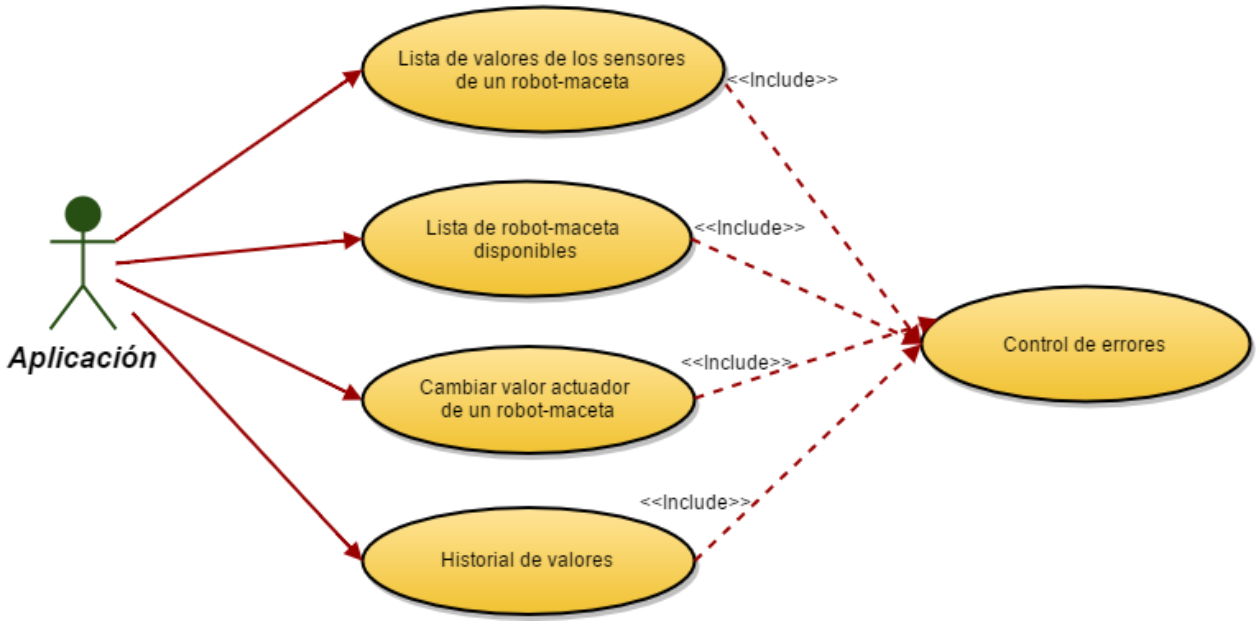


Figura 7 : Caso de uso Aplicación

En este primer diagrama se describe las posibles peticiones que puede hacer la aplicación al sistema. Más tarde se ve qué tipo de contestación va a tener cada petición.

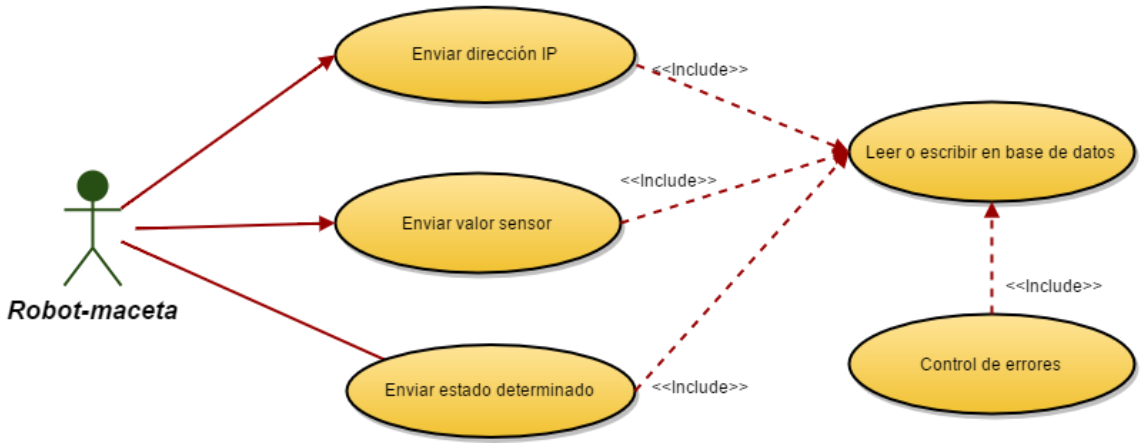


Figura 8 : Caso de uso Robot-maceta

La figura 8 describe los posibles casos de uso que puede realizar el robot-maceta. Por último solo queda mostrar un diagrama referente a los casos de uso que puede tener el servidor. Todos ellos se explicaran más detenidamente en el siguiente apartado.

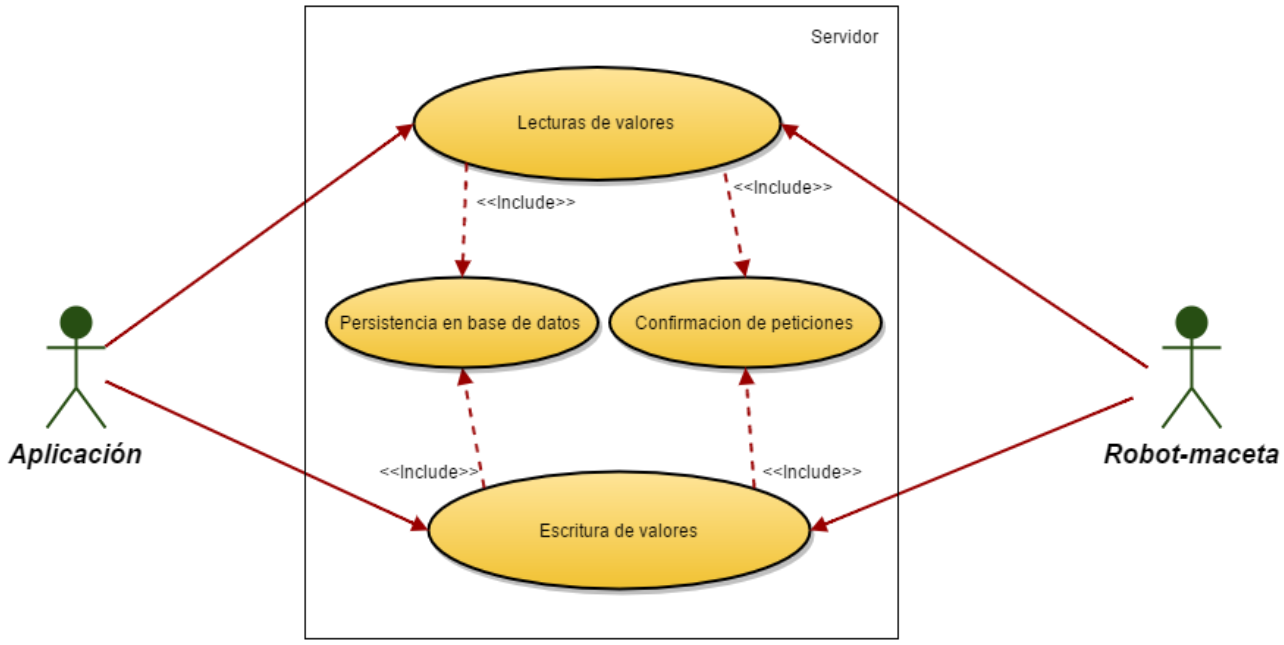


Figura 9 : Caso de uso Servidor

3.5 Requisitos

En esta sección se describe cada uno de los requisitos que se debe cumplir para poder realizar la comunicación y gestión de la información correctamente, teniendo en cuenta los casos de uso anteriores. Para ello se clasifican en las partes que componen la arquitectura.

- Requisitos que se deben cumplir para la aplicación.
- Requisitos que se deben cumplir en el servidor.
- Requisitos que se deben cumplir en los robots.

Requisitos de la Aplicación

Número de requisito	REA01		
Nombre de requisito	Valores sensores		
Tipo	☒ Requisito	☐ Restricción	
Descripción del requisito	La aplicación solicita una lista de los valores de los distintos sensores que existen en el robot.		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla 9 : Requisito aplicación 01

Número de requisito	REA02		
Nombre de requisito	Cambiar valor actuador		
Tipo	☒ Requisito	☐ Restricción	
Descripción del requisito	La aplicación envía la información del identificador de la maceta, el actuador y el valor al que quiere que poner el actuador.		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla 10 : Requisito aplicación 02

Número de requisito	REA03		
Nombre de requisito	Lista Macetas		
Tipo	☒ Requisito	☐ Restricción	
Descripción del requisito	La aplicación en cualquier momento puede solicitar el número de macetas que estén operativas o que tengan un estado determinado.		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla 11 : Requisito aplicación 03

Número de requisito	REA04		
Nombre de requisito	Historial valores		
Tipo	☒ Requisito	☐ Restricción	
Descripción del requisito	La aplicación puede pedir el historial referente a un sensor y aun robot determinado.		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla 12 : Requisito aplicación 04

Requisitos de Servidor

Número de requisito	RES01		
Nombre de requisito	Saber dirección IP		
Tipo	☒ Requisito	☐ Restricción	
Descripción del requisito	El servidor debe saber en todo momento cual es la dirección de cada de uno de los robots que tiene a su cargo.		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla 13 : Requisito Servidor 01

Número de requisito	RES02		
Nombre de requisito	Recibir alarma		
Tipo	☒ Requisito	☐ Restricción	
Descripción del requisito	El servidor puede recibir un aviso de alarma indicando el estado de alguno de sus robots.		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla 14 : Requisito Servidor 02

Número de requisito	RES03		
Nombre de requisito	Almacenar valores		
Tipo	☒ Requisito	☐ Restricción	
Descripción del requisito	El servidor debe almacenar de forma persistente el valor tanto de un sensor como de un actuador o un estado.		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla 15 : Requisito Servidor 03

Requisitos de Robots-Macetas

Número de requisito	RER01		
Nombre de requisito	Enviar Valor		
Tipo	☒ Requisito	☐ Restricción	
Descripción del requisito	Los robots-maceta cada vez que cambian un valor de un sensor deben comunicarlo al servidor.		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla 16 : Requisito Robot 01

Número de requisito	RER02		
Nombre de requisito	Enviar IP		
Tipo	☒ Requisito	☐ Restricción	
Descripción del requisito	Cada robot-maceta debe enviar su dirección IP en el momento que realice la conexión con la red .		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla 17 : Requisito Robot 02

Número de requisito	RER03		
Nombre de requisito	Enviar Estado		
Tipo	☒ Requisito	☐ Restricción	
Descripción del requisito	Cada robot-maceta debe enviar su estado al servidor en el momento que él quiera.		
Prioridad del requisito	☒ Alta/Esencial	☐ Media/Deseado	☐ Baja/ Opcional

Tabla 18 : Requisito Robot 03

3.6 Conclusiones

En este apartado se ha descrito el producto general con todos los componentes y su labor. Seguidamente se han extraído los aspectos clave que hay que implementar y todos los componentes. Finalmente se han redactado los requisitos funcionales necesarios para lograr las metas establecidas. Por consecuente el siguiente paso es abordar el diseño de todos los elementos que componen la arquitectura de comunicaciones.

Capítulo 4

Diseño

Una vez llegado a este apartado el proyecto ya está bien detallado. Se debe tener claro que elementos hay que diseñar, por ello aquí se va a definir la arquitectura general, los componentes hardware y los componentes software.

4.1 Arquitectura General

La figura 10 muestra la arquitectura general y los elementos que componen el sistema.

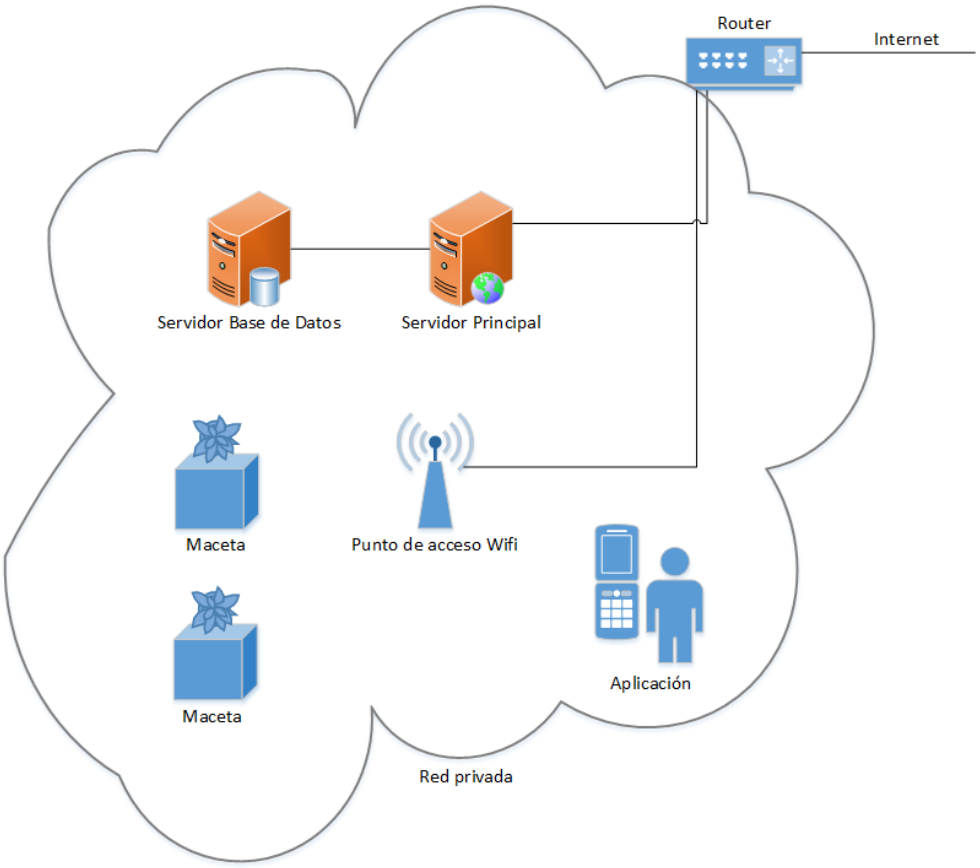


Figura 10 : Diagrama de la arquitectura

El servidor es el principal elemento de la comunicación ya que hace de intermediario entre las macetas y la aplicación. Esta tarea resulta compleja porque se necesitan encontrar un protocolo que se adapte bien a cada una de las partes.

