Escuela Politécnica Superior de Jaén

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN
ESCUELA POLITÉCNICA SUPERIOR DE JAÉN

TRABAJO FIN DE GRADO

DESARROLLO DE LABORATORIO WEB (WEBLAB) BASADO EN SCORM CON CAPACIDAD DE INTERACCIONAR CON LMS

Alumno: Alberto Vela Ruiz

Tutoras: Prof. D. Elisabet Estévez Estévez
Prof. D. Silvia Satorres Martínez
Dpto: Ingeniería Electrónica y Automática
Área: Ingeniería de Sistemas y Automática

Septiembre, 2015

Universidad de Jaén

Escuela Politécnica Superior de Jaén

Departamento de Ingeniería Electrónica y Automática

Área de Ingeniería de Sistemas y Automática

Doña Elisabet Estévez Estévez y Doña Silvia Satorres Martínez tutoras del Trabajo Final de Grado titulado: Desarrollo de Laboratorio Web (WebLab) basado en SCORM con capacidad de interaccionar con LMS, que presenta Juan Francisco Vico Zafra, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, 1 de Septiembre de 2015

El alumno:		Las tutoras:
Alberto Vela Ruiz	Elisabet Estévez Estévez	Silvia Satorres Martínez

"*Saber que se sabe lo que se sabe y*
que no se sabe lo que no se sabe;
he aquí el verdadero saber ."

Confucio

Agradecimientos

A mi familia y amigos.
A Elisabet Estévez, directora de este Trabajo,
por las facilidades brindadas en
la realización de este Trabajo.
A Alonso Ruano, por su total disposición
y las muy útiles sugerencias aportadas.
A Antonio Quesada, por toda la información aportada.
A Juan Francisco Vico como compañero de fatigas
en este proyecto, por su colaboración y dedicación.

Resumen

Las nuevas tendencias educativas sugieren el entretenimiento como medio para lograr un buen aprendizaje y la utilización de entornos altamente inmersivos. Un Laboratorio Remoto también conocido como WebLab es un sistema software y hardware utilizado para brindar al estudiante la posibilidad de realizar las prácticas de cualquier asignatura desde cualquier lugar conectado a Internet, utilizando hardware real situado físicamente en la Universidad.

La educación está cambiando, está digitalizándose. Internet se ha convertido en una herramienta más para el alumno, que no solo favorece la disponibilidad de los materiales educativos, sino que también permite y fuerza la colaboración entre instituciones y países, haciendo que la educación formal o informal sea una fuerza integradora más.

En áreas científico-tecnológicas los alumnos no solo necesitan acceder a contenidos, también necesitan acceder a laboratorios. Ahora bien, el diseño y mantenimiento de laboratorios presenciales es caro y complicado, lo que frena su desarrollo y uso en los centros educativos, especialmente en países con recursos limitados.

En un principio, y como solución a su falta de disponibilidad, se procedio a simular los laboratorios mediante herramientas de programación. Pero desde la aparición de Internet la comunidad investigadora se ha esforzado en digitalizar estos laboratorios, remotizándolos, permitiendo que los estudiantes accedan a laboratorios reales a través de Internet, donde la conversación fluye a través de la red. Una vez que estos laboratorios están disponibles en Internet, es posible compartirlos con otras entidades educativas, como cualquier otro recurso digitalizado en la red.

En este caso se realizará un laboratorio virtual y un laboratorio remoto sobre el modelado del control de un invernadero con el objetivo de acercar a los alumnos al desarrollo de las plantas especialmente en el concepto de la fotosíntesis.

Descriptores

WebLab, Laboratorio Remoto, Laboratorio Virtual, Enseñanza Remota, Ingeniería Remota, e-Learning, Arduino, Invernadero, Fotosíntesis, Clorofila, Respiración, Plantas.

Índice.

Índice de figuras.

Índice de tablas.

Índice de bloques de código fuente.

1 INTRODUCCIÓN

Cumpliendo con el plan de estudios del Grado en Ingeniería Electrónica Industrial se realiza el presente Trabajo Fin de Grado, para así culminar el proceso de formación. Las nuevas tecnologías e Internet han abierto nuevas posibilidades para la formación a distancia, también conocida bajo el término e-Learning, como son los laboratorios virtuales y remotos. Estos laboratorios permiten experimentar y explorar el comportamiento de un sistema, mediante la simulación y el control remoto del mismo. De esta forma se pueden comprender mejor los conceptos estudiados.

El trabajo que se presenta, es el Desarrollo de Laboratorio Web (WebLab) basado en SCORM (Sharable Content Object Reference Model) [18] con capacidad de interaccionar con LMS (Learning Management System), y está englobado dentro de un trabajo más amplio, que se está desarrollando en el departamento de Ingeniería Electrónica y Automática de la Escuela Politécnica Superior de Jaén en colaboración con el departamento de Didáctica de las Ciencias Experimentales integrado en la Facultad de Humanidades y Ciencias de la Eduacación de la Universidad de Jaén. Dicho trabajo consiste en la creación de un laboratorio virtual y un laboratorio remoto, es decir un laboratorio híbrido que permita a un usuario simular el comportamiento de un proceso y posteriormente realizar la ejecución remota del proceso para comparar así, los datos reales obtenidos con los simulados. Hay que distinguir entre dos tipos de usuarios: el profesor, administrador y supervisor del laboratorio y finalmente el alumno que se encargará de llevar a cabo el control del proceso para conseguir que el sistema se comporte de manera deseada. La realización del laboratorio virtual se está implementando mediante la herramienta EJS (Easy Java Simulations). Además todo este trabajo se está desarrollando bajo el estándar SCORM, para que así pueda ser utilizado en diferentes plataformas e-Learning existentes.

