UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Sistema de Interconexión de Sensores y Actuadores Distribuidos

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Israel Beltrán Arias

Tutor: José Luis Poza Luján
Juan Luis Posadas Yagüe

[2015/2016]

Resumen

Actualmente muchos recursos son malgastados debido a una mala gestión del cuidado de las plantas. Para optimizar estos recursos y ofrecer una alternativa de calidad al usual cuidado del jardín, se propone dotar de inteligencia a las macetas que lo componen. De forma que las macetas puedan mantenerse de manera autónoma. Para gestionar una red de macetas con sus propias características y necesidades, se necesita un protocolo que gestione todas las comunicaciones y avisos que se den en este sistema. En este proyecto se plantea el estudio, diseño e implementación de dicho protocolo, que permite la interactuación de los maceteros (nodos) tanto entre ellos como con el entorno. Para testear el protocolo, se emplean dos microcontroladores que harán de maceteros y que demuestran que se pueden comunicar entre ellos de forma autónoma.

Palabras clave: Arduino, Microcontroldaor, WiFi, Esp8266, Protocolo, Sistema, Usuarios, Jardin,Inteligente, Actuador, Sensor.

Resum

Actualment molts recursos són malgastats a causa d'una mala gestió de la cura de les plantes. Per tal d'optimizar aquests recursos i oferir una alternativa de qualitat a l'usual cura del jardí, es proposa dotar d'intel·ligència als testos que ho componen. De manera que els terstos puguen mantindre's de forma autònoma. Per gestionar una xarxa de testos amb les seues pròpies característiques i necessitats, es necessita un protocol que gestione totes les comunicaciones y avisos que es donen en aquest sistema. En aquest projecte es planteja l'estudi, disseny i implementació d'aquest protocol, que permet la interactuació dels testos (nodes) tant entre ells com amb l'entorn. Per a testejar el potocol, s'emprenen dos microcontroladors que faran de testos i que demostren que es poden comunicar entre ells de forma autònoma.

Palabras clave: Arduino, Microcontroldaor, WiFi, Esp8266, Protocol, Sistema, Usuaris, Jardí,Intel·ligent, Actuador, Sensor.

Abstract

Currently many resources are wasted due to mismanagement of plant care. To optimize these resources and offer an alternative of quality to the usual care of the garden, it is proposed to add intelligence to the pots that compose it, so that the flowerpots can be maintained in an autonomous way. To manage a network of flowerpots with its own characteristics and needs, a protocol that handles all communications and notices that occur in this system is needed. In this project the study, design and implementation of the above mentioned protocol, which allows the interaction of the flowerpots (nodes) both among themselves and with the environment.

To test the protocol two microcontrollers that will work as flowerpots as well are used; these will demonstrate that they can communicate among themselves independently.

Keywords: Arduino, microcontroller, WiFi, Esp8266, protocol, System, Users, intelligent garden, actuator, sensor.

Índice

Ilustraciones

Tablas

Código

1. Introducción

1.1. Contexto

Hoy en día, la labor de mantener un jardín en óptimas condiciones conlleva el uso de una gran cantidad de tiempo y de recursos. Con el fin de optimizar los recursos utilizados y haciendo uso de la tecnología, se pretende crear un jardín que pueda mantenerse de forma autónoma, avisando ante la falta de algún recurso necesario para su confort, tal como ya se hace en los sistemas domóticos [1]. Para ello, se ha propuesto una red de elementos inteligentes que se encuentren alojados en macetas. Estas macetas dispondrán de ruedas para moverse y de un depósito para regar la planta que alojan. Para todo esto se necesita una "inteligencia" que gobierne todos los sensores y actuadores [2] que contenga la maceta.

Por lo tanto se propone este proyecto, que tiene un contexto muy concreto. Se encuentra posicionado entre un marco de 5 proyectos integrados. Uno se encarga de montar el hardware que va a permitir al Sistema ejecutar las funciones programadas, en este caso se trata de un Jardín inteligente capaz de buscar la luz o avisar de la humedad de la tierra, incluso de la capacidad de riego. Otro se encarga del Servidor y la Base de Datos, que conecta el interior del Sistema con el exterior. Un tercero crea una aplicación móvil que interactúa con el Servidor. El cuarto crea una página web que también interactuará con el Servidor de igual forma que con la app. Por último, este proyecto, se encarga de comunicar el Servidor con el Hardware responsable del control de las macetas, así como de la comunicación de las Macetas entre sí.

Para ello se propone crear un protocolo [3] que gestione las comunicaciones en este Sistemas Distribuido y que pueda gestionar también las conexiones con el Servidor. Intentando, por tanto, que el Sistema sea lo más eficiente posible, ya que permitirá cumplir las peticiones o posibles órdenes que envíe un usuario externo a las Macetas, y que las diferentes macetas sean capaces de interactuar con su entorno.

1.2. Objetivos

Por tanto, el objetivo principal del proyecto es: desarrollar un protocolo de comunicaciones en una red Distribuida de microcontroladores, capaz de interactuar con el entorno y de gestionar debidamente las comunicaciones entre los diferentes Usuarios internos del Sistema. Lo que permitirá gestionar de forma autónoma las condiciones necesarias de confort de cada una de las macetas de un "Jardín Inteligente". Este objetivo se desarrolla a través de una serie de objetivos específicos.

- Estudiar aquellos sistemas que desarrollan soluciones similares para así obtener una serie de características generales que faciliten la especificación del sistema.

- Especificar, empleando estándares al uso, los requisitos del sistema para así poder diseñar el mismo desde un punto de vista de la Ingeniería informática.
- Diseñar el sistema empleando estándares al uso, de forma que cubran todos los requisitos del objetivo anterior.
- Implementar y evaluar el diseño realizado para comprobar que los requisitos iniciales se han cumplido

1.3. Descripción documento

El presente documento está estructurado en 6 capítulos con la intención de ir detallando y concretando el desarrollo del sistema. El contenido de los capítulos se expondrá a continuación.

Capítulo 2. Estudio del entorno:

Uno de los primero pasos a la hora de desarrollar un proyecto de estas características es hacer un estudio y un análisis sobre los sistemas similares existentes actualmente. Esto permite valorar la situación de mercado, los puntos fuertes y las características que tiene que tener el sistema propuesto. Estos aspectos son los que se recogen en este capítulo.

En este capítulo, se comenzará mostrando aquellos sistemas similares al proyecto y a continuación se realizará un análisis de los sistemas mostrados que dará lugar a una recapitulación de las características que se consideren importantes para el sistema.

Capítulo 3. Especificación de requisitos:

Con el fin de determinar con mayor formalidad el sistema, se hará una especificación de requisitos de manera formal utilizando el estándar IEEE 830 [4]. A partir de las características determinantes en el capítulo anterior se puede especificar la funcionalidad que tendrá el sistema.

Capítulo 4. Diseño:

Antes de la implementación del sistema, se tendrá que llevar a cabo un diseño del mismo. Para abordarlo se utilizarán diferentes diagramas con la intención de aclarar cómo se implementará el sistema posteriormente. El diseño se basará en los requisitos funcionales detallados en el capítulo anterior.

Capítulo 5. Implementación:

Llegados a este punto se implementará el sistema basándose en el diseño del capítulo anterior. Se describirá la tecnología utilizada, las líneas de código más relevantes y se mostrarán capturas y tablas con las pruebas realizadas.

Capítulo 6. Conclusiones:

Finalmente, se concluirá este documento con las conclusiones que se hayan extraído y se mostrará la bibliografía consultada

2. Estudio del entorno

2.1 Introducción:

En este apartado se va a descubrir el inmenso uso que se le puede dar a las placas Arduino [5] y a los diferentes microcontroladores, sobre todo en referencia a los distintos sensores y actuadores que permiten conectar y compartir con otros dispositivos. A unos cuantos microcontroladores se les puede agregar un adaptador inalámbrico, más conocido como WiFi [6] que permite comunicar casi cualquier parte del mundo, se abre por tanto un abanico de posibilidades respecto a qué placas utilizar y a qué sistemas se pueden conectar.

Dado que las placas Arduino son muy utilizadas, más baratas y manejables que las Raspberry Pi [7], este proyecto se va a centrar en el estudio de estas placas. Lo que más se ha encontrado son las comunicaciones entre los diferentes sensores con la placa, pero no así la comunicación entre placas. En su mayoría, las placas de microcontrolador como Raspberry pi o Arduino actúan de servidor, (otras menos potentes envían la información a un servidor externo).

Se pueden conectar en serie o por WiFi dos placas pero en los dos casos hay siempre un limitante en la comunicación. Por eso este estudio se centra en los protocolos de comunicación de las placas Arduino.

Primero, se estudiarán las comunicaciones que tienen con otras placas y dispositivos. En segundo lugar, como comunicar dos placas Arduino, y por último intentar comunicar una red de placas Arduino en un mismo sistema. Queda por resolver qué modelo de comunicación se va a seguir en este sistema, y los diferentes protocolos que se van a utilizar.

2.2 Sistemas Similares:

En este apartado se van a mostrar los sistemas que se asemejan a este proyecto para su consiguiente análisis, del que se va a extraer un conjunto de características que facilitarán la identificación de los requisitos y el modelaje del Sistema.

Coche de televigilancia

El primer Sistema se trata de un vehículo teledirigido que se maneja remotamente desde cualquier aplicación con acceso a WiFi. Este sistema se caracteriza en especial por el manejo de la placa Raspberry Pi, que actúa como emisor WiFi, y que envía la información a cualquier servidor web que se desea. La placa Raspberry Pi es una placa que gracias a sus conexiones puede utilizarse como un micro ordenador.

La imagen que se puede ver a continuación es el robot diseñado por SISTEMAS O.R.P. Se trata de un coche de televigilancia.

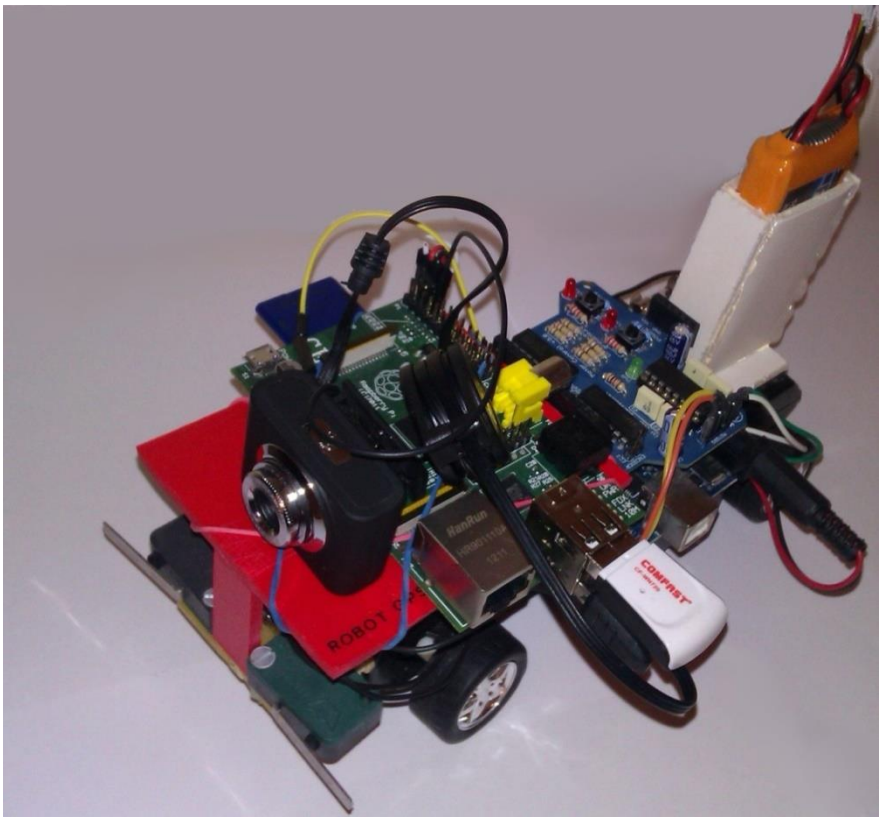

Ilustración 1 . Coche de televigilancia SISTEMAS O.R.P.[1]

Se compone de una placa Arduino conectada a unos motores, un coche RC y una Raspberry Pi conectada a WiFi y a una webcam.

La Raspberry actuará como controlador de la cámara web y como servidor. Además está conectada por puerto serie al Arduino que va a actuar como controlador de los demás dispositivos, y permite que se mueva el conjunto controlando las ruedas.

Coche inteligente WiFi-Arduino

El segundo Sistema es un robot inteligente, que permite grabar imágenes en HD. Viene de una marca china, Huijing. Y se compone de un chasis con dos ruedas, una placa Arduino Uno, un módulo WiFi, una cámara usb HD y unos sensores de ultrasonido para detectar posición. Se puede encontrar a la venta en AliExpress.

[1] http://www.sistemasorp.es/2013/03/23/televigilancia-con-un-coche-rc-arduino-y-la-raspberry-pi/

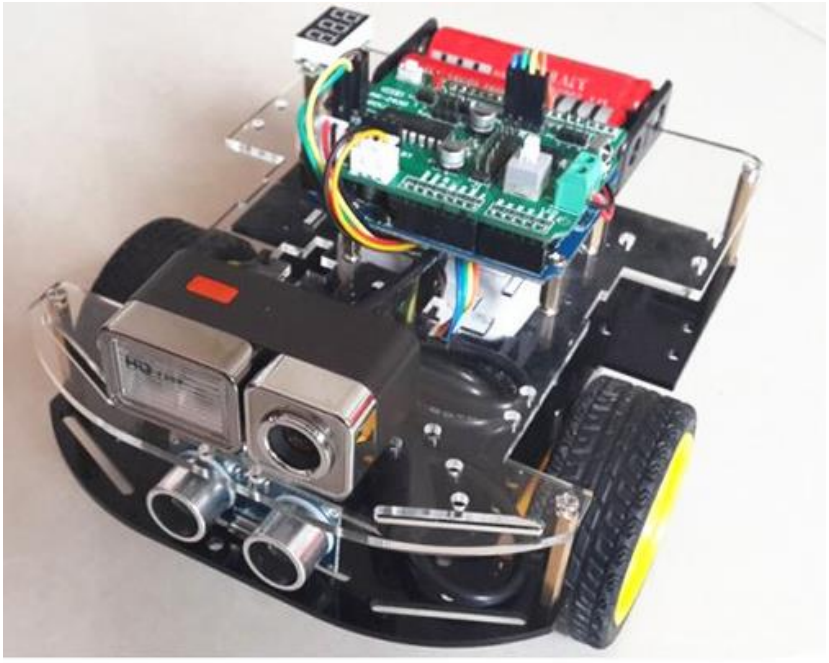

Ilustración 2. Coche inteligente WiFi-Arduino. [2]

Como se puede apreciar este mecanismo pone al descubierto las múltiples utilidades y la variedad de sensores que se pueden configurar a la placa Arduino Uno de forma que un mismo dispositivo dé una variada y abundante información. En este caso el Arduino actúa como controlador de todos los sensores (ultrasonido, cámara usb HD, etc) y comunica mediante un módulo WiFi a la red que se desea conectar.

Robot by PS3 controler

El tercer Sistema es un coche que permite el manejo remoto desde un mando de la Play Station 3. Por medio de la tecnología WiFi y Bluetooth. Este vehículo no es de ninguna empresa, si no que se trata de un proyecto de un particular en una página web donde se muestra las máquinas y dispositivos que se han fabricado, éste en concreto es un usuario de “instructables”, llamado “gtri2013”.

[2] http://es.aliexpress.com/store/product/FOR-arduino-wifi-wireless-video-smart-car-smart-car-kit-FOR-arduino-car-four/820217_1905434008.html

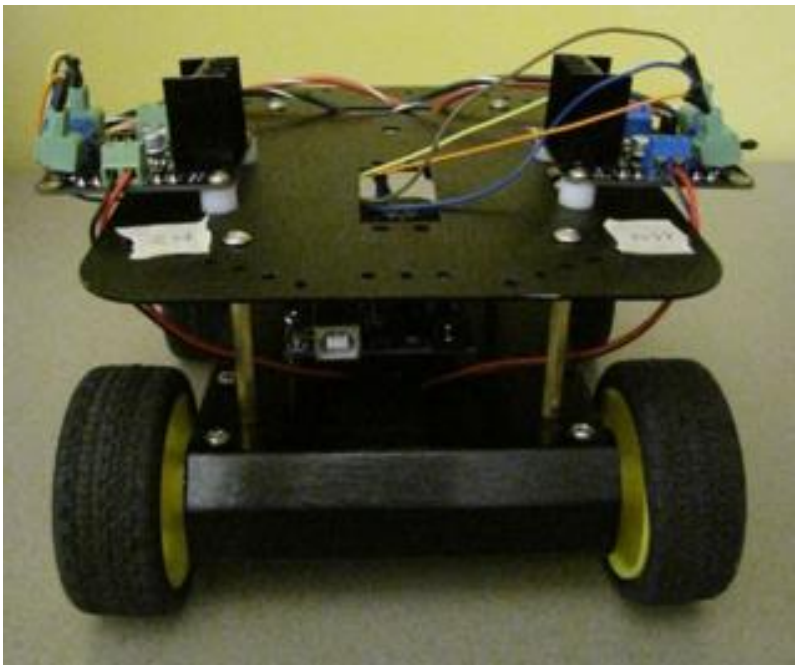

Ilustración 3. Robot by PS3 controler. [3]

Se precisa para montarlo un chasis o estructura que aguante el conjunto de los dispositivos que lo componen y unas ruedas. En este vehículo se utilizan dos placas Arduino Uno, con sus respectivos dos Shield WiFi para conectar los Arduino con su red privada y un dispositivo para utilizar Bluetooth. También se utiliza un mando de la Play Station 3 para poder dirigir el vehículo. Se utiliza la comunicación serie entre las placas para gestionar a la vez los dos motores.

Coche inteligente Bluetooth multifunción

El cuarto Sistema es un coche que utiliza una tarjeta controladora Arduino Uno, control remoto MCU, receptor de infrarrojos, módulo de ultrasonido y adaptador Bluetooth. Este vehículo se puede encontrar a la venta en una tienda online llamada “Banggood”.

[3] http://www.instructables.com/id/Robot-driven-by-PS3-controller-through-Arduino-and/

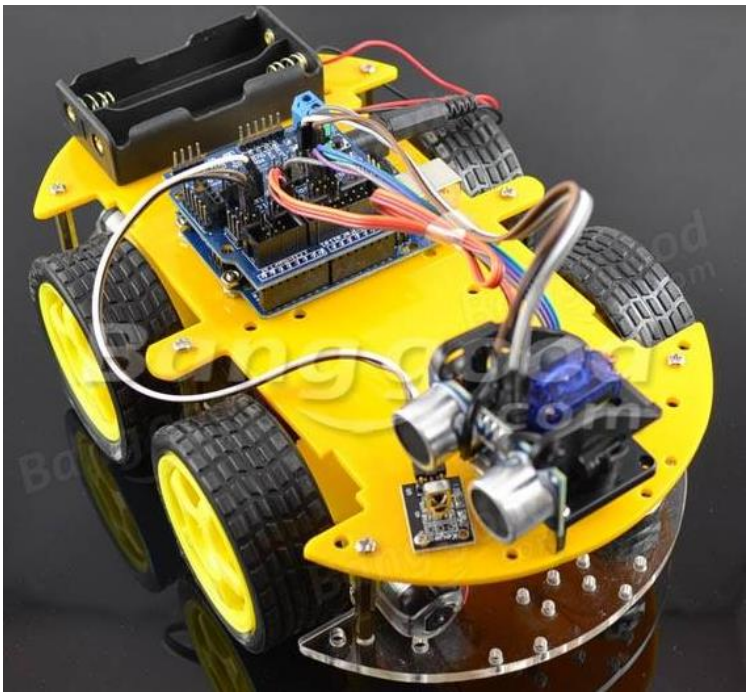

Ilustración 4. Coche inteligente Bluetooth multifunción.[4]

Como se aprecia con este ejemplo se puede conectar una gran variedad de sensores con este controlador y esto permite una gran escalabilidad con los módulos reutilizables. Este vehículo se puede mover solo, siguiendo un recorrido establecido, utilizando los infrarrojos y evitando obstáculos y colisiones con el ultrasonido. Además también se puede manejar mediante un mando radiocontrol.

