Centro de Estudios de Postgrado

Máster en Técnicas Geoespaciales para la Gestión Inteligente del Territorio.

UNIVERSIDAD DE JAÉN
Centro de Estudios de Postgrado

Trabajo Fin de Máster

DISEÑO DE SISTEMA PARA LA CAPTURA Y ANÁLISIS DE PARÁMETROS AMBIENTALES EN UNA PARCELA DE OLIVAR.

Alumno/a: **Guillermo Galiano Parras**

Tutor/a: Prof. D. Manuel A. Ureña Cámara
Prof. D. Alejandro Sánchez García

Dpto: Ingeniería Cartográfica, Geodésica y Fotogrametría.

Ingeniería Electrónica y Automática.

Diciembre, 2015

Titulo:	**Diseño de sistema de captura y análisis de parámetros ambientales en una parcela de olivar.**
Autor:	**Guillermo Galiano Parras.**
Tutores:	**Manuel A. Ureña Cámara** **Alejandro Sánchez García.**
Departamento:	**Ingeniería Cartográfica, Geodésica y Fotogrametría.** **Ingeniería Electrónica y Automática.**
Fecha:	**16 de Diciembre de 2015**

Resumen

En este trabajo se desarrollará un sistema de estudio y análisis de los parámetros ambientales que tienen influencia en la producción del olivar.

Por una parte, se desarrollará un Datalogger mediante el hardware Arduino, cuya función será la de la captura de la información de los sensores. Por otra parte, se desarrollará una web capaz de monitorizar y analizar la información obtenida.

Para llevar a cabo el flujo de información entre la placa de Arduino y la web de visualización, se ha diseñado un servidor capaz de gestionar la información en una base de datos PostgreSQL, así como un Servidor Web para mapas (Geoserver), que proporcionarán la información que solicite el Cliente Ligero.

Palabras clave: Olivar, Agricultura de precisión, Arduino, hardware libre, código abierto, Geoserver, Openlayers, PostgreSQL, Postgis.

Title:	**Catchment system design and environmental parameters analysis in an olive grove.**
Author:	**Guillermo Galiano Parras.**
Tutors:	**Manuel A. Ureña Cámara** **Alejandro Sánchez García.**
Departament:	**Cartographic, Geodetic Engineering and Photogrammetry.** **Electronic and Automation Engineering.**
Date:	**16 December 2015**

Abstract

In this work, it will be developed a study and analysis system of the environmental parameter, which has an influence on the olive grove production.

On the one hand, it will be developed a Datalogger by a hardware called Arduino, whose function will be the capture of the sensor's information. On the other hand, it will be developed a website able to monitor and analyse the information that has been obtained.

In order to carry out the information flow between the Arduino and the display website, it has been designed a server capable to process the information in a PostgreSQL database as well as a web server for maps (Geoserver), which will provide the information the Slim Client request.

Keywords: Olive grove, Precision Agriculture, Arduino, Open hardware, Open source, Geoserver, Openlayers, PostgreSQL, Postgis.

Contenido

Lista de Ilustraciones.

Lista de Ilustraciones.

CAPÍTULO 1: Introducción.

1.1. Motivación.

La idea del presente Trabajo Fin de Máster surge de lo aprendido en el Máster en Tecnologías Geoespaciales para la Gestión Inteligente del Territorio, en la búsqueda de aplicaciones tanto para las Smart Cities como para Smart Territories y que pudiera tener una utilidad en nuestra sociedad actual.

Poniendo el foco en la provincia de Jaén, surge de inmediato el sector agrícola como el más importante, destacando de manera muy significativa el sector Olivícola, del cual es el mayor productor mundial. Este sector, aún siendo el más importante económicamente, no ha sufrido una revolución tecnológica.

Con el objetivo de que el Trabajo Fin de Máster tuviese una utilidad real y aplicable, surge la idea de desarrollar un sistema de captura y análisis para el sector olivícola, en el que se dé un paso del *Olivar Tradicional* a un *Olivar de Precisión,* que apueste por un modelo más eficiente y respetuoso con el medio ambiente.

1.2. Objetivo y alcance del proyecto.

El objetivo principal del Proyecto fin de Máster es conseguir un sistema que permita la captura, gestión estructurada, análisis y visualización de parámetros ambientales que influyen directamente en desarrollo de Olivar.

Como objetivos secundarios que se pretenden alcanzar con el proyecto:

- **Desarrollo de un hardware (Arduino) para la captura de datos.**
  Diseñar y construir un prototipo de Datalogger, que integre los sensores y actuadores necesarios para poder llevar a cabo la captura y el almacenamiento de los datos.

- **Diseño de una red de sensores para la gestión de un finca de olivar.**
  Se estudiara la distribución adecuada de los sensores en un entorno real donde poder poner en práctica el desarrollo del proyecto.

- **Almacenamiento y análisis de la información.**

La información obtenida de hardware desarrollado será almacenada y gestionada por una base de datos geográfica que contendrá información georeferenciada de los datos.

- **Visualización de la información.**
  El producto final del proyecto será una Web donde poder visualizar la información obtenida por los sensores a través de una cartografía especifica.

CAPÍTULO 2: Antecedentes.

2.1. Agricultura de Precisión.

Los sistemas agrícolas siempre han estado orientados a conseguir un máximo rendimiento y optimización de sus productos, desde la introducción de nuevas metodologías agrarias a modificaciones genéticas de las plantas.

En los años '60 la agricultura de altos insumos permitió que se duplicara el rendimiento de distintos cultivos, en especial de los cereales (maíz, trigo y arroz), siendo responsable de este incremento el uso de semillas mejoradas de variedades de alto rendimiento, el riego, los fertilizantes y otros agroquímicos. No obstante, esto ha conferido una alta vulnerabilidad a los sistemas agrícolas, debido a la alta uniformidad genética y la alta dependencia de los insumos, lo que ha motivado que los incrementos en rendimientos se han venido desacelerando. Pero estas nuevas técnicas y el alto uso de insumos ha derivado en un alto costo medioambiental, lo que según Eswaran*et al*. (2001) ha conducido a una disminución cercana al 50% en la capacidad productiva de la tierra.

En consecuencia, se requiere de la implementación de sistemas de producción que no solo estén orientados al logro de altos rendimientos, sino que también reduzcan el impacto negativo de las prácticas de manejo sobre el medio ambiente, aprovechen las condiciones agroecológicas particulares de cada sitio e incrementen las ganancias. (Ovalles, 2006).

Sobre esta base se define la Agricultura de Precisión: "Es un compendio de técnicas que permiten optimizar el uso de insumos, las labores realizadas al cultivo y la producción final, reduciendo los costes de producción y el impacto sobre el medio ambiente. Se configura como un conjunto de herramientas que permiten realizar cada una de las tareas que componen la actividad agrícola con el mayor nivel de homogeneidad y eficiencia. Estas herramientas están basadas generalmente en las nuevas tecnologías y su aplicación abarca desde la toma minuciosa de datos del cultivo, hasta la ejecución de las operaciones mecanizadas mediante maquinaria agrícola." (Juan Agüera Vega, 2012).

En un ámbito más específico, actualmente la Olivicultura de precisión o el Olivar de Precisión se está desarrollando por dos ramas o metodologías diferentes, pero ambas tienen como objetivo la gestión y ayuda en la toma de decisiones del agricultor.

Por una parte, el uso de la fotogrametría y la teledetección obtienen un muestreo del suelo georeferenciado . "Mediante la combinación de las bandas espectrales es posible obtener, para cada píxel de la imagen, información cuantitativa de los parámetros biofísicos relacionados con la dinámica de la vegetación" (Rios, 2014). Por lo tanto, combinando las diferentes bandas espectrales es posible obtener índices que representen el estado de la vegetación en ese instante.

La rapidez de la toma de información, así como la gran extensión que es capaz de cubrir estas imágenes es una de las grandes ventajas.

Una de las desventajas de este método era la falta de resolución de las imágenes, que no permitían el estudio de pequeñas extensiones. En la actualidad, el avance tecnológico de los UAV ha favorecido al desarrollo de estas técnicas a una escala más pequeña.

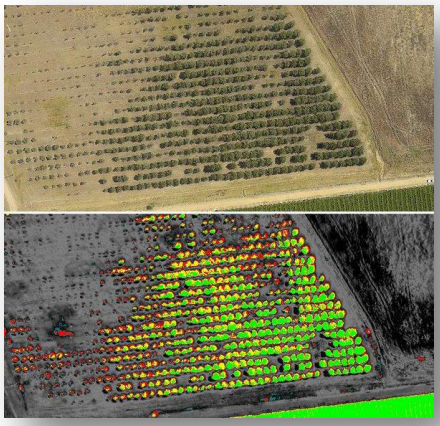

Ilustración 1 Índice de Vegetación (NDVI) en el olivar

Los fotogramas muestran el estado actual del cultivo en un instante exacto, si lo que se pretende es controlar la continuidad de los factores fundamentales en el crecimiento de la planta realizando un estudio de la evolución este no sería el método adecuado, ya que supondría un elevado coste la continua producción de imágenes espectrales.

Por otra parte, se encuentra el uso de Datalogger o Registradores de Datos, los cuales están integrados por microprocesadores, además de una serie de sensores capaces de capturar y monitorizar en el tiempo y en una ubicación determinada parámetros de la planta, ambientales y del suelo.

Este método nos permite actuar de manera localizada en el cultivo, el conocimiento específico de los diferentes parámetros que controlan los sensores favorece una actuación concreta en nutrición de la planta, estimación de cosecha, mejora de la eficacia en el riego o la predicción de enfermedades del cultivo. (Olivicultura de precisión).

Ilustración 2 Ejemplo de Datalogger.

A favor de este método, encontramos la posibilidad de un muestreo continuo, incluso en algunos casos, hasta en tiempo real de los parámetros, así como la utilización de una gran variedad de sensores específicos.

Como desventaja al método, se encuentra la necesidad de homogeneizar la información en sectores o en

parcelas debido al elevado coste de situar un *Datalogger* en cada planta.

El objetivo final es la ayuda en la toma de decisiones del agricultor, ambos métodos tienen en común el uso de las TIC's (Tecnologías de la Información y la Comunicación) y SIG (Sistemas de Información Geográfica) capaces de almacenar de forma estructurad la información en una base de datos espacial, además de servir como soporte en el tratamiento y la visualización de la información. Del mismo modo, el uso de Big Data proporciona grandes cantidades de información externa que puede ser útil para la toma de decisiones.

2.2. Tipos de sensores del mercado.

Actualmente existe una gran variedad de tipos de sensores capaces de ser gestionados por microprocesadores y que se utilizan en el sector de la agricultura. Es por ello que en este trabajo se distinguirá en dos tipos diferentes: Sensores de bajo coste (Low Cost) y sensores Industriales.

2.2.1. Sensores Low-Cost.

Son los más habituales y los que podemos encontrar en la mayoría de los aparatos electrónicos que nos rodean. Éstos se caracterizan por tener un funcionamiento bastante básico y suelen utilizarse para el prototipado.

- ***Sensor de gotas de lluvia.***

Ilustración 3 Sensor YL-83

Se trata de un sensor analógico aunque contiene un comparador que permite una salida digital. El funcionamiento se basa en medir la variación de conductividad eléctrica cuando hay alguna gota de lluvia en la superficie del sensor.

- ***Sensor de presión barométrica.***

Ilustración 4 Sensor BMP180

Esta pequeña placa incluye un sensor de presión barométrica BMP180 de alta precisión con un rango de medida de entre 300 y 1100 hPa (Hecto Pascal). Está basado en tecnología piezo-resistiva de alta eficiencia, linearidad, larga duración y bajo consumo. Además de presión también es capaz de medir altura y temperatura.

- ***Sensor de Humedad relativa y temperatura.***

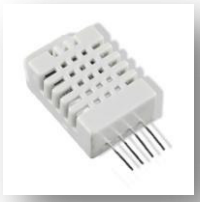

Ilustración 5 Sensor DHT22

Usa un sensor de capacidad para medir la humedad y un termistor para medir la temperatura del aire que lo rodea. Está diseñado para medir temperaturas entre 0 y 50°C con una precisión de ±2°C y para medir humedad entre 20% y 80% con una precisión de 5% con periodos de muestreo de 1 segundo.

- ***Sensor de luminosidad.***

Ilustración 6 Sensor LDR

Una fotorresistencia es un componente electrónico cuya resistencia disminuye con el aumento de intensidad de luz incidente. Puede también ser llamado fotorresistor, fotoconductor, célula fotoeléctrica o resistor dependiente de la luz, cuyas siglas, LDR, se originan de su nombre en inglés *light-dependent resistor*. Su cuerpo está formado por una célula o celda y dos paticas. En la siguiente imagen se muestra su símbolo eléctrico.

El valor de resistencia eléctrica de un LDR es bajo cuando hay luz incidiendo y muy alto cuando está a oscuras.

- ***Sensor de humedad en el suelo.***

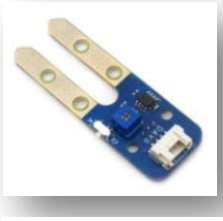

Ilustración 7 Sensor FC-28

El funcionamiento de este sensor es muy básico. Se trata de un sensor analógico que devuelve una tensión proporcional al nivel de humedad del medio. De este modo es capaz de medir con relativa precisión si el suelo está seco o húmedo.

- ***Termómetro infrarrojo,***

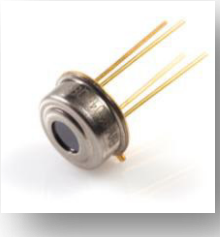

Ilustración 8 Sensor MLX90614

Este sensor mide la temperatura de una porción de la superficie de un objeto, a través de la luz infrarroja que emite y la teoría del cuerpo negro. La distancia de medida depende del Ángulo de captura del sensor.

- ***Sensor de temperatura.***

Ilustración 9 Sensor LM-35

Existen una amplia gama de sensores de medida de la temperatura, tanto sensores analógicos como digitales. El funcionamiento de estos sensores se basa en que, cuando dos metales diferentes tienen contacto, produce un pequeño voltaje de circuito abierto. Se puede usar este voltaje termoeléctrico para calcular la temperatura.

2.2.2.Sensores Industriales.

Estos sensores suelen ser herramientas desarrolladas para el control de alguna variable concreta. Se caracterizan por una mayor complejidad en su funcionamiento y un mayor coste.

- ***Dendrómetros .***

Ilustración 10 Sensor LVDT

Cuando un árbol transpira, pierde agua con una tasa mayor a la de absorción por las raíces, por lo que se deshidrata parcialmente. Esto se traduce en una reducción del diámetro del tronco que, aunque no suele ser mayor de una fracción de milímetro, puede medirse con dendrómetros de alta precisión, fijados al tronco de los árboles. Este método permite el registro continuo del estrés hídrico del árbol.

(Fernandez, y otros, 2012)

- ***Medidor del grosor de la hoja.***

Ilustración 11 Sensor de grosor de hoja.

El grosor de la hoja es un parámetro que se modifica en base a pequeñísimos cambios en la disponibilidad de agua para la planta. Con la carencia de agua, las células de las hojas se compactan, de manera que se puede hacer una correlación entre el grosor y el estado hídrico de la planta. (Luis Gurovich, 2015).