Mediante el presente TFG se pretende diseñar un software de comunicaciones para la gestión remota de equipos docentes de control de procesos. Se implementará un laboratorio remoto para un invernadero construido a escala que permitirá comparar prácticamente los resultados obtenidos en el laboratorio virtual. También se deberá poder visualizar todo el proceso mediante una cámara IP.

Se personalizará la interacción LMS-alumno mediante el uso de las comunicaciones SCORM-LMS, para ello se usarán el paquete Java scormRTE, usado para simplificar la programación de aplicaciones JAVA y la obtención de Applets.

La estructura que va a seguir esta memoria es la siguiente:

- En el apartado 2 se hablarán de los objetivos que se persiguen.
- El apartado 3 irá dedicado al estado del arte y el contexto.
- En el apartado 4 se recogen los aspectos referentes al desarrollo del proyecto.
- El apartado 5 engloba el código programado para Arduino.
- El apartado 6 contiene información acerca de la instalación y desarrollo del servidor XAMPP.
- En el apartado 7 se encontrarán los códigos realizados en NetBeans.
- El apartado 8 recoge el desarrollo de la interfaz de los applets en EJS.
- En el apartado 9 se comenta brevemente el desarrollo del bloque SCORM.
- En el apartado 10 se encuentran las conclusiones obtenidas y posibles mejoras.
- Los apartados 11 y 12 recogen los pliegos de condiciones técnicas y legales respectivamente.
- El apartado 13 engloba el presupuesto del proyecto.
- En el apartado 14 se encuentran todas las referencias y bibliografías utilizadas.
- En el apartado 15 se recogen una serie de anexos de este proyecto.

2 OBJETIVOS

En este capítulo se indicará en cierto detalle en qué consistirá el proyecto, cuáles son sus objetivos, y qué resultados pretenden obtenerse.

2.1 Objetivos

El objetivo principal de este proyecto es realizar un modulo de aprendizaje SCORM en el que se ejecutarán dos applets. Se desarrollarán dos aplicaciones mediante EJS, una para el laboratorio virtual y otra para el laboratorio remoto. En el laboratorio virtual se simulara el crecimiento de la eficiencia fotosintética, para 10 plantas escogidas para el caso, en función de la intensidad lumínica que les afecte . En el laboratorio remoto se podrá interaccionar con un invernadero construido a escala para ver como reaccionar ante la acción de diferentes

actuadores. Para ello se han instalado un conjunto de sensores conectados a un microcontrolador que se encargar de recoger datos en todo momento. Además se dispone de una cámara para ver la imagen del invernadero en todo momento. Se trata en resumen del desarrollo de un Web-Lab híbrido.

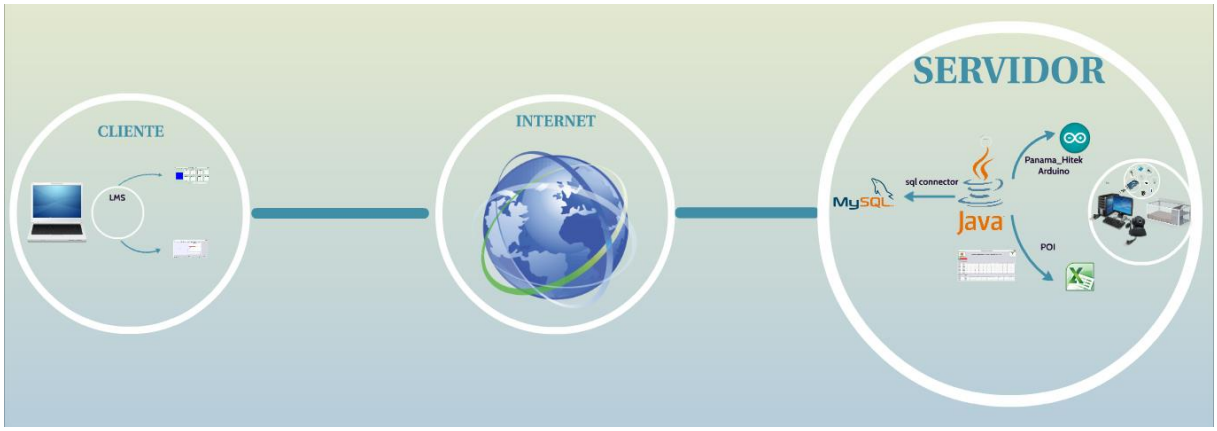


Figura 2-1 Esquema WebLab híbrido

2.1.1 Objetivos Socioeconómicos

En disciplinas técnicas y científicas, la realización de experimentos en entornos reales es fundamental para consolidar los conceptos adquiridos en las aulas teóricas. Sin embargo, a causa de distintos motivos, los laboratorios reales no siempre están disponibles, lo cual sin duda, supone restricciones en el aprendizaje. Afortunadamente, las nuevas tecnologías basadas en Internet, pueden ser utilizadas para mejorar la accesibilidad a los experimentos.

Por otro lado, el gasto requerido para la adquisición y mantenimiento de un laboratorio dentro de un centro educativo para su utilización puntual es a menudo difícilmente amortizable. Gracias a los laboratorios remotos, este coste puede ser compartido entre varias instituciones y siempre se gozará de una disponibilidad total del laboratorio.

2.1.2 Objetivos Tecnológicos

El principal objetivo tecnológico del presente proyecto es el desarrollo del software y la instalación del hardware necesario para implementación de un WebLab híbrido. Una herramienta de estas características implica unos importantes retos tecnológicos y científicos que se pueden dividir en cuatro áreas:

OT1. **Implementación de una plataforma de gestión para el despliegue de laboratorios remotos mediante hardware de bajo coste.**

Desplegar un RLMS (Remote Laboratory Management System) en una plataforma embebida de bajo coste (Arduino Mega 2560), proporcionando todas las herramientas que permitan la administración, configuración y parametrización del mismo.