En la arquitectura también hay un elemento de persistencia como lo es una base de datos. Resulta muy útil para poder guardar información de las macetas, de los valores de sensores y los actuadores. Este elemento es imprescindible porque permite desacoplar los robots-maceta de las peticiones que provienen de la aplicación.

Por último se realiza el despliegue en una red privada, aunque también se podrían separar los componentes en diferentes redes y conectarlas mediante internet. En tal caso el sistema podría caer en fallos de seguridad.

4.2 Persistencia

Lo primero que se diseña es la parte de la persistencia. Para conseguir esta persistencia se usa una base de datos. Pero antes es necesario saber qué tipo de base de datos puede resultar más apropiada para este proyecto. Existen diferentes tipos de bases de datos pero las más utilizadas y documentación tienen son las siguientes:

- Bases de datos relacionales
- Base de datos orientada a objetos

Después de analizar bien los requisitos se llega a la conclusión de que el proyecto requiere una implementación y un diseño de persistencia no demasiado complejo. Por ello se decide utilizar una base de datos relacional.

Seleccionado el tipo de base de datos, se comienza a diseñar el esquema que se va a seguir.

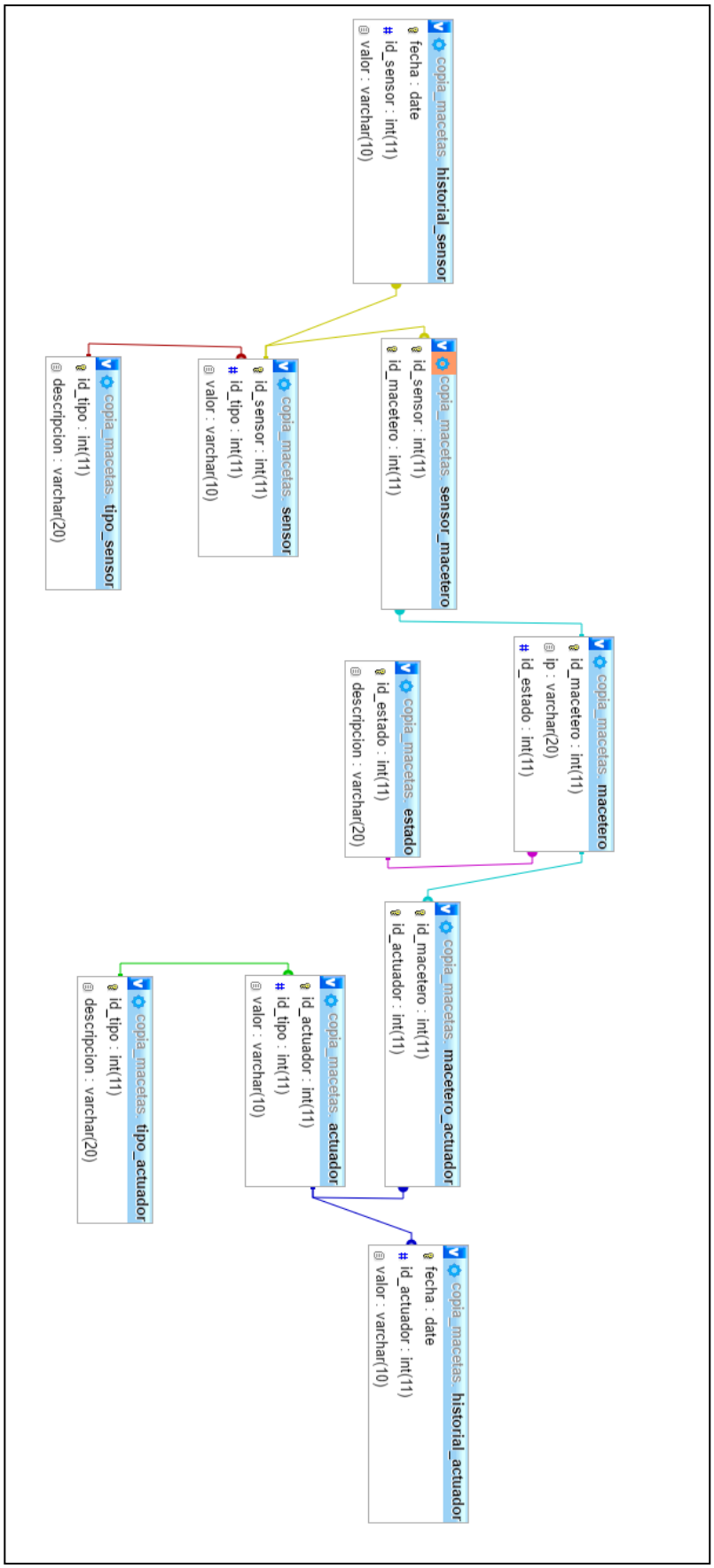


Figura 11 : Esquema base de datos

Realizado dicho diseño, el próximo paso es seleccionar qué tipo de tecnología debe implementar el servidor. Se puede ver en el capítulo dos que el sistema de persistencia más utilizado es MySQL.

Este modelo resulta muy conveniente para el proyecto ya que solo es necesario guardar valores simples, enteros o cadenas. También cabe destacar que para insertar valores en la tabla de *historial_actuador* o la de *historial_sensor* se va a hacer uso de triggers. Los triggers son procesos que se disparan cuando se inserta, actualiza o borra un registro de una tabla determinada. Por ello se diseña un trigger que cada vez que se cambie el valor de un sensor, automáticamente inserta en el historial correspondiente el valor anterior a la actualización. Todo esto lo permite MySQL.

4.3 Hardware

El presente proyecto no requiere un hardware demasiado complejo. En particular, para el servidor se va a necesitar un ordenador lo suficientemente potente para poder atender las diferentes peticiones tanto de los robots-maceta como la de las aplicaciones de los usuarios. También se necesita una tarjeta de red que puede ser inalámbrica o por cable aunque por motivos de eficiencia se recomienda por cable Ethernet. Como elemento de interconexión se requiere un router, al cual se conecten todos los elementos que componen la arquitectura.

A pesar de no tener que desarrollar nada de la parte del robot, va a resultar interesante diseñar un prototipo de robot-maceta con unos pocos elementos de entrada y salida. El prototipo es necesario para poder testear el sistema completo y ver que comunica todo adecuadamente.

Para poder montar el robot-maceta se va a hacer uso de un dispositivo Arduino, que es el mismo controlador que utiliza otro miembro del equipo para conseguir el movimiento de los robots-maceta.

Figura 12 : Arduino Uno

A parte del dispositivo principal (Figura 12) se debe tener un módulo que permita realizar las comunicaciones desde el Arduino. Dado que dichas comunicaciones se realizan por Wifi (IEEE 802.11), se hace uso de un dispositivo llamado ESP8266 (Figura 13) que es el encargado de conectar el robot-maceta con la red.

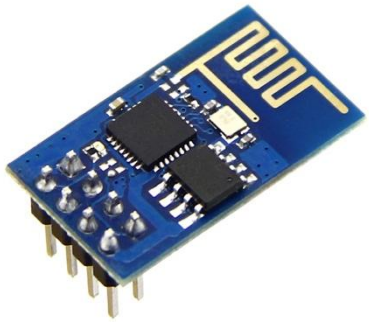


Figura 13 : Módulo ESP8266

Elegidos estos módulos solo queda concretar qué elementos electrónicos de entrada y salida hacen falta. Como se trata de montar un dispositivo para testear las comunicaciones no se van a necesitar muchos elementos en esta parte. Únicamente se va a utilizar un diodo led que hace la función actuador y un sensor de luz. Además de cables para conectar los sensores y placa donde se distribuye todo.

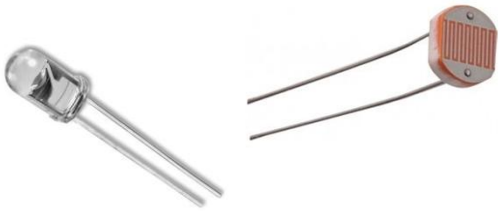


Figura 14 : Diodo Led (Izquierda) y Sensor LDR de luz (Derecha)

La figura 14 muestra el resto de hardware que se necesita para poder llevar a cabo el proyecto y poder probarlo en un entorno real.

4.4 Protocolo de comunicación

Esta va a ser la parte principal que se ha de desarrollar e implementar en el sistema. Se deben diseñar unas pautas para conseguir una comunicación fiable y clara entre las dos partes que se quieren comunicar. Hay diferentes protocolos que pueden ayudar a conseguirlo. En este caso se utiliza un

protocolo llamado REST, el cual se adapta muy bien a este tipo de proyectos con sensores. Esto ya se mostrará en el capítulo de implementación.

El protocolo está dividido en tres partes:

- Envío de lectura de sensores al servidor.
- Envío de valores actuadores por parte de la aplicación al robot - maceta.
- Envío de alarmas al servidor.

Para cada una de estas partes se puede utilizar una sincronía diferente para conseguir así una mayor robustez.

Es importante destacar que tratándose de un proyecto en grupo, hay que tener en cuenta lo que necesitan y ofrecen el resto de componentes del equipo. Por ello hay que estar continuamente en contacto.

Comunicación lectura de sensores al servidor

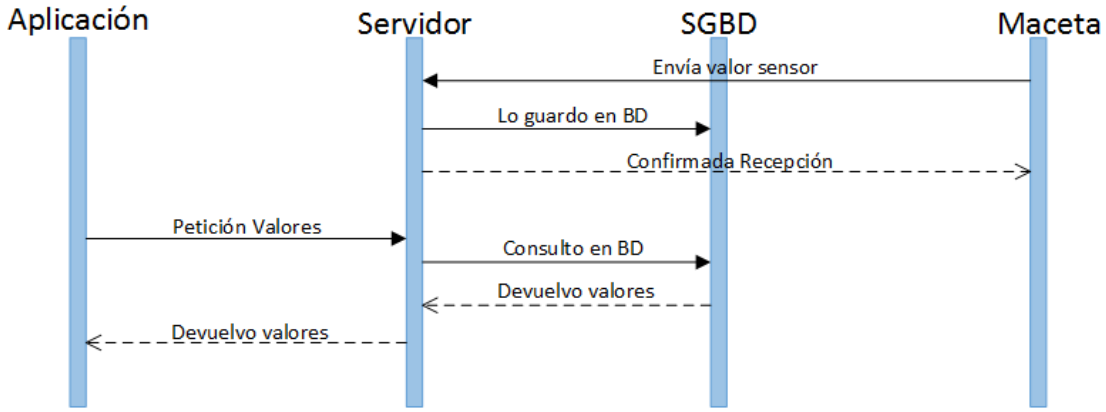


Figura 15 : Diagrama protocolo lectura sensores

Esta secuencia de comunicación es posiblemente la más realizada con la aplicación móvil. Por este motivo se decide desacoplar los robots-maceta. De esta forma la aplicación siempre recibirá contestación de la base de datos aunque no sea el valor más reciente.

Comunicación escribir valor actuador en macetero

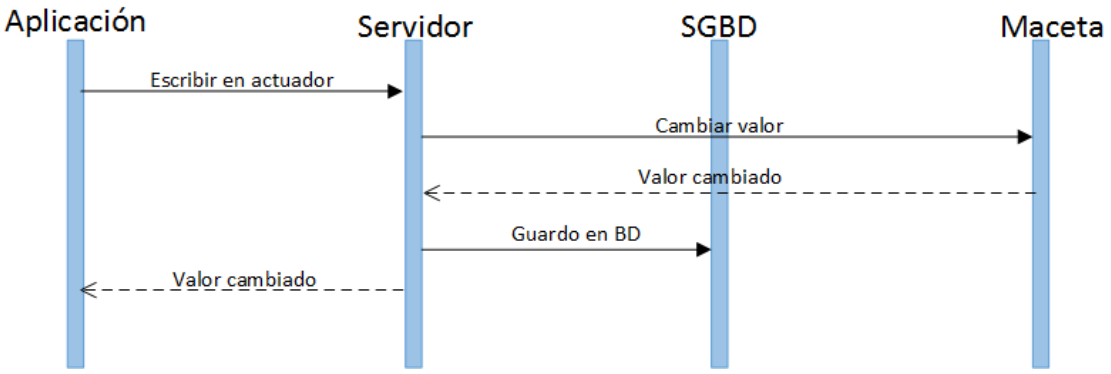


Figura 16 : Diagrama protocolo escritura actuadores

En el caso de enviar el valor de un actuador ya no se puede realizar este desacople. Es por ello que se debe comunicar al robot-maceta la solicitud del cambio y esperar la respuesta, el cual debe estar en modo servidor.

