La planta que controla tu red WIFI, te habla, avisa por teléfono y enseña Morse

WIFIPlant

Proyecto fin de Máster Universitario en Software Libre
Universidad Oberta de Catalunya (UOC)

Alumno: Pedro Cadavid Martínez
Tutor UOC: Pierre Bourdin
Tutor externo: Gorka Gorrotxategi Zurimendi

GRACIAS

A mi familia, *me habéis dado el tiempo y la paciencia.*

A los tutores de la UOC, *me habéis ido marcando el camino.*

Al equipo de Irontec, *me habéis hecho uno de los vuestros.*

Abstract

We live in a connected world. The revolution that the informaton technologies are experiencing during the last years, gives us every day new ways and opportunities.

"PC" or " personnel computer " was one of the origins of everything. Thanks to them, great part of our daily tasks were facilitated when we got massively in our homes. Before that, they already had been implanted in many factories.

The arrival and great diffusion of these devices was the origin of other needs, such as the interconnection between them and team working. Here is where the networks and their maximum expression were born with Internet. An open window to knowledge powered by the interaction between persons and machines.

Networks have been evolving as well as the devices with which we accede to them. We started with PC. Atfer that, the portable PC appeared and then any kind of mobile device. So we can wonder the following question: where is the following jump?

Maybe we have the answer to this question (probably in a unconscious way). We are living a transition from connecting to networks using devices to another fact where we conect devices to networks in order to interact with them. Concepts like "IOT" or "Internet of things" are not extrange for us. We are leading what is known as Industrial Revolution 4.0 where we tend to digitize all the productive processes independently of the sector.

By now, we have mentioned the importance of the technological revolution we are living but not its nature: an open nature. No doubt, this incredible diffusion and evolution has been made possible largely because of the technologies that sustain them, are open. In fact, a philosophy of work that has been used by hackers and Free Software devolopers for years that have been discovered as a way of progress and evolution. A collaborative evolution centred on persons and not in privileged groups.

This project tries to reach these technologies focusing them close to environmental values and open technologies as model to progress. This is the way WIFIPlant is born, an integrated system capable to interactuate with physical and chemical parametres of a plant in order to manage a WIFI access point.

Abstracto

Vivimos en un mundo conectado. La revolución que están experimentando las tecnologías de la información durante los últimos años, nos abre cada día nuevos caminos, antes impensables.

“PC” o “personal computer” fue uno de los orígenes de todo. Gracias a él comenzamos a facilitar gran parte de nuestras tareas cotidianas cuando se introdujo masivamente en nuestros domicilios. Previamente, ya se había implantado en muchos centros de trabajo.

La llegada y gran difusión de estos dispositivos, comenzó a dar origen a otras necesidades, como la interconexión entre ellos y/o que pudiesen trabajar en equipo. Aquí nacieron las redes y su máxima expresión con Internet. La ventana abierta al conocimiento mediante la interacción entre personas y máquinas.

Las redes han ido evolucionando así como los dispositivos con los que accedemos a ellas. Pasamos de usar el PC, a el PC portátil y a partir de ahí todo tipo de dispositivos móviles. Podemos pues, hacernos la siguiente pregunta: ¿cuál es el siguiente salto?

Aunque puede que en gran parte de manera inconsciente, la respuesta a esta pregunta la estamos dando nosotros en este momento. Estamos en plena transición de conectarnos a la red mediante dispositivos a conectar dispositivos a la red para interactuar con ellos. Conceptos como IOT o “Internet of things” no se nos antojan extraños. Protagonizamos pues, el inicio de lo que ya se ha bautizado como la Revolución Industrial 4.0, donde tendemos a digitalizar todos los procesos productivos independientemente del sector.

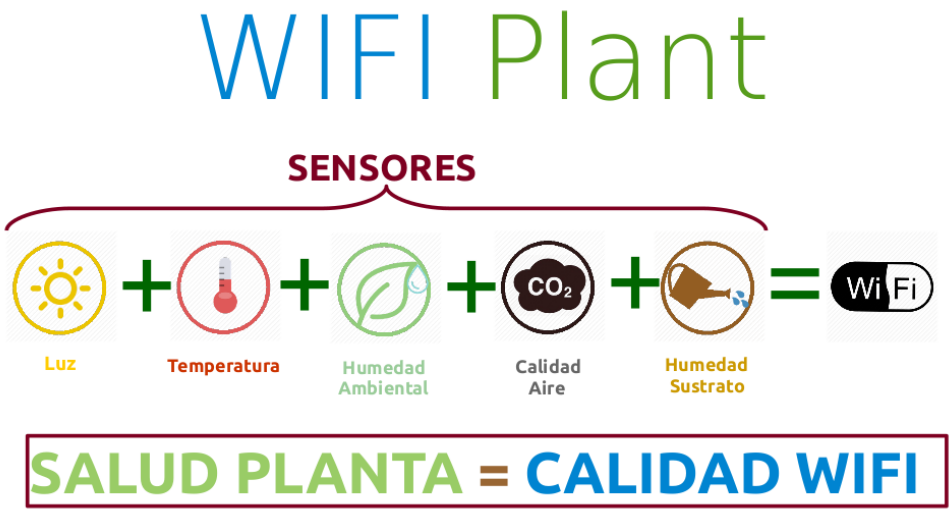
Hasta este momento, hemos mencionado la importancia de la revolución tecnológica que vivimos pero no tanto de su naturaleza: una naturaleza abierta. Sin duda, ésta rápida difusión y evolución es en gran parte posibilitada porque las tecnologías que la sustentan son abiertas. Precisamente, una filosofía de trabajo por la que apostaron hace años un grupo de hackers y desarrolladores de Software Libre y hoy reveladas como camino de progreso y evolución. Una evolución colaborativa y centrada en las personas y no en grupos privilegiados.

El presente proyecto pretende realizar una aproximación a estas tecnologías ambientándolo entorno a la difusión de valores ambientales y la educación en tecnologías abiertas como modo de progreso. Así nace WIFIPlant, un sistema integrado capaz de medir e interactuar con los parámetros fisico-químicos de un ser vivo para condicionar la calidad de un punto de acceso WIFI.

Índice

1.1 ¿Qué es WIFIPlant?

WIFIPlant es un proyecto con vocación de difundir valores ambientales desde el respeto y cuidado de un ser vivo (una planta) mediante la monitorización de sus necesidades fisico-químicas (sensores) y condicionando la calidad de un punto de acceso WIFI. Así, podría ser objeto de uso en campañas de sensibilización entorno a esta materia, centros de divulgación ambiental, etc.



1.2 Definición del proyecto WIFIPlant y objetivos

WIFIPlant - *"La planta que controla tu red WIFI, te habla, avisa por teléfono y enseña Morse"* apuesta por una integración de los contenidos impartidos en el Máster en Software Libre de la UOC entorno a diferentes tecnologías abiertas. Así, encontramos un punto de acceso WIFI con usuarios gestionados mediante un servidor radius (Freeradius), una aplicación en Bash para configurarlo y gestionarlo así como una aplicación web de monitorización.

El objetivo es condicionar la calidad del punto de acceso WIFI a la salud de una planta. Para determinar dicha salud, se plantea la ponderación de la medida de 5 sensores (Temperatura y humedad ambiental, humedad del sustrato, luz y calidad del aire) conectados a la misma (mediante una Raspberry Pi) y partiendo de unos valores de referencia en una BBDD (diferentes especies tienen diferentes necesidades fisico-químicas).

Aparte de condicionar la calidad del punto de acceso, la planta informa sobre su estado de salud mediante la implantación de una pantalla LCD que muestra las mediciones de los sensores. Del mismo modo, un altavoz conectado a la Raspberry Pi puede ayudar a que la planta envíe mensajes hablados. Por su parte, una

instalación de Asterisk, pretende ayudar a que la planta pueda notificar alertas mediante llamadas telefónicas.

WIFIPlant pretende aglutinar diferentes tecnologías abiertas tanto software como hardware entorno a un proyecto sólido orientado a difusión de éstas tecnologías y con un marcado carácter educativo sobre la base del cuidado de un ser vivo y sus necesidades. Así, la calidad de un punto de acceso WIFI estaría condicionado por la buena salud de una planta, la cuál, "decidirá" en función de su estado de salud si el punto de acceso funcionará mejor o peor.

Para ello, el proyecto requiere la realización de diferentes tareas u objetivos y su interrelación entre ellos. Así podemos destacar:

- Instalación y configuración de una versión servidor Raspbian en una Raspberry Pi

- Configuración de un servidor Freeradius en la Raspberry Pi

- Configuración de un software punto de acceso y autentificación via FreeRadius (hostapd)

- Implantación de 5 sensores ((temperatura y humedad ambiental, luz, humedad del sustrato y nivel de CO), así como programación de las lecturas de las mismas.

- Implantación de una BBDD para gestionar los requerimientos de diferentes especies de plantas y las lecturas de los sensores, alertas, etc.

- Desarrollo de un panel PHP para monitorizar el estado de la planta (lectura actual de los sensores, últimas alertas, etc).

- Configuración del servidor mediante scripts que gestionen la lectura de sensores de forma automática una vez arrancado el sistema y realizen la lectura cada X segundos.

- Desarrollo de una aplicación en Bash que permita:

  1. Añadir/borrar usuarios al freeradius
  2. Configurar la red (LAN y WLAN del la Raspberry Pi)
  3. Configurar el punto de acceso (SSID, canal de emisión, etc)
  4. Monitorizar en tiempo real los equipos conectados al punto de acceso (mostrando usuario freeradius, IP y MAC del equipo conectado)

- Configuración de Asterisk en la Raspberry Pi y configuración de un trunk SIP para que la planta pueda realizar llamadas de alerta a los números previamente configurados.

- Implantación de una pantalla LCD para que podamos leer los datos de los sensores en tiempo real.

- Desarrollo de un sistema de audio para que la planta pueda enviarnos alertas mediante mensajes de voz.

- Añadir botones para seleccionar el idioma, activar o desactivar el audio y apagar el sistema

1.3 Definición de tareas y tecnologías libres empleadas

En la **tabla 1**, podemos encontrar los diferentes de bloque de tares de las que consta el proyecto.

TAREA	SOFTWARE NECESARIO	DIFICULTAD
Configurar un punto de Acceso con servidor Radius	GNU/Linux Hostapd Freeradius Isc-dhcpd-server Bash	Baja-Media
Diseño imagen corporativa	GIMP	Baja
Script de gestión usuarios, punto acceso y red	Bash	Media-alta
Plataforma para realización de llamadas por la planta	Asterisk	Media-alta
Desarrollo de panel de control WEB	PHP	Media
Programación pantalla LCD con mensajes y colores	Bash	Media
Implantar sensores en planta	Bash Python	Media-alta
Implantar BBDD plantas con necesidades quimicas	MySQL	Media
Almacenar lecturas sensores en BBDD	Bash MySQL	Media-alta
Acciones en función de condificones ambientales	Bash MySQL	Media-alta

Tabla 1

1.4 Análisis de requisitos

La viabilidad de un proyecto TI está condicionado por numerosos factores. Para simplificar su identificación y poder minimizarlos establecemos la primera y más evidente diferenciación: los factores técnicos y los económicos. Aunque ambos están interelacionados, el incumplimiento de uno de ellos, implica la invialidad de un proyecto.

Desde el punto de vista técnico, un proyecto está influenciado por aspectos como la diversidad de tecnologías que incluye el mismo, su dificultad de implantación o la necesidad de recursos humanos necesario para ejecutarlo. En este sentido, no podemos olvidarnos que la mayor parte de los proyectos informáticos tienen una doble componente: software por un lado y hardware por otro. Los requerimientos de éste último grupo, condicionan en muchas ocasiones la viavilidad del proyecto.

Por su lado, una vez "superado" el condicionamiento técnico y por tanto, cuando el proyecto es viable desde este punto de vista, necesitamos atender a los recursos económicos necesarios para atender esa componente técnica.

La concepción de un proyecto "integrador" que incluya diferentes aspectos relacionados con la materia de Máster Universitario en Software Libre, requiere

el empleo de numerosas tecnologías de diferente naturaleza. En este sentido, destacamos en desarrollo de software en lenguajes Bash y PHP, la gestión del código fuente del fabricante de sensores en lenguaje Python, la configuración de diferentes servicios como red, DHCP y gestión de procesos o la propia adaptación de una distribución GNU/Linux para una Raspberry Pi de la forma más minimalista posible, hacen necesaria una correcta organización y planificación integrada de los mismos.

Siguiendo con el análisis de los requisitos técnicos de WIFIPlant, tenemos que analizar la destacada presencia de componentes hardware. A la mencionada Raspberry Pi, añadimos la placa para interconectar sensores analógicos y digitales, botones, la pantalla, así como otros elementos como el adaptador WIFI o el altavoz.

Dispositivo	Precio
Rasberry PI 2	41.99
GrovePI	45.75
Sensor DHT22 (temp+hum)	13.45
Sensor humedad sustrato	7.25
Sensor luz	4.25
Botones x 3	8.25
Sensor calidad aire	17.5
Pantalla LCD	19.75
Altavoz X-MiniE	19.90
Tiesto bamboo Vildapel	5.99
Adaptador WIFI nanoUSB	8.95
TOTAL	**193.03**

Tabla2

Una vez superada una descripción de los requisitos técnicos que nos valida la viabilidad de WIFIPlant, podemos analizar el condicionante económico. Si consideramos las necesidades de desarrollo planteadas, este aspecto no es significativo. Por su parte, los elementos hardware acaparan la mayor parte del mismo. La **tabla 2,** muestra cada uno de ellos así como su importe aproximado.

1.5 Análisis de viabilidad

El análisis de requisitos simplificado que hemos planteado en el punto anterior mediante un división en su vertiente técnica y económica nos ha revelado las necesidades que como proyecto plantea WIFIPlant.

En cuanto a las primeras, destaca la diversidad de tecnologías que participan para llevarlo a cabo y por consiguiente, la necesidad de un cierto conocimiento/investigación sobre las mismas. En éste sentido, aunque la cantidad de tareas a desarrollar es numerosa, no se trata, salvo algún caso puntual, de configuraciones especialmente complejas o sobre los que no exista una destacable cantidad de fuentes documentales. Por esta razón, en el plano técnico lo consideramos viable.

En lo correspondiente a la parte económica, como hemos mostrado en la **tabla 2** con los gastos de hardware el proyecto implica un inversión en material entorno a 200€. Aunque las cífras económicas pueden esconder matices entorno a la naturaleza del proyecto (atractivo, beneficio, interés, etc) desde el punto de

vista económico, se trata de un proyecto viable, sobretodo considerando la magnitud del mismo.

Llegados a éste punto, tanto el análisis de viabilidad económico como técnicos resultan favorables, por lo que el proyecto en su conjunto es viable. Ahora bien, es importante destacar que en el plano económico, no se ha considerado el potencial impacto que tendrían las horas de trabajo de la(s) personas implicadas en el desarrollo del mismo. No obstante, entendemos interesante realizar una aproximación a partir de la programación horaria presentada en el punto "Definición detallada de tareas y subtareas y programación horaria". Así, la dimensión del proyecto enfocada como una herramienta de márketing tecnológico y que pretende aglutinar la mayor parte de las áreas en las que presta servicio la empresa, se estima que dicho impacto es mínimo en relación con su fortaleza.

1.6 Retos y dificultades del proyecto

Una definición de proyecto como el expuesto, tiene la evidente ventaja del atractivo conjunto, en el que un ser vivo puede comunicarse con nosotros e interactuar en función de su estado de salud, mejorarando o empeorando la calidad de nuestra red WIFI.

Sin embargo, la correspondiente necesidad de tanta variedad de tecnologías así como su intetrelación entre ellas, ha supuesto el principal reto a superar. Para minimizar el potencial impacto de una tarea no superada en el conjunto, se ha apostado por definir un "cortafuegos" entre las tareas que lo componen, separando el núcleo del mismo del resto de tareas, que pasan a ocupar un papel relevante, pero no determinante. Así, una tarea no superada puede tener como impacto un proyecto menos atractivo, pero no así un proyecto incompleto.

Una manera sencilla aplicar las premisas formuladas, consiste en realizar una asignación de prioridades y eliminar la interdependencia entre tareas, más allá de las ineludibles. En la **tabla 3**, podemos ver las diferentes tareas con su correspondiente prioridad asignada. Como es de esperar, el núcleo del proyecto consiste en implementar un punto de acceso WIFI con autenticación radius así como la implantación de sensores en una planta. Así, el resto de tareas vertebran el proyecto.

Tarea	Prioridad
Configurar un punto de Acceso con servidor Radius	Alta
Programación script control Usuarios, WIFI y red	Alta
Diseño BBDD control parámetos planta y estaciones	Alta
Implantación de sensores Temp, Hum, Soil ,Lux, CO2	Alta
Sistema control remoto openvpn	Baja
Añadir criterios accesibilidad en aplicación WEB	Baja
Programación sistema comunicación Asterisk	Baja
Diseño imagen corporativa	Baja
Programación web sistema control	Media
Integración completa, puesta en marcha y pruebas	Media

Tabla 3

Una definición más detallada de las prioridades establecidas se realizó en la PEC2, en la que mediante tres categorías se consigue medir el impacto que implicaría la no realización de una tarea en la integridad del proyecto.
Otro punto importante a superar en la realización del proyecto es el trabajo en conjunto en el seno de una empresa. En este sentido, se cuenta con una importante ventaja ya que se comparte espacio con numerosos profesionales que pueden ser de ayuda en la implementación de las diferentes tareas. En este sentido, el volumen de tareas de WIFIPlant, le otorga un mayor potencial.

Como contrapartida, es necesario considerar que los ritmos del proyecto tienen que adecuarse a la realidad y necesidades de la propia empresa, de forma que una tarea puede ser demorada o condicionada hasta que los recursos para llevarla a cabo estén disponibles (Ej. un servidor web, material, etc). Llegados a este punto, nos encontramos ante una segunda escala de prioridades. Por un lado, las tareas de nuestro proyecto y por otro, las tareas de la empresa. De ésta manera, debe considerarse la integración de las primeras en las segundas.

Una forma de minimizar las necesidades externas entorno a la realización del proyecto ha sido lograda abriendo simultáneamente la realización de varias tareas. Así, una parada temporal en una, no implica que el proyecto no avance, ya que se puede dedicar recursos a una segunda tarea no condicionada. De la misma manera, un elemento que ha sido de gran utilidad en este sentido, ha sido la virtualización. Así con varias máquinas Virtualbox, se ha podido simular el funcionamiento los diferentes servicios implementados sin la necesidad de recurrir a su aplicación directa sobre el hardware. Por consiguiente, en muchos casos, la implantación sobre hardware podemos etiquetarlo como un “paso a producción”.