WiFi 3-wheeled robot

El quinto Sistema, la ilustración que se puede ver a continuación, se trata de un robot multidireccional que desarrollo le empresa Mimetics Digital Education, se compone de una placa Arduino, conectado a un chip ESP8266 [8] para dotarlo de comunicación WiFi, y poder comunicarlo así con el entorno. Tiene sensores de ultrasonido, un chasis con motores y ruedas en tres direcciones.

[4] http://www.banggood.com/es/Multifunction-Bluetooth-Controlled-Robot-Smart-Car-Kits-For-Arduino-p-906628.html

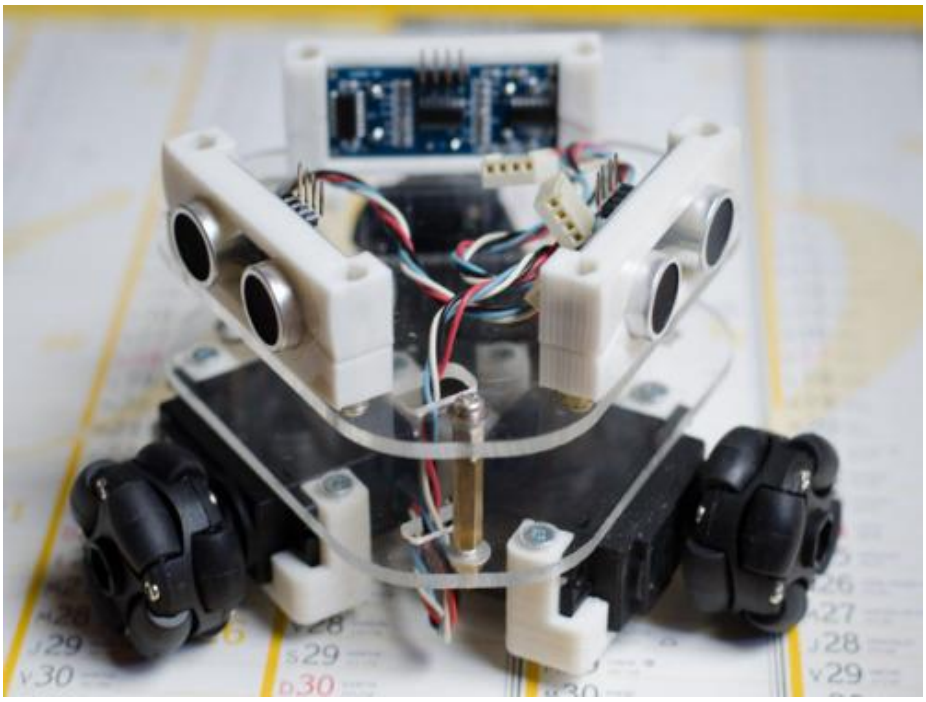

Ilustración 5. WiFi 3-wheeled robot. [5]

Como se puede apreciar al tener los tres motores y los tres ultrasonidos conectados al mismo Arduino, le permite que un mismo programa o una misma configuración puedan manipular todos los componentes dotándolo de una funcionalidad como mantener una dirección, pese a que lo muevan o lo giren, como una brújula, etc. Además cuenta con una pantalla LCD que permite conocer el estado del Sistema y ayuda a interactuar con él.

Robot coche cámara HD- WiFi

El sexto Sistema es un vehículo motorizado con múltiples dispositivos y sensores. Multifuncional, con variedad de comunicación, auto-control y control dirigido con diferentes dispositivos. Este vehículo está a la venta en Aliexpress, pero proviene de la tienda Kuongshun Electronic Limited, una tienda china.

[5] http://mimeticscanada.tumblr.com/post/127314023427/thingiverse-github-laurentfr-w3w-bot-a

Ilustración 6. Robot coche cámara HD- WiFi. [6]

Como se puede apreciar, Arduino tiene una gran capacidad de adaptación a diferentes sensores y estructuras, y te permite desde el autogobierno de un sistema hasta el control dirigido desde PC's o dispositivos móviles.

Miroad SM2 Robot Car

El séptimo Sistema es un vehículo inteligente hecho con Arduino, capaz de seguir un circuito de forma autónoma, evadir obstáculos y evitar colisiones, gracias a los diferentes sensores de posición, infrarrojos, ultrasonido, etc. También es capaz de mostrar en una pantalla LCD los mensajes programados y los posibles mensajes de error.

[6] http://es.aliexpress.com/item/Wireless-Wifi-Robot-Car-Kit-for-Arduino-Hd-Camera-Ds-Robot-Smart-Educational-Robot-Kit-for/32644952644.html?spm=2114.43010408.3.62.no4NMP

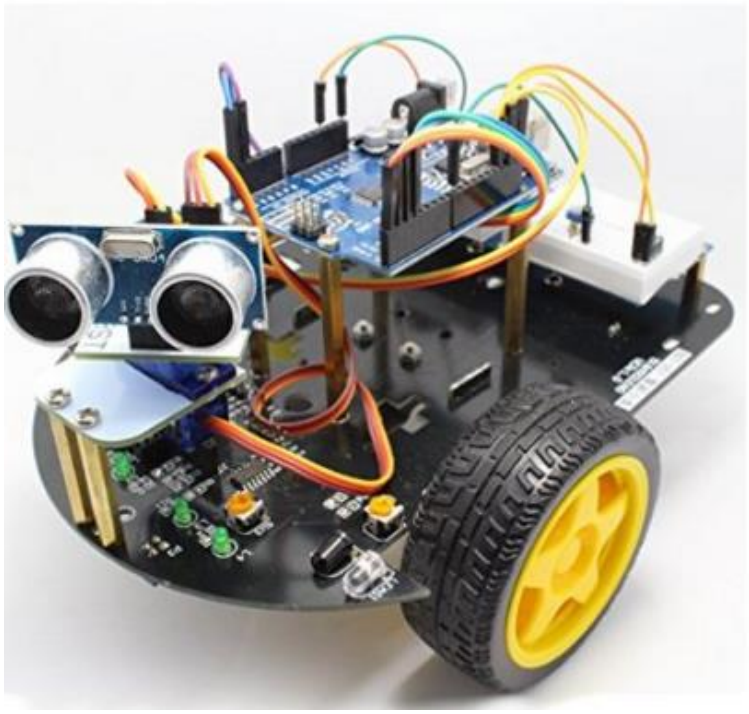

Ilustración 7. Miroad SM2 Robot Car[7]

Este robot se vende actualmente en Amazon, un usuario llamado Miroad y por un precio económico. Permite poner muchos sensores y expandir este diseño con tus propios módulos. Este modelo no tiene un módulo de conexión WiFi, sin embargo, cuenta con un mando con el que controlar el robot y detector de caminos mediante infrarrojos.

ESP8266 WiFi RC car

El octavo Sistema es un vehículo teledirigido, que se puede manejar desde cualquier Smartphone. Se comunica mediante el chip ESP8266 WiFi de Arduino que envía los estados y las órdenes del motor, para saber los cambios de velocidad. La dirección se cambia desde el Smartphone con la propia inclinación del móvil, a partir de un programa creado por el usuario.

[7] http://www.amazon.com/Miroad-Intelligent-Robotics-Obstacle-avoidance/dp/B010XGLEKU

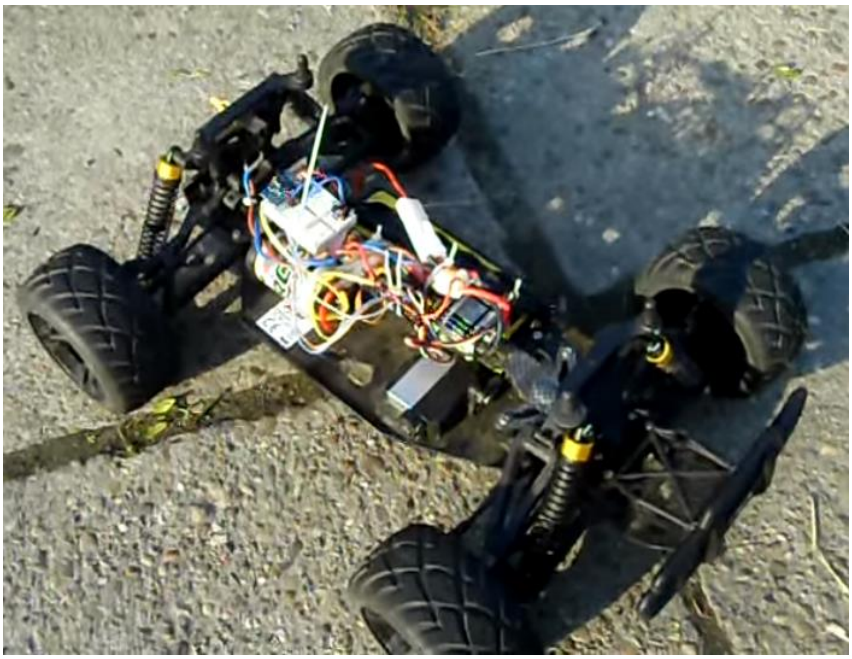

Ilustración 8. ESP8266 WiFi RC car [8]

Este vehículo no está en venta, pero sí el programa que lo puede controlar, que se puede descargar en la página de su creador. Y se puedes dejar un donativo de 5$, para ayudarle con sus siguientes proyectos. Se puede encontrar también en aplicaciones de la tienda de Google Play, como RoboRemo. Este vehículo no utiliza ninguna placa como microcontrolador, sino únicamente el chip WiFi ESP8266 que controla con las señales recibidas los motores.

2.3. Análisis

2.3.1. Análisis Cuantitativo:

Se han recopilado un conjunto de sistemas que cumplen algunos de los puntos que se va a abordar en este proyecto y se ha realizado un profundo análisis sobre estos. Para mostrarlo de forma sencilla y ordenada se ha preparado una tabla con los diferentes puntos analizados y unos diagramas.

En este apartado se ha tratado el material que utiliza cada sistema, y todo el hardware del que se compone. Como placas y sensores.

Sensores: Los diferentes sensores que utilizan los sistemas que se han estudiado.

- Ultrasonido: Si el sistema estudiado utiliza sensor de ultrasonido.
- Velocidad: Si el sistema estudiado utiliza sensor de velocidad.
- Infrarrojos: Si el sistema estudiado utiliza sensor de infrarrojos.

[8] http://www.roboremo.com/esp8266-wifi-rc-car.html

- RGB: Si el sistema estudiado utiliza sensor de imágenes RGB (red, green, blue).

Placas: Las diferentes placas de microcontroladores que utiliza cada sistema.

- Raspberry: Si el sistema utiliza la placa Raspberry.
- Arduino: Si el sistema utiliza una placa Arduino.
- Otro: Si el sistema utiliza cualquier otro controlador diferente a los vistos con anterioridad.

	Sensores					Placas		
Sistemas	UltraSonido	Velocidad	Infrarojos	RGB	Motores	Raspberry	Arduino	Otro
S1	No	No	No	Si	Si	PI modelo B	Duemilanove (UNO)	No
S2	Si	No	No	Si	Si	No	UNO R3	No
S3	No	No	No	No	Si	No	UNO	No
S4	Si	Si	Si	No	Si	No	Atmega-328p	L298N
S5	Si	Si	No	No	Si	No	NANO	GY-85 9-dof IMU
S6	Si	No	Si	Si	Si	No	UNO R3	No
S7	Si	No	Si	No	Si	No	UNO R3	No
S8	No	Si	No	No	Si	No	No	N.D.[9]

Tabla 1. Tipos de Sensores y placas de los diferentes sistemas a estudiar. (Propio)

En la Tabla 1, se encuentra una variedad de sistemas, que utilizan Arduino, ya que en estas placas se ha centrado el estudio de este proyecto. Y también se puede apreciar la variedad de sensores que puede llegar a tener un sistema. Han aparecido casi siempre motores que permiten el movimiento de los sistemas.

En este apartado se estudian las comunicaciones que puede tener cada sistema y qué tipo de conexión y protocolo utilizan.

Comunicaciones: Los diferentes tipos de comunicación que puede utilizar cada sistema.

- Serie: Si el sistema puede comunicarse por serie.
- Bluetooth: Si el sistema puede comunicarse vía Bluetooth.
- Radiocontrol: Si el sistema cuenta con comunicación radiocontrol.
- WiFi: Si el sistema cuenta con el tipo de comunicación WiFi

WiFi: El diferente dispositivo o componente que se utiliza para comunicar vía WiFi cada sistema.

- Shield: Si el sistema utiliza el shield de Arduino para utilizar la comunicación WiFi.

[9] No disponible.

- ESP8266: Si el sistema utiliza el chip ESP8266 para comunicar vía WiFi.
- Otra: Si utiliza cualquier otro dispositivo para comunicar vía WiFi.

Protocolos: Los diferentes protocolos que pueden utilizar los sistemas estudiados en sus comunicaciones.

- TCP/IP: Si utiliza el protocolo TCP/IP [9].
- Ethernet: Si utiliza un protocolo de Ethernet.
- Propio: Si utiliza un protocolo propio para controlar las comunicaciones.

	Comunicaciones				WiFi		Protocolos		
Serie	Bluetooth	Radiocontrol	WiFi	Shield	ESP8266	Otra	TCP/IP	Ethernet	Propio
Si	No	No	Si	No	No	Raspberry pi	Si	No	Si
Si	No	No	Si	No	No	NS/NC	N.D.	N.D.	N.D.
Si	Si	No	Si	Si	No	Adapter Wireless-N300	N.D.	N.D.	N.D.
Si	Si	Si	No	No	No	No	N.D.	N.D.	N.D.
Si	No	No	Si	No	Si	No	Si	Si	Si
No	No	No	Si	No	No	MTK7620N	N.D.	N.D.	N.D.
Si	No	No	No	No	No	No	N.D.	N.D.	N.D.
Si	No	No	Si	No	Si	No	Si	N.D.	Si

Tabla 2. Tipos de comunicación y protocolos. (Propio)

Como se puede observar en la Tabla 2, se aprecian una gran cantidad de sistemas que utilizan Bluetooth y comunicación WiFi para controlar y manejar los diferentes Sistemas o dispositivos. Sin embargo no se encuentra con tanta frecuencia es uso el Chip ESP8266 en que se ha centrado este proyecto debido a su económico precio y a su reducido tamaño.

Tampoco es fácil encontrar Sistemas con un Protocolo propio que comunique mediante WiFi el sistema con el usuario. Y lo que no se ha visto casi por ningún sitio son sistemas que comuniquen más de un controlador entre sí.

2.3.2. Análisis Cualitativo

Aquí se van a analizar todos los sistemas desde una perspectiva más subjetiva, haciendo una valoración cualitativa de sus características y componentes. Los aspectos principales que se van a tener en cuenta serán:

- Apariencia: Si la interfaz del Sistema (del tipo que sea) es agradable a la vista. Tiene un estilo apreciable, etc.
- Facilidad de montaje: Si se explica de forma sencilla como construirlo.
- Facilidad de uso: Si se puede manejar el Sistema de forma sencilla.

- Facilidad de aprendizaje: Si es fácil e intuitivo la forma de aprender a manejar el sistema.
- Facilidad de programación: Si el código se encuentra ordenado, documentado y es entendible sin mucho esfuerzo.
- Escalabilidad: Si se pueden añadir más funcionalidades o componentes con facilidad
- Modulación: Si se puede modular sus componentes y, por lo tanto, acoplar y desacoplar tanto componentes como funcionalidades.

Todos estos aspectos se recogerán en una tabla, donde se medirán con una puntuación del 1 al 5 de menor a mayor satisfacción.

Sistemas	Apariencia	Facilidad de montaje	Facilidad de Uso	Facilidad de Aprendizaje	Facilidad de Código	Escalabilidad	Modulación
S1	2	3	4	3	4	3	2
S2	3	1	3	1	1	4	4
S3	1	2	3	2	1	3	2
S4	3	2	2	1	1	3	3
S5	5	3	4	3	4	4	4
S6	4	1	2	1	1	4	4
S7	3	1	2	1	1	5	5
S8	5	4	4	4	4	4	3

Tabla 3. Análisis Cualitativo

Como se aprecia en estas tablas se puede comprobar que ahora mismo en el mercado hay muchos Sistemas que no son fáciles de manejar, ni de aprender a hacerlo. Por otro lado, parece ser que casi todos tienen una gran capacidad de modularse y simplificar cada una de las funciones para poder luego quitar o añadir funcionalidades y permitir también ser más escalable.

2.4. Síntesis

A partir de los análisis desarrollados en el apartado anterior, es posible determinar una serie de características que el sistema deberá cumplir. Estas características deben servir como guía para el desarrollo de las mismas. En principio, sin que sea excluyente a otras posteriores, se determinan las siguientes:

- CA01: El sistema deberá ser capaz de gestionar una gran cantidad de dispositivos o sensores que garanticen el confort de la maceta.
- CA02: El sistema deberá ser capaz de conectarse vía WiFi a los diferentes Arduinos y con el servidor.
- CA03: El sistema deberá ser capaz de comunicarse vía WiFi a los diferentes Arduinos y con el servidor.

- CA04: El sistema deberá contar con un protocolo que gestione las comunicaciones entre los diferentes dispositivos y que tenga una fiabilidad necesaria.
- CA05: El sistema deberá tener una interfaz operativa que muestre de forma inequívoca el estado del mismo.
- CA06: El sistema debe ser sencillo de programar, por lo que se recomienda emplear cierta modularidad en los códigos.
- CA07: El sistema deberá dar soporte a la modularidad de los robots-macetas
- CA08: El sistema debe disponer de un protocolo que garantice la correcta comunicación entre todos los módulos y componentes.
- CA09: El sistema deberá ofrecer a los módulos la capacidad de conectarse y desconectarse del servidor.

2.5 Tecnología a utilizar

Las tecnologías que se van a utilizar para este proyecto serán las placas que se alojaran en los maceteros, junto con sus componentes necesarios para poder gestionar autónomamente el macetero y sus necesidades de confort. A parte, también necesitarán los componentes de comunicación para enviar y recibir información del Servidor y el resto de Usuarios.

El microcontrolador que se va a utilizar en este proyecto es Arduino UNO R3 ATmega328. Se trata de una placa con 14 pines digitales de entrada y salida, 6 entradas analógicas y con una conexión USB.

Ilustración 9. Arduino UNO R3

El Arduino irá conectado a todos los sensores que necesita el macetero para controlar su confort. Así como a los actuadores que le posibilitarán moverse y abastecerse de agua. También se conectará a un chip de comunicación, que le permitirá recibir y enviar la

información pertinente en el sistema. Se trata del ESP8266, que es un puente de puerto serie a WiFi. Incluye un microcontrolador para manejar el protocolo TCP/IP y el software necesario para la conexión 802.11. Dispone de entradas/salidas (I/O) digitales .

Ilustración 10. ESP8266

El ESP8266 de cada Arduino, conecta de manera inalámbrica todos los componentes del sistema utilizando el estándar IEEE802.11, también conocido como WiFi.

Por último, el servidor que recibirá los mensajes de los Arduinos, utilizará una capa REST para atender a todas las solicitudes que tenga, ya sea internas al sistema o desde el exterior mediante una aplicación web o de móvil.

