



**Universidad de Pinar del Río**

**Facultad de Ciencias Técnicas**

**Dpto. de Telecomunicaciones y Electrónica**

## **Trabajo de diploma.**

**Título: Sistema de Adquisición de datos espaciales con terminal Android y volcado en nube simulada para la Agricultura de Precisión**

(Tesis en opción al título de Ingeniero en Telecomunicaciones)

**Autor: Fco. Javier Guembe Lara**

**Pinar del Río. 2018**



**Universidad de Pinar del Río**

**Facultad de Ciencias Técnicas**

**Dpto. de Telecomunicaciones y Electrónica**

## **Trabajo de diploma.**

**Título: Sistema de Adquisición de datos espaciales con terminal  
Android y volcado en nube simulada para la Agricultura de  
Precisión**

(Tesis en opción al título de Ingeniero en Telecomunicaciones)

**Autor: Fco. Javier Guembe Lara**

**Tutores: DrC. José Raúl Vento Álvarez**

**Ing. Marcos Lázaro Álvarez Arteaga**

**Pinar del Río, 2018**

## PENSAMIENTO

*“Los filósofos se han limitado a interpretar el mundo de distintos modos; de lo que se trata es de transformarlo.”*

**Karl Marx**

*“Las obras de conocimiento deben ser libres, no hay excusas para que no sea así”*

**Richard Stallman**

## PÁGINA DE ACEPTACIÓN

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

Presidente del Tribunal

Secretario

Vocal

**Ciudad y fecha:**

## DECLARACIÓN DE AUTORIDAD

Declaro que soy autor(a) de este Trabajo de Diploma y que autorizo a la Universidad de Pinar del Río, a hacer uso del mismo, con la finalidad que estime conveniente.

Firma: \_\_\_\_\_



Fco. Javier Guembe Lara  
guembe.100649@e.unavarra.es

Fco. Javier Guembe Lara autoriza la divulgación del presente trabajo de diploma bajo licencia Creative Commons de tipo **Reconocimiento No Comercial Sin Obra Derivada**, se permite su copia y distribución por cualquier medio siempre que mantenga el reconocimiento de sus autores, no haga uso comercial de las obras y no realice ninguna modificación de ellas. La licencia completa puede consultarse en: <http://creativecommons.org/licenses/by-nc-nd/2.5/ar/legalcode>

Fco. Javier Guembe Lara autoriza al Dpto. de Telecomunicaciones adscrito a la Universidad de Pinar del Río a distribuir el presente trabajo de diploma en formato digital bajo la licencia Creative Commons descrita anteriormente y a conservarlo por tiempo indefinido, según los requerimientos de la institución, en el repositorio de materiales didácticos disponible en: <https://github.com/jguembe/sistema-adquisicion-datos-espaciales-android-y-nube-agricultura-precision/>

Fco. Javier Guembe Lara autoriza al Dpto. de Telecomunicaciones adscrito a la Universidad de pinar del Río a distribuir el presente trabajo de diploma en formato digital bajo la licencia Creative Commons descrita anteriormente y a conservarlo por tiempo indefinido, según los requerimientos de la institución, en el repositorio de tesinas disponible en: <http://revistas.mes.edu.cu>

## AGRADECIMIENTOS

En primer lugar quiero agradecer a mis tutores Vento y Marcos. A Vento por aparecer un día en la Universidad Pública de Navarra y proponerme un proyecto tan bonito y a la vez reto para mí, además de ayudarme en todo cuanto necesité en esta tierra, antes extraña para mí, llamada Pinar del Rio. A Marcos por hacer todo lo posible por ayudarme.

A toda la ciudad de Pinar del Rio, por hacer de un sitio extraño donde aterricé en un hogar que nunca olvidaré. A mi compañero de cuarto Juan Zegarra, por hacer de mi estancia una aventura continua. A todos los/las compañeros/as de geología, con los/las que reí y disfrute. A todos/as las trabajadores/as de la UPR, en concreto a los y las de la residencia. En especia a Fermin, Jon y a Sucel, por darme de comer.

No puedo dejar atrás todo lo que despedí por seis meses en Iruña, y por los que gracias a todos/as ellos/as he podido estar aquí. A todas/os mis compañeras/os de la NUP, que hicieron de cuatro años de estudio una montaña rusa de emociones y lucha. A todos/as mis amigos/as de Berriozar en especial a mi “koadrilla”. A mis compañeras/os de militancia, por serlo todo para mí.

Y por último y por ello más importante, a toda mi familia. En especial a mi madre, mi padre y mi hermana. Por dármelo todo, por permitirme estar aquí. A todos/as los/las demás que me vieron empezar la carrera, pero no podrán verla terminar, aunque estaban seguras de que lo lograría. A mi familia de Murcia, Archena, que a pesar de estar lejos, siempre los siento cerca.

MILA ESKER GUZTIOI!

## **DEDICATORIA**

Este trabajo se lo dedico a toda mi familia. En especial y con el mayor de los cariños, a mi abuela Isabel, mi abuela Margarita y mi abuelo Fermin, ya fallecidos. Siguiendo de ellas sus hermanas vivas La Josefica y M<sup>a</sup> de Jesus.

## **RESUMEN**

En este trabajo se realiza el diseño e implementación de un sistema de adquisición de datos espaciales y una nube simulada que se encarga de recibir y manipular los datos. Para ello se hace uso de un teléfono móvil inteligente con sistema operativo Android (terminal de adquisición de datos), un sensor externo de temperatura y humedad, y un ordenador portátil (usado para simular la nube). El teléfono se encarga de obtener latitud, longitud y altitud, con el receptor GPS que incorpora. También, adquiere la información de temperatura y humedad del sensor externo mediante Bluetooth. Posteriormente, se encarga de enviar toda la información a la nube simulada. La nube consta de un interfaz REST que se encarga de tramitar las peticiones (como la de recoger dichos datos) y una base de datos donde se almacenan los datos. Además, todos los datos son enviados y recibidos en formato JSON, concretamente bajo el estándar GeoJSON, concebido para información geográfica.

El sistema puede ser utilizado para la Agricultura de Precisión, que tanto se ha estado desarrollando en los últimos años debido a la escasez de recursos naturales y los problemas ambientales. Sin embargo, los sistemas desarrollados son de alta tecnología, alto coste y poco accesibles. Este trabajo demuestra como se puede realizar un sistema con tecnologías accesibles. En el mismo, los datos capturados se han graficado para mostrar su utilidad en la práctica sobre el terreno.

## **PALABRAS CLAVES:**

Android, sensores, GPS

## SUMMARY

This thesis explains the design and implementation of an spatial data acquisition system and a simulated cloud that manages and stores the data. For this, a smartphone with Android Operative System (as data acquisition terminal), one temperature and humidity external sensor and a laptop (used to simulate cloud) its been used. The smartphone provides latitude, longitude and altitude information, thanks to the GPS receiver that's incorporated. In addition, it acquires humidity and temperature information from the external sensor via Bluetooth. Then, it sends all the data to the simulated cloud. This cloud is composed by a REST interface that is responsible for receiving and managing the requests and a database which stores all the data. Furthermore, all the data is transmitted and received in JSON format, specifically under the GeoJSON standard, created for the geographic information.

This system, may be used in Precision Agriculture, which has been developed for many years responding to the natural resources poverty and the environment problems. Although, the actual developed systems have really high technology, but too expensive and with poor accessibility. This paper demonstrates how it's possible to make a system with accessible technology. The captured data has been graphed to validate the usefulness in the practice on the terrain.

## KEY WORDS:

Android, sensors, GPS

## LABURPENA

Gradu amaierako lan honetan, datu espazialak eskuratzen dituen sistema baten diseinua eta implementazioa garatu egin da. Horrekin batera datu horiek jasotzen eta gordetzen dituen hodei simulatua garatu egin da. Sistema hau funtzionarazteko hurrengo aparailuak erabili dira: Android sistema eragilea duen mugikorra (datuak eskuratzen dituen terminala), temperatura eta hezetasun sentsorea eta ordenagailu eramangarri bat (hodei simulatua). Mugikorrik latitudea, longitudea eta altitudea biltzen ditu, GPS hargailuari esker eta temperatura zein hezetasuna lortzen ditu sentsoretik Bluetooth bidez. Ondoren, informazio guztia hodei simulatuari bidaltzeaz arduratzen da. Hodeia bi elementuz osaturik dago, alde batetik REST interfazea, zeinak eskaerak tramitatzen dituen, eta datu basea, non datuak gordetzen diren. Gainera datu guztiak JSON formatuan bidaltzen dira, GeoJSON estandarra erabiliz, estandar hau informazio geografikorako diseinatu baita.

Sistema hau Zehaztasunezko Nekazaritzarako erabili daiteke, baliabide naturalen urritasuna eta ingurumen arazoak bultzatuta azken urteetan hainbeste garatu den sektorea. Hala ere, goi-mailako teknologia erabriz sortu izan diren sistemak garestiak eta eskuratzeko zailak dira. Lan honek sistema bat teknologia eskuragarriekin nola egin daitekeen erakusten du. Sistema honen praktikarako erabilgarritasuna erakusteko jasotako datuak grafikoen bidez adierazi dira.

## HITZ GAKOAK:

Android, sentsoreak, GPS

TABLA DE CONTENIDO	Pág.
INTRODUCCIÓN.....	1
CAPÍTULO I. FUNDAMENTOS TECNOLÓGICOS PARA EL DESARROLLO DEL PROYECTO.....	5
1.1. AP: Agricultura de Precisión.....	5
1.1.1 Tecnologías usadas en AP.....	6
1.1.2. Técnicas AP.....	8
1.1.3. Conclusión.....	11
1.2. IDE : Infraestructuras de Datos Espaciales - Spatial Data Infrastructures (SDI).....	12
1.2.1. IG: Información Geográfica.....	12
1.2.2. SIG: Sistemas de Información Geográfica.....	12
1.2.3. IDE.....	13
1.2.4. Interoperatividad: Normas, estándares y protocolos.....	14
1.2.5. Origen y desarrollo.....	15
1.2.6. Componentes IDE.....	16
1.2.7. Conclusión.....	18
1.3. Sistema de obtención de datos geográficos.....	19
1.3.1. Sistema operativo Android.....	19
1.3.1.1. Características.....	20

1.3.1.2. Arquitectura.....	21
1.3.1.3. Android Studio.....	23
1.3.1.4. Estructura de un proyecto Android.....	23
1.3.1.5. Componentes de una aplicación.....	25
1.3.2. Bluetooth.....	27
1.3.3. GPS: Sistema de posicionamiento GNSS.....	31
1.4. Sistema de transferencia y almacenamiento de datos: Nube.....	34
1.4.1. <a href="#">Transferencia</a> de IG a la nube.....	34
1.4.1.1. REST: REpresentational State Transfer.....	36
1.4.1.2. Formato de datos IG para su trasferencia.....	38
1.4.2. Almacenamiento de IG en la nube.....	42
CAPÍTULO II. DISEÑO E IMPLEMENTACIÓN.....	45
2.1. Arquitectura del proyecto.....	46
2.2. Manejo del terminal de adquisición y envío de datos espaciales (Android).....	47
2.2.1. Manejo de Bluetooth para la adquisición de datos del sensor externo.....	47
2.2.2. Manejo de GPS para la obtención de la georeferencia.....	54
2.2.3. Manejo de WIFI y envío de datos al emulador de Nube de almacenamiento de datos.....	57
2.3. Implementación nube con REST API.....	63

2.3.1. Base de datos.....	63
2.3.2. Servicio Web REST.....	66
CAPÍTULO III. RESULTADOS OBTENIDOS.....	71
3.1. Caso de estudio 1: Delimitación de terreno para sembrado.....	71
3.2. Caso de estudio 2: Superficie con pendiente.....	73
3.3. Caso de estudio 3: Superficie con sol y sombra.....	75
CONCLUSIONES.....	77
RECOMENDACIONES.....	78
ÍNDICE DE FIGURAS.....	82
ÍNDICE DE TABLAS.....	83
REFERENCIAS BIBLIOGRÁFICAS.....	84
BIBLIOGRAFÍA.....	87
ANEXOS.....	92

## INTRODUCCIÓN

Las Naciones Unidas, dentro de los objetivos del milenio plantea la necesidad inmediata de mejorar la producción de alimentos para dar respuesta al crecimiento poblacional del planeta de forma económicamente sustentable y con mecanismos de distribución adecuados. Tal objetivo de trabajo indica la aplicación de los desarrollos científico técnicos a la producción de alimentos en todas las dimensiones del proceso. Se suman también los requerimientos de la seguridad en la sanidad alimentaria, la conservación de los recursos naturales y las leyes de la economía de mercado. En este escenario, las tecnologías de la información y comunicación juegan un papel protagónico, dado el espectacular desarrollo de las mismas en las últimas décadas y su potencialidad de penetrar en prácticamente todos los procesos de producción agrícola e industrial en vías del aumento de la eficiencia de la producción alimentaria.

Los recientes desarrollos en la micro y nano electrónica, las tecnologías de las comunicaciones, la capacidad de sensado de variables físico-químicas, ambientales, de forma remota, la inteligencia computacional, los servicios de almacenamiento de grandes volúmenes de información (*Big Data*) y la minería de datos se presentan como herramientas de apreciable utilidad en todo este escenario.

En la agricultura tradicional se reconoce la diversidad de rendimientos entre las distintas áreas de cultivo, aun siguiendo los mismos procedimientos (laboreo, siembra, abonado, riego, etc.). Esto ha demostrado la necesidad de la adecuación de cada proceso a las condiciones específicas de cada ubicación geográfica en cada momento. Como respuesta ha aparecido en los últimos años la agricultura de precisión (AP), encaminada la aplicación de las acciones precisas en tiempo y espacio para producir las condiciones óptimas de producción agrícola.

En la agricultura de precisión confluyen los sistemas de información de datos espaciales (IDEs), como célula organizativa de los grandes volúmenes de

información georeferenciada que caracterizan el escenario de actividad. Evidentemente, la captación de los datos es uno de los procesos más importantes y en los que actualmente se aprecia una notable actividad de investigación científica y desarrollo de nuevos sistemas.

Un importante tributo a la agricultura de precisión son los novedosos sistemas de geolocalización satelital o denominados GPS (*Global Positioning Systems*), que permiten georreferenciar cada variable a tener en cuenta en la toma de decisiones dentro de los procesos tecnológicos agrícolas.

Los GPSs proporcionan, entre otros datos, las coordenadas latitud, longitud, altitud del punto donde se encuentra su antena), datos que, sumados con temperatura, humedad, composición del aire, etc, permiten las decisiones correctas para los procesos de riego, abonado, etc.

En el mercado han aparecido una apreciable cantidad de sistemas de adquisición de datos espaciales, los que comúnmente son de muy altas prestaciones, pero a un elevado costo, a menudo prohibitivo para productores o gestores de estos procesos.

Se presenta como **situación problemática** la necesidad de un sistema de adquisición de datos espaciales a costo razonable, con tecnologías disponibles por parte de los productores agrícolas y cualquier otro participante en el proceso de producción.

En este punto, los dispositivos móviles como teléfonos y tabletas son protagónicos como terminal personal para los procesos de captura, intercambio y acceso a la información, dada su penetración masiva, costo, potencialidad, movilidad y versatilidad, entre otras ventajas de su uso. Este proceso se desarrolla sobre diversas plataformas de tecnologías de redes que abarcan las propias redes de telefonía celular, las redes inalámbricas en todas sus variantes y las tecnologías de redes personales.

Los teléfonos inteligentes incorporan potencialidades que los ubican como posibles terminales de captura de datos espaciales. Con este punto de partida se ha desarrollado este proyecto de fin de carrera que tuvo como **hipótesis** inicial que: “el

uso del teléfono inteligente, con sus sensores asociados, permitiría el desarrollo de un sistema de adquisición de datos espaciales adecuado a la gran mayoría de necesidades de la agricultura de precisión.

En esta investigación se ha tenido como **objeto de estudio** los Sistemas de Adquisición de Datos Espaciales. Y se definió como el **objetivo general de la investigación**: “desarrollar un sistema de adquisición de datos espaciales, usando un teléfono móvil con sistema operativo Android como terminal de obtención de los datos para su uso en la agricultura de precisión”. Para lograr el cumplimiento de este objetivo general se trazaron los siguientes **objetivos específicos**:

1. Estudiar los sistemas de adquisición de datos espaciales y los IDEs.
2. Estudiar el sistema operativo Android y el receptor GPS incorporado a los teléfonos inteligentes.
3. Estudiar y seleccionar los sensores de temperatura y humedad disponibles para su asociación a los teléfonos inteligentes.
4. Estudiar la programación de aplicaciones en el sistema operativo Android.
5. Desarrollar una aplicación para la captura de la posición dada por el receptor GPS, y los sensores asociados, con capacidad para enviar los datos a la nube para su almacenamiento.
6. Desarrollar una aplicación para la recepción y almacenamiento de los datos en el escenario de nube simulada.
7. Desarrollar una aplicación para el graficado de los datos.

El presente trabajo se estructuró en tres capítulos:

Un primer capítulo con los fundamentos tecnológicos para el desarrollo del proyecto en el que son revisadas las tecnologías de Agricultura de Precisión, IDEs, Sistemas de obtención de datos geográficos, Sistema Operativo Android para teléfonos Inteligentes, Almacenamiento de datos en Nubes con acceso TCP/IP (Internet), así

como las redes personales para la conexión de sensores externos al teléfono.

En el capítulo dos se define la arquitectura del sistema propuesto, así como los diferentes componentes que lo conforman y se aborda el diseño y la implementación de forma detallada de las aplicaciones realizadas.

En el capítulo tres, se informa de los resultados alcanzados en la implementación práctica, donde se detalla el cumplimiento de los objetivos propuestos.

Finalmente se presentan las Conclusiones, Recomendaciones y Anexos.

Para el desarrollo de la investigación se utilizaron los siguientes métodos científicos:

Teóricos:

- Histórico-Lógico: Para el estudio teórico de los antecedentes de los sistemas de posicionamiento global, sistemas de sensores y el uso en aplicaciones Android.
- Inductivo-Deductivo: Para identificar los principales elementos para el diseño de la aplicación.

Empíricos

- Análisis documental: Para evaluar características, parámetros y requerimientos de las aplicaciones Android y las tecnologías utilizadas
- Simulación: para la comprobación del funcionamiento de la propuesta a desarrollar.

## CAPÍTULO I. FUNDAMENTOS TECNOLÓGICOS PARA EL DESARROLLO DEL PROYECTO

En este primer capítulo se aborda el contexto tecnológico en el que se desarrolla esta investigación, así como los fundamentos que le dan soporte. Comenzando con la Agricultura de Precisión, continuando con las Infraestructuras de Datos Espaciales y cerrando el capítulo con las tecnologías para la adquisición de datos espaciales y su envío y almacenamiento.

### 1.1. AP: Agricultura de Precisión

Con el desarrollo de la electrónica, y por consiguiente de la tecnología (tecnologías de comunicación, sensores, minería de datos, inteligencia artificial...), del último siglo, se ha llegado a un mundo en el que la tecnología está integrada en todos los campos, tanto en ámbitos productivos como sociales: la era digital. En el caso de la agricultura no es distinto, de ahí nace lo que se llama Agricultura de Precisión (AP). Esto tiene su razón objetiva, dada la coyuntura mundial en la que abundan los problemas ambientales, energéticos, alimentarios, sociales y económicos.

