Trabajo de Fin de Grado

Aplicación de XBee a sensado agrícola

XBee application to agricultural sensing

Abraham Barreto Luis de la Guardia

La Laguna, 5 de julio de 2016

D. **Alberto Francisco Hamilton Castro**, con N.I.F. 43.773.884-P profesor Titular de Universidad adscrito al Departamento de “Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores” de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

“Aplicación de XBee a sensado agrícola”

ha sido realizada bajo su dirección por D. **Abraham Barreto Luis de la Guardia**, con N.I.F. 54.044.102-J.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de julio de 2016

Agradecimientos

Quiero agradecer especialmente a mi familia y amigos, por su apoyo y comprensión.

A Airam por su ayuda a la hora de escoger las magnitudes útiles en el cultivo objetivo

A mi tutor por el apoyo prestado

Resumen

El objetivo de este trabajo ha sido desarrollar un sistema basado en Internet de las Cosas (IoT) y Sistemas empotrados relacionados con el medio rural. En particular se ha optado por construir una red de estaciones meteorológicas asociadas a un determinado cultivo.

Desde un servidor central, y por medio de una web podremos configurar la red de estaciones meteorológicas, ver los datos de forma gráfica, exportarlos, y generar alertas y sugerencias de actuaciones en el cultivo. Por ejemplo: recolectar antes, usar un antifungico, inspeccionar la zona de cultivo ...

En la medida de lo posible se ha optado por sistemas con consumos lo más reducidos y comunicaciones inalámbricas para facilitar y abaratar la puesta en marcha y el mantenimiento.

Palabras clave: Internet de las Cosas, Sistemas Embebidos, Webservices, Estación meteorológica.

Abstract

The objetive of this project has been to develop a system based on “Internet of Things” and “Embedded Systems” in rural areas. Particullary, it has been chosen to build a net of wheather stations associated with a specific crop.

In a central server, on a web page, we could configurate the Weather Station net, see the data in a graphic way, export it and generate alert and suggestion of action in the crop. For example: collect earlier, use a antifungal, inspect the growing field ...

As much as possible we choose low consumption systems and wireless conections to make easier and cheaper setup and maintenance.

Keywords: Internet of Things, *Embedded Systems*, Webservices, Weather Station

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Sistema propuesto

La idea es construir un sistema de alerta meteorológica asociado al cultivo de la viña, según una serie de parámetros (humedad ambiental, del suelo, insolación, temperatura) que nos permita: evitar problemas derivados de condiciones adversas a través de una serie de sugerencias, poder acceder en tiempo real a las condiciones meteorológicas y disponer de un histórico.

Dicho sistema consta de diversos subsistemas embebidos trabajando juntos para poder tener estaciones meteorológicas de bajo consumo y una estación central que coordina y comunica con las estaciones.

Además, el sistema se adapta a zonas con mala cobertura de internet móvil y sin acceso a ADSL, situación que es común en los cultivos de viña. Solventamos este problema conectando el servidor auxiliar con el servidor central por medio de una conexión wifi de larga distancia.

Capítulo 2
Conocimientos previos:

2.1 Internet de las Cosas (IoT)

Hablando en general, Internet de las Cosas se refiere a la interconexión de objetos cotidianos, equipados con capacidad de proceso. Internet de las Cosas aumentará el alcance de Internet mediante la integración de objetos para la interacción a través de sistemas embebidos. Esto nos llevará a una red altamente distribuida con dispositivos comunicándose con los seres humanos, así como otros dispositivos. Algunos objetos sólo transmitirán información mientras que otros tendrán capacidad también de interactuar más activamente con otros elementos en la red.

Conforme las tecnologías subyacentes a Internet de las Cosas van generalizándose y abaratándose, se están abriendo enormes oportunidades a la vez que mayores retos. Ejemplo de estos retos son:

- Creación de topología de red para la conexión eficiente entre cada uno de los diversos elementos y su correcta identificación.
- Interferencias en la conexión inalámbrica: Al crecer el número de elementos hay que repartir el ancho de banda
- Comunicaciones inalámbricas de bajo consumo: Se han creado diversos estándares e implementaciones (IEEE 802.15.4, ZigBee, 6LoWPAN)

En los últimos años, Internet de las Cosas ha ganado mucha atención por parte de investigadores y profesionales de todo el mundo. [1]

2.2 ZigBee

2.2.1 Descripción

ZigBee es un conjunto de protocolos de alto nivel de comunicación que se utiliza para la transmisión digital e inalámbrica de datos buscando un bajo consumo. A nivel físico y de control de acceso al medio está basado en el estándar IEEE 802.15.4. Ha sido desarrollado por la "ZigBee Alliance". [2]

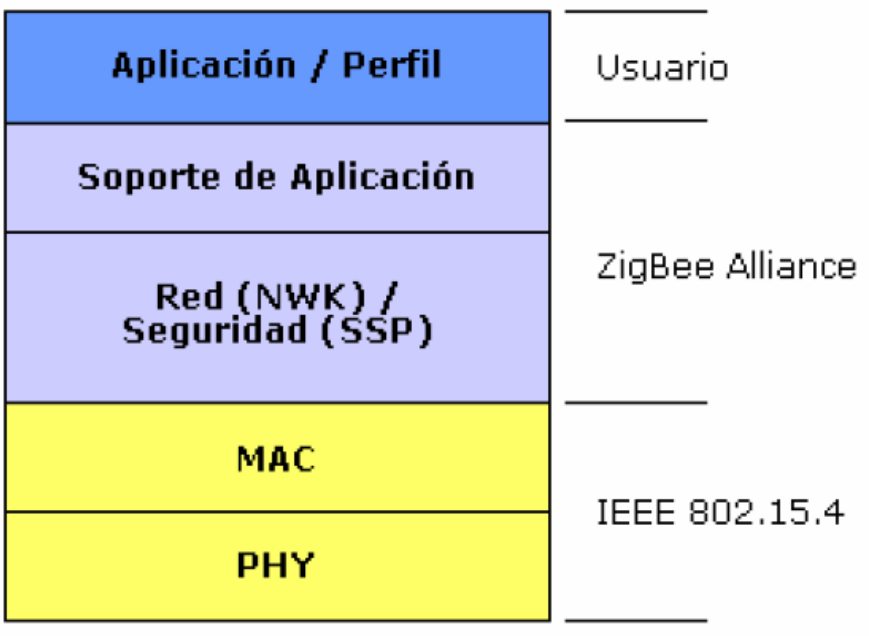

Figura 2.1: Protocolos de ZigBee

Está pensado para aplicaciones de domótica e Internet de las Cosas. Tiene una velocidad de hasta 250 kbps y un consumo de 30 mA transmitiendo y de 3 µA en reposo.

Hace uso de **CSMA/CA** (acceso múltiple con escucha de portadora y evasión de colisiones) al igual que la WiFi para solventar el problema del acceso múltiple al medio de transmisión.

Cómo técnica de modulación hace uso de la modulación por desplazamiento de fase (**OQPSK**) y Espectro ensanchado por secuencia directa (**DSSS**).

ZigBee utiliza la banda ISM para usos industriales, científicos y médicos. Puede utilizar diferentes frecuencias, y para cada una de ella dispone de un número posible de canales. Específicamente, 868 MHz en Europa (1 canal) , 915Mhz en Estados Unidos y Japón (10 canales) y 2,4 GHz en todo el mundo (15 canales).

A la hora de diseñar dispositivos, se suele optar por la banda de 2,4 GHz. [8]

2.2.2 Tipos de dispositivos

Se definen tres tipos distintos de dispositivo ZigBee según su papel en la red:

- *Coordinador ZigBee:* El tipo de dispositivo más completo. Debe existir al menos uno por red. Sus funciones son las de encargarse de controlar la red y los caminos que deben seguir los dispositivos para conectarse entre ellos.
- *Router ZigBee:* Interconecta dispositivos en la red, además de ofrecer un nivel de aplicación para la ejecución de código de usuario (Puede actuar también cómo dispositivo final). En este proyecto no haremos uso de Router Zigbee
- *Dispositivo final:* Posee la funcionalidad necesaria para comunicarse con su nodo padre (el coordinador o un router), pero no puede transmitir información destinada a otros dispositivos. De esta forma, este tipo de nodo puede estar dormido la mayor parte del tiempo, aumentando la vida media de sus baterías. Un dispositivo final tiene requerimientos mínimos de memoria y es por tanto potencialmente más barato. [2] [8]

2.3 Sistemas embebidos

Los sistemas embebidos son sistemas de computación integrados, construidos con un fin específico, que controlan una o varias funciones, con recursos limitados e incluso en condiciones ambientales hostiles.