- ***Medición de flujo de savia.***

Ilustración 12 Sensor Sap Flow

La cantidad de agua consumida por un árbol se puede estimar a partir de la medida del flujo de savia en el tronco. En arboles leñosos como el olivo se basan en la inserción de unos sensores en la xilema, el tejido conductor por el que se transporta la savia desde las raíces hasta las hojas. Mediante este proceso se pude calcular el déficit hídrico del planta. (Fernandez, y otros, 2012)

- ***Termometría de diferencial infrarrojo.***

Ilustración 13 Termómetro Infrarrojo

Es un muy buen indicador del coeficiente de estrés hídrico del cultivo ya que la temperatura está directamente relacionada con el agua disponible para la planta. “Cuando la disponibilidad de agua es limitada, se cierran algunos estomas de la hoja y como resultado la temperatura de la hoja es más alta que la temperatura del aire. En cambio, cuando están abiertos todos los estomas, porque la disponibilidad de agua es adecuada, se observa que la temperatura de la hoja es menor que la del aire." (Luis Gurovich, 2015).

- ***Medidor de potencial de turgencia de la hoja.***

Ilustración 14 Sonda ZIM

El sensor registra la evolución del potencial de turgencia en hoja. A medida que el nivel de estrés hídrico en el árbol aumenta, el potencial de turgencia de sus hojas disminuye. (Fernandez, y otros, 2012).

- *Sensor de crecimiento del fruto.*

Ilustración 15 Sensor de crecimiento

Este sensor controla de manera permanente el crecimiento a partir de la medida milimétrica del diámetro del fruto. Este factor es importante para controlar el estado de maduración del cultivo. (Luis Gurovich, 2015).

2.3. Arduino.

Ilustración 16 Arduino Open-Source

Arduino es una plataforma de código abierto que surgió como una herramienta para el prototipado rápido y fácil, tanto de hardware como de software. Fue desarrollado para estudiantes que no tuviesen mucho conocimientos de programación y electrónica, aunque en la actualidad existe una gran comunidad Arduino que desarrollan y comparten sus proyectos.

Las instrucciones se mandan a la placa mediante un Lenguaje de Programación Arduino (basado en *Wiring)*, a través del software específico de Arduino (IDE), basado en procesos.

Existe una amplia gama tanto de ***board*** (Plataforma o circuito principal de una computadora que integra y coordina todos los sus demás elementos. También es conocida como *placa base*.) como de ***Shields*** (Son placas impresas que se pueden conectar al *Board* para ampliar sus capacidades, pudiendo ser apilada una encima de la otra.) dentro del hardware Arduino, según el uso que se pretenda dar.

Name	Processor	Operating/Input/Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	USB
Due	ATSAM3X8E	3.3 V / 7-12 V	84 MHz	12/0	54 / 12	2 Micro
Gemma	ATtiny85	3.3 V / 4-16 V	8 MHz	1/0	03-feb	Micro
Leonardo	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20-jul	Micro
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14-jun	Regular
Yùn	ATmega32U4 AR9331 Linux	5 V	16 MHz 400MHz	12/0	20-jul	Micro
Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6/0	10-oct	2 Micro

Tabla 1 Características básicas de Arduino Boards

2.4. Bases de Datos.

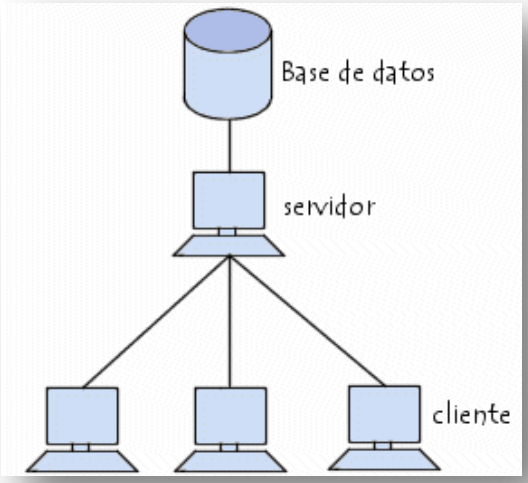

Ilustración 17 Esquema BBDD-Cliente

El término *base de datos* se refiere a una pieza de Software en la cual se puede almacenar, de manera estructurada, diferentes tipos de datos con la menor redundancia posible. Las bases de datos son herramientas intermedias que sirven para conectar distintos usuarios y programas con estos datos. Por lo tanto, el término *base de datos* está relacionado con el de *red,* cuya función es la de compartir su información. De aquí la denominación base. Por otra parte, el nombre *Sistemas de información* se utiliza para englobar todos los mecanismos que se incluyen en la transmisión de los datos que se han instalado.

(Enciclopedia CCM, 2015).

2.5. Cliente ligero.

La definición de cliente ligero engloba tanto un software como un equipo real que utiliza los recursos de otras unidades para hacer la mayor parte de su tarea. Un cliente ligero es un elemento que compone una red, este ejecuta el rol de cliente mientras que el servidor de la red se encarga del trabajo real, puesto que el Cliente no es capaz de realizar muchas de las funciones por sí mismo. Contiene la información básica para su funcionamiento y conexión con el servidor, mientras que el Equipo Servidor proporciona el resto de la potencia de computación.

En términos de software, el cliente ligero es una interface básica, la cual visualiza los datos y utiliza las herramientas como lo haría un sistema normal, aunque realmente es otro programa el que se ejecuta en un servidor remoto y realiza el procesamiento.

Ilustración 18 Servidor de datos

2.5.1. Cliente ligero de mapas

Estos **clientes ligeros** son proyectos que permiten acceder a información geoespacial en ambientes Web. Este tipo de productos son utilizados para construir desde sencillos visores Web, hasta completos geoportales, e incluso herramientas de edición vía Web.

Los clientes ligeros para mapas más potentes en la actualidad son:

- OpenLayers

  OpenLayers es un proyecto desarrollado por la compañía estadounidense MetaCarta, que ha pasado a formar parte de los proyectos incubados en **OSGeo**. OpenLayers es un cliente Web-GIS ligero construido con clases de Javascript, sin dependencia de servidores de mapas concretos.
  Ofrece un interfaz de usuario simplificado que ataca a servicios WMS y WFS de forma transparente para el usuario y desarrollador. Las características por las que destacó OpenLayers en su difusión en la comunidad es la simplicidad de uso, el soporte de tiles y caché y el acceso a mapas de Google Maps y Yahoo Maps. (Pro Developer).

- mapfish

  MapFish es un aplicación web 2.0 mapping compuesta por dos partes: MapFish Cliente y MapFish Servidor. MapFish cliente es un JavaScript framework basado en OpenLayers para el cartografiado de mapas y en ExtJS para la parte GUI (widgets). MapFish Servidor permite llevar la composición y gestión de varios módulos que pueden ser implementados en lenguajes de programación tales como Python, Java, PHP.
  Mapbender. (Pro Developer)**.**

- Leaflet

  Leaflet es una librería JavaScript de código abierto. Está diseñada para simplificar el rendimiento y facilidad de programación incluyendo los procesos más utilizados en la edición de mapas. Soporta tanto plataformas móviles como clientes web. (Leaflet Open-source).

- Mapbender

  Mapbender ha sido desarrollado por un conjunto de programadores y empresas liderados por la organización alemana WhereGroup. Cliente Web-GIS construido con Javascript, que ofrece un interfaz de usuario configurable no dependiente de ningún servidor de mapas concreto. Permite interactuar con servicios WMS, WFS(-T) y WMC. Incluye interfaces de

administración de usuarios, grupos y servicios OGC (OWS). Una característica diferenciadora de Mapbender es que constituye un completo geoportal, con unas facilidades de configuración y administración muy potentes. Mapbender es un proyecto oficial de la Fundación **OSGeo**, con una comunidad bastante abierta muy presente en Alemania. (Pro Developer).

- MapServer
  open source web mapping

MapServer es un motor de renderizado de datos geográficos de Código Abierto escrito en C. Más allá de exploración de datos SIG, MapServer permite crear "mapa de imágenes geográficos", es decir, los mapas que pueden dirigir contenido al usuario. MapServer fue originalmente desarrollado por la Universidad de Minnesota (UMN) con el proyecto ForNet en cooperación con la NASA, y el departamento de recursos naturales (MNDNR). Más tarde fue recibido por el proyecto TerraSIP, un proyecto patrocinado entre la NASA y la UMN y un consorcio de intereses en el manejo de la tierra.
Ahora MapServer es un proyecto OSGeo, y es mantenido por un número creciente de desarrolladores de todo el mundo. (Map Server).

CAPÍTULO 3: Entorno de Estudio.

3.1. Selección del área de estudio.

Para el desarrollo del proyecto, se busca una entorno adecuado donde poder realizar una prueba lo más realista posible del sistema que se desarrolla en el mismo. El sistema que se pretende desarrollar tiene la misma validez para el olivar como para otro tipo de cultivo, como pistacho, almendro, cítrico o viñedo. Aunque debido a la localización del estudio, Provincia de Jaén, se opta por el Olivar puesto que es el cultivo predominante en la zona.

Se busca una extensión que sea lo más representativa posible a las parcelas de la Provincia de Jaén. Es por ello que se elige para este propósito, unas fincas de olivar de secano con una extensión media (7ha). En la *Ilustración 19* se puede ver la información de la parcela servida por el SIGPAC.

Datos Identificativos SIGPAC 2015	
Provincia	23 - Jaén
Municipio	900 - Jaén
Polígono	43
Parcela	119
Coord. Medias ETRS89	
X	424762.523
Y	4190982.85
Información de la parcela	
Uso	Olivar
Superficie(ha)	62.3680
Perímetro	1088.71
Pendiente media (%)	21.96
Coef. Regadío (%)	0.00

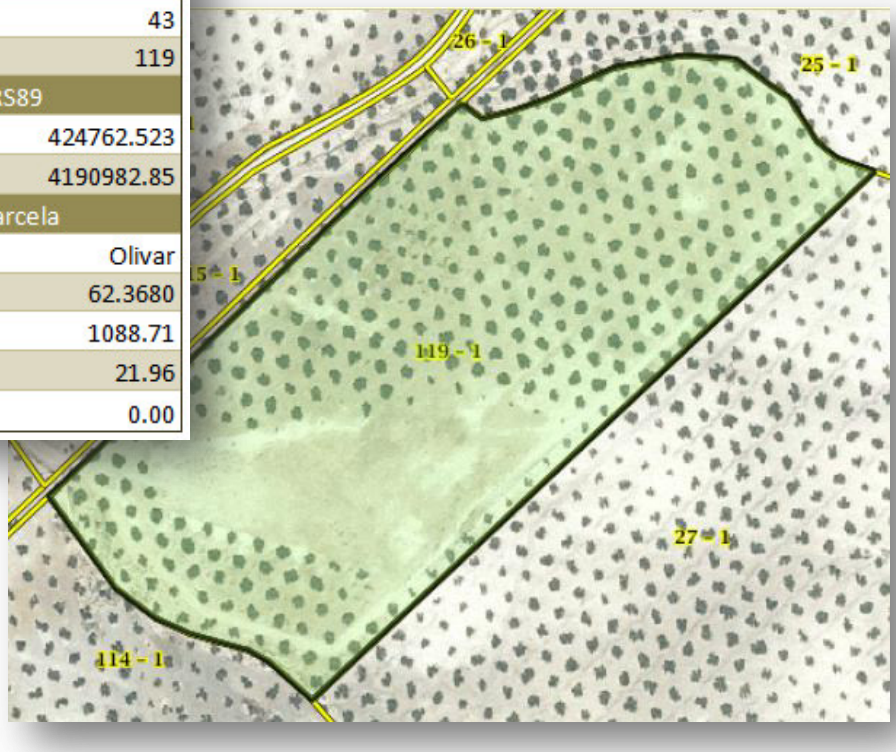

Ilustración 19 Datos Catastrales de la zona de estudio.

3.2. Localización y emplazamiento.

La localización del área de estudio se encuentra dentro de la provincia de Jaén, específicamente dentro del término municipal de Jaén. La zona del cultivo se halla de la campiña jienense, cerca de la delimitación de los términos de Fuerte del Rey, Torredelcampo y Jaén.

La finca se ubica en una ladera entre el *Cerro de los Pijos* y el Cortijo de Los *Barrios* y tiene fácil acceso a la carretera A-311 (Jaén-Andújar), mediante su intersección con el Camino a Mengíbar, el cual trascurre por los límites de la finca.

En la *Ilustración 20* se puede observar la situación de la zona de estudio respecto a las poblaciones más cercanas y su emplazamiento en su entorno.

Ilustración 20 Situación y Emplazamiento

CAPÍTULO 4: Material y Método.

En este capítulo se hace referencia a las técnicas y equipos empleados en la experimentación y desarrollo del sistema de captura, así como, a los diferentes software que se han ido utilizando durante todo el proceso.

4.1. Material.

Para la elección de los diferentes elementos del módulo de captura de datos se han tenido en cuenta compatibilidad, resistencia, resolución y tipo de comunicación. Siempre pretendiendo realizar un prototipo Low-cost pero que cumpliera con los objetivos propuestos.

4.1.1. Microcontrolador.

Ilustración 21 Arduino UNO

Para la elaboración del hardware del sistema se utiliza una placa Arduino UNO. Este tipo de placa es especialmente útil para el prototipito del hardware debido a su versatilidad y facilidad de conexión con los periféricos.

Características de la placa:

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Ilustración 22 Características Arduino UNO

4.1.2. Sensores.

- ***FC-28***

  Este sensor mide la Humedad del Suelo en su alrededor. Es un sensor analógico que se compone por dos sondas, que crean una corriente eléctrica entre ellas, y mide la resistencia a esta corriente. La humedad del suelo ayuda a la conductividad, por lo que a mayor agua en el suelo menor resistencia eléctrica. (Moisture Sensor).

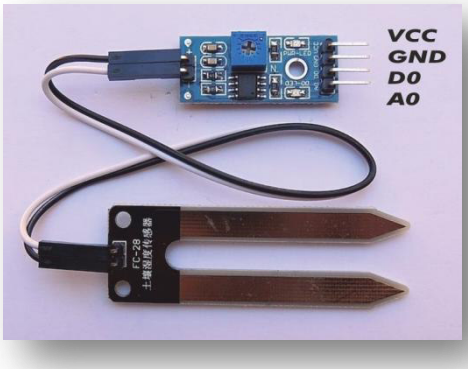

Ilustración 23 Distribución de Pins en Sensor FC-28

Especificación	
Fuente de alimentación:	3.3V o 5V
Voltaje de salida	0 ~ 4.2v
Corriente	35mA

Tabla 2 Características FC-28

- ***DS18B20***

  Es un sensor digital de temperatura de la casa Dallas Semiconductor. La información del DS18B20 es enviada a través del protocolo 1-wire, la cual permite el envío de los datos de varios sensores a través del mismo pin de entrada. La medida se toma en la cabecera metálica del sensor, que es impermeable, lo que permite la medida de la temperatura en el interior del suelo sin ningún problema. (Dallas Semiconductor).

Ilustración 24 Sensor DS18B20

R1	R0	Thermometer Resolution	Max Conversion Time
0	0	9 bit	93.75 ms
0	1	10 bit	187.5 ms
1	0	11 bit	375 ms
1	1	12 bit	750 ms

Ilustración 25 Parámetros de configuración del sensor BS18B20.

- ***LDR***

  Un sensor LDR es un fotorresistor analógico que varía el valor en función de la cantidad de luz que incide sobre él. Cuando no se incide luz sobre el sensor, toma medidas muy altas, mientras que al incidir la luz sobre él, los valores son bajos. (Girod, 2013).