*“De modo que la agricultura de precisión es la aplicación de tecnologías y principios para el manejo de la variabilidad espacial y temporal asociada a todos los aspectos de la producción agrícola con el propósito de mejorar la productividad del cultivo y la calidad ambiental.” [1]*

La Agricultura de Precisión comúnmente se desarrolla en tres etapas diferentes:

- 1) Recolección de datos del terreno en cuestión.
- 2) Análisis. Procesamiento e interpretación de la información de cara a la toma de decisiones.
- 3) Aplicación de las decisiones.

A pesar de que se diferencien esas etapas, no quiere decir que discurran cronológicamente (aunque puede serlo), ni que sean acciones concretas en el tiempo. De hecho, existen implementaciones de AP que la etapa de recolección de datos es continua, debido a que se instalan sensores que monitorizan continuamente el estado de algo. También, puede ser periódica (cada cierto tiempo se recogen datos para visualizar la evolución) o puntual (que tan solo se haga una o dos recolecciones de datos). Del mismo modo, la relación entre las etapas puede ser distinta: Por ejemplo, que se recolecten datos a medida que se cosecha, se analicen en tiempo real y se tomen las medidas oportunas al momento. O podría ser lo contrario, primero recolectar datos del terreno, después analizarlos. En base a ello tomar decisiones y a posteriori aplicarlas.

El objetivo de la AP es obtener datos para transformarlos en información organizada para la toma de decisiones enfocado en fortalecer la sostenibilidad social, ambiental y económica. Esto hace que cambie como se entiende la nueva agricultura, y con ello la toma de decisiones y la forma de gestionarlas.

### **1.1.1 Tecnologías usadas en AP**

A continuación se exponen los tipos de tecnologías que se aplican frecuentemente en la AP, en la actualidad:

- Sistemas de Navegación Global por Satélite (*Global Navigation Satellite System* o GNSS): Facilitan la posición mediante satélites con emisores de radiofrecuencia. ([1.3.3.GPS: Sistema de Posicionamiento GNSS](#))
- Sensores Remotos (*Remote Sensors* o RS):  
Estos aportan información como altura, temperatura, presión, porcentaje de clorofila, humedad, viento, incidencia de sol...
- Sistemas de Información Geográfica (SIG) o *Geographic Information System* (GIS) :

Estos sistemas ayudan a guardar, analizar y manejar los datos ([1.2.2. SIG: Sistemas de Información Geográfica](#)). En concreto, para los GIS aplicados a la agricultura se emplea el término AgGIS.

- Computadoras:

Hoy en día imprescindibles. Sirven para soportar físicamente la SIG, así como realizar distintas tareas.

- Fotogrametría:

Disciplina que busca obtener información de sensores o imágenes sin contacto con el elemento en cuestión, para poder realizar mediciones, analizar y representar.

- Máquinas o vehículos:

Aunque su función suele ser secundaria (normalmente la de portar el sensor para hacer las mediciones) también se deben mencionar: cosechadoras, fertilizadoras, robots, aviones/avionetas, drones o vehículos aéreos no tripulados (VANT o UAV), satélites sensores...

- Electroválvulas, servomecanismos y otros tipos de actuadores. Sirven para modificar el sistema dosificador, conducción automatizada y un largo etcétera.
- VRT o *Variable Rate Technology*. (Explicada en el siguiente apartado de Técnicas AP).
- Sistemas de ayuda a la decisión:

A menudo, la cantidad de información recabada puede ser muy grande y diferente entre sí, y por consiguiente dificulta la toma de decisión. Para ayudar con esa decisión están estos sistemas, que combinan la información, calculan parámetros e incluso proponen como actuar en cada caso.

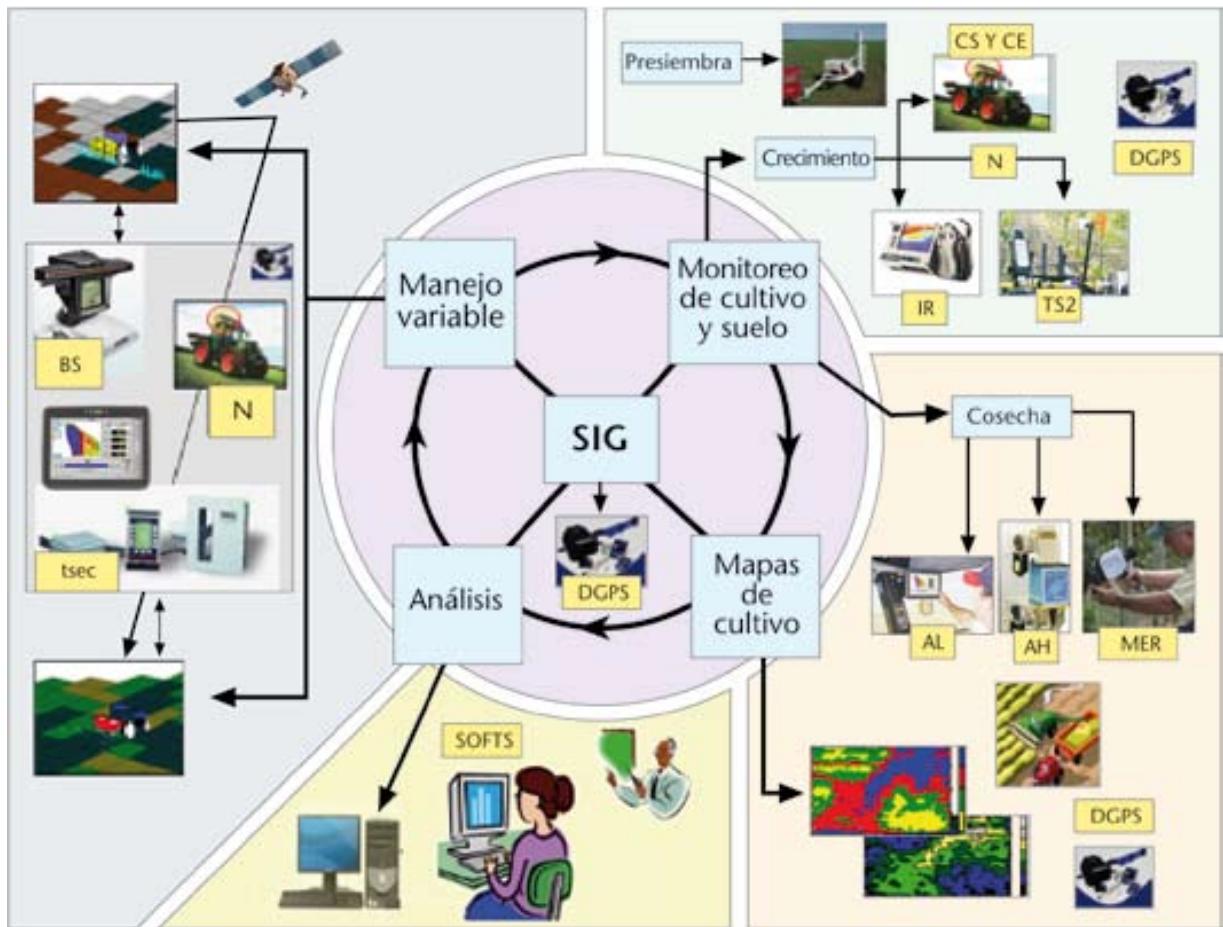


Figura 1: Ciclo de Aplicación de tecnología en la AP [2]

### 1.1.2. Técnicas AP

La AP tiene muchas formas de implementarse y no todas ellas son iguales, sino que todo lo contrario, hay una gran variedad, y más que han de desarrollarse. De hecho, en su evolución ha ido tratando problemas de distinta índole, lo que ha requerido nuevos métodos y sistemas. Dentro de la infinidad de posibilidades, se puede agrupar/clasificar los distintas técnicas por objetivos:

- Técnicas orientadas a la gestión de la variabilidad espacial de las propiedades de los suelos y estado de los cultivos:

Es un hecho que los terrenos cultivables no son uniformes, que existe variabilidad intraparcelaria que surge como consecuencia de la diversidad de las propiedades de los suelos, agentes patógenos o microclimas que afectan de manera selectiva al conjunto parcelario. Un buen conocimiento de la heterogeneidad es muy importante porque no toda la parcela puede necesitar cierto producto o otro, incluso podría no ser beneficioso para una zona y necesario para otras zonas concretas. Esta técnica es usada para muchos fines como la fertilización, tratamientos herbicidas, laboreo con profundidad variable, siembra a distintas densidades, riego con dosis adaptadas...

Para esta labor existen máquinas agrícolas equipadas con tecnología de distribución variable (VRT: *Variable Rate Technology*), que permiten aplicar dosis distintas según la necesidad de la zona y sin que el operario tenga que intervenir. Existen dos tipos:

- Máquinas VRT basadas en sensor:

Un sensor óptico analiza el espectro de luz reflejada por el cultivo (tele-detección), y con dicha información ajustan la dosis que se aplica en dicho cultivo, mediante un mecanismo de control del dosificador. Además, existen prometedoras investigaciones que combinan el sensor óptico con cámaras y análisis de imágenes, para tener mejor identificación de “malas hierbas” etc.

- Máquinas VRT basadas en mapas:

A diferencia del sistema anterior, el análisis del cultivo se hace previamente al aplicado del producto. Mediante estos análisis se generan los mapas de prescripción, que son los que indican los cambios de dosis según la zona. La información de estos mapas es controlada mediante computadoras con Sistemas de Información Geográfica para Agricultura (AgSIG). En estos mapas se suelen reflejar distintas informaciones: propiedades químicas, compactación de la tierra, humedad, conductividad eléctrica, plagas y enfermedades... Con esta técnica es necesario un

receptor de posicionamiento GNSS, para que en función de la ubicación de la máquina se aplique la dosis adecuada.

- Técnicas orientadas a la ayuda al guiado y a la uniformidad de las operaciones mecanizadas:

Estas técnicas hacen que a la hora de aplicar productos sobre el suelo, se distribuya de forma uniforme. Aunque entra en contradicción con la variabilidad terrenal anteriormente explicada, en casos de cultivos extensivos de grandes terrenos la variabilidad es difícil de tener en cuenta y menos significativa. De modo que se aplica una dosis homogénea calculada a partir de un promedio de todo el terreno.

Existen diferentes causas que provocan que no se aplique la misma cantidad de producto en toda la parcela:

- Cuando se aplica un dosificador con caudal constante pero la máquina agrícola va a distintas velocidades:

Las zonas donde va más despacio recibirá más dosis que en las que vaya más rápido. La solución más simple y antigua es mecánica: a través de una rueda conectada a la tracción de la máquina regula la dosis según la velocidad de la misma. No obstante, el resbalamiento y el distinto hundimiento de la rueda en el suelo hace que no sea del todo exacto, pero se acerca bastante. Para buscar exactitud existen sistemas más avanzados como puede ser mediante receptores GNSS que varían el caudal según la posición.

- Cuando las pasadas contiguas de la máquina se solapan o se generan huecos:

Sucede cuando la máquina agrícola pasa dos veces por el mismo sitio o lo contrario, deja una zona sin pasar. Para evitarlo surgen los sistemas de ayuda al guiado y guiado automático vía posicionamiento GNSS. Funciona programando la línea de partida y la distancia que se quiere dejar entre líneas. Con esa información el sistema

genera las líneas paralelas y va comparando la posición de la máquina con la línea más próxima. Si el sistema es de ayuda al guiado, se muestra en una pantalla la situación y avisa si se sale de la línea para que el conductor corrija la desviación. Los sistemas de guiado automático, actúan en la dirección de la máquina y corren la desviación sin necesidad de intervención del conductor.

Además, existe otro sistema llamado “control de tramos”. Es muy útil en casos, en los que la máquina esté obligada a desviarse de su trayectoria por existencia de algún obstáculo como árboles, postes... El sistema detecta las zonas ya trabajadas y no aplica el tratamiento en cuestión, lo detiene hasta llegar a una zona no trabajada.

### **1.1.3. Conclusión**

Como se puede observar, los sistemas que se han desarrollado en general, tanto en su investigación como en su uso, son de alto coste. Esto supone que su uso se va a limitar a grandes parcelas y proyectos de grandes dimensiones, y es un hecho que las técnicas de guiado y uniformidad han sido las más desarrolladas y utilizadas, orientadas a grandes producciones. No obstante, no se debe olvidar que la AP también puede ayudar a los pequeños productores familiares y comunales a ser menos dependiente de insumos externos a la explotación, mejorando la sostenibilidad. Para ello, hay que desarrollar tecnología de bajo coste y sencilla en su uso, al alcance de todas y todos. Es por ello que este proyecto va dirigido al pequeño agricultor que en la mayoría de los casos solo necesita variables como la altitud para planificar los sistemas de riego o conocer la humedad de cada zona para dirigir el mismo, o rectificar problemas de desniveles o drenajes. O simplemente para conocer límites de parcelas. Además, permite el análisis de la temperatura de las diferentes áreas, propiciando la planificación del riego o las posibilidades de riesgo de plagas.

Por otro lado, la AP abre un mundo de posibilidades que no tiene porque estar ligada únicamente a la agricultura. No obstante está poco desarrollada para otros usos, como lo puede ser para temas ambientales o conservación de la tierra.

## **1.2. IDE : Infraestructuras de Datos Espaciales - Spatial Data Infrastructures (SDI)**

En este apartado se analizan las tecnologías actuales de las Infraestructuras de Datos Espaciales. Lo que se expone a continuación es la fusión de la informática y la cartografía y el desarrollo que ha tenido. Es importante que previa explicación de las IDEs se aclaren los conceptos de IG y SIG, dado que tienen gran importancia y no se puede comprender que es una IDE y cuál es su origen sin las siguientes explicaciones:

### **1.2.1. IG: Información Geográfica**

La Información Geográfica (IG) es aquella información que se relaciona con localizaciones de la tierra. IG puede ser cualquier información cartográfica como mapas, orto-fotos, imágenes de satélite... o cosas más complejas, con múltiples componentes o capas, con información muy diversa. Cada vez adquiere mayor importancia y valor, dado que, es muy necesaria para una gestión eficiente de los recursos naturales, así como, para el control de fenómenos naturales, cuidado del medio ambiente y un largo etcétera. De hecho, ha tenido un crecimiento muy grande a medida que se ha ido descubriendo su utilidad y se han ido desarrollando las Tecnologías de la Información (TI). A la disciplina entorno a la IG se le puede llamar Geomática o Tecnologías de Información Geográfica (TIG).

### **1.2.2. SIG: Sistemas de Información Geográfica**

Los sistemas que se encargan de almacenar, consultar, gestionar y analizar datos IG (en grandes cantidades) se llaman Sistemas de Información Geográfica (SIG). La información que maneja es información espacial georreferenciada, lo que combina información geométrica con información temática.

Los SIG están formados por componentes hardware y software:

En cuanto al hardware hay distintos elementos: Computadora de trabajo, unidades de almacenamiento masivo de IG, redes informáticas y sistemas de salidas (pantallas, impresoras...).

Por otro lado, en software se necesita una Base de Datos y un programa o software que le dote de inteligencia al conjunto del SIG y permita al usuario realizar las distintas tareas.

Los SIG ayudan en la toma de decisiones frente a distintos escenarios, de ahí su valor. No obstante, estos sistemas están aislados de otros sistemas y de la obtención de la IG. Además, para el volcado de esos datos al SIG (que pueden tener formatos muy distintos) o para operar con otros SIG (que pueden usar otros protocolos muy distintos), es necesaria la intervención humana.

### **1.2.3. IDE**

De las carencias y el desarrollo de los SIG nace la Infraestructura de Datos Espaciales (IDE), cuyas características más importantes son: interoperatividad, accesibilidad, cooperación e intercambio. Existen múltiples definiciones de IDE, en las que cada organismo que la define hace hincapié en algún aspecto más que en los otros [3]. Una definición puede ser la siguiente: Es una infraestructura que soporta IG de forma integral y eficaz para compartirla, analizarla y acceder a ella, mediante un conjunto de recursos como pueden ser aplicaciones, servidores web y bases de datos, como ayuda en la toma de decisiones y el desarrollo de nueva información.

La IDE, no debe ser un concepto fijo e invariable, sino que debe dar pie a seguir desarrollándose y de ir adquiriendo una complejidad mayor, y a su vez mayor prestación. Puede llegar a ser una infraestructura muy potente tanto en la toma de decisiones, así como en la generación de nueva información. Aplicando tecnologías de *Internet Of Things* (IoT), *Data Mining* (DM) y de Inteligencia Artificial (IA) llegará muy lejos.

#### **1.2.4. Interoperatividad: Normas, estándares y protocolos**

Es importante y necesario que la IG sea interoperable entre distintas IDEs y sistemas. Esto significa que sean compatibles para comunicar, compartir y cooperar entre las distintas infraestructuras sin intervención mediadora humana. En consecuencia, se logra reutilizar los datos generados en un proyecto, y utilizarlos en otros, de manera colaborativa. De esta manera se ahorra la producción de los mismos y los generados sirven a otros proyectos. También, porque es muy conveniente evitar información duplicada y tener acceso a la información más actualizada. Además, existen fenómenos transfronterizos (migración de animales salvajes, epidemias, contaminación...), que no entienden las artificiales fronteras humanas, y para analizarlas es imprescindible la adquisición de la IG en todos los territorios que abarque el fenómeno.

Para ello, existen distintos organismos en todo el mundo que tratan de estandarizar y normalizar este tipo de dato, creando normas, estándares especificaciones y protocolos. En concreto, el *Open Geospatial Consortium* (OGC) [4] y la *International Organization for Standardization* (ISO).

El OGC es una organización sin ánimo de lucro y formada por universidades, empresas y administraciones públicas que ha desarrollado e impulsado estándares y protocolos para lograr cierta uniformidad. Ellos crean los estándares abiertos, pero son los que crean las IDEs los que deciden si aplicarlos para ser compatibles o no.

Por otro lado, el ISO se encargar de la normalización asociada a las IDEs. Lo hace a través del comité técnico ISO/TC 211, que se encarga de la Geomática y la IG. Hay que mencionar que junto con la OGC crearon la familia de normas ISO 19100, muy importantes para la normalización de las IDEs.

Finalmente, existe una serie de recomendaciones y acuerdos dadas por organismos territoriales, para favorecer la interoperatividad del territorio en cuestión.

### **1.2.5. Origen y desarrollo**

Todo el desarrollo que se ha dado no sería posible sin el desarrollo tecnológico que se ha dado en el último siglo; creando sistemas de posicionamiento, computadoras... potentes y adquiribles. También, el software libre ha sido muy importante, ya que se empezaron a crear herramientas software SIG con esta filosofía, extendiendo así su uso. Como ya se ha mencionado anteriormente, el hecho de que a su vez se fue generando mucha IG, tras ver su valor y poder, hacen necesario el desarrollo de nuevas infraestructuras para su manejo y para “exprimir” más su potencial.

Esas razones del surgimiento de las IDE, van de la mano de hechos políticos relevantes que clamaron la creación y desarrollo de IDEs:

Todo surgió con la Conferencia de las Naciones Unidas sobre Medio Ambiente y Desarrollo en Río de Janeiro en 1992 [3], donde se visualizó la relevancia de la IG para tratar los problemas hablados. Pero la expansión de las IDE comenzó en 1994 con la Orden Ejecutiva 12906 del Presidente Bill Clinton, señalando que “*La Información Geográfica es crítica para promover el desarrollo económico, mejorar nuestra gestión de los recursos naturales y proteger el medio ambiente*”. Así el gobierno norteamericano creó la NSDI (*National Spatial Data Infrastructure*). De este modo, en el marco territorial de EE.UU. las administraciones pueden acceder y compartir las IG más actualizadas. Ese mismo año se fundó la OGC. En 2004 se creó la asociación GSDI (*Global Spatial Data Infrastructure*), para la cooperación internacional. Luego se unió la Comisión Europea con la Directiva 2007/2/CE, a través de la cual se creó INSPIRE (*Infraestructure for Spatial Information in Europe*), de obligado cumplimiento a partir de 2007. Cada país tiene su propia IDE, en concreto en el estado español existe la IDEE, que intenta homogeneizar e integrar la información generada por distintas administraciones públicas. De forma similar sucedió en los países de Iberoamérica: Instituciones nacionales y regionales como el

Instituto Panamericano de Geografía e Historia (IPGH) o el Comité Permanente para las IDE de las Américas (PC-IDEA) creadas para apoyar las IDE y coordinar la información geográfica en Iberoamérica. Otro ejemplo es el de la IDE de la República de Cuba (IDERC) [5].