2.6 Conclusiones

En este capítulo, se ha realizado un estudio de Sistemas similares al que se propone. Luego se ha realizado un análisis de los sistemas en base a sus características, valorándose de forma cuantitativa y cualitativa, recogiendo toda la información en tablas.

A partir de estos sistemas analizados se puede deducir que la tecnología que se va a emplear está muy extendida, de hecho, hay una gran variedad de sensores y dispositivos al alcance de cualquiera para obtener todo tipo de información y muchos protocolos con los que comunicar las diferentes placas.

Finalmente, se han sintetizado las características más destacables o deseables de los sistemas. Estas características deberán ser la base sobre la que se especificarán los requisitos del proyecto. Estos requisitos se muestran en el siguiente capítulo.

3. Especificación de Requisitos

3.1. Introducción

Una vez realizado el estudio de sistemas similares y el análisis de los mismos, estableciendo las características y la tecnología que se va a utilizar, se procederá a una definición formal del sistema, utilizando para eso el estándar IEE 830 "especificación de requisitos de software".

Con la especificación de requisitos se pretende determinar de manera completa la descripción del sistema que se va a desarrollar. Se definirá la terminología que se va a emplear en este proyecto, indicando las palabras relevantes y acrónimos utilizados. También se indicarán las personas involucradas en el proyecto. Luego se abordará desde el punto de vista del Usuario, las características vistas anteriormente, de las que saldrán los requisitos y las funcionalidades.

3.2. Terminología

En este proyecto se va utilizar un lenguaje enfocado a las comunicaciones y a diferentes conceptos hardware. Para esto, es indispensable conocer algunos términos que se utilizan a menudo en estos ámbitos y que ayudan a la comprensión de lo que se intenta lograr.

Macetero: en este proyecto, se considera que un macetero es el elemento que se pretende disponer en el Sistema. Constará de un microcntrolador, con todas sus capacidades y sensores. De un depósito de agua, ruedas para desplacerse, y la maceta en sí.

Vecinos: en este proyecto, se entiende por vecino al macetero que se encuentre registrado en el Sistema y que se tenga su identificador y dirección ip en la memoria.

Protocolo: en este proyecto, se considera que un macetero es el elemento que se pretende disponer en el Sistema. Constará de un microcntrolador, con todas sus capacidades y sensores. De un depósito de agua, ruedas para desplacerse, y la maceta en sí.

Síncrono: en este proyecto, se considera que un elemento de comunicación es síncrono cuando el emisor espera a la respuesta de la solicitud por parte del receptor.

Asíncrono: en este proyecto, se considera que un elemento de comunicación es asíncrono cuando el emisor no espera la respuesta de la solicitud por parte del receptor.

Acoplado: en este proyecto, se considera que una función es acoplada cuando la comunicación que interviene es una comunicación síncrona.

Desacoplado: en este proyecto, se considera que una función es desacoplada cuando la comunicación que interviene es una comunicación asíncrona.

Arduino: en este proyecto, el microcontrolador empleado será Arduino, por tanto se entenderá que cuando se refiera en el texto a Arduino, se está hablando de cualquier microcontrolador que se quiera utilizar. En este caso Arduino UNO Rev3 (basado en AT Mega).

ESP8266: en este proyecto, el chip de comunicaciones empleado será el ESP8266, por tanto se entenderá que cuando se refiera en el texto a ESP8266, se está hablando de cualquier chip de comunicaciones que se quiera utilizar. En este caso ESP8266.

Sistema: en este proyecto, se considera que un sistema es el conjunto de maceteros, usuarios y componentes que se relacionan entre sí para alcanzar un objetivo.

Servidor: en este proyecto, se considera servidor al equipo que controla las llamadas externas u órdenes para transmitirlas a los Maceteros.

Usuario: en este proyecto, se considera que un usuario es cualquiera de los actuadores o componentes que actúan con el Sistema, como Servidor, Maceteros o actuadores externos.

Usuario interno: en este proyecto, se considera que un usuario interno es aquel que pertenezca al Sistema implementado en la intranet donde están los Maceteros, o parte interna del Servidor.

Usuario externo: en este proyecto, se considera que un usuario externo es aquel que pertenezca al Sistema implementado la parte externa del Servidor o Internet.

EEPROM: en este proyecto, la memoria empleada será la EEPROM interna de Arduino, por tanto se entenderá que cuando se refiera en el texto a EEPROM, se está hablando de cualquier memoria que se quiera utilizar.

WiFi: en este proyecto, se considera comunicación WiFi a la conexión de forma inalámbrica de cualquier Usuario del Sistema.

3.3. Funcionalidad

3.2.1 Equipo involucrado

Este apartado muestra la información de todas las personas involucradas en el proyecto así como una descripción del sistema en general.

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

Tabla 1. Miembro Axel Guzmán Godia

Nombre	Laia Ferrando Ferragud
Rol	Desarrolladora Aplicación móvil.
Categoría profesional	Estudiante
Responsabilidades	Diseño aplicación móvil
Información de contacto	laferfer@inf.upv.es

Tabla 2 Miembro Laia Ferrando Ferragud

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

Tabla 3. Miembro Israel Beltrán Arias

Nombre	Alejandro Delgado
Rol	Desarrollador comunicaciones
Categoría profesional	Estudiante
Responsabilidades	Implementación comunicaciones módulo-servidor
Información de contacto	-

Tabla 4. Miembro Alejandro Delgado

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

Tabla 5. Miembro José Luis Poza Luján.

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

Tabla 6. Miembro Juan Luis Posadas Yagüe.

Una vez detallada la información referente al equipo que compone el sistema, se pasa a detallar cómo está distribuido éste. El sistema está dividido en diferentes partes.
La primera es la orientada al montaje de una maceta con diferentes elementos hardware, además de la programación del microcontrolador Arduino para la automatización de su funcionamiento.
La segunda está centrada en la comunicación entre macetas (protocolo) así como en la comunicación desde una maceta al servidor.
La tercera es la dirigida a gestionar por un lado la base de datos en la que se guardará la información de sensores y actuadores, y por otra parte implementar el servidor que se encargará de comunicarse con la aplicación móvil y con las macetas mediante REST [10].
La cuarta, descrita en este documento, es la formada por el diseño y la implementación de una aplicación móvil para monitorizar los datos de las macetas y tener cierto control sobre ellas.

3.2.2 Casos de Uso

Los principales objetivos a alcanzar en este proyecto, tratan los siguientes puntos, desde el punto de vista del Usuario, siendo el usuario cada dispositivo Arduino que entra en el sistema:

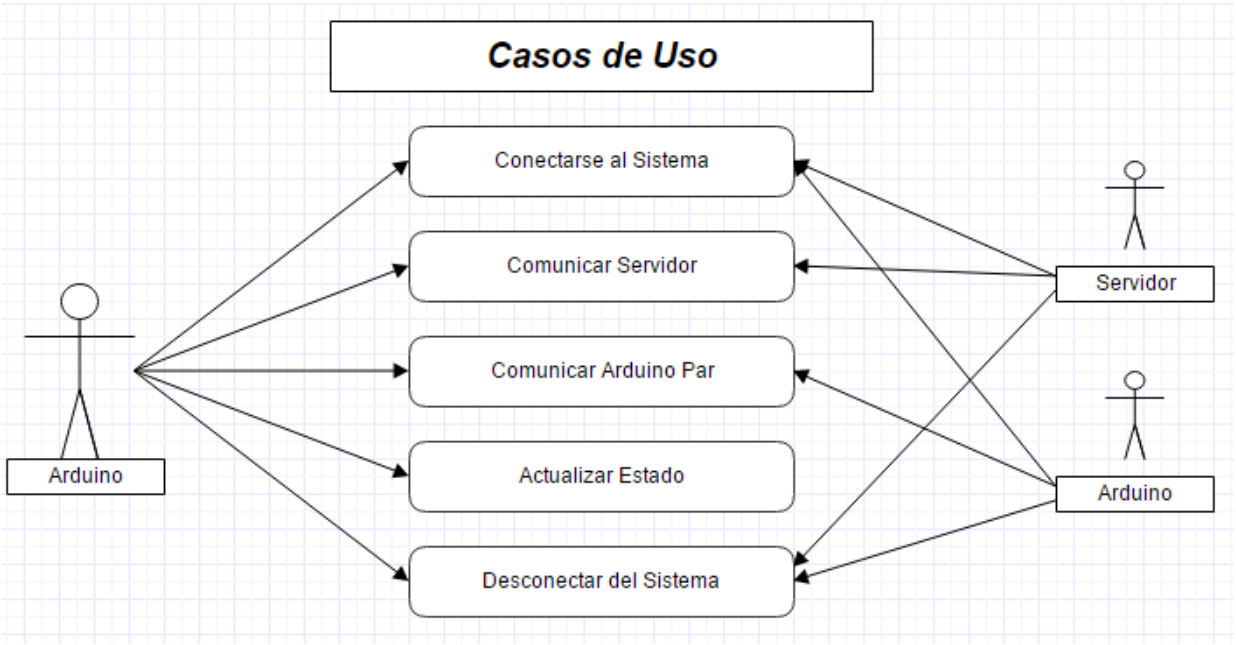

Ilustración 11. Casos de Uso

CU01 Conectarse al Sistema.

Este Caso de Uso sólo se puede realizar una vez por Usuario propio. Es decir, que un Arduino sólo podrá realizar Inicializarse en el Sistema una única vez mientras que no se haya desconectado previamente.

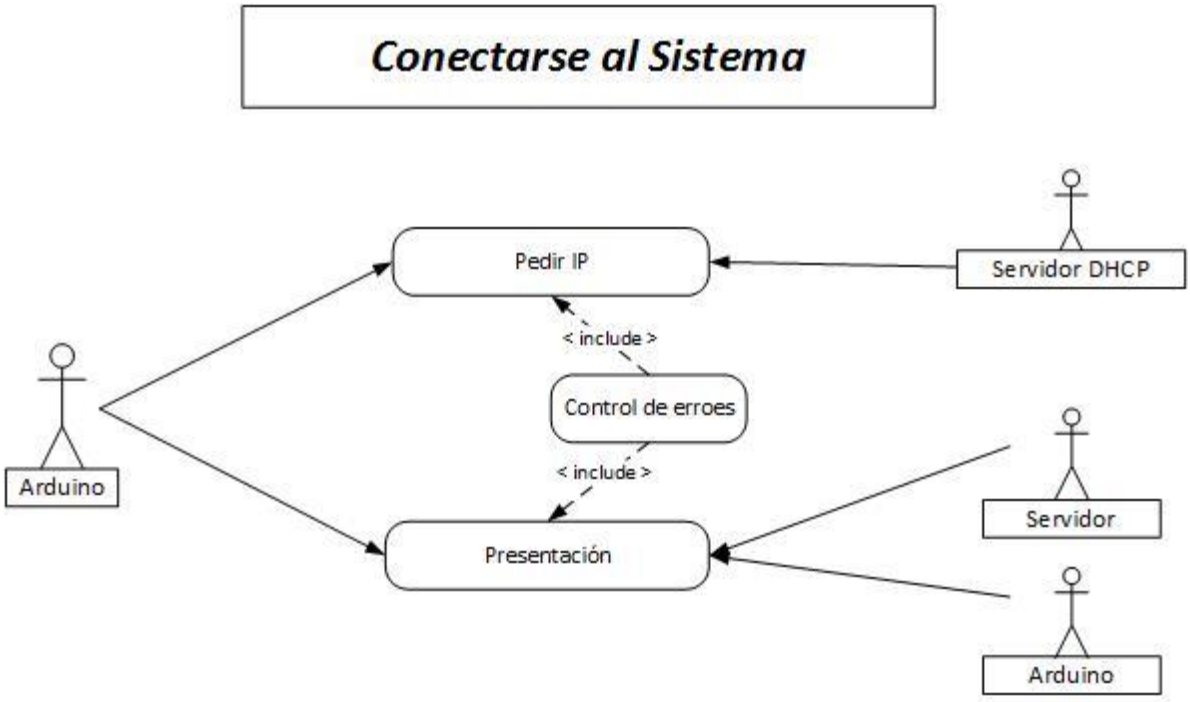

Ilustración 12. Funcionalidad Caso de Uso 1. (Propio Visio)

Como se puede apreciar en este Caso de Uso van a interactuar todos los Usuarios del Sistema. El primero será el propio Arduino que va interactuar con el Servidor DHCP (punto de acceso) para pedir que se le asigne una IP interna del Sistema. Y luego mediante una difusión comunicará su dirección y estado a todos los demás Usuarios.

Dentro de este caso se encontrarán dos funcionalidades básicas que serán: Pedir IP y Difusión, también se contará con una funcionalidad que será común a casi todos los Casos de Uso, el Control de Errores.

CU02: Comunicar Servidor.

Este Caso de Uso se realizará con frecuencia por el propio Arduino, ya que casi continuamente se van a estar enviando o recibiendo mensajes con el Servidor, interactuando y compartiendo así el conocimiento del Sistema.

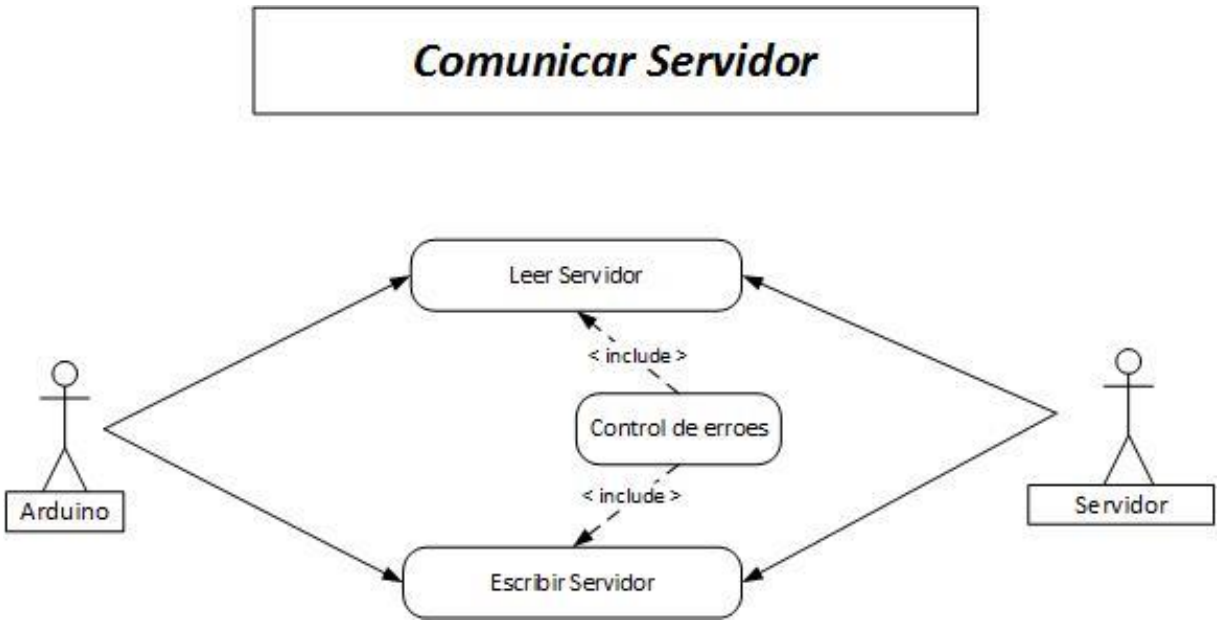

Ilustración 13. Funcionalidad Caso de Uso 2 (Comunicar Servidor)

Como se puede apreciar en este Caso de Uso van a interactuar dos Usuarios, uno será el propio Arduino que va a interactuar con el otro Usuario, el Servidor.

Dentro de este caso se encontrará dos funcionalidades básicas que serán: Leer Servidor y Escribir Servidor, también se contará con una funcionalidad que será común a casi todos los Casos de Uso, el Control de Errores.

CU03: Comunicar Arduino Par.

Este Caso de Uso, como el anterior, se realizará con frecuencia en el propio Arduino. Porque no solo se debe mantener informado al Servidor de los cambios producidos en el Sistema, sino que se debe difundir entre el resto de los Usuarios. Lo que permitirá interactuar y compartir información de una forma Distribuida.

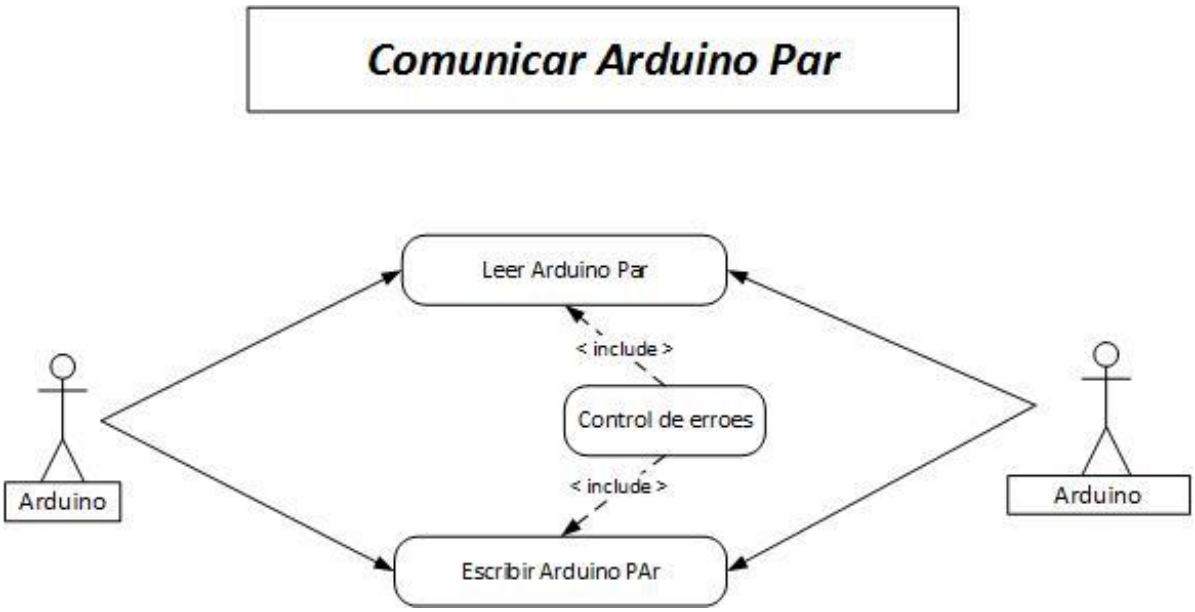

Ilustración 14. Funcionalidad Caso de Uso 3 (Comunicar Arduino Par)

En este Caso de Uso van a interactuar dos Usuarios, uno será el propio Arduino y el otro será un Arduino semejante al primero, Arduino Par.

Dentro de este caso se encontrará dos funcionalidades básicas que serán: Leer Arduino Par y Escribir Arduino Par, también se contará con una funcionalidad que será común a casi todos los Casos de Uso, el Control de Errores.

CU04: Actualizar estado

En este Caso de Uso, se muestran todas las funcionalidades que deben tener los Arduino para su gestión interna. Aquí por lo tanto se muestra la actualización del estado del sistema, para ello se hará uso de las variables globales o la memoria EEPROM [11] que contienen los Arduinos, que permite compartir variables de estado de todo el sistema de manera persistente. Las variables de estado se van a separar en variables de comunicación y variables de control.