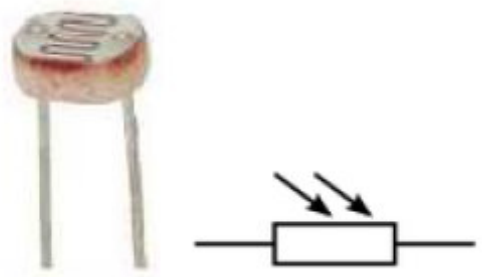

Ilustración 26 Representación eléctrica de una fotorresistencia

Especificación	
Resistencia (con luz)	~1k Ohm
Resistencia (oscuridad):	~10k Ohm
Vmax	150V
Disipación:	100mW max

Tabla 3 Características Sensor LDR

- ***DHT11***

  Se trata de un sensor de señal digital compuesto por sensores de temperatura y humedad. Su tecnología garantiza una alta fiabilidad y estabilidad a largo plazo. El sensor incorpora un microcontrolador de 8-bit en la que el fabricante almacena la calibración de los sensores, de tal modo que las lecturas del sensor están perfectamente calibradas. (Temperature and Humidity Sensor).

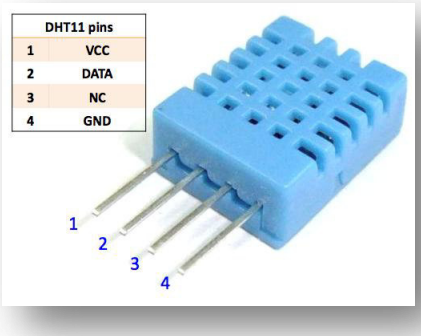

Ilustración 27 Distribución Pins Sensor DHT11

Especificación	
Alimentación	3Vdc ≤ Vcc ≤ 5Vdc
Rango de medición temperatura	0 a 50 °C
Precisión temperatura	±2.0 °C .
Rango de medición de humedad	20% a 90% RH.
Precisión humedad	4% RH.

Tabla 4 Características Sensor DHT11

4.1.3. Shields.

- ***MicroSD Card***

  Es una simple solución para la trasferencia de datos entre Arduino y una tarjeta SD estándar. Como el módulo SD, precisa una gran cantidad de trasferencia de datos y utiliza la librería SPI para la comunicación con el microcontrolador. El hardware de SPI utiliza pines mucho más rápidos que bit-banging.

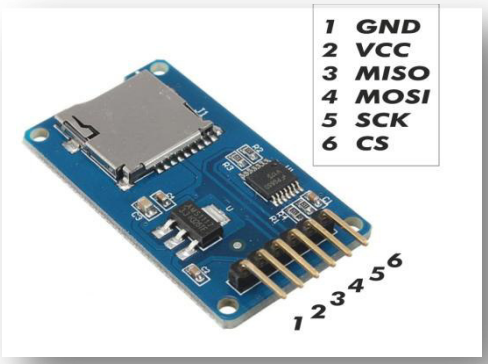

Ilustración 28 Distribución Pins en Shields MicroSD

Especificación	
Voltaje de funcionamiento:	4.5 ~ 5.5 V VCC
Corriente	0,2 ~ 200mA
nivel eléctrico Interfaz	3.3V / 5V
interfaz estándar	SPI

Tabla 5 Características Shields MicroSD

4.1.4. Esquema.

En este apartado se muestra el esquema del prototipo de hardware para la captura de parámetros ambientales. Tanto el Shields Micro SD, como los sensores DHT11 y DS18B20 utilizan las entradas Digital. Mientras que, los sensores FC-28 y LDR, al igual que el LED de notificación, utilizan las entradas analógicas a la placa.

El Pin de alimentación (Vcc), así como el Pin a tierra (GND) de cada periférico son reagrupados ayudados de las conexiones horizontales de la *protoboard* y, posteriormente, comunicados con la placa de Arduino.

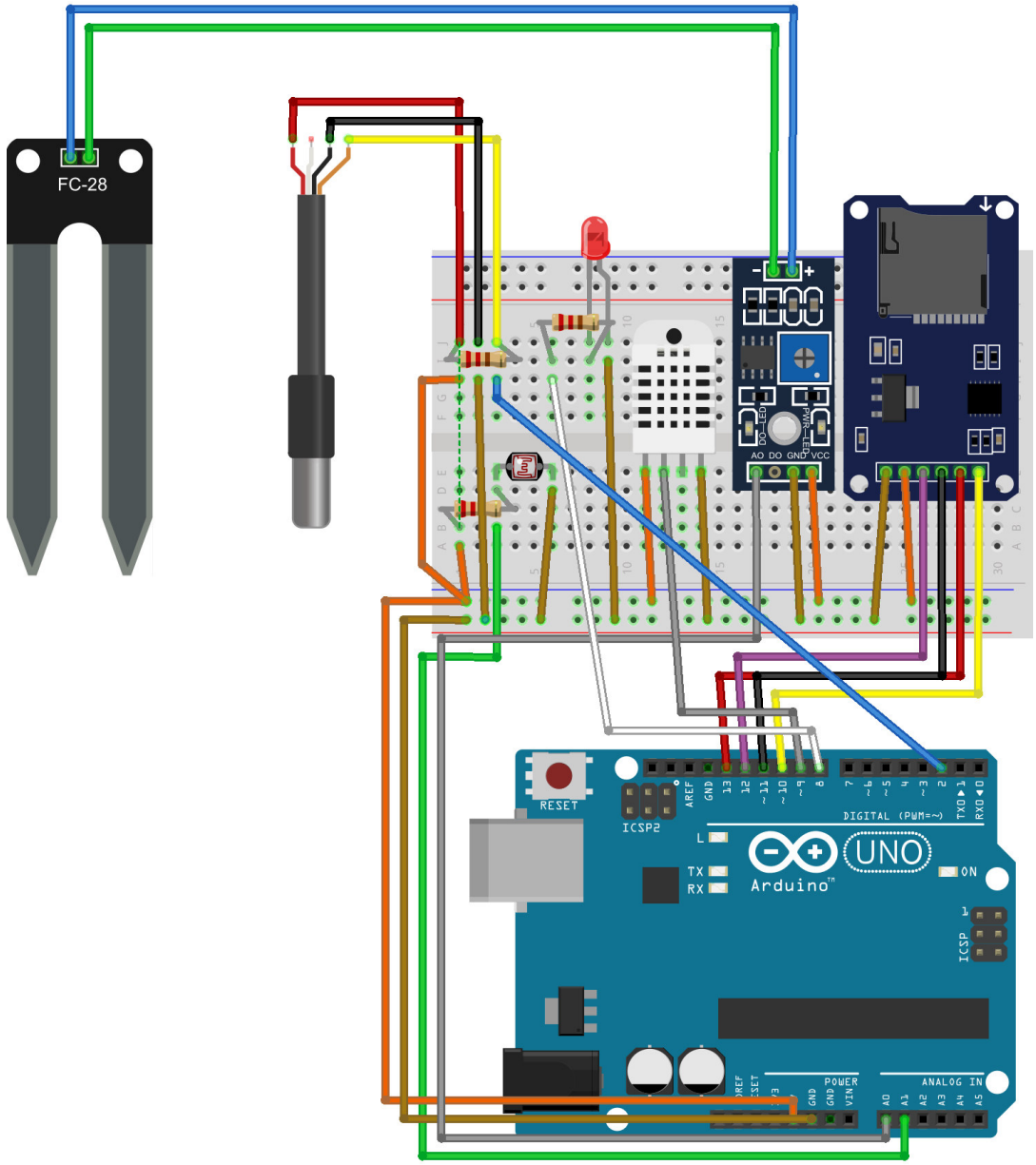

Ilustración 29 Esquema del sistema de captura de información.

4.1.5. Circuito Eléctrico.

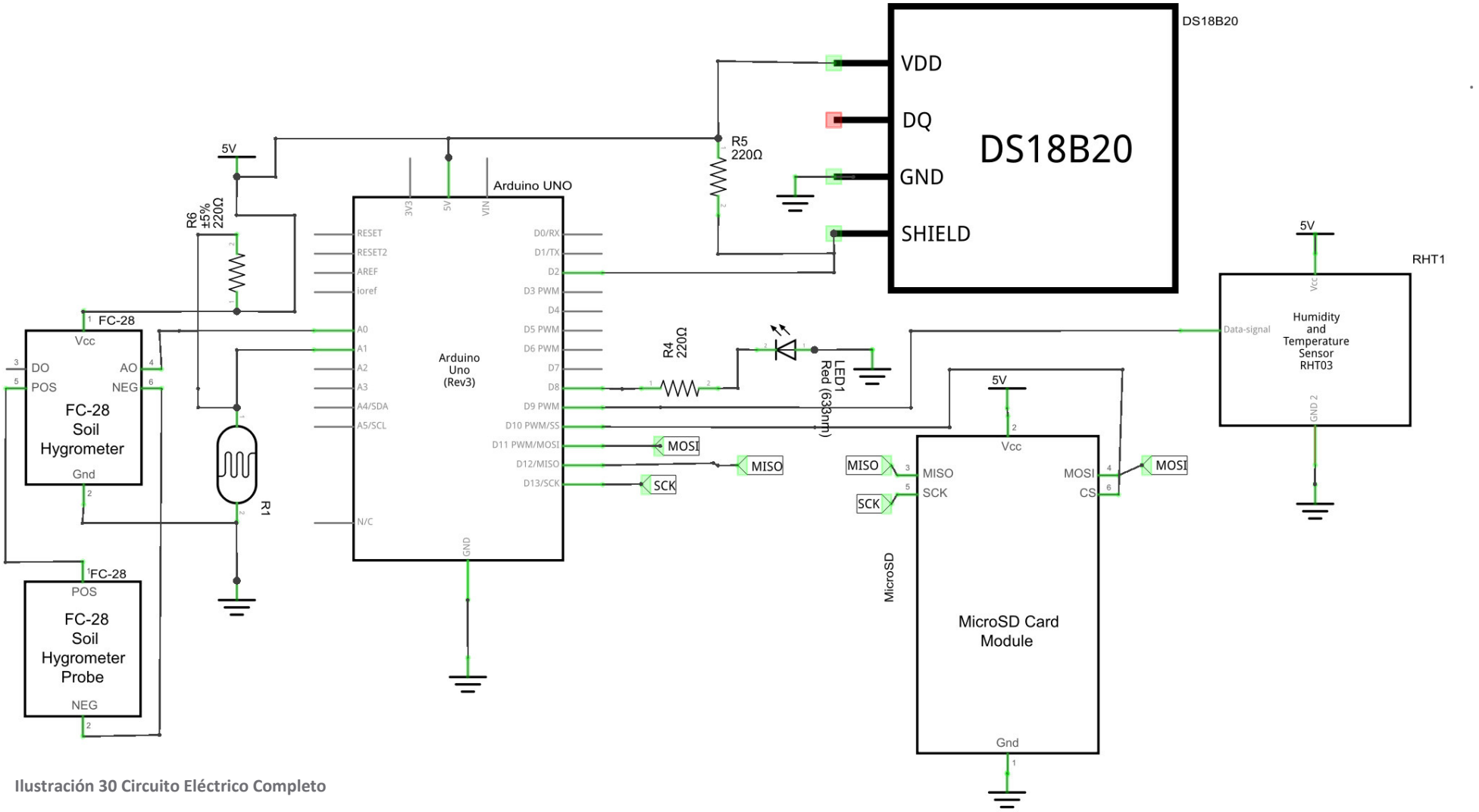

Ilustración 30 Circuito Eléctrico Completo

4.2. Software.

La parte del hardware del sistema no tendría ninguna utilidad sin el empleo de un software que programe y gestione todo el flujo de información desde su captura hasta su visualización final. En este proyecto se han empleado únicamente software de código abierto, los cuales están al alcance de cualquier usuario.

4.2.1. Arduino 1.6.6.

Arduino 1.6.6 es la última versión de Arduino Software (IDE), se trata de un programa de código abierto que hace que sea fácil programar el código y subirlo a la *Board*. El entorno del software está escrito en Java y basado en procedimientos. (Arduino, 2015).

4.2.2. PostgreSQL/Postgis.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa *multiprocesos* en vez de *multihilos* para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Para la gestión de la información dentro de la Base de Datos se utiliza el lenguaje SQL:

El SQL es el lenguaje estándar ANSI/ISO de definición, manipulación y control de bases de datos relacionales. Es un lenguaje declarativo: sólo hay que indicar qué se quiere hacer. En cambio, en los lenguajes procedimentales es necesario especificar cómo hay que hacer cualquier acción sobre la base de datos. El SQL es un lenguaje muy parecido al lenguaje natural; concretamente, se parece al inglés, y es muy expresivo. Por estas razones, y como lenguaje estándar, el SQL es un lenguaje con el que se puede acceder a todos los sistemas relacionales comerciales. (Escofet, 2014).

4.2.3. GeoServer.

GeoServer es un Servidor Web que permite servir mapas y datos de diferentes formatos para aplicaciones Web, ya sean clientes ligeros o programas GIS de escritorio. Esto significa que puedes almacenar datos espaciales en casi cualquier formato que desees. (OSGeo, 2015).

GeoServer soporta numerosos estándares (OGC):

- Web Map Service (WMS)
- Web Feature Service (WFS), WFS-T (transaccional)
- Web Coverage Service (WCS)
- Filter Encoding (FE)
- Style Layer Descriptor (SLD)
- Geography Markup Language (GML)

4.2.4. Openlayer.

OpenLayers es una librería JavaScript que permite incluir mapas georreferenciados en cualquier página web.

Al ser una librería del lado del cliente, la descarga de estos se realiza directamente desde el navegador a través de Ajax; no genera tráfico en el servidor, los mapas se descargan directamente del servidor de mapas que suele ser una pieza diferenciada a nivel de sistemas.

OpenLayers permite sobreponer distintas capas sobre una básica, añadir indicadores o puntos en el mapa con leyendas, así como polígonos y proporciona su propio api para dibujarlos de una manera sencilla.

Incorpora un set de controles básicos y una *toolbar* de controles avanzados y se puede incluir controles propios haciendo uso del API. (Suárez, 2015).

CAPÍTULO 5:
Diseño e Implementación.

El sistema propuesto en este proyecto representa el ciclo de procedimientos necesarios desde la captura de los datos hasta la ayuda en la toma de decisiones del usuario final (Véase *Ilustración 31*).

- La captura de la información se realiza mediantes sensores programados con **Arduino**.
- La información de los sensores es registrada en archivos de texto y es importada a la base de datos **Postgres**, donde se almacena y gestiona.
- La geolocalización de los datos se realiza a través de **Postgis**.
- Las diferentes capas de información son agregadas a un Servidor de Mapas, **GeoServer**.
- La visualización de los resultados se hace desde un **Cliente Ligero**.

Ilustración 31 Ciclo del Sistema

5.1. Desarrollo del código de Arduino.

El lenguaje de programación de Arduino tiene una estructura bastante simple que se compone en su estructura más básica por dos partes o funciones. Estas dos funciones encierran bloques que se componen de declaraciones, instrucciones o estamentos.

```
void setup(){            //Primera Parte
  Instrucciones;
}
void loop(){             //Segunda Parte
  Instrucciones;
}
```

5.1.1. Librerías.

Las librerías son una colección de programas que facilitan la ejecución de una serie de funciones relacionadas.