Comunicación de alarmas al servidor

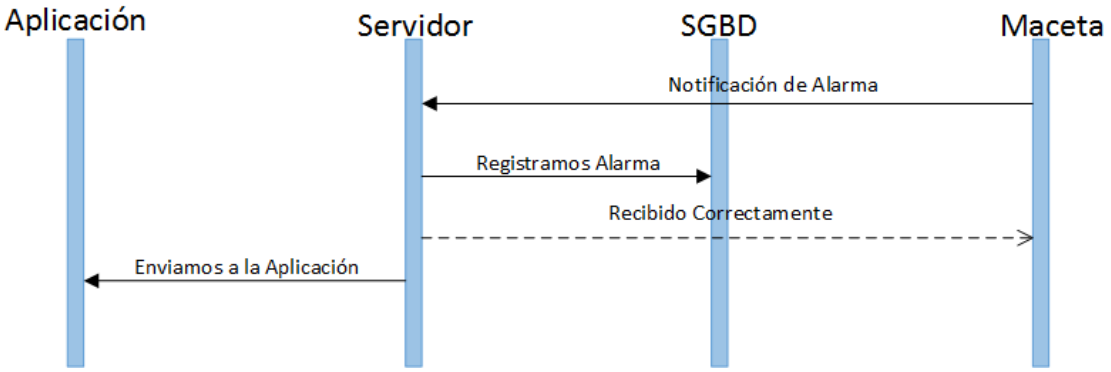


Figura 17 : Diagrama protocolo alarmas o cambios de estado

El último protocolo de comunicación se utiliza para enviar un cambio de estado o alarma en el robot-maceta y registrarlo en la base de datos. Más tarde se comunica a la aplicación de dicho cambio.

4.5 Aplicación Móvil

Una vez llegados aquí solo queda diseñar cómo va a ser la aplicación para realizar las pruebas de las comunicaciones. Básicamente debe mostrar los sensores y cambiar el valor de un actuador en el prototipo de robot-maceta construido.

La aplicación de prueba dispone de un menú principal donde poder elegir la opción a realizar.

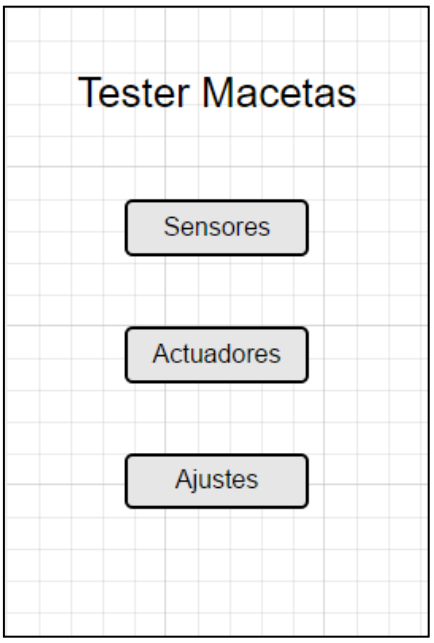


Figura 18 : Menú Aplicación Móvil

Consta de tres opciones. La primera es *Sensores* en la cual mostrara los robots activos para poder obtener todos los sensores.

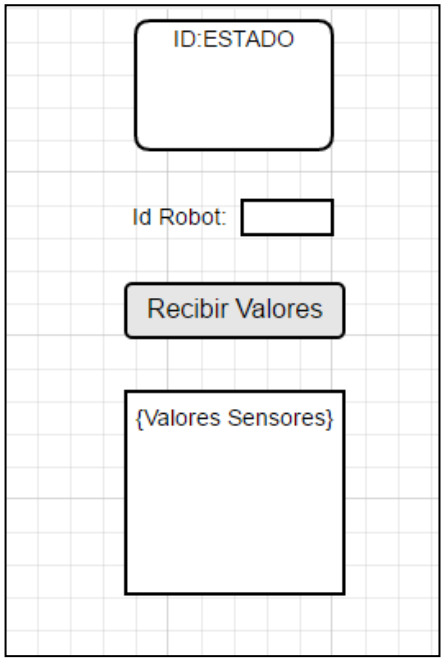


Figura 19 : Opción Sensores App

La segunda opción es *Actuadores.* En esta es donde se puede cambiar el valor a un actuador de un robot determinado.

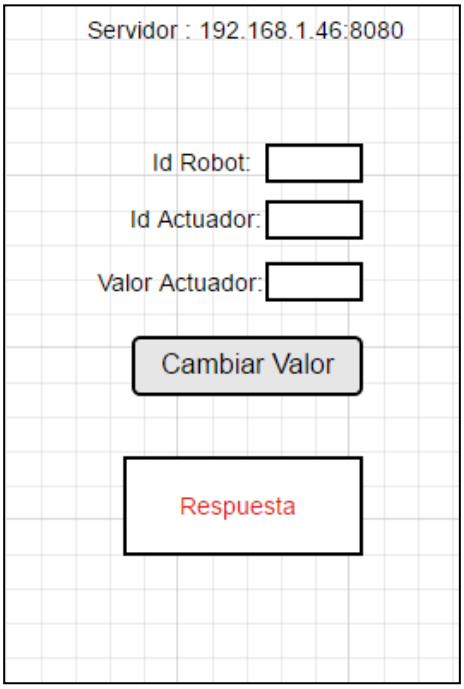


Figura 20 : Opción Actuadores App

Por último queda la opción de *Ajustes*.

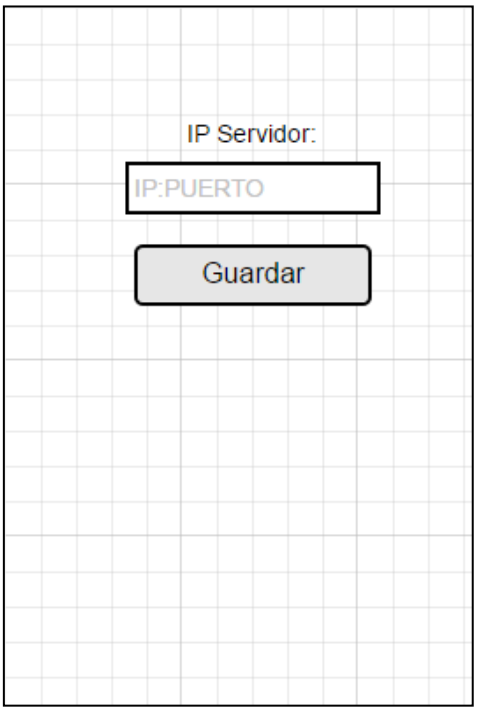


Figura 21 : Opción Ajustes App

Esta sección es muy útil cuando se requiera algún tipo de funcionalidad adicional. En especial servirá para poder cambiar la dirección IP del servidor para realizar las solicitudes. Esto es muy conveniente ya que si no estuviera y cambiara la IP del servidor principal, se tendrá que volver a compilar todo el

proyecto de la aplicación. De esta forma se crea una variable general donde se encuentre alojada la dirección del servidor actual.

Para darle un aspecto más atractivo se ha elegido una imagen lo más apropiada para este proyecto.

Figura 22 : Icono App

Esta imagen se usa como icono de la aplicación móvil y fondo para las diferentes secciones que la componen.

4.6 Herramientas empleadas

Servidor

Para llevar a cabo este proyecto se utiliza la herramienta de programación Eclipse. El motivo principal de esta elección es que permite dentro del mismo entorno arrancar un servidor para poder probar nuestras implementaciones. El servidor incluido es de tipo Tomcat. Se trata de un servidor web sencillo ya que ocupa muy poco espacio y su utilización dentro del entorno de eclipse resulta intuitiva. De no ser así cada vez que se modificara alguna implementación de del programa se debería exportar todo el proyecto e irse a un servidor externo para probarlo. Resultaría muy costoso en tiempo por muy pequeña que fuese nuestra modificación.

Figura 23 : Entorno de programación eclipse.

Se ha utilizado la versión eclipse Juno. Otro motivo por el que se elige eclipse es porque se usa el lenguaje de programación de Java y este entorno está pensado principalmente para este lenguaje.

Figura 24 : Lenguaje de programación Java.

También se ha necesitado un pequeño portal para gestionar la base de datos en MySQL. Se trata de phpmyadmin [7]. Este portal permite manejar tanto la estructura de la base de datos como los datos que la contienen. Resulta cómodo cuando se quiera editar algún campo u otro tipo de modificación.

Figura 25 : Aplicación phpmyadmin

Robot-Maceta

La implementación de este prototipo robot-maceta como bien se ha comentado con anterioridad, se va a llevar a cabo con Arduino. Estas placas tienen un software específico y gratuito que podemos bajar de la página oficial[5]. Lo que permite es programar con relativa facilidad con un lenguaje parecido a Java. Básicamente este va ser el principal software para la implementación del robot-maceta.

[5] Página oficial : https://www.arduino.cc/en/Main/Software

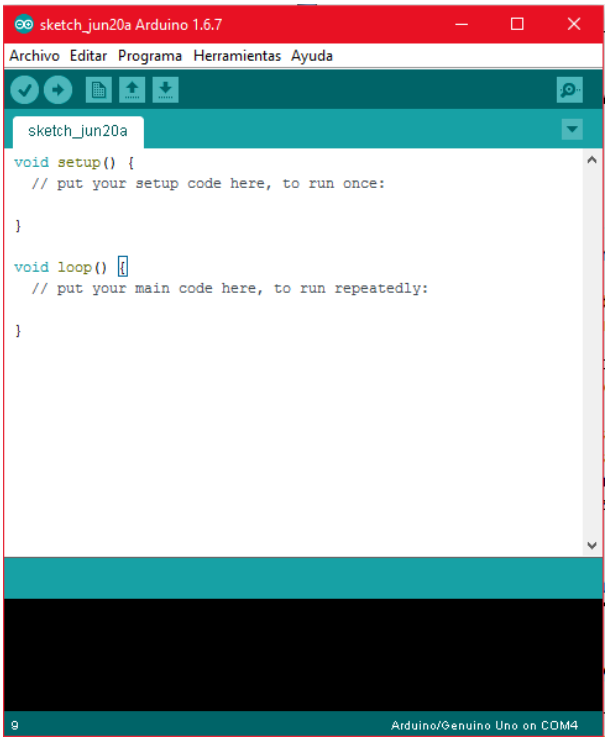


Figura 26 : Entorno desarrollo Arduino

También existen librerías específicas para utilizar otros dispositivos relacionados con Arduino. En este caso el dispositivo ESP8266 dispone de una librería para gestionar las conexiones fácilmente y más rápido, aunque su instalación resulta muy complicada. Cabe recordar que el robot-maceta únicamente se ha creado para realizar pruebas y comprobar que nuestro protocolo de comunicación funciona correctamente, por tanto no se va a hacer mucho hincapié en su implementación ni en la utilización de la librería.

Aplicación Móvil

Finalmente, para desarrollar la aplicación móvil que ayude a probar y testear el sistema de comunicaciones se utiliza el *App Inventor*[6] [8] [10].Es un entorno para crear aplicaciones en línea sin necesidad de descargarse ningún tipo de software y totalmente gratis.

Figura 27 : Logo App Inventor

[6] Página web : http://ai2.appinventor.mit.edu/

La forma de implementar el código es muy novedosa porque utiliza bloques que pueden ir añadiéndose para formar la aplicación. De esta forma se realiza visualmente la implementación. A continuación aparece un ejemplo de código en bloques.

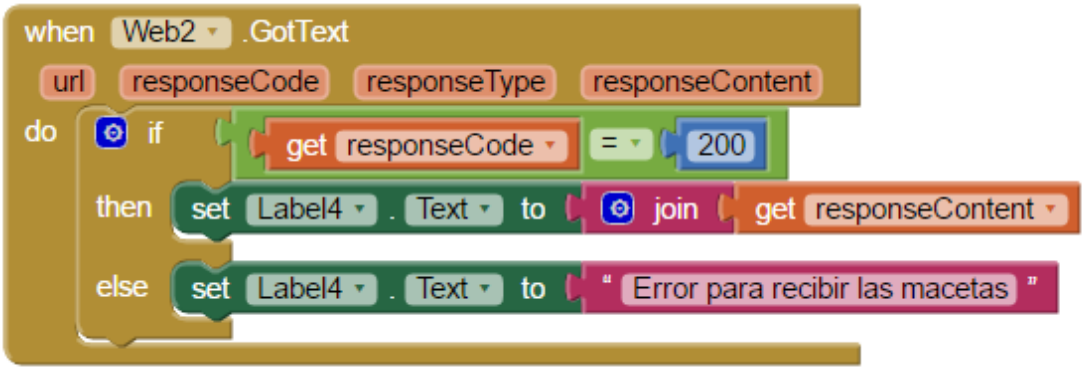


Figura 28 : Programación bloques App Inventor

A la hora de diseñar la interfaz es prácticamente igual a cualquier otra herramienta de desarrollo que exista en el mercado.