Ejemplos de ello serían un taxímetro, un módem, el sistema de control de un ascensor ... En algunos sistemas de mayor tamaño, cómo podría ser un coche, pueden llegar a coexistir varios sistemas embebidos.

Estos sistemas de procesamiento integran procesador, memoria, procesamiento digital, comunicaciones software, FPGAs, etc. según las necesidades específicas, usualmente en la misma placa base [3][10]

2.3.1 XBee

Son unos chips fabricados por la empresa Digi que utilizan el protocolo ZigBee. Fueron diseñados para aplicaciones que

requieren de un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible.

Hacen uso del microcontrolador MC9S08GT60 y del chip de radiofrecuencia MC13193. [9]

Existen varias series:

- **XBee Series 1:** Primer modelo de la serie. No son configurables y son incompatibles con el resto de los modelos
- **XBee Znet 2.5 (Formalmente Series 2):** Deben ser configurados antes de ser usados. Pueden funcionar en modo Transparente o por medio de comandos API, pero todo esto depende de cual firmware se configure en los módulos. Pueden formar redes mesh
- **ZB (Series2):** Módulo Znet 2.5 con un nuevo firmware. Funciona en modo transparente o por medio de comandos API. Se puede actualizar el firmware de un Znet 2.5 para que funcione cómo un Series 2.
- **2B(el actual módulo Series2)**: Son nuevos módulos que poseen mejoras en el hardware respecto de los de la Serie 2, mejorando por ejemplo el uso de la potencia. Funcionan con el Firmware del módulo ZB, pero debido al cambio de hardware, ya no pueden funcionar con el firmware del módulo Znet 2.5. Por lo que hay que tener cuidado si agregas uno de estos módulos a una red ya existente que utilice módulos Znet 2.5.
- **900MHz:**Estos módulos pueden funcionar con dos diferentes tipos de firmware, el firmware DigiMesh y el firmware Point-to-Multipoint. Digi actualmente vende ambos módulos, el hardware es el mismo, pero con diferentes firmware. Estos módulos son no configurables. Los módulos 900MHz tienen una velocidad de datos de aproximadamente 156 Kbps, menor a la de otros módulos (250 Kbps).
- **XSC:** Módulos de 900MHz con menor velocidad de datos a cambio de un alcance aumentado. La velocidad de datos en los módulos XSC es de alrededor de 10 Kbps. Si se usa una antena de alta ganancia se puede tener un alcance de alrededor de 24 Km y de 9,6 Km con una antena regular. Estos módulos no requieren configuración externa y tienen otras diferencias incluyendo un conjunto de comandos diferente a los anteriores por lo que se recomienda revisar los comandos antes de sustituir módulos 900Mhz por éstos.

Además, disponemos de distintos tipos de antena según el uso que se quiera dar:

- **Chip Antenna:** Básicamente es un pequeño chip que actúa como antena. Rápido, sencillo y barato.
- **Wire Antenna (Whip Antenna):** Pequeño cable que sobresale por la parte superior del chip.
- **u.FL Antenna**– Un conector pequeño para conectar tu propia antena. Ideal si tienes tu equipo en una caja y deseas la antena afuera de ésta.
- **RPSMA Antenna**– Un conector más grande para conectar tu propia antena. Nuevamente, esto es perfecto si tienes tu equipo en una caja y deseas la antena afuera de ésta.[7]

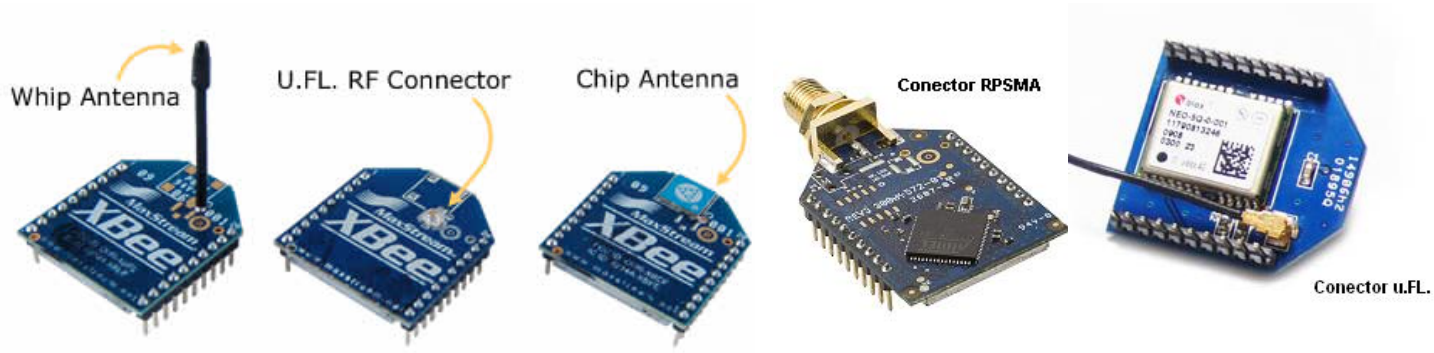

Figura 2.2: Tipos de antenas y conectores XBee

Hay que tener en cuenta que dependiendo del tipo de antena escogida se deberá evitar colocar objetos metálicos que cubran el módulo en aras de no atenuar la señal. [6]

En este proyecto se ha optado por modelos “XBee Series2” con “Chip antenna”.

Los módulos XBee pueden funcionar en modo transparente o en modo API.

En modo transparente funcionan cómo sustitución de un cable serie (USB) entre dos puntos. Los datos recibidos a través del pin DIN se encolan para su trasmisión inalámbrica. Cuando se reciben datos a través de la red inalámbrica, se sacan a través del pin DOUT [6]

En modo API, los datos que entran y salen del módulo está contenido en “Frames” que definen operaciones o eventos en dicho módulo. La API proporciona diversas opciones configurables para los

módulos, paquetes de recepción de información, broadcasting, etc. [6]

2.3.2 XBee – Patillaje Módulo Znet 2.5

En la tabla 2.1 se muestra el patillaje de los módulos XBee utilizados en este proyecto:

Pin#	Nombre	Dirección	Descripción
1	VCC	-	Fuente de alimentación
2	DOUT	Salida	Salida de datos de la UART
3	DIN / **$\overline{\text{CONFIG}}$**	Entrada	Entrada de datos de la UART
4	DIO12	Cualquiera	E/S Digital 12
5	**$\overline{\text{RESET}}$**	Entrada	Resetear Modulo (pulso de reseteo debe ser de al menos 200 ns)
6	PWM0 / RSSI / DIO10	Cualquiera	Salida PWM 0 / Indicador de Fuerza de la señal RX / E/S Digital 10
7	PWM / DIO11	Cualquiera	E/S Digital 11
8	[reservado]	-	No conectar
9	**$\overline{\text{DTR}}$** / SLEEP_RQ/ DIO8	Cualquiera	Pin de Linea de Control de Dormir o E/S Digital 8
10	GND	-	Tierra
11	DIO4	Cualquiera	E/S Digital 4
12	**$\overline{\text{CTS}}$** / DIO7	Cualquiera	Flujo de Control Listo-Para-Enviar (CTS) o E/S Digital 7
13	ON / **$\overline{\text{SLEEP}}$** / DIO9	Salida	Indicador de Estatus del Módulo o E/S Digital 9
14	[reservado]	-	No conectar
15	Associate / DIO5	Cualquiera	Indicador de Asociación o E/S Digital 5

16	**$\overline{\text{RTS}}$** / DIO6	Cualquiera	Flujo de Control Solicitud-Para-Enviar (RTS) o E/S Digital 6
17	AD3 / DIO3	Cualquiera	Entrada Analógica 3 o E/S Digital 3
18	AD2 / DIO2	Cualquiera	Entrada Analógica 2 o E/S Digital 2
19	AD1 / DIO1	Cualquiera	Entrada Analógica 1 o E/S Digital 1
20	AD0 / DIO0 / Commissioning Button	Cualquiera	Entrada Analógica 0, E/S Digital 0 o Botón de Puesta en Marcha

Tabla 2.1: Patillaje

Notas:

• La dirección de la señal está especificada respecto al módulo

[6]

2.3.3 XBee - Comandos AT utilizados

Para la comunicación con los módulos se hace uso de instrucciones AT con un frame específico. Existe información exhaustiva de éstos en el Datasheet de los módulos XBee. [6]

Para este proyecto haremos uso de los siguientes, explicados de forma resumida:

- **D**XY: Lectura / Escritura de la configuración de las entradas/salidas, siendo X el número de pata e Y el modo de configuración.
  - X valdrá 0-3 para las entradas analógicas
  - Y Puede ser:
    - 0 Entrada digital no monitorizada
    - 1 Reservado para funciones alternativas específicas del pin
    - 2 Entrada analógica, entrada única (“single ended”). Las lecturas que se realicen tendrán valores de 0 a 0x3FF, siendo este último valor el máximo valor que puede leer (1.2 Voltios)

        - 3 Entrada digital, monitorizada
        - 4 Salida digital, por defecto bajo (0)
        - 5 Salida digital, por defecto alto (1)
        - 6-9 Funcionalidades alternativas, cuando sea aplicable
    - Ejemplo: Pin 2, salida digital en alta: D25
- **SP**X: “Sleep Period” Tiempo que el nodo permanece dormido.
    - Siendo X un número de centésimas de segundo. Su rango va de 0x20 a 0xAF0 con una resolución de un cuarto de segundo.
    - En el nodo Coordinador, representa el tiempo que guardará los paquetes en el buffer antes de descartarlos. En este caso, el tiempo que guarda los paquetes se computa cómo el valor de SP * 2.5, sin exceder 30 segundos.
    - En los nodos “hoja” duerme por el tiempo que se establezca mediante este comando. Puede configurarse para que duerman SP * SN (siendo SN un factor multiplicador entre 1 y 0xFFFF), pero dormir más de 28 segundos implica que no se puede asegurar que los nodos recibirán los comandos que se le envíen. Debido a esto se ha descartado hacer uso de esta opción.
- **ST**X: Tiempo antes de dormir, o dicho de otra forma, el tiempo que un nodo está despierto antes de dormir de nuevo. Sólo es aplicable en nodos “hoja”
    - Siendo X un número en el rango de 1 a 0xFFFE milisegundos
- **SM**: Modo de sueño. (Sólo los nodos hoja pueden dormir) Posibles valores:
    - 0: Dormir desactivado.
    - 1: Dormir cuando el pin 9 (SLEEP_RQ) esté en alta.
    - 4: Dormir cíclico.
    - 5: Dormir cíclico. Igual al anterior con la salvedad de que también se puede despertar al nodo con una transición alta a baja en el pin 9 (SLEEP_RQ).
- **IS:** Realiza una lectura de todas las entradas analógicas y digitales configuradas.
- **IR**X: Igual que el anterior comando, pero realizando las

lecturas de manera cíclica.

  - Siendo X un número en el rango 0 a 0xFFFF (ms)

- **ND**: Éste comando obliga a todos los nodos “hoja” a identificarse. Es útil para hacer el setup inicial o si tenemos nodos de los cuales desconocemos su identificador.
- **NI**: Este comando lee el nombre de un nodo o, seguido de un string, establece el nombre de un nodo (hasta 20bytes).
- **ID**x: Configura el identificador de Red (PAN ID), estando x en el rango 0 - 0x3FFF, 0xFFFF. En el coordinador establece cual será su PAN ID, siendo 0xFFFF autoseleccionar la red. En los nodos “hoja” establece a que PAN ID se unirán, usándose el valor 0xFFFF para que se una a cualquier red

  [6]

2.3.4 Raspberry Pi

Es un SOC (System On a Chip), un pequeño PC en un circuito integrado, que permite correr Linux. Tiene un consumo menor que un PC de escritorio, y es de un tamaño reducido.

Existen actualmente los modelos A y B, que van desde la versión 1 a la 3. Existe un modelo reducido llamado Zero que no se guía por esta convención de nombres. [4]

En la siguiente tabla podemos ver las diferencias entre los distintos modelos. Los precios son orientativos.

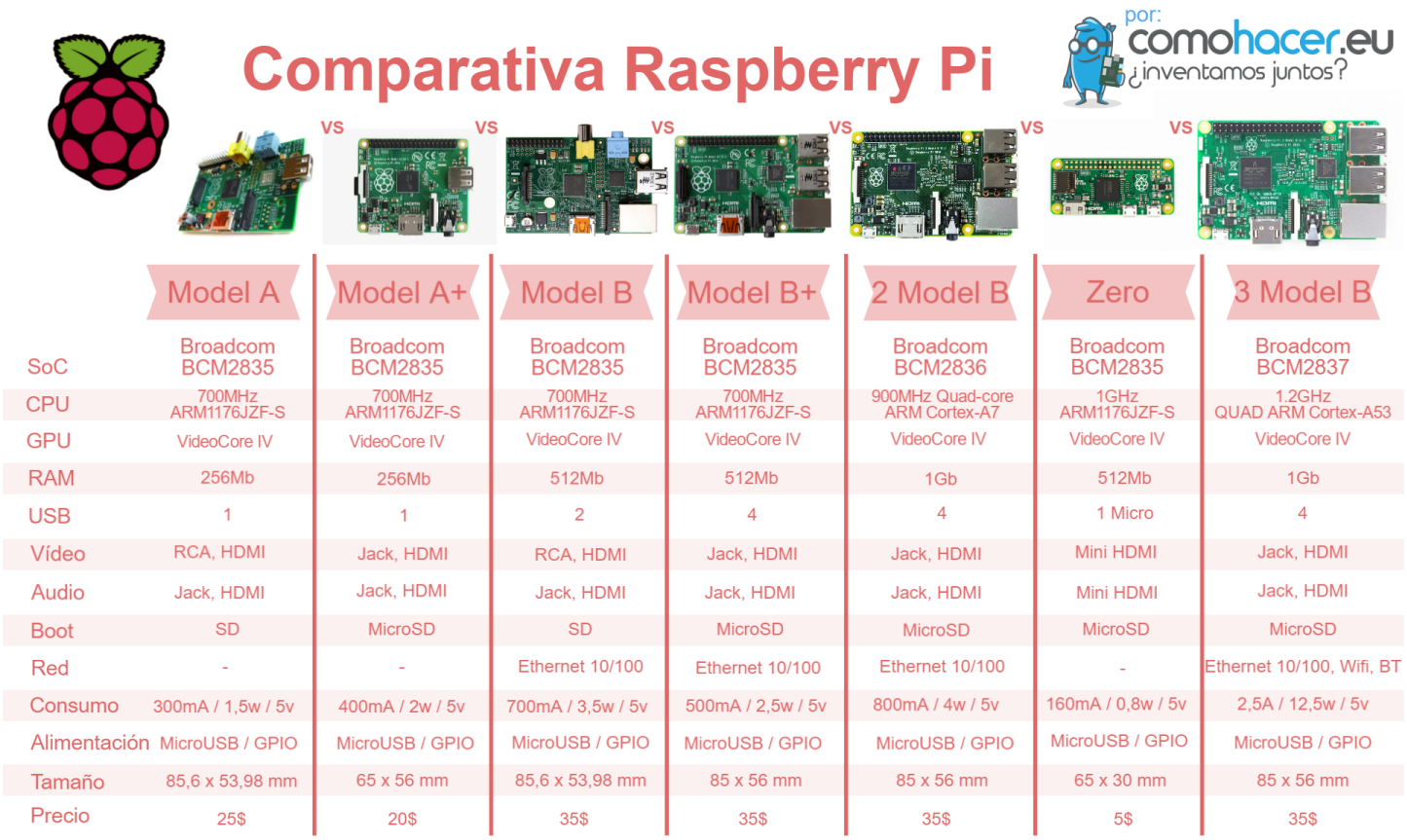

Figura 2.3: Comparativa modelos Raspberry Pi

2.4 Webservices

El término Webservices describe una forma estandarizada de integrar aplicaciones WEB mediante el uso de XML, SOAP (o XML-RPC) y WSDL sobre los protocolos de la Internet. XML es usado para describir los datos, SOAP se ocupa para la transferencia de los datos, WSDL se emplea para describir los servicios disponibles.

Los Webservices comparten la lógica del negocio, los datos y los procesos, por medio de una interfaz de programas a través de la red. Es decir conectan programas, por tanto son programas que no interactúan directamente con los usuarios.

Los Webservices permiten a distintas aplicaciones, de diferentes orígenes, comunicarse entre ellos sin necesidad de escribir programas costosos, esto porque la comunicación se hace con XML. Los Webservices no están ligados a ningún Sistema Operativo o Lenguaje de Programación. Por ejemplo, un programa escrito en Java puede comunicarse con otro escrito en Perl; Aplicaciones

Windows pueden comunicarse con aplicaciones Unix.