1.6.1 Medidas de control, prevención y correctoras

Una vez comprobadas las potenciales debilidades derivadas de los compromisos asumidos en el proyecto, se plantea realizar una estrategia de actuación. En éste sentido, se apuesta realizar una categorización de 3 categorías a cada una de las tareas que componen el proyecto en función de su prioridad. Así, definimos los siguientes bloques:

Prioridad 1

Definición: Tareas de cuya realización se compromete la realización del proyecto.
Tareas incluidas: Aquellas entorno a la creación y configuración del punto de Acceso (Configurar el AP, la autencitación Freeradius, la lectura de sensores y las herramientas básicas de configuración

Prioridad 2

Definición: Tareas que no comprometiendo la integridad, se han considerado como aspectos relevantes del proyecto y que le dan una gran personalidad.
Tareas incluidas: La aplicación WEB de monitorización, la plataforma de emisión de llamadas, el acceso remoto mediante openvpn.

Prioridad 3

Definición: Tareas fuera del kernel del proyecto, concebidas para hacerlo más atractivo.
Tareas incluidas: La imagen corporativa del proyecto y el sistema de mensajes hablados por la planta mediante un altavoz interno.

1.7 Programación de tareas

La **tabla 4** muestra un gráfico de Gant con la programación temporal de las tareas previamente definidas.

Tarea	Duración	Comienzo	Fin
Configurar un punto de Acceso con servidor Radius	**11 días**	**21/09/16**	**10/10/16**
Instalar software necesario	1 día	21/09/16	21/09/16
Programación driver WIFI compatible con punto acceso	1 día	22/09/16	22/09/16
Programación script de configuración completo	7 días	23/09/16	29/09/16
Expulsar conexión a cliente en punto de acceso	1 día	30/09/16	30/09/16
Programación script control Usuarios, WIFI y red	**20 días**	**01/10/16**	**20/10/16**
Gestión usuarios Freeradius en Bash	5 días	01/10/16	05/10/16
Gestión AP WIFI HostAP en Bash	5 días	06/10/16	10/10/16
Configuración de red mediante en Bash	8 días	11/10/16	18/10/16
Diseño módulo de monitorización en Bash	2 días	19/10/16	20/10/16
Diseño BBDD control parámetos planta y estaciones	**3 días**	**21/10/16**	**23/10/16**
Diseño E-R	1 día	21/10/16	21/10/16
Diseño BBDD sobre MySQL-Server	1 día	22/10/16	22/10/16
Introducción parámetros 20 especies de plantas	1 día	23/10/16	23/10/16
Implantación de sensores Temp, Hum, Soil ,Lux, CO2	**10 días**	**24/10/16**	**02/11/16**
Instalación sensores y programación	1 días	24/10/16	24/05/16
Programación script control sensores y acciones	4 días	25/10/16	28/10/16
Control colores pantalla LCD condicionado por planta	5 días	28/10/16	02/11/16
Sistema control remoto openvpn	**2 días**	**03/11/16**	**04/11/16**
Instalación openvpn, creación certificados	1 día	03/11/16	03/11/16
Pruebas de funcionamiento, estabilidad y rendimiento	1 día	04/11/16	04/11/16
Añadir criterios accesibilidad en aplicación WEB	**8 dias**	**05/11/16**	**12/11/16**
Análisis de requisitos	2 días	05/11/16	06/11/16
Implementación de diseño	6 días	07/11/16	12/11/16
Programación sistema comunicación Asterisk	**15 días**	**13/11/16**	**27/11/16**
Compilación Asterisk en RPI	1 día	13/11/16	13/11/16
Implantación trunk SIP	10 días	14/11/16	23/11/16
Programación dialplan	4 días	24/11/16	27/11/16
Diseño imagen corporativa	**4 días**	**28/11/16**	**01/12/16**
Dieño logo específico WIFIPlant	1 días	28/11/16	28/11/16
Diseño imagen corporativa	3 días	29/11/16	01/12/16
Programación web sistema control	**10 días**	**02/12/16**	**11/12/16**
Desarrollo PHP monitorización tiempo real	5 días	02/12/16	06/12/16
Desarrollo panel de control	5 días	07/12/16	11/12/16
Integración completa, puesta en marcha y pruebas	**5 días**	**12/12/16**	**16/12/16**
Integración todos servicios	2 días	12/06/16	13/12/16
Corrección errores	2 días	14/12/16	15/12/16
Prueba funcional	1 día	16/12/16	17/12/16

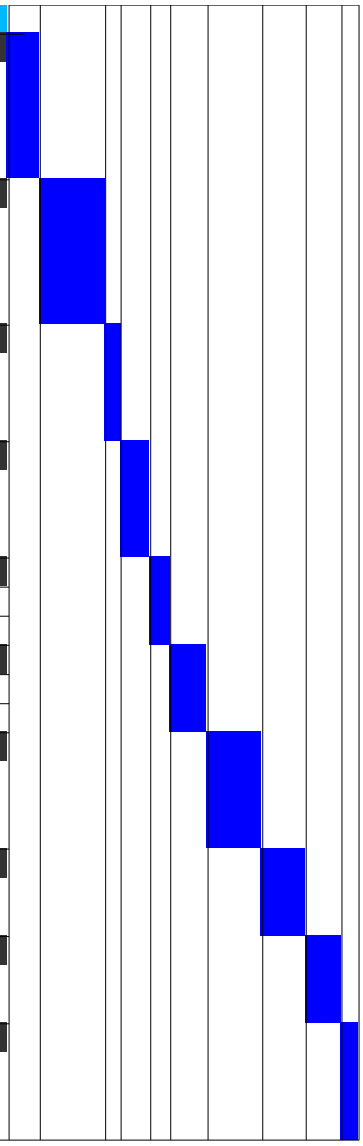


Tabla 4

1.8 Irontec - Internet y Sistemas sobre GNU/Linux

De cara a focalizar el plan de trabajo que aquí proponemos, se pretende en primer lugar, contextualizar la empresa en las que se realiza la parte práctica del proyecto. Irontec - Internet y Sistemas sobre GNU/Linux [1] es una de las principales empresas del sector TIC especializadas en Software Libre del país que lleva prestando servicios entorno a tecnologías abiertas durante la última década.

Actualmente cuenta con entorno a 30 trabajadores y entre los servicios que presta destacan VoIP, implantación de sistemas y redes, desarrollo de aplicaciones (WEB, móvil) y márketing tecnológico.

1.9 Ambientación de las tareas entorno a la actividad de la empresa

El presente proyecto pretende aglutinar la mayor parte de contenido impartido en el Máster. Para ello, se parte de los módulos obligatorios y optativos seguidos durante el estudio de las asignaturas del Máster, encontrando 3 bloques o módulos:

- Administración de sistemas sobre GNU/Linux
- Desarrollo de aplicaciones WEB
- Implantación de BBDD
- Análisis de tecnologías abiertas para desarrollo de proyectos.

Una vez definidos los bloques que formarán la columna vertebral sobre la que se sustenta el proyecto, se apuesta por integrar estos módulos entorno a la actividad de la empresa. De ésta manera, encontramos otros 3 grandes bloques:

- VoIP
- Sistemas y redes
- Desarrollo
- Márketing tecnológico

De la fusión de ambos entornos, el proyecto propuesto "WIFIPlant", apuesta por diseñar un trabajo que incluya la mayor parte de las tecnologías presentadas en los puntos anteriores (Máster+Actividad de la empresa) creando un producto con un importante atractivo como márketing tecnológico.

Como proyecto, WIFIPlant ha sido concebido entorno a una doble vertiente. Por un lado, el encaje entorno a los itinerarios cursados en el programa del Máster Universitario en Software Libre de la UOC. Por otro, su integración en la actividad de la empresa Irontec [1], especializada en ofrecer soluciones sobre tecnologías abiertas en áreas como Sistemas, VoIP, redes, marketing tecnológico, etc. La **tabla 5** muestra la relación entre los diferentes itinerarios cursados en el Máster y su correspondiente encaje entorno a la actividad de la empresa.

Itinerario del Máster	Área de trabajo en la empresa
Administración de sistemas sobre GNU/Linux	Sistemas
Desarrollo de aplicaciones WEB	Desarrollo
Implantación de BBDD	Desarrollo
Integración de tecnologías abiertas en proyectos	Investigación

Tabla 5

Como puede observarse, el planteamiento incorpora una gran variedad de tecnologías, lo cuál, haría complicado un proyecto especializado entorno a una única disciplina. Para cumplir éstos requisitos, se ha apostado por diseñar una idea "nuclear" (el punto de acceso WIFI condicionado por sensores) y entorno al mismo, incorporar otras tecnologías dentro del ámbito del Máster y de la actividad de la empresa (aplicación WEB, integración con sistemas VoIP , diseño de imagen corporativa, etc).

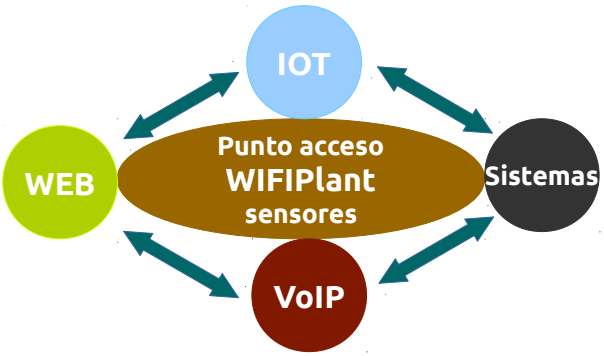


Gráfico 1

De ésta manera, conseguimos moldear un proyecto en “capas”, que tiene bien definido un objetivo principal que es escalado y enriquecido incorporando funcionalidades al mismo.

Licencias de software

2.1 Cocktail de licencias

Como cualquier proyecto TI de cierta dimensión, WIFIPlant hace uso de numerosos proyectos libres existentes. Por esta razón, se antoja aconsejable buscar una licencia con un grado de compatibilidad intraespecífica compatible con la de todos los proyectos empleados. De ésta manera, WIFIPlant podría potencialmente, ser liberado con una única licencia.

Antes de realizar una aproximación a los aspectos legales que intervienen en el proyecto, debemos considerar la doble naturaleza que lo integra: software por un lado y hardware por otro, todo enfocado sobre la base de tecnologías abiertas. En este sentido, es posible abrir un importante foro de discusión sobre la plataforma Raspberry Pi, que si bien no tiene una especificación (ni componentes) abiertos que lo definan como hardware libre, si puede entenderse como un "*camino abierto*". Argumenta ésta postura la política favorable al uso y divulgación de software libre por parte la fundación que soporta el proyecto [3]. Incluso hay quien lo etiqueta directamente como hardware libre [4].

Por su parte, posturas antagónicas a ésta, podemos encontrarlas en aquellas que exigen a los proyectos de hardware libre que sus especificaciones y componentes se liberen bajo las mismas premisas que recogen las 4 libertades del software libre [4].

Una vez dimensionado el proyecto entorno a sus aspectos legales, encontramos interesante buscar una licencia software que sea compatible de forma integral con todos y cada uno de los proyectos que lo integran. Sin embargo, por simplicidad y dada la naturaleza diferente de los diferentes subproyectos que lo componen, no se antoja menos adecuado definir una licencia para cada uno de los desarrollos aquí presentados.

Como paso previo, analizamos las tareas que componen el proyecto desde el punto de vista del software implicado y sus licencias. Así, en la **tabla 6** podemos ver cada una de las licencias implicadas así como su compatibilidad con la GPL.

TAREA	SOFTWARE IMPLICADO	Licencia	Compatibilidad GPL
Aplicación Bash gestión punto acceso	GNU/Linux	GPLv2	Si
	Hostapd	OpenBSD	Si
	Freeradius	GPLv2	Si
	Isc-dhcpd-server	ISC-License	Si
	Bash	GPLv2	Si
Script de gestión usuarios, punto acceso y red	Bash	GPLv2	Si
Plataforma llamadas telefónicas planta	Asterisk	GPLv2	Sí
Desarrollo de panel de control WEB	PHP	PHP-License	No* (solo des. internos)
Programación pantalla LCD con mensajes y colores	Bash	GPLv2	Si
Implantar sensores en planta	Bash	GPLv2	Sí
	Python	Python 2.7	Sí
	Scripts Grove	MIT	Sí
Implantar BBDD plantas con necesidades quimicas	MySQL	GPLv2*	Sí* (Sólo proyectos libres)

Tabla 6

2.2-Elección de licencias para los diferentes desarrollos

Cada un de estas tareas, da lugar a la creación de nuevo software, el cuál, tenemos que analizar desde el punto de vista no tanto de sus funcionalidades como de sus dependencias respecto a otro software.

WIFIPlant.sh

Objetivo: Aplicación Bash que permite configurar y gestionar un punto de acceso con autenticación radius.
Licencia elegida: GPLv3
Justificación: Aunque hace uso de aplicaciones como Hostapd, Freeradius o ISC-DHCP-server, no se trata de una aplicación que tenga dependencias entorno a librerías o modifique código de estas aplicaciones. Se trata de un software de configuración y gestión.

BashGPIO.sh

Objetivo: Aplicación Bash que gestiona el control a los pines GPIO
Licencia elegida: GPLv3
Justificación: En sí misma puede considerarse una librería en Bash para gestionar pines GPIO. Consta de funciones creadas a partir del acceso nativo a los pines GPIO en la Raspberry PI.

Bash_Morse.sh

Objetivo: Traducir texto a morse en forma de texto, luminoso en leds y en pantalla LCD.
Licencia elegida: GPLv3
Justificación: En este caso, tenemos que tener en cuenta que esta aplicación hace uso de "lcd_rgb.sh" proporcionada por el fabricante Grove[6] para gestionar la pantalla. Al estar esta última liberada bajo licencia MIT, podemos emplear licencia GPL.

lcd_rgb.sh

Objetivo: Controlar la pantalla LCD que mostrará la información de la planta.
Licencia elegida: MIT
Justificación: En este caso, se mantiene el código proporcionado por el fabricante [6] al que se le añaden algunas funciones para realizar más sencillo su uso. Aunque podríamos usar una licencia GPL y a pesar de nuestro aporte de código, apostamos por mantener la licencia propuesta por el fabricante.

check_buttons.sh

Objetivo: Comprobar si se ha pulsado alguna de los 3 botones (activar/desactivar audio, cambiar idioma y el de apagado del sistema).
Licencia elegida: GPLv3
Justificación: Es un desarrollo autónomo y no dependiente de software de terceros.

check_sensors.sh

Objetivos: Realiza las lecturas de los sensores en y realiza las acciones programadas en función de los resultados (mensajes hablados, alertas, llamadas telefónicas, etc)
Licencia elegida: GPLv3
Justificiación: Aunque es el motor sobre el que se sustentas las acciones de la planta, el software que emplea es compatible con ésta licencia.

check_all_checks.sh

Objetivo: Controlar que están corriendo los scripts que se encargar de chequear el estado de los botones, de la VPN y de los sensores
Licencia elegida: GPLv3
Justificación: Se trata de un simple script en Bash que comprueba el que están corriendo los procesos necesarios del sistema y de arrancarlos en su caso.

Ficheros fabricante GrovePi para lectura de los sensores

Ficheros: air_quality.py, moisture.py, temp_hum.py, light.py, audio_button.py, languaje_button.py,power_button.py
Objetivo: Realizar las lecturas de los sensores y comprobar si se han pulsado alguno de los 3 botones de la planta.
Licencia elegida: MIT
Justificación: Aunque se han realizado modificaciones y adaptaciones del código, la mayoría de ellas son cambios menores y orientados a simplificar el código al mínimo y adaptarlo a las necesidades del proyecto. Así mantenemos la licencia elegida por el fabricante.

Aplicación web panel de control

Objetivo: Controlar la lectura de datos en tiempo real y configuración
Licencia elegida: GPLv3
Justificación: Es un desarrollo en mayormente en PHP con CSS y algo de Javascript. De éste último lenguaje, hace uso de la librería justGage [7], la cuál está liberada con licencia MIT, por lo que podemos liberar nuestro código con GPL.

Logos, imagen corporativa y diseños gráficos

Objetivo: Otorgar una personalidad e identificación al proyecto.
Licencia elegida: Reconocimiento - NoComercial - SinObraDerivada (by-nc-nd)
Justificación: La imagen de cualquier proyecto es una seña de identidad del mismo y que pretende representar gráficamente su naturaleza. Así establecer un equilibrio entre el "respeto" de la "marca" y la divulgación y rl "fair use" de estos recursos, es conseguido con esta licencia.

2.3 Medición de "*copyleft*"

Ya tenemos desgranados uno a uno cada uno de los ficheros con el código fuente que componen el proyecto y la licencia elegida para cada uno de ellos. Somos por consiguiente, capaces de realizar una valoración integral del "grado" de "*copyleft*" de WIFIPlant.

Como podemos observar en la **tabla 7**, 2 licencias son las que han usado para el conjunto de desarrollos (GPLv3 y MIT), predominando aproximadamente un 90% la GPLv3 ya que como se ha mencionado, la licencia MIT se ha relegado para los desarrollos incorporados para la lectura de los sensores originales del fabricante GrovePi y manteniendo su elección.

Considerando que las licencias GPL están en el extremo con más "copyleft" y las BSD en el opuesto, la licencia MIT se encuentra en un punto más próximo a éstas últimas y por tanto sin un destacado carácter "copyleft". Sin embargo, la clara predominancia de la licencia GPLv3, otorga al conjunto del proyecto un marcado cáracter "copyleft".

Desarrollo	Licencia
WIFIPlant.sh	GPLv3
BashGPIO.sh	GPLv3
Bash_Morse.sh	GPLv3
lcd_rgb.sh	GPLv3
check_buttons.sh	GPLv3
check_sensors.sh	GPLv3
check_all_checcks.sh	GPLv3
Ficheros lectura sensores	MIT
Aplicación WEB	GPLv3
Diseño gráficos	by-nc-nd

Tabla 7

Gráfico 2

3.1 Consideraciones técnicas y estructura de la BBDD

Aunque en principio, un esquema sencillo de BBDD sería más que suficiente para realizar la monitorización de una sóla planta y una sóla estación (Raspberry PI), se plantea un esquema relacional preparado para poder monitorizar decenas de plantas, controlando las macetas donde están instadas así como la posibilidad de emplear varias Raspberry Pi y así asignar la monitorización de las plantas por macetas y equipos.

Un aspecto importante entorno a las BBDD en un hardware como la Raspberry Pi es el número de lecturas/escrituras. En este sentido, cobra especial relevancia la unidad de almacenamiento al tratarse de un dispositivo en estado sólido como una tarjeta microSD, por lo que es importante minimizar las escrituras. Así el diseño de la BBDD se centra en el control, gestión y monitorización de la planta, dejando el almacenamiento de datos únicamente circunscrito a éstas tareas de monitorización (log de acciones especialmente relevantes).

En la figura, se muestra las tablas diseñadas así como las relaciones representadas por colores.

3.2 Definición de las tablas

Tabla plantas

Contiene el nombre y descripción de las diferentes especies de plantas. Así podemos definir las diferentes necesidades fisico-químicas.

Variable	Tipo
id_pl	primary key
nombre_cie	varchar
nombre_com	varchar
altura	double
hojas	varchar
flores	varchar

Tabla 8

Tabla maceta

Identifica las macetas y describe su volumen.