Como se ha comentado anteriormente este laboratorio remoto consiste en el control de un invernadero, para ello se apoya en las tareas de control y monitorización que realiza un microcontrolador Arduino sobre un invernadero. El Arduino se encarga de recuperar datos que generan unos sensores y, en función de los mismos, activar/desactivar una serie de actuadores. Además se encarga de mantener comunicaciones con un sistema servidor establecido en un PC.

OT2. **Desarrollo de las comunicaciones entre todos los subsistemas**

Desarrollo de todo el sistema de comunicaciones que permita la interacción entre los diferentes subsistemas necesarios para el desarrollo de un laboratorio remoto basado, sí es posible, en hardware y software libre.

Una parte imprescindible en las telecomunicaciones es poder realizar una labor a distancia sin la necesidad de encontrarse en un emplazamiento físico predeterminado, es decir, dar una herramienta a las personas para que puedan interactuar con algo, que para ellos es conocido, pero que de otro modo les impediría desplazarse debido a la necesidad de permanecer continuamente a la expectativa de ciertos resultados.

En este caso, la lógica a la hora de desarrollar esta herramienta sería:

- Leer los datos de los sensores.
- Guardar en un registro.
- Recuperar los datos del registro.
- Modificar los parámetros de trabajo del sistema.

Por último, nombrar las comunicaciones que se realizán con el LMS para intercambiar los datos del alumno y mandar el resultado final del desarrollo del WebLab.

OT3. Desarrollo de la interfaz de usuario.

Desarrollo de un interfaz de usuario mediante software EJS (Easy Java Simulations) que permita la experimentación conjunta y simultánea de profesores y estudiantes en una misma sesión.

Para poder monitorizar de forma remota, cómoda y eficiente se ha programado un cliente EJS que pueden acceder a la base de datos para modificar los parámetros que gobiernan el funcionamiento del sistema y observar el funcionamiento del mismo de forma gráfica. El acceso a la base de datos se logra gracias al paquete JAVA SRA desarrollado específicamente para este proyecto, este paquete permite conectarse con la base de datos para leer y modificar sus registros facilmente.

OT4. Creacción de una base de datos

Desarrollo de todo el software necesario para la gestión y almacenamiento de todos los datos recogidos por los sensores aún y cuando no hay ningún cliente ejecutando la aplicación.

OT5. Creacción de un módulo de aprendizaje SCORM

Programación de un módulo de aprendizaje al que finalmente accederá el alumno y donde se encontrarán insertados las applets de ambos laboratorios desarrollados.

3 ESTADO DEL ARTE Y CONTEXTO

En este contexto, el presente proyecto supone una labor de integración de tecnologías pertenecientes a distintas áreas de la ingeniería. Como fase previa al análisis del proyecto, se procede a estudiar el Estado del Arte de los tres campos involucrados: los WebLabs, la tecnología y el software usado.

3.1 Los WebLabs.

Los centros de enseñanza técnica tienen laboratorios para permitir que los estudiantes puedan llevar a cabo ejercicios prácticos que complementen los conocimientos adquiridos en las clases teóricas. En muchas ocasiones, estos laboratorios están equipados con dispositivos que el estudiante no tiene por qué poseer, principalmente por el precio o el tipo de equipamiento. Un laboratorio remoto es un sistema hardware y software que permite utilizar estos dispositivos reales desde cualquier distancia conectada por una red informática, como por ejemplo Internet.

3.1.1 Evolución

Aunque la educación a distancia tiene sus orígenes a mediados del siglo XIX, ha sido la gran revolución tecnológica causada por el nacimiento de Internet a comienzos de los años 90 la que ha propiciado la expansión y la llegada de la educación a distancia a casi cualquier parte del mundo. Con este nuevo modelo educativo surge la necesidad no solo de poner al alcance del alumno contenidos teóricos, sino también contenidos prácticos, lo que ha favorecido el desarrollo y despliegue de los laboratorios remotos como herramienta de aprendizaje para facilitar el acceso a distancia de los contenidos y elementos prácticos de las materias.

Los primeros cursos a distancia de los que se tiene constancia tuvieron lugar en 1840 cuando Isaac Pitman, el inventor de la taquigrafía, tuvo la idea de comenzar a dar cursos por correspondencia. Poco tiempo después, la Universidad de Londres fue el primer centro universitario en el mundo en ofrecer cursos a distancia a través de su External System, establecido en el año 1858, el cual, a pesar del éxito alcanzado, el tipo y el número de cursos ofrecidos, permaneció limitado a unas pequeñas áreas de conocimiento, principalmente debido al poco grado de interacción entre el instructor o profesor y el estudiante, ya que en

aquella época, el intercambio de materiales únicamente era posible a través de cartas manuscritas, cuya entrega dependía siempre del servicio postal de correos.

En la segunda mitad del siglo XX, el panorama de los cursos a distancia comenzó a cambiar con la fundación en 1969 de la United Kingdom's Open University (OU), la cual ofrecía cursos en los cuales se utilizaban unas técnicas de enseñanza mixtas basadas en cursos presenciales y a distancia. A pesar de tener un funcionamiento similar a los cursos ofrecidos por la Universidad de Londres, en este caso los cursos eran mucho más diversos y contaban con textos y materiales cuidadosamente confeccionados así como de grabaciones de audio y video, complementadas con emisiones abiertas de radio y televisión. Adicionalmente, se organizaban sesiones a través de comunicaciones telefónicas en las que participaban el profesor y un alumno o bien un grupo de alumnos, por lo que podían mantener una comunicación fluida entre ellos.