#include <Time.h>

La libreria "Time.h" permite obtener la fecha y hora (hour(); minute(); second(); day(); weekday(); month(); year();) en arduino, con o sin necesidad de un hardware externo de cronometraje. La librería permite contabilizar el tiempo una vez inicializada una fecha.

#include <SD.h>

Sirve para la lectura y escritura de tarjetas Micro SD. La librería soporta archivos FAT16 y FAT32 en los estándares SD Cards y SDHC Cards. La comunicación entre el microcontrolador y la tarjeta SD se realiza mediante SPI.

#include <SPI.h>

SPI (Serial Peripheral Interface) es un protocolo de envío de datos del microcontrolador para comunicarse con uno o varios periféricos a corta distancia.

#include "DHT.h"

Es la librería que controla a la familia de sensores DHT (Humedad y temperatura).

#include <OneWire.h>

1-Wire es un protocolo de envío de información por el cual se consigue mandar y recibir datos a través de un solo cable. El sistema permite el control de varios periféricos mediante un único cable de información, esto es posible ya que cada uno de estos es identificado con un address.

#include <DallasTemperature.h>

Es la librería que controla el sensor DS18B20 (temperatura), utiliza el protocolo 1-wire para la comunicación entre el sensor y el microcontrolador.

5.1.2. Void Setup().

Es la parte encargada de la configuración. Es la primera función a ejecutar en el código, se ejecuta una sola vez, y debe de contener la declaración de las variables y se inicializan las comunicaciones.

En la función Setup() se establece la comunicación con el monitor serial (9600) para mostrar la información en la pantalla.

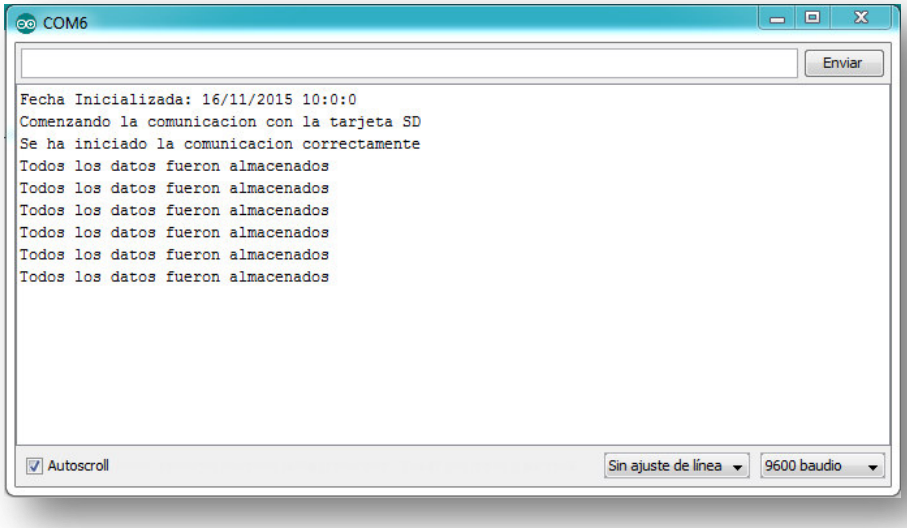

Ilustración 32 Monitor Serial de Arduino

Arduino Uno no posee un reloj interno, sólo es capaz de medir intervalos de tiempo (milisegundos) por lo que es necesario inicializar una fecha con la función setTime(), a partir de la cual se calcula el momento en que se efectúan las medidas.

La función abre la comunicación con la Tarjeta SD, crea un archivo específico para cada sensor, en este caso JAEN001.csv., y escribe la siguiente cabecera dentro del documento:

```
Placa de Arduino Nº1

Medidas del sensor: DHT11 LDR FC-28 DS18B20

Fecha iniciada: 23/9/2015 10:0:0
```

```
ID, Fecha, Hora, Humed_Amb(%), Temp_Amb(ºC), Luminosidad(lux), Humed_Suelo(%),  Temp_Suelo(ºC)
```

Por último, se cierra el archivo JAEN001.csv, se inicializan los sensores y se definen los PIN de entrada y de salida.[1]

5.1.3. Void Loop().

Esta parte es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo. La función Loop() contiene las instrucciones que se repiten de manera permanente en el programa.

Lo primero que realiza la función es guardar el instante en el que se realizarán las medidas, mediante la función:

```
time_t t = now();
```

Del mismo modo se consultan los sensores y se guardan sus medidas en sus variables correspondientes.

Las entradas analógicas proporciona una medición codificad en forma de un valor digital con un número N de bits. En el caso de los sensores analógicos utilizados realiza la medida entre un intervalo de 0 a 1023. Por ello se realiza una trasformación de los valores medidos a una escalas más cuantificables. En el caso del sensor LDR se utiliza el siguiente código para transformar la medida analógica en lux.

```
V = analogRead(LDRPin);   // valor medido por el sensor LDR

      //ilum = ((long)(1024-V)*A*10)/((long)B*Rc*V);
ilum = ((long)V*A*10)/((long)B*10*(1024-V)); // trasformamos la medida en lux
```

Posteriormente, se abre el archivo JAEN001.csv y se guarda el contenido de las variables utilizando el mismo formato que la Tabla *"Datos_Sensores"* de la Base de Datos, donde posteriormente se importará. Se incrementa en uno el identificador y se cierra el archivo.

Los datos de los sensores son almacenados siguiendo el orden que se establece en la cabecera:

```
0  23/9/2015   10:0  34.00  26.00  797  20  26.50
1  23/9/2015   10:0  34.00  26.00  797  21  26.50
2  23/9/2015   10:0  34.00  26.00  797  22  26.50
3  23/9/2015   10:0  34.00  26.00  843  25  26.50
4  23/9/2015   10:0  34.00  26.00  855  20  26.50
5  23/9/2015   10:0  34.00  26.00  831  19  26.50
```

[1] El código completo de Arduino se puede consultar en los Anexos del documento.

```
6   23/9/2015    10:0   34.00   26.00   843   15   26.50
7   23/9/2015    10:0   34.00   26.00   855   10   26.50
8   23/9/2015    10:0   34.00   26.00   808   2    26.50
9   23/9/2015    10:0   34.00   26.00   797   1    26.50
10   23/9/2015    10:0   34.00   26.00   808   2    26.50
```

Cuando se han grabado los datos de los sensores en el archivo, se enciende un Led Rojo durante un segundo, utilizando delay(1000).

Por último, mediante otro delay() se establece el tiempo que tardará la función voop() en realizar una nueva iteración.

5.2. Desarrollo del Servidor.

Para el desarrollo del sistema es necesaria una plataforma que almacene y administre la información proveniente del hardware desarrollado. La bases de dataos y el servidor web proporcionan los servicios y la infraestructura necesarios para la finalidad del proyecto.

5.2.1. Creación de una Database.

El primer paso es crear una nueva Database en Postgres 9.4, para ello utilizaremos el administrador gráfico PgAdmin III.

Una vez dentro de nuestra Base de datos, la cual llamaremos Postgres, se agregan las extensiones de Postgis y se crearán las tablas necesarias para la gestión del sistema, dentro del Schema Public:

Table datos_sensores: Esta tabla recoge todos los datos que se almacenan en la Tarjeta SD de los diferentes sensores de la red.

```
CREATE TABLE public.datos_sensores
(
  cod_sensor character varying(10),
  id integer,
  fecha_hora timestamp without time zone,
  hum_amb integer,
  temp_amb integer,
  lum integer,
```

```
  hum_suelo integer,
  temp_suelo integer
)
```

Table sensores: Esta tabla almacena la información de los diferentes sensores de la red, tanto los datos identificadores como la geolocalización de los mismos.

```
CREATE TABLE public.sensores
(
  cod_sensor character varying(10),
  nombre character varying(50),
  geom geometry(Point,4258),
  CONSTRAINT sensores_pkey PRIMARY KEY (gid)
)
```

La columna "geom" almacena la geométrica proveniente de Postgis, guarda en la tabla la información de una capa puntual con las coordenada de los sensores en EPSG:4258 (ETRS89).

5.2.2. Importación de los datos.

La información recogida por los sensores se importa manualmente a la tabla datos_sensores. El archivo .csv de cada sensor contiene una estructura similar a la de la tabla, de modo que se puede importar de manera simultánea toda la información del archivo.

Para la importación de los datos se utiliza la función COPY:

```
COPY datos_sensores FROM 'c:/TFM/JAEN001.csv' DELIMITERS ',' CSV;
```

5.2.3. Selección de la información.

La tabla *datos_sensores* recoge toda la información de los distintos sensores, la cual se va almacenando según el orden de importación. Para la gestión de los datos y la combinación de las diferentes tablas se utiliza la función SELECT.[2]

```
SELECT sensores.cod_sensor, sensores.nombre, sensores.geom, t1.fecha_hora, datos_sensores.id,
datos_sensores.hum_amb, datos_sensores.temp_amb, datos_sensores.lum, datos_sensores.hum_suelo,
datos_sensores.temp_suelo

FROM sensores, datos_sensores,
```

[2] Todas las sentencias SQL empleadas se pueden consultar en los Anexos del documento.

```
                (SELECT cod_sensor, MAX(fecha_hora)as fecha_hora FROM datos_sensores GROUP BY
cod_sensor) as t1

WHERE sensores.cod_sensor = t1.cod_sensor AND datos_sensores.fecha_hora = t1.fecha_hora

ORDER BY sensores.cod_sensor
```

Con este código se consigue combinar la información de la tabla *datos_sensores* con la geometría de la tabla sensores. Se utiliza como identificador de unión la columna *cod_sensor* que encuentra en las dos tablas. Además con una subconsulta se seleccionan el valor de cada sensor con la fecha más alta.

Output pane

Data Output | Explain | Messages | History

	cod_sensor character v	nombre character varying(50)	geom geometry(Point,4258)	fecha_hora timestamp without time	id integer	hum_amb integer	temp_amb integer	lum integer	hum_sue integer	temp_su integer
1	JAEN001	JAEN LOS BARRIOS	0101000020A21000079E92	2015-09-23 12:09:00	127	36	22	407	0	85
2	JAEN002	JAEN LOS BARRIOS	0101000020A2100000C1CAA	2015-09-23 11:18:00	77	36	22	400	0	85
3	JAEN003	JAEN LOS BARRIOS	0101000020A2100000C976B	2015-09-23 11:26:00	85	36	22	417	0	85
4	JAEN004	JAEN LOS BARRIOS	0101000020A21000003F355	2015-09-23 12:02:00	120	36	22	407	0	85
5	JAEN005	JAEN LOS BARRIOS	0101000020A2100000000	2015-09-23 10:25:00	25	36	22	420	0	85
6	JAEN006	JAEN LOS BARRIOS	0101000020A210000023DBF	2015-09-23 10:11:00	11	35	22	516	0	85
7	JAEN007	JAEN LOS BARRIOS	0101000020A2100000508D9	2015-09-23 10:08:00	8	35	23	516	0	85
8	JAEN008	JAEN LOS BARRIOS	0101000020A2100000643BD	2015-09-23 10:42:00	42	36	22	410	0	85
9	JAEN009	JAEN LOS BARRIOS	0101000020A21000001B6A8	2015-09-23 10:56:00	55	36	22	414	0	85
10	JAEN010	JAEN LOS BARRIOS	0101000020A21000001B2FD	2015-09-23 10:02:00	2	35	24	553	2	85
11	JAEN011	JAEN LOS BARRIOS	0101000020A2100000B29DE	2015-09-23 11:09:00	68	36	22	407	0	85

Ilustración 33 Resultado de la búsqueda SQL

5.2.4. Servidor Web de mapas.

Para la posterior visualización de las distintas capas en el cliente ligero, se crea un servidor de mapas con GeoServer.

El primer paso en GeoServer es crear un espacio de trabajo, el cual contendrá los diferentes servicios del proyecto.

Ilustración 34 GeoServer, Espacio de trabajo.

Dentro del espacio de trabajo, se crean los distintos Almacenes de Datos que contendrán los diferentes orígenes de datos (Shapefiles, Postgis, WMS…).

Ilustración 35 GeoServer, Almacén de datos.

Las capas del proyecto se agregan como un nuevo recurso de un Almacén de trabajo, y es necesario definir el Nombre, la Proyección y el Estilo de la capa. En la *Ilustración 36* se observa cómo cada capa debe ser habilitada para que su visualización se realice con éxito.

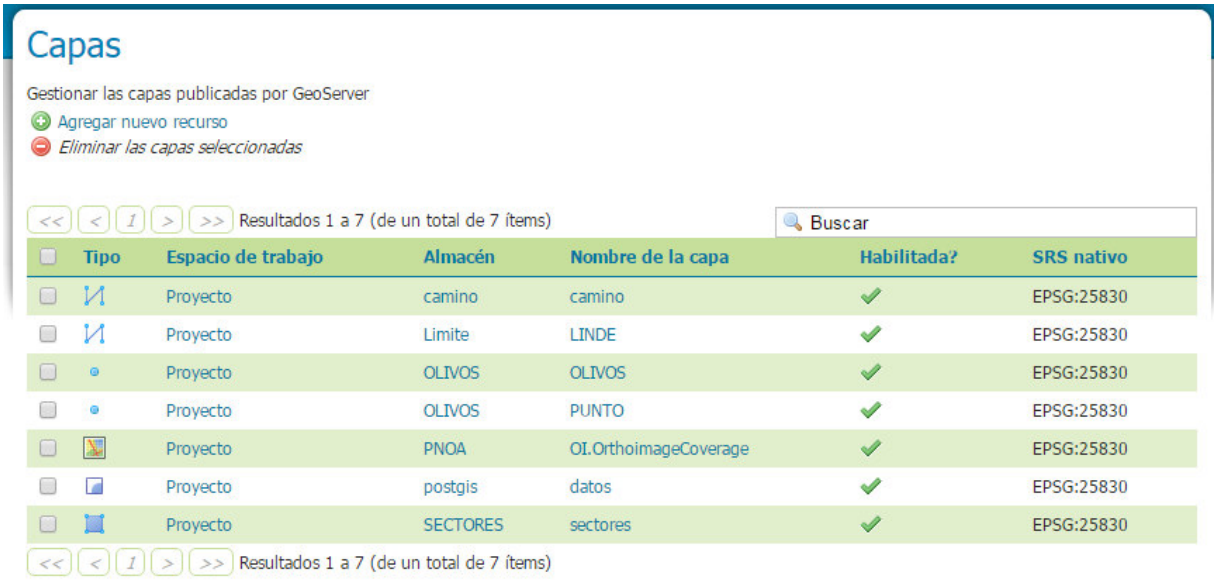

Ilustración 36 GeoServer, Editor de capas.

En el caso de la capa "datos", agregada desde el almacén Postgis, se añade la consulta SQL, que se muestra en el apartado anterior, para filtrar dentro de la tabla "Datos_sensores" los últimos valores añadidos de cada sensor. Como se muestra en la *Ilustración 37,* GeoServer interpreta la consulta SQL e incluye los diferentes atributos de la capa.

También se tiene la posibilidad de definir parámetros en la vista SQL, que se declaran como '%Parámetro%' en la sentencia SQL, y posteriormente tanto sus valores por defecto como su expresión regular.