Variable	Tipo
id_ma	primary key
volumen	double

Tabla 9

Tabla maceta_planta

Asocia las plantas con las macetas.

Variable	Tipo
id_ma	primary key
id_pl	primary key

Tabla 10

Tabla estaciones

Contiene los equipos Raspberry Pi empleados para monitorizar.

Variable	Tipo
id_es	primary key
descripcion	varchar

Tabla 11

Tabla estacion_maceta

Permite controlar las macetas que tiene asignada cada estación (Raspberry Pi) y por tanto las plantas que tiene cada estación.

Variable	Tipo
id_es	primary key
id_ma	primary key

Tabla 12

Tabla humedad

Controla los límites máximo y mínimo de humedad de cada planta.

Variable	Tipo
id_pl	primary key
min_hum	double
max_hum	double

Tabla 13

Tabla temperatura

Controla los límite máximo y mínimo de la humedad de cada planta.

Variable	Tipo
id_pl	primary key
min_temp	double
max_temp	double

Tabla 14

Tabla soil

Controla los límte máximo y mínimo de humedad del sustrato de cada planta.

Variable	Tipo
id_pl	primary key
min_soil	double
max_soil	double

Tabla 15

Tabla lux

Controla los límte máximo y mínimo de luz adecuada para cada planta.

Variable	Tipo
id_pl	primary key
min_lux	double
max_lux	double

Tabla 16

Tabla air

Controla los límte máximo y mínimo de calidad de aire para cada planta.

Variable	Tipo
id_pl	primary key
min_air	double
max_air	double

Tabla 17

Tabla planta_humedad

Recoge los valores leídos de humedad.

Variable	Tipo
id_hu	primary key
id_ma	primary key
id_pl	primary key
humedad	double
fecha	datetime

Tabla 18

Tabla planta_temperatura

Recoge los valores leídos de temperatura.

Variable	Tipo
id_te	primary key
id_ma	primary key
id_pl	primary key
temp	double
fecha	datetime

Tabla 19

Tabla planta_soil

Recoge los valores leídos de humedad del sustrato.

Variable	Tipo
id_so	primary key
id_ma	primary key
id_pl	primary key
soil	double
fecha	datetime

Tabla 20

Tabla planta_lux

Recoge los valores leídos de iluminación.

Variable	Tipo
id_lux	primary key
id_ma	primary key
id_pl	primary key
lux	double
fecha	datetime

Tabla 21

Tabla planta_air

Recoge los valores leídos de calidad del aire.

Variable	Tipo
id_air	primary key
id_ma	primary key
id_pl	primary key
air	double
fecha	datetime

Tabla 22

Tabla alertas

Almacena las alertas enviadas por la planta a modo de log en BBDD.

Variable	Tipo
id_ma	primary key
id_pl	primary key
mensaje	varchar
fecha	datetime

Tabla 23

Tabla usuarios

Almacena los usuarios que tendrán acceso al panel de control web. Por seguridad el campo “clave” se almacenará con el valor MD5 pasado en PHP.

Variable	Tipo
id_usu	primary key
usuario	varchar
nombre	varchar
apellido1	varchar
apellido2	varchar
clave	varchar

Tabla 24

Esquema general

En la tabla 25 de la siguiente página, podemos ver un esquema general con las diferentes tablas que componen la base de datos.

Tabla plantas					
id_pl	nombre_cie	nombre_com	altura	hojas	flores

Tabla maceta	
id_ma	volumen

Tabla maceta_planta	
id_ma	id_pl

Tabla estaciones	
id_es	descripcion

Tabla estacion_maceta	
id_es	id_ma

Tabla humedad		
id_pl	min_hum	max_hum

Tabla temperatura		
id_pl	min_temp	max_temp

Tabla soil		
id_pl	min_soil	max_soil

Tabla lux		
id_pl	min_lux	max_lux

Tabla air		
id_pl	min_air	max_air

Tabla planta_humedad				
id_hu	id_ma	id_pl	humedad	fecha

Tabla planta_temp				
id_lu	id_ma	id_pl	temp	fecha

Tabla planta_soil				
id_lu	id_ma	id_pl	soil	fecha

Tabla planta_lux				
id_lu	id_ma	id_pl	lux	fecha

Tabla planta_air				
id_lu	id_ma	id_pl	air	fecha

Tabla alertas				
id_al	id_ma	id_pl	mensaje	fecha

Tabla 25

4.1 Diseño integral

WIFIPlant es en sí un conglomerado de subproyectos, donde confluyen diferentes tecnologías. Ésta singularidad establece la necesidad de implementar un enfoque integral que defina los vasos comunicamentes entre cada uno de ellos. Así se apuesta por incluir aspectos de diseño tanto para el conjunto del proyecto, como para los elementos software.

4.2 Esquema general

El requisito de unificar una gran cantidad de elementos hardware (sensores,botones, altavoz, placas de electrónica, cableado,etc) nos obliga a establecer un diseño de cómo queremos que sea nuestro proyecto y cómo será la ubicación de cada uno de estos elementos. El **gráfico 1**, nos muestra la parte física de cómo será el diseño del tiesto que albergará la planta y la ubicación de cada uno de los elementos electrónicos a instalar.

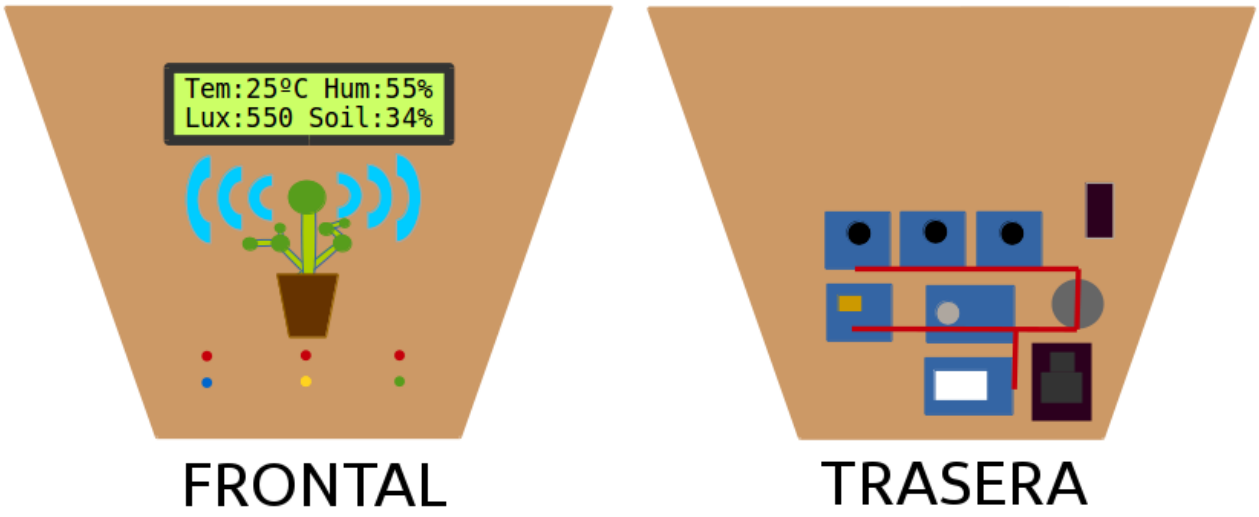
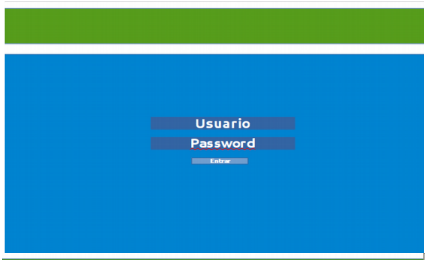


Gráfico 3

4.3 Diseño y prototipado: Aplicación WEB

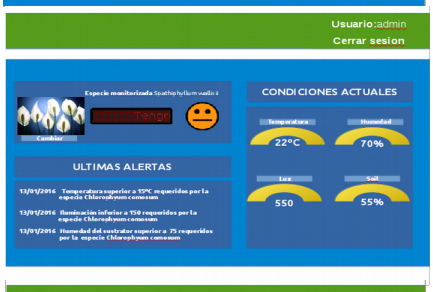
El panel de control WEB nos permite visualizar gráficamente el estado de la planta. Para ello, es necesario establecer una ordenación y colocación de los elementos que permita transmitir la información proporcionada por los sensores. En este sentido, apostamos por realizar un prototipo de la aplicación WEB que nos permita simular su funcionamiento. En nuestro caso, adjunto a este proyecto se incluye el prototipo “PrototipoWEB.odp” que se trata de una presentación LibreOffice que permite navegar a través de los distintos elementos de la aplicación. En las siguientes ilustraciones podemos ver unas capturas básicas con el prototipo de la aplicación web. El prototipo completo puede consultarse en el documento anexo a este proyecto “PrototipoWEB.odp”.

Página 1-Página login



Página 2-Principal
Especie 1

Página 2-Principal
Especie 2



Página 3-Selección de especie

5.1 Raspberry Pi

Aunque el último modelo de Raspberry PI (versión 3) trae como principal novedad la inclusión integrada de WIFI y bluetooth, WIFIPlant ha sido implementada sobre la versión anterior (Versión 2) cuyas características principales podemos visualizar en la **tabla 20** y **gráfico 2.** Un esquema electrónico de esta placa está disponible en los **anexos**.

Hardware	Tipo
CPU	ARM Cortex-A7 900 Mhz
RAM	1 GB
Puerto USB	4
Gráficos	HDMI
Audio	Mini jack
Ethernet	1

Tabla 26

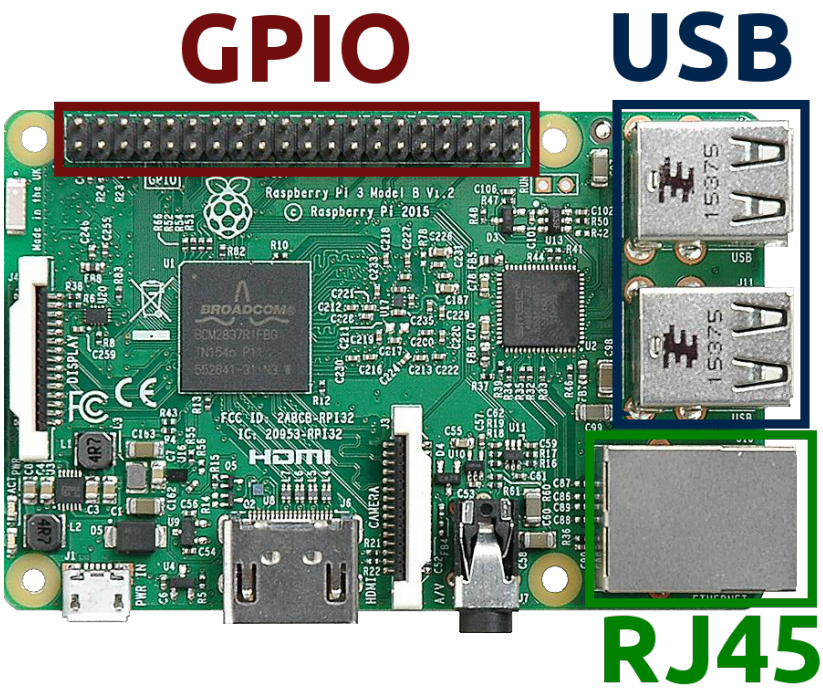


Gráfico 4

5.2 GrovePI

Se trata de una placa de expansión que se conecta sobre parte de los GPIO de las Raspberry PI permitiendo conectar fácilmente sensores tanto digitales como analógicos. Entorno a éstos últimos, cubre una de las principales carencias de la Raspberry Pi, ya que ésta no cuenta con pines GPIO para lecturas analógicas. Para solventar esta carencia, puede recurrirse a conversores analógico/digital a los que se conectarán los sensores. En nuestro caso, dado el número de elementos a interconectar (5 sensores, 3 botones y una pantalla), apostamos por una placa que facilita la gestión de las conexiones mediante puertos. Un esquema de los puertos podemos verlo en los **gráfico 3 y 4**. Un esquema electrónico de la placa está disponible en la sección **anexos**.

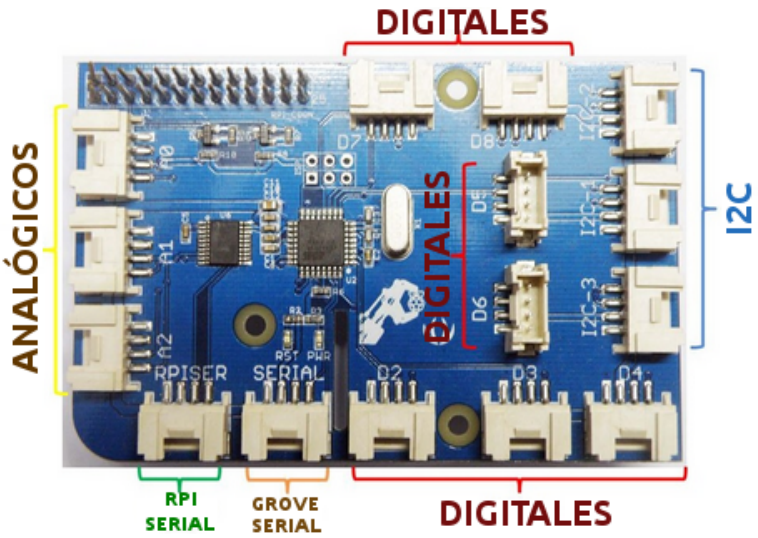


Gráfico 5

5.3 Sensores

Tipo conexión: analógico
Descripción: Este sensor doble sensor permite determinar la temperatura y humedad ambiental con un rango de precisión importante.

Imagen 1

Dato	Valor
Voltage funcionamiento	3.3~6V
Rango humedad	5-99%
Rango temperatura	-60
Precisón humedad	+-2%
Precisón temperatura	+-0.5%

Tabla 23

Grove Sensor de humedad del sustrato

Tipo conexión: Analógico
Descripción: Permite determinar la cantidad de humedad presente en el sustrato. Los valores que nos devuelve incluye un rango entre 0 y 1000 a partir del cuál, podemos determinar el porcentaje de humedad.

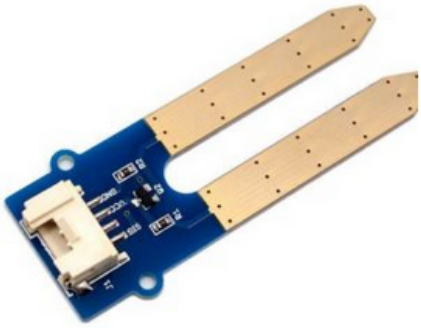


Imagen 2

Dato	Valor
Voltage funcionamiento	3.3~5V
Rango humedad	300-900

Tabla 27

Grove Light sensor

Tipo conexión: analógico
Descripción: Se trata de un sensor que consta de una resistencia LDR que mide la resistencia al paso de la luz. De ésta manera, cuando la luz ambiental se incrementa, la resistencia del sensor disminuye. A partir de dichas variaciones, permite determinar la cantidad de luz. El valor medido representa la cantidad aproximada de lúmenes. Disponible esquema en **anexos**.

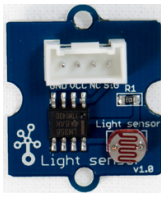


Imagen 3

Dato	Valor
Voltage funcionamiento	3~5V
Intensidad funcionamiento	0.5~3 mA
Tiempo respuesta	20-30 msegs
Pico longitud onda	540 nm
Peso	4 g

Tabla 28

Grove Air Quality

Tipo conexión: Analógico
Descripción: Es capaz de detectar compuesto contaminantes como monóxido de carbono, alcohol, acetona, etc. A mayor cantidad de compuestos contaminantes devolverá un valor superior.

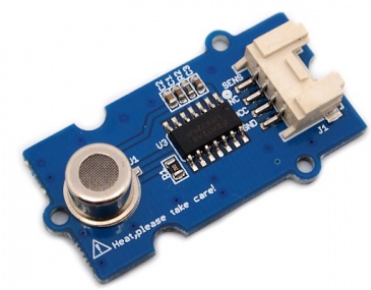


Imagen 4

Dato	Valor
Voltage funcionamiento	3.3~5V
Rango valores	1-1000

Tabla 29

Grove Button

Tipo conexión: Digital
Descripción: Nos permite programar diferentes acciones al pulsarlo. En nuestro caso, cambiar el idioma, activar, o desactivar el audio y apagar el sistema de forma segura.

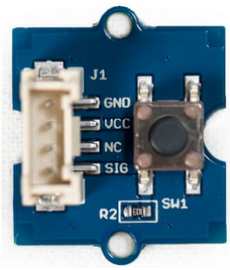


Imagen 5

Pantalla LCD

Tipo conexión: I2C
Descripción: Permite mostrar en la pantalla los valores leídos por los sensores, datos de la conexión como la IP del sistema, etc.

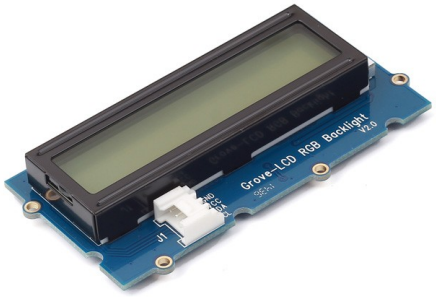


Imagen 6

Dato	Valor
Voltage funcionamiento	5V
Intensidad funcionamiento	<60mA
CGROM	10880 bit
CGRAM	64x8 bit

Tabla 30

Altavoz X-MiniE

Descripción: Permite que la planta emita mensajes hablados con datos de las lecturas de los sensores.

Imagen 7

Dato	Valor
Potencia de salida	2,5 W
Impedancia	3,6 Ohm
Sensibilidad	82 dB
Respuesta de frec. mínima	150 Hz
Respuesta de frec. máxima	20 kHz

Tabla 31

Adaptador WIFI nano usb

Imagen 8

Dato	Valor
Chipset	Realtek
Standard WIFI	802.11b/g/n

Tabla 32

Adaptador hembra-hembra

Descripción: Permite acomodar el puerto RJ45 con la caja diseñada para el tiesto de la caja.

Imagen 9

5.4 Conexión del hardware

Uno de los esquemas más básicos y que es base de aprendizaje en cualquier proyecto electrónico consiste en la conexión de un led. En el caso de la Rasberry PI, el polo negativo se conecta a un pin tierra y el positivo al GPIO. A éste último también irá unida la correspondiente resistencia.

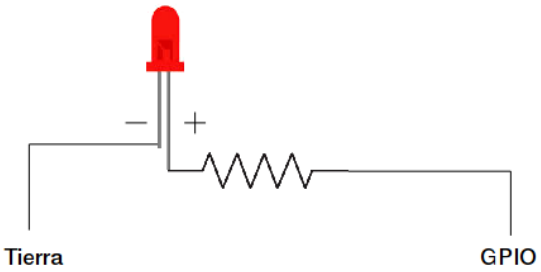


Gráfico 6

Esquema de conexión de sensores analógicos

Como se ha mostrado anteriormente, la placa GrovePI cuenta con 3 puertos analógicos. La lectura de los sensores analógicos (humedad sustrato, luz o calidad de aire) se conecta a estos puetos.