No obstante, no todo es tan sencillo, hay ciertas circunstancias y contextos políticos que dificultan compartir Información Geográfica. Y es que hacerlo tiene sus consecuencias, y cada organismo/país define sus políticas de acceso y uso, limitando las IDEs.

#### **1.2.6. Componentes IDE**

Los elementos que componen una IDE son: Estándares, datos, metadatos, software y servicios IDE y redes de comunicaciones:

- Estándares:

Se trata de un conjunto de especificaciones aplicadas a una materia de manera que dos entes “distintos” puedan entenderse, cual “idioma”. Ya se ha explicado su importancia para la interoperatividad.

- Datos:

Se clasifican en dos tipos:

- Datos de referencia: Forman el mapa base o la base en la que se refieren los datos temáticos. Están ligados a la localización espacial en el terreno, datos de cartografía básica. Pueden ser coordenadas, redes de transporte o hidrológica, relieve, límites administrativos...
- Datos temáticos: Aportan información sobre un fenómeno o alguna propiedad en concreto y por tanto tiene un fin en sí mismo. Clima, industria, vegetación, población, circulación, recursos...

- Metadatos:

Son aquellos datos que describen como son los datos espaciales que conforman la IG o los servicios que puede ofrecer una IDE, para conocer sus características y identificarlos de manera única:

- ¿Qué? Título y descripción. Calidad de datos.
- ¿Cómo? Escala, sistema de referencia por coordenadas.
- ¿Cuándo? Fecha de obtención de los datos.
- ¿Quién? Responsable de la generación de los datos, administrador, usuarios, restricciones...
- ¿Dónde? La extensión geográfica de los datos, si es un servicio web la URL.
- ¿Por qué? Razones para la recogida de los datos y su uso.

Es imprescindible que el usuario de estos datos conozca bien frente a qué datos/servicios está, porque, a menudo, los creadores de estos datos y los que los usan no son las mismas personas, y eso requiere una representación y entendimiento común. Eso da la posibilidad de tratarlos de forma adecuada para el fin concreto. Así se ahorran fuerzas y se agiliza el trabajo, además de mejorar los resultados que se obtendrán. Además, el usuario debe poder interpretar los metadatos sin problemas, es por ello que también existen estándares a la hora de la definición de los datos:

La Norma ISO de la familia ISO 19100 que determina los Metadatos es la: ISO 19115 “Geographic information – Metadata”. Esta norma Internacional proporciona un modelo de Metadatos con terminología, definiciones y procedimientos comunes. También existen organismos que crean otros estándares o variaciones del mismo, así como implementaciones concretas territoriales.

- Software y servicios IDE

Hay gran cantidad de software que integran las funcionalidades propias de una IDE. Existe tanto software libre como propietario. Es importante que cumplan con el estándar de la OGC.

Los servicios son software que ofrecen funcionalidades a los usuarios, mediante interfaces y librerías. Hay distintos tipos de servicios documentados: Servicio de Mapas en Web (WMS), Servicio de Fenómenos en la Web (WFS), Servicio de Coberturas en Web (WCS) y Servicio de Catálogo (CSW). Está muy extendido el uso de Geoportales Web IDE, que mediante un navegador, conectándose a un servidor web permite consultar, visualizar y usar ciertas funcionalidades de la IDE. Para esta labor también existen aplicaciones propias.

- Redes de comunicaciones:

Hoy en día es imprescindible que las IDEs estén conectadas, y no resulta difícil gracias a la extensión que tiene Internet. Es la forma de hacer accesibles las IDEs y de difundir Información Geográfica.

### **1.2.7. Conclusión**

Una IDE puede llegar a ser una infraestructura muy compleja, extensa, completa y muy potente. No obstante, cuanto mayor es, resulta ser más complicada de realizar por requerimientos, así como por razones económicas. Por tanto una IDE de alto nivel es difícil de adquirir, solamente es posible para grandes proyectos o gobiernos.

En este proyecto, el sistema que se ha desarrollado, tiene muchas de las características propias y funcionalidades de las IDEs. Además, a sido creada pensando en la interoperatividad, usando el formato estándar GeoJSON. Se puede considerar como una Infraestructura de Datos Espaciales (IDE) a pequeña escala, y con funcionalidades concretas, que podría expandirse y abarcar nuevos retos.

### **1.3. Sistema de obtención de datos geográficos**

En este apartado se presentan las tecnologías concretas utilizadas para el desarrollo de la tesis, así como los dispositivos usados. Ha de anotarse que estas tecnologías son bastante recientes, pero que ya están altamente instauradas en la sociedad y son de fácil adquisición, por lo que va en línea con los objetivos del proyecto. El elemento fundamental es el terminal Android. Estos terminales no solo hacen la función de teléfonos móviles, sino que incluyen una gran cantidad de posibilidades con la cantidad de tecnologías y sensores que incorporan hoy en día cualquiera de estos dispositivos. De modo que se explica el propio sistema operativo, y seguidamente, las tecnologías que incluye un terminal Android y que son necesarias para su uso en la captura de IG en este proyecto.

#### **1.3.1. Sistema operativo Android**

Android es un sistema operativo creado para funcionar en dispositivos (normalmente con pantalla táctil) como *smartphones*, *tablets*, relojes inteligentes, televisores, automóviles... Su desarrollo lo empezó la empresa Android Inc. respaldada económicamente por Google, y comprada en 2005. Dos años más tarde se creó la *Open Handset Alliance* (OHA) con el fin de desarrollar estándares abiertos para teléfonos móviles. La OHA es una alianza comercial de diversas compañías tecnológicas (fabricantes de dispositivos móviles, desarrolladores de aplicaciones, operadores de comunicaciones, fabricantes de chips...), y liderada por Google. Junto con la formación de la OHA, presentaron el sistema operativo Android. Actualmente, el sistema operativo ha llegado a su octava versión, siendo el sistema operativo móvil más utilizado en el mundo, con una cuota de mercado superior al 80% en 2017, muy por encima de IOS. [6]

Android se caracteriza por que empezó siendo un sistema operativo de código abierto. No obstante, Google ha ido cerrándolo poco a poco, sustituyendo las

funciones del núcleo libre AOSP por la capa de software privativo Gapps [7]. A pesar de todo, el núcleo linux, el framework, el interfaz del sistema, la instalación inicial y una serie de aplicaciones básicas son de código abierto [8]. También, existe una plataforma para desarrolladores en la que se ayuda y se permite el uso de múltiples herramientas del sistema y de los dispositivos móviles.

### **1.3.1.1. Características**

Las principales características de este sistema operativo son:

- Plataforma adaptable a pantallas de mayor resolución.
- Utilización del sistema de gestión de bases de datos relacionales SQLite.
- Soporte de un gran número de tecnologías de conectividad: GSM, EDGE, WiFi, HSDPA, LTE, entre otros.
- Soporte para el envío de mensajes de texto de tipo SMS (del inglés Short Message Service) y MMS (del inglés Multimedia Message Service).
- Navegador web basado en WebKit con soporte para JavaScript.
- Soporte para gran número de formatos multimedia.
- Soporte para recepción continua de datos, o streaming.
- Soporte para dispositivos adicionales: cámara de fotos y vídeos, receptores GPS, acelerómetros, sensores de luz, sensores de proximidad, etc.
- Soporte para pantallas multitáctil.
- Soporte para red inalámbrica personal Bluetooth.
- Soporte para video llamadas.
- Soporte para multitarea.
- Reconocimiento de voz.[9]

### 1.3.1.2. Arquitectura

La arquitectura de este sistema operativo es por capas. De la capa más cercana al hardware (kernel) hasta la capa software (aplicaciones):

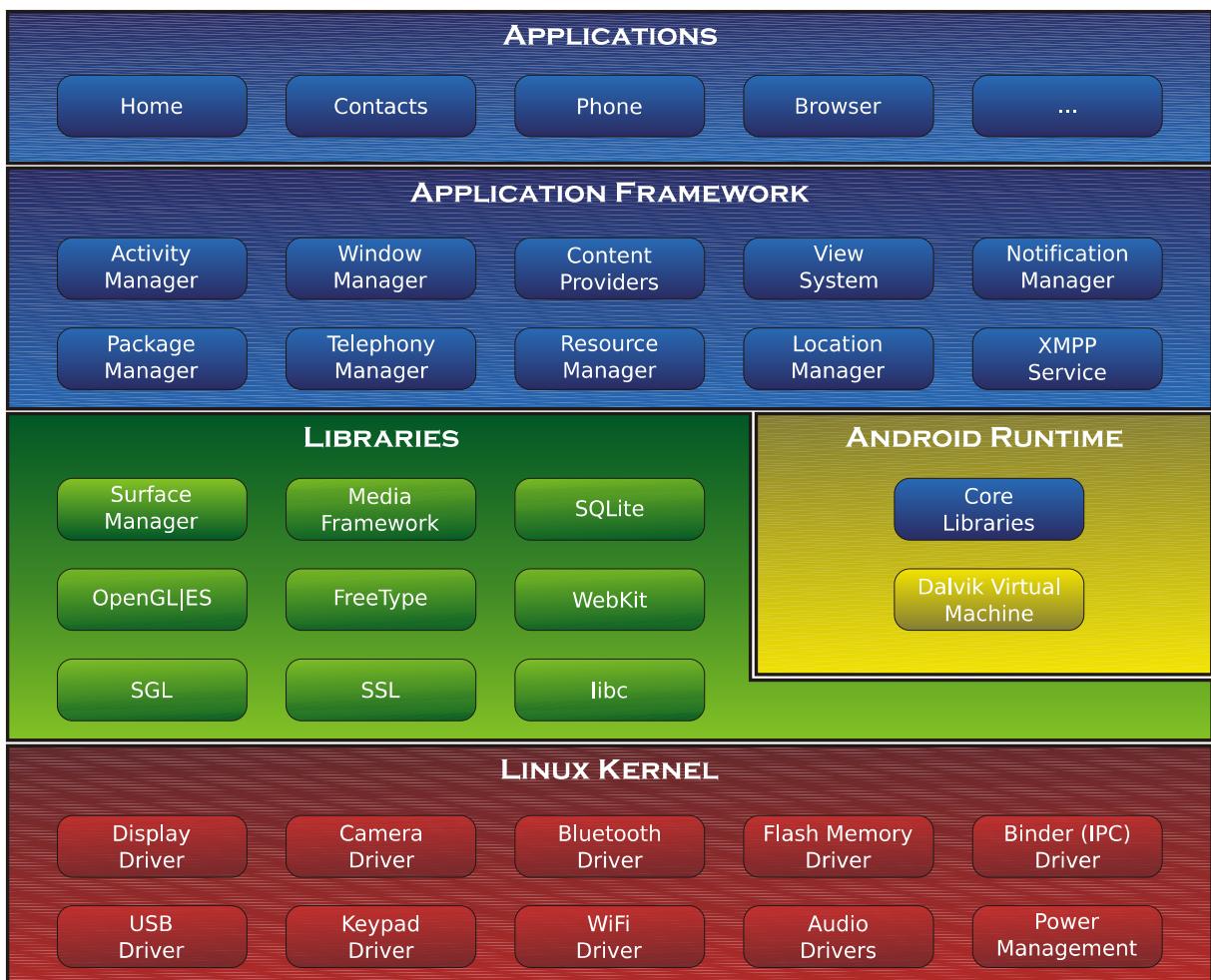


Figura 2: Arquitectura del sistema Android [6]

- Kernel de Linux

El núcleo principal es una modificación del kernel Linux, para adecuarlo al entorno de dispositivos móviles. Este núcleo es la abstracción entre hardware y software, que permite interactuar al software superior con el hardware. Se

encarga de los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores.

\*NOTA: No se debe mezclar linux con los sistemas operativos GNU/Linux.

- Librerías

La siguiente capa es un conjunto de librerías nativas en C/C++ que son usadas para manipular ciertos componentes. Las librerías son las siguientes: *Surface manager* (gestor gráfico de ventanas), *Media framework* (manejo de audio e imagen), *SQLite* (motor de base de datos), *OpenGL* (librería gráfica de 2D y 3D), *FreeType* (gestión gráfica de fuentes de texto), *Webkit* (motor de renderización web), *SGL* (*Skia Graphics Engine*), *SSL* (*Secure Sockets Layer*) y *libc* (librería c estándar).

- Runtime de Android

Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecutaba hasta la versión 5.0 archivos en el formato de ejecutable Dalvik (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato .dex por la herramienta incluida dx. Desde la versión 5.0 utiliza el ART, que compila totalmente al momento de instalación de la aplicación.[6]

Se aprecia que a pesar de desarrollarse en Java, no se utiliza una máquina virtual de Sun para su ejecución ni tampoco archivos .class.

- Marco de trabajo de las aplicaciones (*Framework*)

Se trata de la API que interactúa con el programador (a nivel de aplicación, como indican las iniciales). Facilita al programador el acceso a todos los recursos y gestores para la creación de aplicaciones. Las más importantes son las siguientes:

*Activity Manager* (gestiona el ciclo de vida de las apps), *Window Manager*, *Telephone Manager* (llamadas, mensajes...), *Content Providers*, *View System*, *Location Manager* (localización y posicionamiento), *Notification Manager*, *XMPP Service*, *Package Manager*, *Resource Manager*, *Sensor Manager*, Cámara y Multimedia.

- Aplicaciones

Las propias de Android (Google) y las de cualquier desarrollador. Escritas en lenguaje de programación Java.

#### **1.3.1.3. Android Studio**

Es el entorno de desarrollo integral (IDE) oficial de Google para los desarrolladores de aplicaciones. Este programa esta disponible en la web de desarrollo de Android, bajo una licencia de software libre Apache 2.0. Está programado en Java y es multiplataforma. De esta forma Google facilita el desarrollo de apps, lo que lo beneficia económicamente, y logra un mayor control sobre la producción respecto al anterior entorno (Eclipse IDE con el complemento *Android Developer Tools*).

#### **1.3.1.4. Estructura de un proyecto Android**

Los elementos (código, recursos gráficos, datos...) están organizados en directorios diferenciados por funciones y tipo de lenguaje, dado que Android incluye ficheros en lenguaje XML con el fin de evitar cargar los ficheros java con texto. Todo ello, hace más clara la comprensión del código fuente y la navegación entre los recursos. A continuación se listan los recursos y carpetas relevantes en cualquier proyecto Android:

- Fichero AndroidManifest.xml

Contiene todos los aspectos principales de la aplicación. En este fichero se define el nombre de la aplicación, icono, componentes, permisos necesarios... Ubicado en: “/src/main/AndroidManifest.xml”.

- Carpeta /src/main/java.

En este directorio se encuentran todos los ficheros .java con el código fuente (interfaces, objetos, clases auxiliares...).

- Carpeta /src/main/res:

Contiene todos los recursos necesarios como imágenes, cadenas de textos, diseños ... Los recursos se clasifican a su vez en más directorios dentro de esta carpeta:

- Anim: Archivos XML que definen las animaciones. Directorio: “/src/main/res/anim”.
- Drawable: En este directorio se encuentran las imágenes en formato PNG o JPEG. Su ubicación es: “/src/main/res/drawable”.
- Layout: Aquí se encuentran todos los diseños de la interfaz gráfica (*layout*) de nuestro proyecto. Se ubican en el directorio “/src/main/res/layout”.

- Menu: Archivos XML que definen las plantillas de los menús del proyecto. Directorio: “/src/main/res/menu”.
  - Mipmap: Contiene los iconos de la aplicación con sus diferentes resoluciones. Anteriormente se encontraban en el directorio *drawable*, pero en las últimas versiones de Android Studio han sido movidos a este directorio. Ubicación “src/main/res/mipmap/”.
  - Raw: En esta carpeta se ubican los archivos multimedia descomprimidos, de esta manera Android sabe que no debe procesarlos más tarde. Se encuentra en el directorio “/src/main/res/raw”.
  - Values: Archivos XML que definen valores constantes. Por cada tipo de valor existe un fichero: cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), colecciones de valores (arrays.xml), tamaños (dimens.xml) etc. Se encuentra en: “src/main/res/values”. [10]
- Fichero /app/build.gradle
- Contiene información necesaria para la compilación del proyecto: versión del SDK, mínima versión Android soportada, bibliotecas externas utilizadas...
- Carpeta /app/libs
- Puede contener las bibliotecas de Java externas (ficheros .jar) que utilice la aplicación. Normalmente se hace referencia a ellas en el fichero build.gradle, para que formen parte de la compilación.
- Carpeta /app/build/
- Contiene elementos de códigos en Java generados automáticamente al compilar el proyecto.

### **1.3.1.5. Componentes de una aplicación**

En este apartado se analizan los distintos elementos software que componen una aplicación en Android. Se pueden implementar hasta varios tipos de componentes. Estos deben ser declarados de forma explícita en el fichero *AndroidManifest.xml*:

- Activity

Representan el componente principal de la interfaz gráfica. Cada *activity* es una ventana o pantalla, que es independiente de las demás, lo que no significa que no puedan estar interconectadas.

- View

Son los componentes básicos para construir la interfaz gráfica de la aplicación. Android pone a disposición del público una gran cantidad de controles: cuadros de texto, botones, listas desplegables, imágenes... También existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

- Service

Estos componentes no tienen interfaz gráfica y se ejecutan en segundo plano. Sirven para realizar cualquier tipo de acciones como actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales.

- Content Provider

Es el mecanismo de Android que permite compartir una serie de datos entre distintas aplicaciones. Por ejemplo: los contactos.

- Broadcast Receiver

Sirve para detectar y reaccionar ante mensajes globales generados por el sistema. Por ejemplo: aviso de batería baja.

- Widget

Son los elementos visuales que se muestran en la pantalla principal, con la finalidad de tener una interacción rápida con el usuario o mostrar actualizaciones en dicha pantalla.

- Intent

Es el elemento de comunicación de los componentes anteriormente explicados. Permiten hacer cosas como mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje *broadcast*, iniciar otra aplicación...

[11]

### **1.3.2. Bluetooth**

A continuación, se procede a explicar el funcionamiento de la tecnología *Bluetooth* integrada en un teléfono móvil inteligente con sistema operativo Android, dado que en el proyecto se usa esta tecnología de enlace de datos para comunicar con el sistema de sensores externos:

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPANs) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por Radiofrecuencia en la Banda ISM de los 2,4 Ghz. La especificación de Bluetooth definiría un canal de comunicación de máximo 720 kb/s con rango óptimo de 10 metros (opcionalmente 100 metros con repetidores). Su frecuencia de tráfico, con la que trabaja, se encuentra en el rango de 2,4 a 2,48 GHz con amplio espectro y saltos de frecuencia con posibilidad de transmitir en Full Duplex con un máximo de 1600 saltos/s, los cuales se dan entre un total de 79 frecuencias con intervalos de 1Mhz. [12]

Se aprecia que la tecnología *Bluetooth* sirve para transmisión de datos punto a punto (*point-to-point*) o multipunto (*multipoint*) sin cables y a corta distancia. Esto da pie a

muchas funcionalidades si se integra en un teléfono móvil, y por eso hoy día, todos lo incluyen. Además, junto con los móviles es una tecnología clave para el *Internet of Things* (en auge) ya que sirve para conectar e intercambiar información entre todos los dispositivos, de una forma breve, sencilla y práctica.