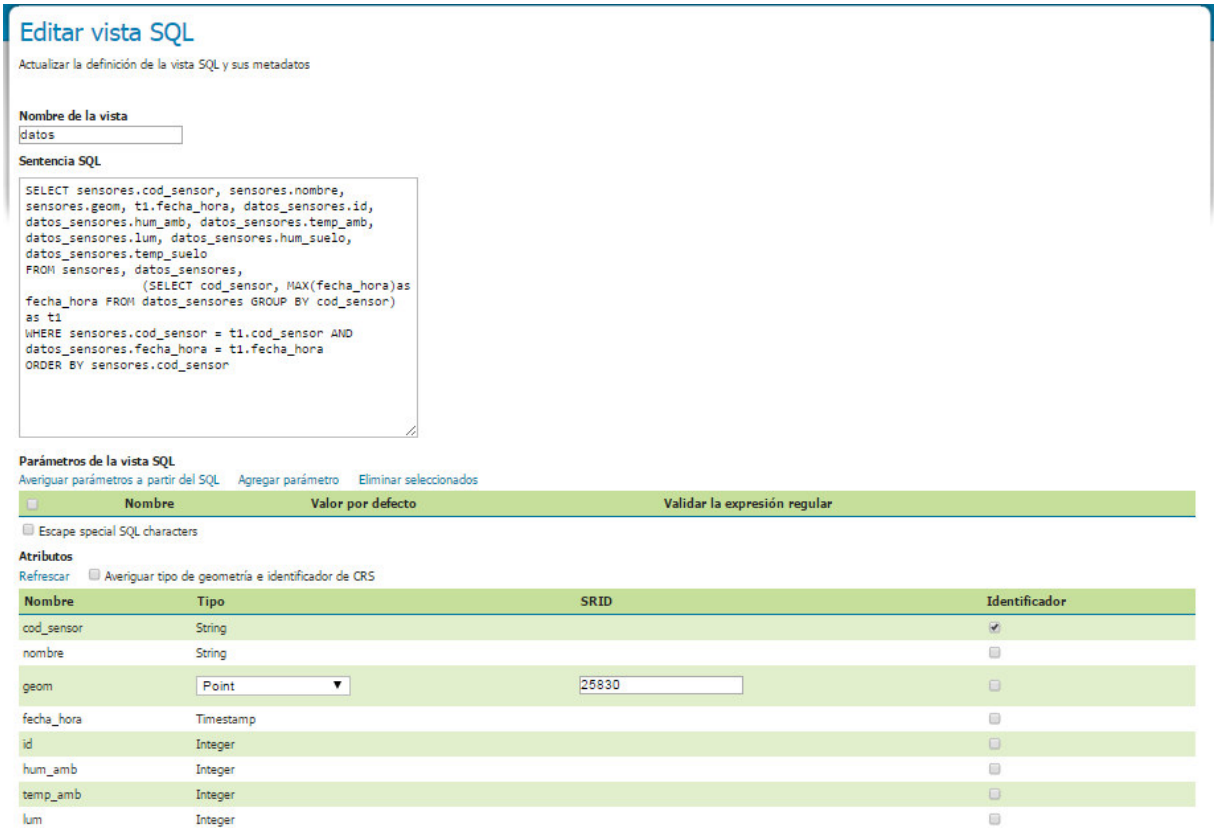

Ilustración 37 GeoServer Añadir vista SQL

Los estilos de las capas se añaden a GeoServer mediantes archivos SLD (Styled Layer Descriptor), estos archivos son esquemas XML propuestos por OGC para definir el estilo visual de cada una de las capas de objetos geográficos que componen el mapa.[3]

En la *Ilustración 38* se muestra la edición del estilo SLD de la capa vectorial "Camino".

[3] Los estilos SLD de las diferentes capas se encuentran en el anexo que acompaña al proyecto.

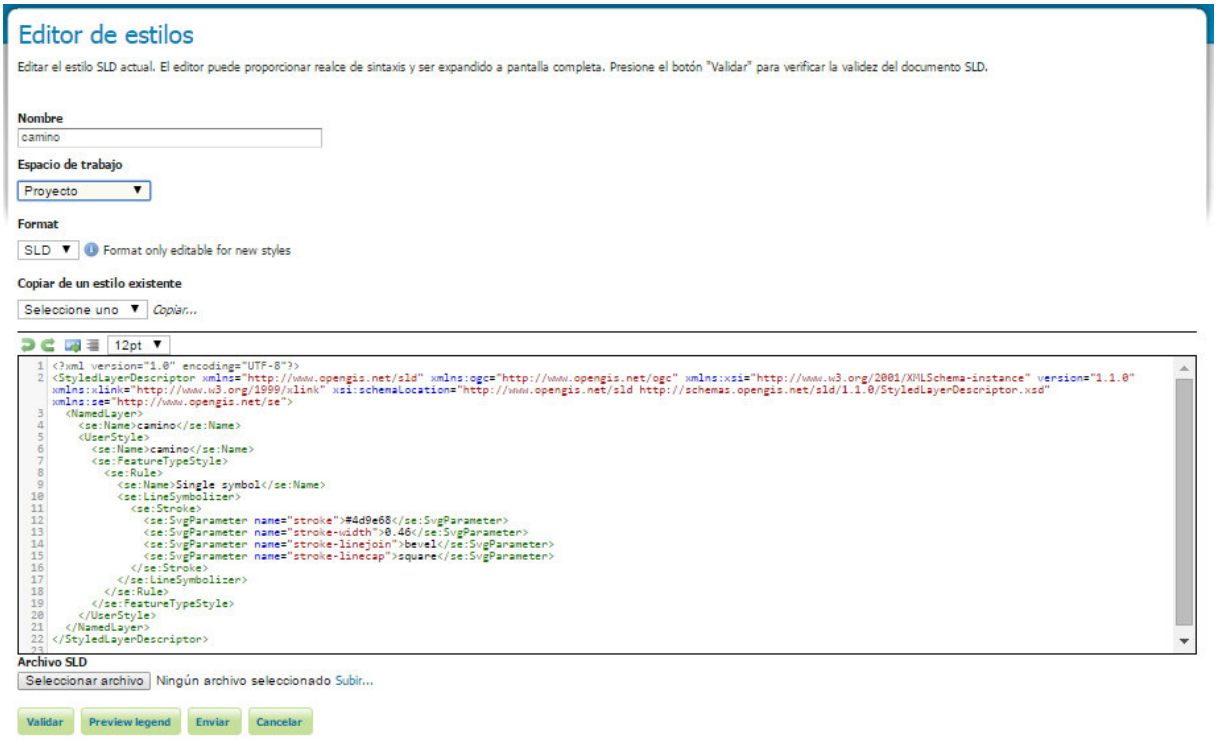

Ilustración 38 GeoServer, editor de estilos SLD

5.3. Desarrollo del Cliente Ligero.

El cliente ligero es la última parte del proyecto, donde poder visualizar los resultados obtenidos del sistema. El cliente es una página programada en HTML, la cual puede ser ejecutada por cualquiera de los navegadores o clientes de acceso a internet.

5.3.1. Programación en HTML.

Html es un lenguaje de marcas para la elaboración de Páginas web. Los archivos HTML se componen por Tags o etiquetas, que describen el contenido del documento. El contenido del código queda inscrito dentro del <HTML>, el cual incluye dos elementos básicos: Head y Body.

- La etiqueta <Head> contiene los metadatos del documento. Esta información normalmente define titulo, estilos, link, scripts, css y otra meta información del archivo.
- La etiqueta <Body> se usa para indicar el contenido del documento html.

Un ejemplo básico puede ser el siguiente:

```
<HTML>
  <HEAD>
    <TITLE>Hola mundo</TITLE>
  </HEAD>
  <BODY>
    <P>Hola Mundo</P>
  </BODY>
</HTML>
```

5.3.2. Programación en JavaScript.

La librería OpenLayer se declara en la cabecera mediante el código

```
<script src="http://www.openlayers.org/api/OpenLayers.js"></script>
```

A continuación se programa el <script> que gestionará el mapa de la página.

Se establecen las opciones del mapa (Proyección, Unidades, Extensión…) y se define la variable "map" que contendrá el mapa de Openlayer .

```
map = new OpenLayers.Map("divMapa", options)
```

Se añaden las capas WFS que contendrá el mapa, añadiéndolas de una en una desde sus respectivos orígenes:

- Orto: Ortofotografía de PNOA usada como capa base.
- Catastro: Capa vectorial de las parcelas catastrales.
- Aerial: Capa raster proveniente de bing map.
- Sectores: Capa vectorial de las diferentes divisiones en las que se estructura la finca y que corresponde con uno de los sensores de la red.
- Camino: Capa vectorial de las vías de acceso de la finca.
- Datos: Capa puntual que representa la posición de los sensores de la red y que contiene los últimos datos recolectados por los sensores.

Las capas son añadidas al mapa a través de la función:

```
map.addLayers([orto,catastro,aerial,sectores,camino,datos])
```

Una vez definidos todos los elementos del mapa se añaden los controles de este. Que permitirán una interlocución fácil y dinámica al usuario de la web.

Por último, se crea la variable "info", la cual contiene un evento *getFeature*. Éste se encarga de realizar una petición a Geoserver para obtener la información de la capa "datos". La función se muestra a través de un *popup* que aparecerá en el mapa una vez se realice un *click* en alguno de los sensores de la capa.[4]

```
info = new OpenLayers.Control.WMSGetFeatureInfo({
      url: 'http://localhost:8080/geoserver/Proyecto/wms',
      title: 'Identify features by clicking',
```

[4] El código completo del Scipt se puede consultar en los Anexos del proyecto.

```
        layers: [datos],
        queryVisible: true,
        eventListeners: {
                getfeatureinfo: function(event) {
                        map.addPopup(new OpenLayers.Popup.FramedCloud(
                                "chicken",
                                map.getLonLatFromPixel(event.xy),
                                new OpenLayers.Size(200,200),
                                event.text,
                                null,
                                true
                        ));
                document.getElementById("info").innerHTML=event.text;
                }
        }
});
map.addControl(info);
info.activate();
```

Cuando se ejecuta la función "Info" (Véase *Ilustración39*), se realiza una petición getFeauteInfo a Geoserver. Éste recibe la petición, como se muestra en la siguente imagen, y manda los datos de la capa a un *popup* que se mostrará en el mismo mapa.

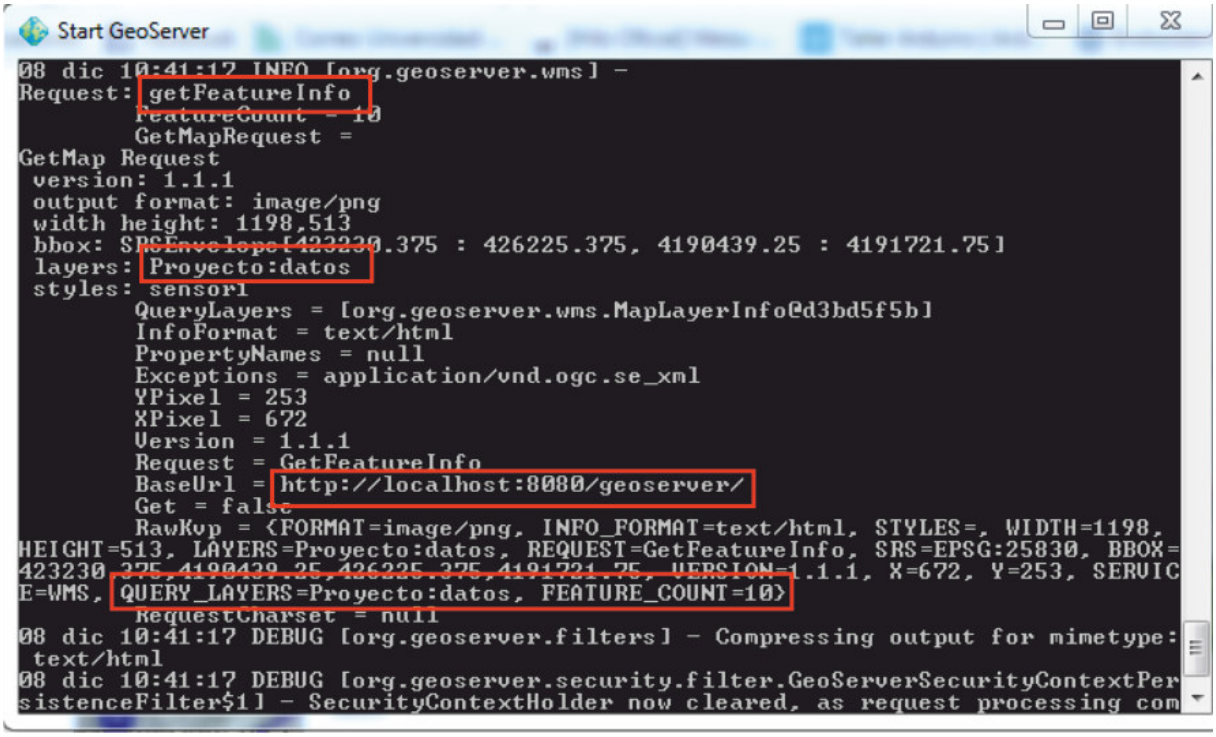

Ilustración 39 Petición de GetFeatureInfo

Para la ejecución del sistema se supondrá un entorno local bajo el navegador "Google Chrome". Para ello será necesario un mecanismo que habilite las peticiones entre el navegador local y el Servidor Web de Mapas (GeoServer) y que, por lo tanto, permita a determinados dominios realizar peticiones AJAX cruzadas sobre dominios ajenos.

Por razones de seguridad, navegadores restringen las peticiones HTTP de origen cruzado iniciados desde dentro de scripts. Para la solución de este problema, se recurre a la extensión "Allow-Control-Allow-Origin" de Google Chrome. Este recurso permite realizar peticiones "XMLHttpRequests" cuando el recurso viene de un dominio diferente del que realizo la petición (JavaScript). Por lo tanto, se permiten a aplicaciones web, bajo un determinado dominio, hacer peticiones AJAX de dominio cruzado sobre dominio ajeno.

5.4. Implementación.

A continuación se muestran los resultados obtenidos por el sistema diseñado en el proyecto. El largo proceso en el desarrollo del prototipo para el módulo de captura y la falta de tiempo y recursos ha provocado que no se haya podido desarrollar un módulo por cada nodo de la red de sensores y, por ello, las posteriores pruebas se realizarán con una simulación de los previsibles datos capturados.

5.4.1. Modulo de captura de datos.

La captura *in situ* de la información en la zona de estudio se lleva a cabo a partir del prototipo de hardware desarrollado en el proyecto. Para la implementación de la red de sensores es necesario posicionar un módulo por cada sector en los que se ha dividido el área de trabajo.

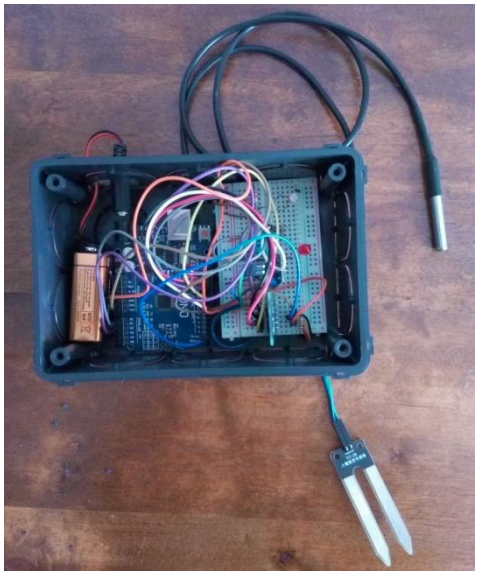

Ilustración 40 Contenido del Módulo.