Sin embargo, no fue hasta comienzos de la década de los 90 cuando se introdujo el concepto de e-Learning y tuvo lugar la aparición de los primeros laboratorios remotos. La idea del e-Learning (anglicismo de aprendizaje electrónico) estaba basada en el uso de un nuevo y potente medio de comunicación: Internet, el cual abría las puertas por primera vez a una interacción a gran escala entre el estudiante y el profesor. Internet en un primer momento facilitó la virtualización de los contenidos de los libros, haciendo posible su consulta a través de la red desde cualquier parte del mundo y en cualquier momento. Así comenzaron a utilizarse las herramientas tecnológicas asociadas a Internet para desarrollar soluciones que permitieran realizar y mejorar el acceso al conocimiento y al aprendizaje. Internet se afianzó como una herramienta global para la diseminación de la información y un medio para la colaboración y la interacción entre las personas y las computadoras sin importar su localización geográfica.

El e-Learning es una disciplina bien conocida desde finales de los años 80. Desde aquellos comienzos universitarios y no comerciales se ha pasado a una industria de la didáctica, sobre todo en lo que se refiere a tecnología. Por otra parte, siempre hubo empresas que fabricaban equipos didácticos para prácticas universitarias o de otro tipo. A veces estos equipos eran exactamente los mismos que los usados en la industria, pero otras muchas eran equipos específicos para laboratorios específicos, como ALECOP, Ingeniería de Microsistemas Programados, Feedback , etc.

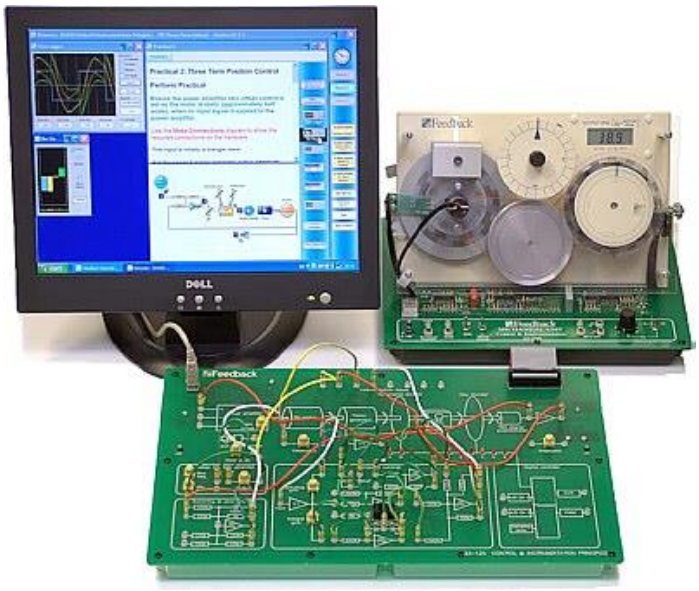


Figura 3-1 Feedback 33-100

A mediados de los 90 aparece el primer laboratorio remoto accesible por internet (proyecto PEARL). De esta manera se unen el e-Learning (actividades educativas a distancia) y los experimentos.

Gracias a Internet, la educación a distancia ha ido convirtiéndose poco a poco en una alternativa y en un complemento a la educación presencial de los métodos tradicionales de enseñanza, dando lugar al concepto de aprendizaje semipresencial o más conocido en su término anglosajón blended-learning o simplemente b-learning.

Por consiguiente y gracias a las tecnologías actuales que posibilitan un acceso a la información de una forma cada vez más rápida y ubicua, la educación a distancia y las plataformas de aprendizaje online están aumentando rápidamente. Como ejemplo, en el año 2004 la matriculación en cursos a distancia en las universidades de Estados Unidos aumentó en 2,4 millones de alumnos, un 18% más que en el año anterior.

El desarrollo tecnológico de laboratorios remotos ha presentado avances significativos en los últimos años a tenor de los numerosos proyectos de investigación que se desarrollan en este ámbito, auspiciados por la Unión Europea, como por ejemplo las iniciativas DYNACORE, MARVEL, PEARL, CYBERLAB. El impacto de las tecnologías de la información y la comunicación ha propiciado la creación de redes educativas, como es el caso de la European Schoolnet, la red de excelencia PROLEARN o las iniciativas directamente enfocadas en el desarrollo y difusión de la experimentación remota, como es el caso de la red australiana de laboratorios remotos Labshare y el consorcio de universidades unidas en torno

al proyecto VISIR. A nivel internacional destacan el Global Online Laboratory Consortium, la Red de Laboratorios Remotos RexNet y especialmente la plataforma iLab desarrollada por el MIT. También se encuentran en la Red, repositorios web de laboratorios remotos como es el caso del proporcionado por el proyecto europeo LiLa y Lab2go, cuyo objetivo es proporcionar un entorno web común para compartir recursos y experiencias relativas a la experimentación remota, tanto para desarrolladores como profesores y estudiantes de diversas áreas de conocimiento.

Se puede marcar el año 2000 como el año de despegue de los laboratorios remotos en el mundo universitario, aunque aún siguen siendo poco profesionales en diversos aspectos.

3.1.2 Definición

Se pueden encontrar numerosas definiciones de lo que es un WebLab tal y como se constata en la tabla, elaborada en base a los investigadores de referencia en esta área de conocimiento. Como se puede ver, no existe una definición que sea universalmente aplicable y entendible cuando se habla de conceptos como laboratorio remoto, WebLab (o web-lab), laboratorio virtual, tele-lab y online lab. Es más, muchas veces estos términos son usados como sinónimos e incluso en algunas publicaciones no queda claro cuales son las diferencias entre todas estas herramientas.