La tecnología *Bluetooth* nació en 1994 de la mano de la compañía Ericsson. En 1999, se unieron otras compañías (Intel, Nokia, Thosiba e IBM) y posteriormente muchas otras hasta llegar a más de 20.000. En sus comienzos estaba muy limitado y tenía muchas carencias (velocidad de transmisión, consumo de batería...), pero versión a versión se a ido mejorando. En la actualidad su uso está muy extendido y la tecnología desarrollada. La última versión más extendida es la 4.0, aunque recientemente la 5.0 ya está lista. En la versión cuatro hay un subconjunto llamado *Bluetooth Low Energy* (BLE), con pilas de protocolos totalmente distintas, con el objetivo de permitir crear enlaces rápidos y sencillos con bajo consumo de energía para pequeños dispositivos con pila de botón.

En el proyecto se usa BLE para recibir en el terminal Android la información que el sensor adquiere, es decir, para la obtención de datos de temperatura y humedad del sensor externo BeeWi SmartClim.

Con BLE existen 2 formas de transmitir datos: mediante mensajes anunciadores (*advertisements*) o mediante conexión directa. Dado que en el proyecto se usan los *advertisements* para obtener la información del sensor, se explica en que consisten:

En cuanto a la capa física, es parecida a la del *bluetooth* clásico pero tiene diferencias sustanciales que las hacen incompatibles ( la distribución de los canales es distinta):

Tabla 1: comparación BLE y *Bluetooth* clásico [13]

	BLE	CLASIC	
Type	BLE	BR	EDR
Modulation	GFSK 0.45 to 0.55	GFSK 0.28 to 0.35	DQPSK / 8DPSK
Date Rate	1Mbit/s	1 Mbit/s	2 and 3 Mbit/s
Channels	40	79	79
Spacing	2MHz	1MHz	

Los 2.4GHz de ancho de banda utilizado por *Bluetooth* se encuentra entre las frecuencias 2402MHz y 2480MHz. La versión *Low Energy* usa 40 canales de 1MHz con guarda de 2MHz entre canales. De los 40 canales solo se usan 3 para los paquetes *advertisement*, y los demás se usan para las conexiones. Al enviar esos paquetes se envían por los tres canales, uno tras otro en un cierto intervalo variable. Podría ser no enviar por los tres, sino por menos, con el fin de gastar menos energía, aunque si están ocupados no servirá.

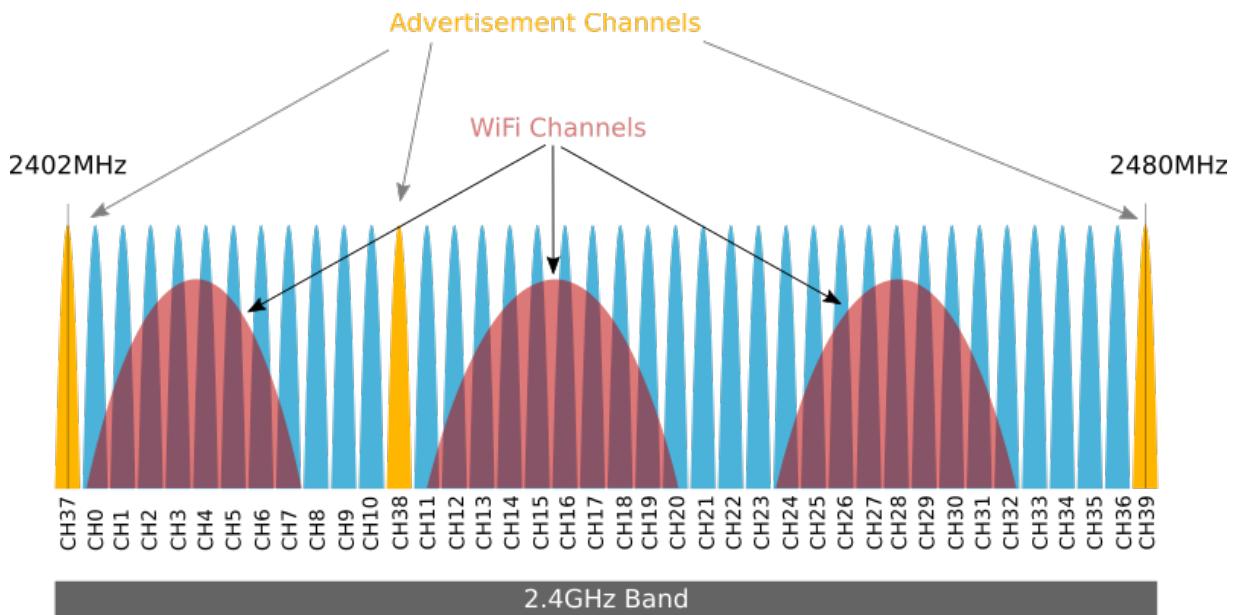


Figura 3: Distribución de canales BLE en el espectro [13]

Una trama *advertising* BLE también es distinta a la estructura de paquete del *bluetooth* clásico, y consta de las siguientes partes: Preámbulo, dirección de acceso, Unidad de datos del *advertising* o PDU (*Packet Data Unit*) y un código de redundancia cíclica (CRC). La PDU contiene una cabecera de dos bytes y una carga útil de datos que varía entre 6-37 Bytes:

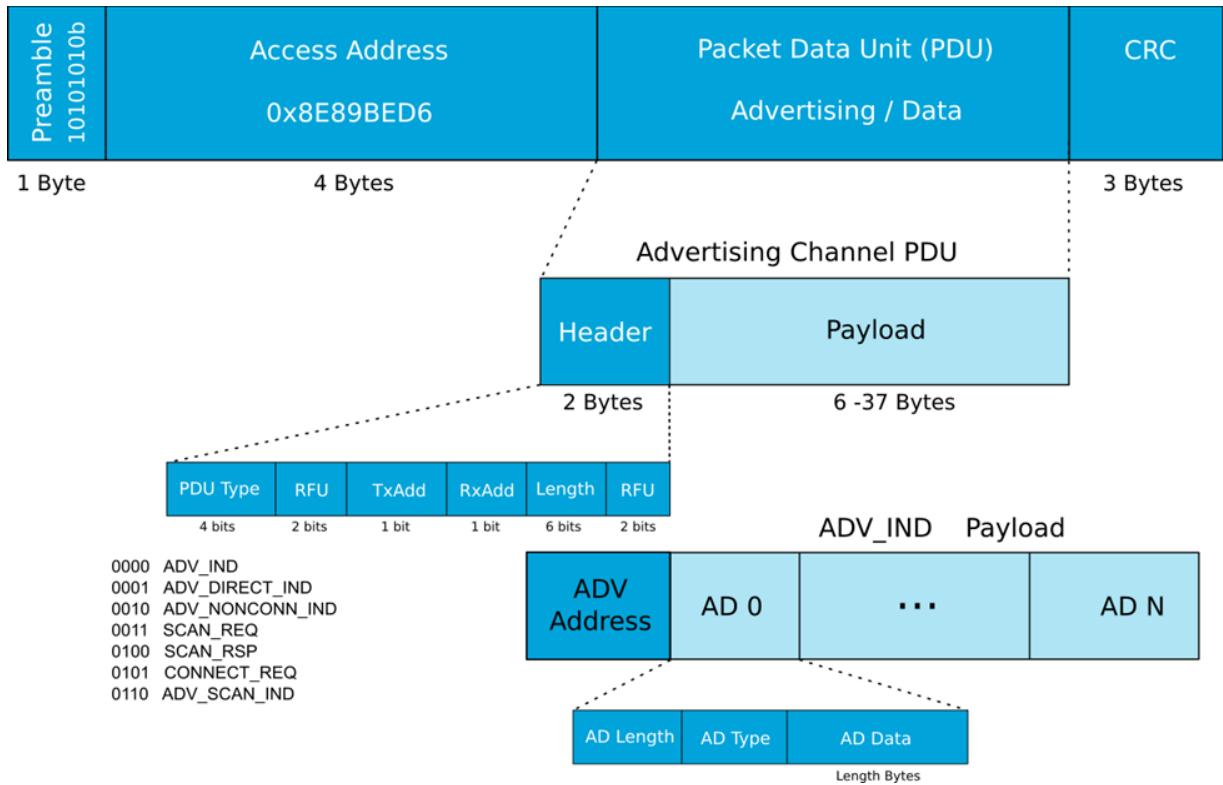


Figura 4: Paquete Advertising BLE [13]

### 1.3.3. GPS: Sistema de posicionamiento GNSS

En los apartados anteriores se ha señalado la importancia de obtener una buena ubicación para conocer el posicionamiento sobre la tierra. La llegada de los *smartphones* ha hecho que hoy en día cualquiera pueda saber su posición, ya que la mayoría de estos dispositivos incluyen un receptor de posicionamiento.

Las siglas GNSS hacen referencia a Sistemas de Posicionamiento Global por Satélites. Hoy en día existe más de un sistema de este tipo: GPS, GLONASS, GALILEO, COMPASS/BEIDOU, IRNSS, QZSS... El sistema GPS (*Global Positioning System*) es hoy día el más difundido, el que incorporan los *smartphones*, y por ello el que se usa en esta tesis:

El sistema GPS está diseñado para que se pueda usar en cualquier lugar del planeta Tierra, a cualquier hora y en cualquier condición climática. Para poder dar tal servicio es necesario tener al menos 24 satélites orbitando y en funcionamiento. Se encuentran aproximadamente a 20.200 Km sobre la superficie y se reparten el globo en seis planos orbitales, donde en cada uno hay al menos cuatro satélites. Esto se hace para asegurar que cualquier receptor independientemente de su ubicación espacio-temporal tenga comunicación con un mínimo de cuatro satélites (lo necesario para lograr la ubicación). Además existe un segmento de control que con distintas bases de monitoreo Americanas en distintos lugares del mundo, se sigue el estado de los satélites, se envían los ajustes necesarios y se calculan las órbitas. Un usuario solo necesita un receptor con su antena para permitirle comunicarse con los satélites y obtener así la ubicación.

Cada satélite tiene sus códigos específicos emitidos constantemente, en dos ondas portadoras L1 (1 575,42 MHz) y L2 (1 227,60 MHz). Sobre las ondas portadoras se modulan los códigos: C/A en la portadora L1, de uso civil y sobre las portadoras L1 y L2 el código P, de uso militar. Además de los códigos, se transmiten diversos mensajes sobre los satélites en funcionamiento (almanaque), correcciones de los relojes, calidad de las señales y otras informaciones del sistema. [1]

El cálculo de la posición se basa en el principio de triangulación: Sabiendo la posición de varios satélites y la distancia hacia ellos, es posible calcular la ubicación de dicho punto.

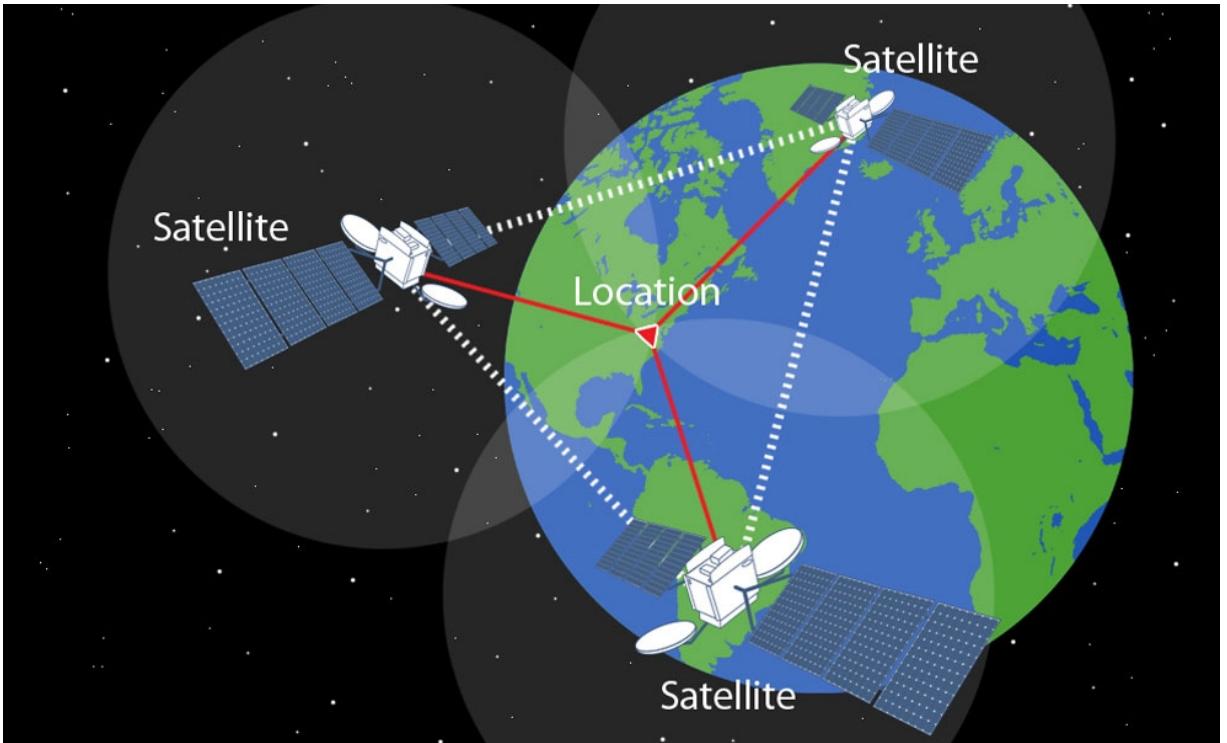


Figura 5: Principio de triangulación

No obstante, para conocer la distancia ( $d$ ) hacia los satelitales es necesario conocer la velocidad de la propagación de la señal que se transmite desde el satélite hasta el receptor (es igual a la velocidad de la luz,  $v$ ) y el tiempo que le lleva a la señal llegar al receptor ( $t$ , medido gracias a la sincronización de los relojes).

$$d = v * t$$

La triangulación es un proceso en el que se van añadiendo las informaciones transmitidas por cada satélite, y que cuanta más información de distintos satélites haya, más preciso será la ubicación obtenida. Todo empieza con una primera medición, donde se calcula a qué distancia está el receptor del satélite. Con esta información se puede decir certamente que el receptor se encuentra en algún punto sobre la esfera con radio de la distancia ( $d$ ) y siendo el centro el satélite. Añadiendo una segunda medición de un segundo satélite se puede acotar la ubicación a algún punto del perímetro de la circunferencia resultante de la

intersección de ambas esferas. Con un tercer satélite las intersecciones crean dos circunferencias que interceptan en dos puntos distintos. Teóricamente, con tres mediciones es suficiente para obtener la ubicación, dado que uno de los dos puntos da un valor absurdo. Por el contrario, es una posición dada en 2D únicamente. De modo que con una cuarta medición, se desecha uno de los dos puntos y al añadir la altitud se convierte en 3D. Además se usa para minimizar errores de sincronización entre relojes y todo ello conlleva que se obtenga finalmente una ubicación lo suficientemente precisa.

#### **1.4. Sistema de transferencia y almacenamiento de datos: Nube**

Anteriormente se ha expuesto el potencial de los terminales Android en recolectar Información Geográfica. Sin embargo, no es adecuado almacenar y manejar toda esa cantidad de datos en el mismo dispositivo, sino que lo optimo es volcar la IG a una nube, ya que se trata de un escenario de IoT y *Big Data*. Además, el servicio de nube en la red brinda una serie de posibilidades que permite expandir sin límites cualquier SIG, como hacer que el sistema sea multidispositivo, tratamiento de datos o acceso a la información en cualquier lugar y momento.

Por tanto, existen dos problemáticas a solucionar: como transferir dicha información y como almacenarla.

##### **1.4.1. Transferencia de IG a la nube**

En primer lugar, lo que se necesita es el enlace entre el dispositivo móvil y la nube. Lógicamente este enlace será INTERNET, la red universal. Para ello, hay que conectar el dispositivo Android a la red, bien mediante WIFI o por datos móviles. Posteriormente, al nivel de enlace hay que dotarle de nivel de red y de aplicación para poder transferir datos, en resumidas cuentas, hay que elegir los protocolos de comunicación. Esta decisión no es sencilla debido a la falta de estandarización en el

ámbito de las nubes (*cloud computing*). En la actualidad los fabricantes están desarrollando sus propias nubes, cada uno a su manera, lo que puede traer problemas de interconexión entre nubes. Por otro lado, la Computación en la Nube puede ser vista como una red de comunicaciones con servidores, clientes y servicios similares a los de cualquier otra red de comunicaciones basada en una Internet. Es así, que el intercambio de paquetes está regido por protocolos de comunicaciones típicos en un ambiente de computación distribuido tales como TCP, IP, SMTP y HTTP. De hecho, los protocolos típicos encontrados en cualquier nube son TCP y HTTP. [14]

Existen otras posibilidades como por ejemplo [15]:

- Web Socket: Esta tecnología proporciona un canal de comunicación cliente-servidor bidireccional y *full-duplex* sobre un único *socket* TCP. Este protocolo define el prefijo “ws://” o “wss://” (la versión *WebSocket Secure*). No obstante algunos *proxys* no permiten este tipo de protocolo haciendo que la conexión falle.
- *Message Queuing Telemetry Transport* (MQTT): Es un protocolo máquina a máquina (M2M) diseñado para el IoT, para clientes pequeños (sensores...). Funciona sobre TCP/IP. Esta pensado para redes de sensores con un servidor central que recolecta toda esa información, lejos de servir para transferencia de datos a la nube.
- *Advanced Message Queuing Protocol* (AMQP): Este es un protocolo punto a punto para el intercambio de mensajes de servidor a servidor. También funciona sobre TCP/IP y tamaño de paquetes reducido (60 Bytes).
- *Constrained Application Protocol* (CoAP): Está diseñado para trasladar el modelo HTTP a dispositivos y redes restrictivas. Sin embargo, realiza las interacciones las realiza de forma asíncrona sobre UDP, con paquetes mucho más pequeños.

-M2MXML: Protocolo de mensajería M2M. Los mensajes son pequeños documentos XML con atributos asociados y comandos definidos en el estándar.

Sin embargo, estas anteriores no son las más adecuadas para un servicio de nube en internet. La comunicación será de tipo cliente-servidor y no M2M ni servidor-servidor. Además, con el objetivo de que el servicio sea funcional en cualquier red, hay que tener en cuenta que muchos tipos de conexiones son bloqueados por los proxys y los cortafuegos por motivos de seguridad. Sin embargo, hay un tipo de tráfico que no se suele bloquear y que es el más abundante en internet: Las conexiones TCP/IP usando el puerto 80 designado para tráfico HTTP (o puerto 443 para HTTPS). Existen servicios web, es decir, conjuntos de protocolos y estándares para intercambiar datos entre aplicaciones mediante interfaces software. Los más destacados son REST y SOAP. Ambos son de tipo cliente-servidor, y funcionan sobre HTTP/TCP/IP (anunque SOAP puede ser utilizado sobre cualquier protocolo de transporte).