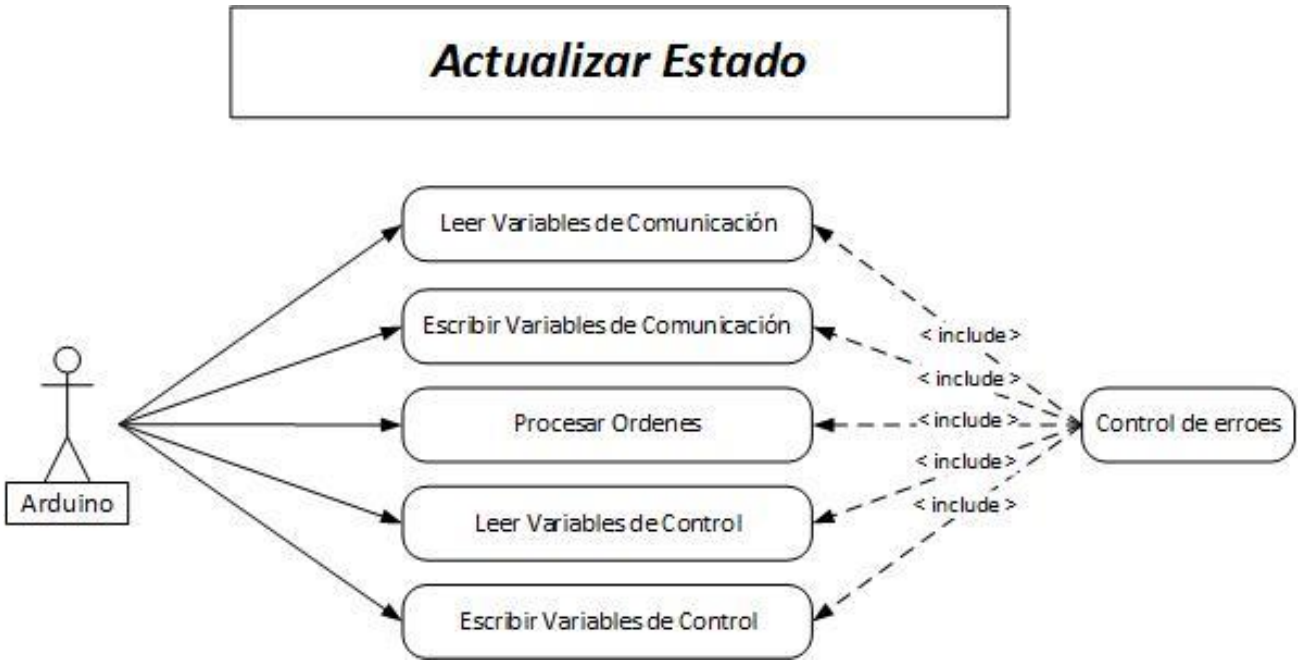

Ilustración 15. Funcionalidad de Caso de Uso 4 (Actualizar Estado)

En este Caso de Uso solo existe un usuario que será el propio Arduino, ya que no necesita que otros usuarios intervengan para realizar las siguientes cuatro funcionalidades: Leer variables de comunicación, Escribir variables de comunicación, Leer variables de Control y Escribir variables de Control. Además se contará con una funcionalidad de Control de Errores que servirá para comprobar que toda la información se haya leído o escrito correctamente, sin pérdidas.

También existe una funcionalidad que será Procesar Ordenes, que se encarga de decir qué es lo que va a ejecutar el Arduino dependiendo de las órdenes recibidas del Servidor o de los demás Arduinos, representadas como variables que previamente se han almacenado en el propio Arduino. Con toda la información que precisa puede decidir qué acción realizará el Arduino.

CU05: Desconectarse del Sistema

Este Caso de Uso como el CU01 solo se puede realizar una vez por Usuario. Ya que después de ejecutar este CU, el Arduino se encontrará fuera del Sistema, y por lo tanto será libre de volver a conectarse o de permanecer inactivo.

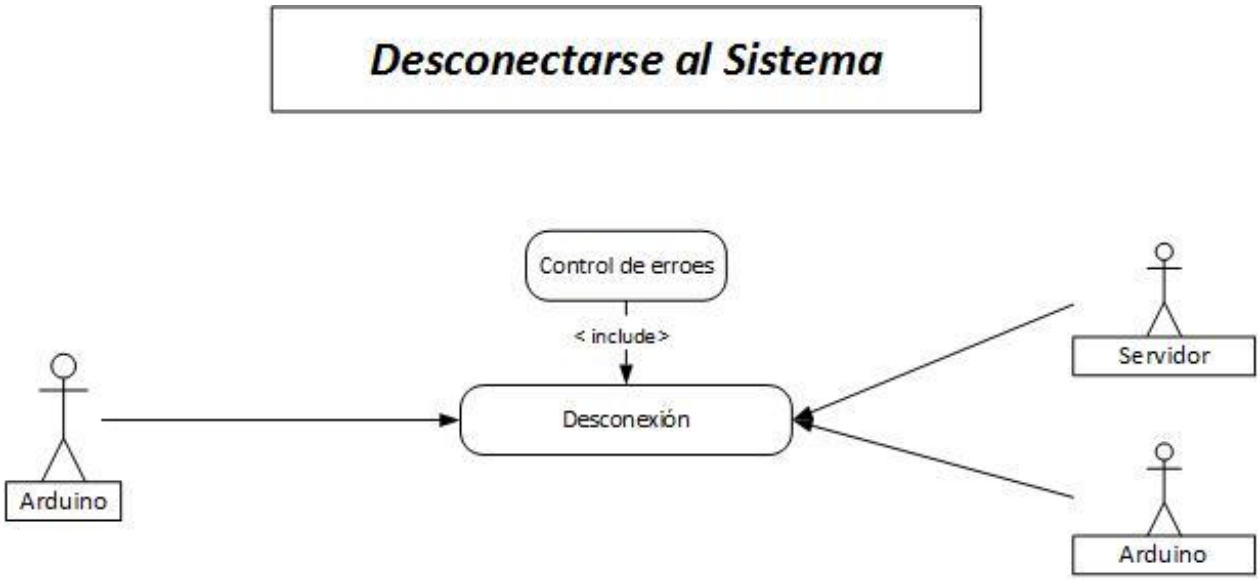

Ilustración 16. Funcionalidad Caso de Uso 5 (Desconectarse del Sistema)

Como se puede apreciar en este Caso de Uso van a interactuar todos los Usuarios del Sistema. El Arduino que va ainteractuar con el Servidor y el resto de Usuarios del Sistema mediante una difusión que comunica su desconexión.

Dentro de este caso se encontrará una única funcionalidad básica que será: Difusión, que hace referencia a la correcta forma de salir del Sistema. No cuenta como tal el hecho de que se haya apagado el dispositivo o haya sido desconectado abruptamente. También se contará con una funcionalidad que será común a casi todos los Casos de Uso, el Control de Errores.

3.2.3 Tablas de Funcionalidad

Aquí se va a indicar mediante unas tablas de funcionalidad todos los requisitos que se han de cumplir en el Proyecto y que contemplan todos los Casos de Uso vistos anteriormente.

Número de requisito	**RE01**
Nombre de requisito	Conectarse al Sistema
Tipo	☐ Requisito de Usuario ☒ Requisito del Sistema ☐ Requisito Funcional ☐ Restricción
Descripción del requisito	Cada Usuario se deberá inicializar en el Sistema, Solicitando una Dirección IP al Servidor DHCP, este punto de acceso le dará una IP que será parte de su identificación ante el resto de los usuarios. Luego realizará una difusión, que informe a todos los elementos del Sistema su dirección, su identificador y estado actual. Incluyendo todas sus variables de Control y comunicación. Luego permanecerá en espera.
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Tabla 7. Requisitos del Sistema (conexión)

Número de requisito	**RE02**
Nombre de requisito	Comunicar Servidor
Tipo	☐ Requisito de Usuario ☐ Requisito del Sistema ☒ Requisito Funcional ☐ Restricción
Descripción del requisito	Cada Usuario se debe poder comunicar con el Servidor, para poder intercambiar información y actualizar así la información que tenga el Servidor sobre el estado del propio Usuario. También se tiene que poder solicitar datos al Servidor que deberá responder si puede o no trasmitir. En caso afirmativo le devolverá la información solicitada sobre el propio usuario o los distintos elementos vecinos.
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Tabla 8. Requisitos del Sistema (comunicación con el servidor)

Número de requisito	**RE03**
Nombre de requisito	Comunicar Arduino Par
Tipo	☐ Requisito de Usuario ☐ Requisito del Sistema ☒ Requisito Funcional ☐ Restricción
Descripción del requisito	Los Usuario se deben poder comunicar entre el resto de los Usuarios, para poder intercambiar información con ellos y actualizar así la información que tiene cada uno sobre el resto del Sistema. De forma que todos tengan conocimiento del estado completo del Sistema. Para ello se deberá solicitar al Servidor la dirección del Arduino al que se desea acceder, para luego poder iniciar la comunicación con el Usuario deseado.
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Tabla 9. Requisitos del Sistema (comunicación entre pares)

Número de requisito	**RE04**
Nombre de requisito	Actualizar Estado
Tipo	☐ Requisito de Usuario ☐ Requisito del Sistema ☒ Requisito Funcional ☐ Restricción
Descripción del requisito	Los Usuarios se deberán actualizar cada vez que sea preciso, y no solo cuando la información recibida tenga referencia con el propio Arduino, si no que con cada cambio que haya en el Sistema se debe actualizar al estado actual. Se deben actualizar las variables de comunicación y las de control de todos los que se tenga información. Las actualizaciones de estado se podrán realizar mediante un envío cada cierto tiempo establecido o cuando se produzca algún cambio de estado.
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Tabla 10. Requisitos del Sistema (actualización del estado)

Número de requisito	**RE05**
Nombre de requisito	Desconectarse del Sistema
Tipo	☐ Requisito de Usuario ☒ Requisito del Sistema ☐ Requisito Funcional ☐ Restricción
Descripción del requisito	Todos los Usuarios se pueden desconectar del Sistema mandando un aviso a todos los Usuarios del Sistema. Se realizará una difusión, que informe a todos los elementos del Sistema que su dirección vuelve a quedar libre y que todas las variables suyas vuelven a estar vacías.
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/ Opcional

Tabla 11. Requisitos del Sistema (desconexión)

3.4. Conclusiones

En este capítulo se ha realizado una especificación de requisitos de manera formal utilizando el estándar IEE 830. Los diferentes requisitos establecidos por el sistema se han determinado a partir de los diagramas de Casos de Uso realizados, que a la vez ha sido elaborado a partir de las características definidas en el capítulo anterior.

Una vez realizada la especificación formal de requisitos, y basándose en los requisitos anteriores, se procederá al diseño del sistema, aspecto que se tratará en el siguiente capítulo.

4. Diseño de Ardu-Red

4.1. Introducción

A partir de la especificación de requisitos del capítulo anterior se ha realizado el diseño del sistema utilizando descripciones técnicas estandarizadas como UML [7] o similares. En este capítulo se abordará el diseño el sistema.

A continuación se muestra una especificación conceptual, que muestra a grandes rasgos como se va a componer el Sistema. Luego se observará que para implementar todos los requisitos se va a aplicar una arquitectura de 3 capas. Se explicará cómo se va a desarrollar en cada una de las capas los requisitos que se han especificado, indicando a que funcionalidades hacen referencia y de que requisito ha salido cada funcionalidad. También se dará un esbozo de cómo se propone implementar estas funcionalidades y de que estructura va a tener el producto propuesto, utilizando diferentes imágenes y diagramas con el fin de facilitar la comprensión.

4.2. Especificación conceptual:

En este apartado se va a explicar de forma genérica e informal, como se organiza el sistema. Se va a contar con tres tipos de control, entre los Usuarios Finales, las personas del mundo real, y los Actuadores Finales, los sensores y actuadores de las macetas.

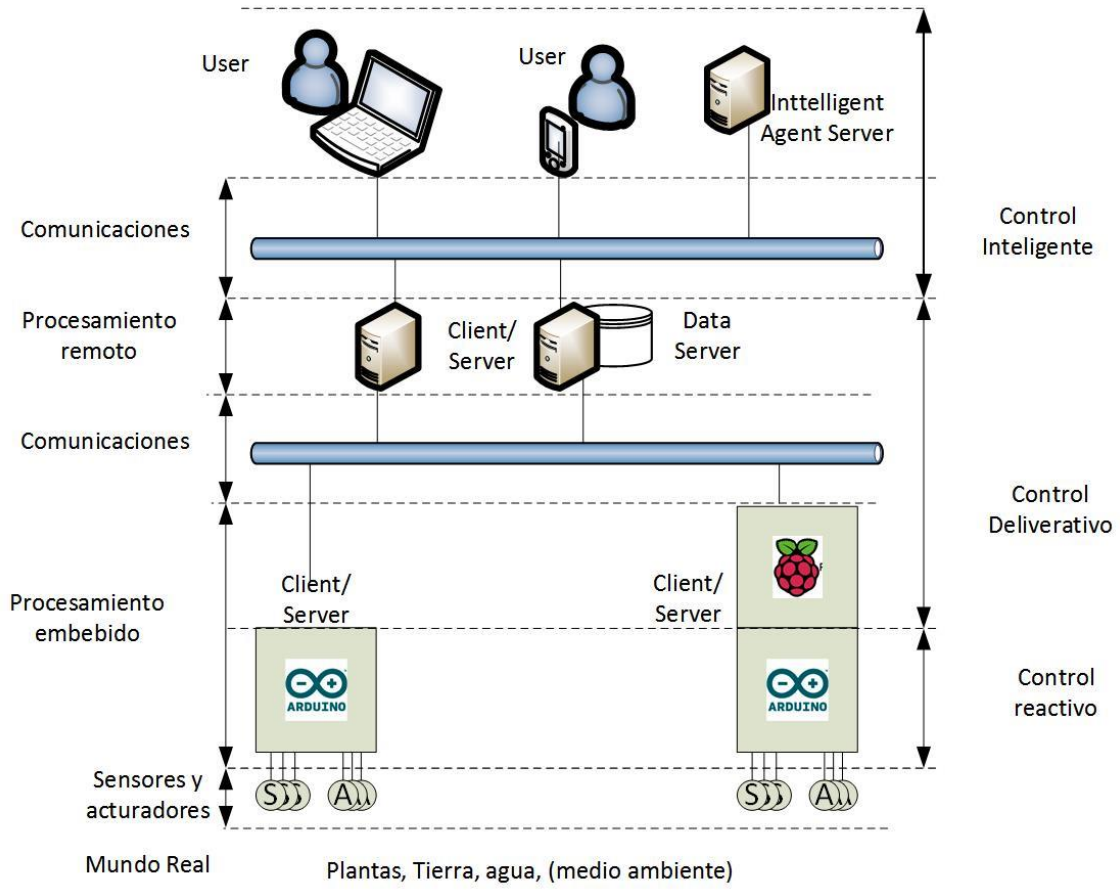

Ilustración 17 Diagrama conceptual

El Control Inteligente será el que se encargue de introducir órdenes a la base de datos del Sistema y de solicitar medidas, estado de sensores o lo que se disponga, o permita en la aplicación. Todo esto accediendo a los datos de que se han guardado desde el Sistema en la Base de Datos.

El Control Deliberativo, que es el predominante en este proyecto, y al que más importancia se le dará, se encargará de comunicar lo que desea cada parte y reflejarlo en la Base de Datos. Esto es plasmar las órdenes que se solicitan desde las aplicaciones de usuarios, en modificaciones de variables de la placa solicitada. También corresponde a este control mantener la información de todo el Sistema continuamente actualizado y una gestión de todos los protocolos de comunicación que tenga la Base de Datos.

Por último, el Control Reactivo es el que se encarga de gestionar las Placas Arduino, con los Sensores y Actuadores a los que está conectado, manejando su lógica mediante variables internas. Lo que realizará las acciones que se necesiten de los actuadores, o se informará de las medidas que se hayan solicitado de los Sensores (humedad, temperatura, etc.)

4.3. Especificación formal

El modelo que se va a utilizar es el conocido modelo de tres capas: Persistencia, Negocio y Presentación. En este caso casi todo el peso del sistema se encuentra en la capa de Negocio, y una parte en la capa de Persistencia.

4.3.1. Capa persistencia

En la Capa de persistencia se ncunetra una necesidad vital para poder cumplir los objetivos de comunicar y mantener la información de todo el sistema en cada uno de los usuarios. En este caso se va hacer uso de variables globales y de las memorias EEPROM que se encuentran en las placas Arduino. En ellas se mantendrán las variables de control y las de comunicación preservada de las caídas de tensión o desconexiones espontaneas.

Este modelo de persistencia es un modelo de Red, o Malla, que permite una mayor rapidez de difusión, y un almacenamiento de datos y recuperación de fallos muy favorables, dado que toda la información del sistema se almacena por igual en cada uno de los usuarios del sistema.

Como se puede apreciar en los diferentes modelos, la Capa de Persistencia va a residir en la Base de Datos que controlará el Servidor, y en las pequeñas memorias internas con las que cuentan las Placas Arduino (EEPROM).

Más concretamente, la memoria EEPROM, interna a cada Arduino, es de 1KB, por lo que se tendá que segmentar la información relativa a cada Arduino. Contando para cada uno 32 bytes en los que se informarán de todas las variables de control y comunicaciones. Habrá 24 bytes para las variables de control, y habrá 12 para las variables de comunicación.

De forma que cada Arduino tenga su propia información desde el byte 0 al 31, y los Arduinos vecinos tendrán las siguientes fragmentos de memoria (32-63, 64-95,etc).

Posición EEPROM	Variable global
IP (0-3)	Red. Ard1.ip.ip1 Red. Ard1.ip.ip1 Red. Ard1.ip.ip1 Red. Ard1.ip.ip1
ID (4)	Red.Ard1.id
Estado (5-7)	Red. Ard1.est.EstMov Red. Ard1.est.EstDep Red. Ard1.est.EstCom
UEr (8)	Red. Ard1.UEr
Md (9)	Red. Ard1.Md
Mi (10)	Red. Ard1.Mi
Bh (11)	Red. Ard1.Bh
Ta (12)	Red. Ard1.Ta

Ha (13)	Red. Ard1.Ha
Hs (14)	Red. Ard1.Hs
La (15-16)	Red. Ard1.La
D (17-18)	Red. Ard1.D
Nd (19)	Red. Ard1.Nd
SS (20)	Red. Ard1.Ss
SI (21)	Red. Ard1.Si

Tabla 12.Organización de la Estructura de la memoria EEPROM (propio)

- UEr: Indica el código del último Error (1) int
- Md: Motor derecho (1) int [] v
- Mi: Motor izquierdo (1) int [] v
- Bh: Bomba hidráulica (1) int [0,1] l/s
- Ta: Sensor de Temperatura (1) int [] ºC
- Ha: Sensor de Humedad del aire (1) int [] %
- Hs: Sensor de Humedad del suelo (1) int [] %
- La: Sensor de Luz (2) int [] lumens
- D: Sensores de Distancia de Ultrasonido (2) int [3, 300] cm
- Nd: Nivel depósito [0,50,100] %
- SS: Sensor de nivel Superior, donde mide el nivel de agua si se llena.
- SI: Sensor de nivel Inferior, donde mide el nivel de agua si se vacía.

Como se puede apreciar en la tabla, la memoria EEPROM guarda una relación con las variables globales, de forma que sea extremadamente sencillo salvar y recuperar la información de cada Arduino. Por lo que, se va a utilizar esta estructura idéntica para cada Arduino, variando su ruta en el struct para facilitar la interacción entre ellos y para una mejor gestión de la información de los Vecinos, esencial a la hora de desarrollar un sistema Distribuido.

Ejemplo: Red. Ard1.Ha // Red. Ard2.Ha. Accediendo al mismo sensor de diferentes Arduinos. ESTADOS.