Tabla 3-1 Definiciones WebLab

Definiciones
Son plataformas basadas en Internet que proporcionan acceso a bancos de pruebas experimentales reales, no disponibles en el lugar físico en el que se encuentra el usuario. [16]
Son laboratorios que utilizan datos reales obtenidos desde la experimentación realizada a través de un interfaz web. [17]
La idea general de un laboratorio remoto es habilitar el acceso a los laboratorios físicos o a puestos desde lugares distantes a los mismos, haciendo uso de una infraestructura de comunicación adecuada. [18]
Un laboratorio remoto se corresponde a aquella situación en la que el control y la observación de los instrumentos y del objeto bajo prueba del experimento son proporcionados por un ordenador, el cual es accedido remotamente gracias a una red de comunicaciones específica. [19]

Los laboratorios remotos son instalaciones experimentales que se pueden acceder a través de Internet, permitiendo a los estudiantes y educadores llevar a cabo experimentos desde cualquier lugar del mundo en cualquier momento. [20]
Un laboratorio remoto es un sistema de apoyo a la docencia basado en un laboratorio virtual y de tele-presencia accesible a través de una red basada en protocolos TCP/IP que permita al alumno practicar de una forma lo más similar posible a como si estuviese en las dependencias del laboratorio, dándole la posibilidad de manejar las simulaciones o interactuar con las plantas reales. [21]

3.1.3 Tipos de WebLabs.

Tabla 3-2 Caracterización de laboratorios

	Local	Remoto
Físico (Real)	**Laboratorio Manual**	**Laboratorio Remoto**
Virtual (Simulado)	**Laboratorio Virutal**	**Laboratorio Virtual Remoto**

Así pues y con el objetivo de diferenciar los tipos de laboratorios mencionados previamente se va a realizar una comparativa entre laboratorio manual y virtual (simulado) y laboratorio local y distribuido. Los criterios siguientes permitirán establecer una primera orientación respecto a la naturaleza del laboratorio (físico o virtual) y el modo de acceso de los alumnos (local o remoto), lo que permite analizar la tabla y extraer las siguientes reflexiones:

- Dependiendo del tipo de interacción del usuario con el experimento, ésta puede ser de dos tipos:
    - El usuario puede controlar directamente los dispositivos y la instrumentación, iteración que se corresponde con la llevada a cabo en el laboratorio tradicional.
    - El usuario controla los dispositivos y la instrumentación a través de la interfaz proporcionada por un ordenador, utilizando instrumentación virtual o entornos mixtos virtuales-reales.

- Atendiendo a la naturaleza de los experimentos que se pueden llevar a cabo en el laboratorio, éstos podrán estar basados en la utilización de dispositivos y equipos físicos o en modelos simulados de los dispositivos o equipos físicos.

- Y finalmente, se contemplan dos tipos de situaciones para la localización de los usuarios y los laboratorios, ambos en la misma localización o bien en diferentes localizaciones, es decir, separados geográficamente.

Los laboratorios remotos pueden ser clasificados en tres áreas dependiendo del tipo de control:

- Instrumentación remota: El laboratorio remoto consiste en uno o varios experimentos en los cuales los usuarios sólo pueden activar sus entradas (switches virtuales, generadores de señales. . .) y ver sus salidas reales o virtuales a través de una webcam (LEDs, señales en un osciloscopio. . .). Dos ejemplos de este tipo de laboratorio remoto son el Remote Access Laboratory de la Universidad de Limerick y el laboratorio remoto del Instituto Tecnológico de Blekinge [22].

- Parametrización remota: La principal diferencia entre este laboratorio remoto y el anterior es que en éste el usuario puede cambiar los parámetros de control para modificar la lógica del sistema. Un ejemplo de este Laboratorio Remoto es el Automatic Control Telelab de la Universidad de Siena [23]. En este laboratorio remoto, el usuario puede manipular algunos parámetros (control de posición, velocidad, nivel o flujo) y ver los resultados a través de una webcam.

- Lógica de Control Remota: En este caso, el usuario puede cambiar tanto la lógica como la parametrización del sistema. Un ejemplo simple sería un modelo didáctico básico (LEDs, 7 segmentos. . .) controlados por un CPLD o un FPGA que ha sido programado por el estudiante. Aquí el riesgo es más alto, puesto que el estudiante podría destruir el sistema debido a un fallo del programa, ya que el estudiante tiene el control total de las variables del experimento. Ejemplos de este tipo de laboratorio remoto son el Shell & Tube Heat Exchanger Experiment del MIT o el Remote Laboratory Project del Instituto Tecnológico de Blekinge [22].

Tabla 3-3 Tipos de laboratorios remotos según control

Tipo de Control Remoto	Nivel de Control sobre el experimento
Control de Instrumentación	Sólo es posible cambiar las entradas y ver las salidas
Control de Parametrización	Solo es posible cambiar las entradas y parámetros del sistema
Control de Lógica	El cliente puede cambiar tanto las entradas como los parámetros como la lógica del sistema

3.1.4 Arquitectura de un laboratorio remoto

Un laboratorio remoto consiste en una plataforma tecnológica que puede permitir a un alumno remoto a través de su ordenador comunicarse, observar y/o controlar a distancia un experimento situado en lejos de él, haciendo uso de Internet como medio de comunicación.

Para hacer posible la comunicación entre el usuario y el experimento, la arquitectura de un laboratorio remoto está basada en el paradigma cliente-servidor, en donde las tecnologías asociadas a Internet son el centro de los desarrollos actuales. Incluso utilizando diferentes alternativas tecnológicas, la mayoría de las soluciones están fundamentadas en esta simple arquitectura. En un extremo de la comunicación se sitúa el experimento junto con su equipamiento e instrumentos y un ordenador que permite por una parte, el control del experimento y los instrumentos y por otra, proporciona conexión a Internet actuando como un servidor del laboratorio en un extremo de la comunicación. En el otro extremo, el usuario utilizará una aplicación cliente (en la mayoría de las situaciones un navegador web), que le ofrecerá la posibilidad de controlar y monitorizar remotamente el experimento.