A continuación se describen brevemente los estándares en los que se basan los Webservices:

- **XML**: Abreviación de Extensible Markup Language. El XML es un lenguaje de marcas creado por W3C. Permite a los desarrolladores crear sus propias etiquetas siempre que sigan un formato. De esta forma, se pueden expresar diferentes datos y gramáticas de forma legible.
- **SOAP**: Abreviación de Simple Object Access Protocol , es un protocolo de mensajería construido en XML que se usa para transmitir y responder mensajes estructurados en los Web Services. Los mensajes SOAP son independientes de los sistemas operativos se transportan por los protocolos cómo: SMTP, MIME y HTTP. Se suele hacer uso de HTTP
- **WSDL:** Abreviación de Web Services Description Language; es un lenguaje especificado en XML que se ocupa para definir los Webservice. Con este lenguaje expresamos las operaciones disponibles y los parámetros de entrada y salida. Muchas librerías que implentan Webservice permiten su generación automática en base a unos parámetros de configuración. Es un estándar de uso público. [5]

Capítulo 3
Desarrollo

3.1 Diagrama de bloques

El sistema se divide en tres grandes bloques: Servidor central, servidor auxiliar y nodos XBee. Cada uno de estos grandes bloques se puede subdividir a su vez tal cual muestra la siguiente figura:

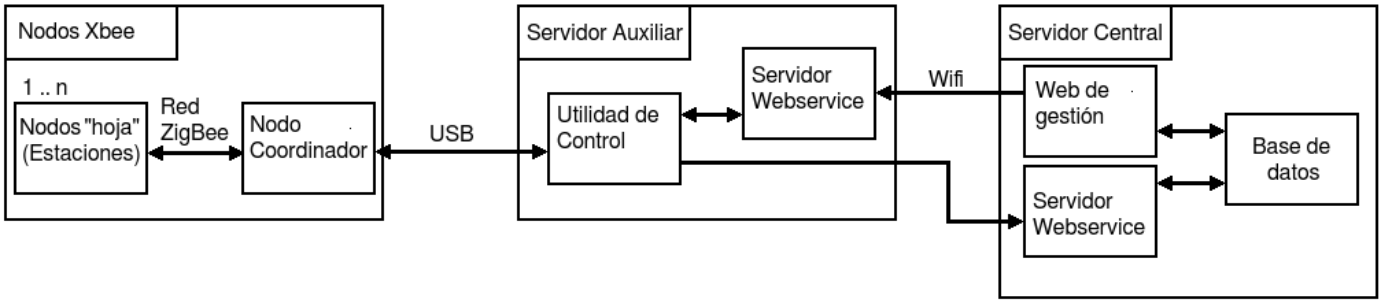

Figura 3.1: Diagrama de bloques

En este capítulo describiremos con más detalle cada una de las partes que conforma el sistema.

3.2 Opciones consideradas

3.2.1 Comunicación entre servidores

Se consideraron varias opciones diferentes para comunicar ambos servidores.

Cómo primera opción se barajó utilizar una comunicación vía socket. Los socket están soportados en multiples lenguajes de programación, garantizan la transmisión de información sin errores y en el orden enviado.

En su contra tienen que la comunicación se establece de punto a

punto (IP + puerto) y sin una definición de cómo se realiza. Esto obliga a gestionar cómo se envía y recibe la información.

Además, dado que hay varios tipos de operaciones diferentes en la comunicación entre los dos servidores, así que se hubiera tenido que implementar varios socket, o uno con algún formato específico ad-hoc que permitiera diferenciar el tipo de transmisión. Hay que añadir que si la información viaja a través de internet, es susceptible de pasar por un firewall, que suelen filtrar la mayor parte de puertos excepto el 80.

Al final se optó por el uso de webservices. Se valoró utilizar XML-RPC o SOAP cómo protocolo de envío de información. Para el proyecto que se esta abordando ambas soluciones son igual de válidas, aunque SOAP está más extendido. Tras hacer pruebas con ambas opciones, y dado que se tenía mayor experiencia en uso de SOAP, se optó por éste último.

Los Webservices tambien están portados a múltiples lenguajes de programación. Tienen la ventaja de mostrar una interfaz externa definida (mediante WSDL), poder definir varias operaciones diferentes, tener capacidad de autentificación, entre otras. Ésto facilita el envío de datos entre diferentes lenguajes de programación y permite una mayor escalabilidad. Además, al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en *firewall* cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera

3.2.2 Lenguajes a usar en los servidores

Ya se disponía de código suministrado por el tutor desarrollado en Python, por lo que se optó por extender su funcionalidad y seguir desarrollando en Python el resto de funcionalidades del servidor auxiliar.

Para el servidor central se valoró utilizar Python, dónde existe una gran comunidad de desarrollo web, y varios Framework para dicho fin, cómo podrían ser Django, TurboGears o web.py [11]

Cómo segunda opción se valoró PHP, y debido a la mayor experiencia en desarrollo en PHP, la facilidad para montar un servidor en diferentes plataformas, y la disponibilidad de gran cantidad de librerías (conexión a BBDD, webservices, XML, etc.) se optó por este último lenguaje.

Aunque los Webservices se basan en estándares, las diferentes implementaciones por lenguajes pueden presentar algunos problemas. En este caso, la decisión de usar lenguajes diferentes disminuyó el tiempo de desarrollo del servidor central pero aumento el tiempo dedicado a los Webservices.

3.2.3 Librerías Webservice

Una vez que se optó por usar Webservices, se debió de escoger que librerías se usarían en ambos servidores. Para PHP se valoró usar la propia librería nativa SOAP o NuSOAP debido a que ambas son las que mayor cantidad de documentación. Se optó por NuSOAP tras hacer algunas pruebas por la tener disponible mejor documentación y ejemplos, la autogeneración de wsdl y su estabilidad respecto a la otra opción.

Python carece de librería nativa SOAP, pero dispone de varias librerías de amplio uso.

Dos de las más utilizadas y las que se valoraron son Suds y PySimpleSoap. La primera es más antigua y tiene mayor cantidad de documentación. Lleva un tiempo sin actualizarse (desde 2010) y no tiene soporte para Python 3.

PySimpleSoap es simple de usar y es capaz de autogenerar el WSDL. Está en desarrollo activo. En su contra tiene la menor cantidad de documentación y la falta de implementación de algunos detalles de los protocolos. [12][13]

3.2.4 Sensores

Para empezar se consultó con un ingeniero agrónomo las magnitudes físicas más relevantes para el cultivo de viña, específicamente. Según su criterio, éstas eran la temperatura y humedad ambiental, humedad del suelo y grado de insolación.

El módulo XBee dispone de 10 entradas digitales. Cómo primera opción para la lectura de la temperatura y humedad ambiental se valoró utilizar el sensor digital DHT11.

Realizar una lectura de un sensor digital se requería un número de instrucciones excesivamente elevado, y eso hubiera obligado a tener un gran trafico en la red y un gran consumo. Además, hubiera supuesto agregar una complejidad excesiva. Para solventar este problema se optó por hacer uso de una placa de Arduino. La idea era realizar la lectura por medio del Arduino, y luego hacer uso de la salida PWM y un condensador para hacer la lectura. Para ahorrar energía, la idea era usar el pin 13 **SLEEP** para activar el Arduino.

Dado el aumento de costo y consumo que supondría el uso de un Arduino se descartó el uso de sensores digitales en favor de sensores analógicos. Esto nos limita el número de magnitudes medibles a 4, número de entradas analógicas del módulo, si no queremos complicar la electrónica.

Para la insolación se valoró utilizar una LDR, una célula fotoeléctrica o un fotodiodo. De estos tres sensores, el primero es el más barato. Tiene un tiempo de respuesta superior al de las otras dos opciones, pero en esta aplicación no es necesario un tiempo de respuesta corto.

Para la humedad del suelo se probó el sensor YL-38, que contiene a la sonda YL-69 y la lógica suficiente para hacer lecturas digitales. Al haberse descartado el uso de sensores analógicos, se opto por prescindir del YL-38 y usar directamente la sonda (con su correspondiente resistencia) para hacer lecturas analógicas.

3.3 Módulos XBee

3.3.1 Nodos “hoja”

Los nodos hoja son los encargados de la recepción de las lecturas de los sensores. Para identificarlos se puede usar una dirección de 64 bits, una de 16 bits o una cadena de texto, opción que se ha escogido por simplicidad y facilidad de uso.

Tienen conectados 4 sensores analógicos en la configuración por defecto: humedad y temperatura del aire, Humedad del suelo, Insolación. Dichos sensores están conectados a las entradas AD0 a AD4 en el orden en el que se ha descrito. Están configurados en la red 0x333. Antes de comenzar a hacer las lecturas hay que configurar dichas líneas cómo entradas analógicas mediante los comandos AD02 – AD12 – AD22 – AD32.