Los posible estados en los que se puede encontrar los Arduinos están separado en tres partes, pero en este proyecto los interesantes son los de comunicación. A continuación se muestra la tabla de estados:

Estado de movimiento	Estado de deposito	Estado de comunicaciones
00: Inicial	00:Inicial	00:Inicial
01:Stop	01:Vacio	01:No IP & No conectado
02:Adelante	02:Normal	02: IP & No conectado
03:Atrás	03:Lleno	03: IP & conectado(espera)
04:Girar a la izquierda	99:Error	04:Recibiendo Petición
05:Girar a la derecha		05:Enviando Mensaje
99:Error		99:Error

Tabla 13. Posibles Estados del Sistema (Elaboración del proyecto)

4.3.2. Capa de negocio

4.3.2.1. Diagrama de flujo

Aquí se puede observar el comportamiento que han de tener los Arduinos, de acuerdo al protocolo que se ha diseñado para este sistema. Las acciones que debe implementar el Arduino se pueden dividir en las acciones únicas y continuas. Las únicas serán las acciones que solo se realizarán una única vez por elemento del Sistema. Estas son:

1. Pedir IP
2. Leer EPPROM
3. Broadcart

Las acciones continuas serán aquellas que se ejecutarán perpetuamente dentro de un bucle mientras dure su funcionamiento. La secuencia que deben seguir es la siguiente:

1. Leer Petición
2. Procesar Petición
3. Escribir EPPROM
4. Control
5. Leer EPPROM
6. Multicast
7. Difusión

En este esquema de actividad se representa la actividad que debe realizar cada Arduino, y cómo reaccionará en el tiempo a las peticiones que le soliciten.

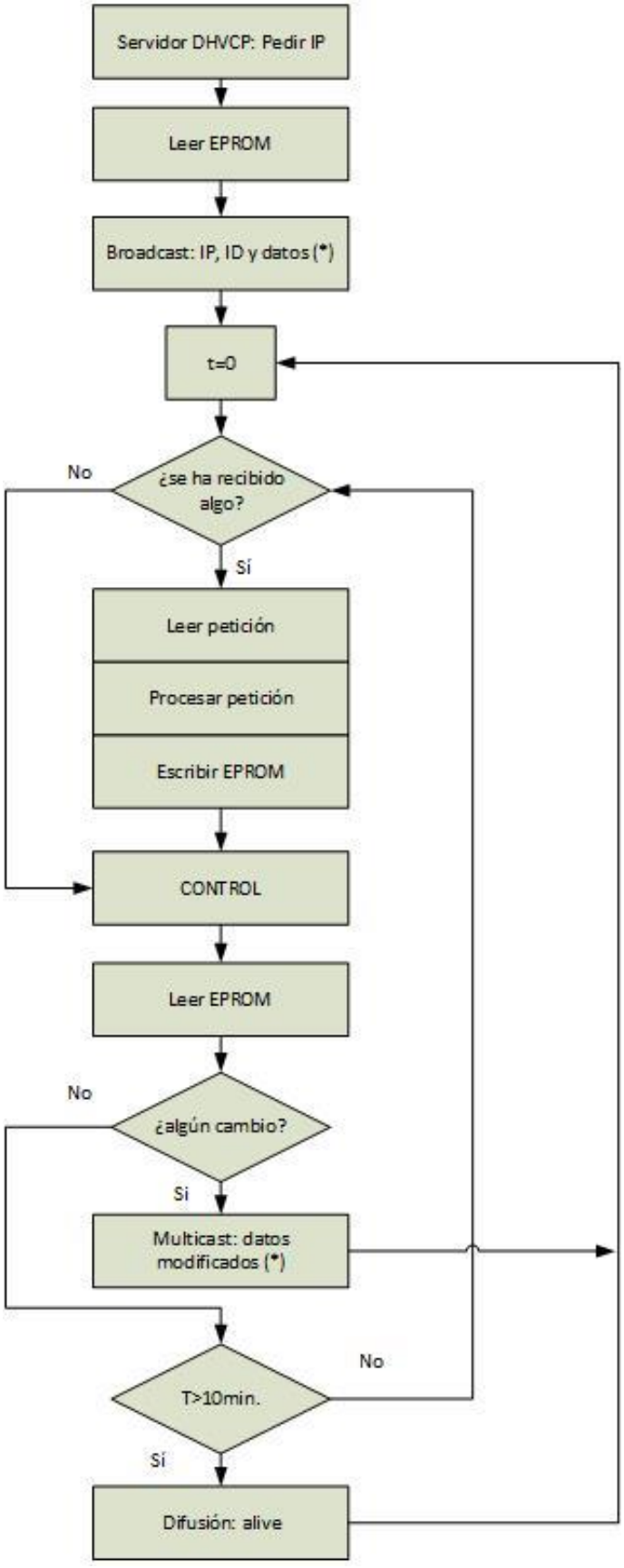

Ilustración 18 Diagrama de Actividad (Visio, Jose Luis Poza Lujan)

Como se puede apreciar en el diagrama, después de haberse identificado en él, se permanecerá en un bucle con un contador que será el tiempo máximo del Arduino para enviar un mensaje de que sigue "vivo". Internamente también tendrá un bucle de espera para recibir cualquier mensaje que le llegue. Y tanto si procesa algún mensaje como si no se ejecutará la parte referente al control. Luego habrá una lectura de lo que haya podido cambiar o escribir la parte de control. Si se ha dado algún cambio, se difunde a todos los componentes del Sistema. Si no ha habido cambio aumenta el contador mencionado antes.

A continuación se va a definir cada una de las funcionalidades en funciones específicas, partiendo de los diagramas de uso vistos en el capítulo 3.

4.3.2.2. UML

Ahora se va a definir la estructura estática de los usuarios y componentes que componen el Diseño de este proyecto. En el siguiente diagrama de Clases se observan los Usuarios que interactúan con los Maceteros; los Usuarios externos, el Servidor y otros posibles Maceteros Vecinos.

Los componentes que tiene cada Macetero son: uno o más Arduinos, y cada Arduino tiene una memoria y conectado el ESP8266.

Los Maceteros pueden conectarse al Servidor, o con otros Maceteros. También pueden enviar y recibir peticiones, o mensajes, entre ellos. El Servidor DHCP se especifica como un servidor concreto porque es el encargado de asignar al Macetero una IP dentro del Sistema.

Un usuario Externo puede accionar varios Maceteros, y puede conectarse a uno o más Servidores, para llevar a cabo las acciones disponibles en la app o en la web de los otros proyectos integrados.

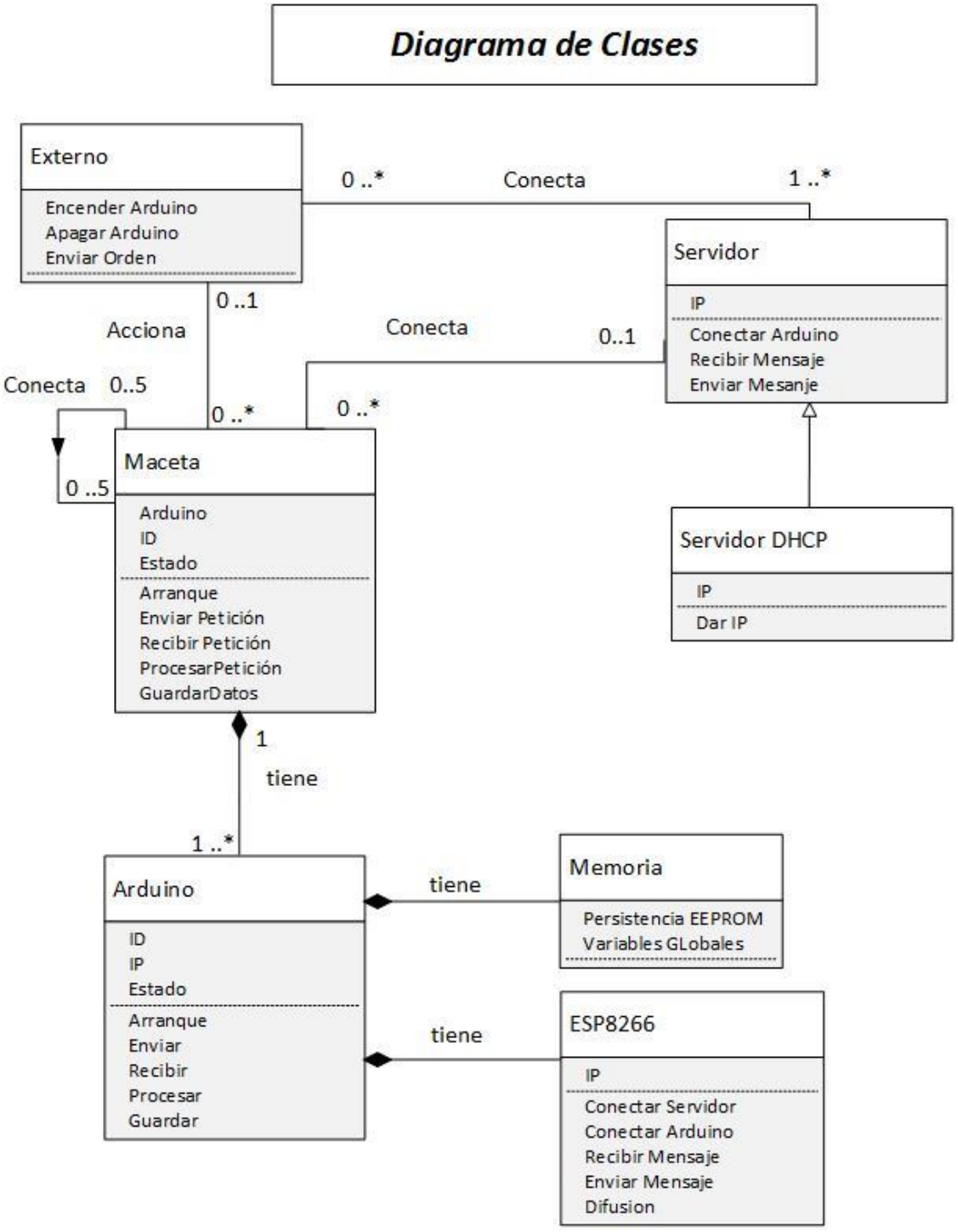

Ilustración 19. Diagrama de Clases (elaboración propia)

Como se puede apreciar en este diagrama, hay una restricción en el Macetero, ya que el Arduino que lo compone solo es capaz de mantener 5 conexiones, por lo que no se podrá superar 5 comunicaciones a la vez.

4.3.2.3. Dinámicos: secuencia y actividad

FU01: Pedir IP

La primera función que debe realizar los Arduino es la inicialización en el Sistema. Lo primero será que un Usuario Externo al Sistema encienda el Arduino, luego éste necesitará una ubicación a la que los demás elementos puedan acceder para pedirle información. Para ello se va a pedir al Servidor DHCP, que actuará como punto de acceso que le asigne una dirección IP interna, única para cada Arduino que se conecte al Sistema, con la que se va identificar desde ahora con el resto de elementos. Después de tener esta IP, leerá todos los datos que tenga de sus sensores y estados de la memoria EEPROM, e informará al Servidor y el resto de los Usuarios del Sistema de todos sus datos.

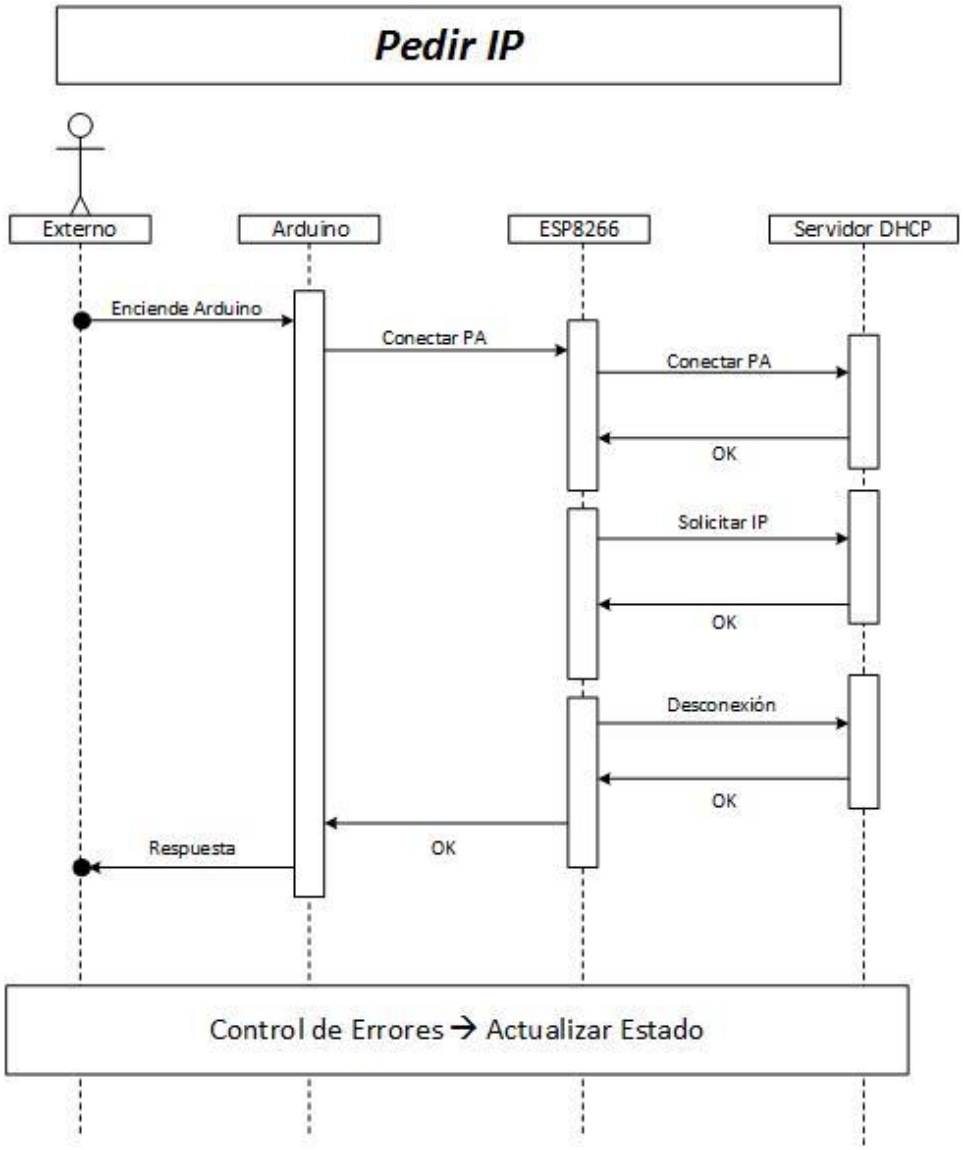

Ilustración 20. Diagrama de Secuencia 1 (Pedir IP)

Como se puede apreciar en el Diagrama, intervienen cuatro usuarios. El usuario externo será quien encienda el Arduino físicamente. El Arduino le comunicara a su chip ESP8266 que se quiere conectar al punto de acceso para ejecutar la funciona de Pedir IP, este se conectará al Servidor DHCP o PA(Punto de Acceso) y le pedirá una IP que le servirá de identificador dentro del Sistema. Por último se desconecta y acaba esta función.

Para no sobrecargar el Diagrama de Secuencia 1, se han omitido todos los time_out que tiene cada una de las llamadas desde el ESP8266 al Servidor DHCP.

Aparte de estas funciones internas que se tendrán que ejecutar se cuenta con un Control de Errores que informará de cualquier posible percance que haya ocurrido mientras se ejecutaba cada una de las funciones. Esta función se encontrará en todas las demás Funcionalidades. Si ha ocurrido algún error se procederá a enviar un mensaje con la variable que contiene el último error ocurrido. Luego de ejecutar el Control de Errores, se procederá a Actualiza el Estado del propio Arduino. Guardando en la EEPROM, o en las variables globales toda la información de las variables de Control y Comunicaciones. Las funciones que debe realizar para cumplir este diagrama son:

Conectar Servidor: Ésta función lo que hace es solicitar conexión al Servidor DHCP, para poder solicitar una dirección IP al mismo.

Código	CFUN01	Nombre	Conectar servidor DHCP (FU01)
Input			Output
+ IP Servidor			+ Resultado de la conexión
Procesamiento			
Se conecta con el servidor (IP) –> función propia que utiliza las librerías “SoftwareSerial.h” y “WiFi.h” Si servidor Ok → Se ha conectado con el Servidor. Se pasa a la siguiente función (Solicitar IP) Si servidor Time Out → No se ha conectado con el Servidor. Se acaba la función.			
Errores			
+ Servidor no localizado: ERR001 + Servidor desconectado durante la ejecución de la función: ERR0002			
Casos de Uso que cubre			Requisitos que cubre
CU01			RE01

Solicitar IP: Ésta función solicita al Servidor DHCP, una IP para identificarse dentro del Sistema.

Código	CFUN02	Nombre	Solicitar IP (FU01)
Input			Output
+ Mensaje de Solicitud IP			+ Dirección IP interna asignada
Procesamiento			
Se solicta una asignación de IP al servidor(Mensaje) –> función propia que utiliza las			

librerías “SoftwareSerial.h” y “WiFi.h” Si servidor OK → Se ha obtenido una IP que se va a guardar como propia en las variables globales. Luego se pasa a la siguiente función (Desconexión) Si servidor Time Out → No se ha conectado con el Servidor. Se acaba la función.
Errores
+ Servidor no ha podido asignar dirección IP: ERR003 + Servidor desconectado durante la ejecución de la función: ERR0004

Casos de Uso que cubre	Requisitos que cubre
CU01	RE01

Desconexión: Ésta función se desconecta de forma correcta del Servidor DHCP.

Código	CFUN03	Nombre	Desconexión (FU01)
Input			Output
+ Mensaje de Desconexión			+ Desconexión aceptada
Procesamiento			
Se desconecta del Servidor esperando la aceptación del mismo –> función propia, y se hace uso de las librerías “SoftwareSerial.h” y “WiFi.h” Si servidor OK → Se ha desconectado correctamente. Si servidor Time Out → No se ha recibido la desconexión, se escribe el error en la variable de último error.			
Errores			
+ Servidor no responde: ERR005 + Servidor desconectado durante la ejecución de la función: ERR0006			

Casos de Uso que cubre	Requisitos que cubre
CU01	RE01

Control de Errores: Ésta función se encarga de detectar y notificar cualquier error que se haya producido mientras se ejecutaban las otras funciones.

Código	CFUN04	Nombre	Control de Errores (FU01)
Input			Output
+ Errores durante el procedimiento.			+ Mensaje de los Errores ocurridos durante el procedimiento.
Procesamiento			
Se comprueban los errores que se hayan mostrado durante el procedimiento de esta función. Se crea un mensaje con el valor de la variable “último error” y se envía al Sistema. Funciones propias y captura de los Errores en los procedimientos anteriores.			
Errores			
+ Error no controlado: ERR007			

Casos de Uso que cubre	Requisitos que cubre
CU01, CU02, CU03, CU04	RE01,RE02, RE03,RE04

FU02: Presentación en el Sistema

La segunda función con la que cuenta los Arduino es la de su Presentación al resto del Sistema. Para ello, necesita saber su identificación y toda la información relativa a sí mismo, que se encuentra en la primera sección de la EEPROM si se usa o en las variables globales. Por lo tanto deberá leer sus datos y realizar una difusión mediante un Broadcast de toda su información a todos los Usuarios del Sistema. De esta forma, todos guardan su información y será totalmente visible y accesible desde cualquier lugar del Sistema y por cualquier usuario del mismo.