Otra elemento muy interesante de este entorno es cuando se quiere instalar la aplicación en un terminal móvil. La herramienta dispone de un código QR que se escanea con un teléfono y que permite descargar dicha aplicación para más tarde instalarla y probarla. Así se evita tener que conectar el teléfono al ordenador mediante un cable y de esta forma perder más tiempo.

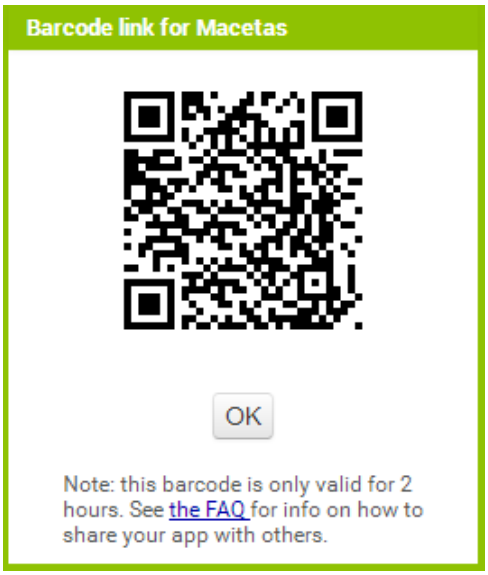


Figura 29 : Código QR para descargar nuestra App

4.7 Conclusiones

En este capítulo se ha desarrollado todo el diseño del proyecto. En primer lugar se han realizado los protocolos generales del sistema de comunicación. Más tarde se ha diseñado las partes de software servidor y de la aplicación. Finalmente el hardware utilizado, además de las herramientas adicionales.

El próximo apartado se empieza a implementar y montar el proyecto completo. Se programara cada uno de los elementos para que al final se puedan realizar las pruebas pertinentes de comprobación.

CAPÍTULO 5

Implementación, Implantación y Evaluación

En este capítulo se empieza a desarrollar el proyecto teniendo en cuenta todos los aspectos de diseño descritos antes. Una vez realizada la implementación el siguiente paso es implantarlo para más tarde poder realizar las pruebas pertinentes y así comprobar que cumple con la funcionalidad especificada.

Es importante destacar que como se trata de un sistema de comunicaciones es necesario realizar diferentes pruebas para poder conseguir robustez. Para ello como se ha comentado en el capítulo anterior la implementación de un programa básico en Arduino y una aplicación móvil con las características necesarias y básicas para poder probar el sistema en un entorno lo más real posible. Pero antes se ha de realizar la implementación de los protocolos de comunicación con REST.

5.1 Implementación

Protocolos y servicio REST

Esta es la base principal de toda la arquitectura de comunicaciones ya que toda petición tanto de la aplicación como de los robots pasan por el servidor para ser gestionadas.

El tipo de servidor que se va a utilizar es uno de tipo REST. Se trata de un protocolo que utiliza las URLs[7] para realizar peticiones enviando en la misma dirección parte de los datos. Por ejemplo cuando se quiera recibir todos los sensores de un robot en la aplicación el robot-maceta deberá usar la siguiente Url:

```
http//:192.168.1.1:8080/rest/sensores/{identificador_robot}
```

[7] Definición: Es una secuencia de caracteres que se utiliza para nombrar y localizar recursos, documentos e imágenes en la red.

La finalidad del servicio es partir la Url y dependiendo de qué palabra contenga devolverá una información u otra. En este caso se ha de buscar los sensores que se tengan guardados del identificador que se ha pasado en la Url y devolver una lista de sus sensores. Esta es una de las ventajas que tiene REST, permite que la misma dirección contenga información. Este protocolo es muy utilizado para redes de sensores por este motivo.

Otra característica, es la posibilidad de elección en el formato de los datos que retornan una vez procesada la petición. Aquí se usan dos tipos de formatos. El primero es en texto plano, es interesante para peticiones relacionadas con los robots-maceta, ya que únicamente se recibe un valor o a lo sumo dos. El otro formato que devuelve el servidor hacia la aplicación es tipo JSON. Se trata de una notación muy utilizada para intercambiar datos. Permite parsear [8] fácilmente los datos para procesarlos más rápidamente en la parte del cliente.

La aplicación está dividida en dos paquetes. El primero se llama *Persistencia*, dentro contendrá todos los métodos necesarios para las peticiones que necesiten hacer uso de la base de datos. También contiene los métodos para realizar las conexiones y desconexiones. El segundo es un paquete que se llama *servicios*. Dentro de este paquete existen tres clases: *PeticionesAPP*, *PeticionesRobot* y *ItemsApp*. La primera de las clases contiene los métodos a los que la aplicación puede acceder y de donde retorna los datos que se soliciten. La clase de *PeticionesRobot* básicamente tiene la misma función que la primera pero en este caso contiene métodos que pueda acceder el robot-maceta para pedir información al servidor. La última clase de lo único que se encarga es de registrar dichos servicios en el servidor.

Figura 30 : Distribución Archivos Servidor

[8] Definición: Proceso de analizar una secuencia de símbolos a fin de determinar su estructura.

También hay que tener en cuenta que se debe instalar una serie de librerías que hacen que el servicio REST funcione.

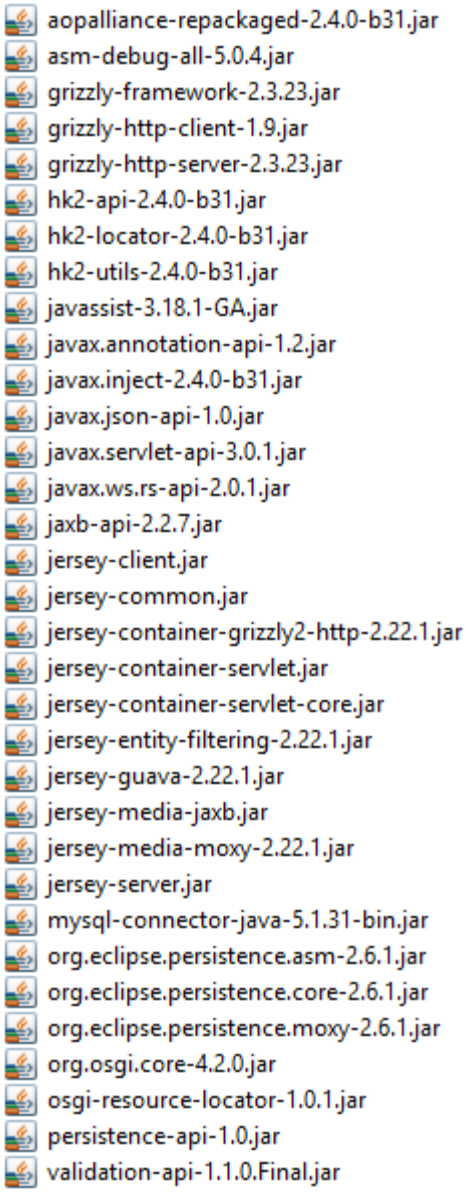


Figura 31 : Librerías para REST

Para poner el servidor REST en funcionamiento se debe configurar un archivo con nombre web.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
id="WebApp_ID" version="3.0">
  <display-name>rest</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <description>JAX-RS Tools Generated - Do not modify</description>
    <servlet-name>JAX-RS Servlet</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>javax.ws.rs.Application</param-name>
      <param-value>services.ItemsApp</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>JAX-RS Servlet</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Código 1 : Archivo web.xml

Una vez todo está configurado y arrancado el servidor Tomcat , se procede a programar los requisitos que exige la aplicación. El primero método es el que devuelve todos los valores de los sensores de un determinado robot-maceta.

```
@GET
@Path("sensores/{id_maceta}")
@Produces({MediaType.APPLICATION_JSON})
  public String getSensores(@PathParam("id_maceta") String id_maceta) throws SQLException {
    System.out.println("Se ha solicidado valores de sensores del la maceta " + id_maceta);
    return DatosBBDD.ListaValorSensores(id_maceta);
  }
```

Código 2 : Lista sensores REST

Lo que hace este método es coger el identificador del robot-maceta que se envía en la URL y lo transfiere a un método que busca en la base de datos todos sensores con dicho identificador.

El método que accede a la base de datos se encuentra como se ha dicho antes dentro del paquete de persistencia. Simplemente lanza una sentencia SQL[9] y retorna al método anterior los datos ya formateados en JSON para enviárselos a quien los haya solicitado.

[9] Definición: Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

```
public static String ListaValorSensores(String id_macetero){
    try {
          java.sql.Connection con = conectar_BBDD();
          String query = "SELECT tipo_sensor.descripcion, sensor.valor " +
                "FROM sensor_macetero INNER JOIN (tipo_sensor " +
                "INNER JOIN sensor ON tipo_sensor.id_tipo = sensor.id_tipo) " +
                "ON sensor_macetero.id_sensor = sensor.id_sensor " +
                "WHERE sensor_macetero.id_macetero =" + id_macetero + ";";

          java.sql.Statement st = con.createStatement();
          ResultSet rs = st.executeQuery(query);
          String res = "";
          while (rs.next()) {
            res += "{" + rs.getString("descripcion");
            res += ":" + rs.getString("valor") + "}\n";
          }
          st.close();
          con.close();
          return res;
    }
    catch (Exception e) { return "Error";}
}
```

Código 3 : Lista sensores desde Base de Datos

A continuación se ve cómo esta implementado el método de escribir valores en el robot-maceta.

El mecanismo es el mismo, es decir, enviar valores por la dirección URL a la hora de escribir. Por tanto se utiliza la siguiente dirección para solicitar a un robot que cambie un valor de su actuador.

http://IP_ROBOT / {id_actuador}/ {valor_actuador}

El robot con dicha IP lo recibe ,parte la dirección para obtener el actuador y que valor debe cambiar. Para ello antes se debe saber qué dirección IP tiene el robot-maceta al que se va a enviar la información. Esto se soluciona con un método que más tarde se describe y lo que hace es que cuando el robot adquiere por primera vez una dirección de red la envía al servidor para que la guarde en la base de datos al lado de su identificador único.

A continuación se muestra un pequeño esquema del proceso completo.

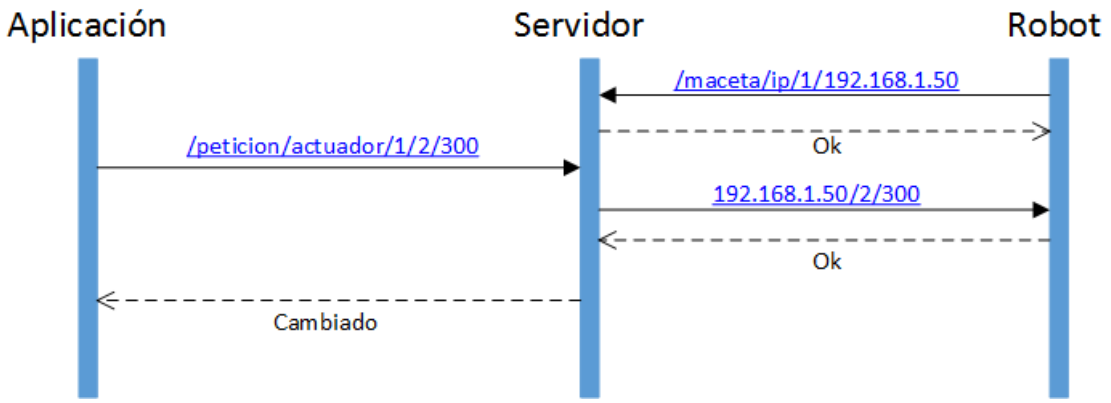


Figura 32 : Secuencia Cambio Valor

Es importante cada vez que se pida o se solicita algo contestar con un mensaje de confirmación o en caso contrario devolver el error que se ha producido.

La implementación para cambiar el valor del actuador es la siguiente:

```
@GET
@Path("actuador/{id_maceta}/{id_actuador}/{valor_actuador}")
@Produces(MediaType.APPLICATION_JSON)
  public String setActuador(@PathParam("id_maceta") String id_maceta,@PathParam("id_actuador") String id_actuador,
                            @PathParam("valor_actuador") String valor_actudor) throws SQLException {

    //Buscar en la base de datos la dirección ip del robot con el Id que nos pasan.
      String IP_maceta = DatosBBDD.DimeIpMaceta(id_maceta);
      if (IP_maceta == ""){ return "Fallo al recuperar la IP"; }
      String GET_URL = "http://" + IP_maceta +"/" + id_actuador + "/" + valor_actudor + "/";
      try {
          URL obj = new URL(GET_URL);
          HttpURLConnection con = (HttpURLConnection) obj.openConnection();

          con.setRequestMethod("GET");
          int responseCode = con.getResponseCode();
          if (responseCode == 200) System.out.println("Respuesta de la maceta " + id_maceta + " confirmada." );
          BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
          String inputLine;
          StringBuffer response = new StringBuffer();
          while ((inputLine = in.readLine()) != null) { response.append(inputLine);}
          in.close();

          //Guardo en la Base de Datos.
          DatosBBDD.GuardarValorActuador(id_maceta,id_actuador,valor_actudor);
          return "OK";

      } catch (IOException e) {
          System.out.println("Respuesta No recibida");
          return "Error al conectar con " + IP_maceta;
      }
  }
```

Código 4 : Cambiar valor actuador

Se puede observar en el código 4, que al principio busca en la base de datos la dirección IP teniendo como criterio el identificador que recibido en la misma petición.