Los servicios SOAP (*Simple Object Access Protocol*) se basan en una serie de mensajes XML con una estructura detallada. Por otro lado, los servicios REST (*Representational State Transfer*) se basan solamente en el protocolo HTTP y sus métodos, convirtiéndose en un proceso más **simple y convencional** que SOAP. También ofrece una gran **escalabilidad**, por lo que se convierte en la mejor opción para desarrollar un *Cloud* preparado para soportar peticiones de múltiples sensores [15]:

#### 1.4.1.1. REST: REpresentational State Transfer

Es una arquitectura de servicios web que fue definida por Roy Fielding en su tesis del año 2000. Sirve para implementar aplicaciones web distribuidas. A diferencia de otros servicios web como SOAP o WSDL no usa tecnologías nuevas (no es nada nuevo: ni

un formato, ni un protocolo...), sino qué utilizando la arquitectura original de la web, se define un estilo de arquitectura.

Para su implementación se usa HTTP 1.1 (*HyperText Transfer Protocol*). Este protocolo es de tipo cliente/servidor usado en la web. Inicialmente diseñado para trasferir páginas HTML, pero sirve para cualquier contenido (texto plano, binario, XML, JSON...). Usar HTTP es muy ventajoso debido a que es el protocolo web más utilizado hoy en día y eso implica que funciona sin problemas en casi cualquier escenario.

Por otro lado, el protocolo HTTP dispone de distintos métodos con los que el cliente puede hacer distintas peticiones al servidor. El método dice que tipo de acción se quiere realizar con un recurso de la nube. Esta característica es la que utiliza REST para diferenciar las peticiones, y supone simplificar y facilitar el crearlas y gestionarlas. Estos son los métodos más utilizados:

- GET: Solicita una representación de un recurso especificado. No modifica nada en el recurso.
- POST: Enviar datos al recurso especificado.
- PUT: Carga en el servidor una representación de un recurso especificado.
- DELETE: Elimina el recurso especificado.

Si el mensaje llega al servidor, envía una respuesta HTTP. Estas respuestas contienen:

- Código de estado (200 OK, 201 Created, 404 Not Found...) [16]
- Cabeceras: Metainformación de la respuesta.
- Cuerpo del mensaje (opcional y distinto según a que petición responda):
  - GET: Representación del recurso pedido.
  - POST: Vacío o con el resultado del procesamiento.
  - PUT: Normalmente vacío.

- DELETE: Normalmente vacío.

Una aplicación REST está compuesta por recursos. Los recursos pueden ser una colección (una serie de recursos del mismo tipo) o individuales (un recurso concreto). El identificador global de cada recurso es la URL. Una petición consta de:

- Una URL: Que identifica al recurso sobre el que actuar.
- Un método de acceso (GET, POST, PUT...): Que especifica la acción a realizar sobre el recurso.
- Cabeceras: Metainformación de la petición.
- Cuerpo del mensaje (opcional). Solo con algunos métodos.

REST es una arquitectura sin estado, nunca se guarda de facto el estado en el servidor. Toda la información necesaria que se requiere para solicitar un recurso debe estar en el mensaje de la petición HTTP del cliente. Esto le da la ventaja de ser escalable, no tiene problema de almacenamiento de variables de sesión. Sin embargo, puede incluirse estados o sesiones en la cabecera del mensaje.

Por tanto, un Web Service de tipo REST es capaz de dar un buen servicio para transmitir la IG a la nube. Además, deja libertad para enviar el contenido en distintos formatos. En el siguiente apartado se estudia el formato más adecuado para enviarla:

#### **1.4.1.2. Formato de datos IG para su trasferencia**

Como anteriormente se ha mencionado, debido a la importancia de la estandarización para que los datos obtenidos puedan ser capturados o entendidos por otro sistema, se ha adoptado una estructura de datos en formato JSON para dicho intercambio. En concreto el formato GeoJSON, que define una serie de reglas y normas que regulan el formato del propio JSON para su uso con datos geoespaciales.

Existen otros formatos estándares que se plantean como solución al mismo problema, como es el caso del Lenguaje de Marcas Geográficas (GML). GML fue creado por la OGC y luego aprobado como norma internacional ISO del TC211 (ISO19136:2007). Por eso, la ventaja que tiene sobre los demás formatos es que se basa en un estándar abierto y muy aceptado, lo que supone que es compatible con una amplia variedad de aplicaciones. No obstante, es poco práctico para enviar y procesar por la sobrecarga de información que contiene, además del problema *Cross-Domain* [17].

Con el fin de evitar dicho problema y hacer más ligeros los ficheros, surgen los formatos GeoJSON y TopoJSON basados en JSON. Un objeto GeoJSON puede representar una geometría, un fenómeno o una colección de fenómenos. Soporta los siguientes tipos de geometría: *Point*, *LineString*, *Polygon*, *MultiPoint*, *MultiLineString*, *MultiPolygon* y *GeometryCollection*. Los fenómenos en GeoJSON contienen un objeto de geometría (georeferencia) y sus propiedades (variables como temperatura, conductividad...). En este caso, la IG son una serie de mediciones por puntos en un área determinada, por lo que se utilizará una colección geométrica de puntos. A continuación se muestra un ejemplo de una colección de dos muestras puntuales con sus propiedades climáticas:

```
{"type":"FeatureCollection",
 "features":[
 {"geometry": {
 "coordinates": [
 [-83.6823018,22.41341261,12.966529681],
 "type":"Point"
 },
 "properties":{}}
```

```

    "humidity":75,
    "temperature":29.1,
    "time":"16:10:19_16-01-2018"
},
"type":"Feature"

},

{
"geometry": {
"coordinates": [
-83.6846219,22.4143798,17.7319489864],
"type":"Point"
},
"properties":{
"humidity":77,
"temperature":28.4,
"time":"16:10:50_16-01-2018"
},
"type":"Feature"
}
]
}

```

Algunas de las ventajas de GeoJSON sobre GML son:

- No tiene problema de *Cross-domain*.
- Menor tamaño de fichero.

- Fácil de leer y escribir, para los seres humanos, y de analizar y generar, para las máquinas.

Por otro lado, TopoJSON es una extensión de GeoJSON que codifica la topología mediante un nuevo tipo, “*Topology*” , que contiene objetos GeoJSON. En lugar de representar geometrías discretas, las geometrías de los ficheros TopoJSON se definen a partir de segmentos de líneas compartidas denominadas arcos. Elimina redundancias ofreciendo representaciones mucho más compactas de la geometría que con GeoJSON. Además, cada arco está definido por sus coordenadas cuantificadas, por lo que logra un fichero de menor tamaño. No obstante, esto no siempre es así, dado que los objetos geométricos puntos y múltiples puntos están representados directamente con las coordenadas, como en GeoJSON, en lugar de arcos [17]. Es por eso que para guardar la información de multipuntos, no supone ninguna ventaja, al contrario, comparando los tamaños de los ficheros se observa que TopoJSON es mayor. En la siguiente imagen se aprecia la comparación entre los distintos formatos, siendo el contenido el mismo para todos: Dos puntos georeferenciados con varias propiedades (corresponde con el ejemplo GeoJSON mostrado arriba):



*Figura 6: Comparación de tamaños por formatos*

En cualquier caso, estos formatos han tenido un gran desarrollo en los últimos años, y al estar correctamente estandarizado no es problema convertir de un formato a otro sin perder información. De hecho existen múltiples herramientas que lo hacen *online* o mediante una librería.

Por último, destacar que para la visualización de capas en formato GML, GeoJSON y TopoJSON existen librerías y servicios como OpenLayers y MapBox (que grafican sobre OpenStreetMap).

#### **1.4.2. Almacenamiento de IG en la nube**

Para terminar de explicar todo el proceso, queda mencionar como se almacena la información que se envía mediante un servicio REST.

Se contemplan dos posibilidades de almacenamiento. Por un lado, guardar la IG en formato GeoJSON en ficheros texto normales con extensión .geojson. Por otro lado, volcar la información a una base de datos. La opción de guardar en fichero es la más simple y sencilla, pero guardar en base de datos brinda muchas más ventajas y posibilidades, sobre todo hablando de *Big Data* y *Data Mining*. Una base de datos es un conjunto de datos pertenecientes al mismo contexto y almacenados sistemáticamente para su posterior uso. Dada la importancia que tienen hoy en día existen múltiples soluciones.

Según la variabilidad se pueden clasificar en estáticas o **dinámicas**. Las estáticas son aquellas que se inserta la información en la base de datos para su posterior uso a modo de lectura. Las BB.DD. dinámicas, son aquellas que en cualquier momento se opera con ellas, añadiendo, modificando o eliminando información de las mismas. Para un sistema de adquisición de datos espaciales y su posterior volcado en nube, queda claro que debe de ser una base de datos dinámica, con la que se pueda añadir información a medida que tomen nuevas mediciones, o borrar las mediciones que no interesen.

También, existen distintos tipos según el modelo de administración de datos, es decir, según la lógica implementada para organizar los datos:

- Jerárquicas: Los datos se organizan en forma de árbol, siendo la raíz el primer nodo (sin padres), y la hoja el último (sin más hijos). Este tipo de base de datos es muy útil para grandes cantidades de datos y muy compartidos, pero ineficiente para representar datos redundantes.
- De red: Parecida a la anterior solo que se permite que los nodos tengan más de un parente. Esto es la solución eficiente al problema de la redundancia, pero sigue siendo muy compleja de manejar.
- Transaccionales: Su único fin es el envío y recepción de datos a grandes velocidades. Estas bases son muy poco comunes.
- **Relacionales**: Este es el modelo que más se usa hoy en día para administrar datos dinámicamente. Se basa en “relaciones” que en forma lógica forman “tuplas”. Una forma de interpretarlo es que cada relación es una tabla compuesta por registros (tuplas) y cada una compuesta por distintas columnas (campos). Incluir, consultar o borrar información se realiza mediante una serie de consultas que ofrecen muchas posibilidades. El lenguaje más extendido para hacer estas consultas es SQL (*Structured Query Language*), dado que es simple y sencillo de utilizar.
- Orientadas a objetos: En este caso se almacenan objetos con su estado y comportamiento. Además, se pueden definir operaciones sobre los datos. Las consultas se hacen con SQL:2003, una ampliación de SQL para soportar objetos. Es un modelo bastante reciente.

La base de datos implementada en este proyecto es relacional. Se ha utilizado este modelo por su versatilidad, facilidad, capacidad y extensión que tiene. Se puede clasificar la información, de manera que se almacene ordenadamente, accesible y de fácil lectura. Además, en un futuro, permite realizar otras funciones como búsqueda

por parámetros específicos. Por ejemplo para visualizar o extraer solamente las mediciones realizadas en un intervalo temporal o las que hayan medido cierta temperatura. Estas tareas se realizan de forma simple, fácil y rápida mediante una base de datos de este tipo. Sin embargo, estas tareas serían todo lo contrario en el caso de hacerlas extrayendo la información de ficheros de texto. Estas funcionalidades permiten que en un futuro se puedan implementar herramientas de análisis de datos y estadísticas, para obtener una información valiosa que permita sacar conclusiones y ayude en la toma de decisiones.

## CAPÍTULO II. DISEÑO E IMPLEMENTACIÓN

El objetivo de este proyecto es desarrollar un sistema de adquisición de datos espaciales con su posterior volcado a nube, con el objeto de sacarle rendimiento a la cantidad de sensores y funcionalidades que incorporan hoy en día los smartphones. El sistema debe servir para la agricultura a pequeña escala, así como para realizar estudios ambientales. El teléfono inteligente se encarga de la recolección de datos geográficos (geolocalización, temperatura y humedad). No obstante, hoy en día son pocos los teléfonos que incorporan sensores de temperatura y humedad, porque se utiliza un sensor externo que se conecta al terminal. Por otro lado, la información se almacena en una nube, aunque para la práctica se ha utilizado una nube simulada. En resumen, el sistema implementado tomará medidas de temperatura, humedad, latitud, longitud y altitud, y se guardarán en el ordenador, que estará simulando una nube.

En este capítulo se presenta el diseño del sistema así como su implementación práctica y validación en escenario real de práctica. Se detalla la aplicación realizada para la obtención de la información geográfica implementada en un sistema operativo android. Del mismo modo, se expone el emulador de nube de datos con API REST creado para recibir y almacenar la información enviada por el terminal android, para su posterior análisis.

- Terminal Android:

Es la central que recoge todos los datos para su posterior envío al servidor. Se encarga de obtener los datos de temperatura y humedad del sensor externo y de obtener la ubicación (latitud, longitud y altitud) mediante los satélites GPS. Posteriormente, envía dichos datos al servidor.

- Emulador de nube de datos:

Es la aplicación ejecutada en ordenador remoto enlazado mediante red inalámbrica IEEE 802.11 que recibe los datos obtenidos por el teléfono inteligente Android. Para recibir los datos, se ha implementado un interfaz API REST, con el que mediante consultas *http* se puede realizar de una forma sencilla.

## 2.1. Arquitectura del proyecto

En el siguiente esquema se puede apreciar todos los elementos físicos que intervienen en la implementación del proyecto (terminal android, sensor externo, satélites y emulador de nube de datos), así como los canales de comunicación que se establecen y relacionan a los mismos para el intercambio de información.

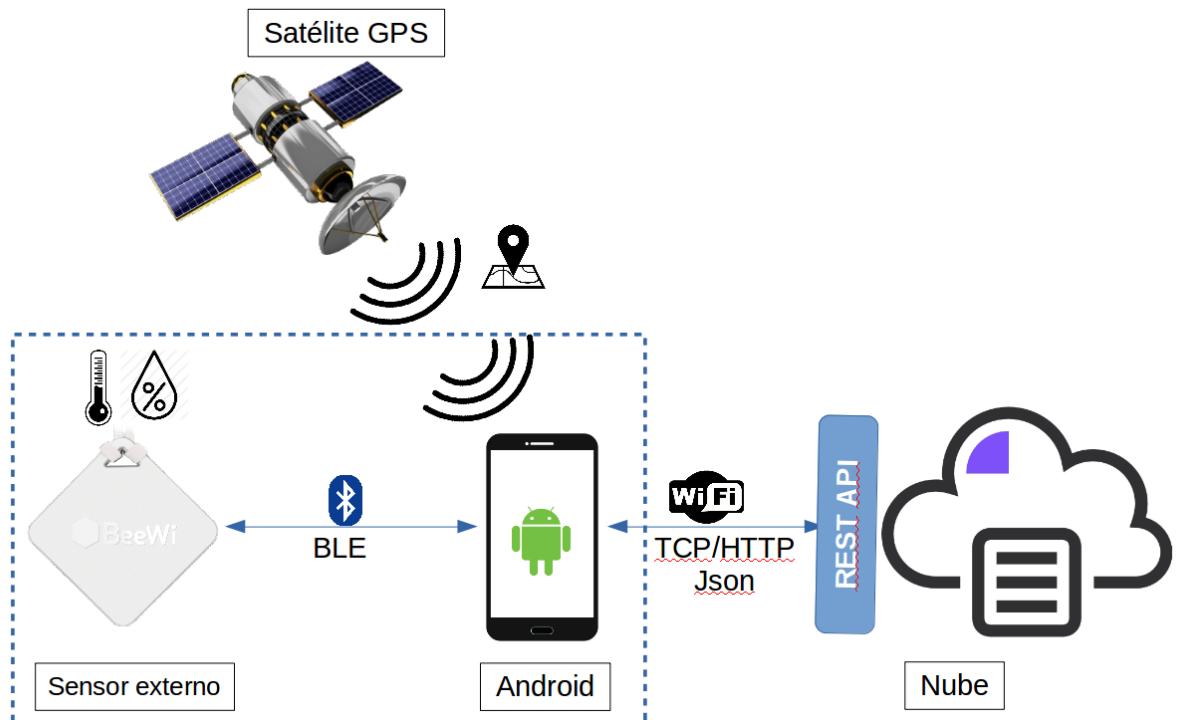


Figura 7: Esquema del sistema implementado

## **2.2. Manejo del terminal de adquisición y envío de datos espaciales (Android)**

En este apartado se expone la implementación de la aplicación para dispositivos Android. Esta *app* se encarga básicamente de la lectura de la información y su envío a la nube. Por un lado se obtiene la medición de temperatura y humedad del sensor externo BeeWi vía Bluetooth. Por otro lado se obtiene la ubicación mediante satélites GPS. Finalmente se envía esa información a la nube. La manera de operar es la siguiente: Cuando en el dispositivo tiene activado Bluetooth, ubicación GPS y WIFI, se puede iniciar la captura de datos. Al iniciarla se comienza pidiendo la ubicación, y cuando ya se logra, se le pide la información al sensor. Cuando ya toda la información está disponible se convierte a formato GeoJSON y se envía a la nube. Este proceso se repite con una frecuencia mínima variable y configurable (aunque depende de la rapidez con la que se reciba la ubicación del GPS). Mientras la captura esté activa seguirá dicho proceso.

### **2.2.1. Manejo de Bluetooth para la adquisición de datos del sensor externo**

La obtención de datos de temperatura y humedad se hace mediante una conexión inalámbrica (*Bluetooth*) entre el sensor externo BeeWi SmartClim y el dispositivo android. En este apartado se explica como se implementa para el caso concreto.

Es muy importante mencionar que para el manejo de BLE se usa una API totalmente nueva, diferenciándose así de la API del *bluetooth* clásico. La implementación del manejo del BLE en el sistema operativo Android, entró a partir de la versión 4.3 *Jelly Bean*, nivel de API 18. Es por eso que al crear el proyecto se debe seleccionar esa versión o superior. En este caso se decide usar la versión más baja posible (v4.3), para que sea lo más accesible, porque de este modo funcionará en cualquier dispositivo con dicha versión o superior.