Esquema de conexión de sensores digitales

En este caso, GrovePi cuenta con más puertos, alcanzando un número de 7. A ellos conectamos los sensores de temperatura así como los 3 botones.

Esquema de conexión de la pantalla LCD

La pantalla LCD se trata de un dispositivo ISC, por lo que lo conectaremos a uno de los 3 puertos que nos proporciona la placa

Sistema operativo y servicios

6.1 Elección del sistema operativo

La enorme popularización de la Raspberry Pi ha hecho que posible que contemos con prácticamente una versión de las diferentes distribuciones adaptadas a la arquitectura ARM. Entre todas ellas, destaca Raspbian, con soporte oficial y basada en Debian, cuante con mayor cantidad de documentación que el resto de alternativas.

WIFIPlant no hace un uso excesivo de recursos, pero sin embargo, instalar un sistema como Raspbian con entorno gráfico se antoja poco eficiente. Por este motivo, se apuesta por la distribución Minibian [11], derivada de ésta pero construida con un sistema mínimo sobre el cuál se pueden ir instalando los paquetes estrictamente necesarios. En nuestra pruebas base, comprobamos que una instalación base, tan sólo consume entorno a 50 MB de RAM frente a los más de 120 MB de Raspbian.

Distribución	Consumo RAM
Raspbian	120
Minibian	50

Tabla 33

6.2 Configuración de un punto de acceso con autenticación radius

Si dejamos de lado el diseño propio de entornos domésticos de un punto de acceso con autenticación WEB o WPA y apostamos por un servidor radius, encontramos un esquema básico en el que por un lado se encuentra el punto de acceso y por otro un servidor radius sobre el que consulta los usuarios que podrán acceder al punto de acceso. En nuestro planteamiento, ambos servicios son realizados por la misma máquina (la Raspberry PI).

Requisitos

- El paquete hostapd para GNU/Linux
- El paquete “freeradius”
- Tarjeta WIFI compatible con “hostapd” y drivers compatibles

Paso 1-Comprobamos que Minibian reconoce la interfaz WIFI. Necesitamos conocer el chipset concreto de nuestra tarjeta, ya que los drivers que instala el sistema por defecto, no suelen permitir configurar la tarjeta en modo punto de acceso (sólo cliente).

```
RPI# iwconfig
RPI#lsusb
```

```
Bus 001 Device 002: ID 0bda:8176 Realtek Semiconductor Corp. RTL8188CUS 802.11n WLAN
```

Imagen 10

Paso 2-En nuestro caso, nuestra tarjeta WIFI USB es una Realtek 8188CUS. Necesitamos los drivers específicos, así como el software de punnto de acceso concreto. Descargamos los drivers y compilamos.

```
RPI#apt-get install git
RPI#git clone https://github.com/dz0ny/rt8192cu.git
RPI#cd rt8192cu
RPI# apt-get install make
RPI#make
RPI#make install
```

Paso 3-Deshabilitamos el driver WIFI que usa la tarjeta por defecto y desenchufamos y volvemos a enchufar la tarjeta WIFI.

```
RPI# nano /etc/modprobe.d/8192cu.conf
```

```
blacklist rtl8192cu
```

Paso 4-Instalamos la versión de hostapd de Minibian y luego compilamos.

```
RPI# apt-get install hostapd
```

Paso 5-Descargamos los drivers desde la web de Realtek [5] con el código fuente de la aplicación "hostapd" que usaremos para configurar el punto de acceso.

```
RPI#apt-get install unzip
RPI# unzip RTL8188C_8192C_USB_linux_v4.0.2_9000.20130911.zip
RPI#cd RTL8188C_8192C_USB_linux_v4.0.2_9000.20130911
RPI#cd wpa_supplicant_hostapd/
RPI#tar zxvf wpa_supplicant_hostapd-0.8_rtw_r7475.20130812.tar.gz
RPI#cd wpa_supplicant_hostapd-0.8_rtw_r7475.20130812
RPI#cd hostapd
RPI#make
RPI#cp  hostapd hostapd_cli /usr/local/sbin/
```

Paso 6-Instalamos freeradius y isc-dhcp-server

```
RPI#apt-get install freeradius
RPI#apt-get install isc-dhcp-server
```

Paso 7-Configuramos hostapd, con datos como el canal donde emitirá el punto de acceso, el SSID, la dirección IP y password del radius.

```
RPI# nano  /etc/hostapd/hostapd.conf
```

```
interface=wlan0     #Interfaz WIFI
driver=rtl871xdrv   #Driver correspondiente a Realtek
ssid=WIFIPlant         #Nombre de la red
```

```
hw_mode=g
channel=1        #Canal de emisión
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2

wme_enabled=1
ieee80211n=1
auth_algs=3

ieee8021x=1
eapol_version=2
eap_message=hello-IES-AP
eapol_key_index_workaround=1

wpa_key_mgmt=WPA-EAP
wpa_pairwise=TKIP CCMP
rsn_pairwise=CCMP
wpa_group_rekey=600
wpa_gmk_rekey=86400

eap_server=0

own_ip_addr=127.0.0.1
nas_identifier=a22AP

auth_server_addr=127.0.0.1        #Dirección ip del server Radius
auth_server_port=1812          #Puerto del Radius
auth_server_shared_secret=UOC1234    #Password del Radius

acct_server_addr=127.0.0.1
acct_server_port=1813
acct_server_shared_secret=UOC1234
ctrl_interface=/var/run/hostapd
```

Paso 8-Configuramos el cliente que se conectará al freeradius (el puneto de acceso)

```
RPI# nano /etc/freeradius/clients.conf
```

```
client 127.0.0.1 {
secret = UOC1234
shortname = WIFIPlant #Nombre del SSID
}
```

Paso 9-Configuramos los usuarios que se autentificarán en WIFI a través del servidor RADIUS. Para ello añadimos una línea por cada usuario que queramos agregar.

```
RPI# nano /etc/freeradius/users
```

```
pedroc Cleartext-Password := "UOC1234"
```

Paso 10-Configuramos la red. Es espcialmente importante red de la WIFI, que será la que obtendrán los equipos conectados al WIFI.

```
RPI# nano /etc/network/interfaces
```

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp   #Puede ser estática. Nos conectara a Internet

iface wlan0 inet static #IP fija para la interfaz WIFI
address 192.168.20.1
netmask 255.255.255.0
```

Paso 11-Configuramos el servidor DHCP

```
RPI# nano /etc/dhcp/dhcpd.conf
```

```
authoritative;
ddns-update-style none;
default-lease-time 600;
max-lease-time 7200;
log-facility local7;

subnet 192.168.20.0 netmask 255.255.255.0 {   #Subred de la interfaz WIFI
    range 192.168.20.10 192.168.20.50;                  #Rango de Ips que derá
el servidor
    option broadcast-address 192.168.20.255;
    option routers 192.168.20.1;
    default-lease-time 600;
    max-lease-time 7200;
    option domain-name "local";
    option domain-name-servers 8.8.8.8, 8.8.4.4;
    interface wlan0;                 #Interfaz WIFI del punto de acceso
    }
```

Paso 12-Para que tengamos conexión, tenemos que habilitar el enrutamiento entre interfaces en el equipo.

```
RPI# echo 1 > /proc/sys/net/ipv4/ip_forward
```

También podemos editar el archivo "/etc/sysctl.conf" para que se mantenga en cada reinicio.

```
RPI# nano /etc/sysctl.conf
```

```
net.ipv4.ip_forward=1
```

Paso 13-También es necesario enmascarar el tráfico en la interfaz conectada a Internet.

```
RPI#iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Paso 14-Finalmente, podemos lanzar freeradius y hostapd de forma simultánea para monitorizar las conexiones en tiempo real.

```
RPI#freeradius -X
RPI# hostapd -d /etc/hostapd/hostapd.conf
```

6.3 Diseño de la infraestructura SIP

De cara a habilitar a la planta para que pueda avisarnos mediante llamadas sobre su estado de salud, necesitamos una infraestructura para poder realizar las llamadas. Para ello, definimos un trunk SIP contra un gateway, mediante el cuál sacaremos las llamadas a través de nuestra línea analógica.

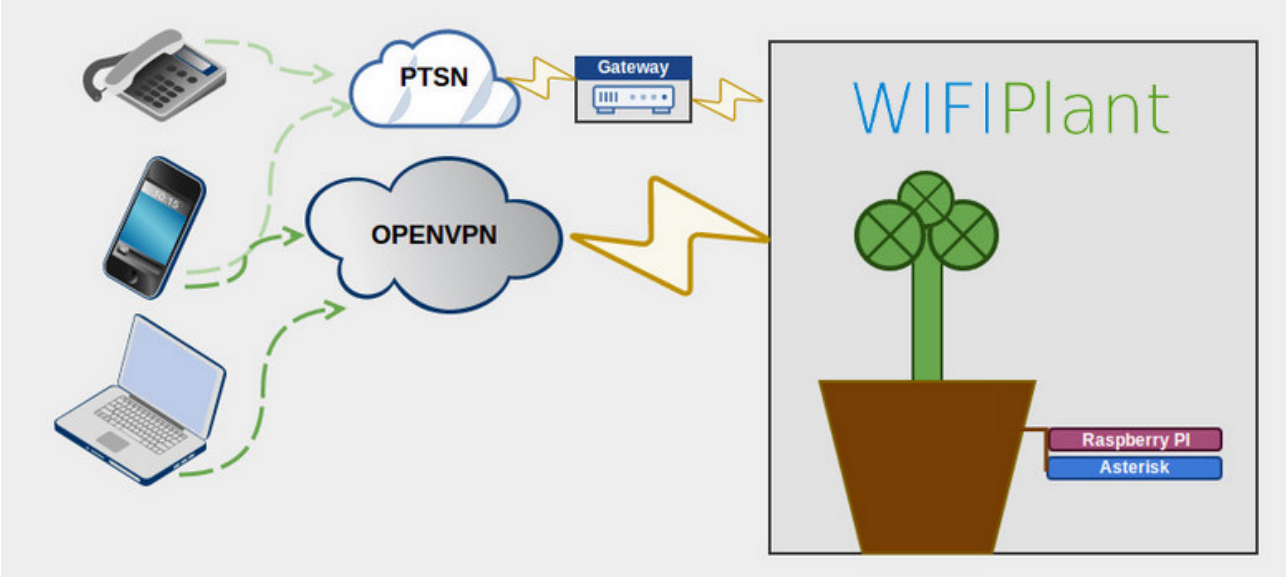


Gráfico 7

6.3.1 Instalación de Asterisk 11 en Minibian GNU/Linux

Paso 1-Vamos a la carpeta /usr/src

```
RPI#cd /usr/src
```

Paso 2-Descargamos la versión Asterisk 11. Pongo el comando entre comillas para que se muestre correctamente, pero tendrías que eliminarlas.

```
RPI#svn checkout http://svn.asterisk.org/svn/asterisk/branches/11 asterisk-11
```

Paso 3-Entramos en el directorio creado en la descarga de Asterisk 11

```
RPI#cd asterisk-11
```

Paso 4-Instalamos las dependencias

```
RPI#apt-get install libncurses5-dev libsqlite3-dev libssl-dev libiksemel-dev
```

Paso 5-Ejecutamos el “configure”

```
RPI#./configure -disable-xmldoc -disable-asteriskssl
```

Paso 6-Instalamos Asterisk

```
RPi#make menuselect & make & make install
```

6.3.2 Definición del peer y comunucación con el gateway

Necesitamos crear un peer en la Raspberry PI para conectar ésta con el gateway y así poder realizar llamadas.

Paso 1-Definimos el peer en Asterisk

```
RPI#vim /etc/asterisk/sip.conf
```

```
[gateway]
type=peer
host=192.168.0.80
context=desde-gateway
qualify=yes
disallow=all
allow=alaw
insecure=port,invite
port=5062
dtmfmode=rfc2833
trunk=yes
directmedia=yes
```

Paso 2-Aplicamos los cambios en Asterisk

```
RPI#asterisk -rx “sip reload”
```

Paso 3-Finalmente, definimos el usuario en el gateway

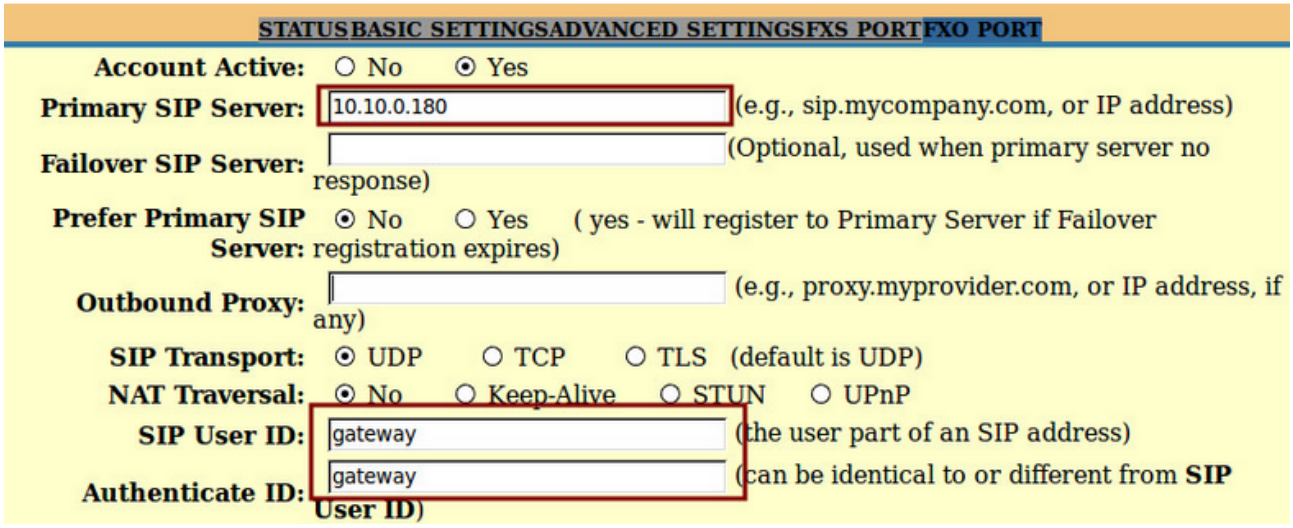


Imagen 11

6.3.3 Realización de llamadas mediante Asterisk

Una vez definido el trunk SIP, podemos realizar llamadas a través de él. En el caso de la planta, es necesario que reproduzca una locución con la alerta que va a informar. Para ello, se utiliza la aplicación "Channel originate y Playback" de Asterisk, las cuáles son llamadas desde las aplicaciones en Bash.

Para realiozar una llamada usamos la línea de comandos de Asterisk

```
RPI#asterisk -rx "channel originate SIP/612345678 application Playback
tengo_calor"
```

El comando anterior lo integramos dentro de los scripts que leen los sensores

```
 if [[ "$max_temp" -gt "temp" ]]; then     # Si la temperatura que nos da el
sensor es superior al límite superior, la planta nos llamará protestando por el
calor
          asterisk -rx "channel originate SIP/612345678 application Playback
tengo_calor"
  fi
```

6.4 Infraestructura openvpn

La instalación de un cliente openvpn nos ofrece dos importantísimas ventajas. La primera, poder mantener la infraestructura de llamadas SIP inalterada, ya que el trunk SIP se levanta por medio de la VPN, contando con acceso al gateway de forma remota para emitir llamadas. Del mismo modo, podemos monitorizar remotamente el estado del sistema y acceder mediante una infraestructura segura en caso de ser necesario.

Creación de los certificados en el servidor openvpn

```
server#cd /usr/share/doc/openvpn/examples/easy-rsa
server#cp -a 2.0/ /etc/openvpn/easy-rsa
server#cd /etc/openvpn/easy-rsa
server#./clean-all
server#./build-ca
server#./build-key-server server
server#./build-key wifiplant
```

Instalación de los certificados en WIFIPlant

Paso 1-Instalamos el paquete openvpn

```
WIFIPlant#apt-get install openvpn
```

Paso 2-Copiamos la ca y certificados creados en el servidor openvpn a la ruta /etc/openvpn

```
WIFIPlant#mkdir /etc/openvpn/wifiplant
WIFIPlant#cp ca.crt wifiplant.crt wifiplant.key /etc/openvpn
```

Paso 3-Creamos la configuración a la que se conectará

```
WIFIPlant#vim /etc/openvpn/wifiplant.conf
```

```
client
dev plant
dev-type tap
tls-client
keepalive 10 60
pull
proto udp
remote wifiplant.irontec.com 5678
ca /etc/openvpn/wifiplant/ca.crt
cert /etc/openvpn/wifiplant/wifiplant.crt
key /etc/openvpn/wifiplant/wifiplant.key
resolv-retry infinite
nobind
persist-key
persist-tun
tls-client
remote-cert-tls server
auth-user-pass
comp-lzo
verb 1
```

Paso 4-Lanzamos el cliente

```
WIFIPlant#/etc/init.d/openvpn start wifiplant
```

6.5 Limitación de la velocidad de conexión en una interfaz

Para limitar la velocidad de la conexión WIFI, se aplica dicha limitación a la interfaz ethenet, ya que según nuestro esquema, la única forma de ofrecer conexión a internez en una interfaz WIFI en la Raspberry Pi es mediante una conexión puente entre través ambas interfaces.

De ésta forma, se emplea la herramienta “wondershaper”

```
RPI#apt-get install wondershaper
```

Para limitar la velocidad a 30 MB de bajada y 3 MB de subida

```
RPI#wondershaper eth0 30000 3000
```

Para limitar la velocidad a 3 MB de bajada y 3 MB de subida

```
RPI#wondershaper eth0 3000 3000
```

Para limitar la velocidad a 280 KB de bajada y 280 KB de subida

```
RPI#wondershaper eth0 280 280
```

7.1 Arquitectura del sistema

El motor de WIFIPlant está formado por dos tipos de aplicaciones fundamentalmente. Por un lado, un conjunto de aplicaciones Bash que trabajan en segundo plano (se cargan automáticamente durante el inicio del sistema). Por otro, una aplicación WEB de control y monitorización. Ésta segunda no hace ningún tipo de llamada a aplicaciones del sistema ni tampoco lectura de sensores. Simplemente, lee los datos con las lecturas almacenados por las aplicaciones en Bash y procesa los datos para representarlos de una forma gráfica.