Además, es posible y deseable que los nodos “hoja” duerman de manera cíclica para reducir su consumo. Se puede configurar tanto el tiempo que pasan en estado de sueño cómo el tiempo que permanecen despiertos por medios de los comandos SP y ST respectivamente. Una vez realizado esto, hay que habilitar el modo de sueño cíclico a través del comando SM4.

Las lecturas de los sensores se realizarán de forma cíclica también, tarea que se realiza a través del comando IR, y después de haber configurado los pines adecuados cómo entradas analógicas.[6]

Estas tareas de configuración se realizan desde el servidor principal como se verá más adelante.

Se alimentarán a través de baterías o de pequeñas placas solares.

3.3.2 Nodo Coordinador

El nodo coordinador se halla alojado junto (y conectado a) la Raspberry pi que actúa como servidor auxiliar. Es el encargado de montar la red ZigBee cómo se ha comentado con anterioridad, así cómo de comunicarse con el resto de nodos de la red. Toda transmisión de datos y comandos desde y hacia los nodos “hoja” se realiza a través del nodo coordinador.

Si se hace uso del modo de sueño cíclico hay que configurar el nodo coordinador para que guarde los paquetes que se van a enviar al nodo cómo mínimo el mismo tiempo que los nodos “hoja” duermen, para evitar una posible pérdida de éstos. Para ello disponemos del comando SP.[6]

Cuando arranca el programa de control, hemos de realizar un descubrimiento de nodos en la red, tarea que se realiza enviando el comando ND al nodo Coordinador.

Al igual que sucedía con la configuración de los nodos “hoja” es el servidor central el encargado de enviar dichos comandos.

Se alimenta por medio de un cable USB, que también es usado para la comunicación con el servidor auxiliar.

3.4 Servidor auxiliar

3.4.1 Descripción

El servidor auxiliar estará ubicado en una RaspberryPi con Linux instalado, particularmente, Rasbian. Aloja el software de conexión con la red Xbee y los webservices que se comunican con el servidor central. Dicho software de control está desarrollado en Python. Tal y cómo se ha descrito anteriormente, realiza la comunicación con la red Xbee a través de una conexión USB, y a su vez se conecta al servidor central por medio de un repetidor Wifi con una antena de largo alcance. De esta manera, podemos montar nuestro sistema en zonas sin cobertura de internet, situación habitual en las zonas de cultivo de las zonas altas, por ejemplo. Cómo requisito adicional, ha de situarse otro repetidor Wifi en una zona con internet y linea de visión directa al servidor auxiliar, idealmente junto con el servidor central.

Se ha partido del código suministrado por el tutor y se ha modificado para poder realizar el envío y recepción de las instrucciones y muestras de datos de la red XBee hacia el servidor central por medio de webservices.

3.4.2 Utilidad de control

Es un programa en Python que a través de la interfaz serie (USB) se conecta al nodo coordinador y realiza la comunicación de los comandos desde y hacia la red XBee. Consta de 6 clases:

- **Conexiones.py:**Gestiona la conexión con el módulo XBee a nivel físico.
- **DialogaAPI.py:**Clase para gestionar el dialogo mediante API, el envío de comandos y respuestas a XBee. Debe recibir una conexión inicializada y una interfaz de comunicación.
- **ConsultaAPI.py:** Consola interactiva con la red XBee. Envía y recibe comandos AT. Se puede usar para tareas de mantenimiento y depuración.
- **InterfazComunicacion.py:** Cliente Webservice que realiza los envíos hacia el servidor central. Es aquí dónde tendremos que configurar la dirección del servidor central.
- **UtilidadControl.py:** Programa principal. Hace uso de las otras clases e implementa el servidor webservice mediante la librería pysimplesoap. Por medio de éste, puede recibir comandos destinados a la red, tanto al nodo coordinador cómo a los nodos “hoja”
- **DatosMuestreo.py:** Representación de una lectura de datos, usada para encapsular las muestras analógicas y digitales.

3.4.3 Webservices

Pasamos a describir los webservices usados en el servidor auxiliar con más detalle. Podemos dividirlos entre clientes (**InterfazComunicacion.py)** que se comunican con el servidor central y el servidor (**UtilidadControl.py)**. En esta sección describiremos las funciones que provee el servidor, ya que la parte del cliente hará uso de las funciones que describiremos en el apartado del servidor central.

- Servidor:
  - EnvioLocal: nos permite enviar un comando al nodo coordinador
  - EnvioRemoto: enviar comandos a un nodo hoja (estación)

Ambas funciones reciben cómo parámetro el comando (string) a enviar y devuelven el id (entero) que le asigna el programa gestor a dicha instrucción. En el caso de comandos que se envían a nodos

“hoja”, dichos comandos incluyen tanto el nombre del nodo cómo la instrucción a enviar separados por el símbolo “:”,por ejemplo , si quisiéramos enviar el comando D12 al nodo E09 la instrucción quedaría codificada de la siguiente manera: E09:D12

3.5 Servidor central

3.5.1 Características

El servidor central estará ubicado en un ordenador de sobremesa, correrá un servidor web apache y un servidor MariaDB.

En el servidor web tendremos la página de gestión, los webservices que se encargarán de comunicarse con el servidor auxiliar y la base de datos.

Almacena los datos obtenidos de la red de sensores, datos de configuración y un histórico de comandos.

3.5.2 Webservices

El servidor central hace uso de un cliente (**gestion/funciones.php**, clase Ayudas) y un servidor (**ServidorSOAPFinal.php**). El cliente hace llamadas a los webservices descritos en el servidor auxiliar, para enviar comandos a la red XBee.

Las respuestas de la red XBee se gestionan desde el servidor webservice. Pasamos a describir cada una de las funciones de éste:

- **recibirRespuestaATLocal**: Se encarga de gestionar un comando enviado desde el nodo coordinador. Si la respuesta es del comando “ND” actualiza la tabla de nodos. Guarda la respuesta en la tabla comandoslocales.
- **recibirRespuestaATRemota**: Se encarga de gestionar un comando enviado desde un nodo hoja (estación). Actualiza la fecha de la última transmisión para el nodo que envía la respuesta. Guarda la respuesta en la tabla comandosremotos.
- **RecibirLecturaMuestreo**: Gestiona una lectura de los sensores. Transforma la lectura de voltaje en la magnitud correspondiente y la almacena en la BBDD. Si hay alguna alerta asociada a alguna magnitud asociada a un sensor, envía el correo con la sugerencia asociada. En la siguiente sección se explica mejor el mecanismo de envío de alertas.

3.5.3 Web de gestión

El servidor alojará también una web de gestión, a través de la cual interactuaremos con el resto del sistema. Hace uso de bootstrap, jquery y Chart.js.

Pasaremos a describir las páginas de gestión:

- **Configuración de sensores**:

  En esta página podremos ver, crear, modificar y eliminar los sensores de los cuales haremos uso. Tienen un nombre, miden una magnitud y tienen una ecuación que transforma la medida de voltaje en una medida de la magnitud medida. Puede hacerse uso de funciones de PHP y las constantes VCC (voltaje de la fuente) y V (voltaje medido)

Figura 3.2: Listado de sensores

Figura 3.3: Modificar sensores

- **Gestión de las estaciones**:

  En esta página veremos cada uno de los nodos hoja, su nombre, fecha de la última transmisión, horas inactivo y modo de funcionamiento. Cada fila tiene un código de color que se configura en la página de configuración para ver rápidamente si hace tiempo que no transmite (hay dos niveles: atención y error, amarillo y rojo).

  También podremos buscar nuevos nodos (en el icono de la lupa) de manera automática a través de la instrucción ND [6]

  En el último botón al lado del título enviaremos la configuración almacenada a los nodos.

Figura 3.4: Gestión de las estaciones

En la página de modificar podremos configurar el nombre, el modo y qué sensores tiene conectado a cada una de las posibles entradas analógicas (AD0-AD3). Se puede ver (pero no modificar) la fecha de la última transmisión y si el nodo ha sido configurado. Un nodo no está configurado cuando lo encontramos a través de la búsqueda automática.