Más tarde se construye la Url con la dirección IP que se ha obtenido de la base de datos, el identificado del actuador del robot-maceta y el valor al que se quiere cambiar. Una vez hecha la Url se envía y permanece a la espera de la respuesta del robot-maceta. En el momento que se recibe la respuesta se

comprueba si ha sido correcta, es decir si es igual a *200 OK,* ya que esto indica que el dispositivo lo ha recibido y más tarde se responde *ok* hacia la aplicación que haya realizado la petición. En caso de fallo el método devuelve error y la dirección IP del robot-maceta del que no se ha podido conectar.

Al final de este método justo antes de devolver la respuesta se hace una llamada a una función del paquete de persistencia que guarda el nuevo valor cambiado en el actuador.

```
public static void GuardarValorActuador(String id_maceta,String id_actuador,String valor_actuador) throws SQLException {
    java.sql.Connection con = conectar_BBDD();
    java.sql.Statement estatuto = con.createStatement();
    estatuto.executeUpdate("UPDATE macetero_actuador INNER JOIN actuador " +
                        "ON macetero_actuador.id_actuador = actuador.id_actuador " +
                        "SET actuador.valor = " + valor_actuador + " WHERE " +
                        "macetero_actuador.id_macetero = " + id_maceta + " AND " +
                        "macetero_actuador.id_actuador =" +id_actuador+ ";");
    estatuto.close();
    con.close();
}
```

Código 5 : Guardar Valor Actuador en Base de datos

Una vez realizado las implementaciones referentes a las peticiones de la aplicación se pasa a las peticiones del robot-maceta. Básicamente su desarrollo es muy parecido a los anteriores. La diferencia es que el robot-maceta tiene otras funcionalidades. Como puede ser la de enviar la IP al servidor en el momento que se conecta a la red. Este método vuelve a recoger los datos necesarios de la Url y procesa la petición. Su procesado es guardar la dirección IP en la base de datos.

```
@GET
@Path("ip/{id_robot}/{ip}")
@Produces(MediaType.TEXT_PLAIN)
public String setIp(@PathParam("id_robot") String id_robot,@PathParam("ip") String ip) throws SQLException {
    DatosBBDD.GuardarIp(id_robot,ip);
    return "HTTP/1.1 200 OK\r\nok\r\n\r\n";
}
```

Código 6 : Recepción IP robot-maceta

Al final de la función se retorna nuevamente la cabecera HTTP que confirma que todo ha sido recibido correctamente. El robot-maceta lo recibe y sabe que la comunicación ha sido un éxito. El resto de las peticiones que hace el robot-maceta muestran un desarrollo semejante.

Por tanto solo queda mencionar algún método auxiliar que se necesita. En concreto el que se utiliza para realizar conexiones con la base de datos en MySQL.

```
public static Connection conectar_BBDD() {
    try {
        java.sql.Connection con ;
        con = DriverManager.getConnection("jdbc:mysql://192.168.1.46:3306/macetas","macetas","macetaspower");
        return con;
    } catch (SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage());
        return null;
    }
}
```

Código 7 : Conexión base de datos MySQL

Esta función es llamada desde cualquier otro método que necesite realizar alguna gestión con la base de datos. Cabe destacar que es importante cerrar la conexión creada al acabar sino conducirá a un error.

En la mayoría de métodos se ha utilizado una implementación de seguridad para en caso de error poder gestionarlos correctamente. Esto es muy importante en todo tipo de implementaciones y en concreto en sistemas que impliquen comunicaciones como este.

Las tablas que se muestran a continuación listan todas las URLs y los datos que devuelven al solicitarlo desde la aplicación o desde el robot-maceta.

Aplicación:

Función:	Pedir lista Sensores
URL:	http://IP_SERV:PUERTO_SERV/rest/peticion/sensores/{id_maceta}
Descripción:	Devuelve una lista en formato JSON del nombre de los sensores y su respectivo valor del id_maceta que se ha solicitado.
Respuesta:	{"temperatura_aire":29} {"temperatura_suelo":20} {"humedad_suelo":20} {"luz_1":60} {"luz_2":75} {"Ultrasonido_1":36} {"Ultrasonido_2":25} {"NivelAguaSup":100} {"NivelAguaInf":0}
Respuesta Error:	Error

Tabla 19 : URL Lista Sensores

Función:	Pedir lista Robot-Macetas
URL:	http://IP_SERV:PUERTO_SERV/rest/peticion/macetas
Descripción:	Devuelve una lista JSON con las macetas disponibles en el sistema y el estado en el que se encuentren.
Respuesta:	{1:Correcto} {2:Correcto}
Respuesta Error:	Error

Tabla 20 : URL Lista de robot-macetas

Función:	Enviar Valor Actuador
URL:	http://IP_SERV:PUERTO_SERV/rest/peticion/actuador/{id_maceta}/{id_actuador}/{valor_actuador}
Descripción:	Envía el valor del actuador al macetero que se le indique y devolverá un *OK* que indicara que el servidor tiene constancia de que la maceta ha recibido este valor.
Respuesta :	Ok
Respuesta Error:	Error al conectar con 192.168.1.15

Tabla 21 : URL Enviar valor actuador

Función:	Recibir Historial de un actuador
URL:	http://IP_SERV:PUERTO_SERV/rest/peticion/historial/{id_maceta}/actuador/{id_actuador}/
Descripción:	Se envía a la aplicación una lista en formato JSON con la fecha y hora en la que se tomó el valor aparte del propio valor del actuador.
Respuesta :	{"2016-05-11 20:19:39.0":0, "2016-05-13 03:23:48.0":100}
Respuesta Error:	Error

Tabla 22 : URL Historial de un actuador

Función:	Recibir Historial de un sensor
URL:	http://IP_SERV:PUERTO_SERV/rest/peticion/historial/{id_maceta}/sensor/{id_sensor}/
Descripción:	Se envía a la aplicación una lista en formato JSON con la fecha y hora en la que se tomó el valor aparte del propio valor del sensor.
Respuesta :	{"2016-05-13 07:18:36.0":20, "2016-05-13 12:33:18.0":31, "2016-06-14 18:57:01.0":300, "2016-06-14 18:57:24.0":300, "2016-06-14 19:13:15.0":300, "2016-06-14 19:21:20.0":323, "2016-06-14 19:25:31.0":321, "2016-06-14 19:26:02.0":294, "2016-06-14 19:27:24.0":327}
Respuesta Error:	Error

Tabla 23 : URL Historial de un sensor

Robot:

Función:	Enviar dirección IP
URL:	http://IP_SERV:PUERTO_SERV/ rest/maceta/ip/{id_maceta}/{ip_maceta}
Respuesta:	El robot-maceta después de saber su dirección IP enviara su identificador y la dirección IP al servidor. El servidor responderá con un *OK*.
Respuesta:	Ok
Respuesta Error :	Error

Tabla 24 : URL Enviar dirección IP

Función:	Enviar valor sensor
URL:	http://IP_SERV:PUERTO_SERV/rest/maceta/sensor/{id_maceta}/{id_sensor}/{valor_sensor}
Descripción:	La maceta envía el valor de un sensor al servidor con su identificador para que el servidor sepa quien ha sido. Responde con un OK.
Respuesta:	Ok
Respuesta Error:	Error

Tabla 25 : URL Enviar valor sensor

Función:	Enviar Alarma o estado
URL:	http://IP_SERV:PUERTO_SERV/rest/maceta/sensor/{id_maceta}/{estado}
Descripción:	Envía un cambio de estado del robot-maceta que el servidor podrá interpretar como alarma o simplemente cambio de estado.
Respuesta:	`Ok`
Respuesta Error:	`Error`

Tabla 26 : URL Enviar Alarma o estado

Base de datos

Para implementar la base de datos se hace uso únicamente de la herramienta de phpmyadmin. Esta permite crear la base de datos y las tablas que la constituyen. Pero también existen comandos SQL para crear la estructura e insertar datos desde una consola o intérprete de comandos. A continuación vemos los dos posibles métodos.

```
MariaDB [macetas]> CREATE TABLE sensor (
    -> id_sensor int,
    -> id_tipo int,
    -> valor varchar (10)
    -> );
```

Figura 33 : Crear tabla modo consola

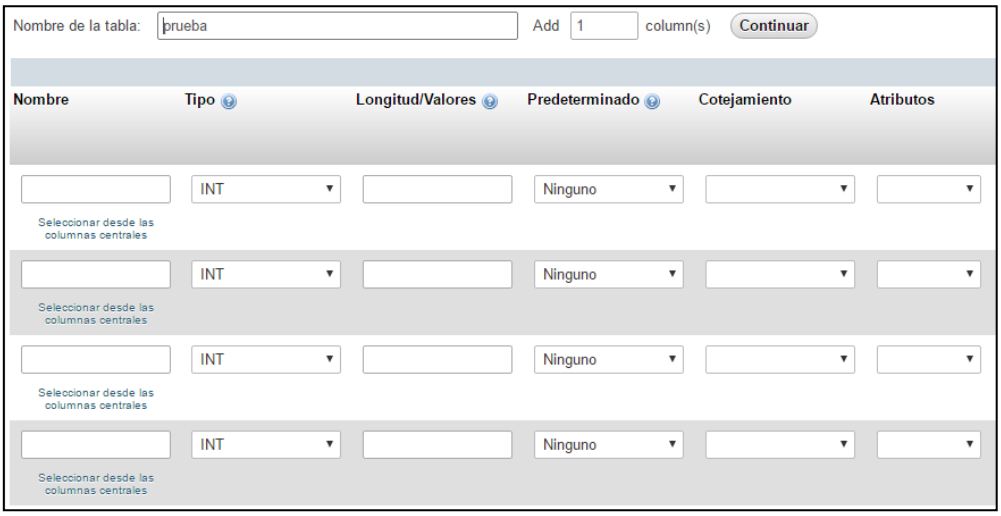


Figura 34 : Crear tabla phpmyadmin

Para este proyecto se utiliza el modo de inserción con phpmyadmin ya que resulta menos costoso.

Por último este portal también permite agregar triggers de manera muy cómoda y guiada. Existe un apartado llamado disparadores en cada tabla. Es aquí donde se implementa el trigger para que se guarde automáticamente un valor en el historial de sensor o de actuador.

```
CREATE TRIGGER `InsertarEnHistorial` AFTER UPDATE ON `sensor`
 FOR EACH ROW INSERT INTO historial_sensor (fecha,id_sensor,valor)
VALUES (CURRENT_TIMESTAMP,old.id_sensor,old.valor)
```

Figura 35 : Trigger sensor

Esta pequeña implementación se lanzara cada vez que se actualice el valor de un sensor en la tabla de *sensor* e inserta en *historial_sensor* el antiguo valor antes de que se cambie. Este valor se encuentra en *old.valor*. De esta forma se consigue rellenar la tabla historial.

Robot-maceta

Una vez implementados todos los servicios por parte del servidor se deberá empezar a probar nuestras comunicaciones. También se implementa un pequeño prototipo de robot-maceta con funciones básicas para poder testear todo el sistema de comunicaciones.

La funcionalidad del prototipo robot-maceta es muy sencilla. Lo único que permite es encender un LED cuando se recibe del servidor la petición de cambiar el valor de un actuador y enviar seguidamente el valor de un sensor de luz. Todo esto se realiza con la placa de Arduino y con su software específico.

Parece relativamente sencillo, sin embargo se debe diseñar un pequeño protocolo dentro del robot-maceta que este a la espera y cuando reciba una petición de cambiar actuador procese dicha petición cambie el LED.

Primero se configura el dispositivo de comunicaciones en el robot. Como bien se ha explicado antes se trata del dispositivo ESP8266 [11] que se comunica por Wifi. Para poder conectarse hay que enviarle una serie de comandos AT[10] indicando el nombre de la red Wifi y la contraseña. También hay que configurar una serie de parámetros.

Todo esto lo consigue la siguiente implementación:

[10] Definición: Son cadenas de caracteres que se convirtieron en estándar abierto de comandos para configurar y parametrizar módems.

```
String ordenes[]=
    {   "AT+CIOBAUD=9600",
        "AT+CWMODE=3",
        "AT+CIPMUX=1",
        "AT+CIPSERVER=1,80",
        "AT+CWJAP=\"" +(String)AP + "\",\"" + (String)AP_CONTRASENYA + "\"",
        "AT+CIFSR",
        "AT+CIPSTATUS",
        "END"         // Para reconocer el fin de los comandos AT
    };
```

Código 8 : Comandos AT ESP8266

```
int index = 0;
while(ordenes[index] != "END")
    {   SerialEsp8266.println(ordenes[index++]);
        while ( true)
          {    String s = GetLineWIFI();
               if ( s!= "") Serial.println(s);
               if ( s.startsWith("no change"))
                        break;
               if ( s.startsWith("OK"))
                        break;
               if ( s.startsWith("ready"))
                        break;
          }
       Serial.println(".....................");
  }
```

Código 9 : Procesado Comandos AT

La función que realiza la implementación anterior es enviar cada uno de los comando AT para configurar el dispositivo inalámbrico.