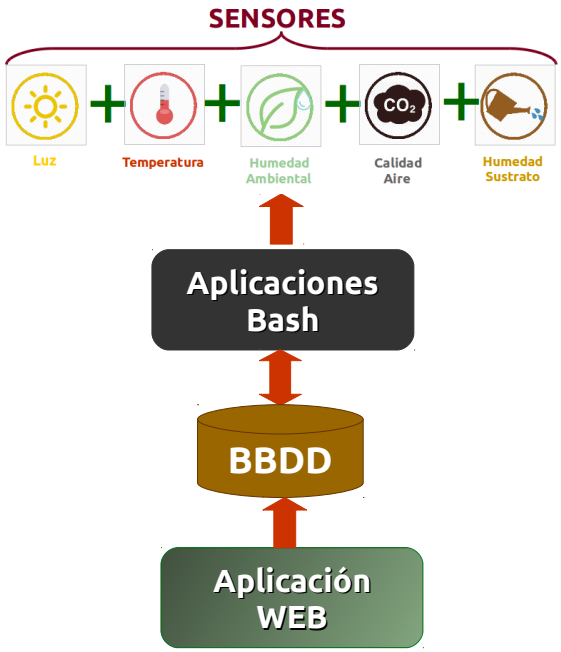


Gráfico 7

7.2 Librerías en Bash para trabajo con GPIO

Ficheros: BashGPIO.sh, Bash_Morse.sh y lcd_rgb.sh

Permiten controlar el acceso a los pines GPIO de la Raspberry PI y así programar funciones para encender/apagar leds, escribir mensajes en la pantalla LCD, emitir señales lumínicas en lenguaje Morse, etc.

7.2.1 Preparación de un pin en Bash

El control de pines de la Raspberry Pi en Bash es sencillo especificando la direcctión (entrada/salida) y el estado (on/off). El proceso en línea de comandos sería el siguiente:

```
RPI#echo "1" > /sys/class/gpio/export
```

```
RPI#echo "out" > /sys/class/gpio/gpio1/direction
RPI#echo "in" > /sys/class/gpio/gpio1/direction
RPI#echo “1” > /sys/class/gpio/unexport
RPI#pin_value=`cat /sys/class/gpio/gpio1/value`
```

7.2.2 Función en Bash para la lectura/escritua de pines

Creamos la función gpio() que permite pasar como argumentos bien leer o escribir el número de pin.

Nota: Función localizada en el fichero “Bash_GPIO.sh”

```
gpio()
#gpio <accion> num_pin valor
#accion pueden ser: input, output, clean y read_pin
{
    local pin
    pin=$1

    gpio output $pin 0
}
```

Ejemplos del funcionamiento de la función sería:

```
gpio read_pin 4 #Leer el valor del pin 4
gpio output 4 1 #Para escribir un “1” en el pin 4
gpio input 4"     #Para preparar el pin 4 para recibir un valor
gpio clean 4       #Para limpiar el pin 4
```

7.2.3 Función en Bash para iluminar/apagar un led

Fichero: Bash_Morse.sh

A partir de la función gpio, podemos crear funciones para enceder o apagar leds.

```
swicht_off_led()
{
    local pin
    pin=$1

    gpio output $pin 0
}
```

7.2.4 Función en Bash para traducir texto a lenguaje Morse en luz

Morse es un lenguaje simple formado por la combinación de “-” y “.”, lo cuál permite formar letras o números en base a una combinación de éstos. Para ello creamos la función “to_Morse()” que nos transforma a Morse la cadena que le hayamos pasado.

Fichero: Bash_Morse.sh

Parte1-Se recorre una cadena carácter a carácter a partir de su longitud

```
to_Morse()
(...)
length_string=${#string_to_convert}
for i in `seq 0 1 $length_string`;
    do
        char[$i]=`echo ${string_to_convert:$i:1}`
```

Parte2-Se crea un array con la definición de todas las letras y números en Morse

```
case "${char[$i]}" in

            0)  char[$i]="-----"
                ;;
            1)  char[$i]=".----"
```

Parte3-Se concatena cada uno de los caracteres tranformados en una cadena final en Morse

```
 in_Morse=$in_Morse" "${char[$i]}
```

7.2.5 Conversión de un texto a una señal luminosa en Morse en un led

Fichero: Bash_Morse.sh

Empleamos las dos funciones anteriores que permiten encender y apagar leds y transformar texto a Morse, para emitir una señal luminosa en un led seleccionado. Para ellos consideramos la fracción de segundo que representarán los “-” y los “.”. Se ha establecido que una raya son “-” puede representarse con 3 / 4 parte de tiempo, mientras que un punto serían 1 / 4 parte.

```
Morse_to_Light()
# Converts an input Morse string into Morse output light using a led.
# Morse_to_Light $string 4
(...)
for i in `seq 0 1 $length_string`;
    do
        char[$i]=`echo ${in_Morse:$i:1}`

        case "${char[$i]}" in

            "-") swicht_on_led $pin
                 sleep 0.8
                 swicht_off_led $pin
                 sleep 0.8
                ;;
            ".") swicht_on_led $pin
                 sleep 0.2
                 swicht_off_led $pin
                 sleep 0.2
                ;;
```

Ejemplo para emitir un mensaje en Morse en un led concreto:

```
Morse_to_Light "Tengo frío" 4
```

7.2.6 Conversión de un texto a una señal luminosa en Morse en una pantalla LCD

Ficheros: Bash_Morse.sh y "lcd_rgb.sh"

Se trata del mismo sistema que la función "Morse_to_Light()" pero aplicada a la pantalla LCD. En este caso se crea la función "Morse_to_lcd()" que hace uso de la librería en Bash proporcionada por GrovePi [6].

```
Morse_to_lcd()
# Converts an input Morse string into Morse output light using a Grovepi
rbg_led.
# Usage: Morse_to_lcd "SOS"
(…)
 case "${char[$i]}" in

            "-") lcd_set_color "black" #Función de lcd_rgb.sh
                 sleep 0.8
                 lcd_set_color "red"
                 sleep 0.8
                ;;
(...)
```

7.2.7 Programación de la pantalla LCD

El fabricante de la pantalla nos ofrece varios ejemplos de código en diferentes lenguajes de programación [6]. Nosotros apostamos por su versión en Bash, ya que se integra perfectamente en el proyecto. Así creamos "lcd_rgb.sh" a partir del fichero proporcionado por GrovePi y creamos funciones para escribir texto en pantalla y el color que mostrará.

Función lcd_write_text()

Permite escribir texto en la pantalla LCD. También permite decidir en qué línea será escrito

```
lcd_write_text()
#Writes a text in the LCD screen in the selected line (1 or 2)
#Usage:
(...)

 for i in `seq 0 1 $length_string`;
        do
        i2cset -y 1 $character $letters ${char[$i]}
        done
```

Función lcd_set_color()

Permite cambiar el color de fondo de la pantalla LCD

```
lcd_set_color()
(…)
 case "$1" in

       'green')
       red=0x00
       green=0xFF
       blue=0x00
         ;;
```

7.3 Aplicaciónes Bash para control de servicios y lecturas de sensores

Se trata de un conjunto de scripts que se lanzan durante el inicio del sistema y permiten comprobar si los diferentes servicios (Hostapd, freeradiuas, dhcp) están corriendo así como la lecturas de los sensores y almacenamiento en BBDD. Del mismo modo, se encargan de realizar las diferentes acciones en función de los datos obtenidos (efectuar mensajes hablados, realizar llamadas telefónicas, cambiar el color de la pantalla, etc).

7.3.1 Script check_buttons.sh

Tipo: Residente.
Función: Comprobar si se ha pulsado uno de los 3 botones (apagado, cambiar idioma, desactivar audio.

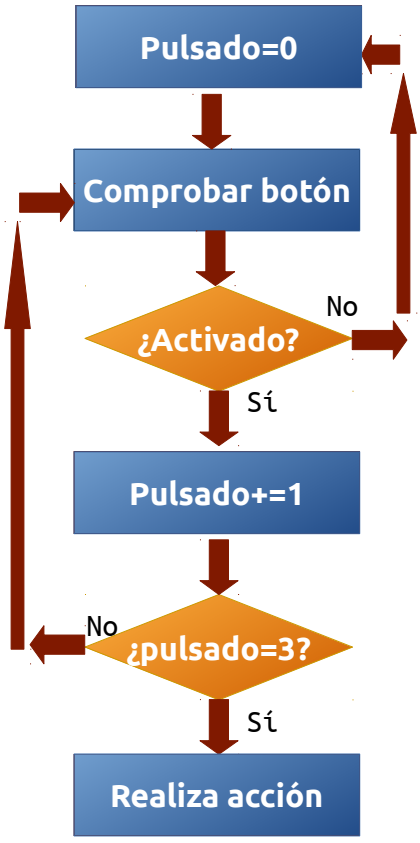


Gráfico 8

La comprobación del estado del botón debe ser especialmente tolerante a fallos ya que los botones pueden desactivar funciones como el audio o apagar el sistema. Según se ha comprobado, una programación incorrecta puede probocar "falsos" positivos. Para evitarlo, cuando se detecta un cambio de estado en eun botón, se hace inmediantamente dos comprobaciones nuevas. Sólo en el caso de que tres comprobaciones certifiquen el estado del botón, se ejecuta la acción.

7.3.2 Script check_sensors.sh

Tipo: Residente
Función: Es lo motor de las acciones de WIFIPlant. Realiza constantemente las lecturas de los sensores y almacena los datos en la BBDD. Del mismo modo, también se encarga de realizar las distintas acciones como enviar alertas mediante voz, lumínicas, etc.

En algunas ocasiones, los sensores reportan lecturas erróneas. Los más comunes son valores fuera de rango o nulos (nan). Para evitarlos, todas las lecturas se repiten constantemente hasta que el valor obtenido es un valor en rango. Esto se consigue comprobando en primer lugar, si el valor obtenido es un número y a continuación si es un valor en rango (acorde a la lectura del sensor).

Gráfico 9

7.4 Aplicación Bash configuración punto de acceso

Nombre del fichero: WIFIPlant.sh
Función: Dotar de una interfaz para la configuración de los servicios de red, DCHP y usuarios del punto de acceso.

Se trata de otra de los principales desarrollos del proyecto. Permite la configuración de los usuarios del punto de acceso así como de los servicios asociados (red, DCHP, Freeradius y hostapd). Del mismo modo, permite una monitorización básica de los usuarios y dispositivos conectados al punto de

acceso, mostrando el nombre de usuario autenticado en el Freeradius, la dirección IP y MAC del dispositivo conectado.

7.4.1 Configuración del servicio de red y DHCP

Esta función nos permite configurar cualquiera de las dos interfaces de red de la Raspberry PI. Dada la arquitectura de WIFIPlant, a la interfaz ethernet le asignaremos una configuración fija o DHCP (actuando a modo de WAN) y a la interfaz inalámbrica le asignaremos una IP fija (la aplicación sólo permite esta opción) y el rango DHCP que servirá a los clientes del punto de acceso.
Cada vez que se arranca la aplicación WIFIPlant.sh, se comprueba si la red está correctamente configurada. En caso de que alguna de las interfaces no estén configuradas, se informa y se lanza el menú de confuguración.

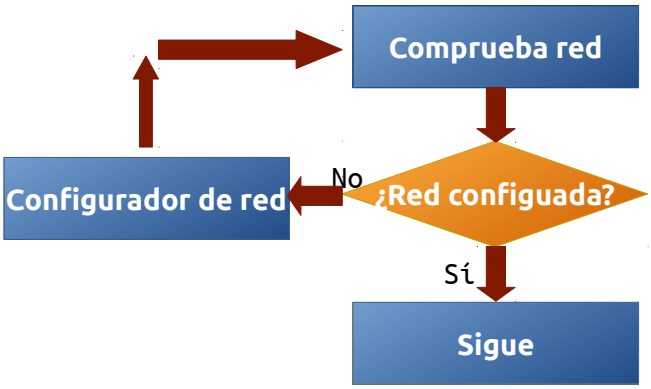


Gráfico 10

Función comprueba_interfaces()

Esta función comprueba las interfaces ethernet y WIFI comenctadas al sistema. Si no existe está configurada alguna de ellas, lanza el asistente de configuración de red.

```
comprueba_interfaces()
{

 int_cable=`ip addr show | grep "state UP" | awk '{print $2}' | cut -f1 -d":" |
grep eth`
    int_WIFI=`ip addr show | grep "state UP" | awk '{print $2}' | cut -f1 -d":"
| grep wlan`

    #Si no tenemos activa alguna, las configuramos y volvemos a probar
    if [[ -z $int_cable ]] || [[ -z $int_WIFI ]]; then
        msg_interfaces
        elige_interfaz_cable
}
```

Función elige_interfaces()

Permite seleccionar las interfaces de red (cableada y WIFI) entre las encontradas en el sistema. Así, WIFIPlant permite añadir interfaces de red permitiendo la configuración del sistema de forma transparente y pudiendo añadir nuevos servicios o funcionalidades.

```
#(...)
#Detectamos el número de tarjetas cableadas

num_interfaz_cable=`ip link show|grep ^[0-9]| grep -v lo|cut -f2 -d":"|sed
's/^[ \t]*//' |grep eth |wc -l`

#Mostramos las iterfaces cableadas
 for i in `seq 0 1 $num_interfaz_cable`;
    do
       linea=`expr $i + 1`
        interfaz_cable[$i]=`ip link show|grep ^[0-9]| grep -v lo|cut -f2 -d":"|
sed 's/^[ \t]*//' |grep eth |head -$linea|tail -1`
       echo_c "$linea ${interfaz_cable[$i]}"
done
#Continúa leyenda la opción seleccionada por el usuario
```

Función configura_red()

Para separar la configuración de la interfaz cableada de la inalámbrica, se crea un fichero en "/etc/network/interfaces" con el nombre de la interfaz. Así, sólo se modifica en cada caso la configuración de cada tarjeta. En todos los casos, el procedimiento es el mismo. Primero se solicita al usuario los parámetros de configuración (IP, máscara, puerta de enlace) y una vez comprobados y validados se crea el fichero con dicha configuración.

```
configura_red()

#(...)Previamente se ha recogido y validado los valores

 int_cable_conf[2]="address $direccion_ip"
        int_cable_conf[3]="netmask $mascara"
        int_cable_conf[4]="gateway $puerta_enlace"

for i in `seq 0 1 4`;
        do
            echo ${int_cable_conf[$i]} >>  /etc/network/$int_cable.txt
        done
#Sigue (…)
```

Función configura_dhcp()

En la interfaz WIFI, deberemos configurar el servicio DHCP para la asignación de IPs a los clientes que se conecten al punto de acceso. La función solicita el rango inicial y final del pool DHCP a partir de la subred configurada en dicha interfaz.

```
configura_dhcp()

 read octeto_rango_inicio

    #Primero comprobamos si hemos introducido un número
    validate_number $octeto_rango_inicio

#Siguen comprobaciones y resto de parámetros
```

```
    echo ${dhcp_conf[0]} >> /etc/dhcp/dhcpd.conf

#Se almacena el resto de parámetros
```

7.4.2 Gestión del punto de acceso

Del punto de acceso podremos configurar el SSID de la red, el canal de emisión y el password con el que conectará con el servidor Freeradius para validar los usuarios.

Función configura_hostapd()

Recoge los valores de de configuación y los aplica reiniciando el servicio.

```
configura_hostapd()

 echo_c "Va a modificar la configuración del punto de acceso. ¿Es correcta la
nueva configuración? (S/N)"
    echo -e "\e[92m"
    echo_c "Nombre de la red: $SSID"
    echo_c "Canal de emisión: $canal"

#Sigue almacenando los datos (...)
```

7.4.3 Configuración de los usuarios del punto de acceso

Necesitamos contar con la posibilidad de añadir y eliminar usuarios que podrán conectarse al punto de acceso. Para ello, se edita el fichero "/etc/freeradius/users" donde están almacenados los usuarios, por lo que una función que recoja, comprueba y alamacene el nombre de usuario y password, permite completarlo.

Función new_freeradius_users()

Simplemente recoge los valores y los almacena en el fichero de configuración de usuarios del servicio Freeradius.

```
new_freeradius_users()

#Solicitamos datos de acceso
new_user_passwd="${new_user_passwd%?}"

        freeradius_users_conf[0]="$new_user Cleartext-Password
:= \"$new_user_passwd\""

#Almacenamos los datos recogidos previamente
 echo ${freeradius_users_conf[$i]} >> /etc/freeradius/users
```

7.4.4 Monitorización de usuarios conectados al AP

Usamos el comando "arp" para localizar los equipos que potencialmente se han conectado al punto de acceso. A partir de ahí, localizamos su MAC y comprobamos en la BBDD de Freeradius para obtener el nombre de usuario con el que se han logado.

Función connected()

Se encarga de optener la IP, MAC y nombre de usuario Freeradius con el que se ha logado los distintos equipos conectados al punto de acceso.

```
connected()

arp | grep $int_WIFI | grep C > connected.txt

#Optebemos la IP y la MACs conectadas
ips_conect[$i]=`cat connected.txt  | grep $int_WIFI | grep C | awk {'print $1'}
| head -$linea | tail -1`
            macs_conect[$i]=`cat connected.txt | grep $int_WIFI | grep C | awk
{'print $3'} | head -$linea | tail -1`

#Sacamos el nombre de usuario del Freeradius
 usuario_logueado[$i]=`mysql -u root -pPassword -h localhost -e "use
radius;SELECT username FROM radacct WHERE callingstationid like '%$MAC_cortada
%' limit 1" | grep -v username`
#Continúa (...)
```

7.4.5 Tolerancia a errores

Tratándose de una aplicación de configuración, solicita la introducción de diferentes parámetros al usuario (direcciones IP, máscaras subred, nombre de usuarios, passwords, etc). Por este motivo, se han implementado varias funciones para validar que los datos introducidos entre las que destacan:

- Comprobación de dirección IP válida
- Comprobación de máscara de red válida
- Comprobación de variables vacías
- Comprobación existencia de ficheros previa modificación
- Comprobación de espacios en passwords
- Conversión a “*” los caracteres introducidos en passwords
- Conversión numérica de todas las variables en las que se realzan cálculos
- Igual dimensionamiento de longitud de variables para visualización en pantalla

7.5 Aplicación WEB de monitorización en PHP

Como se ha definido en el apartado 4 correspondiente al prototipado, la aplicación web está concebida para elegir la especie de planta así como su monitorzación en tiempo real.

Según el planteamiento proyectado, de cara a minimizar el impacto sobre el sistema, la aplicación WEB no realiza ningún tipo de lectura sobre los sensores directamente, sino que lee de la BBDD los datos almacenados con las lecturas por parte de las aplicaciones en Bash que trabajas en segundo plano.

7.5.1 Seguridad y abstracción de la BBDD

Con el objetivo de tener un control centralizado del acceso a la BBDD, así como para poder modificar dichos datos de forma sencilla sin tener que cambiar las credenciales en varias páginas, se ha implementado el acceso a la BBDD mediante un fichero XML, del cuál se extraen los datos para la autenticación.

Fichero config.xlm

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

<configuracion>
    <host>localhost</host>
    <BBDD>wifiplant</BBDD>
    <usuario>root</usuario>
    <password>UOC1234</password>
</configuracion>
```

Obtención de los datos de conexión

Para cargar los datos del fichero “config.xml” hacemo uso de la función simplexml_load_file.