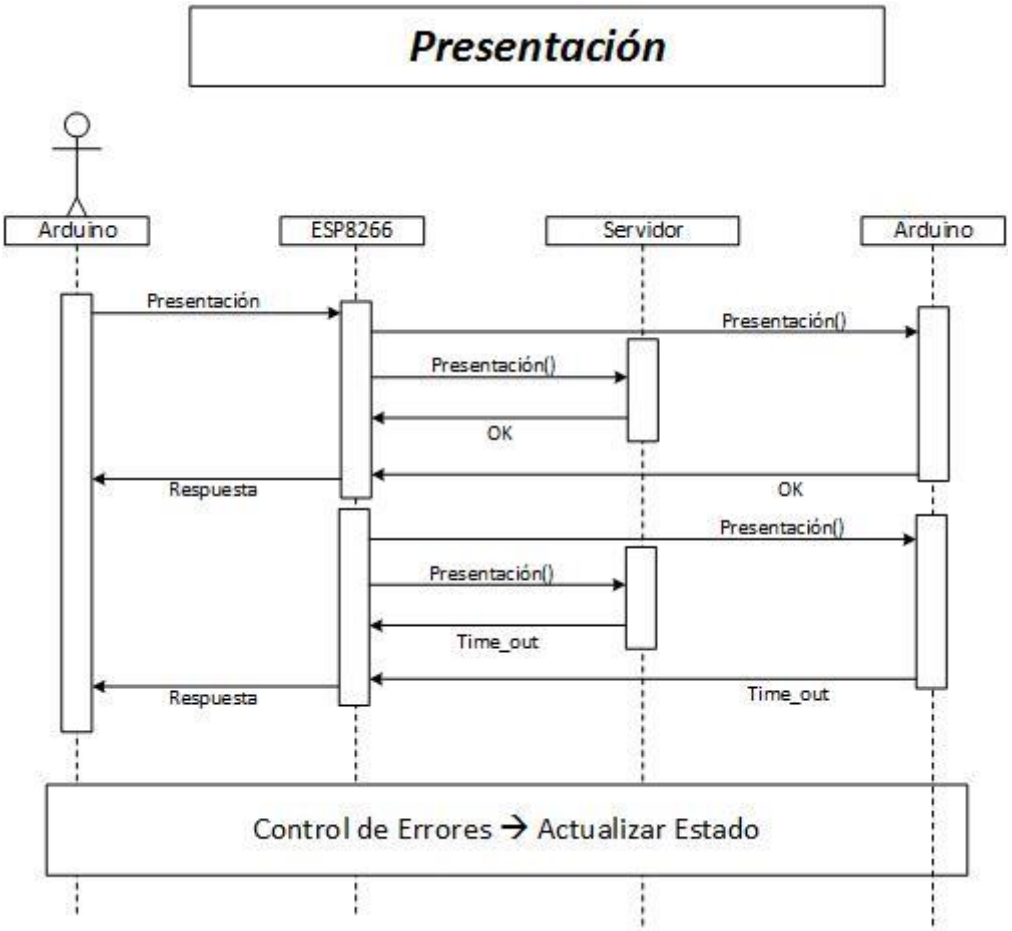

Ilustración 21. Diagrama de Secuencia 2 (Identificación en el Sistema)

Como se puede apreciar en el Diagrama, interviene cuatro usuarios. El primero será el usuario Arduino que le comunicará a su chip ESP8266 el mensaje que debe enviar a todo el Sistema para su Identificación en el Sistema. Luego el chip ESP8266 enviará una difusión con la información del Arduino al Servidor y al resto de Arduinos del Sistema.

También se contará con el Control de Errores para notificar cualquier posible error de ejecución de las otras funciones. Las funciones que debe realizar para cumplir este diagrama son:

Presentación en el Sistema: Esta función envía a todos los Usuarios del Sistema toda la información del propio Arduino.

Código	CFUN05	Nombre	Presentación en el Sistema (FU02)
Input			Output
+ Mensaje de Presentación			+ Resultado de la difusión
Procesamiento			
Se realiza una difusión a todos los elementos del Sistema informando de todos las datos internos del Arduino (Mensaje) –> función propia que utiliza la librería "WiFiUDP.h". Si Usuarios OK → Se ha recibido la información del Arduino. Si Usuarios Time Out → No se ha recibido la información del Arduino. Se escribe el error en la variable de último error.			
Errores			
+ Servidor no responde: ERR008 + Arduino no responde: ERR0009			
Casos de Uso que cubre			Requisitos que cubre
CU01			RE01

Control de Errores: Esta función se encarga de detectar y notificar cualquier error que se haya producido mientras se ejecutaba la difusión.

Código	CFUN06	Nombre	Control de Errores (FU02)
Input			Output
+ Errores durante el procedimiento.			+ Mensaje de los Errores ocurridos durante el procedimiento.
Procesamiento			
Se comprueban los errores que se hayan mostrado durante el procedimiento de esta función. Se crea un mensaje con el valor de la variable "último error" y se envía al Sistema. Funciones propias y captura de los Errores en los procedimientos anteriores.			
Errores			
+ Error no controlado: ERR007			
Casos de Uso que cubre			Requisitos que cubre
CU01, CU02, CU03, CU04			RE01,RE02, RE03,RE04

FU03: Leer Petición

La tercera función que debe poder realizar el Arduino es la capacidad de poder leer las instrucciones u órdenes que envía el servidor u otro usuario del Sistema por medio de peticiones al chip ESP8266. Así, se podrá interactuar con usuarios externos al Sistema y permitirá interactuar a los Usuarios internos demostrando una independencia al mundo exterior. Estas instrucciones e información se deberán guardar en variables globales y en la memoria EEPROM, si se utiliza, de los Arduino en forma de variables de control que maneja el Arduino. Con el añadido si se utiliza la EEPROM de que se mantenga la información pese a una desconexión con el sistema o una caída de tensión del elemento.

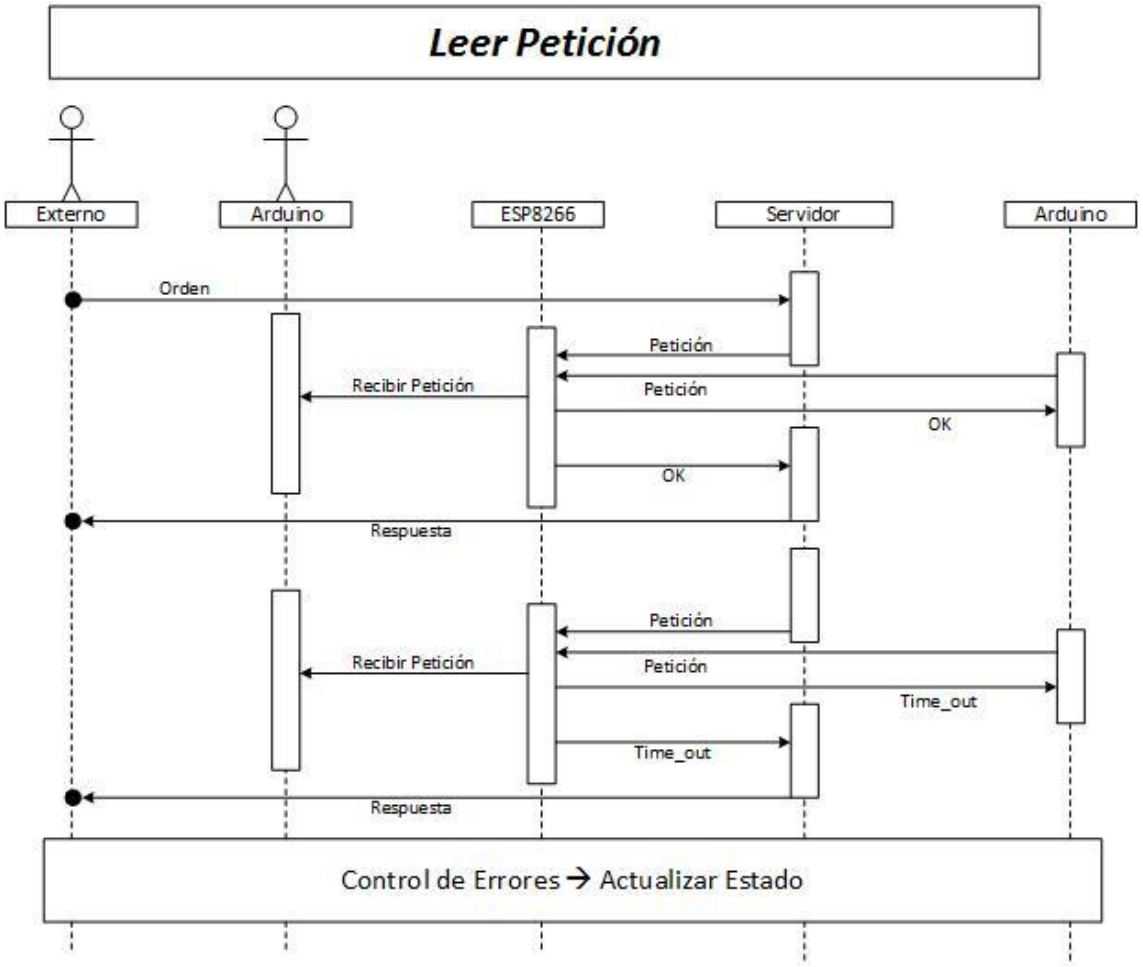

Ilustración 22. Diagrama de Secuencia 3. Leer Petición (Visio Propio)

Como se puede apreciar en el Diagrama, interviene cinco usuarios. El primero será el usuario externo que dará una orden al Servidor, o un Arduino del Sistema que envía alguna información. Esto llega, por medio del chip ESP8266, al Arduino especificado que recibe la orden solicitada por medio de un mensaje o petición. Después el Arduino contesta si ha recibido la petición, por medio del chip ESP8266,a quién lo haya enviado. Por último el Servidor contesta al usuario externo que había enviado la orden al Arduino.

Se cuenta también con el Control de Errores que notificará cualquier posible error de ejecución de las otras funciones. Guardando y enviando el último error ocurrido, y actualizando después se estado en caso que sea necesario. Las funciones que debe realizar para cumplir esta función son:

Recibir Petición: Esta función recibe los datos relativos al Arduino del Servidor.

Código	CFUN07	Nombre	Recibir Petición (FU03)
Input			Output
+ Mensaje de Petición/Orden			+ Resultado de la recepción
Procesamiento			

Se recibe la Petición u Orden mediante un mensaje (Mensaje) –> función propia que utiliza la librería "SoftwareSerial.h". Si Mensaje Recibido → OK, Se ha recibido la información del Arduino. Se pasa a la siguiente Función interna(Procesar Petición) Si Time Out → No se ha recibido la información del Arduino. Se escribe el error en la variable de último error.
Errores
+ Datos no esperados: ERR010 + Datos nulos: ERR011 + Time_out alcanzado: ERR012 + Servidor desconectado durante la ejecución de la función: ERR0002

Casos de Uso que cubre	Requisitos que cubre
CU02, CU03, CU04	RE02, RE03,RE04

Control de errores: Esta función se encarga de detectar y notificar cualquier error que se haya producido mientras se ejecutaba las funciones anteriores.

Código	CFUN08	Nombre	Control de Errores (FU03)
Input			Output
+ Errores durante el procedimiento.			+ Mensaje de los Errores ocurridos durante el procedimiento.
Procesamiento			
Se comprueban los errores que se hayan mostrado durante el procedimiento de esta función. Se crea un mensaje con el valor de la variable "último error" y se envía al Sistema. Funciones propias y captura de los Errores en los procedimientos anteriores.			
Errores			
+ Error no controlado: ERR007			

Casos de Uso que cubre	Requisitos que cubre
CU01, CU02, CU03, CU04	RE01,RE02, RE03,RE04

FU04: Procesar Petición.

La cuarta función que debe cumplir el Arduino es la de Procesar las Peticiones u órdenes que ha recibido anteriormente. Para ello se descompone el mensaje recibido y se trata para obtener la información relevante del mensaje. Previamente ha sido acordada la forma en la que se envían datos; tanto en el Servidor como por el resto de Usuarios del Sistema. Las funciones que debe realizar para cumplir esta función son:

Procesar Petición: Ésta función procesa el mensaje y extrae los datos importantes recibidos en la Petición de otro Usuario del Sistema.

Código	CFUN09	Nombre	Procesar Petición (FU04)
Input			Output
+ Mensaje de Petición/Orden			+ Resultado del procesamiento
Procesamiento			

Se trata el mensaje de la Petición u Orden segmentándolo y leyendo los datos que se encuentren después del carácter "/", que es el carácter acordado del que siguen los datos (Mensaje) –> función propia que utiliza las librerías "SoftwareSerial.h" y "WiFi.h". Se procesa el número después de cada "/" y dependiendo del número y de cuantos "/" haya en el mensaje se almacenará en una u otra función. Si Mensaje Recibido → OK, Se inicia el procesamiento del mensaje. Si Time Out → No se ha recibido la información del Arduino. No se puede procesar el mensaje. Se escribe el error en la variable de último error.
Errores
+ Datos no esperados: ERR013 + Datos nulos: ERR014 + Time_out alcanzado: ERR015 + Servidor desconectado durante la ejecución de la función: ERR0002

Casos de Uso que cubre	Requisitos que cubre
CU02, CU03, CU04	RE02, RE03,RE04

Control de errores: Esta función se encarga de detectar y notificar cualquier error que se haya producido mientras se ejecutaba las funciones anteriores.

Código	CFUN10	Nombre	Control de Errores (FU04)

Input	Output
+ Errores durante el procedimiento.	+ Mensaje de los Errores ocurridos durante el procedimiento.

Procesamiento
Se comprueban los errores que se hayan mostrado durante el procedimiento de esta función. Se crea un mensaje con el valor de la variable "último error" y se envía al Sistema. Funciones propias y captura de los Errores en los procedimientos anteriores.
Errores
+ Error no controlado: ERR007

Casos de Uso que cubre	Requisitos que cubre
CU01, CU02, CU03, CU04	RE01,RE02, RE03,RE04

Lo último que se realiza en esta función es guardar las variables globales con los valores recibidos o tratados. O guardarlos en la EEPROM si se va a utilizar. Y en este momento se deja paso al código de Control que no se ve en este proyecto, y que pertenece a otro proyecto, pero que sí que se encuentra dentro del contexto de este proyecto.

En este momento se procesan las ordenes e instrucciones que se le han solicitado en la petición, por medio de la modificación de las variables de Control. Teniendo ahora toda la información necesaria para tomar una decisión, ya que puede leer los sensores y las órdenes enviadas con anterioridad, para saber qué orden tiene preferencia y hasta qué punto la puede ejecutar.

Siempre que le sea posible el Arduino debe proceder a ejecutarlas. Igualmente esta parte la decidirá el código de Control que le permitirá decidir si puede o no moverse a partir de los datos que se reciben de los sensores. Si no tiene ninguna orden pendiente pasaría a actualizar su estado.

Luego, se retorna del código de Control y se precisa una lectura de todas las variables del propio Arduino para comprobar si ha habido algún cambio, alarma o error en ésta parte del código del Arduino.

Si se ha producido algún cambio se deberá informar a todo el Sistema del mismo, que es la función que se muestra a continuación. Si no únicamente aumentaría el contador que condiciona el mensaje “IsAlive” del propio Arduino.

FU05: Escribir Servidor.

La quinta función que debe cumplir los Arduino es la de escribir en el Servidor, cualquier información que se precise. Lo más usual será enviar las variables que se hayan modificado, o un mensaje de “alive”, que indicará al Servidor que sigue conectado al Sistema. De esta forma el servidor tiene conocimiento de los cambios que se han producido en el Sistema, y pueda enviarlos a la BBDD donde se almacenarán y estarán a disposición de las diferentes consultas que puedan realizar los usuario externo al Sistema.

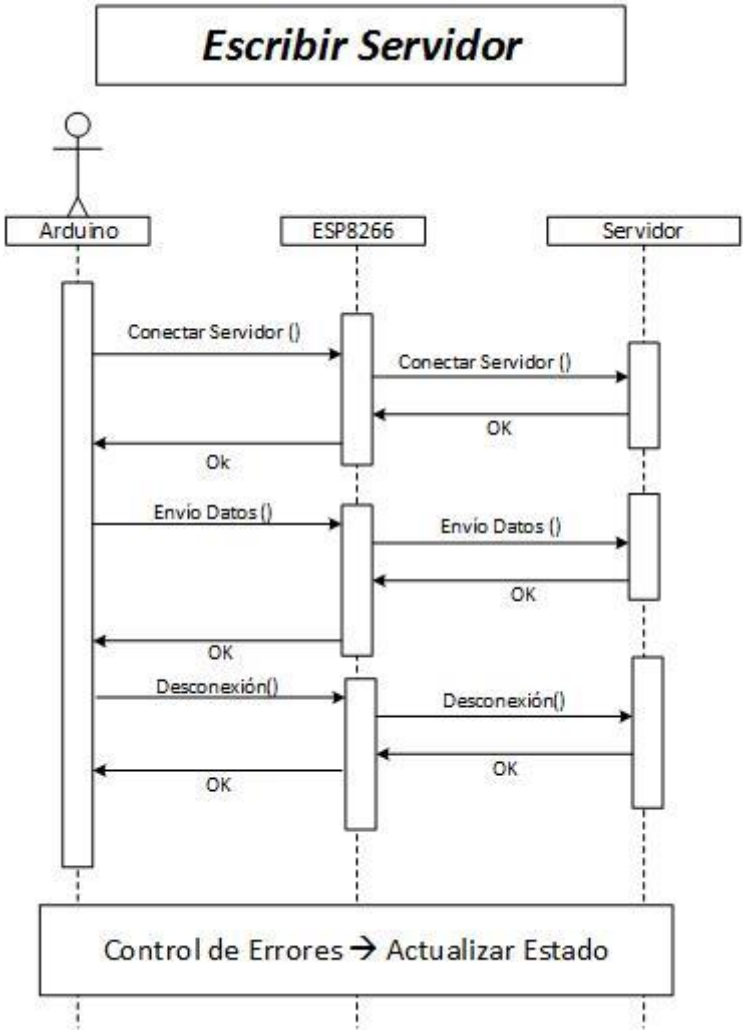

Ilustración 23. Diagrama de Secuencia 4. Escribir Servidor (Visio propio)

Como se puede apreciar en el Diagrama, interviene tres actores. El que comienza la acción es el propio Arduino. Ya sea porque al leer las variables que tiene en la EEPROM, o en las variables globales, comprueba que ha habido modificaciones, o bien por que sea acabado el time_out. Mandará una solicitud de conexión al Servidor, por medio del chip ESP8266. Después, le enviará la información que se precisa, y se desconectara del Servidor. Y por último, el Servidor guardará la información recibida en la Base de Datos (BBDD).

Para no sobrecargar el diagrama de Secuencia 4, se han omitido todos los time_out que tiene cada una de las llamadas desde el ESP8266 al Servidor.

Se cuenta también con el Control de Errores que notificará cualquier posible error de ejecución de las otras funciones. Y que luego dará paso a Actualizar el Estado siempre que sea preciso, para mantener los cambios o errores producidos en esta función. Las funciones que debe realizar para cumplir esta función son:

Conectar al Servidor: Esta función lo que hace es solicitar conexión al Servidor, para poder enviar la información que se ha modificado.

Código	CFUN11	Nombre	Conectar servidor (FU05)
Input			Output
+ IP Servidor			+ Resultado de la conexión
Procesamiento			
Se conecta con el servidor (IP) –> función propia que utiliza las librerías "SoftwareSerial.h" y "WiFi.h" Si servidor Ok → Se ha conectado con el Servidor. Se pasa a la siguiente función (Enviar Datos) Si servidor Time Out → No se ha conectado con el Servidor. Se acaba la función.			
Errores			
+ Servidor no localizado: ERR016 + Servidor desconectado durante la ejecución de la función: ERR0017			
Casos de Uso que cubre			Requisitos que cubre
CU02, CU04			RE02,RE04

Enviar Datos: Esta función envía los datos relacionados con el Arduino al Servidor.