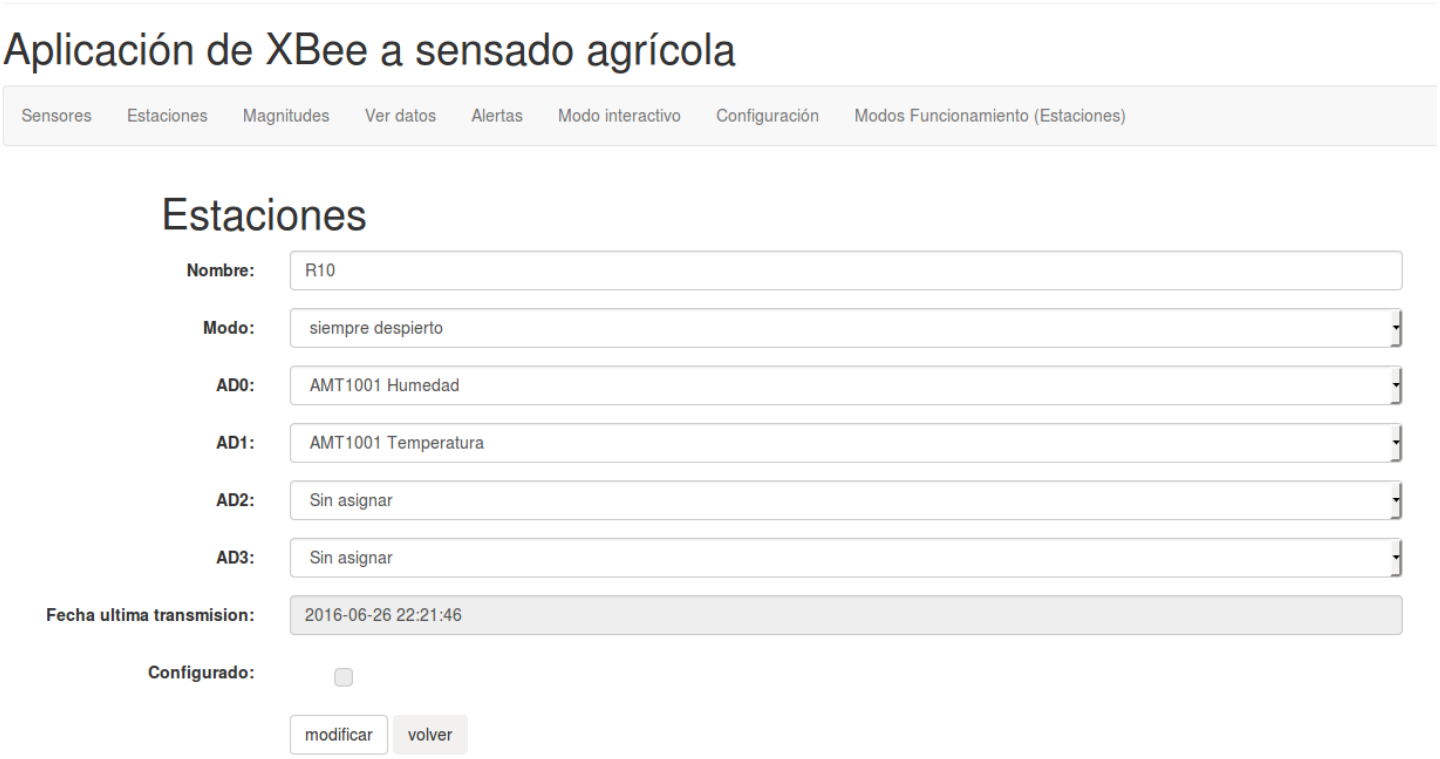

Figura 3.5: Modificar una estación

- **Configurar magnitudes**:

  En estas páginas se configuran las magnitudes que miden los sensores (temperatura, humedad, etc.). Cada una de ellas tiene un nombre, una magnitud y una abreviatura.

Figura 3.6: Listado de las magnitudes

Figura 3.7: Modificar una magnitud

- **Ver datos:**

  Aquí veremos los datos recabados de forma gráfica, para cada nodo y sensor. También podremos descargar todos los datos en formato CSV comprimidos en un ZIP.

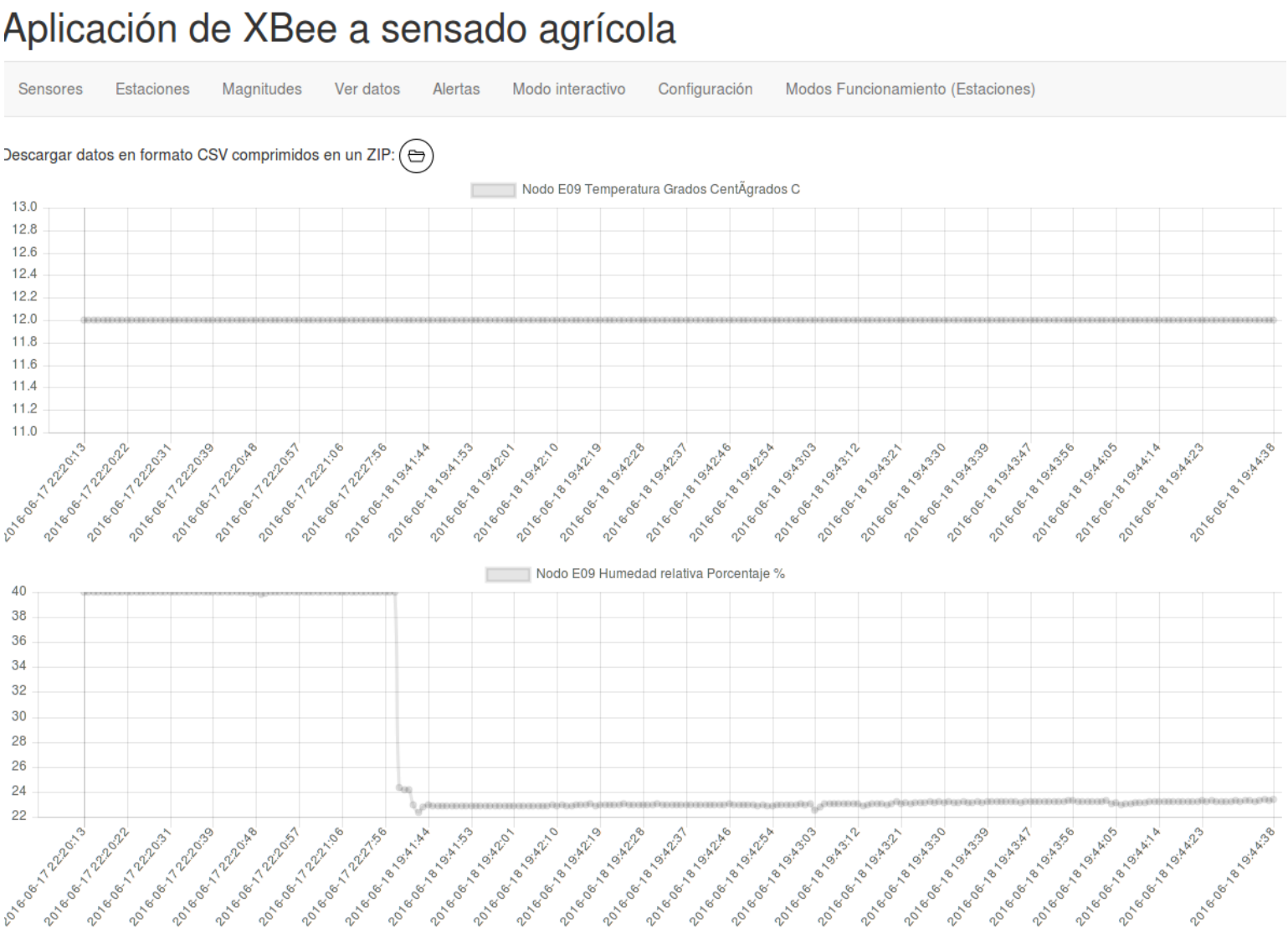

Figura 3.8: Ver Gráficas con los datos

- **Alertas**:

  Aquí vemos y configuramos las alertas. Cada alerta tiene un nombre, se establece sobre una magnitud cuando el valor sea mayor o menor que un limite que estableceremos. Cuando se cumpla la condición, enviaremos un correo electrónico a la dirección configurada en la página de configuración con la sugerencia de la alerta. En la sugerencia se sustituye la cadena de texto “%s” por el nombre de la estación que ha generado la alerta.