```
 $config= simplexml_load_file('config.xml');
        $host=$config->host;
        $usuario=$config->usuario;
        $password=$config->password;
        $base=$config→BBDD;
```

7.5.2 Conexión a la BBDD

En este caso, hacemos uso de las funciones mysqli_connect ya que PHP nos recomienda su uso frente a mysql_connect ya que ha quedado desfasadas.

```
        $tabla1="planta_humedad";
        $tabla2="planta_temperatura";
        $tabla3="planta_lux";
        $tabla4="planta_humedad_suelo";
        $conexion=mysqli_connect($host,$usuario,$password,$base);

        if ($conexion==FALSE)
        {
            echo ('Error en la conexion');
        }
        else
            {
                mysqli_select_db($conexion,$base);//Conectamos

                $resultado=mysqli_query ($conexion,"SELECT humedad from $tabla1
ORDER BY fecha desc limit 1");
```

7.5.3 Seguridad de los passwords

La contraseñas almacenadas en la BBDD están guardadas codificadas mediante el algoritmo MD5. Aunque no es robusto como otros más modernos, nos permite almacenar las contraseñas de forma cifrada de una manera sencilla. En el entorno del proyecto WIFIPlant, es más que suficiente.

```
$usuario=$_POST['usuario'];
   $passwd=$_POST['passwd'];
   $passwd=md5($passwd);
   $consulta="select usuario, clave, nombre, apellido1 from usuarios where
usuario='$usuario'
               and Clave='$passwd'";
   $result=mysqli_query($conexion, $consulta);
```

7.5.4 Librería Javascript para mostrar gráficas

Dentro del propio código de la página, incluímos la librería y el código Javascript con el valor leído por el sensor en tiempo real.

```
    <script src="./js/justgage.1.0.1.js"></script>

<script>
      var g1, g2, g3, g4;

      window.onload = function(){
        var g1 = new JustGage({
          id: 'g1',
          //value: getRandomInt(0, 50),
          value: $temperatura,
          min: 0,
          max: 50,
          title: 'Temperatura',
          label: 'ºC'
        });
</script>
```

Imagen corporativa

8.1 Elección de colores

El cáracter del proyecto hace que destaque colores asociados a la Naturaleza, con el marrón de la tierra, el verde de las plantas y el azul del agua. La elección de los tonos se realiza sobre la necesidad de realizar un contraste entre estos elementos y la propia definición de la planta.

Colores del logotipo

Color	HTML
Verde tallos	aecf00
Verde hojas	649d1c
Azul ondas	00ccff
Marrón maceta	663300

Tabla 34

Colores del nombre

Color	HTML
Azul	0084d1
Verde	649d1c

Tabla 35

Dado que no se trata de diseños orientados a la impresión de artes gráficas, la representación de los colores se realiza en HTML, en lugar de emplear el correspondiente Pantone, propio de las imágenes corporativas. No obstante, es interesante destacar que en el entorno open source existen alternativas a Pantone muy interesantes como las paletas “Pantano” del diseñador español Jesus David (JesusdA)[12]

8.2 Logotipo de WIFIPlant

El logotipo pretende ser una fusión de elementos. Por un lado el botánico (plantas) y por otro el tecnológico. En este sentido, el más representativo es el punto de acceso WIFI. Buscando una combinación de estos elementos, destacan el asentamiento de una planta en su maceta y un tallo principal a modo de nodo emisor electromagnético.

Por su parte, el cáracter tecnológico del proyecto se ha plasmado en la representación gráfica de la planta, en la que las hojas son los nodos propios que observamos en la interconexión de redes. Al mismo tiempo, simula una estación base o repetidor. La naturaleza inalámbrica se hace patente en las ondas.

Como puede evidenciarse, una parte del logotipo tiene una expresión más “subliminal” (hojas y tallos) frente a otra intencionadamente evidente (ondas WIFI) intentando buscar un equilibro con un fuerte contraste de elementos.

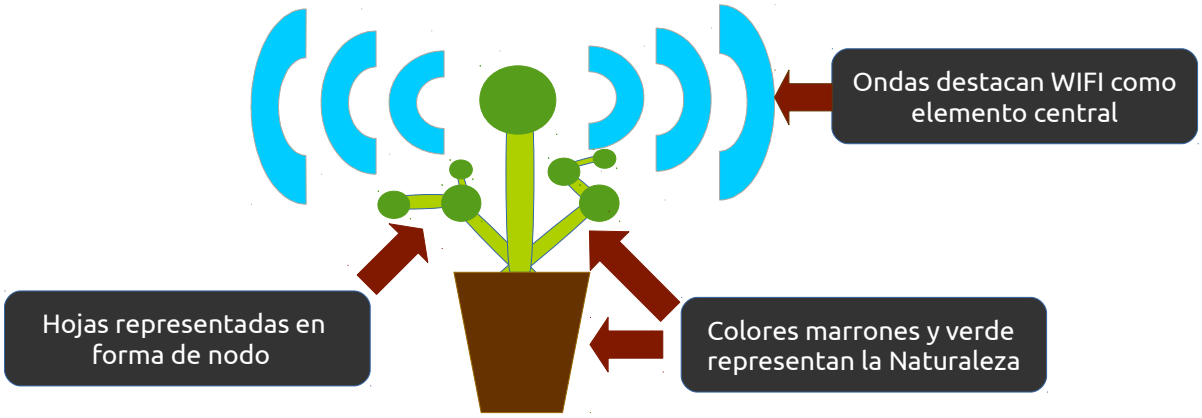


Gráfico 11

8.3 Composición del nombre

Para el nombre del proyecto se ha apostado por la combinación de dos palabras “WIFI” y “Plant” representadas cada una por dos colores asociados a las mismas y representados por la unión de ambos en una palabra.

WIFIPlant

Gráfico 12

Fuente: Deja Vu Sans Light **Colores:** 0084d1 y 649d1c

Pruebas de funcionamiento

9.1 Aplicación de configuración WIFIPlant.sh

Los diferentes servicios (red, Freeradius, hostpd) que gestiona son configurados de manera correcta. Del mismo modo, la monitorización en tiempo real es precisa mostrando los usuarios y equipos conectados.

9.2 Botones y script check_buttons.sh

Como se ha comentado en su implementación, se ha observado que puntualmente, se producía un falso positivo de pulsación de un botón. Esto causaba que se ejecutase una acción sin haber sido pulsado un botón (Ej. reinicio del sistema). Para evitarlo, se ha modificado las funciones que comprueban el estado de los botones para que haga 3 comprobaciones cuando se detecta un cambio de estado. Durante un funcionamiento continuado de varios meses no se ha reproducido el error, por lo que se considera solventado.

9.3 Script check_sensors.sh

Al tratarse del núcleo que mide los sensores del sistema, debe ser especialmente tolerante a fallos. Durante la fase de desarrollo, se han encontrado problemas en los casos en los que los sensores arrojaban lecturas fuera de rango (Ej. Temperatura 65000) o bien, valores nulos. Una vez introducidas las comprobaciones en las diferentes funciones, no se ha vuelto a detectar problemas.

9.4 Aplicación WEB

El funcionamiento de la aplicación WEB no ha presentado prácticamente ningún problema. Esto es consecuencia de la delegación de la capa de lectura de sensores y comprobación de servicios a las aplicaciones residentes en memoria y a la aplicación de configuración en Bash. Así, la aplicación web simplemente realiza las lecturas almacenadas en la BBDD y muestra las imágenes asociadas con el estado.

9.5 Rendimiento del punto de acceso

Una vez concluido y comprobado el funcionamiento correcto de los diferentes elementos que incluyen el proyecto, podemos realizar las pruebas de funcionamiento del punto de acceso. En éste sentido, se antojan fundamentales dos valores. Por un lado, el número de clientes conectados al punto de acceso y por otro, la velocidad de la conexión. Por esta razón, elaboramos sendas pruebas comparándolas con un punto de acceso comercial (TP-Link N600) [11].

La elaboración de las pruebas frente a un router comercial como el TP-Link N600 se consideran representativas ya que se trata de un modelo de gama media y popular que puede también permite instalar firmware de código abierto. Del mismo

modo, cumple el standard (802.11N), al igual que la tarjeta WIFI empleada para el diseño de WIFIPlant.

De cara a obtener datos representativos, necesitamos asimilar las condiciones entre ambos sistemas para intentar hacerlos lo más fiables posibles. De éste modo, se han tenido en cuenta los siguientes aspectos:

- Las redes WIFI de ambos se configuran en el mismo canal (11)
- Durante la realización de las pruebas, se desactiva la emisión WIFI del otro punto de acceso para evitar interferencias
- El modelo TP-Link N600 se configura con autenticación Radius, realizando dicha autenticación contra el servidor Freeradius de WIFIPlant.
- Se realizan 3 mediciones de cada prueba y se calcula la media para así obtener datos más fiables.
- Sólo un único cliente (WIFI o DHCP por cable) estará conectado a cada dispositivo durante las pruebas.
- La prueba se ejecuta desde el mismo dispositivo (Macbook Pro 8.1)

9.5.1 Velocidad de la conexión de descarga

Objetivo: Descargar un fichero de 100 MB de un servidor WEB en el menor tiempo posible.