Código	CFUN12	Nombre	Enviar Datos (FU05)
Input			Output
+ Datos a enviar			+ Recepción de los datos enviados
Procesamiento			
Se conecta con el Servidor, y se envía información deseada(Mensaje)–> función propia que utiliza las librerías "SoftwareSerial.h" y "WiFi.h" Si servidor OK → Se ha enviado los datos que se precisaban. Luego se pasa a la siguiente función (Desconexión) Si servidor Time Out → No se ha conectado con el Servidor. Se acaba la función y se guarda el error en la variable último error.			
Errores			
+ Servidor no ha podido recibir los datos: ERR018 + Servidor desconectado durante la ejecución de la función: ERR0019			
Casos de Uso que cubre			Requisitos que cubre
CU02, CU04			RE02,RE04

Desconexión: Esta función se desconecta de forma correcta del Servidor.

Código	CFUN13	Nombre	Desconexión (FU05)
Input			Output
+ Mensaje de Desconexión			+ Desconexión aceptada
Procesamiento			
Se desconecta del Servidor esperando la aceptación del mismo –> función propia, y se hace uso de las librerías "SoftwareSerial.h" y "WiFi.h" Si servidor OK → Se ha desconectado correctamente.			

Si servidor Time Out → No se ha recibido la desconexión, se escribe el error en la variable de último error.	
Errores	
+ Servidor no responde: ERR020 + Servidor desconectado durante la ejecución de la función: ERR0021	
Casos de Uso que cubre	Requisitos que cubre
CU02, CU04	RE02,RE04

Control de errores: Esta función se encarga de detectar y notificar cualquier error que se haya producido mientras se ejecutaba las funciones anteriores.

Código	CFUN14	Nombre	Control de Errores (FU05)
Input			Output
+ Errores durante el procedimiento.			+ Mensaje de los Errores ocurridos durante el procedimiento.
Procesamiento			
Se comprueban los errores que se hayan mostrado durante el procedimiento de esta función. Se crea un mensaje con el valor de la variable "último error" y se envía al Sistema. Funciones propias y captura de los Errores en los procedimientos anteriores.			
Errores			
+ Error no controlado: ERR007			
Casos de Uso que cubre			Requisitos que cubre
CU01, CU02, CU03, CU04			RE01,RE02, RE03,RE04

FU06: Escribir Arduinos vecinos

La sexta función que debe cumplir el Arduino es la de poder escribir las modificaciones de las variables y todo cambio que se haya producido en el Sistema, a todos los elementos de alrededor. De forma que todos los vecinos conozcan tus cambios y el conocimiento del Sistema esté en cada elemento. Dotando así de mayor independencia respecto al Servidor, posibilitando una configuración del Sistema diferente, o mayor distribución.

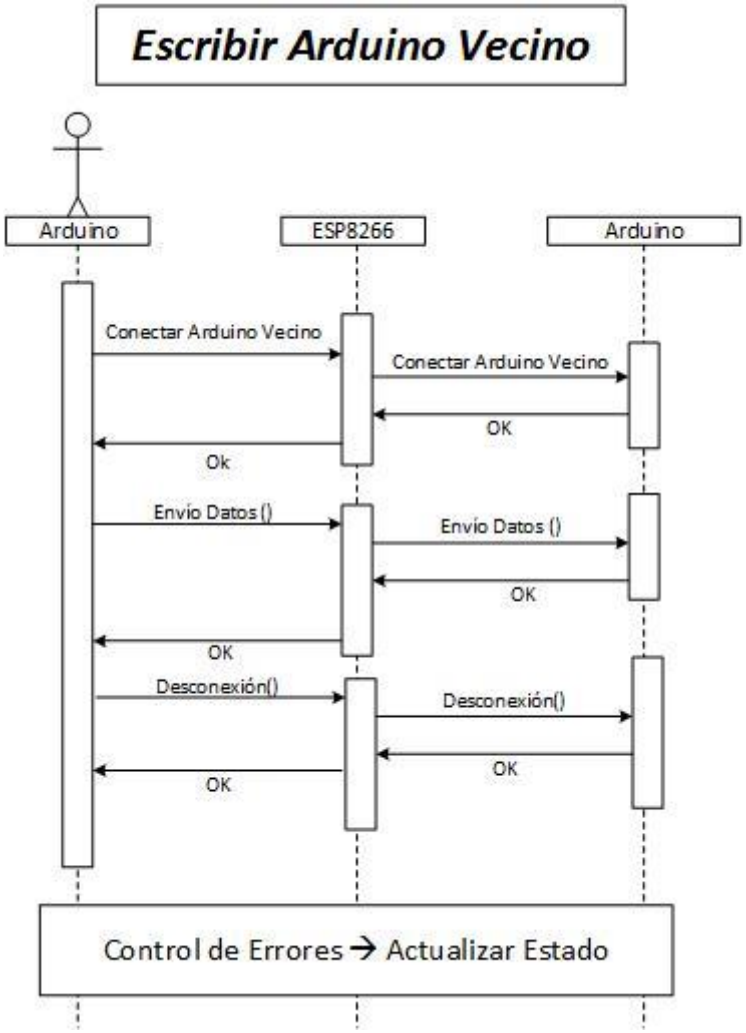

Ilustración 24. Diagrama de Secuencia 5. Escribir Arduinos Vecinos (Visio propio)

Como se puede apreciar en el Diagrama, interviene tres actores. El que comienza la acción es el propio Arduino, como en la Función anterior, FU05. Igual que Escribir en Servidor su comprueban las modificaciones, o el time_out y se procede al envío de la información modificada. Se manda una solicitud de conexión al Arduino objetivo, por medio del chip ESP8266. Después, le enviará la información que se precisa, y se desconectara del Arduino. Y por último, el Arduino deberá guardar esta información en sus variables globales o EEPROM, si se utiliza.

Para no sobrecargar el diagrama de Secuencia 5, se han omitido todos los time_out que tiene cada una de las llamadas desde el ESP8266 al Arduino objetivo.

Se cuenta también con el Control de Errores que notificará cualquier posible error de ejecución de las otras funciones. Y que luego dará paso a Actualizar el Estado siempre que sea preciso, para mantener los cambios o errores producidos en esta función. Las funciones que debe realizar para cumplir esta función son:

Conectar Arduino Vecino: Esta función lo que hace es solicitar conexión al Arduino específico que se quiere enviar la información que se ha modificado.

Código	CFUN15	Nombre	Conectar Arduino Vecino (FU06)
Input			Output
+ IP Arduino Vecino			+ Resultado de la conexión
Procesamiento			
Se conecta con el Arduino (IP) –> función propia que utiliza las librerías "SoftwareSerial.h" y "WiFi.h" Si Arduino Ok → Se ha conectado con el Arduino. Se pasa a la siguiente función (Enviar Datos) Si Arduino Time Out → No se ha conectado con el Arduino. Se acaba la función.			
Errores			
+ Arduino no localizado: ERR022 + Arduino desconectado durante la ejecución de la función: ERR0023			
Casos de Uso que cubre			Requisitos que cubre
CU03, CU04			RE03,RE04

Enviar Datos: Esta función envía los datos relacionados con el Arduino al Arduino objetivo.

Código	CFUN16	Nombre	Enviar Datos (FU06)
Input			Output
+ Datos a enviar			+ Recepción de los datos enviados
Procesamiento			
Se conecta con el Arduino objetivo, y se envía información deseada(Mensaje)–> función propia que utiliza las librerías "SoftwareSerial.h" y "WiFi.h" Si Arduino OK → Se ha enviado los datos que se precisaban. Luego se pasa a la siguiente función (Desconexión) Si Arduino Time Out → No se ha conectado con el Arduino. Se acaba la función y se guarda el error en la variable último error.			
Errores			
+ Arduino no ha podido recibir los datos: ERR024 + Arduino desconectado durante la ejecución de la función: ERR0025			
Casos de Uso que cubre			Requisitos que cubre
CU03, CU04			RE03,RE04

Desconexión: Esta función se desconecta de forma correcta del Arduino objetivo.

Código	CFUN17	Nombre	Desconexión (FU06)
Input			Output
+ Mensaje de Desconexión			+ Desconexión aceptada
Procesamiento			
Se desconecta del Arduino objetivo esperando la aceptación del mismo –> función propia, y se hace uso de las librerías "SoftwareSerial.h" y "WiFi.h"			

Si Arduino OK → Se ha desconectado correctamente. Si Arduino Time Out → No se ha recibido la desconexión, se escribe el error en la variable de último error.	
Errores	
+ Arduino no responde: ERR026 + Arduino desconectado durante la ejecución de la función: ERR0027	
Casos de Uso que cubre	Requisitos que cubre
CU03, CU04	RE03,RE04

Control de errores: Esta función se encarga de detectar y notificar cualquier error que se haya producido mientras se ejecutaba las funciones anteriores.

Código	CFUN18	Nombre	Control de Errores (FU06)
Input			Output
+ Errores durante el procedimiento.			+ Mensaje de los Errores ocurridos durante el procedimiento.
Procesamiento			
Se comprueban los errores que se hayan mostrado durante el procedimiento de esta función. Se crea un mensaje con el valor de la variable "último error" y se envía al Sistema. Funciones propias y captura de los Errores en los procedimientos anteriores.			
Errores			
+ Error no controlado: ERR007			
Casos de Uso que cubre			Requisitos que cubre
CU01, CU02, CU03, CU04			RE01,RE02, RE03,RE04

FU07: Desconexión del Sistema

La última función que se desarrolla en este proyecto será la de Desconexión. Esta desconexión se hará mediante una difusión muy parecida a la realizada al entrar en el Sistema, Identificación en el Sistema.

Cuenta con todos los Usuarios del Sistema que entran en interacción con el propio Arduino y actualizan los estados referentes a este equipo y a sus datos al desconectarse. Quedando también libre esa dirección en el Servidor.

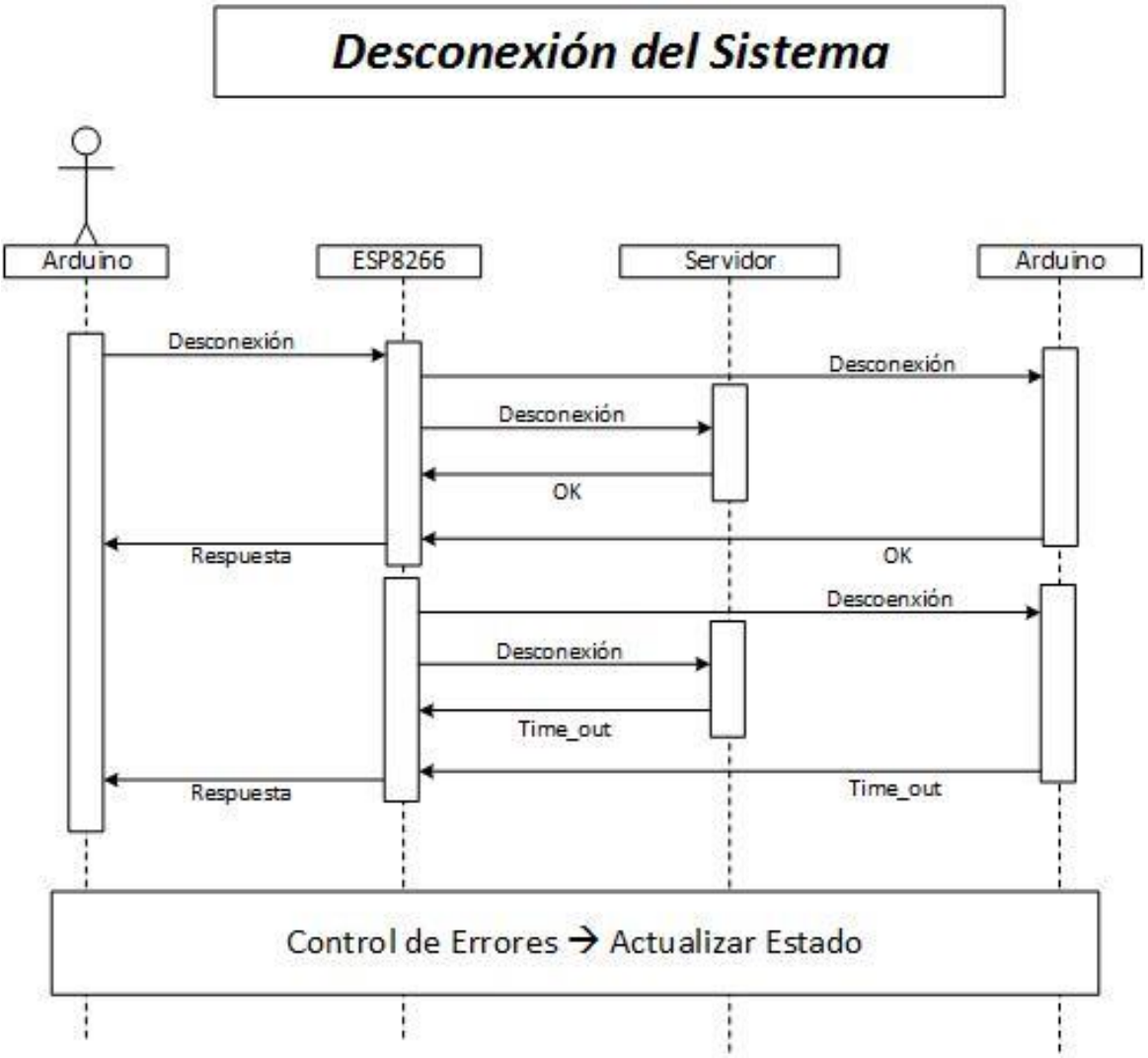

Ilustración 25. Diagrama de secuencia 7 (Desconexión del Sistema)

Desconexión del Sistema: Esta función envía a todos los Usuarios del Sistema un mensaje que les indique que tiene que borrar la información que almacenaban de este Arduino.

Código	CFUN19	Nombre	Desconexión del Sistema (FU07)
Input			Output
+ Mensaje de Desconexión			+ Resultado de la difusión
Procesamiento			
Se realiza una difusión a todos los elementos del Sistema informando que deben borrar todos los datos internos del Arduino (Mensaje) –> función propia que utiliza la librería "WiFiUDP.h". Si Usuarios OK → Se ha recibido el mensaje del Arduino y proceden a borrar su información. Si Usuarios Time Out → No se ha recibido el mensaje del Arduino. Se escribe el error en la variable de último error.			
Errores			
+ Servidor no responde: ERR028			

+ Arduino no responde: ERR0029	
Casos de Uso que cubre	Requisitos que cubre
CU05	RE05

Control de errores: Esta función se encarga de detectar y notificar cualquier error que se haya producido mientras se ejecutaba las funciones anteriores.

Código	CFUN18	Nombre	Control de Errores (FU07)
Input			Output
+ Errores durante el procedimiento.			+ Mensaje de los Errores ocurridos durante el procedimiento.
Procesamiento			
Se comprueban los errores que se hayan mostrado durante el procedimiento de esta función. Se crea un mensaje con el valor de la variable "último error" y se envía al Sistema. Funciones propias y captura de los Errores en los procedimientos anteriores.			
Errores			
+ Error no controlado: ERR007			
Casos de Uso que cubre			Requisitos que cubre
CU01, CU02, CU03, CU04			RE01,RE02, RE03,RE04

4.3.3. Capa de presentación

En este apartado únicamente se cuenta con los mensajes de error que ha sucedido en alguno de los procesos y que se deberán enviar tanto al Servidor, como transfórmalos en órdenes para los Arduinos.

4.3.3.1. Mensajes

En este apartado se detalla los mensajes que aparecerán a los usuarios externos al Sistema dependiendo de la diferente actividad que estén realizando. O que información se enviará al Servidor para que este informe al resto de Usuarios.

Los mensajes pueden ser de los siguientes tipos:

- Mensajes de Respuesta: Aquellos que devuelven algunas funciones informando del Estado del Sistema o de algún dato necesario.
- Mensajes de Error: Aquellos que provengan de algún error interno de alguna función o de algún problema con las conexiones.
- Mensajes de Alarma: Aquellos que prevengan de un estado del Sistema que corre riesgos o que se interpretan como no deseados.

En todos ellos se sigue la siguiente estructura:

Cabecera/Ruta	ID Emisor	Separador	Sensor	Separador	Valor	IP Emisor

Los campos del mensaje son los siguientes:

- Cabecera/Ruta: El comienzo del mensaje con la ruta donde se debe guardar el valor enviado.
- ID Emisor: El identificador que cada usuario tiene en el Sistema. (int)
- Separador: El carácter que se utilizará para poder controlar y procesar el mensaje.
- Sensor: El identificador que cada sensor tiene en cada Arduino. (int)
- Valor: El valor que se envía respecto al sensor especificado. (int)
- IP Emisor: La dirección ip del emisor para mayor facilidad de la gestión de los mensajes, y procesamiento del mismo.

4.4. Conclusiones

En este capítulo se ha definido la arquitectura que se va a utilizar en este proyecto, detallándose la estructura de las variables globales o de la memoria EEPROM del Arduino en la capa de Persistencia. En la capa de Negocio se ha indicado la estructura que tienen los Usuarios y las clases involucradas en las funcionalidades que se intentan cubrir. Pasando luego a explicar cada una de las funciones que se extraen del diseño de los diagramas de secuencia, que se han especificado en referencia al diagrama de actividad. Por último, en la capa de Presentación, se ha diseñado los mensajes se han de transmitir a los usuarios correspondientes.

En el Siguiente capítulo se va a implementar las funciones del código, basándose en el Diseño realizado en este capítulo. Dando lugar a las implantación del diseño y realizando una batería de pruebas que aseguren el cumplimiento del diseño, y por tanto, de los requisitos.

5. Implementación e implantación

5.1. Introducción

Una vez realizado el diseño del sistema, se procederá a su implementación, su evaluación y por último a su implantación. Por lo que respecta a la implementación, se detalla y explica partes importantes del código, donde se han encontrado más dificultades, o donde se ha conseguido de una manera eficiente, reducir el esfuerzo de la creación de código de calidad, logrando su objetivo optimizando los recursos de los que se dispone. También se muestra como se ha intentado sacar la mayor parte de la funcionalidad, para separar en módulos las funciones diferentes y agrupar las funciones comunes.

En la evaluación, se indican las pruebas que se han realizado para probar que se han cubierto todos los requisitos y las funcionalidades especificadas en el diseño.

En cuanto a la implementación, dado que este proyecto se centra en al comunicaciones y sus protocolos, se muestra la implantación en un "parcela" de pruebas controlada, compuesta por dos únicos Arduinos que se comunican entre sí, y se mencionan todos los aspectos necesarios para la instalación y desarrollo del sistema.

5.2. Implementación

5.2.1. Persistencia (SQL, etc.)

Aquí se va mostrar las variables globales que van a almacenar toda la información de los Arduino, también se cuenta con la memoria interna EEPROM para salvar los datos al desconectarse controladamente y leerlos al inicializarse. Para una mayor accesibilidad y eficiencia de esta memoria se ha implementado una estructura organizada de la siguiente manera:

```
typedef struct Red_Arduinos{
  Arduino   Ard1, Ard2, Ard3, Ard4, Ard5;
  Arduino   Aux;
};
struct Arduino{
 struct IP ip;
 int id;
 struct Estado est;
 int Md,Mi,Bh,Ta,Ha,Hs,La,D,Nd,Ss,Si;
 boolean usable;
 };
```

Código 1. Definiciones de estructuras.

También se especifican aquí las funciones que se han utilizado desde la librería EEPROM.h para facilitar el acceso y uso de esta memoria:

```
Red.Aux.id = EEPROM.read(4);
EEPROM.write(Direccion, Dir_ip);
```

Código 2. Uso de la EEPROM.