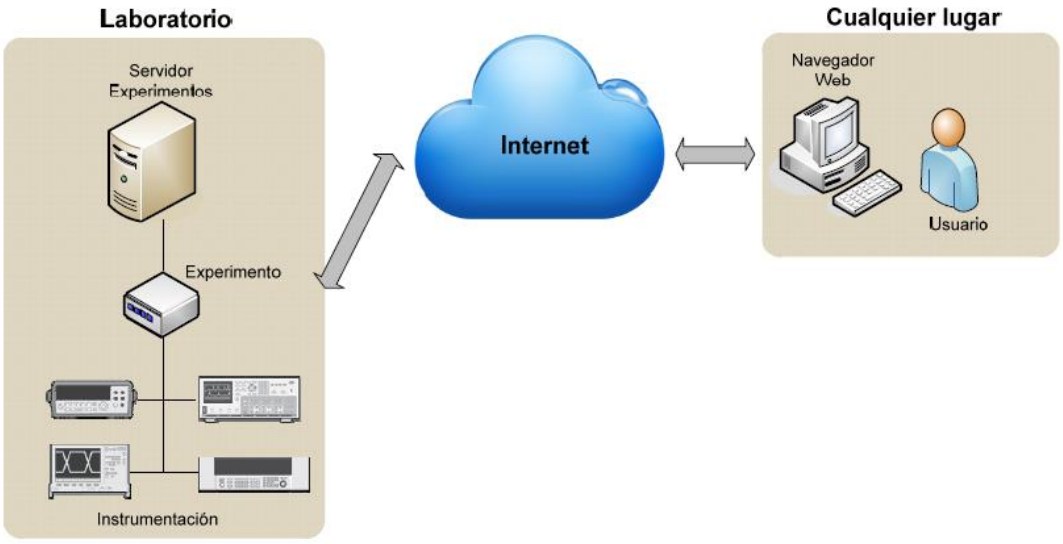


Figura 3-2 Arquitectura básica cliente-servidor de un laboratorio remoto

En algunos casos la aplicación cliente de un laboratorio remoto puede integrarse bien en los denominados Learning Management Systems (LMS) o bien en los más actuales entornos denominados Personal Learning Enviroments (PLE), lo que facilita la implicación del laboratorio remoto en el proceso de aprendizaje al estar su acceso embebido en un espacio dedicado completamente al estudio de una materia concreta.

Por lo tanto en la arquitectura de un laboratorio remoto se denominará cliente al software a través del cual el usuario accede e interactúa con los experimentos disponibles. En función de la actividad a desarrollar, el cliente facilitará al estudiante las aplicaciones o servicios necesarios para, por ejemplo, enviar y/o leer un fichero al/desde el servidor, mostrar los resultados del experimento, permitir controlar y monitorizar los equipos del experimento o mostrar en tiempo real mediante un servicio de video y/o audio lo que sucede durante el ejercicio práctico.

Por consiguiente, el mejor cliente a proporcionar al usuario será aquel mediante el cual pueda realizar todas las actividades que llevaría a cabo en un laboratorio tradicional sobre el mismo experimento, sin ninguna restricción y a través de una aplicación estándar, como es un navegador de Internet. Para ello, las tecnologías de desarrollo deben permitir integrar en el navegador todas las aplicaciones y servicios necesarios para ello.

Análogamente, en la arquitectura de un laboratorio remoto se denomina servidor al software que se encarga del control y gestión del experimento. El servidor está en comunicación directa tanto con el cliente remoto como con el experimento y los instrumentos

que le rodean, por lo que es a través de él como accede el cliente al laboratorio. Así, y aunque el cliente es una parte importante del laboratorio remoto, es en el servidor donde más esfuerzos de diseño y desarrollo se concentran, siendo sus tareas principales:

- Gestión y administración de los usuarios del laboratorio. En un laboratorio remoto, al igual que en el tradicional, el profesor necesita poder realizar más acciones que los alumnos, como por ejemplo: puede crear sesiones de laboratorio con experimentos específicos, puede asignar materiales a cada sesión, asignar turnos a los alumnos, ver cuántas veces el alumno accede al laboratorio, etc.

- Autentificación, gestión de permisos, integridad y privacidad de las comunicaciones. El servidor deberá estar dotado de las herramientas y tecnologías necesarias para garantizar la seguridad en las comunicaciones, tanto de los datos intercambiados entre el cliente y el servidor, como proteger a este último de posibles ataques exteriores que pueden atentar contra los datos de los usuarios almacenados.

- Control y gestión de los experimentos disponibles en el laboratorio. El servidor es el encargado de gestionar la comunicación tanto con el experimento como con la instrumentación y demás componentes disponibles en el laboratorio, por lo que deberá implementar los protocolos de comunicación necesarios para el control de los mismos.

Siendo el cliente y el servidor los elementos principales de un laboratorio remoto, también son necesarios los sistemas de comunicaciones que intervienen en el proceso:

- Comunicación cliente-servidor: En la gran mayoría de los laboratorios remotos existentes la comunicación entre el alumno y los elementos sujetos a experimentación del laboratorio se realiza utilizando Internet como plataforma para el intercambio de información entre ambos extremos de la comunicación. Existen hoy en día multitud de tecnologías para implementar dicho enlace de comunicaciones, pero es importante recordar que cualquier transacción de información realizada a través de la World Wide Web está basada en el protocolo HTTP, por lo que la tecnología utilizada para la implementación de la comunicación entre el cliente y el servidor deberá tener en consideración los requisitos establecidos para ellos por HTTP.