Inicialmente, para usar cualquiera las funcionalidades de comunicación del *Bluetooth* en Android, se debe declarar el permiso de uso “*BLUETOOTH*”. Y si además se quiere manipular la configuración se debe declarar el permiso “*BLUETOOTH\_ADMIN*” [18]:

```
<uses-permission android:name="android.permission.BLUETOOTH"/>  
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

Para utilizar cualquier función del *Bluetooth* es necesario el objeto java predefinido llamado “*Bluetooth Adapter*”. Este objeto es la forma de interactuar con la propia antena *Bluetooth*. Dicho objeto se obtiene de la siguiente forma:

```
// Definir el objeto Bluetooth adapter.  
private BluetoothAdapter mBluetoothAdapter;  
...  
// Inicializar Bluetooth adapter.  
final BluetoothManager bluetoothManager =  
    (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);  
mBluetoothAdapter = bluetoothManager.getAdapter();
```

Antes de usarlo se debe comprobar si el *Bluetooth* está activado y si no lo está activarlo. Se puede hacer mediante una sentencia “*if*” que compruebe si no está activado, comprobando que el objeto “*BluetoothAdapter*” sea nulo o sino lo está llamando a su método “*isEnabled()*” :

```
if (mBluetoothAdapter == null || !mBluetoothAdapter.isEnabled()) {  
    // si no está activado pregunta al usuario si quiere activarlo.  
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
```

```
        startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);  
    }  
}
```

Para comenzar el escaneo BLE, el mismo objeto “*BluetoothAdapter*” contiene un método llamado “*startLeScan()*”. Este método coge como parámetro la función de devolución “*BluetoothAdapter.LeScanCallback*”, que como su nombre indica recibe los resultados del escaneo. Continuando, existen diversas formas de controlar el escaneo ante el dilema de cuando pararlo o iniciararlo para ahorrar energía. En el proyecto se ha adoptado la propuesta de *google* de la sección de desarrolladores de android [18], en la que se propone iniciar o parar el escáner al llamar al método “*scanLeDevice()*” y sino se finaliza antes, finalizarlo al pasar cierto periodo predefinido. Para ello hace falta un objeto “*Handler*” que permite ejecutar en otro hilo el final del escaneo tras el periodo establecido:

```
private BluetoothAdapter mBluetoothAdapter;  
private Handler mHandler;  
// Se define el periodo de escaneo en 10 segundos.  
private static final long SCAN_PERIOD = 10000;  
...  
private void scanLeDevice(final boolean enable) {  
    if (enable) {  
        // Terminar escaneo al alcanzar el periodo de escaneo.  
        mHandler.postDelayed(new Runnable() {  
            @Override  
            public void run() {  
                mBluetoothAdapter.stopLeScan(mLeScanCallback);  
            }  
        }, SCAN_PERIOD);  
        // Iniciar el escaneo si llaman al metodo con un “true”.  
        mBluetoothAdapter.startLeScan(mLeScanCallback);  
    }  
}
```

```

} else {
    // Parar el escaneo si llaman al método con un “false”.
    mBluetoothAdapter.stopLeScan(mLeScanCallback);
}
...
}

```

Por último, se expone como se reciben los resultados, es decir, como implementar el interfaz “*BluetoothAdapter.LeScanCallback*” antes mencionado. Funciona de la siguiente manera: cada vez que recibe la transmisión de un dispositivo ejecuta el código que el programador ponga. De cada transmisión está interfaz obtiene tres variables: un objeto “*BluetoothDevice*” (es el objeto que reúne la información del dispositivo que ha transmitido), un número entero (indicador de fuerza de la señal recibida) y un array de bytes (información que envía el dispositivo):

```

private BluetoothAdapter.LeScanCallback mLeScanCallback =
    new BluetoothAdapter.LeScanCallback() {
        @Override
        public void onLeScan(final BluetoothDevice device, int rssi, byte[] scanRecord)
{
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // Acciones a realizar al recibir transmisión.
        }
    });
}
};

```

Después de tener bajo control el manejo del BLE solo queda extraer la información de temperatura y humedad del sensor BeeWi. Dicha información viene en el *array* de bytes (*scanRecord*) que el sensor transmite cuando el terminal Android hace el escaneo. Este *array* está formado por 61 bytes, de los cuales solo 44 son útiles (concuerda con la teoría en la que un paquete BLE *advertising* contiene como máximo 47 bytes de los que 3 bytes son CRC). Un ejemplo puede ser el siguiente (representación del valor número entero de cada byte separados por espacio):

2	1	6	14	-1	13	0	5	0	-25	0	2	62	7	0	0	6	71	5	18	25	0	30	0	2	10	0	16	9	66	101	101						
87	105	32	83	109	97	114	116	67	108	105	109	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Los bytes que contienen la información de interés para el proyecto son:

- bytes nº 9 y 10: Temperatura
- byte nº 12: Humedad
- byte nº 17: Nivel de batería

Tanto la humedad como el nivel de batería vienen dados por el mismo valor de número entero que devuelven los bytes correspondientes (en el ejemplo superior sería 62% de humedad y 71% de batería). Sin embargo, para conocer el valor de la temperatura no es tan sencillo:

Hay que tener en cuenta que un byte puede llegar a dar como máximo 256 valores, que pueden ir del -127 al 128, y como la temperatura se da con un decimal (por ejemplo: para exponer 10,3°C se da el valor 103), no es posible cubrir todo el rango de temperaturas (-20°C a +85°C) con un único byte; solo es posible informar de una variabilidad de 25,6°C. Por esa razón son necesarios 2 bytes al menos. El sensor los usa de la siguiente forma:

Mediante el byte nº9 da la diferencia de temperatura sobre una referencia. La aporta en grados celsius (grados centígrados) y con la decimal incluida pero sin la coma (es

decir  $12,3^{\circ}\text{C}$  sería 123). Por otro lado el byte nº10 indica cual es la referencia. Aunque no solo dependen de este byte sino que también del signo del byte anterior (el diferencial). Dicha referencia es siempre un múltiplo de 256, como ya se ha explicado su razón, aunque el décimo byte no puede dar más de ese valor, y por ello solo se proporciona el multiplicador correspondiente. Las referencias se reparten de la siguiente manera:

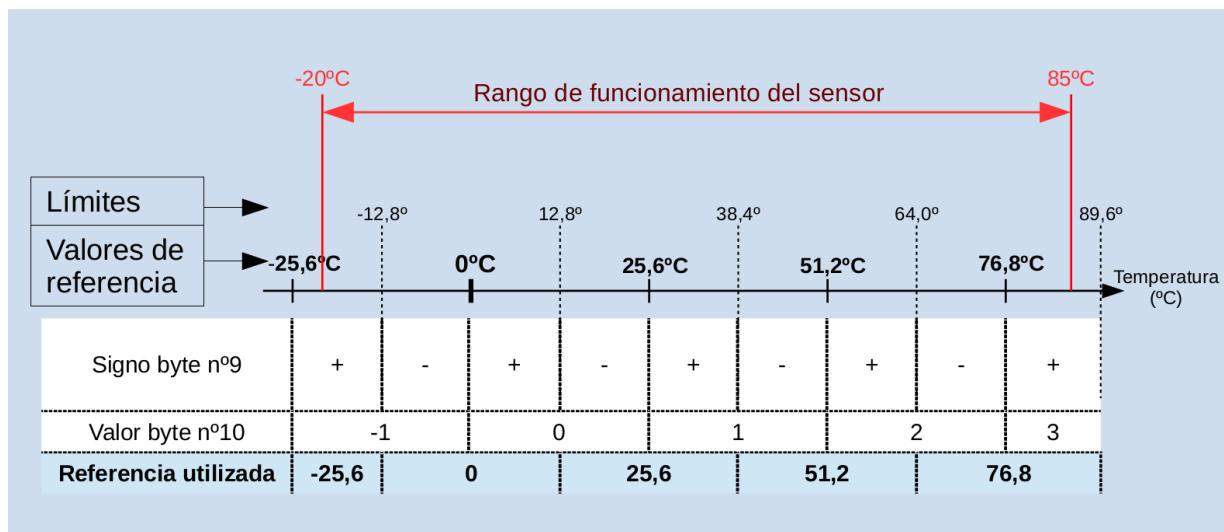


Figura 8: Referencias de temperatura según bytes (BeeWi SmartClim)

Por ejemplo: al tener en los bytes 9 y 10 los valores -35 y 0 respectivamente, significa que al ser negativo el noveno byte y 0 el valor del décimo, significa que la referencia es  $25,6^{\circ}\text{C}$ , a la que hay que aplicarle el diferencial (el byte noveno):

La temperatura será:  $25,6 + (-35/10) = 22,1^{\circ}\text{C}$

El siguiente código es el empleado en el proyecto para la resolución. Se hace mediante un objeto creado llamado “BytesConverter”. El objeto tiene un parámetro de entrada, que es el array de bytes. También tiene un método llamado “convertData()” con el que los bytes se convierten en información y devuelve un objeto de tipo “SensorData” que no es más que un objeto que almacena dicha información (temperatura, humedad y nivel de batería).

El objeto “BytesConverter” se usa de la siguiente manera:

```
BytesConverter bytesConverter = new BytesConverter(scanRecord);
SensorData sensorData = bytesConverter.convertData();
```

Código del objeto en cuestión:

```
public class BytesConverter {
    private byte[] bytearray;
    private SensorData sensorData;
    private float temperature;
    private byte batterylevel, humidity;
    // Constructor para crear el objeto a partir de un byte array.
    public BytesConverter (byte[] bytearray){
        this.bytearray=bytearray;
    }
    // Este método convierte los bytes en información
    // Devuelve un objeto SensorData donde se almacena la info.
    public SensorData convertData(){
        if ((bytearray!= null) && (bytearray.length > 0)) {
            batterylevel = (byte) Math.max(0, Math.min(100, bytearray[17]));
            humidity = (byte) Math.max(0, Math.min(99, (int) bytearray[12]));
            // Cálculo de temperatura.
            if ((int) bytearray[9]>=0){
```

```

temperature = (int) bytarray[9] + (int) bytarray[10] * 256;

}else{

    temperature = (int) bytarray[9] + ((int) bytarray[10]+1) * 256;

}

temperature= temperature/10;

// Introducir la información en el objeto SensorData mediante su constructor.

sensorData = new SensorData(batterylevel,humidity,temperature);

return sensorData;

}else{

    return null;

}

}

}

```

## 2.2.2. Manejo de GPS para la obtención de la georeferencia

Android permite implementar de una forma sencilla la obtención de la información que proporcionan los satélites GPS [19]. En este caso, solo interesa la ubicación, que está formada por latitud, longitud y altitud.

Lo primero que se debe hacer es declarar los permisos de uso del GPS. El permiso necesario es “ACCESS\_FINE\_LOCATION”, pero para versiones Android 5.0 o superiores es necesario también el permiso de uso del *hardware*:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
<!-- Solo necesario para Android 5.0 (API level 21) o superior. -->
<uses-feature android:name="android.hardware.location.gps" />
```

Para obtener los datos de localización simplemente hay que pedirle a la clase “*LocationManager*” que dé actualizaciones de ubicación mediante su método “*requestLocationUpdates*”. A este método hay que pasarle cuatro parámetros de entrada: El proveedor de la información (en este caso los satelitales GPS), el tiempo mínimo en recibir las actualizaciones (en mili-segundos), distancia mínima entre la penúltima y la última ubicación en la que mande actualizaciones (en este caso cero metros para que no se rija por este criterio) y el *listener* que atienda las actualizaciones del GPS [19]:

```
// Crear el objeto LocationManager:
LocationManager locationManager;
// Inicializar el objeto obteniendo el servicio de ubicación de Android:
locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Actualizaciones de la ubicación GPS cada segundo:
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
1000, 0, locationListener);
```

Se puede establecer como tiempo mínimo cero mili-segundos y como distancia mínima cero metros. Esto significará que el proveedor GPS dará la información tan rápido como pueda. Hay que aclarar que esto no significa que estrictamente vaya a cumplir esos criterios porque existen impedimentos (como el retardo y fallos en la transmisión) que no aseguran dichas exigencias. Por eso, se refiere a esos criterios como valores mínimos, y por tanto no aseguran nada, solo sirven a modo de indicación.

El *listener* hay que definirlo previamente. Contiene métodos asociados a los eventos que recibe del propio proveedor:

- *onLocationChanged(Location location)*:

El método es lanzado cuando se recibe una nueva actualización de la ubicación. Devuelve un objeto “*Location*”, donde se puede extraer toda esa nueva información. Para obtener latitud, longitud y altitud se deben usar los métodos de este objeto “*.getLatitude()*” , “*.getLongitude()*” y “*.getAltitude()*” respectivamente.

- *onStatusChanged(String provider, int status, Bundle extras)*:

Este es lanzado cuando el estado del proveedor cambia, como puede ser que su estado cambie a fuera de servicio, disponible o temporalmente deshabilitado.

- *onProviderEnabled(String provider)*:

Lanzado cuando se habilita el proveedor seleccionado.

- *onProviderDisabled(String provider)*:

Este al contrario, cuando se deshabilita.

En resumen, el *listener* puede crearse de la siguiente forma, aunque existen otras. Esta es la más adecuada para el proyecto porque se puede implementar dentro de la misma clase principal:

```
LocationListener locationListener = new LocationListener() {  
    public void onLocationChanged(Location location) {  
        // Llamada cuando el proveedor envía la actualización de localización.  
    }  
};
```

```

// Crear las variables de interés.
double mlat,mlong,malt;
mlat = location.getLatitude();
mlong = location.getLongitude();
malt = location.getAltitude();

}

public void onStatusChanged(String provider, int status, Bundle extras) {}

public void onProviderEnabled(String provider) {}

public void onProviderDisabled(String provider) {}

};

```

Para terminar, solo falta detener el proceso de actualizaciones de ubicación GPS. Simplemente, hay que llamar al método “`.removeUpdates()`” del objeto “`LocationManager`” que anteriormente se ha creado y se ha iniciado:

```

// Dejar de actualizar.
locationManager.removeUpdates();

```

### **2.2.3. Manejo de WIFI y envío de datos al emulador de Nube de almacenamiento de datos**

En los apartados anteriores se explica como obtener los datos. Una vez obtenidos, solo falta enviarlos al servidor. El envío se hace vía WIFI y por eso es necesario manejarlo de forma adecuada. Esta vez para utilizarlo únicamente hace falta declarar el permiso de uso del *internet* [20]:

```

<uses-permission android:name="android.permission.INTERNET" />

```

Los datos se envían en formato JSON ya que es una forma estructurada y liviana de organizar la información (como se ha explicado anteriormente). Existe una librería de Google que permite trabajar fácilmente con este formato y se ha utilizado para ello. La librería se llama “GSON” y previo a su uso se debe incluir en el proyecto Android. Para incluirla únicamente hace falta añadir una línea en el *build.gradle* del módulo *app*:

```
dependencies {
    compile 'com.google.code.gson:gson:2.2.4'
}
```

Es muy sencillo de utilizar. Sirve para convertir un objeto a JSON o de un JSON a objeto. Los objetos se crean mediante clases modelos, en las que simplemente se definen una serie de variables. En la conversión un objeto se convierte en un nivel de JSON, en el que se incluyen los nombres de las variables y sus valores. Se pueden hacer tantos niveles como se quiera, introduciendo objetos dentro de objetos. La clase modelo utilizada para crear el objeto con toda la información y además con la estructura propia de un GeoJSON es la siguiente:

```
public class AllDataGeoJson {
    // Crear sub-objetos.
    private String type="Feature";
    private Properties properties;
    private Geometry geometry;
    // Constructor.
    public AllDataGeoJson(){}
    public class Properties{
        // Declaración de variables.
        private String time;
```

```

private byte humidity;
private f temperature;
private double altitude;
private String tag;

// Constructor.

public Properties(String time, byte humidity, float temperature, double
altitude, String tag){

    this.time = time;
    this.humidity = humidity;
    this.temperature = temperature;
    this.altitude = altitude;
    this.tag = tag;
}

public class Geometry{

    // Declaración de variables.

    private String type="Point";
    private double[] coordinates=new double[3];
    // Constructor.

    private Geometry(double longitude, double latitude, double altitude){

        this.coordinates[0]=longitude;
        this.coordinates[1]=latitude;
        this.coordinates[2]=altitude;

    }

}

// Métodos para introducir la información del objeto.

```

```

    public void setProperties(String time, byte humidity, float temperature,
double altitude, String tag){

    this.properties= new Properties(time, humidity, temperature, altitude,
tag);

}

public void setGeometry(double longitude, double latitude, double altitude){

    this.geometry=new Geometry(longitude, latitude, altitude);

}

}

```

Una vez que se declare el objeto y se le introduzcan los datos, se puede convertir a JSON de la siguiente forma:

```

// Declaración del objeto.

private AllDataGeoJson allDataGeoJson = null;

// Crear el objeto.

allDataGeoJson = new AllDataGeoJson();

// Introducir información con sus métodos.

allDataGeoJson.setGeometry(mlong,mlat);

allDataGeoJson.setProperties(fechaHoraActual(time),sensorData.getHumidity(),s
ensorData.getTemperature(),malt,tag);

// Crear el JSON con la librería.

if (allDataGeoJson != null) {

    Gson gson = new Gson();

    String geojson = gson.toJson(allDataGeoJson);

}

```

También se puede realizar el proceso inverso, convertir un JSON en objeto, pero para ello hace falta tener la clase modelo adecuada al JSON:

```
AllDataGeoJson allDataGeoJson = gson.fromJson(geojson, AllDataGeoJson.class);
```

Una vez con la información formateada y lista, queda enviarla al emulador de nube de datos. El envío se hace a través de los protocolos TCP/IP, siendo un paquete HTTP. Para ello se hace uso de otra librería llamada Android Async Http (obsérvese que la programación del envío es únicamente a nivel de aplicación, sin manipular directamente los niveles físico, enlace y red). Esta librería ayuda a hacer peticiones REST. Para añadir la librería al igual que la anterior:

```
dependencies {  
    compile 'com.loopj.android:android-async-http:1.4.9'  
}
```

Con la librería incorporada se puede proceder ha hacer las peticiones REST asíncronas. En el momento en que estén todos los datos (temperatura, humedad y ubicación), se envían en formato JSON con el método PUT de la API REST. Al cliente de la librería, para dicho tipo de petición, hay que pasarle cuatro parámetros de entrada: El contexto, la URL destino (es el servidor con la URI de dicha solicitud de la API), la información a enviar como entidad (el GeoJSON), el tipo de dato enviado (JSON) y el *listener* que atenderá la respuesta del servidor [20]:

```
// Crear el cliente  
AsyncHttpClient client = new AsyncHttpClient();  
  
// Enviar solicitud HTTP PUT con json  
client.put(this,
```

```

"http://" + ipcloud + "/tesis/index.php/api/v1/update/" + measurementid,
    new StringEntity(geojson), "application/json",
    new CaptureResponseHandler());

```

Hay que señalar que tal y como se puede apreciar en el código previo, la URL está compuesta por dos variables de texto: “*ipcloud*” y “*measurementid*”. Esto es así porque la aplicación permite configurar la ip del servidor, así como el nombre o identificador de la captura.

Por último, se definir el *listener* que atenderá la respuesta del servidor a la solicitud. Ha de mencionarse, que entrará en funcionamiento de forma asíncrona, cuando el servidor responda. Se hace mediante la creación de una clase extendida de la clase “*AsyncHttpResponseHandler*”, ésta última incluida con la librería. La clase implementa los métodos “*onSuccess*” y “*onFailure*”, que serán llamados cuando la respuesta sea válida o no lo sea, respectivamente [20]:

```

public class CaptureResponseHandler extends AsyncHttpResponseHandler {

    public CaptureResponseHandler(){

    }

    @Override

    public void onSuccess(int statusCode, Header[] headers, byte[] responseBody)
    {
        // Acciones a realizar cuando la respuesta es favorable(estado HTTP 200):
        // Se recoge el mensaje enviado por el servidor:
        String msg = new String(responseBody, "UTF-8");
    }
}

```

```

@Override

    public void onFailure(int statusCode, Header[] headers, byte[] responseBody,
Throwable error) {

    // Acciones a realizar cuando la respuesta No es favorable

    // (estado HTTP 4xx):

    // Se recoge el mensaje enviado por el servidor:

    String msg = new String(responseBody, "UTF-8");
}

}

```

### **2.3. Implementación nube con REST API**

Seguidamente se explica como se implementado el servicio de nube a nivel de emulador para no simplificar las pruebas sin tener que utilizar internet de forma directa.

La nube consta de un servidor con un interfaz REST API mediante la cual el cliente (el terminal android) interactúa con la nube. Esta REST API se encarga de tramitar las peticiones realizadas por el cliente. Además, mediante esas peticiones del cliente la nube gestiona la base de datos, donde se almacenan los datos enviados por el teléfono.

Para la creación de la nube simulada se ha usado el servidor XAMPP, que con su instalación incluye todo lo necesario para la nube del proyecto: Apache, MariaDB, phpMyAdmin, PHP... Es un servidor multiplataforma y se ha instalado y verificado en una computadora con Ubuntu 16.04 LTS.

### 2.3.1. Base de datos

La base de datos del sistema, no es más que el software que se encarga de almacenar toda la información recibida por el teléfono. El sistema de gestión de base de datos utilizada en el proyecto es MariaDB [21], que es una base de datos relacional con licencia GPL (General Public License). Es una derivación de MySQL, de hecho, ambas creadas por la misma persona. Esta base de datos también utiliza SQL como lenguaje para manejarla y se usa para realizar las peticiones a la BD.