Cuando la placa Arduino esté lista para recibir peticiones se enciende un LED verde que indica que está preparado. En ese momento el programa estará en un bucle esperando. Cuando reciba una petición se saldrá momentáneamente del bucle para ir a una función que se encargara de procesar la petición (Código 10).

```
void ProcesarPeticion(String cadenaPeticiion)  {
    String tipo,valor = "";
    tipo = getValue(cadena,'/',0);
    valor =  getValue(cadena,'/',1);
    Serial.println("Id Actuador: " +tipo+ "\nValor: " + valor );

    if (valor.toInt() > 100) {
        digitalWrite(10, HIGH);
    }
    else {
      digitalWrite(10, LOW);
    }
    //Envio Respuesta al Servidor
    http("HTTP/1.1 200 OK\r\noki \r\n\r\n","0");
    delay(100);
    SerialEsp8266.println("AT+CIPCLOSE=0");
}
```

Código 10 : Función procesar petición

En la función de *ProcesarPeticion* del código 10, se hace uso de otra función llamada *getValue* (). Esta función divide la respuesta y se queda únicamente con los datos que interesan. Lo primero que se necesita es el identificador del actuador que se quiere cambiar y seguidamente el valor. Una vez procesada la petición se debe de encender el led. Pero solo se encenderá cuando el valor sea mayor a cien, en caso contrario se apagara dicho led.

Más tarde queda enviar la confirmación al servidor para indicar que todo ha ido correctamente. Y por último cerrar la conexión con el comando *AT+CIPCLOSE*.

También se ha añadido un led rojo que se encenderá cuando se haya recibido una petición al dispositivo y se esté procesando. Una vez procesado se apagara el led rojo.

Para acabar queda enviar el valor de un sensor al servidor cuando haya cambiado. En este caso, como es para probar el sistema de comunicaciones se realiza una vez finalizado el procesamiento de la petición de cambio de valor.

```
void EnviarSensor (int id_sensor_arduino,int id_sensor_servidor) {
    String cmd = "AT+CIPSTART=1,\"TCP\",\"" + (String)IP_SERVIDOR +"\"," + PUERTO_SERVIDOR;
    String mensaje = "GET /rest/maceta/sensor/" + (String)ID_MACETA + "/" + id_sensor_servidor + "/"+analogRead(id_sensor_arduino)+";
    do {
      Serial.print("Enviando Valor Sensor");
      SerialEsp8266.println(cmd);
      delay(100);
      while (!SerialEsp8266.available()) {
        delay(100);
    };
    if (SerialEsp8266.find("OK")) {
      http(mensaje, "1");
      Serial.println("........Valor Sensor enviado.");
       SerialEsp8266.println("AT+CIPCLOSE=1");
        break;
    } else {
      Serial.println("... ERROR");
      SerialEsp8266.println("AT+CIPCLOSE=1");
    };
  } while (true);
}
```

Código 11 : Enviar el valor de un sensor

Esta implementación abrirá una conexión con el servidor y más tarde envía a una Url en forma de GET. En esta dirección Url se envían los datos necesarios para que el servidor pueda gestionar y guardar bien el valor enviado.

También se controla que no haya errores durante la transmisión y se cierra la conexión una vez acabado.

Básicamente con estas implementaciones en Arduino ya se pueden realizar todas las pruebas que se necesitan para cumplir con los requisitos.

Aplicación móvil

Lo último que queda por desarrollar es una aplicación para un dispositivo móvil. También se va a utilizar el robot-maceta para testear el sistema de comunicaciones. Además, se pueden hacer solicitudes al servidor REST utilizando un simple navegador web. Pero queda más atractivo el diseño de una sencilla APP para dicho propósito.

Se realiza una simple implementación con un programa llamado APP Inventor. Este programa, las características del cual ya se han descrito previamente, va a permitir un diseño ágil a la vez que rápido. No se requiere un diseño muy avanzado de la APP ya que otra componente del proyecto se encarga de esta parte. Únicamente se confeccionan unas pocas implementaciones que posibilitan la comunicación entre el servidor y el robot-maceta.

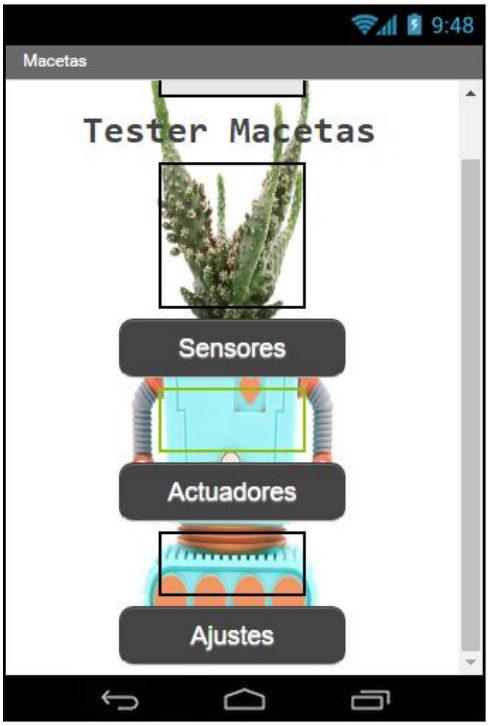


Figura 36 : Menú principal App

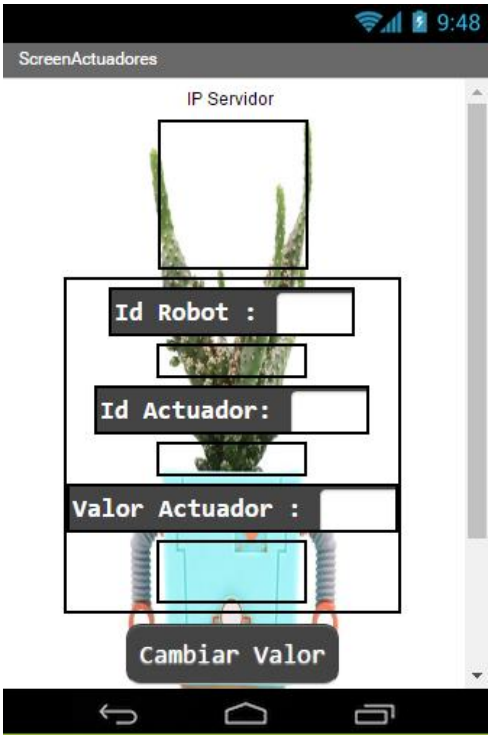


Figura 38 : Sección cambiar actuador

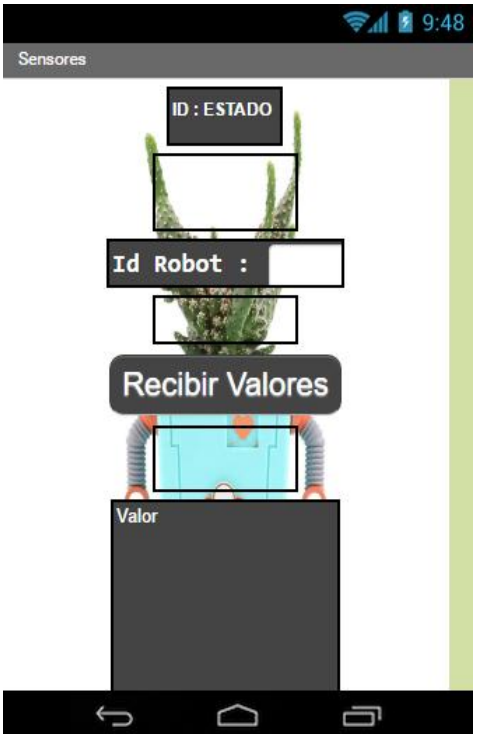
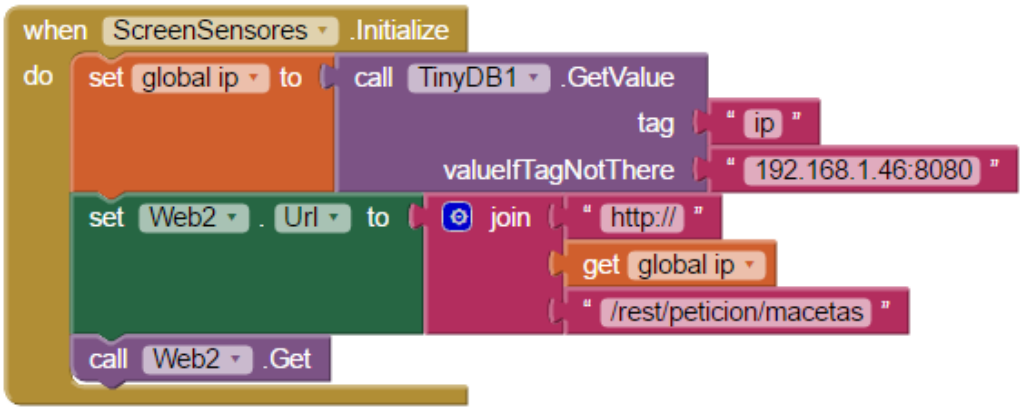


Figura 37 : Sección leer sensores

Figura 39 : Sección Ajustes

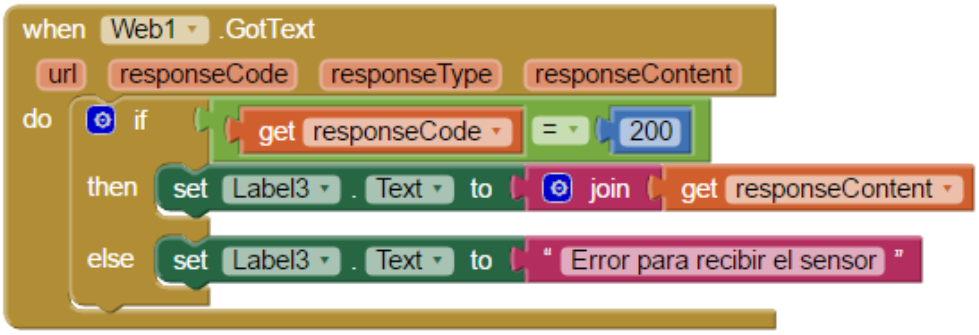
Estos son los diferentes diseños disponibles para las secciones que componen la aplicación. Los rectángulos transparentes únicamente se utilizan para centrar los elementos dentro de una misma pantalla. Más tarde, en el apartado de implantación, se puede ver cómo queda el diseño en un dispositivo real.

Solamente resta ver como se ha implementado el código para su correcto funcionamiento. La finalidad principal de la APP es enviar peticiones GET y recibir valores. Esto se puede contemplar en el código 12.



Código 12 : Solicitud GET

La implementación de código en esta plataforma de desarrollo se realiza por bloques. Lo único que se hace es construir una petición con la dirección IP del servidor, que se encuentra en una variable global y después se añaden el resto de los parámetros. En el momento que se reciba el valor o valores que se hayan solicitado, se ejecutará automáticamente el siguiente fragmento de código.



Código 13 : Recepción de Valores

Esto comprueba si la recepción ha sido correcta, de ser así, mostrara los valores devueltos por el servidor, en caso contrario se mostrara el mensaje de error: *Error para recibir el sensor*. A falta de unos detalles, esta es la base de la implementación de la APP.

5.2 Implantación

El siguiente paso que se debe hacer es integrar todo en un entorno real. Para ello, una vez acaba la implementación de los servicios del servidor se procede a la extracción y colocación total en el sistema final. Tratándose de un servidor Tomcat, el traslado resulta muy fácil, solo se tiene que exportar a un archivo de tipo *war*.

Una vez creado el archivo, existen dos opciones para instalarlo en el servidor. La primera opción para alojar el archivo, es desde el portal de *managed* de Tomcat, al cual se puede acceder con el navegador y la dirección IP del servidor. Desde ahí es posible instalarlo y desinstalarlo en cualquier momento. La segunda opción es copiar el archivo *war* en la carpeta de configuración de Tomcat que se denomina *webapp*. En el arranque, el servidor se encarga de extraer el paquete e instalarlo.