Ilustración 41 Carcasa del Módulo.

Ilustración 42 Módulo Instalado.

Tanto el microcontrolador como los sensores y la *protoboard* se alojan dentro de una carcasa que los proteja de las inclemencias del tiempo pero que, a su vez, permita la captura del los parámetros ambientales. Ésta incluye en su parte superior una lámina de acetato transparente que permite la lectura del sensor LDR.

5.4.2. Aplicación Web.

Los resultados del sistema propuesto se pueden consultar de manera interactiva por el usuario final a través de un Cliente Ligero que es gestionado por una Aplicación Web.

Para la fase de prueba del proyecto se ha implementado una web (SmartOlive.com) a través de un Servidor HTTP Apache, el cual se utiliza en modo local con el fin de pre-visualizar y probar código mientras éste es desarrollado.

Al cargar la web aparece la pantalla de Inicio como se muestra en la *Ilustración 43*. Esta primera página, además de mostrar un sistema de registro (Usuario y Contraseña) y las diferentes secciones de la web, contiene información de Smart Olive y un vídeo corporativo.

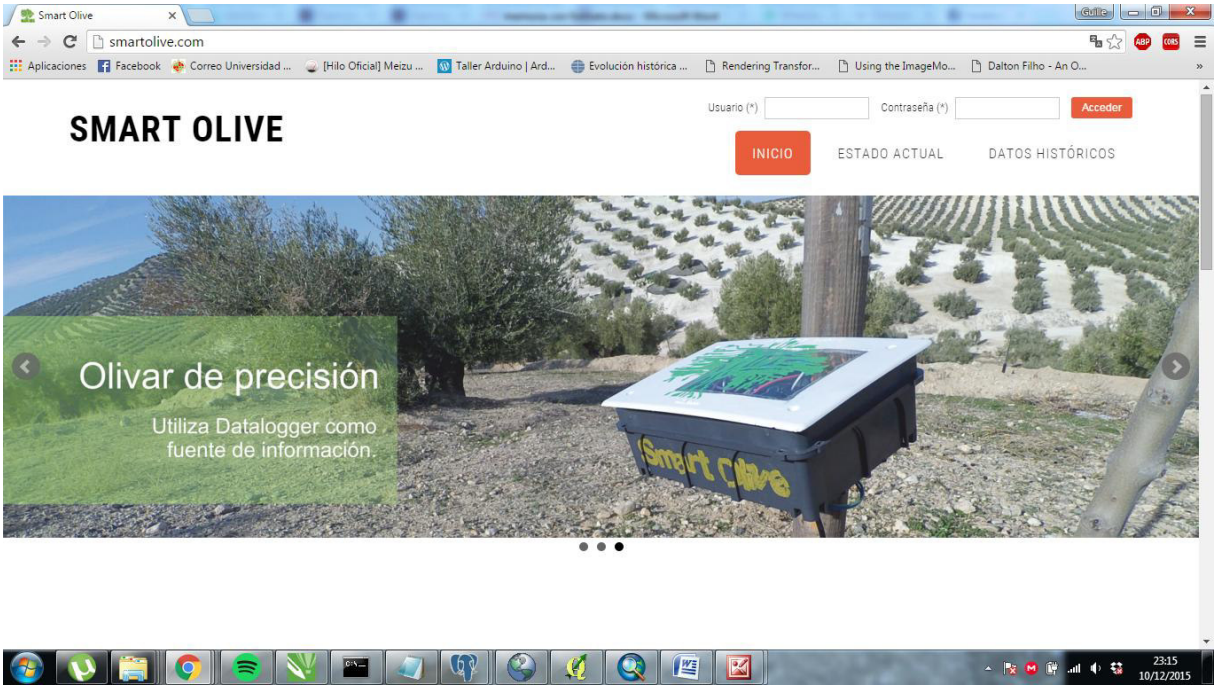

Ilustración 43 smartolive.com Pantalla de Inicio

Si accedemos a la opción "Estado actual", como se muestra en *la Ilustración 47*, se visualizará un mapa en el que poder activar y desactivar capas de interés. Además si se hace "Click" en la capa "Datos", se muestran los últimos datos capturados por los diferentes módulos.

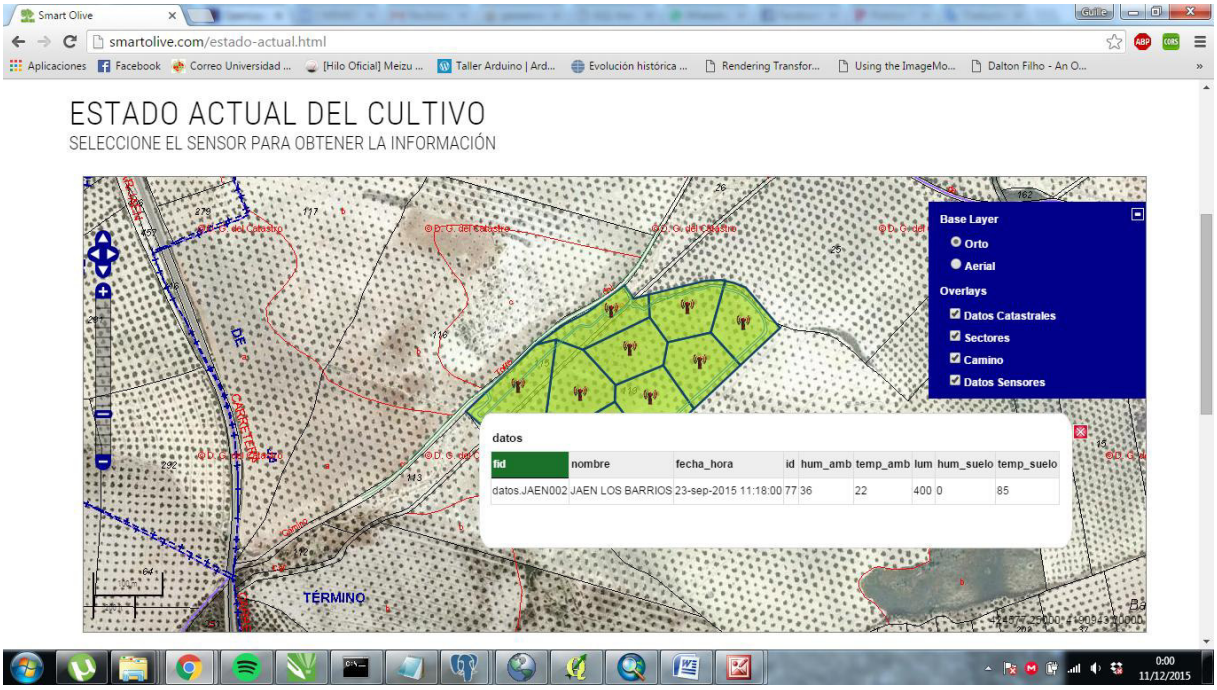

Ilustración 44 Smartolive.com Estado Actual

La información se muestra en un popup que emerge de manera interactiva al seleccionar cada uno de los diferentes sensores de la zona de estudio, mostrando tanto el Código del sensor como los diferentes parámetros capturados y el instante en el que se realizó la toma de información.

5.5. Diagrama de eventos del sistema.

Para una visión más clara de las relaciones entre las diferentes partes del sistema, se presenta la *Ilustración 45*, donde se muestran las interconexiones con un diagrama de eventos.

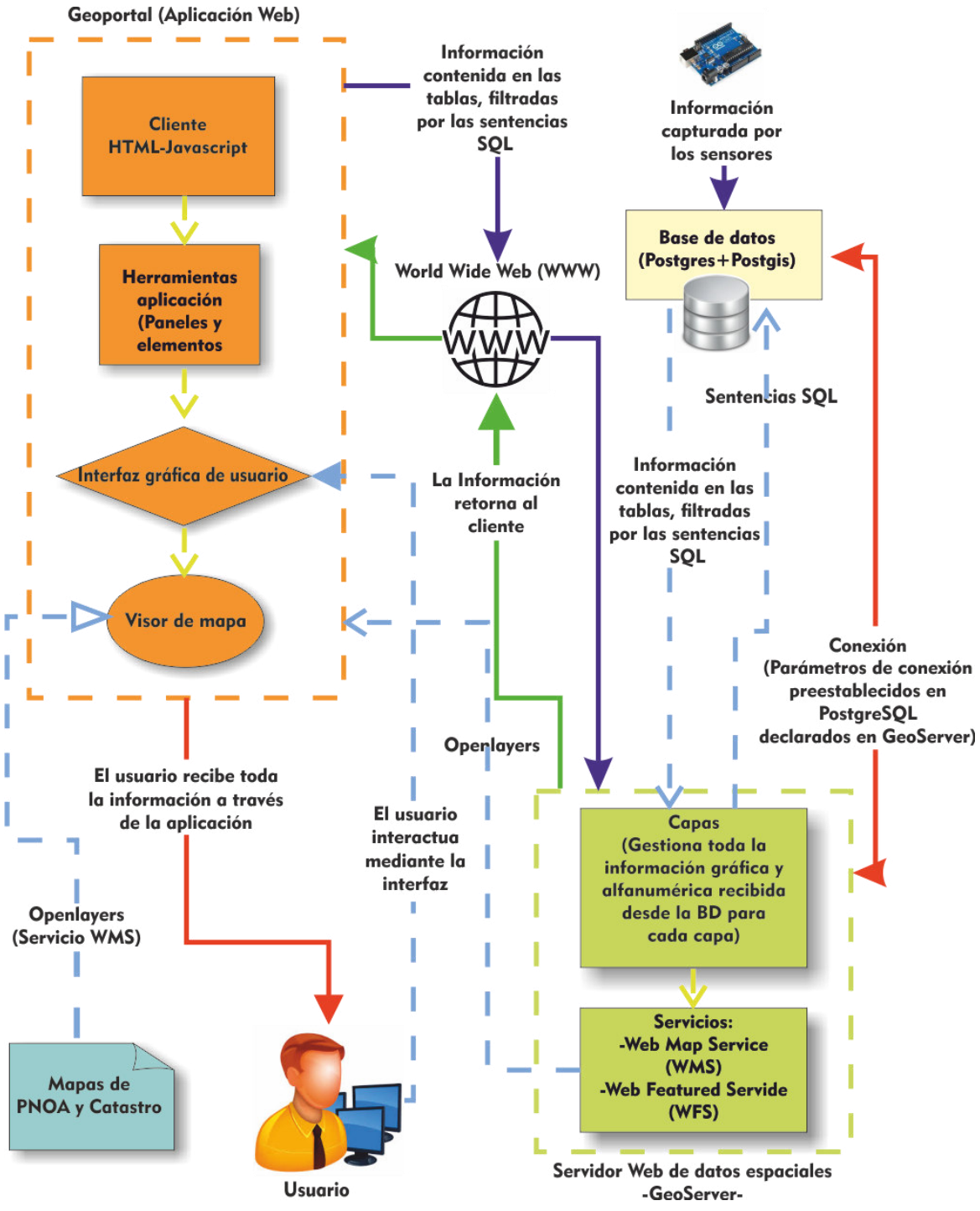

Ilustración 45 Diagrama de Eventos.

CAPÍTULO 6: Conclusiones y Líneas Futuras

6.1. Conclusiones.

El prototipo de hardware ha permitido la captura de las condiciones ambientales del área de estudio, y por consiguiente se han podido llevar a cabo los objetivos planteados al inicio del proyecto. Se han cubierto todas las áreas de desarrollo propuestas en el trabajo, tanto de software como de hardware, y se ha conseguido el objetivo principal del proyecto que era el diseño de un sistema para la captura y análisis de información relevante en el olivar a través de una interfaz gráfica.

La programación del microprocesador Arduino, basada en programación orientada a objetos, facilita la incorporación de nuevos periféricos a los ya existentes. Del mismo modo, la herramienta se plantea como una plataforma en la que poder desactivar los sensores que no sean necesarios en cada nodo de la red. La falta de un *shiedls* de comunicación remota entre la placa y la base de datos, provoca que sea necesario el almacenamiento de los datos en la propia placa del sensor y su posterior volcado manual al servidor.

Las tareas que lleva a cabo la aplicación web se han desarrollado empleando funciones y clases, lo que supone una aplicación fácilmente escalable.

El diseño de la base de datos Postgres y el uso de identificadores específicos para cada sensor permite una fácil escalabilidad del sistema. Además a través del posicionamiento mediante Postgis se puede realizar una conexión directa con el servidor de mapas.

El Cliente Ligero de Mapas permite la interoperabilidad de recursos provenientes de fuentes diferentes, tanto de información propia, como puede ser los datos de los sensores o los Shapefiles de *Geoserver,* a recursos de otros servidores como la Ortofoto del PNOA o la capa WMS del Catastro. Y todo esto puede ser visualizado de manera continua y dinámica en una Aplicación Web.

Como conclusión final, se puede afirmar que el sistema diseñado en este proyecto es un primer paso para el desarrollo de una red de Datalogger que registre la información ambiental del cultivo. Sirve como un punto de partida para nuevas líneas de investigación en la agricultura de precisión.

6.2. Líneas futuras.

El presente proyecto es un punto de partida en la agricultura de precisión en el olivar, un sistema de captura y análisis de la información que pueda llegar a ser un sistema real. El sistema ha sido creado y diseñado para ser escalado. Para que un mismo usuario pueda gestionar de manera online sus diferentes fincas del mismo modo que una cooperativa agraria o una asociación de regantes puedan dar un servicio adicional a sus socios.

Es por ello que se presentan unas posibles mejoras futuras a implementar:

- Integración de un sistema telemático en el hardware para la transmisión de la información con el servidor. Utilizando tecnologías ya existentes como GPRS, bluetooth, Zigbee… O una asociación de estas técnicas.
- Inserción de sensores industriales específicos que midan tanto parámetros fisiológicos de la planta, como el estrés hídrico o la falta de nutrientes.
- Integración de una fuente de alimentación sostenible y duradera en el hardware, como podría ser el uso de pequeñas placas solares en los sensores, y una gestión del estado de las baterías desde la web.
- Desarrollo de un protocolo para móviles que sirva como alternativa a la aplicación web.
- Conectividad del servidor local con otros servidores públicos externos para un mayor análisis de datos, como podrían ser servidores de datos atmosféricos estatales (AEMET).

El sistema desarrollado ha sido planteado para el análisis del olivar, aunque puede ser fácilmente exportado a otros cultivos leñosos y que están actualmente en auge en la zona, como el Pistacho o el Almendro.