La base de datos creada, contiene inicialmente una tabla llamada “MEASUREMENT\_FILES”, en la cual se registra cada nueva medición. La tabla contiene tres columnas, que son *id*, *timestamp* y *filename*. Por tanto se guarda el id de la medición, el tiempo en el momento en el que se creo el registro y el nombre del fichero GeoJSON asociado a la medición. Esta tabla nos sirve para tener bajo control las mediciones que hay y se van creando. Es importante tener en cuenta que la *id* de la medición es de valor único, no puede haber ni dos ni más que un solo registro con una misma *id*. Si se usa la misma *id*, los datos se guardarán en dicha medición, no se crea otra.

Por tanto con cada medición nueva (con nuevo *id*), se añade un registro en dicha tabla. También, se crea una tabla nueva con el nombre de la *id*. En esta tabla es donde se guarda la información de las mediciones tomadas por el *smartphone*. Cada tabla tiene cada dato en una columna distinta: *time* (instante de la medición puntual), *tag* (es un campo de texto donde se guarda el identificador del sensor que ha tomado los datos), *longitude* (longitud), *latitude* (latitud), *altitude* (altitud), *humidity* (humedad) y *temperature* (temperatura).

Tabla 2: Ejemplo tabla de una medición de la base de datos

time	tag	longitude	latitude	altitude	humidity	temperature
16:10:19_16-01-2018	D0:5F:B8:53:6B:13	-83.6823018	22.41341261	19	75	29.1
16:10:24_16-01-2018	D0:5F:B8:53:6B:13	-83.68230074	22.41341403	20	77	28.5

En el proyecto, las peticiones a la base de datos se realizan desde la API REST programada en PHP, y por eso las peticiones (*query*) SQL se hacen a través de funciones implementadas por el propio PHP. Para realizar cualquier tipo de petición SQL, debemos conectarnos a la base de datos previamente. La conexión la hacemos mediante el fichero “*conectionbd.php*” con código PHP :

```
<?php

$dbhost="127.0.0.1";
$dbuser="root";
$dbpass="";
$dbname="Meassurements_db";

// Conexión:
$db = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);

// Si falla la conexión:
if (!$db) {
    echo "Error: " . mysqli_connect_error();
    exit();
}

?>
```

Las ventajas de realizar la conexión en un fichero aparte son que evitamos la redundacia de código, ya que, si se utiliza en distintos ficheros solamente debemos incluirlo. Además, si cambiamos la configuración o la contraseña de acceso a la base de datos solamente debemos modificar ese fichero.

Para incluirlo solamente debemos poner la siguiente línea en el fichero que necesitemos:

```
include ('conectionbd.php');
```

Posteriormente, para realizar acciones en la base de datos utilizaremos la siguiente función PHP:

```
$query = mysqli_query(<base de datos>, "<código SQL>");
```

Un ejemplo de petición (inserción de registro en tabla):

```
//Insertar valores del JSON en tabla  
  
$query3 = mysqli_query($db,"INSERT INTO `mid` (`time`, `tag`, `longitude`,  
`latitude`, `altitude`, `humidity`, `temperature`)  
VALUES ('$time', '$tag', '$long', '$lat', '$alt', '$hum', '$temp');");
```

### 2.3.2. Servicio Web REST

Para crear el REST *web service* se ha usado un *framework* para PHP llamado Slim Framework, que es muy simple y su función es facilitar la creación de la API REST en php.

Para instalar el *framework* es necesario ejecutar el siguiente comando en la carpeta del servidor donde se quiere usar:

```
composer require slim/slim "^3.0"
```

\* Es necesario tener instalado php y composer.

Una vez instalado se crean los ficheros necesarios para el uso del framework. Para utilizarlo en cualquier PHP únicamente se debe poner el siguiente código al principio del fichero:

```
<?php  
use \Psr\Http\Message\ServerRequestInterface as Request;  
use \Psr\Http\Message\ResponseInterface as Response;  
require_once 'vendor/autoload.php';  
  
$app = new Slim\App;
```

Y al final del fichero, para que corra el programa:

```
$app->run();  
?>
```

Una vez implementado el *framework* se puede programar usándolo o bien programar en PHP normal. Es por eso que no tiene ningún inconveniente usarlo, sino todo lo contrario, facilita la programación de la API REST sin sacrificar nada.

En este trabajo se ha creado una API REST con las siguientes peticiones posibles:

Tabla 3: Peticiones API REST

Acción	Método	URL
Obtener JSON con lista de todas las mediciones	GET	http://localhost/tesis/index.php/api/v1/meassurements
Obtener una medición en GeoJSON	GET	http://localhost/tesis/index.php/api/v1/meassurements/{id}
Subir una medición a nube	POST	http://localhost/tesis/index.php/api/v1/uploadgeojson
Actualizar, crear, modificar o añadir puntos a la medición	PUT	http://localhost/tesis/index.php/api/v1/update/{id}
Borrar cierta medición	DELETE	http://localhost/tesis/index.php/api/v1/delete/{id}

En la tabla, en la columna “URL” los campos que aparecen entre corchetes “{}”, son campos variables, que en estos casos todos son “{id}”, y se refieren al identificador de la medición en cuestión. Como se ha explicado en el capítulo uno, esa es la forma en la que identifica al recurso en concreto, mediante la URL de la petición.

La petición que usa la aplicación Android para subir la IG a la nube es “.../api/v1/update/{id}”, con el método PUT. Las demás peticiones sirven para hacer peticiones básicas para manejar la base de datos, y podríamos hacer dichas peticiones con cualquier cliente REST (como por ejemplo el cliente *open source* y multiplataforma “Insomnia”).

Para crear y manejar una petición GET lo único que se debe poner es lo siguiente:

```
$app->get('<URI>', <función que atienda la petición>);
```

Del mismo modo para las peticiones POST, PUT y DELETE, cambiando \$app->get por \$app->post , \$app->put o \$app->delete.

Para la realización de la estructura de peticiones que anteriormente se han descrito se usa el siguiente código dentro del fichero principal “index.php”:

```
// API group

$app->group('/api', function () use ($app) {

    // Version group

    $app->group('/v1', function () use ($app) {

        $app->get('/measurements', 'getMeasurements');

        $app->get('/measurements/{id}', 'getMeasurement');

        $app->post('/uploadgeojson/{id}', 'addMeasurement');

        $app->put('/update/{id}', 'updateMeasurement');

        $app->delete('/delete/{id}', 'deleteMeasurement');

    });

});

});
```

Se puede apreciar que hay cinco tipos de peticiones dentro de dos grupos (API y Versión). Cada petición se maneja con una función externa. La estructura de las funciones que atienden las peticiones son así:

```
function updateMeasurement (Request $request, Response $response) {

    // Obtener atributo id:

    $mid = $request->getAttribute('id');

    // Obtener geojson recibido:

    $mjson = $request->getBody();
```

```
...
$response->getBody()->write("Updated");
// Código de respuesta: 200
$response->withStatus(200);
return $response;
}
```

Donde se tienen dos parámetros respectivos a la petición, que son los parámetros recibidos por la petición HTTP que hace el cliente (variable `$request`) y la respuesta a la petición (variable `$response`). Dentro de la variable `$request` se encuentran los atributos recibidos por un GET, así como la información mandada en un POST o en un PUT. Finalmente en la variable `$response` se introduce la respuesta a la petición según el estado.

## CAPÍTULO III. RESULTADOS OBTENIDOS

En este capítulo se muestran los resultados obtenidos en la realización de esta tesis, poniendo en práctica sobre el terreno el sistema desarrollado. Se muestra los estudios realizados, y en vista de los resultados obtenidos se concluye que el sistema de adquisición de datos espaciales con volcado en nube cumple con los objetivos definidos al inicio de la investigación. Se definen tres casos de estudio, en los que se pone a prueba el sistema en tres contextos distintos.

### **3.1. Caso de estudio 1: Delimitación de terreno para sembrado.**

En este primer caso se pone en práctica el sistema para comprobar que su uso sirve para realizar mapas de delimitación de terrenos, o separación de terrenos. Para ello se tomó una captura de datos con el teléfono móvil Android, y se dió una vuelta a la pista del docente de la UPR en Pinar del Rio. Posteriormente, se introduce el archivo GeoJSON con todas las medidas tomadas en el servicio web “geojson.io” [22], donde se logra el siguiente gráfico. Este servicio nos permite crear, modificar y graficar en 2D código GeoJSON. Para el graficado utiliza el servicio de MapBox, que a su vez utiliza OpenStreetMap:



Figura 9: Gráfico 2D de la captura de datos del caso 1

### 3.2. Caso de estudio 2: Superficie con pendiente.

En este segundo caso se ha tomado una captura de una superficie con pendiente, recorriendola con la aplicación en ejecución. Después de tomar los datos, el caso a estudiar es el desnivel de la superfcie, y la capacidad de hacer gráficas o modelos aproximados al real partiendo de la información geográfica obtenida por el sistema.

Por eso, se ha hecho uso de una nueva herramienta llamada MATLAB para realizar dichas gráficas. MATLAB es un software multiplataforma con entorno de desarrollo integrado y con lenguaje de programación propio, llamado “M”. Para crear las gráficas se ha hecho un script con extensión “.m”. El primer paso es cargar los datos en una matriz. Posteriormente con la función *linspace* se crean los vectores con los que luego con la función *meshgrid* se genera la cuadricula o mallado, para dibujar una gráfica continua en el espacio (dado que solo tenemos puntos en el espacio). Una vez que se tiene una cuadricula en 2D, se le da la forma usando los datos capturados utilizando la función *griddata*. Finalmente, se grafica mediante la función *surf*, dando como resultado una gráfica 3D.

```
dx=linspace(minx,maxx,N);
dy=linspace(miny,maxy,N);
[qx,qy]=meshgrid(dx,dy);
qz=griddata(x,y,z,qx,qy,'natural');
surf(dx,dy,qz),colorbar
```

En la siguiente imagen se puede observar una imagen real de la superficie sobre la que se ha tomado la captura:



Figura 10: Superficie real del caso de estudio 2

El resultado tomando los datos con el sistema implementado y tras graficarlo con MATLAB es el siguiente:

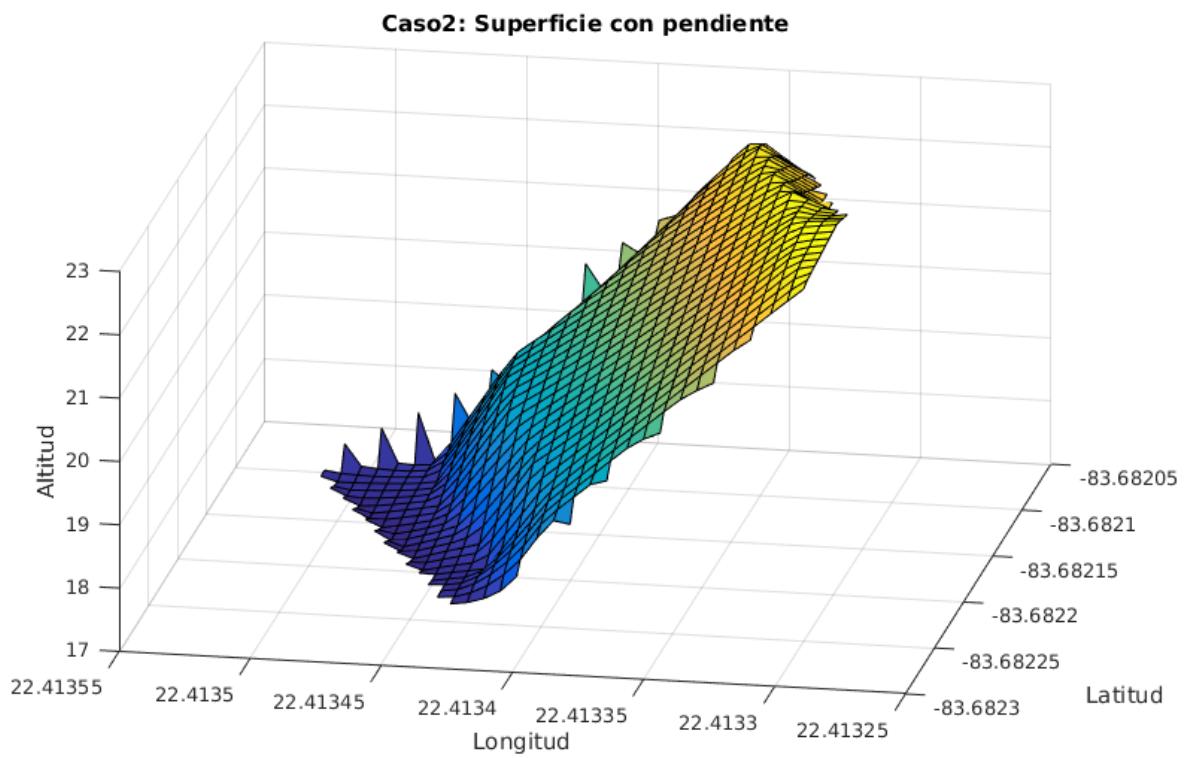


Figura 11: Gráfica de superficie con pendiente (caso 2)

### 3.3. Caso de estudio 3: Superficie con sol y sombra.

Por último, se ha estudiado una superficie donde se encontraba una zona expuesta al sol y otra que no recibía rayos de sol directamente. En este estudio se evalúa la validez del sistema para tomar mediciones de temperatura y humedad. Para ello, después de realizar la captura de la información sobre el terreno se ha graficado. En estas gráficas se ha añadido un cuarto parámetro a las coordenadas (x,y,z), que es la humedad o la temperatura. El cuarto parámetro está representado en la gráfica mediante el color. Para hacerlo se ha añadido otro parámetro a la función de gráfico *surf* que es otro malla solo que moldeado con la información del cuarto parámetro:

```
qv=griddata(x,y,temp,qx,qy,'natural');  
surf(dx,dy,qz,qv);
```

A continuación, se muestra la superficie real en el momento de la captura, seguido las gráficas obtenidas de temperatura y humedad:



Figura 12: Superficie real del caso de estudio 3

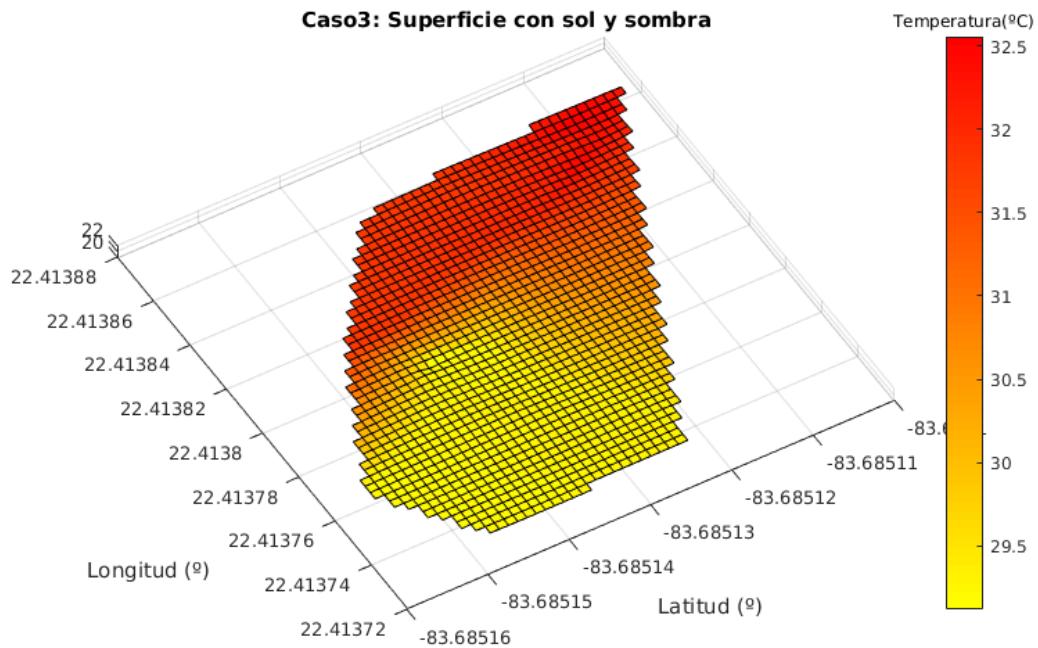


Figura 13: Gráfica de temperatura (caso 3)

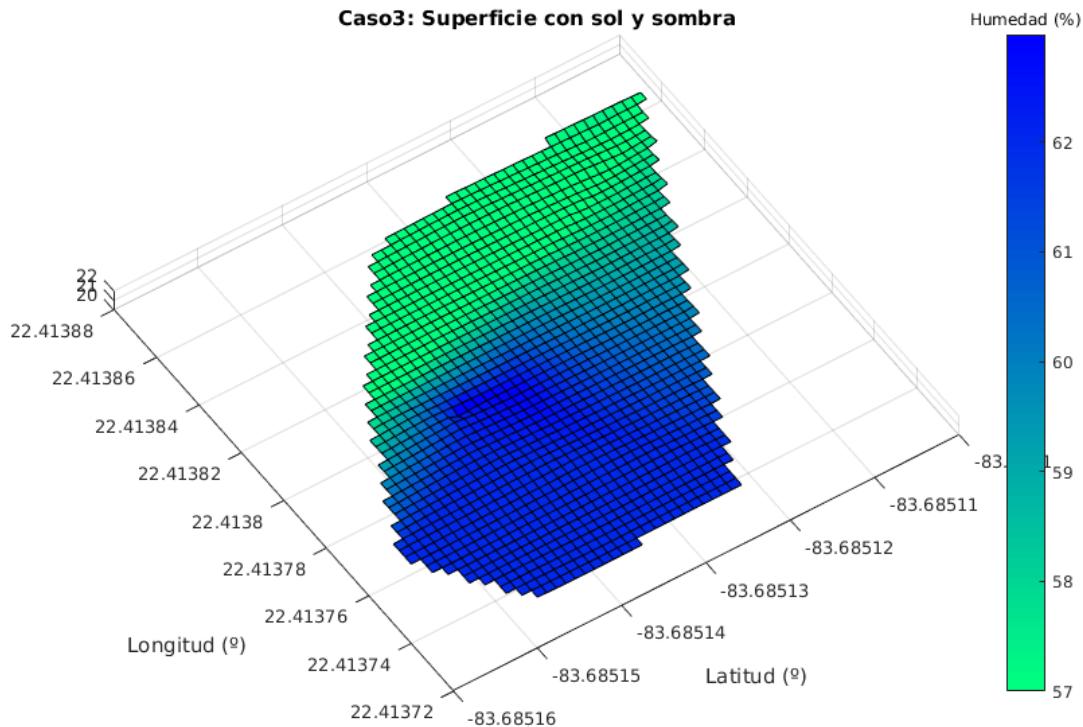


Figura 14: Gráfica de humedad (caso 3)

## **CONCLUSIONES**

1. Se logró el diseño y la implementación de un sistema de adquisición de datos espaciales.
2. Se logró el diseño y la implementación de una nube de datos simulada.
3. Se creó una aplicación en Android para capturar los datos y en PHP para la recepción y almacenamiento de datos (usando SQL).
4. La aplicación sirve para Nube real, en caso de tener un servidor con IP pública y en caso de tener acceso a internet en el terreno de captura.
5. Los resultados obtenidos en las pruebas realizadas demuestran el buen funcionamiento del sistema, aunque la precisión depende del receptor GPS del teléfono móvil.
6. El trabajo realizado, además de aplicarlo en la Agricultura de Precisión, su uso puede extenderse a otras aplicaciones, como el trazado de perfiles de rutas o rastreo.