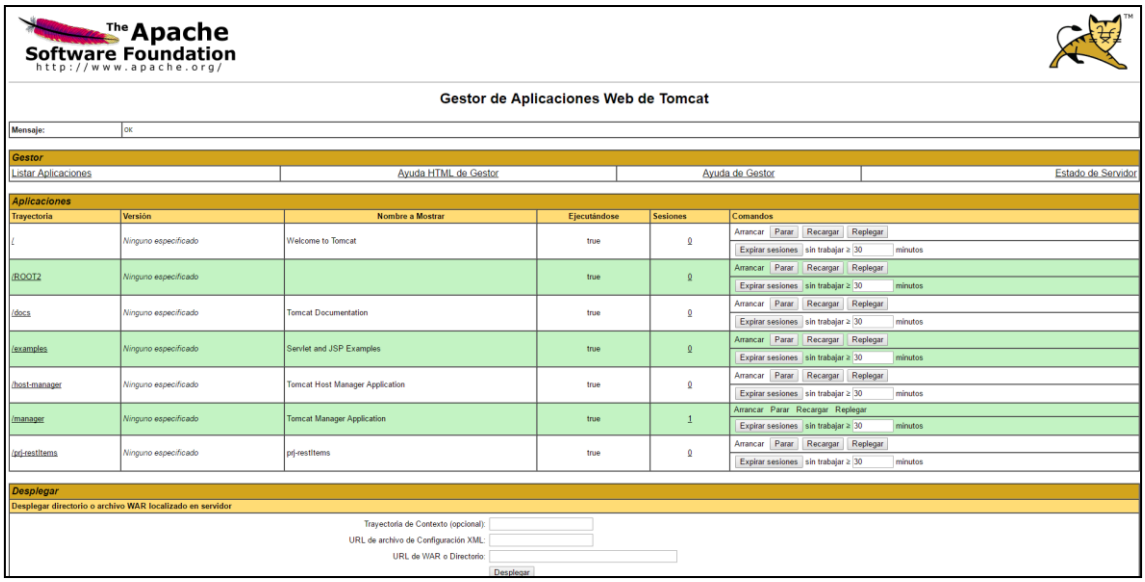


Figura 40 : Portal gestor de aplicaciones en Tomcat

Una vez configurado el servidor, debe albergarse la base de datos de MySQL en el servidor. Es posible utilizar el mismo servidor u otro externo al principal. Gracias al portal de phpmyadmin resultará muy hacedero. Desde ahí se puede montar toda la estructura de tablas, al igual que exportarla para realizar copias a otro servidor o simplemente como copias de seguridad. Es importante comprobar que se puedan efectuar conexiones con la base de datos. Para ello es necesaria la creación de un usuario que tenga permiso para acceder a dicha base de datos.

Una vez que se ha conseguido montar todo lo referente al servidor se puede hacer alguna prueba con un navegador básico, aunque más tarde se realizaran varios test con la aplicación móvil.

Ahora se pasa a montar el prototipo robot-maceta. La figura 41 muestra el esquema final que va a tener el prototipo. Los leds verde y rojo exclusivamente indican cuando el dispositivo está listo para recibir peticiones o cuando se está procesando una petición entrante. Por último, el led blanco o transparente es el que realiza la función de actuador para encenderse desde la aplicación móvil.

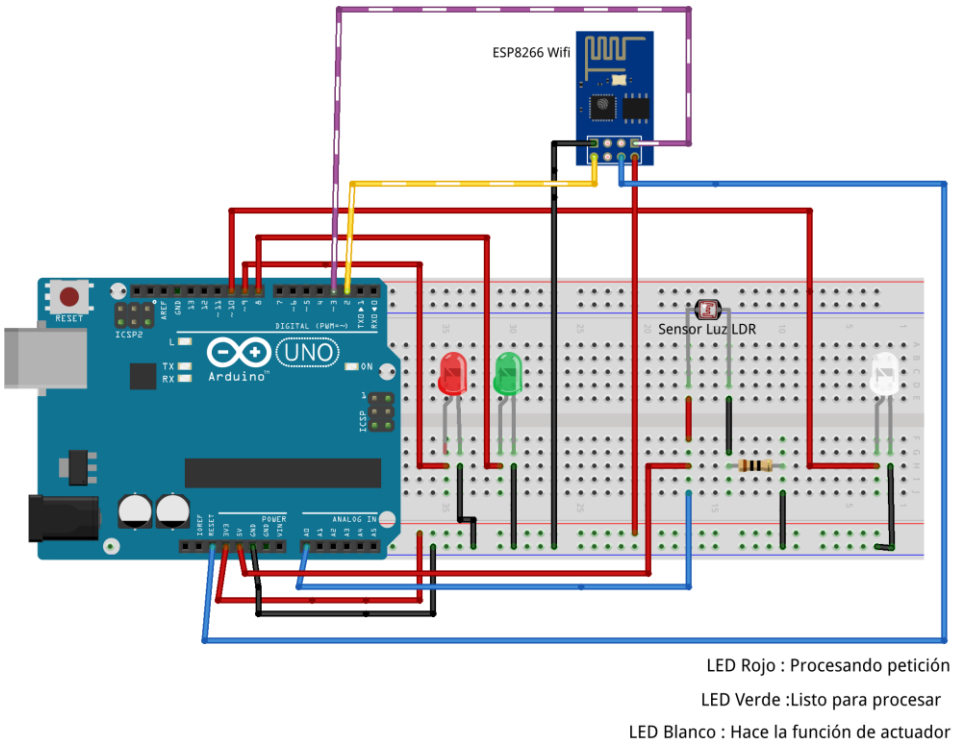


Figura 41 : Esquema del Circuito Robot-maceta

Este es el resultado del circuito ya construido.

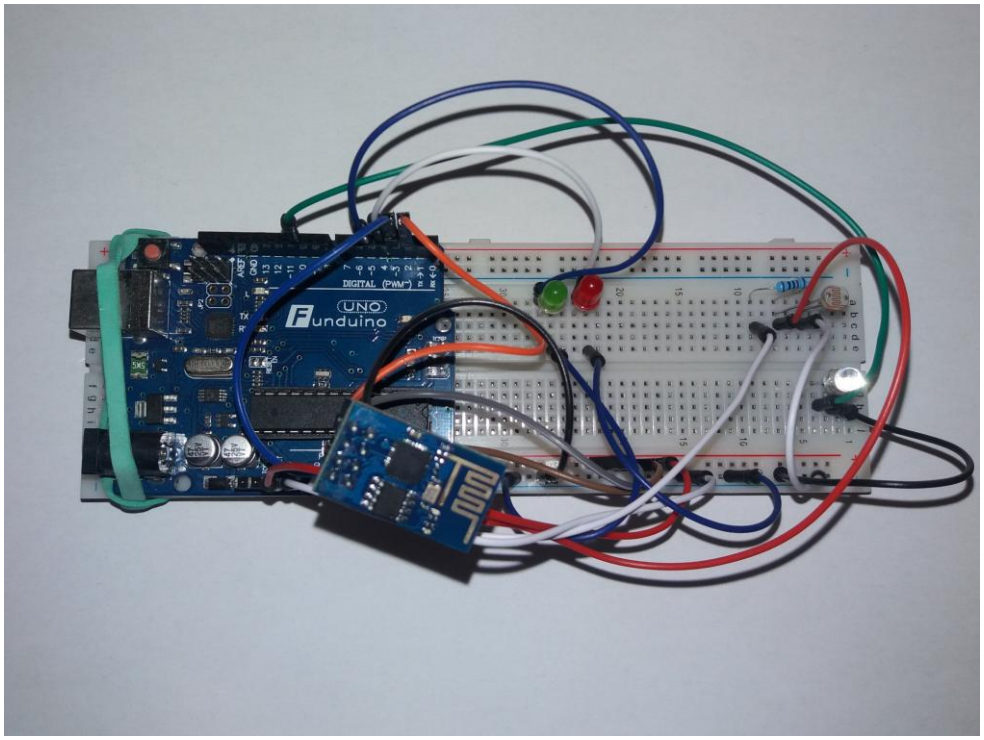


Figura 42 : Prototipo robot-maceta

Ya solo queda compilar el programa con el entorno de desarrollo de Arduino y subirlo a la placa, utilizando para ello un cable USB.

Lista esta parte de la implantación, queda instalar la aplicación en un dispositivo móvil para probar el sistema completo. Resulta muy sencillo compilar la aplicación móvil desde el programa de APP Inventor. Meramente es necesario adquirir una herramienta en el dispositivo móvil que permita escanear códigos QR. Una vez se disponga de esta herramienta es primordial compilar la aplicación para que a continuación aparezca el código QR. Posteriormente al escaneo, comenzará la descarga de un paquete con extensión *apk*. Una vez descargado se procede a su instalación.

Hay que tener en cuenta que es imprescindible tener habilitado en el teléfono la instalación de APPs para aplicaciones de origen desconocido. Esto se puede configurar desde los ajustes del dispositivo móvil.

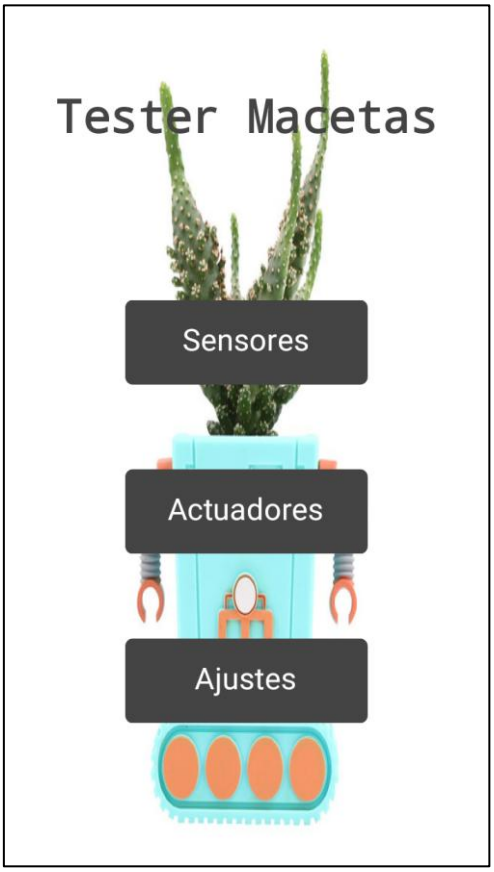


Figura 43 : Captura aplicación móvil final

La figura 43 muestra una captura del teléfono con la aplicación creada y ya en funcionamiento. De modo que ya se puede proceder a probar todo el sistema de comunicaciones completo.

5.3 Evaluación

Llegados a este apartado, queda verificar que se cumplen todas las funcionalidades que se especificaron en el capítulo dos. Por ello se va a empezar comprobando el envió de peticiones desde la aplicación móvil al servidor y su procesado en la parte del servidor.

En el instante en el que se accede a la sección *Sensores* de la aplicación móvil, se ejecuta una petición GET que recibe los identificadores de los robots-maceta y los estados que están en ese momento registrados en el sistema. Una vez dentro de esta sección, se elige un identificador de un robot y se pulsa sobre *Recibir sensores*. De esta forma se comprueban dos requisitos de la aplicación que son resumidos al final del apartado en una tabla.

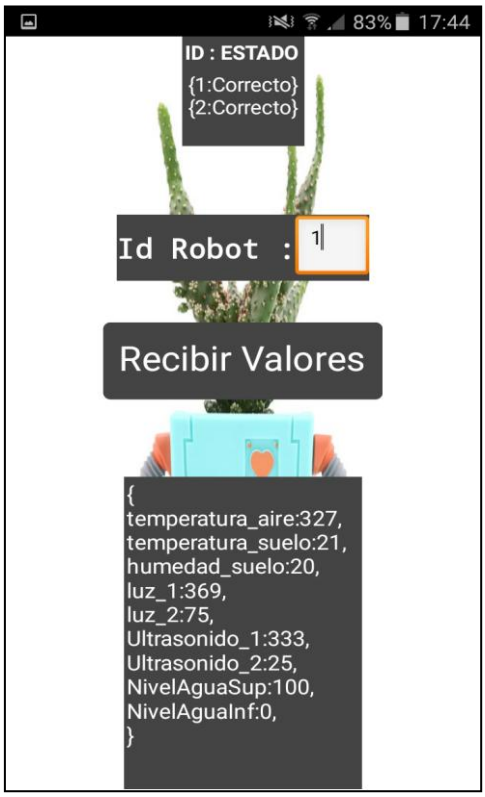


Figura 44 : Respuesta petición sensores

En la figura 44 muestra la aplicación con las respuestas correctas a las peticiones. En el momento que se realiza una petición de cualquier tipo al servidor Tomcat, aparece en la ventana de consola la información referente a la solicitud. Se pueden añadir mensajes de control para saber que se ha procesado.

```
jun 28, 2016 5:44:24 PM org.glassfish.jersey.filter.LoggingFilter log
INFORMACIÓN: 3 * Server has received a request on thread "http-bio-8080"-exec-6
3 > GET http://192.168.1.46:8080/rest/peticion/macetas
3 > accept-encoding: gzip
3 > connection: Keep-Alive
3 > host: 192.168.1.46:8080
3 > user-agent: Dalvik/2.1.0 (Linux; U; Android 5.1.1; SM-J500FN Build/LMY48B)

Se ha solicidado las macetas existentes y su estado
jun 28, 2016 5:44:24 PM org.glassfish.jersey.filter.LoggingFilter log
INFORMACIÓN: 3 * Server responded with a response on thread "http-bio-8080"-exec-6
3 < 200
3 < Content-Type: application/json
```

Figura 45 : Información de la petición en el servidor

La siguiente funcionalidad que se debe probar es la de cambiar de valor un actuador en el robot-maceta. Esta funcionalidad se encuentra en el apartado de *Actuadores* de la aplicación móvil. Solo hace falta escribir el identificador del robot, el actuador y el valor al que se pretende cambiar. El servidor muestra una entrada como la figura 45 y la envía al robot-maceta. Una vez procesada la petición de cambio de valor se envía el valor del sensor de luz instalado en el prototipo. El resultado en la consola de Arduino y el registro de las peticiones del servidor es el siguiente.

```
Id Actuador: 1
Valor: 200
Enviada Respuesta al Servidor

Enviando Valor Sensor........Valor Sensor enviado.
```