ANEXOS

1. Código de Arduino.

```
#include <Time.h>
#include <SD.h>
#include <SPI.h>
#include "DHT.h"
#include <OneWire.h>
#include <DallasTemperature.h>

#define DHTPIN 9     // Pin del Arduino al cual está conectado el pin 2 del sensor
#define DHTTYPE DHT11   // DHT11 sensor humedad

#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

DeviceAddress outsideThermometer = { 0x28, 0xFF, 0x5E, 0x23, 0x00, 0x15, 0x02, 0x28 };

DHT dht(DHTPIN, DHTTYPE);
int id= 0;
const long A = 1000;     //Resistencia en oscuridad KO
const int B = 15;        //Resistencia a la luz (10 Lux) KO
const int LDRPin = A1;   //Pin del LDR
const int fc28 = A0;      //Pin del sensor Fc-28
int CS= 10;
int led= 8;
int V;
int ilum;
int hum_suelo;


File Archivo;

void setup(){

 //Se esablece comunicación con el monitor serial para la comprobación de la
 //carga de datos.
  Serial.begin(9600);

  pinMode(led,OUTPUT); // Led
  pinMode(fc28,INPUT); // Sensor FC28

  setTime(10,00,00,16,11,2015); //Fecha y hora a la que se inicia la lectura
   Serial.print("Fecha Inicializada: ");

      Serial.print(day(now()));
      Serial.print(+ "/") ;
      Serial.print(month(now()));
      Serial.print(+ "/") ;
      Serial.print(year(now()));
      Serial.print( " ") ;
```

```
      Serial.print(hour(now()));
      Serial.print(+ ":") ;
      Serial.print(minute(now()));
      Serial.print(":") ;
      Serial.println(second(now()));


  pinMode(CS, OUTPUT); //Se establece como salida el pin correspondiente a SS.

  Serial.println("Comenzando la comunicacion con la tarjeta SD"); //Se muestra por pantalla que se
va a iniciar la comunicación con la SD

  //Se muestra por el monitor si la comunicación se ha establecido correctamente
  //o ha habido algún tipo de error.
  if (!SD.begin(CS)){

    Serial.println("Se ha producido un fallo al iniciar la comunicacion");
    return;
  }
  Serial.println("Se ha iniciado la comunicación correctamente");

   Archivo = SD.open("JAENO01.csv", FILE_WRITE);

   Archivo.println("Placa de Arduino Nº1");
   Archivo.println("Medidas del sensor: DHT11 LDR FC-28 DS18B20");
      Archivo.print("Fecha iniciada: ");
      Archivo.print(day(now()));
      Archivo.print(+ "/") ;
      Archivo.print(month(now()));
      Archivo.print(+ "/") ;
      Archivo.print(year(now()));
      Archivo.print( " ") ;
      Archivo.print(hour(now()));
      Archivo.print(+ ":") ;
      Archivo.print(minute(now()));
      Archivo.print(":") ;
      Archivo.println(second(now()));
    Archivo.println("Cod_Sensor ID Fecha  Hora Humed_Amb(%)  Temp_Amb(ºC) Luminosidad(lux)
Humed_Suelo(%)  Temp_Suelo(ºC)");

  Archivo.close();

  dht.begin(); // se inicia el sensor DHT11
  sensors.begin(); // se inicia el sensor DS18B20
  sensors.setResolution(outsideThermometer, 10); // se establece la resolucion del sensor
}

void Mostrar_Temperatura(DeviceAddress direccion)
{
  float tempC = sensors.getTempC(direccion);
  Serial.print("Temperatura: ");
  Serial.print(tempC);
}

void loop(){

   time_t t = now(); // almacenamos el tiempo en la variable t

  // Obtiene la Humedad
   float h = dht.readHumidity();
```

```
// Obtiene la Temperatura en Celsius
 float temp = dht.readTemperature();
 // Obtener el nivel de luminosidad
 V = analogRead(LDRPin);   // valor medido por el sensor LDR

      //ilum = ((long)(1024-V)*A*10)/((long)B*Rc*V);    //usar si
       ilum = ((long)V*A*10)/((long)B*10*(1024-V));    // trasformamos la medida en lux


// Control de errores, valida que se obtuvieron valores para los datos medidos
 if (isnan(h) || isnan(t)) {
  Serial.println("Falla al leer el sensor DHT!");
  return;
}
sensors.requestTemperatures();
float tempC = sensors.getTempC(outsideThermometer);

// Obtenemos Humedad del suelo
   hum_suelo = analogRead(fc28);


/*              ESCRIBIENDO DATOS EN LA MEMORIA SD DE ARDUINO                    */

//Se abre el documento sobre el que se va a leer y escribir.
Archivo = SD.open("JAEN001.csv", FILE_WRITE);


//Se comprueba que el archivo se ha abierto correctamente y se procede a
//escribir en él.
if (Archivo){

  //Se escribe información en el documento de texto datos.txt.
    Archivo.print("Jaen001");
    Archivo.print(id);
    Archivo.print(",");
    Archivo.print(day(t));
    Archivo.print(+ "/") ;
    Archivo.print(month(t));
    Archivo.print(+ "/") ;
    Archivo.print(year(t));
    Archivo.print( " ") ;
    Archivo.print(hour(t));
    Archivo.print(+ ":") ;
    Archivo.print(minute(t));
    Archivo.print(+ ":") ;
    Archivo.print(second(t));
    Archivo.print(",");
    Archivo.print(h);
    Archivo.print(",");
    Archivo.print(temp);
    Archivo.print(",");
    Archivo.print(ilum);
    Archivo.print(",");
    Archivo.print(hum_suelo);
    Archivo.print(",");
    Archivo.print(tempC);
    Archivo.println();
  id++;  // Se incrementa el identificador

  Archivo.close();          //Se cierra el archivo para almacenar los datos.
```

```
    digitalWrite(led,HIGH);    // Activamos la salida 8
    delay(1000);              // Esperamos
    digitalWrite(led,LOW);    // Apagamos el led-*


    //Se muestra por el monitor que los datos se han almacenado correctamente.
    Serial.println("Todos los datos fueron almacenados");
  }

  //En caso de que haya habido problemas abriendo datos.txt, se muestra por pantalla.
  else{

    Serial.println("El archivo prueba.txt no se abrio correctamente");
  }
  delay(1000);
}
```

2. Sentencias SQL.

- *Importación archico.csv a tabla Postgres.*

```
COPY datos_sensores FROM 'c:/TFM/prueba1.csv' DELIMITERS ',' CSV;
```

- *Seleccionar último valor de cada sensor.*

```
SELECT sensores.cod_sensor, sensores.nombre, sensores.geom, t1.fecha_hora,
datos_sensores.id, datos_sensores.hum_amb, datos_sensores.temp_amb, datos_sensores.lum,
datos_sensores.hum_suelo, datos_sensores.temp_suelo

FROM sensores, datos_sensores,

                (SELECT cod_sensor, MAX(fecha_hora)as fecha_hora FROM datos_sensores
GROUP BY cod_sensor) as t1

WHERE sensores.cod_sensor = t1.cod_sensor AND datos_sensores.fecha_hora = t1.fecha_hora

ORDER BY sensores.cod_sensor
```

- *Seleccionar valor medio de todos los parámetros de todas las capturas entre una fecha inicial y una fecha final.*

```
SELECT sensores.cod_sensor, sensores.geom, prom.fecha_hora, prom.hum_amb,
prom.temp_amb, prom.lum, prom.hum_suelo, prom.temp_suelo

FROM sensores,

                (SELECT cod_sensor, MIN(fecha_hora) as fecha_hora,

                   AVG(hum_amb)AS hum_amb,AVG(temp_amb) as temp_amb, AVG(lum) as lum,
AVG(hum_suelo) as hum_suelo, AVG(temp_suelo) as temp_suelo

                 FROM datos_sensores

                 WHERE datos_sensores.fecha_hora >= '20150923 11:00' AND
datos_sensores.fecha_hora < '20150923 13:00'

                 GROUP BY cod_sensor) as prom

WHERE sensores.cod_sensor = prom.cod_sensor

ORDER BY sensores.cod_sensor
```

3. Estilos SLD.

- *Capa "Camino".*

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<StyledLayerDescriptor xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.1.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.opengis.net/sld
http://schemas.opengis.net/sld/1.1.0/StyledLayerDescriptor.xsd"
xmlns:se="http://www.opengis.net/se">
  <NamedLayer>
    <se:Name>camino</se:Name>
    <UserStyle>
      <se:Name>camino</se:Name>
      <se:FeatureTypeStyle>
        <se:Rule>
          <se:Name>Single symbol</se:Name>
          <se:LineSymbolizer>
            <se:Stroke>
              <se:SvgParameter name="stroke">#4d9e68</se:SvgParameter>
              <se:SvgParameter name="stroke-width">0.46</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
              <se:SvgParameter name="stroke-linecap">square</se:SvgParameter>
            </se:Stroke>
          </se:LineSymbolizer>
        </se:Rule>
      </se:FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

- ***Capa "Contorno".***

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.1.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.opengis.net/sld
http://schemas.opengis.net/sld/1.1.0/StyledLayerDescriptor.xsd"
xmlns:se="http://www.opengis.net/se">
  <NamedLayer>
    <se:Name>LINDE</se:Name>
    <UserStyle>
      <se:Name>CONTORNO</se:Name>
      <se:FeatureTypeStyle>
        <se:Rule>
          <se:Name>Single symbol</se:Name>
          <se:LineSymbolizer>
            <se:Stroke>
              <se:SvgParameter name="stroke">#0003b8</se:SvgParameter>
              <se:SvgParameter name="stroke-width">1.46</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">bevel</se:SvgParameter>
              <se:SvgParameter name="stroke-linecap">square</se:SvgParameter>
            </se:Stroke>
          </se:LineSymbolizer>
          <se:LineSymbolizer>
            <se:Stroke>
              <se:SvgParameter name="stroke">#c0df16</se:SvgParameter>
              <se:SvgParameter name="stroke-width">0.86</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">round</se:SvgParameter>
              <se:SvgParameter name="stroke-linecap">round</se:SvgParameter>
```

```
            <se:SvgParameter name="stroke-dasharray">4 2</se:SvgParameter>
          </se:Stroke>
        </se:LineSymbolizer>
      </se:Rule>
    </se:FeatureTypeStyle>
  </UserStyle>
 </NamedLayer>
</StyledLayerDescriptor>
```

- *Capa "Sectores".*

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.1.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.opengis.net/sld
http://schemas.opengis.net/sld/1.1.0/StyledLayerDescriptor.xsd"
xmlns:se="http://www.opengis.net/se">
  <NamedLayer>
    <se:Name>poligonos</se:Name>
    <UserStyle>
      <se:Name>poligonos</se:Name>
      <se:FeatureTypeStyle>
        <se:Rule>
          <se:Name>Single symbol</se:Name>
          <se:PolygonSymbolizer>
            <se:Fill>
              <se:SvgParameter name="fill">#a5dd21</se:SvgParameter>
              <se:SvgParameter name="fill-opacity">0.67</se:SvgParameter>
            </se:Fill>
            <se:Stroke>
              <se:SvgParameter name="stroke">#124569</se:SvgParameter>
              <se:SvgParameter name="stroke-width">0.86</se:SvgParameter>
              <se:SvgParameter name="stroke-linejoin">round</se:SvgParameter>
            </se:Stroke>
          </se:PolygonSymbolizer>
        </se:Rule>
      </se:FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

- *Capa "Sensores".*

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.1.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.opengis.net/sld
http://schemas.opengis.net/sld/1.1.0/StyledLayerDescriptor.xsd"
xmlns:se="http://www.opengis.net/se">
  <NamedLayer>
    <se:Name>sensores</se:Name>
    <UserStyle>
      <se:Name>sensores</se:Name>
      <se:FeatureTypeStyle>
        <se:Rule>
          <se:Name>Single symbol</se:Name>
          <se:PointSymbolizer>
```

```
            <se:Graphic>
              <se:ExternalGraphic>
                <se:OnlineResource xlink:type="simple" xlink:href="images/sensor1.svg"/>
                <se:Format>image/svg+xml</se:Format>
              </se:ExternalGraphic>
              <se:Size>9.2</se:Size>
              <se:Displacement>
                <se:DisplacementX>0</se:DisplacementX>
                <se:DisplacementY>-2.8</se:DisplacementY>
              </se:Displacement>
            </se:Graphic>
          </se:PointSymbolizer>
        </se:Rule>
      </se:FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

4. Código HTML.

- *Página de Inicio*

```
<!DOCTYPE HTML>
<html>
   <head>
          <title>Smart Olive</title>
          <meta http-equiv="content-type" content="text/html; charset=utf-8" />
          <meta name="description" content="" />
          <meta name="keywords" content="" />
          <link href='http://fonts.googleapis.com/css?family=Roboto+Condensed:700italic,400,300,700'
rel='stylesheet' type='text/css'>
          <link rel="stylesheet" href="bxslider/jquery.bxslider.css" />
          <!--[if lte IE 8]><script src="js/html5shiv.js"></script><![endif]-->
          <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
          <link rel="icon" type="image/x-icon" href="images/favicon.ico" />
          <script src="js/skel.min.js"></script>
          <script src="js/skel-panels.min.js"></script>
          <script src="js/init.js"></script>
           <script type="text/javascript" src="bxslider/jquery.bxslider.min.js"></script>
          <noscript>
                 <link rel="stylesheet" href="css/skel-noscript.css" />
                 <link rel="stylesheet" href="css/style.css" />
                 <link rel="stylesheet" href="css/style-desktop.css" />

          </noscript>
          <!--[if lte IE 8]><link rel="stylesheet" href="css/ie/v8.css" /><![endif]-->
          <!--[if lte IE 9]><link rel="stylesheet" href="css/ie/v9.css" /><![endif]-->
   </head>
   <body class="homepage">

   <!-- Header -->
          <div id="header">
                 <div class="container">

                        <!-- Logo -->
                               <div id="logo">
                                      <h1><a href="#">Smart olive</a></h1>
                               </div>

                        <!-- Login Form -->

                               <div class="login_form">
                                      <form action="#" method="post">
                                             <div class="field">
                                                    <label for="username">Usuario (*)</label>
                                                    <input type="text" id="username" name="username"
title="Usuario" required>
                                             </div>
                                             <div class="field">
                                                    <label for="password">Contraseña (*)</label>
                                                    <input type="password" id="password" name="password"
title="Contraseña" required>
                                             </div>
                                             <button id="submit" type="submit">Acceder</submit>
                                      </form>
```

```
                        </div>

                    <!-- Nav -->

                        <nav id="nav">
                            <ul>
                                <li class="active"><a href="index.html">Inicio</a></li>
                                <li><a href="estado-actual.html">Estado actual</a></li>
                                <li><a href="right-sidebar.html">Datos históricos</a></li>

                            </ul>
                        </nav>

                </div>
            </div>
    <!-- Header -->

    <!-- Banner -->
            <div class="main_slider_container">
                <ul id="main_slider">
                    <li>
                        <img title="Title" alt="Alternativo" src="images/olivar1.jpg"
                    </li>
                    <li>
                        <img title="Title" alt="Alternativo" src="images/olivar2.jpg"
                    </li>
                    <li>
                        <img title="Title" alt="Alternativo" src="images/olivar3.jpg"
                    </li>
                </ul>
            </div>
    <!-- /Banner -->

    <!-- Main -->
            <div id="page">
<!-- Main  -->
                <center>
                <iframe width="800" height="600" src="https://www.youtube.com/embed/PtR_zqVPv8Q"
frameborder="0" allowfullscreen></iframe>
                </center>
            </div>
    <!-- /Main -->


    <!-- Footer -->
            <div id="footer">
                <div class="container">
                    <div class="row">
                        <div class="3u">
                            <section>
                            <h2> </h2>
                                <img href="#" src="images/logo.jpg" alt="Smart Olive Logo"
style="width:250px;">
                            </section>
                        </div>
                        <div class="3u">
                            <section>
                                <h2>Sobre Smart Olive</h2>
                                <ul class="default">
                                    <li>
```