```
macbook#time wget http://wifiplant.irontec.com/test_descarga.zip
```

Resultados

	TP-Link N600	WIFIPlant
Medición 1	55,00	71,00
Medición 2	52,00	63,00
Medición 3	61,00	59,00
Media	**56,00**	**64,33**

Tabla 36

Como podemos observar, las diferencias entre ambos sistemas no es abultado, aunque arrojando valores ligeramente superiores el modelo comercial.

9.5.2 Velocidad de navegación

Objetivo: Realizar un test de velocidad de un proveedor especializado [14] para comparar los valores arrojados.
Datos de la conexión (bajada/subida): 50MB/5MB

Resultados

	TP-Link N600		WIFIPlant	
	Bajada	Subida	Bajada	Subida
Medición 1	8,44	1,49	7,56	1,34
Medición 2	8,11	1,48	7,43	1,21
Medición 3	8,74	1,48	7,34	1,68
Media	**8,43**	**1,48**	**7,44**	**1,41**

Tabla 37

Al igual que ha sucedido con el test de descargas, el TP-Link N600 se impone a WIFIPlant ligeramente. Como valor insterante, destaca la velocidad de subida que se mentiene prácticamente inalterada, lo que nos puede indicar que hasta una cierta velocidad de navegación, la velocidad se mantiene similar, decantándose a favor del TP-Link a partir de dicha velocidad.

9.5.3 Discusión de los resultados

Los resultados nos aportan unos valores ligeramente superiores a favor del TP-Link N600. Aunque se ha tratado de implementar un entorno equilibrado que permita arrojar resultados fiables, no podemos obviar aspectos que pueden influir distorsionar los valores presentados.

Uno de ellos es la configuración del punto de acceso en el N600. De ésta forma, para la comparación con WIFIPlant, se ha configurado mediante autenticación Freeradius contra la WIFIPlant. Así, podemos determinar que el N600 parte en desventaja al tener que autenticar un usuario por red frente a WIFIPlant, que tiene el servicio corriendo en local. No obstante, se estima que este impacto es relativamente bajo, ya que afecta mayormente al momento de autenticación y no tanto al rendimiento de navegación.

En línea con el punto anterior, tal vez hubiese sido adecuado complementar la prueba de rendimiento implementando el punto de acceso mediante WPA2 en ambos sistemas (configuración más habitual de TP-Link N600).

Conclusiones

10.1 Valoración económica

Cuando en la PEC2 analizábamos la viabilidad del proyecto, uno de los aspectos fundamentales que destacaba del mismo era la excasa necesidad de recursos económicos (considerando la magnitud que suelen alcanzar los proyectos TI).

Una vez completado y superado todos los requisitos, ratificamos este planteamiento. En la **tabla 2**, se muestran los gastos de implantación. Como puede observarse, no hay ninguna variación respecto al planteamiento original, si bien debe estimarse, que no se ha considerado el potencial impacto económico que generarían la(s) persona(s) dedicadas a su implantación.

Una valoración del ratio entre la magnitud del proyecto y de los recursos económicos dedicados, nos muestra una fuerte desviación hacia el primero, por lo que se determina que WIFIPlant es satisfactorio desde el punto de vista económico. A ésta tesis se suman argumentos como el potencial número de proyectos derivados al no estar circunscrito a ningún cliente ni proyecto en exclusividad o su carácter de marketing que puede ser utilizado reiteradamente.

10.2 Seguridad de sistemas IOT

La seguridad ha sido uno de los grandes retos históricos de los sistemas informáticos. Antes de la evolución de las redes e Internet, ésta se veía comprometida fundamentalmente por la difusión de software infectado. Con la implantación de las redes, las intrusiones y amenazas a los sistemas informáticos crecieron exponencialmente.

Centrando este análisis entorno a los productos IOT, encontramos una primera potencial -y evidente- debilidad respecto al resto de sistemas ya que éstos cuentan con hardware más limitado que el resto, necesitan estar conectados y en muchas ocasiones no cuentan con sistemas de seguridad avanzados como el que se puede encontrar en entornos empresariales o centros de datos. Así, dispositivos cotidianos como una webcam, cuentan con un hardware que ofrecen funciones propias de hardware mucho más potente (conexión cableada e inalámbrica, servicio WEB, FTP, procesado de imágenes, video, etc) y todo en un hardware limitado.

La conectividad permanente que sustenta la naturaleza de éstos dispositivos así como el abandono por parte de muchos fabricantes a realizar actualizaciones de firmware que corrijan errores de suguridad, hacen que muchos dispositivos conectados hoy en día sean vuelnerables. Una evidencia la encontramos en el ataque masivo a empresas de la costa Oeste de EEUU que provocó la caída de empresas como Twitter o Amazón y que fue sustentado, por primera vez, en el uso de estos dispositivos [10]

WIFIPlant, aún tratándose de un proyecto sobre una base IOT, es potencialmente menos vulnerable a estos problemas. Argumenta esta afirmación que el proyecto está implementado sobre la base de la Raspberri PI, que a su vez, cuenta con distribuciones GNU/Linux (en nuestro caso Raspbian) que cuentan con actualizaciones de seguridad como la de cualquier hardware de un datacenter y puede por tanto, ser actualizada desde la línea de comandos tan pronto sale una

actualizacón de seguridad y sin la necesidad de esperar a que un fabricante apueste (en caso de hacerlo) por desarrollar una versión de firmware que solvente las vulnerabilidades que se van encontrando.

En nuestro caso, al tratarse de un punto de Acceso WIFI, la principal vulnerabilidad la encontramos en el lado de éste, ya que variados clientes podrán conectarse al mismo. Por su parte, la red WAN conectada por cable al router (o switch) está controlada por los dispositivos de seguridad intermedios.

Así, apostamos por realizar una diferenciación entre los servicios internos que ofrecen la Raspberry (ssh, web, MySQL) y los mismos (u otros) externos. En definitiva, el acceso al panel de control, a la línea de comandos y cualquier otro aspecto de configuración de WIFIPlant, sólo quedará restringido a la interfaz eth0 y será denegado en la wlan0. El primer caso es sencillo de implementar, ya que una regla de iptables que considere la interfaz eth0 como origen, nos permitirá abrir el servicio. Sin embargo, en el caso de la interfaz wlan0, debemos considerar 3 condicionantes: interfaz de origen (wlan), máquina destino y servicio.

Un control estricto de los servicios internos a los que pueden acceder los clientes del punto de acceso, sin limitar el acceso a otros externos

10.3 Conclusiones

Una vez que prácticamente ha sido superada (y concluída) la parte técnica del proyecto, contamos con la experiencia que nos han ido enriqueciendo las diferentes etapas del mismo. De ésta manera, la primera y más relevante conclusión gira entorno a la viabilidad, su dimensión y las tecnologías que lo han hecho posible. Así, podemos afirmar que sólo la naturaleza abierta de las diferentes herramientas que se han utilizado, han permitido abordar este proyecto.

WIFIPlant puede medir las necesidades fisico-químicas de un ser vivo como una planta gracias a electrónica accesible como la Raspberry Pi, la cuál es gestionada por un sistema operativo como GNU/Linux, su punto de acceso está implementado gracias al proyecto Hostapd y los usuarios que se conectan, son controlados mediante un serividor radius Freeradius. La aplicación WEB corre en un servidor Apache, está implementada con tecnologías como PHP, CSS y librerías de Javascript. Gimp y Libreoffice son motores de la imagen corporativa. Este conjunto de aplicaciones interactúan entre ellas gracias a otras decenas más que trabajan en paralelo. Y a todas ellas les une su naturaleza abierta, sin la cuál, no se habría producido la sinergía que germinó su concepción.

Si analizamos detenidamente cada uno los proyectos empleados en WIFIPlant, podemos sumar cientos y cientos de desarrolladores y de horas dedicadas a ellos. Encontramos pues, una extraordinaria virtud del Software Libre, el cuál empodera a personas y organizaciones para poder acometer proyectos que en otros casos, sólo estarían al alcance de grupos muy reducidos. Es decir, WIFIPlant ha sido viable por y gracias a la existencia de tecnologías abiertas. Este es un punto relevante entorno al proyecto ya que en la industria TI, podemos encontrar decenas de proyectos que son viables tanto con tecnologías de naturaleza cerrada o abierta, pero no así WIFIPlant.
Llegados a este punto, encontramos una doble vertiente que nos abren las tecnologías abiertas. Por un lado la dimensión de los proyectos que pueden ser abordados y por otro, el número de personas y organizaciones al que abre la posibilidad de implementarlos. Así, caminamos hacia lo que ha sido bautizada