## **RECOMENDACIONES**

- En Android, las clases startLeScan y StopLeScan para el escaneo BLE están en desuso desde la API 21, y existen otras clases que hacen la misma función y otras, de forma más óptima. Habría que aplicarlo para APIs 21 o superiores, y dejar startLeScan y StopLeScan para las inferiores.
- Añadir corrección de altitud, dado que el GPS nos da la altitud según el modelo WGS-84 (elipsoide de referencia), que no es exactamente lo mismo que nivel sobre el mar.

## **LISTA DE SIGLAS Y ACRÓNIMOS**

ADT	<i>Android Developer Tools</i>
AgGIS	<i>Agriculture Geographic Information System</i>
AP	Agricultura de Precisión
API	<i>Application Programation Interface</i>
BB.DD.	Bases de Datos
BLE	<i>Bluetooth Low Energy</i>
CRC	Código de Redundancia Cíclica o <i>Cyclic Redundancy Check</i>
CSW	Servicio de Catálogo
DM	<i>Data Mining</i>
GIS	<i>Geographic Information System</i>
GLONASS	<i>Global'naya Navigatsionnaya Sputnikovaya Sistema</i>
GML	<i>Geographic Makup Language</i>
GNSS	<i>Global Navigation Satellite System</i>
GPS	<i>Global Positioning System</i>
GSDI	<i>Global Spatial Data Infraestructure</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>HyperText Transfer Protocol</i>
IA	Inteligencia Artificial
IDE	Infraestructura de Datos espaciales

IDEE	Infraestructura de Datos Espaciales de España
IDERC	IDE de la República de Cuba
IG	Información Geográfica
INSPIRE	<i>Infraestructure for Spatial Information in Europe</i>
IoT	<i>Internet Of Things</i>
IPGH	Instituto Panamericano de Geografía e Historia
ISO	<i>Internacional Organization for Standardization</i>
JSON	<i>JavaScript Object Notation</i>
M2M	<i>Machine-to-Machine</i>
NSDI	<i>National Spatial Data Infrastructure</i>
OGC	<i>Open Geospatial Consortium</i>
PC-IDEA	Comité Permanente para las IDE de las Américas
PDU	<i>Packet Data Unit</i>
REST	<i>REpresentational State Transfer</i>
RS	<i>Remote Sensors</i>
SIG	Sistemas de Información Geográfica
SOAP	<i>Simple Object Access Protocol</i>
SQL	<i>Structured Query Language</i>
TIG	Tecnologías de Información Geográfica
UAV	<i>Unmanned Aerial Vehicle</i>

VANT	Vehículo Aéreo No Tripulado
VRT	<i>Variable Rate Technology</i>
WCS	Servicio de Coberturas en Web
WFS	Servicio de Fenómenos en la Web
WMS	Servicio de Mapas en Web
WSDL	<i>Web Services Description Language</i>
XML	<i>Extensible Markup Language</i>

## ÍNDICE DE FIGURAS

Figura 1: Ciclo de Aplicación de tecnología en la AP [2].....	8
Figura 2: Arquitectura del sistema Android [6].....	21
Figura 3: Distribución de canales BLE en el espectro [13].....	30
Figura 4: Paquete Advertising BLE [13].....	31
Figura 5: Principio de triangulación.....	33
Figura 6: Comparación de tamaños por formatos.....	41
Figura 7: Esquema del sistema implementado.....	46
Figura 8: Referencias de temperatura según bytes (BeeWi SmartClim).....	52
Figura 9: Gráfico 2D de la captura de datos del caso 1.....	72
Figura 10: Superficie real del caso de estudio 2.....	74
Figura 11: Gráfica de superficie con pendiente (caso 2).....	74
Figura 12: Superficie real del caso de estudio 3.....	75
Figura 13: Gráfica de temperatura (caso 3).....	76
Figura 14: Gráfica de humedad (caso 3).....	76

## ÍNDICE DE TABLAS

TABLA 1: COMPARACIÓN BLE Y <i>BLUETOOTH CLÁSICO</i> [13].....	29
TABLA 2: EJEMPLO TABLA DE UNA MEDICIÓN DE LA BASE DE DATOS.....	64
TABLA 3: PETICIONES API REST.....	67

## REFERENCIAS BIBLIOGRÁFICAS

- [1] PROCISUR, Manual de Agricultura de Precisión, Montevideo, 2014.
- [2] iQonsulting Ltda. con la colaboración de Stanley Best (Ing. Agr. M.Sc. Dr., ProgapINIA) e Inés Zamora (Ing. Agr. M.Sc.), Tecnologías Aplicables en Agricultura de Precisión, Santiago de Chile, 2008.
- [3] Javier Valencia Martínez de Antoñana, PASADO, PRESENTE Y FUTURO DE LAS INFRAESTRUCTURAS DE DATOS ESPACIALES, Trabajo Fin de Máster, Valencia, 2011.
- [4] Open Geospatial Consortium. OGC. Disponible en: <<http://www.opengeospatial.org>>. Consultado en: 26 de Octubre de 2017.
- [5] IDERC. Empresa Cartografía y Soluciones Geomáticas GeoSí y Grupo Empresarial GEOCUBA. Disponible en: <<http://www.iderc.cu>>. Consultado en: 19 de diciembre de 2017.
- [6] wikipedia. Fundación Wikimedia, Inc.. Disponible en: <<https://es.wikipedia.org/wiki/Android>>. Consultado en: 3 de enero de 2018.
- [7] Axoi Bitarra. Zaharberriku mugikorra software librearekin, 13 de agosto de 2017. Disponible en: <<https://axoibitarra.wordpress.com>>. Consultado en: 20 de enero de 2018.
- [8] David Pérez, El androide libre. ¿Cuánto de Android es libre?. Disponible en: <<https://elandroidelibre.elespanol.com/2016/11/android-es-libre.html>>. Consultado en: 20 de enero de 2018.
- [9] P. Palencia Herranz, SISTEMA DE GESTIÓN DE PERFILES POR UBICACIÓN PARA TELÉFONOS ANDROID, Leganés, 2015.
- [10] Andrea Ardións. ¿Cuál es la estructura de un proyecto en Android Studio? AndroidStudioFaqs te lo cuenta. Disponible en:

<<https://androidstudiofaqs.com/conceptos/cual-es-la-estructura-de-un-proyecto-en-android-studio>>. Consultado en: 20 de enero de 2018.

- [11] Sgoliver . SGOLIVER.NET. Disponible en: <<http://www.sgoliver.net/blog/componentes-de-una-aplicacion-android/>>. Consultado en: 20 de enero de 2018.
- [12] EcuRed. ecured@idict.cu. Disponible en: <<https://www.ecured.cu/Bluetooth>>. Consultado en: 8 de noviembre de 2017.
- [13] Argenox. info@argenox.com. Disponible en: <<http://www.argenox.com/a-ble-advertising-primer/>>. Consultado en: 13 de Noviembre de 2017.
- [14] Jhon Jairo Padilla Aguilar y Javier Pinzón Castellanos, ESTÁNDARES PARA CLOUD COMPUTING: ESTADO DEL ARTE Y ANÁLISIS DE PROTOCOLOS PARA VARIAS NUBES, Universidad Pontificada Bolivariana, 9 de mayo de 2015.
- [15] Dr. José Raúl Vento Álvarez, Internet de las Cosas IoT(Internet of Things IoT), Universidad de Pinar del Rio, Cuba, 2017.
- [16] Wikipedia. List of HTTP status codes. Disponible en: <[https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)>. Consultado en: 22 de diciembre de 2017.
- [17] Antonio Javier Sierra Mendoza, GeoJSON y TopoJSON: comparación entre los formatos de intercambio de Información Geográfica alternativos a GML, , 2013.
- [18] Android Developers. Google. Disponible en: <<https://developer.android.com/guide/topics/connectivity/bluetooth-le.html>>. Consultado en: 24 de octubre de 2017.
- [19] Android Developers. Google. Disponible en: <[https://developer.android.com/guide/topics/location/\\_strategies.html](https://developer.android.com/guide/topics/location/_strategies.html)>. Consultado en: 22 de octubre de 2017.

- [20] Francisco José Naranjo Abad (a.k.a Fran), "ANEXO1: PETICIONES REST", Asignatura Desarrollo de Servicios de Comunicación en Red. UPNA, franciscojose.naranjo@unavarra.es, 2016.
- [21] MariaDB. Oficial web. Disponible en: <<http://mariadb.com>>. Consultado en: 24 de enero de 2018.
- [22] GEOJSON.IO. . Disponible en: <geojson.io>. Consultado en: 25 de enero de 2018.

## BIBLIOGRAFÍA

- [1] Andrea Ardións. ¿Cúal es la estructura de un proyecto en Android Studio? AndroidStudioFaqs te lo cuenta. Disponible en: <<https://androidstudiofaqs.com/conceptos/cual-es-la-estructura-de-un-proyecto-en-android-studio>>. Consultado en: 20 de enero de 2018.
- [2] Android Developers. Google. Disponible en: <<https://developer.android.com>>. Consultado en: 24 de octubre de 2017.
- [3] Antonio Javier Sierra Mendoza, GeoJSON y TopoJSON: comparación entre los formatos de intercambio de Información Geográfica alternativos a GML, 2013.
- [4] Argenox. info@argenox.com. Disponible en: <<http://www.argenox.com/a-ble-advertising-primer/>>. Consultado en: 13 de Noviembre de 2017.
- [5] Axoi Bitarra. Zaharberriku mugikorra sofware librearekin, 13 de agosto de 2017. Disponible en: <<https://axoitarra.wordpress.com>>. Consultado en: 20 de enero de 2018.
- [6] Constantino Valero Ubierna, Agricultura de Precisión: conceptos y situación actual, Universidad Politécnica de Madrid, 2001.
- [7] Daniela Ballari, Luis Vilches, Diego Randolph Perez, Diego Pacheco y Virginia Fernández, Tendencias en infraestructuras de datos espaciales en el contexto latinoamericano, 2014.
- [8] David Pérez, El androide libre. ¿Cuánto de Android es libre?. Disponible en: <<https://elandroidelibre.elespanol.com/2016/11/android-es->

[libre.html](#)>. Consultado en: 20 de enero de 2018.

- [9] EcuRed. ecured@idict.cu. Disponible en: <<https://www.ecured.cu>>. Consultado en: 8 de noviembre de 2017.
- [10] Eduardo Huerta, Aldo Mangiaterra y Gustavo Noguera. GPS Posicionamiento Satelital, UNR Editora, Universidad Nacional de Rosario, 2005.
- [11] Empresa Cartografía y Soluciones Geomáticas GeoSí y Grupo Empresarial GEOCUBA. Disponible en: <<http://www.iderc.cu>>. Consultado en: 19 de diciembre de 2017.
- [12] Esmeralda Rojo de Benito y Esther San José Carreras, Biota Tecnología Forestal, PROYECTO IOT PARA LA TOMA DE DATOS DE TEMPERATURA DEL ASFALTO EN CIUDAD, 2017. Disponible en: <<https://biotatfblog.wordpress.com/>>.
- [13] Francisco José Naranjo Abad (a.k.a Fran), "ANEXO1: PETICIONES REST", Asignatura Desarrollo de Servicios de Comunicación en Red. UPNA, franciscojose.naranjo@unavarra.es, 2016.
- [14] GEOJSON.IO. Disponible en: <[geojson.io](http://geojson.io)>. Consultado en: 25 de enero de 2018.
- [15] Héctor Luis Cué León, Sistema de adquisición de datos espaciales para agricultura de precisión usando un teléfono móvil con Sistema Operativo Android como terminal de obtención de los datos, Universidad de Pinar del Rio (UPR), Cuba, 2017.
- [16] Henri J. G. L. Aalders y Harold Moellering, Spatial Data Infrastructure, Katholieke Universiteit Leuven, 2006.
- [17] IDE de España (IDEE), Introducción a las Infraestructuras de

Datos Espaciales.

- [18] iQonsulting Ltda. con la colaboración de Stanley Best (Ing. Agr. M.Sc. Dr., ProgapINIA) e Inés Zamora (Ing. Agr. M.Sc.), *Tecnologías Aplicables en Agricultura de Precisión*, Santiago de Chile, 2008.
- [19] IUCN (International Union for Conservation of Nature), *MANUAL TALLER DE FORMACIÓN EN GEOPORTALES*, Tánger, 22 de enero de 2014.
- [20] J. Agüera Vega y M. Pérez Ruiz, «ambient@,» [En línea]. Disponible en: <[www.revistaambienta.es](http://www.revistaambienta.es)>. Consultado en: 25 octubre 2017.
- [21] James Smith, British, LOOPJ, Android Async HTTP. Disponible en: <<http://loopj.com/android-async-http/>>. Consultado en: 8 de enero de 2018.
- [22] Javier Valencia Martínez de Antoñana, *PASADO, PRESENTE Y FUTURO DE LAS INFRAESTRUCTURAS DE DATOS ESPACIALES*, Trabajo Fin de Máster, Valencia, 2011.
- [23] Jhon Jairo Padilla Aguilar y Javier Pinzón Castellanos, *ESTÁNDARES PARA CLOUD COMPUTING: ESTADO DEL ARTE Y ANÁLISIS DE PROTOCOLOS PARA VARIAS NUBES*, Universidad Pontificada Bolivariana, 9 de mayo de 2015.
- [24] José Raúl Vento Álvarez, *Internet de las Cosas IoT(Internet of Things IoT)*, Universidad de Pinar del Rio, Cuba, 2017.
- [25] Juan Camilo Fajardo Junco, *Apoyo a la agricultura de precisión en Colombia a partir de imágenes adquiridas desde vehículos aéreos no tripulados (UAV`s)*, trabajo de fin de grado, Pontificia Universidad Javeriana, Bogotá, 2014.
- [26] M. A. Bernabé y C. M. López, *Fundamentos de las Infraestructuras*

de Datos Espaciales (IDE), Madrid: UPM Press, 2012.

- [27] MariaDB. Oficial web. Disponible en: <<http://mariadb.com>>. Consultado en: 24 de enero de 2018.
- [28] Mejor Código, Crear REST API con Slim - PHP y MySQL, 2017. Disponible en: <<http://cursos.mejorcodigo.net>>.
- [29] Miguel Ángel Lozano, Acceso a servicios REST, Universitat d'Alacant, 2017. Disponible en: <<http://www.jtech.ua.es>>.
- [30] Miguel Luaces, Víctor Olaya y Oscar Fonts, Infraestructuras de Datos Espaciales. Disponible en: <<http://volaya.github.io/libro-sig/chapters/IDE.html>>.
- [31] Open Geospatial Consortium. OGC. Disponible en: <<http://www.opengeospatial.org>>. Consultado en: 26 de Octubre de 2017.
- [32] P. Palencia Herranz, SISTEMA DE GESTIÓN DE PERFILES POR UBICACIÓN PARA TELÉFONOS ANDROID, Leganés, 2015.
- [33] Parvez, Create simple REST API using Slim Framework, 2017. Disponible en: <<https://www.phpflow.com>>.
- [34] PROCISUR, Manual de Agricultura de Precisión, Montevideo, 2014.
- [35] RFC 7946, The GeoJSON Format, Internet Engineering Task Force (IETF), 2018.
- [36] Sgoliver . SGOLIVER.NET. Disponible en: <<http://www.sgoliver.net/blog/componentes-de-una-aplicacion-android/>>. Consultado en: 20 de enero de 2018.
- [37] Slim Framework. Official Site. Disponible en:

<<https://www.slimframework.com/>>.

- [38] Stackoverflow, Stack Exchange Inc. Disponible en:  
<<https://stackoverflow.com/>>.
- [39] Tomás Fernandez de Sevilla Riaza, GML: El lenguaje de marcado extendido (XML) para la Ingeniería Geográfica. Ventajas y aplicaciones, Madrid, 15 de Abril de 2005.
- [40] Victor Robles, API REST con Slim, 2015. Disponible en:  
<<https://victorroblesweb.es>>.
- [43] Wikipedia. Fundación Wikimedia, Inc.. Disponible en:  
<<https://es.wikipedia.org>>. Consultado en: 22 de febrero de 2018.
- [42] Yassir Akhayad, Bluetooth 4.0 Low Energy: Análisis de las prestaciones y aplicaciones para la automoción, Trabajo Fin de Grado, Universitat Politècnica de Catalunya, 2016.

## ANEXOS

### Anexo 1. Especificaciones sensor externo BeeWi SmartClim

Radio	Bluetooth Low Energy
Alcance	Hasta 30 metros
Classe	IP 43
Humedad - Gama de funcionamiento	De RH 0 a 100% RH Precisión ± 4.5 % máx a RH 20-80%
Temperatura - Gama de funcionamiento	de -20 a +85 °C / de -4° a 185°F Precisión estándar ±0.5 °C / ± 0.5°F Precisión mínima ±1 °C / ± 0.9°F
Consumo	0,28 mA
Tensión de funcionamiento	3 V (2 pilas AAA)
Autonomía	Aproximadamente 6 meses
Dimensiones	80x80x22 mm
Peso	Aproximadamente 30 g



UNIVERSIDAD DE PINAR DEL  
RÍO  
“HERMANOS SAÍZ MONTES DE  
OCA”  
CALLE MARTÍ No. 270 FINAL. CP. 20100 PINAR DE RÍO



### Proyecto Fin de Carrera Opinión de los tutores

Título del Proyecto: “Sistema de Adquisición de datos espaciales con terminales Android y almacenamiento en nube simulada para agricultura de precisión”

Diplomante: Francisco Javier Guembe Lara

#### Informe

El proyecto de diploma desarrollado responde a la necesidad de sistemas de adquisición de datos espaciales georreferenciados a costo moderado, con destino a su uso en la planificación y ejecución de los procesos agrícolas sobre la base de pequeñas dimensiones que permitan un acertado accionar sobre los cultivos u otras formas productivas, además, que esté al alcance de los pequeños productores de limitados recursos económicos.

La solución propuesta resultó en un sistema prototípico de adquisición de datos espaciales con sensores de temperatura y humedad, quedando georreferenciados con información GPS. Este conjunto se conforma teniendo como base y sistema de medición un teléfono inteligente con Sistema Operativo Android para el manejo de los datos capturados y el proceso de comunicación a la nube de almacenamiento de datos a través de redes inalámbricas TCP.

El trabajo de investigación y su informe final cumplen satisfactoriamente los objetivos propuestos en el mismo. La investigación fue diseñada adecuadamente, definiendo con precisión los objetivos y tareas concretas a desarrollar. Se realizó una extensa recopilación bibliográfica sobre sistemas de agricultura de precisión, sensores de temperatura y humedad, sistema operativo Android, formatos de intercambio de datos espaciales, bases de datos georreferenciadas, etc.

El documento presentado como resultado final de todo el proceso detalla cada paso desarrollado en el proyecto. Quedó plasmado al detalle los fundamentos del funcionamiento de cada componente del sistema y sustentada cada selección de alternativa tecnológica. El mismo constituye un material de consulta de elevado valor para el trabajo futuro con estas tecnologías.

En todo el proceso de este ejercicio de culminación de estudios, el estudiante demostró cualidades humanas que realzan el valor de la investigación y el informe final de resultados. Lo caracterizó la humildad, la tenacidad, la dedicación, el valor, la capacidad de trabajo, la seguridad en sí mismo y su preparación académica.

El proyecto de diploma cumplió ampliamente los objetivos propuestos y demostró de forma evidente el nivel alcanzado por el diplomante durante su formación.

Solicitamos, por tanto, se le otorgue la calificación de 5 puntos.

A 1 de marzo de 2018

Dr. Jose Raúl Vento Álvarez  
Universidad de Pinar del Río.  
TUTOR

Ing. Marcos Lazaro Alvarez Arteaga  
Universidad de Pinar del Río.  
TUTOR