```
                                                        <p><a href="#">Smart Olive nace como una
herramienta para el agricultor con la que poder gestionar de forma precisa sus cultivos.</a></p>
                                                </li>
                                        </ul>
                                        <h2>Contacto</h2>
                                        <ul class="default">
                                                <li>
                                                        <p><a href="#">Telf: 62290619.</a></p>
                                                        <p><a href="#">Email:
ggp00005@red.ujaen.es.</a></p>
                                                </li>


                                        </ul>
                                </section>
                        </div>
                        <div class="3u">
                                <section>
                                        <h2>Sensores y Micro-controladores</h2>
                                        <ul class="style5">
                                                <li><a href="#"><img src="images/sensor1.jpg"
alt=""></a></li>
                                                <li><a href="#"><img src="images/sensor2.jpg"
alt=""></a></li>
                                                <li><a href="#"><img src="images/sensor3.jpg"
alt=""></a></li>
                                                <li><a href="#"><img src="images/sensor4.jpg"
alt=""></a></li>
                                                <li><a href="#"><img src="images/sensor5.jpg"
alt=""></a></li>
                                                <li><a href="#"><img src="images/sensor6.jpg"
alt=""></a></li>
                                        </ul>
                                        <a href="#" class="button">More Collections</a>
                                </section>
                        </div>
                        <div class="3u">
                                <section>
                                        <h2>Enlaces de interés</h2>
                                        <p><strong><a href="https://www.arduino.cc/">Arduino</a>
</strong></p>
                                        <p><strong><a
href="http://www.postgresql.org/">PostgreSQL</a> </strong></p>
                                        <p><strong><a href="http://www.postgis.us/">Postgis</a>
</strong></p>
                                        <p><strong><a
href="http://geoserver.org/">Geoserver</a></strong></p>
                                        <p><strong><a href="http://openlayers.org/">OpenLayers</a>
</strong></p>
                                </section>
                        </div>
                </div>
        </div>
    </div>
</div>
    <!-- /Footer -->

    <!-- Copyright -->
        <div id="copyright" class="container">
```

```
                Design: <a href="http://templated.co">TEMPLATED</a> Images: <a
href="http://unsplash.com">Unsplash</a> (<a href="http://unsplash.com/cc0">CC0</a>)
          </div>

          <script>
                $('#main_slider').bxSlider({
                      mode: 'fade',
                      auto: true
                })
          </script>

   </body>
</html>
```

- ***Pagina de Estado actual.***

```
<!DOCTYPE HTML>
<!--
   Ex Machina by TEMPLATED
    templated.co @templatedco
    Released for free under the Creative Commons Attribution 3.0 license (templated.co/license)
-->
<html>
   <head>
          <title>Smart Olive</title>
          <meta http-equiv="content-type" content="text/html; charset=utf-8" />
          <meta name="description" content="" />
          <meta name="keywords" content="" />
          <link href='http://fonts.googleapis.com/css?family=Roboto+Condensed:700italic,400,300,700'
rel='stylesheet' type='text/css'>
          <!--[if lte IE 8]><script src="js/html5shiv.js"></script><![endif]-->
          <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
          <link rel="icon" type="image/x-icon" href="images/favicon.ico" />
          <script src="js/skel.min.js"></script>
          <script src="js/skel-panels.min.js"></script>
          <script src="js/init.js"></script>

          <script src="http://www.openlayers.org/api/OpenLayers.js"></script>
          <script type="text/javascript" src="libs/proj4js/lib/proj4js-combined.js"></script>
       <script type="text/javascript" src="libs/proj4js/lib/deprecated.js"></script>

          <noscript>
                <link rel="stylesheet" href="css/skel-noscript.css" />
                <link rel="stylesheet" href="css/style.css" />
                <link rel="stylesheet" href="css/style-desktop.css" />

          </noscript>
          <!--[if lte IE 8]><link rel="stylesheet" href="css/ie/v8.css" /><![endif]-->
          <!--[if lte IE 9]><link rel="stylesheet" href="css/ie/v9.css" /><![endif]-->
          <style type="text/css">
       #zonademapa {
           width: 100%; height: 515px;
           border: solid 1px #808080;
                margin-left:15px;
                margin-top:15px;
                margin-bottom:15px;
       } <!--tipo de marco-->
```

```
</style>
<script type="text/javascript">

// sets the HTML provided into the nodelist element
  function setHTML(response){
      document.getElementById('info').innerHTML = response.responseText;
  };
        window.onload= function() {  //esto no se para que es
        var bounds = new OpenLayers.Bounds
        (
               401231, 4180973,
               448231, 4200973
        );
        <!-- borde pner el de geoserver -->

        //view: new OpenLayers.View({
        //     projection: projection,
        //     center: OpenLayers.proj.transform([424933.92, 4190275.09], 'EPSG:4326',
'EPSG:25830'),
        //     extent: extent,
        //     zoom: 12
        //  })

        var options = {
               controls: [],
               maxExtent: bounds,
               maxResolution: 20, <!-- modificar resolucion-->
               projection: "EPSG:25830", <!-- modificar mi proyeccion-->
               units: 'm'
               //center: [424731, 4190973] no me deja poner el centro
        };

        map = new OpenLayers.Map("divMapa", options);      <!-- Se crea el Mapa de OpenLayers-->

        var orto = new OpenLayers.Layer.WMS                              <!-- Se añaden las capas
-->
        (
               "Orto", "http://www.idee.es/wms/PNOA/PNOA?",    <!--Esta es la orto que sirve de
capa base -->
               {layers:'pnoa'},{isBaseLayer: true}
        );


        <!-- no se porque en la capa de catastro me sale una marca de D.G del Catastro-->

        var catastro = new OpenLayers.Layer.WMS(
               "Datos Catastrales",
               "http://ovc.catastro.meh.es/Cartografia/WMS/ServidorWMS.aspx?",
               {layers: 'Catastro',
               transparent: "true",
               format: "image/png"
               },
               {isBaseLayer: false}
        );

        // no sé como cambiar la proyección de la capa bing
        var aerial = new OpenLayers.Layer.Bing({
```

```
            name: "Aerial",
            key: "AiDgMj5C_ATjR981eUlYHgaUgWc7AKHQGQP2h1EVfHIlvY8-g5hkQICqjtSdcC-t",
            type: "Aerial"
        });

              var sectores = new OpenLayers.Layer.WMS(
                     "Sectores",
                     "http://localhost:8080/geoserver/Proyecto/wms",
                     {layers: 'sectores',
                     transparent: "true",
                     format: "image/png"
                     },
                     {isBaseLayer: false}
              );

                     var camino = new OpenLayers.Layer.WMS(
                     "Camino",
                     "http://localhost:8080/geoserver/Proyecto/wms",
                     {layers: 'camino',
                     transparent: "true",
                     format: "image/png"
                     },
                     {isBaseLayer: false}
              );


              var datos = new OpenLayers.Layer.WMS(
                     "Datos Sensores",
                     "http://localhost:8080/geoserver/Proyecto/wms",
                     {layers: 'datos',
                     transparent: "true",
                     format: "image/png"
                     },
                     {isBaseLayer: false}
              );



              map.addLayers([orto,catastro,aerial,sectores,camino,datos]); <!-- Pongo los nombres de
todas las capas var-->
              map.zoomToMaxExtent();

              <!--Para añadir los controles-->
              map.addControl(new OpenLayers.Control.PanZoomBar({
             position: new OpenLayers.Pixel(2, 30)
        }));
              map.addControl(new OpenLayers.Control.MousePosition());
              map.addControl(new OpenLayers.Control.ScaleLine());
              map.addControl(new OpenLayers.Control.LayerSwitcher());
              map.addControl(new OpenLayers.Control.KeyboardDefaults());
              map.addControl(new OpenLayers.Control.Navigation());

              //Proj4js.defs["EPSG:25830"] = "+proj=utm +zone=30 +ellps=GRS80 +units=m +no_defs";
              //Proj4js.defs["EPSG:3857"]= "+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0
+x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null +no_defs";

              //var src = new OpenLayers.Projection('EPSG:25830');
              //var dst = new OpenLayers.Projection('EPSG:2857');

              //Proj4js.transform(src, dst, aerial);
```

```
                //aerial.projection=dst;
;
                info = new OpenLayers.Control.WMSGetFeatureInfo({
                        url: 'http://localhost:8080/geoserver/Proyecto/wms',
                        title: 'Identify features by clicking',
                        layers: [datos],
                        queryVisible: true,
                        eventListeners: {
                                getfeatureinfo: function(event) {
                                        map.addPopup(new OpenLayers.Popup.FramedCloud(
                                                "chicken",
                                                map.getLonLatFromPixel(event.xy),
                                                new OpenLayers.Size(200,200),
                                                event.text,
                                                null,
                                                true
                                        ));
                                        document.getElementById("info").innerHTML=event.text;
                                }
                        }
                });
                map.addControl(info);
                info.activate();
                }



                </script>
   </head>
   <body class="no-sidebar">

   <!-- Header -->
          <div id="header">
                <div class="container">

                        <!-- Logo -->
                                <div id="logo">
                                        <h1><a href="#">Smart Olive</a></h1>
                                </div>
                           <div class="logo">
                                        <a href="/index.php" title="Inicio"><span></span></a>
                                </div>

                                <!-- Login Form -->

                                <div class="login_form">
                                        <form action="#" method="post">
                                                <div class="field">
                                                        <label for="username">Usuario (*)</label>
                                                        <input type="text" id="username" name="username"
title="Usuario" required>
                                                </div>
                                                <div class="field">
                                                        <label for="password">Contraseña (*)</label>
                                                        <input type="password" id="password" name="password"
title="Contraseña" required>
                                                </div>
                                                <button id="submit" type="submit">Acceder</submit>
                                        </form>
```

```
                                </div>

                        <!-- Nav -->
                                <nav id="nav">
                                        <ul>
                                                <li><a href="index.html">Inicio</a></li>
                                                <li class="active"><a href="estado-actual.html">Estado
actual</a></li>
                                                <li><a href="right-sidebar.html">Datos Historicos</a></li>

                                        </ul>
                                </nav>

                        </div>
                </div>
        <!-- Header -->

        <!-- Banner -->
                <div id="banner">
                        <div class="container">
                        </div>
                </div>
        <!-- /Banner -->

        <!-- Main -->
                <div id="page">

                        <!-- Main -->
                        <div id="main" class="container">
                                <div class="row">
                                        <div class="12u">
                                                <section>
                                                        <header>
                                                                <h2>Estado actual del cultivo</h2>
                                                                <span class="byline">Seleccione el sensor para obtener
la información</span>
                                                        </header>
                                                </section>
                                        </div>
                                </div>
                                <div id="zonademapa">
                                <div id="divMapa" style="width: 100%; height: 100%">
                                </div>
                        </div>
                        </div>
                        <!-- Main -->



                </div>
        <!-- /Main -->

        <!-- Featured -->

        <!-- /Featured -->

        <!-- Footer -->
                <div id="footer">
                        <div class="container">
                                <div class="row">
```

```
                        <div class="3u">
                                <section>
                                <h2> </h2>
                                        <img href="#" src="images/logo.jpg" alt="Smart Olive Logo"
style="width:250px;">
                                </section>
                        </div>
                        <div class="3u">
                                <section>
                                        <h2>Sobre Smart Olive</h2>
                                        <ul class="default">
                                                <li>

                                                        <p><a href="#">Smart Olive nace como una
herramienta para el agricultor con la que poder gestionar de forma precisa sus cultivos.</a></p>
                                                </li>
                                        </ul>
                                        <h2>Contacto</h2>
                                        <ul class="default">
                                                <li>
                                                        <p><a href="#">Telf: 62290619.</a></p>
                                                        <p><a href="#">Email:
ggp00005@red.ujaen.es.</a></p>
                                                </li>


                                        </ul>
                                </section>
                        </div>
                        <div class="3u">
                                <section>
                                        <h2>Sensores y Micro-controladores</h2>
                                        <ul class="style5">
                                                <li><a href="#"><img src="images/sensor1.jpg"
alt=""></a></li>
                                                <li><a href="#"><img src="images/sensor2.jpg"
alt=""></a></li>
                                                <li><a href="#"><img src="images/sensor3.jpg"
alt=""></a></li>
                                                <li><a href="#"><img src="images/sensor4.jpg"
alt=""></a></li>
                                                <li><a href="#"><img src="images/sensor5.jpg"
alt=""></a></li>
                                                <li><a href="#"><img src="images/sensor6.jpg"
alt=""></a></li>
                                        </ul>
                                        <a href="#" class="button">More Collections</a>
                                </section>
                        </div>
                        <div class="3u">
                                <section>
                                        <h2>Enlaces de interés</h2>
                                        <p><strong><a href="https://www.arduino.cc/">Arduino</a>
</strong></p>
                                        <p><strong><a
href="http://www.postgresql.org/">PostgreSQL</a> </strong></p>
                                        <p><strong><a href="http://www.postgis.us/">Postgis</a>
</strong></p>
                                        <p><strong><a
href="http://geoserver.org/">Geoserver</a></strong></p>
```

```
                                        <p><strong><a href="http://openlayers.org/">OpenLayers</a>
</strong></p>
                                    </section>
                                </div>
                            </div>
                        </div>
                </div>
   <!-- /Footer -->

   <!-- Copyright -->
          <div id="copyright" class="container">
                 Design: <a href="http://www.smartolive.com">Smart Olive</a> Images: <a
href="http://unsplash.com">Unsplash</a> (<a href="http://www.smart-olive.com">CCO</a>)
          </div>


   </body>
</html>
```

BIBLIOGRAFÍA

Bibliografía

Arduino. 2015. Arduino. [En línea] 2015. http://www.arduino.cc.

Dallas Semiconductor. *Programable Resolution DS18B20.*

Enciclopedia CCM. 2015. Introducción-Bases de datos. [En línea] 2015. http://www.ccm.net.

Escofet, Carme Martín. 2014. *El lenguaje SQL.* 2014.

Fernandez, J.E, Díaz-Espejo, A y Cuevas, M.V. 2012. *¿Podemo regar mejor el olivar?* 2012.

Girod, Anton. 2013. *Detección de luz con sensor LDR.* 2013.

Juan Agüera Vega, Francisco Márquez García. 2012. *Técnicas agrarias sostenibles mitigadoras del cambio climático, Agricultura de Precisión.* 2012.

Leaflet Open-source. [En línea] http://leafletjs.com/.

Luis Gurovich. 2015. El gran desarrollo de los sensores de plantas. [En línea] 2015. http://www.redagricola.com.

Map Server. map server. [En línea] http://mapserver.org/.

Moisture Sensor. [En línea] http://www.dfrobot.com/wiki/index.php?title=Moisture_Sensor_(SKU:SEN0114).

Olivicultura de precisión. [En línea] http://oliviculturadeprecision.com.

OSGeo. 2015. osgeo.org. [En línea] 2015. http://wiki.osgeo.org.

Ovalles, Francisto A. 2006. *Introducción a la Agricultura de Precisión.* 2006.

Pro Developer, Integracion de tecnologías. Clientes ligeros de mapas. [En línea] http://www.prodevelop.es/es/tecs/geo/clienteligero.

Rios, Joaquim Bellvert. 2014. *El uso de la teledetección de alta resolución como herramienta para realizar un manejo eficiente del riego en viñedos.* 2014.

Suárez, Jose Manuel Sánchez. 2015. *Introducción a OpenLayers.* 2015.

Temperature and Humidity Sensor. [En línea]
http://www.dfrobot.com/wiki/index.php?title=DHT11_Temperature_and_Humidity_Sensor_(SKU:_DFR0067).