- Comunicación servidor-experimento: Como se ha dicho anteriormente, entre las tareas del servidor se encuentra controlar los experimentos del laboratorio, los cuales pueden estar formados por el sistema bajo prueba así como la instrumentación necesaria para realizar los test y las medidas correspondientes al ejercicio. Para ello, el servidor debe disponer de un conjunto de algoritmos de control así como de un sistema de comunicaciones que permita por una parte ejecutar dicho control sobre los experimentos y los equipos, y por otra recoger los datos generados durante las pruebas realizadas y que serán enviados al cliente del laboratorio remoto para ponerlos a disposición del usuario.

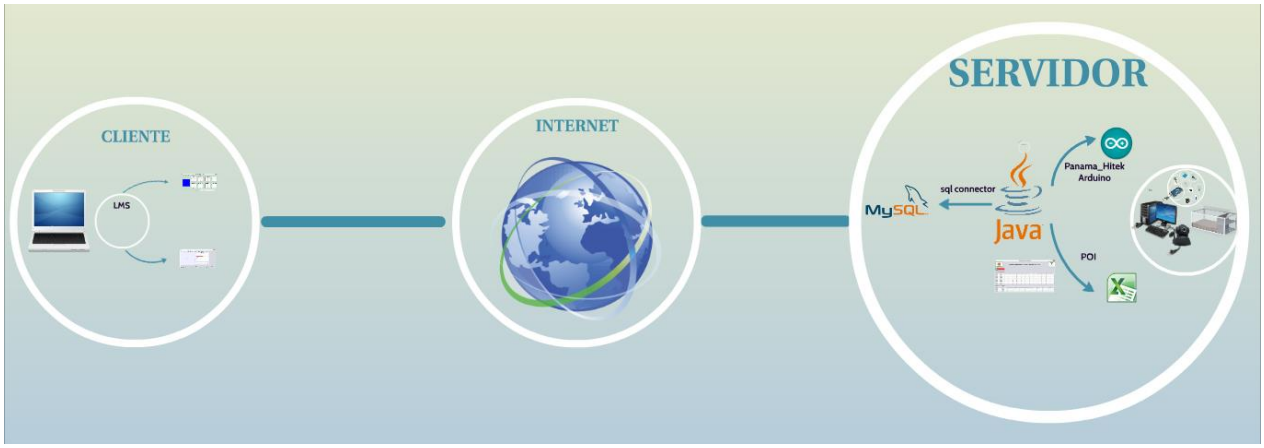


Figura 3-3 Arquitectura del WebLab desarrollado

3.1.5 Características

En este apartado se van a desarrollar tres grandes preguntas que representan las características principales de un laboratorio remoto. Todo laboratorio remoto debe ser capaz de dar respuesta a las siguientes preguntas:

- ¿Es el laboratorio remoto una herramienta didáctica?
- ¿Se puede considerar universal?
- ¿Es el laboratorio remoto tecnológicamente avanzado?

Si se desea que el laboratorio remoto sea una herramienta que esté presente en la vida diaria de la universidad como una utilidad docente más, las preguntas anteriores deberán tener unas respuestas totalmente afirmativas con una serie de pruebas.

En apartados sucesivos se van a ir analizando cada una de estas preguntas, desarrollando cada una de las preguntas, tratando de desgranar las consecuencias que cada una conlleva.

- **Docencia**

A pesar de que se ha venido desarrollando un mercado libre en el ámbito de la educación como resultado de las facilidades que proporciona Internet, las mayores discusiones, cambios y debates sobre la adaptación de la educación al nuevo ciberespacio tienen lugar dentro de las propias instituciones de los sistemas educativos. En el caso de la educación en ingeniería, esta mejora en el sistema educativo tiene un aspecto especial y peculiar debido a la necesidad de añadir al conocimiento teórico las aplicaciones prácticas del mismo.

Del párrafo anterior se extrae que claramente la enseñanza teórica de una materia en el campo de la ingeniería debe llevar ligada una aplicación práctica de los conocimientos teóricos. De este modo, el laboratorio remoto diseñado a tal efecto debe cumplir íntegramente los objetivos planteados por la asignatura.

Por otro lado, tal y como determinaban algunas de las ventajas de los laboratorio remotos, debe facilitar el trabajo del alumno. De este modo, el laboratorio remoto debe presentar una interfaz de usuario lo suficientemente sencilla e intuitiva para que al alumno no le suponga un esfuerzo extra interactuar con el laboratorio de forma remota. De este modo, el control de las entradas y la visualización de las salidas deben ser rápidos y fácilmente asimilables por el alumno.

De igual manera, el alumno debe tener la sensación de encontrarse en un laboratorio y poder interactuar con los dispositivos que allí se encuentren, tal y como si se encontrase físicamente delante de los equipos disponibles. Es decir, no ser un mero espectador del laboratorio remoto, tal y como sucede con algunos laboratorios virtuales, en los que el usuario únicamente observa una simulación de ciertos fenómenos.

Si se trata de un laboratorio remoto en el que se descarga un programa que se debe ejecutar en el servidor, se debe salvaguardar en todo momento la integridad del mismo, por lo que es conveniente que se establezcan ciertos mecanismos que comprueben que el código a ejecutar es seguro para el sistema (Firma en las aplicaciones Java).

Al tratarse de una herramienta educativa, podría estar involucrada dentro de una plataforma educativa del estilo de ILIAS (plataforma utilizada en la Universidad de Jaén), en la que junto con el laboratorio remoto propiamente dicho el alumno pudiera aprender más sobre la materia en cuestión, disponiendo de esta forma de otros recursos didácticos, como manuales, simulaciones, tutoriales (módulos SCORM)

El objetivo principal de un experimento educativo es demostrar fenómenos físicos a los estudiantes, para ayudarles a entender los conceptos y las reglas subyacentes. Durante los experimentos, los estudiantes también ganan experiencia práctica y adquieren habilidades profesionales.