5.2.2. Negocio

Para los diferentes procesos que se han implementado, se ha realizado una agrupación de funcionalidad básica, en el que se agrupa por función y no por usuario para una mayor facilidad. Lo que permite actuar de igual forma sea el Servidor u otro Arduino el que envíe o reciba el mensaje. Por lo tanto, se contará con Enviar Mensaje y Recibir Mensaje que cumplirán con las funcionalidades (o Casos de Uso) de Leer Servidor, Escribir Servidor, Leer Arduinos Vecinos y Escribir Arduinos Vecinos. De ésta forma, la diferencia erradicará en el contenido del mensaje, y no en quien lo envía. Aquí se muestra la recepción de las peticiones en el Arduino:

```
while (SerialEsp8266.available() >0 ) {
   char c = SerialEsp8266.read();
     if (c == 47)  {
         String peticion = "";
         peticion = SerialEsp8266.readString();
         ProcesarPeticion(peticion);
         }
     }
```

Código 3. Recepción de Peticiones

Se utiliza el carácter ASCII 47 para determinar si se ha recibido la petición porque es el carácter que se ha decidido como separador de los campos y valores que se desea enviar. El método de procesamiento de las peticiones es el siguiente:

```
void ProcesarPeticion(String cadena)  {
  arduino = getValue(cadena,'/',0);
   tipo =  getValue(cadena,'/',1);
   valor =  getValue(cadena,'/',2);
}
```

Código 4. Procesamiento de peticiones

Lo que primero hace el procesamiento de peticiones es partir la petición y recuperar los valores que se desean a partir del método getValue(), que se mostrará a continuación. Luego, dependiendo de la información que se haya obtenido decidirá la siguiente acción. El método de tratamiento de la cadena es el siguiente:

```
String getValue(String data, char separator, int index) {
  int found = 0;
  int strIndex[] = {0, -1};
  int maxIndex = data.length()-1;
  for(int i=0; i<=maxIndex && found<=index; i++){
    if(data.charAt(i)==separator || i==maxIndex){
        found++;
        strIndex[0] = strIndex[1]+1;
        strIndex[1] = (i == maxIndex) ? i+1 : i;
    }
  }
return found>index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

Código 5. Procesamiento de la cadena

Este método lo que implementa es la búsqueda de separador indicado, y la obtención del valor inmediatamente posterior. También se ha conseguido agrupar la función de difusión al realizar los envíos de cualquier sensor o simplemente del Arduino indicando que sigue conectado.

```
String Multicast(int arduino, int Sensor, int Valor) {

udp.begin(PUERTO_SERVIDOR);
udp.beginPacket(udp.remoteIP(),udp.remotePort());
String mensaje = "GET /rest/maceta/sensor/" + (String)ID + "/"+Sensor+"/"+Valor+"
HTTP/1.1\r\nHost:"+IP0+"\r\n\r\n";
udp.print(mensaje);
udp.endPacket();

EnviarSensor(IP_SERVIDOR,Sensor, Valor);
 EnviarSensor(IP2,Sensor, Valor);
```

Código 6. Difusión Multicast.

Se contará también con una funcionalidad de Control de Errores, donde se almacenará el último error ocurrido, se preparará un mensaje con el error y después de enviarlo se actualizará el Estado del Arduino. El Control de Errores utiliza la variable global último error, UEr, y se basa en indicar en cada else el posible error encontrado. Cuando se localiza un Error o Alarma se informa al Sistema.

De los tres variables de estado que afectarán a los maceteros, el estado de comunicaciones es el importante en este proyecto. Se actualiza en los diferentes lugares del código mediante la variable EstadoCom. Dependiendo del valor de la variable habrá un LED u otro encendido. De este modo se mantiene en todo momento conocimiento de estado de las comunicaciones de cada Arduino.

5.2.3. Presentación

Para la presentación, se muestran los mensajes que se intercambian con el Servidor, y entre los diferentes Arduinos, junto con la creación del paquete que se desea enviar. El código para la creación del paquete para el envío de los mensajes es el siguiente:

```
void EnviarSensor (String destino,int nombre_sensor, int valor_sensor) {
String cmd = "AT+CIPSTART=1,\"TCP\",\"" + (String)destino +"\"," + PUERTO_SERVIDOR;
  String mensaje = "GET /rest/maceta/sensor/" + (String)ID + "/" + nombre_sensor + "/"+valor_sensor+" HTTP/1.1\r\nHost:"+IP0+"\r\n\r\n";
…}
```

Código 7. Empaquetamiento del mensaje.

En la variable cmd se almacena el comando que se tiene que ejecutar para comenzar la comunicación. Siendo el PUERTO_SERVIDOR el puerto por el que se va a enviar el mensaje, y la variable destino la dirección IP a la que se desea enviar. Y en mensaje, se encapsula la información que se va a enviar conforme se había especificado en el diseño. Aquí se muestra el envío del paquete anteriormente formado:

```
SerialEsp8266.println(cmd);
…
if (SerialEsp8266.find("OK")) {
    http(mensaje, "1"); }
```

Código 8. Comandos para ESP8266

Aquí se muestra como se ejecuta el comando para que el ESP8266 pueda comenzar a transmitir datos, y como se espera la confirmación para hacer la llamada al método http que será el que realmente envíe el mensaje. A continuación el código http:

```
void http(String output, String n){
    SerialEsp8266.print("AT+CIPSEND="+n+",");
    SerialEsp8266.println(output.length()+2);
…
SerialEsp8266.println(output);
…
}
```

Código 9. Envío del mensaje

Como se muestra en este fragmento de código se informa del comienzo del envío y a continuación se envía la cadena que recibida como argumento, habiendo indicado con anterioridad la longitud del mensaje.

5.3. Evaluación

Las pruebas que se van a realizar son pruebas individuales desde el punto de vista del Usuario, y que gracias al diseño que se ha realizado en este proyecto y la relación mantenida entre; las características, los Casos de Uso y requisitos, las funciones, los algoritmos diseñados y el código realizado. Se puede mediante los test que se han

realizado comprobar el funcionamiento del código. Y que sucesivamente debido a la relación antes mencionada permite asegurar el cumplimiento del diseño establecido, y por tanto, de las funciones y requisitos propuestos, que a su vez cubren las características que han impulsado la finalidad de este proyecto.

Prueba	Cobertura	Resultado
Test 01	CU01→(RE01, RE04)	Si se enciende un Arduino, éste se inicializa en el Sistema.
Test 02	CU01→(RE01, RE04)	Si se solicita la IP al Servidor DHCP, se recibe una IP y se almacena en el Arduino.
Test 03	CU01→(RE01, RE04)	Si el Arduino se presenta en el Sistema, los Usuarios almacenan sus datos.
Test 04	CU02→(RE02, RE04)	Si el Arduino recibe una petición del Servidor, la procesa.
Test 05	CU02→(RE02, RE04)	Si el Arduino envía una petición al Servidor, le llega.
Test 06	CU03→(RE03, RE04)	Si el Arduino recibe una petición de otro Arduino, la procesa.
Test 07	CU03→(RE03, RE04)	Si el Arduino envía una petición a otro Arduino, le llega.
Test 08	CU04→(RE03, RE04)	Si la información del Arduino cambia, comunica la información al Sistema.
Test 09	CU04→(RE03, RE04)	Si la información del Arduino cambia al procesar una petición, guarda la información.
Test 10	CU04→(RE04)	Si durante el proceso cambia el estado del Arduino, se modifican las variables.
Test 11	CU04→(RE04)	Si el Arduino se va a desconectar guarda la información en la EEPROM
Test 12	CU05→(RE05, RE04)	Si el Arduino recibe la orden de desconectarse, se desconecta.

Test 13	CU05→(RE05, RE04)	Si el Arduino se va a desconectar guarda la información en la EEPROM

Tabla 14. Pruebas del Sistema.

Como se puede apreciar, cumpliendo las pruebas que comprueban el código relacionado, se está probando también el algoritmo, que a su vez, cumple también con las funcionalidades que se deseaban. Y las funcionalidades cumplen con los requisitos especificados en el proyecto. Por lo tanto, todo está probado y se cumple.

5.4. Implantación

Dado que este proyecto se encuentra en un contexto y un marco que incluye otros proyectos, puede llegar a extenderse y crecer de forma amplia y variada. Por eso se ha decidido, para dar un poco de independencia a este proyecto implantar y realizar unas pruebas para el caso de que solo exista este proyecto, y que se ha cumplimentado todo lo necesario para que el proyecto funcione pese a no tener un Servidor, y pese a no tener el resto de Sensores y Actuadores que se ha hecho mención en este proyecto, en relación con otros proyectos.

La implantación que se va a realizar supone el estudio únicamente de las variables de comunicación y del protocolo que se diseña en este proyecto. Teniendo como producto final, dos placas Arduino conectadas cada una a su propio chip ESP8266, que se conectan a un mismo punto de acceso. Cada Arduino contará con un botón y varios LED de estado, que servirán de pruebas para comprobar que se logra la funcionalidad deseada, especificada partir de los Casos de Uso y de los Requisitos antes descritos en el Sistema. La siguiente tabla describe los estados posibles de este Sistema:

Estados	LED IP	LED Espera	LED Envío	Estado Comunicación
XX:YY:00	OFF	OFF	OFF	Iniciando
XX:YY:02	ON	OFF	OFF	Conectado
XX:YY:03	ON	ON	OFF	Esperando
XX:YY:04	ON	OFF	ON	Enviando
XX:YY:05	ON	OFF	OFF	Procesando

Tabla 15. Estados prueba controlada

De esta forma se puede observar de forma muy sencilla que el envío y recepción de información entre los Arduinos es efectiva y satisfactoria. Con el añadido de que no se necesita el Servidor para comunicarlos, y que gozan de una cierta independencia.

5.4.1. Instalación y soporte

Se van a montar dos placas Arduino Uno, cada uno con un chip ESP8266, en un circuito eléctrico con resistencias, dos botones (uno por Arduino) y 2 LED's cada placa. Se va a cargar en cada uno de los Arduinos el código a probar. El único soporte necesario será que se compruebe que no hay ningún fallo eléctrico en el montaje.

5.5. Conclusiones

En este capítulo se ha definido la implementación que se ha realizado según el diseño establecido en el capítulo anterior. En la capa de Persistencia, se ha mostrado las variables globales que se utilizan, y el uso que se ha hecho de la EEPROM. En la capa de Negocio, se ha mostrado la implementación del procesamiento de peticiones, junto con su procedimiento de cadena y la forma en la que se difunde al Sistema la información que haya sido modificada. En la capa de Presentación, se ha implementado la creación, preparación y el envío de los mensajes que se van a difundir por el Sistema.

También se ha realizado las pruebas necesarias para probar el Sistema, indicándolo en una tabla de resultados. Por último, se ha explicado la forma en la que se ha decidido implantar éste Sistema para las pruebas y sus especificaciones de instalación y soporte.

En el Siguiente capítulo se va a comentar las conclusiones finales obtenidas en parte por los resultados obtenidos, y por todo el conjunto de capítulos anteriores. Comentando las dificultades encontradas y como se han solucionado, las aportaciones que se han obtenido de este proyecto y sus posibles futuras ampliaciones.

6. Conclusiones

6.1. Dificultades y soluciones

A continuación se describen las dificultades más relevantes que se han encontrado a lo largo de la realización de este proyecto y cómo se han logrado resolver.

La primera dificultad que se presentó fue a la hora de buscar sistemas similares. Dado que el chip de comunicaciones utilizado, ESP8266, es bastante reciente, no se lograba encontrar productos terminados que se asemejaran al proyecto, y por tanto, en menor medida se encontraron sistemas con las Placas Arduino que utilizaran también el ESP8266. Ya que casi todos los sistemas que se encontraron se centraban en los sensores y actuadores que controlaba, y no en las comunicaciones que permitía.

Se consiguió resolver haciendo una búsqueda exhaustiva de imágenes de robots, que contarán con el chip o la placa Arduino, o dispositivos semejantes. Luego se buscó la relación en las webs donde se habían encontrado las imágenes y se encontró las fuentes y creadores de dichos sistemas.

El segundo problema que se encontró fueron las dificultades al entender el objetivo del proyecto, junto con los problemas de definir el alcance y objetivos del proyecto. Debido a que el título del proyecto es tan general, se puede encarar desde una gran variedad de frentes y recoge una variedad muy diversa de vías a desarrollar.

Este problema se consiguió solucionar a base de tutorías con los cotutores del proyecto y reuniones grupales, donde se planteaban las dudas del proyecto y se iban poniendo metas alcanzables de reunión en reunión. Utilizando técnicas y herramientas para el desarrollo de proyecto con metodologías ágiles.

El tercer gran problema que se descubrió fue la creación del protocolo, que debido al conjunto de reuniones y el trabajo grupal no dejaba de cambiar. Cada vez que se conseguía alcanzar un acuerdo de la estructura o el alcance del protocolo se descubría una nueva funcionalidad necesaria, o una restricción que imposibilitaba el diseño. Por ello se han encontrado muchas dificultades en el diseño del protocolo, y la implementación escogida.

Este problema no se logró resolver de forma inmediata, sino que se fue adaptando y modificando el diseño y la implementación hasta encontrar el diseño preciso. Que permitía cumplir los requisitos y satisfacer las características establecidas en los primeros capítulos. Teniendo que limitar el proyecto hasta su alcance para no exceder su amplitud y complejidad enormemente.

Se han tenido muchos problemas con la compilación del código Arduino, ya que se ha tenido que aprender a utilizarlo desde 0. Debido también a los fallos con las librerías, que son difíciles de deducir y corregir. Y no siempre se conseguía utilizar la librearía

correcta, o no todas las librearías son para la placa que se utiliza en el proyecto. Si no que algunas librearías se orientan únicamente al chip de comunicaciones, esp8266 y no a su configuración desde un Arduino.

La mayor parte de estos problemas se resolvieron en una búsqueda de soluciones a temas similares en internet, y videos y foros donde se ha podido aprender lo necesario de los Arduinos y su configuración.

Uno de los últimos problemas que han surgido, han sido con la memoria dinámica de los Arduino, EEPROM, y sus limitaciones. Una de ellas es que se sobrepasaba su capacidad debido a cadenas de String, mensajes que se creaban y a algunas variables que utilizaban demasiada memoria. Ya que cada mensajes si cambiaba en lo más mínimo respecto a otro se consideraba nuevo y volvía a reservar una parte de la memoria dinámica. Otro problema con la EEPROM ha sido que tiene una limitación de escritura bastante reducida, y se ha tenido que replantear otra implementación del código.

Este problema se consiguió resolver reduciendo los mensajes y cadenas de String lo máximo posible, y modificando los errores por códigos que se interpretan en una tabla de errores fuera del Arduino. Y utilizando la EEPROM, únicamente para salvar la información necesaria en una desconexión controlada o en la lectura de inicialización.

6.2. Aportaciones

Tecnológicas

La principal aportación del proyecto es el protocolo de comunicaciones del sistema, y el estudio y análisis que cubre el objetivo principal del proyecto.

Adicionalmente, también se aporta el estudio presentado en el capítulo 2 donde se han buscado y analizado una serie de sistemas similares que pueden ser de interés para el lector.

También se proporciona un ejemplo de especificación y diseño del sistema que puede servir de base para el desarrollo de proyectos similares.

Por último, se ha planteado una solución, siempre mejorable, basado en los requisitos del capítulo 3 y el diseño del capítulo 4.

Socio-ambiental

Las aportaciones sociales o medioambientales de este proyecto van en referencia a la óptima gestión de los recursos utilizados y muchas veces tan necesarios en la sociedad. Entre ellos se aprecia una buena gestión del consumo de agua, de luz y de todo lo necesario para abastecer un jardín.

Personales

En el ámbito personal, este proyecto ha ayudado a manejar y superar situaciones de estrés, de cansancio acumulado o la importancia de las fechas de entrega.

Académicas

Este proyecto proporciona una gran variedad de aportaciones, tanto en el ámbito académico o laboral; aprendiendo diferentes tecnologías y lenguajes, útiles en la vida laboral, realizando reuniones grupales de diferentes proyectos, decidiendo vías de desarrollo importantes para continuar el proyecto, etc.

6.3. Trabajo futuro

Se podría dar mayor independencia del Servidor, adquiriendo un modelo más descentralizado y dotando a los Arduinos de una mayor inteligencia, mediante Agentes Inteligentes. Estos podrían comunicar y solicitar información o recursos a los demás Arduinos, e interactuar para alcanzar a satisfacer esa necesidad. Por ejemplo compartiendo el Agua que tiene un Arduino con otro debido a la falta en el depósito del segundo. O a difundir la información desde el Servidor por toda la red de Arduinos hasta que alcance el Arduino deseado.

Referencias

Bibliográficas

[1] Junestrand, S., Passaret, X., & Vázquez, D. (2004). Domótica y hogar digital. Editorial Paraninfo.

[2] Serna, A., Ros, F., & Rico, J. C. (2010). Guía práctica de sensores. Creaciones Copyright SL.

[3] Popovic, M. (2016). *Communication protocol engineering*. CRC press.

[4] Lauesen, S. (2002). Software requirements: styles and techniques. Pearson Education.

[5] McRoberts, M., Levy, B., & Wootton, C. (2010). *Beginning Arduino* (pp. 1-12). New York.: Apress.

[6] Carballar, J. A., & Falcón, J. A. C. (2010). *Wi-Fi: lo que se necesita conocer*. RC Libros.

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

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

[9] Comer, D. E. D. E., Equihua, M., Equihua, S. M., & Huitema, C. H. (1996).*Redes globales de información con Internet y TCP/IP: Principios básicos, protocolos y arquitectura*. Prentice-Hall Hispanoamericana,

[10] Cabello, A. L. C. (2015). Desarrollo de aplicaciones web distribuidas. IFCD0210. IC Editorial.

[11] Blum, R. (2014). *Arduino Programming in 24 Hours, Sams Teach Yourself*. Pearson Education.

[12] Pilone, D., & Pitman, N. (2005). *UML 2.0 in a Nutshell*. " O'Reilly Media, Inc.".

Internet

ISM. Empresa de monitorización de placas solares. http://www.ismsolar.com → Última conexión Junio 2016

Hacedores. Blog de desarrollo de sistemas de hardware. http://hacedores.com→ Última conexión Junio 2016

Saulo. Blog de aprendizaje hardware y redes. http://www.saulo.net → Última conexión Junio 2016

Todopic. Foro de desarrollo hardware. http://www.todopic.com.ar → Última conexión Junio 2016

Roboremo. Página de desarrollo hardware. http://www.roboremo.com → Última conexión Junio 2016

Akirasan. Blog de desarrollo hardware. http://www.akirasan.net → Última conexión Junio 2016

Polaridad. Página de consulta hardware. http://polaridad.es → Última conexión Junio 2016

Arduino a muerte. Blog de desarrollo hardware. http://arduinoamuete.blogspot.com.es → Última conexión Junio 2016

Prometec. Blog de desarrollo de tecnologías. http://www.prometec.net → Última conexión Junio 2016

El Cacharrero. Blog de aprendizaje de tecnologías. http://blog.elcacharreo.com → Última conexión Junio 2016

Iotfrog. Blog de desarrollo de comunicaciones. http://www.iotfrog.com → Última conexión Junio 2016

Foro oficial de ESP8266. http://www.esp8266.com → Última conexión Junio 2016

Panama Hitek. Blog de desarrollo y aprendizaje hardware. http://panamahitek.com → Última conexión Junio 2016

Foro de Arduino, página oficial. https://forum.arduino.cc → Última conexión Junio 2016

Leantec. Blog de desarrollo de hardware y comunicaciones. http://www.leantec.es → Última conexión Junio 2016

Giltesa. Blog de desarrollo hardware. https://giltesa.com → Última conexión Junio 2016

Alojamientos referentes a librerías. http://esp8266.github.io → Última conexión Junio 2016