Figura 46 : Respuesta consola Arduino

```
jun 28, 2016 5:58:00 PM org.glassfish.jersey.filter.LoggingFilter log
INFORMACIÓN: 4 * Server has received a request on thread "http-bio-8080"-exec-8
4 > GET http://192.168.1.46:8080/rest/peticion/actuador/1/1/200/
4 > accept-encoding: gzip
4 > connection: Keep-Alive
4 > host: 192.168.1.46:8080
4 > user-agent: Dalvik/2.1.0 (Linux; U; Android 5.1.1; SM-J500FN Build/LMY48B)

Respuesta de la maceta 1 confirmada.
jun 28, 2016 5:58:02 PM org.glassfish.jersey.filter.LoggingFilter log
INFORMACIÓN: 4 * Server responded with a response on thread "http-bio-8080"-exec-8
4 < 200
4 < Content-Type: application/json

jun 28, 2016 5:58:03 PM org.glassfish.jersey.filter.LoggingFilter log
INFORMACIÓN: 5 * Server has received a request on thread "http-bio-8080"-exec-10
5 > GET http://192.168.1.151/rest/maceta/sensor/1/4/98
5 > host: 192.168.1.151

Se ha recibido el valor 98 de luz del robot 1
jun 28, 2016 5:58:03 PM org.glassfish.jersey.filter.LoggingFilter log
INFORMACIÓN: 5 * Server responded with a response on thread "http-bio-8080"-exec-10
5 < 200
5 < Content-Type: text/plain
```

Figura 47 : Registro de la peticiones en Tomcat

Después de todo esto la aplicación móvil también recibirá un mensaje de *Ok* confirmando que su solicitud se ha llevado a cabo sin ningún problema. La siguiente funcionalidad que se comprueba es registrar la dirección IP y el cambio de estado de un robot-maceta. En la figura 48 se muestra el resultado del proceso de dichas peticiones.

```
1 > GET http://localhost:8080/rest/maceta/estado/2/0
1 > accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
1 > accept-encoding: gzip, deflate, sdch
1 > accept-language: es-ES,es;q=0.8
1 > cache-control: max-age=0
1 > connection: keep-alive
1 > host: localhost:8080
1 > upgrade-insecure-requests: 1
1 > user-agent: Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36

Se ha cambiado el estado del robot-maceta 2 a Correcto
jun 28, 2016 6:18:42 PM org.glassfish.jersey.filter.LoggingFilter log
INFORMACIÓN: 1 * Server responded with a response on thread "http-bio-8080"-exec-3
1 < 200
1 < Content-Type: text/plain

jun 28, 2016 6:18:50 PM org.glassfish.jersey.filter.LoggingFilter log
INFORMACIÓN: 2 * Server has received a request on thread "http-bio-8080"-exec-4
2 > GET http://localhost:8080/rest/maceta/ip/2/192.168.1.150
2 > accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
2 > accept-encoding: gzip, deflate, sdch
2 > accept-language: es-ES,es;q=0.8
2 > connection: keep-alive
2 > host: localhost:8080
2 > upgrade-insecure-requests: 1
2 > user-agent: Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36

Se ha registrado nueva dirección IP para la robot-maceta 2
jun 28, 2016 6:18:50 PM org.glassfish.jersey.filter.LoggingFilter log
INFORMACIÓN: 2 * Server responded with a response on thread "http-bio-8080"-exec-4
2 < 200
2 < Content-Type: text/plain
```

Figura 48 : Registro de las peticiones en Tomcat 2

Solo falta por comprobar el historial de un actuador o de un sensor. Para dicha prueba no se utiliza la aplicación móvil ya que no está diseñada para esta prueba. Se hace uso de un navegador web, en el cual se solicitan los valores históricos de un actuador. La siguiente figura muestra lo que retorna el navegador al enviar la petición.

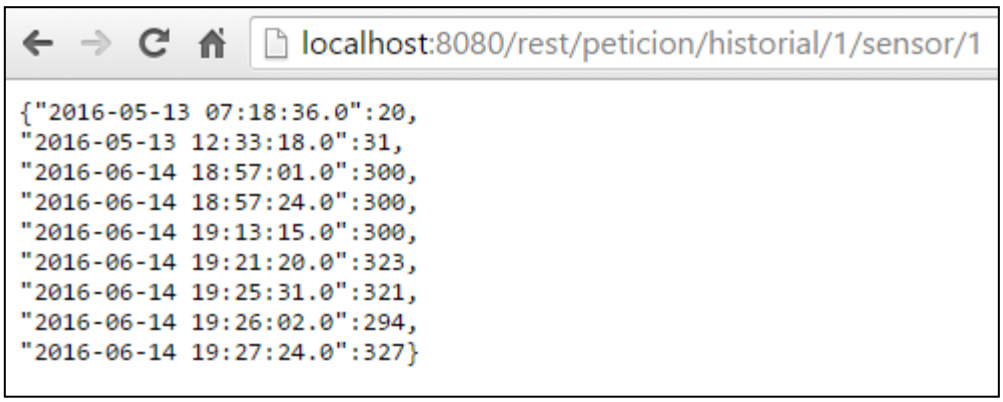


Figura 49 : Resultado petición historial

Las últimas pruebas que se llevan a cabo son las de fiabilidad y control de errores .En estas pruebas se insiste al servidor con peticiones, para ver si es capaz de resolverlas. También se envían peticiones mal formadas o con alguna parte de la arquitectura desactivada para comprobar que el control de errores funciona correctamente.

La tabla siguiente muestra un resumen con todas las pruebas realizadas y las funcionalidades que cubren.

Número de test	Características y requisitos cubiertos	Descripción	Pasada con éxito
Test 01	• Característica CA1 y CA2 • Requisito REA01	Este test comprueba que se pueden realizar peticiones de sensores al servidor.	☑
Test 02	• Característica CA1 y CA2 • Requisito REA02 , RES01 y RES03	Este test comprueba el cambio del valor de un sensor a un robot-maceta.	☑
Test 03	• Característica CA1 y CA2 • Requisito REA03	Este test comprueba que se reciben los identificadores y estados de los robots-macetas registrados en el sistema.	☑
Test 04	• Característica CA1 y CA2 • Requisito REA04	Este test comprueba el envió correcto del historial de un sensor o actuador.	☑
Test 05	• Característica CA1 y CA2 • Requisito RER01	Este test comprueba el envió del valor de un sensor procedente del robot-maceta.	☑
Test 06	• Característica CA1 y CA2 • Requisito RER02 y RES03	Este test comprueba el envió de la dirección IP al servidor y su almacenamiento.	☑
Test 07	• Característica CA1 y CA2 • Requisito RER03 y RES02	Este test comprueba el envió de un estado al servidor.	☑
Test 08	• Característica CA3 • Todos los requisitos	Este test comprueba la fiabilidad y robustez del sistema.	☑
Test 09	• Característica CA4 • Todos los requisitos	Este test comprueba la gestión de todos los errores que puedan llegar a producirse.	☑

Tabla 27 : Pruebas realizadas

5.4 Conclusiones

En este apartado ya se ha conseguido implementar todos los protocolos de la comunicación y el sistema REST completo. Más tarde se ha desarrollado el sistema de hardware de Arduino y la aplicación móvil. Seguidamente se implanta todo en un entorno real y se realizan los test que comprueban que el sistema de comunicaciones cumple con lo establecido.

CAPÍTULO 6

Conclusiones

Este es el último capítulo del trabajo. En él se concluye con lo realizado, se exponen los problemas encontrados y las futuras ampliaciones que puedan existir de este proyecto.

6.1 Trabajo realizado

Lo primero que se ha realizado en este proyecto es la búsqueda de sistemas ya existentes y que presentan alguna similitud al que se ha hecho. Esto proporcionó un análisis tanto cualitativo como cuantitativo de los sistemas encontrados y un punto de partida. Más tarde se decide qué características extraer y que tecnología emplear para un correcto funcionamiento.

Consecuentemente se continúa describiendo el proyecto general y las personas que colaboran. Después se llega al diseño de los casos de uso que posteriormente dan lugar a unos requisitos necesarios.

Una vez especificados los requisitos llega el momento de diseñar todo la arquitectura de la que está compuesta. También se empiezan a diseñar los elementos que se necesitan para realizar la implantación y las pruebas que se van a hacer en un entorno lo más real posible. Los elementos requeridos para estas pruebas son una aplicación móvil y un prototipo de robot-maceta que contiene un sensor y un led que hará de actuador.

A continuación se empieza a desarrollar todos los servicios REST. Se realiza la implementación de la base de datos en el servidor y seguidamente la del robot-maceta y la aplicación móvil.

Finalmente se implantan todos los elementos en un entorno real para poder realizar las pruebas imprescindibles que certifiquen que este proyecto cumple con los objetivos establecidos.

6.2 Problemas encontrados

Es importante destacar los problemas que se han encontrado en el proceso de desarrollo. A continuación se describen los más notables.

Confirmación de envíos

El primer problema que se produjo es cuando se pedía una confirmación de envío al servidor. Muchas veces no se enviaba dicha confirmación o no se procesaba correctamente en el servidor. Finalmente el problema estaba relacionado con el cierre de sockets en la comunicación y el no enviar una cabecera correcta de HTTP.

Protocolo

Este no fue realmente un problema sino un mal diseño del protocolo de comunicación. El primer prototipo que se llevó a cabo funcionaba enviando una petición y esperando que el robot-maceta contestase a dicha petición. Esto conllevaba que el dispositivo Arduino estuviese preguntando al servidor constantemente si existía alguna petición para él. De esta forma se sobrecargaba la red y en el momento que el robot-maceta dejara de funcionar por cualquier motivo se producía un error en la respuesta de la petición. Modificado esto, se permite desacoplar el robot-maceta de las solicitudes de los valores procedentes de los sensores hacia el servidor. En caso de una solicitud de cambio de actuador sí que realiza la comunicación directa con el dispositivo Arduino, pero ya no está continuamente preguntando posibles solicitudes para él.

Conexión base de datos

Cada vez que se quiere realizar alguna operación de lectura o escritura en la base de datos se procede a la llamada de un método que es quien realiza la conexión. Este método devuelve *error* cuando no se puede establecer dicha conexión. Para poder conseguir que la implementación del método fuese la correcta y la que menos errores produjese se tuvo que consultar muchos ejemplos ya que todos no reflejaban el mismo resultado. Finalmente se consiguió un método de conexión fiable.

Implementación en Arduino

Por último, en este apartado, se engloban todos los errores relacionados con los dispositivos hardware. Entre ellos poner el funcionamiento el módulo ESP8266. Este módulo fue costoso de configurar y conectar a una red disponible. También en el desarrollo software para controlar el Arduino se produjeron ciertos errores relacionados con caracteres o particionado de cadenas que se solucionaron con éxito.

6.3 Futuras ampliaciones

Finalmente solo queda destacar las posibles ampliaciones que podría tener este proyecto en el futuro, debido a que en la realización se han producido varias ideas.

La primera de ellas es la realización de un sistema de alarmas avanzado que informe al usuario de la aplicación de un obstáculo que encuentre el robot-maceta en su trayecto. Para esta ampliación se puede hacer uso de servicios de Google que se encargan de gestionar y avisar de estas alarmas.

Otra posible ampliación es la posibilidad de gestionar coordenadas de cada robot-maceta. Esto permitirá el poder saber en cualquier momento en qué posición se encuentra y de esta forma situar en un mapa todos los robots disponibles.

La última ampliación observada es implementar otras peticiones que se puedan realizar. Un ejemplo sería el poder recibir en la APP qué robots-maceta tienen el depósito de agua en estado crítico o cual de ellos se está moviendo en ese momento.

Referencias

Bibliográficas

[1] Banzi, M., & Shiloh, M. (2014). Getting Started with Arduino: The Open Source Electronics Prototyping Platform. Maker Media, Inc..

[2] Richardson, M., & Wallace, S. (2012). Getting started with raspberry PI. " O'Reilly Media, Inc.".

[3] Purushothaman, J. (2015). RESTful Java Web Services. Packt Publishing Ltd.

[4] Richardson, L., & Ruby, S. (2008). RESTful web services. " O'Reilly Media, Inc.".

[5] Brittain, J., & Darwin, I. F. (2007). Tomcat: The Definitive Guide: The Definitive Guide. " O'Reilly Media, Inc.".

[6] Vohra, D. (2012). Java EE development with Eclipse. Packt Publishing Ltd.

[7] Delisle, M. (2009). Mastering phpMyAdmin 3.1 for effective MySQL management. Packt Publishing Ltd.

[8] PÉROCHON, S. H. S. (2014). Android: Guía de desarrollo de aplicaciones para Smartphones y Tabletas (2a edición). Ediciones ENI.

[9] MySQL, A. B. (2001). MySQL.

Páginas web

[10] App Inventor - Primeros pasos. Recuperado 1 de junio de 2016, a partir de https://docs.google.com/presentation/d/1mJGFvmOmClaok8cJ_-5jqYh8GuDUYSWjjfy_3JdrhJk/embed?hl=es&size=l&usp=embed_facebook

[11] Servidor Web con WIFI ESP8266 | Tutoriales Arduino. Recuperado 1 de junio de 2016, a partir de http://www.prometec.net/servidor-web-esp8266/