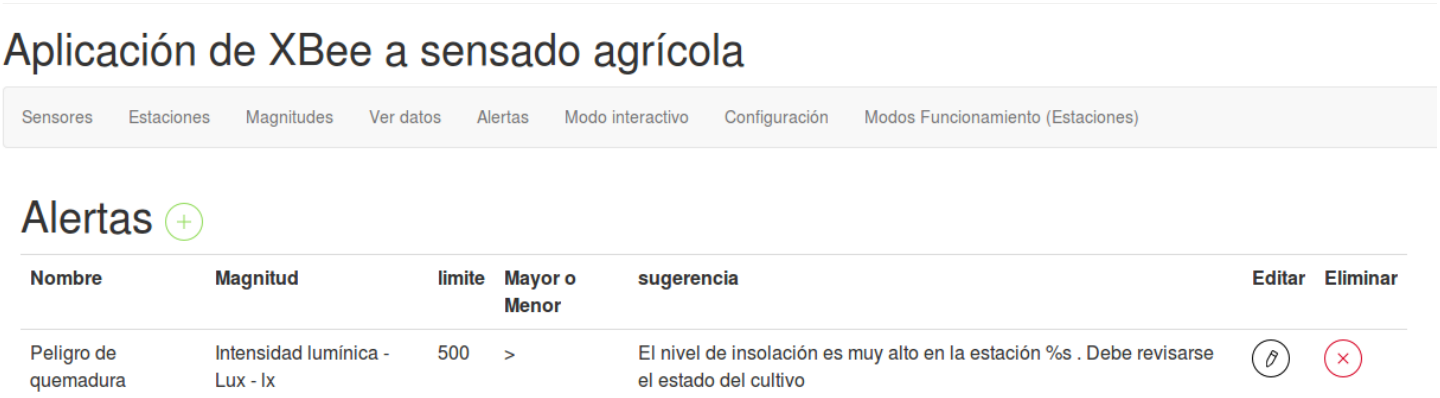

Figura 3.9: Ver Alertas

Figura 3.10: Modificar Alertas

- **Modo interactivo**:

  Esta página nos permite mandar comandos al nodo coordinador o a un nodo “hoja” (llamados aquí estaciones). Podemos seleccionar cual es el destino (coordinador o nodos

“hoja”) por medio de un selector en la parte superior. Cuando se selecciona “estaciones” muestra el nodo de origen.

Al poder estar en modo sueño, se dispone de unos selectores de la velocidad de actualización, de tal forma que con tiempos cortos las respuestas aparecerán más rápido. Se puede desactivar esta opción y actualizar la tabla por medio de un botón en la parte superior. Se muestran en diferente color los comandos enviados de los recibidos.

Figura 3.11: Modo interactivo – nodo coordinador

Figura 3.12: Modo interactivo – nodos “hoja” (estaciones)

- **Página de configuración**:

  Aquí podremos modificar los parámetros de configuración generales: correo electrónico, número de horas inactivas necesarias para que un nodo aparezca en nivel alerta y error, y por último el valor del SP del nodo coordinador, que establece cuanto tiempo guarda dicho nodo los mensajes destinados a los nodos “hoja”. Ha de ser mayor al de los nodos “hoja” para garantizar el envío correcto de los paquetes. [6]

Figura 3.13: Configuración

- **Modos de funcionamiento de las estaciones**:

  Esta página configura los modos de funcionamiento de las estaciones. Los datos están relacionados con el tiempo que un nodo permanece dormido (SP), despierto(ST), si está activo el modo de sueño (SM) y el tiempo de muestreo cíclico (IR). Se recomienda encarecidamente usar los modos preestablecidos.

Figura 3.14: Ver modos de funcionamiento de las estaciones

Figura 3.15: Editar un modo de funcionamiento de las estaciones

3.5.4 Base de datos

La base de datos está ubicada en el servidor central, en un servidor de MariaDB. Se hace uso de el motor de bases de datos InnoDB, lo que nos permitirá tener en cuenta la integridad estructural.

Pasamos ahora a describir el uso de cada tabla:

- **nodos**: Guarda información referente a cada nodo XBee "hoja", nombre, fecha de última conexión, opciones de sueño cíclico, etc.
  - Foreign Key a **modosnodo**
- **modosnodo**: Contiene los diferentes modos de funcionamiento que puede tener un nodo: Tiempo dormido, tiempo despierto, activar modo de sueño y muestreo cíclico. Está pensada para usar los valores por defecto, no para ser modificada por el usuario.
- **magnitudes**: Guarda las magnitudes físicas que medimos (humedad, temperatura, etc.), sus unidades y abreviatura.
- **sensores**: En esta tabla guardaremos los sensores que vamos a usar. Guardaremos el nombre, magnitud que mide y la

formula que transforma una lectura de voltaje del sensor en la magnitud asociada(cadena de texto, se sustituye V por el voltaje. Acepta funciones lambda).

  - Foreign Key a **magnitudes**

- **lecturas**: Cada vez que se recibe una lectura de los sensores, se añade una entrada en esta tabla. Guarda el nodo, la fecha y el voltaje que alimenta al nodo relacionado.

  - Foreign Key a **nodos**

- **lecturas_magnitudes**: Almacena los valores de cada sensor en una lectura, y a que magnitud se refiere.

  - Foreign Key a **lecturas** y a **magnitudes**.

- **nodos_sensores**: Guarda los sensores conectados a cada nodo, y en que pin está conectado (en formato **AD**X, siendo X la entrada digital a la que está conectado el sensor)

  - Foreign Key a **nodos** y a **sensores**

- **comandosremotos**: Histórico de los comandos enviados a nodos XBee “hoja”. Almacena el comando enviado, el nodo relacionado, si es un comando enviado o una respuesta y si es una respuesta, el valor de ésta, tanto en texto (cómo se mostraría en pantalla) como el valor numérico o en string devuelto, según corresponda a la instrucción.

  - Foreign Key a **nodos**

- **comandoslocales**: Análoga a la tabla anterior, pero almacenando los comandos enviados al nodo coordinador.

- **alertas**: Almacena los límites en una magnitud que generan una alerta, y el mensaje que se envía con ella.

  - Foreign key a **magnitudes**

- **configuracion**: almacena opciones de configuración generales: valor de SP en el nodo coordinador (tiempo que almacena los paquetes por si un nodo “hoja” hijo está en modo sueño) [6], la dirección de correo al que se enviarán las alertas, horas inactivas para un nodo necesarias para que se consideran un nivel de atención y de error.

  En la siguiente imagen podemos ver de forma gráfica las relaciones:

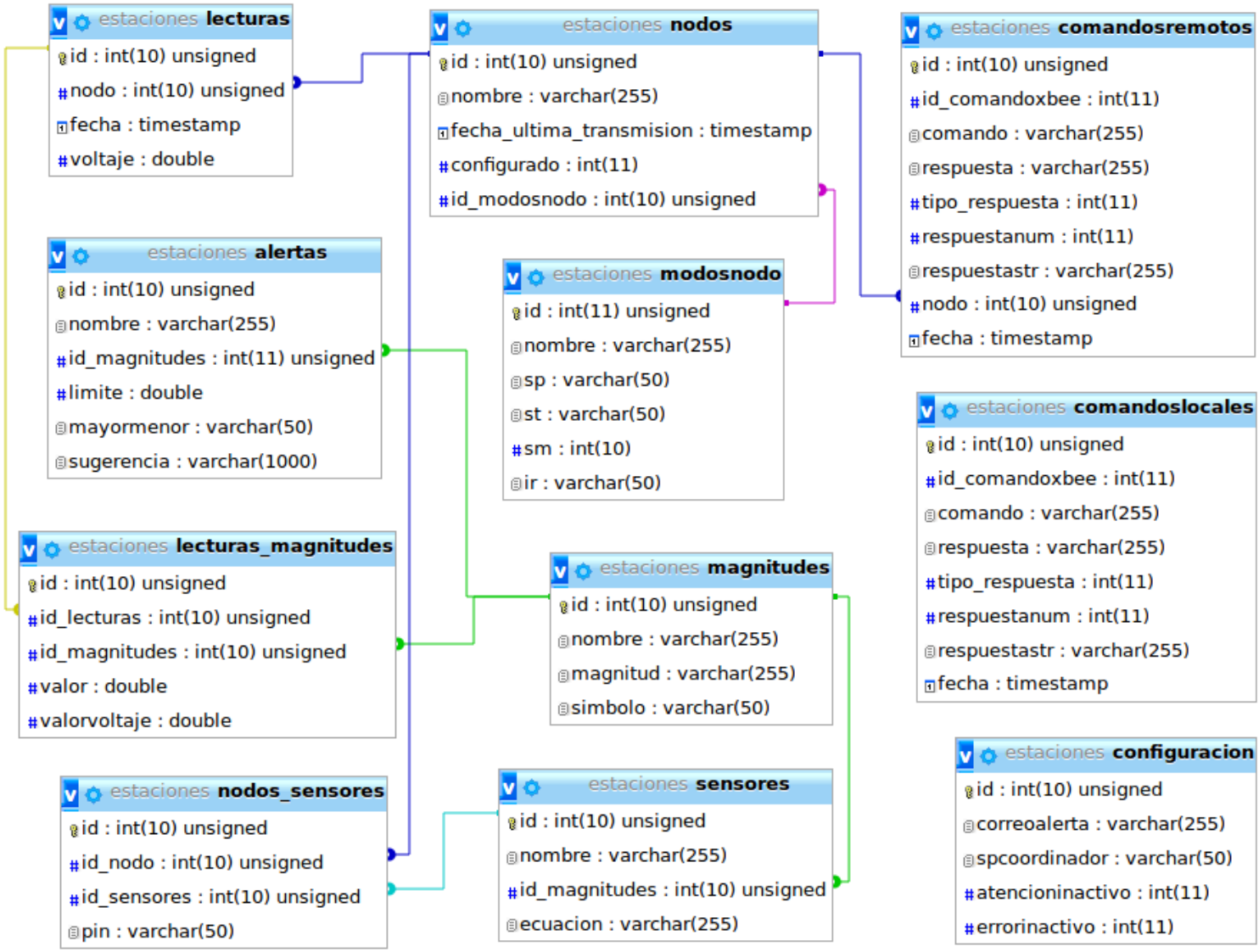

Figura 3.16: BBDD: Tablas y relaciones

Capítulo 4
Puesta en marcha

4.1 Red XBee

Para la configuración de los módulos XBee se hace uso del programa de configuración suministrado por Digi, XCTU.

Se han de grabar los firmware adecuados en cada XBee: uno ha de ser coordinador, el resto, nodos “hoja”.

Una vez hecho esto, se ha de configurar el mismo Identificador de Red y el nombre de nodo (en los nodos hoja) a través del programa modificando los parámetros ID y NI respectivamente.

Cómo paso final los montaremos en el soporte junto con los sensores escogidos y conectaremos el nodo coordinador a la Raspberry Pi mediante un cable USB.

4.2 Servidor Auxiliar

Antes de poner en marcha el servidor hay que modificar un par de ficheros. La forma más cómoda es acceder a la tarjeta de memoria en la que se encuentra el sistema y dejarlo correctamente configurado antes de llevarlo a su ubicación definitiva. De esta forma, una vez se conecte a la corriente eléctrica y al repetidor wifi ya estará disponible para conectarse por medio de ssh.

Hay que configurar la ip en la que se encuentra (ha de ser fija) por medio del fichero “/etc/network/interfaces”. Apuntaremos dicha dirección para introducirla posteriormente en la configuración del servidor central.

Se debe revisar después que la dirección del servidor central que se usa en la clase que gestiona los clientes Webservice (InterfazComunicacion.py) sea correcta.

Una vez tengamos el servidor central en marcha, nos conectaremos por ssh y arrancaremos la utilidad de control.

4.3 Servidor Central

En el servidor central disponemos de un script de instalación en php. Debemos editar el fichero **configuracion.php** con los datos de la base de datos y ejecutar **gestion/setup.php** Éste script nos preguntará por el usuario y contraseña del servidor de base de datos y realizará las siguientes tareas:

- Almacenamiento de los datos de conexión a la base de datos
- Creación de la base de datos
- Añadir en la base de datos las entradas para los valores por defecto de:
  - Configuración del nodo coordinador:
    - SP con valor máximo: 0xAF0 (30 segundos)
  - Magnitudes:
    - Temperatura (Ambiental), grados centígrados ºC
    - Humedad relativa, porcentaje %
    - Humedad del suelo, porcentaje %
    - Intensidad lumínica, Lux lx
  - Sensores, junto con su función característica
    - AMT1001 Humedad
    - AMT1001 Temperatura
    - YL-69 Humedad del suelo
    - LDR Intensidad lumínica
- Enviar comando de descubrimiento de nodos
- Asociar a cada nodo la configuración estándar de sensores
  - En el orden de los sensores, conectados de AD0 a AD3

Capítulo 5
Conclusiones y líneas futuras

5.1 Conclusiones

- Se ha seleccionado varios sensores que permitieran medir las magnitudes útiles para el cultivo y que pudieran ser utilizados fácilmente en los módulos XBee.
- Se ha seleccionado el conjunto de comandos AT necesarios para configurar los módulos XBee y realizar las lecturas de los sensores.
- Se ha diseñado un sistema de comunicación mediante webservices para la comunicación entre el servidor principal y el servidor auxiliar de los comandos y las lecturas de los sensores.
- Se ha configurado y programado mediante Python un servidor secundario para actuar cómo intermediario entre la red XBee y el servidor Central.
- Se ha desarrollado aplicación web en PHP para la gestión del sistema, la comunicación webservice y la base de datos.

5.2 Líneas futuras

- **Mejoras en la frecuencia de muestreo**:

En la solución actual se envían muestras cada 28 segundos cómo máximo, ya que estas se realizan vía la instrucción IR (lectura cíclica)y durmiendo el módulo. Para mejorar el consumo y reducir el volumen de datos, se podría hacer uso de la aplicación CRON y un cgi de php para realizar lecturas a horas específicas o con ciclos más largos vía la instrucción IS (que realiza una lectura)

- **Configuración de las gráficas**

Se podría modificar la generación de gráficas para que no hagan

uso de todas las muestras almacenadas. En vez de eso se podría realizar la media por horas, minutos, etc. según desee el usuario

- **Modo de configuración de los sensores mejorado**

Se podría mejorar la gestión de los sensores permitiendo cambiar las calibraciones individuales.

También se podría mejorar la introducción de la ecuación característica del sensor añadiendo un modo gráfico que la dedujera a partir de la interpolación de diversos puntos introducidos por el usuario. Dicha información es más fácil de obtener a través de los datasheets de los sensores

- **Tiempos de sueño más largo en los nodos “hoja”**

Se podría configurar tiempos de sueño más largos mediante el comando SN para que duerma por un periodo de SP * SN (se describe en el primer apéndice) y haciendo uso del comando SO (“Sleep Options”, Opciones de sueño) con el valor adecuado (SO4). Si se lleva a cabo, hay que tener en cuenta que los comandos enviados a los nodos “hoja” se pueden perder. [6]

Para solventar este posible problema se puede envíar un comando hasta que este sea aceptado por el nodo, o ver si ha respondido en un periodo de SP centésimas de segundo, y en caso de que no llegue la respuesta, volver a enviarla.

Capítulo 6
Summary and Conclusions

- It has been selected several sensors which allow measure the magnitudes useful for the crop and that could be easily used in the XBee modules.
- It has been selected the set of AT commands needed to configure and perform XBee sensor readings.
- It has been designed a communication system using web services for communication between the server and the axillary server of commands and sensor readings.
- It has been configured and programmed using Python a secondary server to act as an intermediary between the XBee network and the central server.
- It has been developed web application in PHP for system, webservice communication and database management.

Capítulo 7
Presupuesto

7.1 Presupuesto

Referencia:

https://www.amazon.es

Concepto	Cantidad	Precio unitario	Total item
Módulos XBee	3	35 €	105€
Raspberry Pi	1	50€	50€
Servidor Principal	1	300€	300€
Repetidor Wifi	2	35 €	70 €
Antena Yagi 18dBi	2	45 €	90 €
Horas de programación	160 Horas	10 €/h	1600 €
Horas de set-up	10 Horas	10 €/h	100 €
Total			2315 €

Tabla 7.1: Presupuesto

Bibliografía

1. http://www.homeworkmarket.com/sites/default/files/q5/04/07/danainfo.acppwiszgmk2n0u279qu76contentserver.pdf
2. http://www.elandroidelibre.com/2015/08/todo-sobre-zigbee-la-tecnologia-ultrabarata-para-comunicacion-inalambrica.html
3. http://www.ikerlan.es/es/que-investigamos/sistemas-embebidos
4. http://comohacer.eu/comparativa-y-analisis-raspberry-pi-vs-competencia/
5. https://msaffirio.wordpress.com/2006/02/05/%C2%BFque-son-los-web-services/
6. Product Manual v1.x.4x - ZigBee Protocol https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-2.5-Manual.pdf
7. http://xbee.cl/que-es-xbee/
8. https://www.cs.tut.fi/kurssit/TLT-6556/Slides/4-802.15ZigBee.pdf
9. http://www.science.smith.edu/~jcardell/Courses/EGR328/Readings/XbeeCodeDevel.pdf
10. http://www.semanticwebbuilder.org.mx/es_mx/swb/Sistemas_Embebidos_Innovando_hacia_los_Sistemas_Inteligentes_
11. https://wiki.python.org/moin/WebFrameworks
12. https://wiki.python.org/moin/WebServices
13. http://stackoverflow.com/questions/206154/what-soap-client-libraries-exist-for-python-and-where-is-the-documentation-for