como “La cuarta revolución industrial” o “Industria 4.0” en la que tendemos a digitalizar todos los procesos productivos independientemente del sector.

Hasta hace relativamente poco, la posibilidad de innovar, estaba condicionada por el acceso a la tecnología (cara y limitada en su mayor parte). Las tecnologías abiertas, superan esta primera barrera abriendo un camino de mejora contínua, ampliando el campo de aplicación y por tanto, mercado y oportunidades.

10.4 Discusión

Una lectura “profunda” o crítica nos permiten encontrar multitud de aspectos que podrían ser objeto de mejora. Obviamente, la ejecución de la parte teórica del proyecto y la toma de contacto entorno a sus necesidades, nos ofrece un conocimiento desconocido al comienzo y que puede ser objeto de experiencia de cara a afrontar futuros proyectos o bien hacer una revisión del actual. De ésta manera, encontramos los siguientes puntos:

Gestión de versiones

La empresa está apostando decididamente por mantenerse actualizada entorno al software de control de versiones. Si bien tradicionalmente ha apostado por Subversion, actualmente está migrando su infraestructura a Git [15]. En éste sentido, hubiese sido interesante haber aprovechado dicha infraestructura en la planificación inicial.

Entorno de programación

Como se ha mencionado en la definición del proyecto, se ha apostado por usar el conocimiento e itinerarios cursados en el Máster, de forma que lenguajes como PHP o Bash toman un papel importante. De ésta forma, se ha conseguido integrarse cómodamente en el ámbito de trabajo de la empresa, ya que está especializada en ambos entornos, especialmente el primero para desarrollo. Sin embargo, para la aplicación WEB se ha apostado por un desarrollo nuevo, al que se le han añadido requisitos de accesibilidad. Aquí encontramos una potencial doble laguna. En primer lugar, podría haberse apostado por algún entorno de desarrollo WEB como Bootstrap o incluso un template con una licencia compatible con el proyecto. De ésta manera, se hubiesen dedicado menos recursos a la programación completa y potencialmente, se podría haber incluído mejoras en un proyecto libre existente, entorno a la accesibilidad.

Criterios de accesibilidad

Relacionado con el punto anterior, se ha conseguido hacer una WEB que se integre en diferentes navegadores y resoluciones y apostando por colores y contenidos accesibles. Sin embargo, ninguno de ambos aspectos está cubierto completamente. Así, una framework o template libre hubiesen ahorrado recursos que podrían haber sido dedicados a la mejora de éste aspecto y de ésta manera, crear un proyecto libre como WIFIPlant y contribuir a la mejora de uno existente de forma simultánea.

Tecnologías empleadas

En su conjunto, las diferentes tecnologías empleadas están actualizadas, especialmente las relativas al software (Debian GNU/Linux, Hostpad, Freeradius, etc). Por su parte, el hardware, cuenta con una versión relativamente reciente de Raspberry Pi (v2B+) y de GrovePi(v2). Sin embargo, el standard WIFI implentado (IEEE 802.11n) en el punto de acceso, aún no siendo obsoleto, la nueva especificación (IEEE 802.11ac) no habría supuesto un importante aumento de gasto. En cualquier caso, éste último aspecto tiene un impacto mínimo, si consideramos que los puertos USB 2.0 de la Raspberry PI, los cuáles no permiten aprovechar la mayor capacidad de velocidad de la especificación (IEEE 802.11ac).

Consideraciones electrónicas

Aunque el objeto sustancial del proyecto se centra entorno a un desarrollo sobre tecnologías abiertas, no podemos olvidar el importante componente electrónico que incorpora (Raspberry PI, sensores, etc). Aunque se ha apostado por documentar los esquemas electrónicos de éstas placas así como la del diseño del conjunto del proyecto, un enfoque más detallado con herramientas especializadas hubiera enriquecido el conjunto.

Referencias

[1] “Irontec” Werb corporativa de Irontec-Internet y Sistemas sobre GNU/Linux https://www.irontec.com [Acceso: 17 de octubre de 2016]

[2] “Github” Repositorio de Github de la Raspberry Pi Foundation https://github.com/raspberrypi?page=1 [Acceso: 17 de octubre de 2016]

[3] “BBVAopen4u” Plataforma del BBVA sobre nuevas tecnologías https://bbvaopen4u.com/es/actualidad/arduino-y-raspberry-pi-dominan-el-hardware-abierto-pero-cada-vez-hay-mejores-alternativas [Acceso: 17 de octubre de 2016]

[4] González Ivan, González Juan, Gómez-Arribas Francisco " *Hardware libre: clasificación y desarrollo de hardware reconfigurable en entornos de GNU/Linux*. Universidad Autónoma de Madrid http://es.tldp.org/Presentaciones/200309hispalinux/8/8.pdf [Acceso: 17 de octubre de 2016]

[5] “Realtek” Centro de descargas de Realtek http://www.realtek.com/downloads/downloadsView.aspx?Langid=1&PNid=48&PFid=48&Level=5&Conn=4&DownTypeID=3&GetDown=false&Downloads=true#RTL8192CU [Acceso: 17 de octubre de 2016]

[6] “Seedstudio Github” Proyecto Grove_LCD_RGB en Github. https://github.com/DexterInd/GrovePi/blob/master/Software/Shell/Grove%20-%20LCD%20RGB%20Backlight/lcd.sh [Acceso: 22 de octubre de 2016]

[7] “justGage” Web del proyecto justGage. http://justgage.com/ [Acceso: 22 de octubre de 2016]

[8] “Seed studio Wiki” Wiki de la empresa Seed Studio. http://wiki.seeed.cc/Grove-Temperature_and_Humidity_Sensor_Pro/ [Acceso: 24 de octubre de 2016]

[9] “Navinbhaskar” Blog dedicada nuevas tecnologías https://navinbhaskar.wordpress.com/2015/09/03/python-on-intel-galileoedison-part6-light-sensor/ [Acceso: 24 de octubre de 2016]

[10] “El Periódico” Web de El Periódico. http://www.elperiodico.com/es/noticias/sociedad/ataque-informatico-deja-sin-servicio-dos-horas-twtter-spotify-5579786 [Acceso: 24 de octubre de 2016]

[11] “Minibian” Web del proyecto Minibian en Sourceforge https://sourceforge.net/projects/minibian/ [Acceso: 21 de noviembre de 2016]

[12] “Jesus David” Blog personal del diseñador Jesus David http://www.jesusda.com/blog/index.php?id=358 [Acceso: 21 de diciembre de 2016]

[13] “TP-Link N600” Web del modelo de router N600 del fabricante TP-Liknk http://www.tp-link.es/products/details/TL-WDR3600.html [Acceso: 23 de diciembre de 2016]

[14] “Speedofme” Portal para medir la velocidad de navegación WEB http://speedof.me/ [Acceso 23 diciembre de 2016]

[15] “Gihbub Irontec” Repositorio de la empresa Irontec en Github http://https://github.com/irontec [Acceso 23 diciembre de 2016]

Anexo 1-Capturas aplicación web

La **imagen 12** muestra el panel de control central.

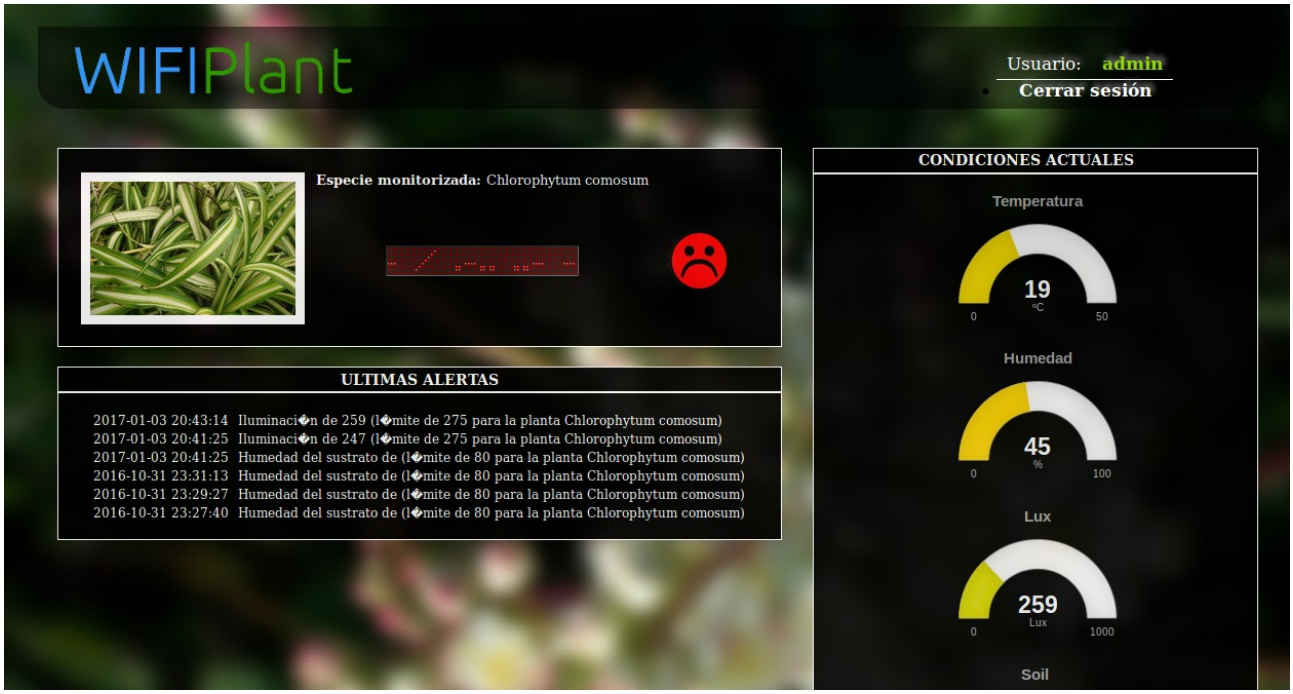


Imagen 12

En la **imagen 13**, observamos el menú que permite elgir la especie a monitorizar.

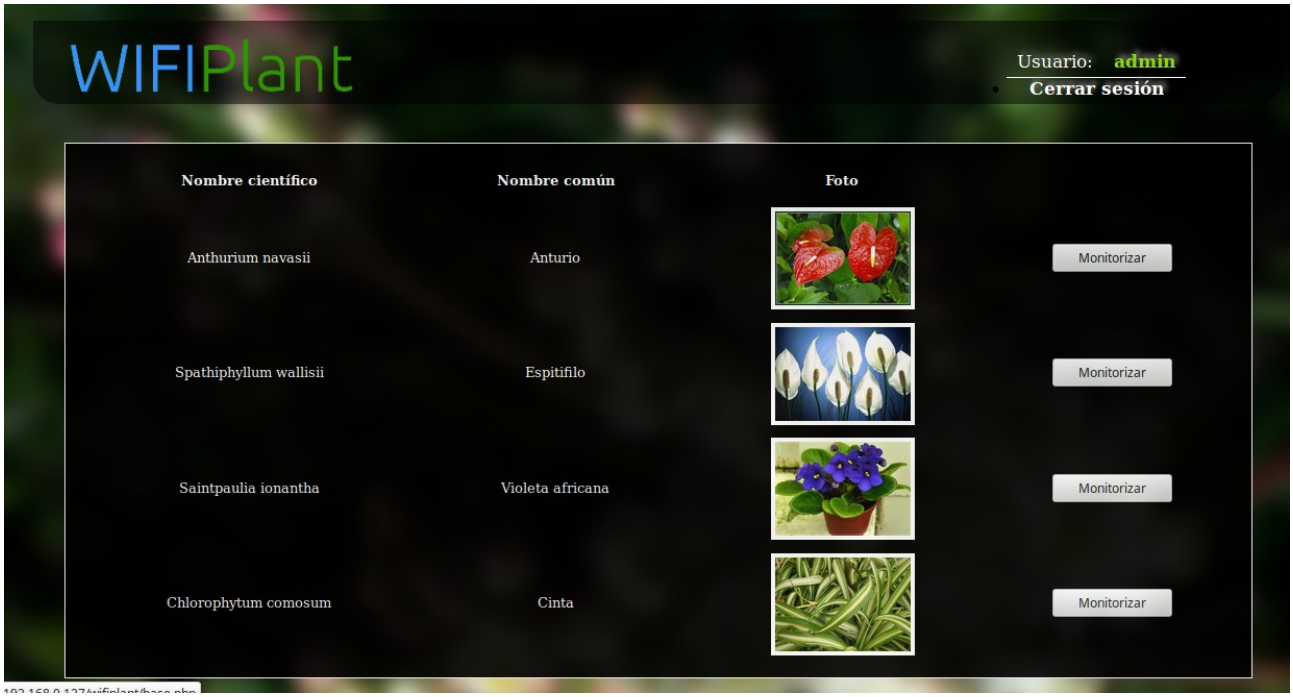


Imagen 13

La **imagen 14** muestra el estado de los sensores

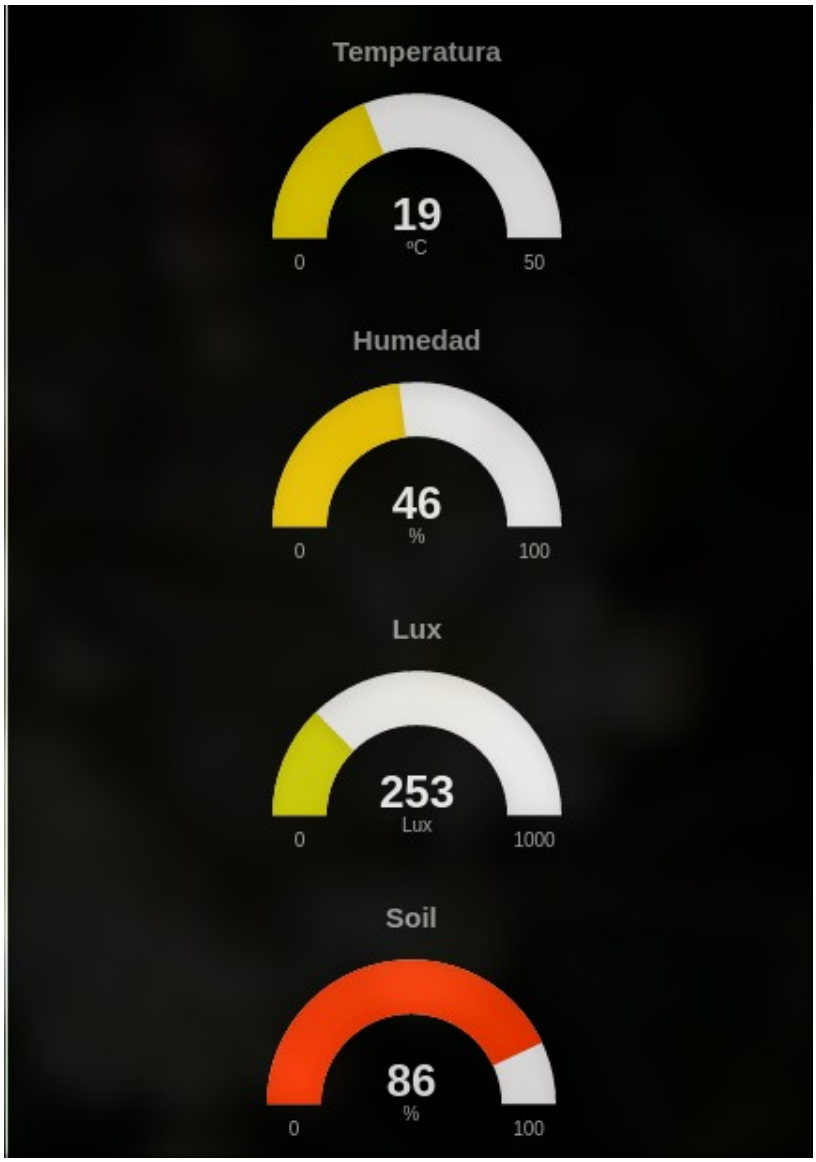


Imagen 14

En la **imagen 15** vemos el panel de login

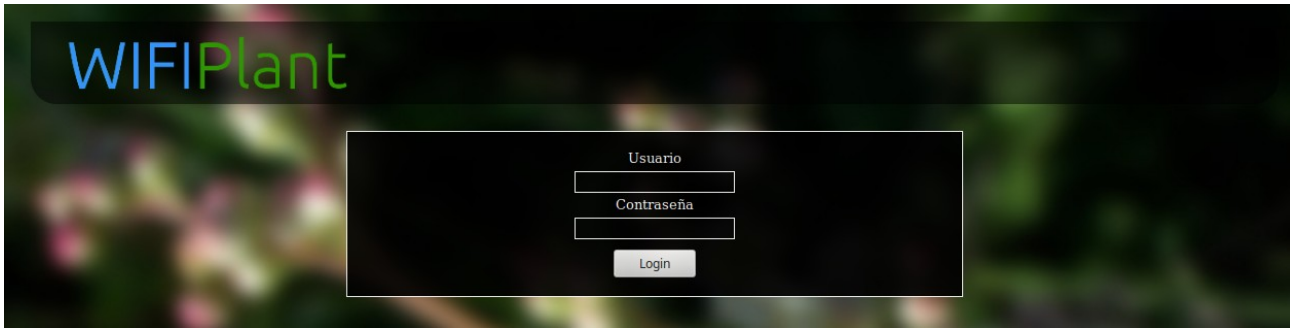


Imagen 15

Anexo 2-Capturas aplicación WIIFPlant

La **imagen 16** muestra el panel principal de control con los principales menús.

Imagen 16

La **imagen 17** muestra el menú de gestión de red, con las distintas opciones de configuración.

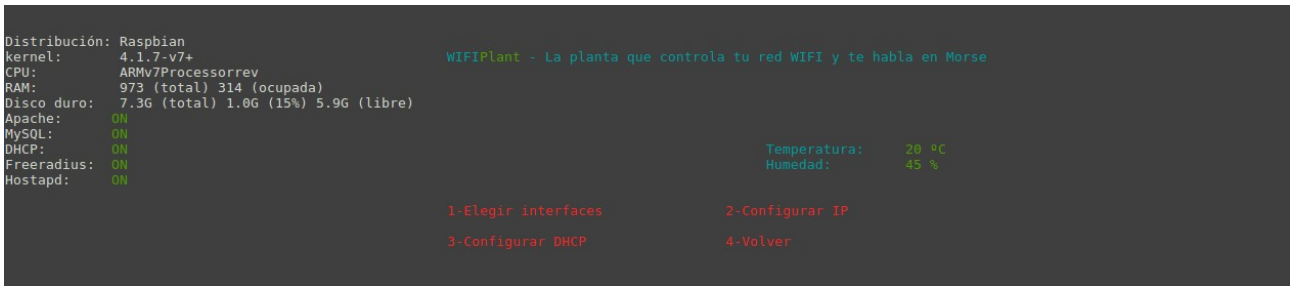


Imagen 17

La **imagen 18** muestra el sistema de monitorización en tiempo real de los clientes conectados al punto de acceso.

Imagen 18

En la **imagen 19** observamos un menú interactivo consultando si los datos de configuración del punto de acceso son correctos antes de proceder a guardarlos.

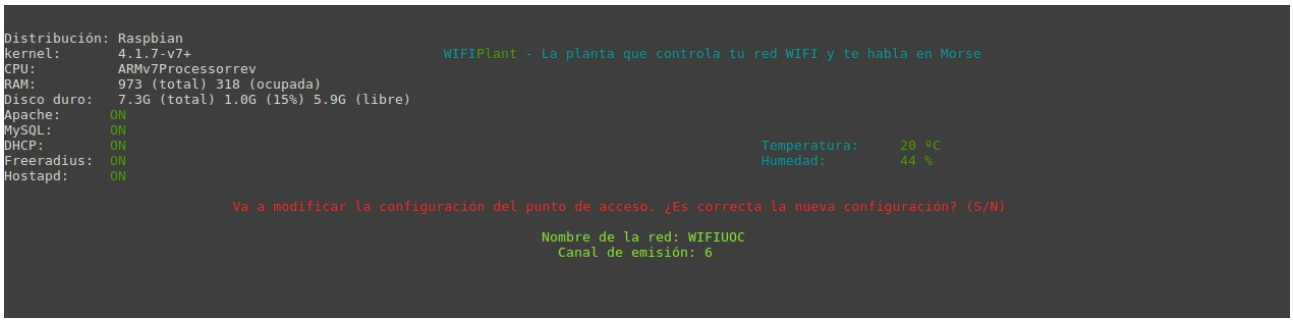


Imagen 19

La **imagen 20** muestra los menús de gestión de usuarios del servicio Freeradius (con los que se permitirá acceder al punto de acceso).

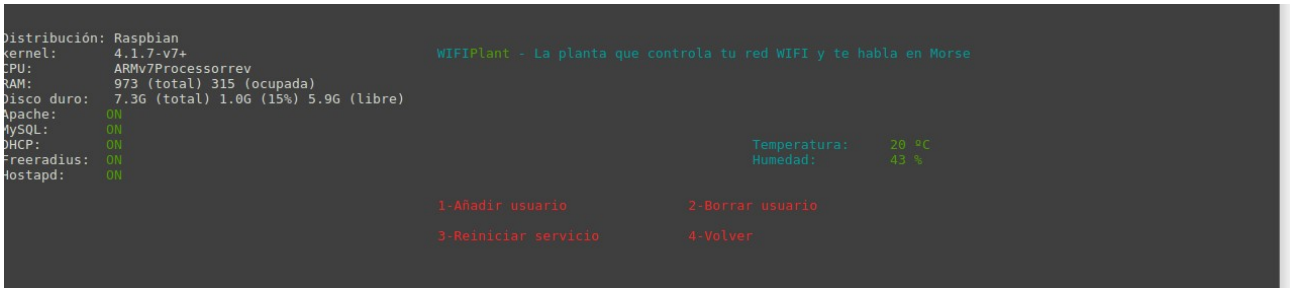


Imagen 20

En la **imagen 21** se muestra el menú interactivo de borrado de usuarios dados de alta en el Freeradius.

Imagen 21

Anexo 3-Capturas diseño final WIFIPlant

La **imagen 22** muestra una vista general.

Imagen 22

La imagen 23 muestra una la parte posterior donde se ubican los botones, sensores y tarjetas de red.

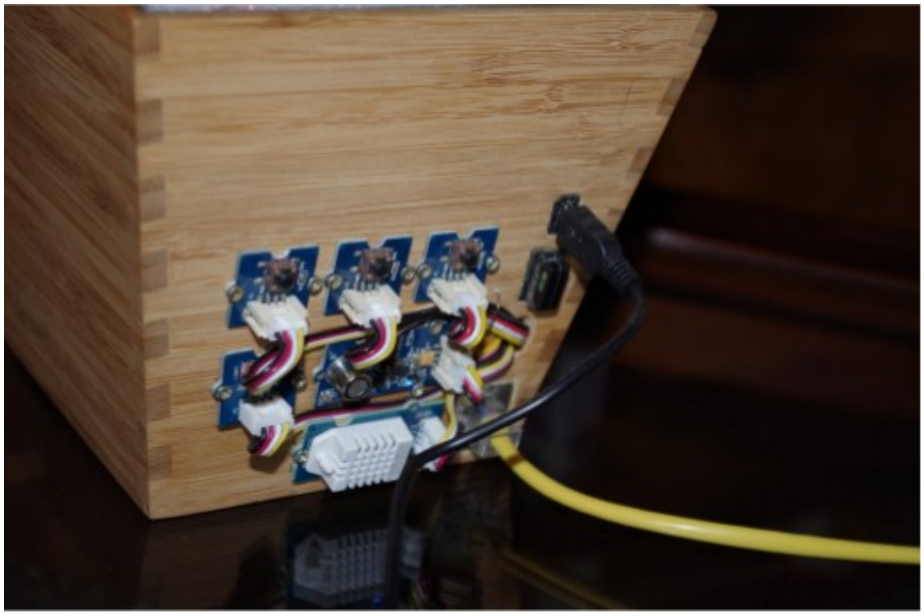


Imagen 23

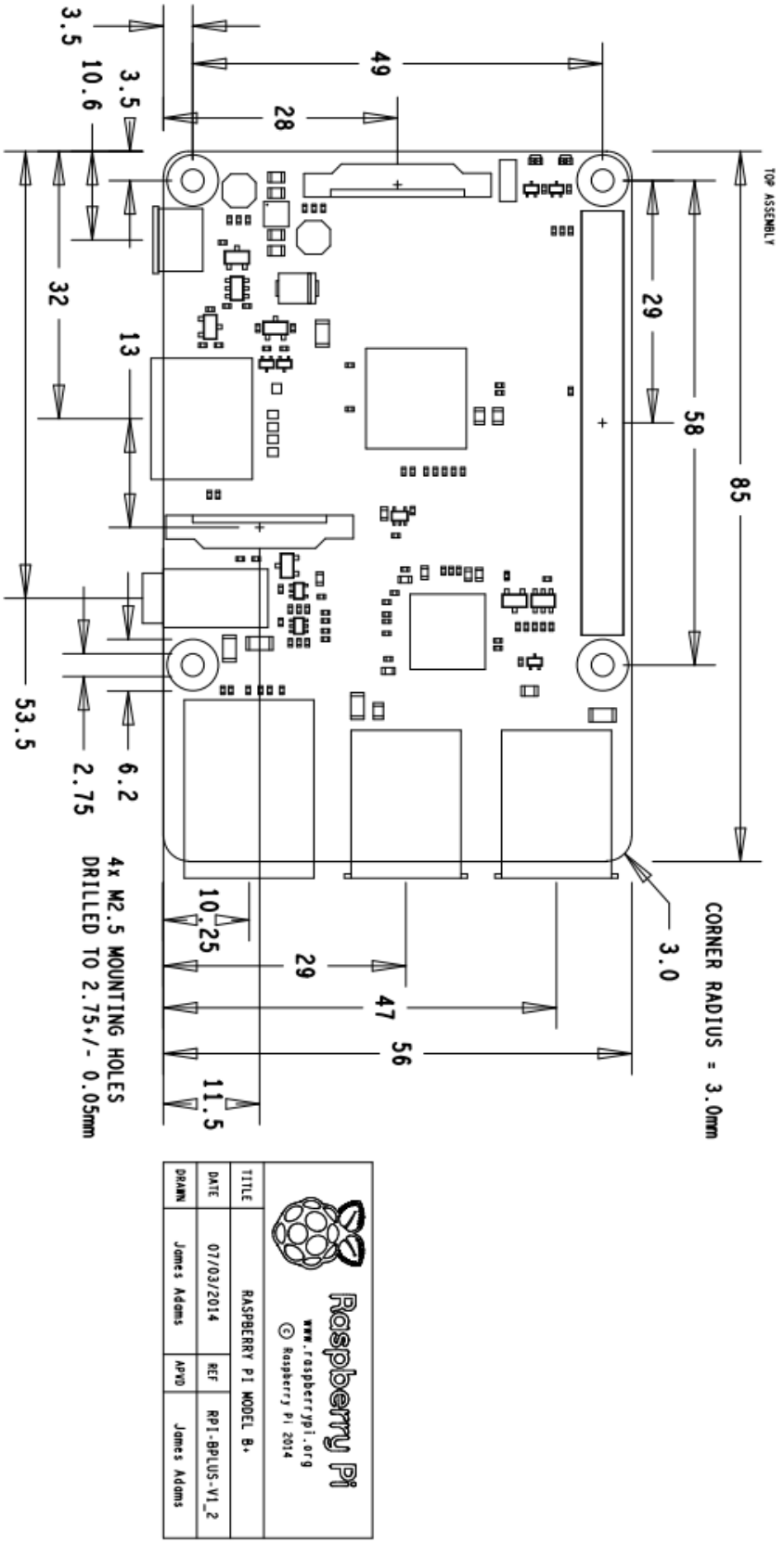
En la **imagen 24** vemos un primer plano con la pantalla mostrando las lecturas actuales de los sensores y en color verde, indicando un correcto funcionamiento del sistema.

Imagen 24

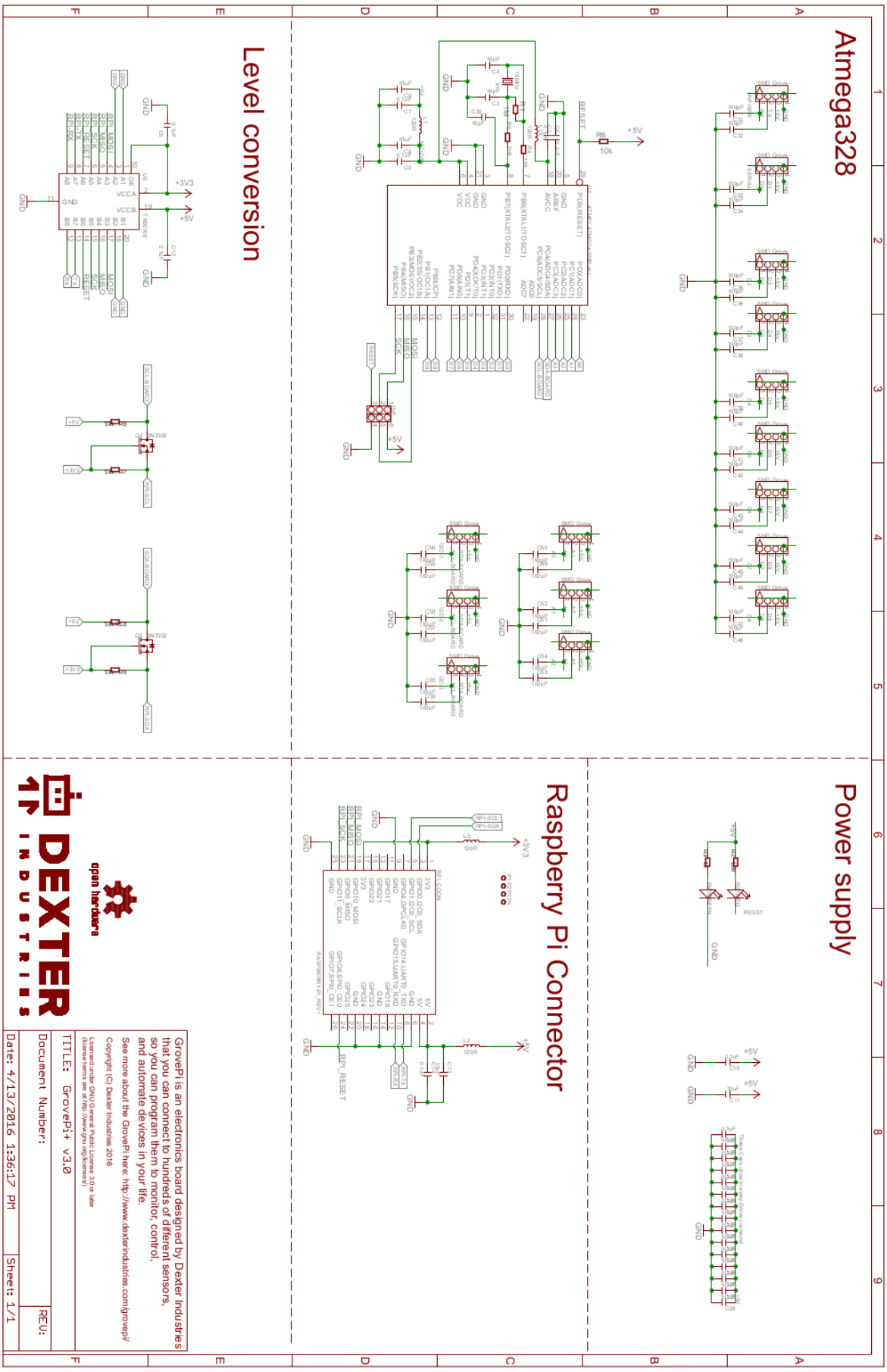
En la **imagen 25** vemos un primer plano con el detalle del sensor de humedad de sustrato.

Imagen 25

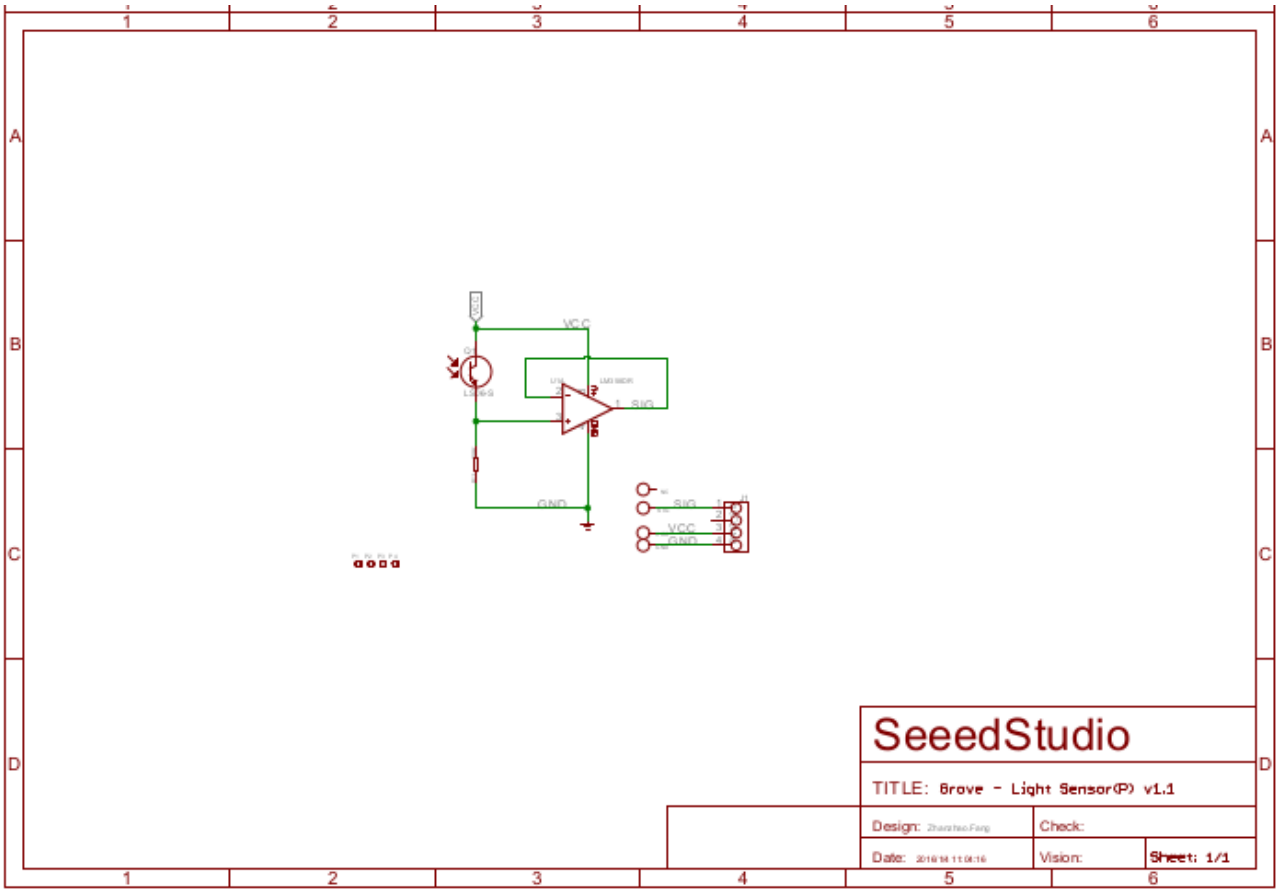
Anexo 4-Esquema placa Raspberry PI



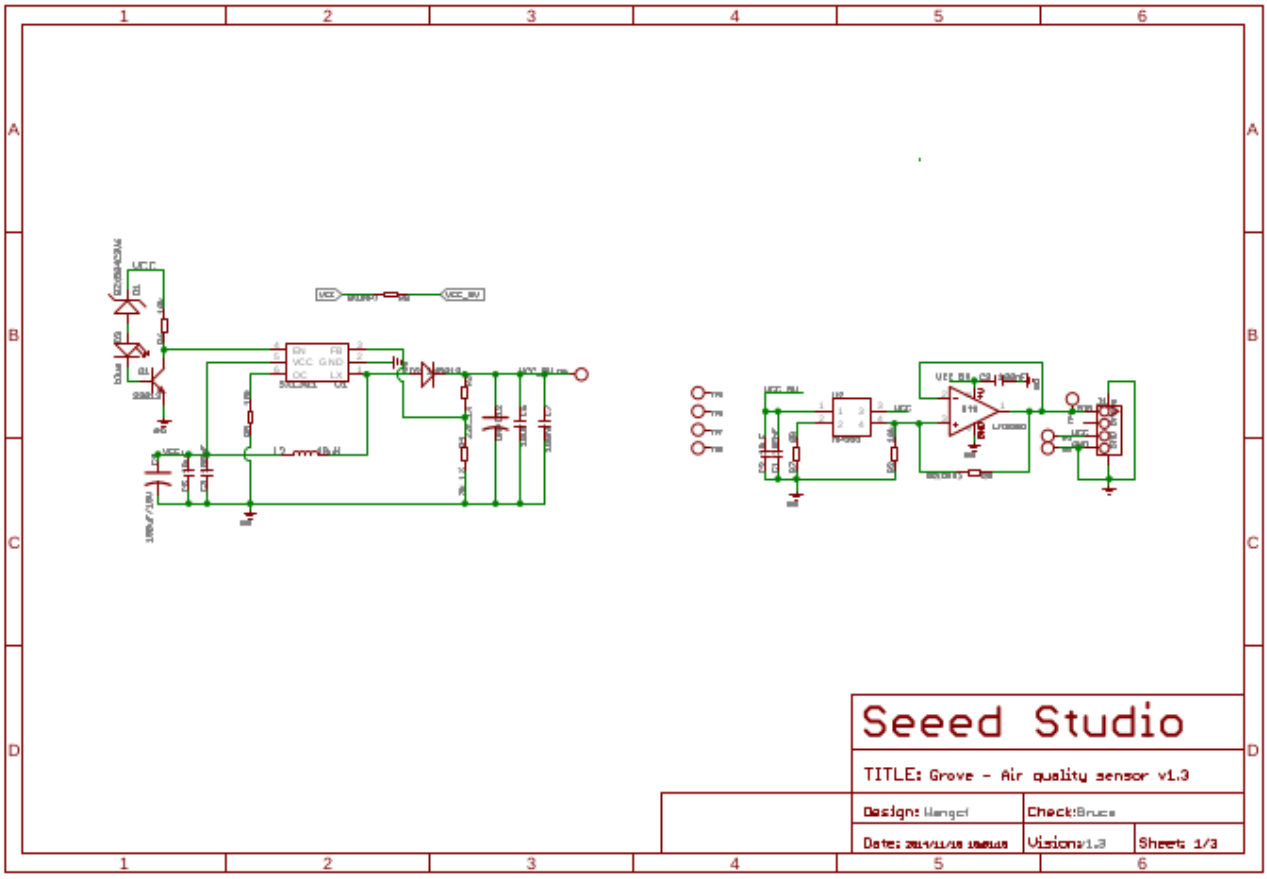
Anexo 5-Esquema placa GrovePi



Anexo 6-Esquema Grove Light sensor



Anexo 7-Esquema Grove Air Quality sensor



Anexo 8-Esquema Grove button