- **Universalidad**

Con esta pregunta tan general se pretende lanzar varias interrogaciones referentes tanto a la accesibilidad y disponibilidad del laboratorio remoto como a la disponibilidad de recursos.

El uso del laboratorio remoto trata de presentarse como un apoyo a la utilización de los laboratorios convencionales, por lo que su disponibilidad debe ser mayor que éstos. Un laboratorio remoto debe estar disponible las 24 horas del día y los 365 días del año, permitiendo por tanto acceso a los experimentos siempre que el alumno lo requiera, sin límite de horarios. Para ello el laboratorio remoto debe permanecer siempre activo y no quedarse obsoleto.

Por otro lado, el laboratorio remoto debería ser un sistema abierto a todo el mundo, de tal manera que cualquiera pudiera conectarse al laboratorio e interactuar con él. Es cierto que se debe proceder a tener un registro de los accesos, pero estós no deberían estar limitados a los alumnos y profesores de la facultad desarrolladora del laboratorio remoto.

Si se cumple la característica descrita en el apartado anterior, el laboratorio remoto debería poder ser usado por personas de muy remota procedencia, por lo que debería estar disponible en varios idiomas.

- **Tecnología**

Esta pregunta sobre las características del laboratorio remoto hace clara referencia a las tecnologías con las que se ha desarrollado el laboratorio remoto y a las tecnologías vinculadas de un modo u otro con el laboratorio remoto: tecnologías con las que se accede al laboratorio remoto y tecnologías a las que se accede a través del laboratorio remoto.

Una de las principales características que debería cumplir el laboratorio remoto es ser multiplataforma, entendiendo de esta manera que el laboratorio remoto pueda ser accedido desde cualquier sistema operativo, o al menos desde los más extendidos (Windows, Mac OS X, GNU/Linux,etc). De esta manera, se estaría también asegurando una universalidad del laboratorio remoto.

Una de las características que se citaban en el primer apartado de este punto era la facilidad de acceso por parte del usuario al laboratorio remoto. Uno de los mayores inconvenientes que encuentra el usuario es tener que instalar ciertos programas en su equipo. Por tanto, si se consigue desarrollar un cliente que sea smart client (no requiere la instalación de software adicional, como ActiveX o Flash), se lograría una mayor simplicación del acceso al laboratorio remoto y con ello una mayor satisfacción del usuario.

Por otro lado, se debe evaluar el medio de comunicación por la que el usuario tiene acceso al hardware: RS-232, TCP/IP, XML, red móvil , etc.

Como resumen, la siguiente tabla agrupa las características que debería tener un laboratorio remoto.

Tabla 3-4 Características de un laboratorio remoto

Didáctico	Universal	Tecnológico
Cumplir los objetivos del curso	Activo 24 horas 365 días	Multiplataforma
Facilitar el trabajo del alumno	Sistema abierto a todo el mundo	Cliente inteligente
Integar al alumno en el experimento	Disponible en varios idiomas	Facilitar el acceso desde sistemas móviles
Integrado en una plataforma educativa	Facilitar el acceso a diferentes dispositivos	Buena calidad del vídeo
Facilitar el acceso al dispositivo	Facilitar el acceso a diferentes laboratorios remotos	

3.1.6 Ventajas

El diseño y el uso de un laboratorio remoto en una facultad de ingeniería proporcionan los siguientes beneficios:

- Mejor aprovechamiento de los equipos. Los equipos del laboratorio están disponibles para los alumnos las 24 horas del día del durante los 365 días del año.

- Organización de los laboratorios. No es necesario mantener los laboratorios abiertos todo el tiempo, sólo es necesario mantener operativo el laboratorio remoto.

- Organización del trabajo del estudiante. Haciendo uso del laboratorio remoto, tanto el alumno como el profesor pueden organizar mejor sus tiempos, incluyendo los tiempos de clase.

- Aprendizaje autónomo. Los laboratorios remotos promueven el autoaprendizaje, fundamental en el nuevo Espacio Europeo de Educación Superior (EEES).

- Acercamiento a la sociedad. Los laboratorios remotos acercan los laboratorios al resto de la sociedad.

- Cursos a distancia. Los laboratorios remotos permiten la organización de cursos de ingeniería a distancia sin la necesidad de que los alumnos se encuentren presentes físicamente en el aula, evitando muchos de los problemas actuales. De esta manera se fomenta la multiculturalidad de los cursos.

- Integración de estudiantes discapacitados. Dado que todos los equipamientos hardware están controlados por un ordenador, pueden ser usados por estudiantes incapacitados utilizando software diseñado especialmente para sus necesidades particulares.

- Máximo rendimiento de las instalaciones de la universidad. El uso de laboratorio remotos evita los tiempos muertos en que el laboratorio no está siendo utilizado.

- Beneficios económicos. Evitando los tiempos muertos, se pueden obtener los mismos resultados minimizando la inversión necesaria en hardware. Y por otra

parte, se evita el coste que conlleva el mantenimiento de los laboratorios tradicionales.

La siguiente tabla resume la principales ventajas que presenta el uso de un laboratorio remoto, tanto para los alumnos y profesores como para la universidad.

Tabla 3-5 Ventaja de los laboratorios remotos

Estudiantes	Profesores	Universidad
Facilita el proceso de aprendizaje.	El profesor puede contactar con los estudiantes de forma autónoma.	El laboratorio está abierto 24 horas al día, 365 días al año.
Los estudiantes se sienten más motivados cuando controlan las prácticas según sus propias iniciativas.	El profesor puede incluir en sus clases ejemplos teóricos-prácticos.	Los laboratorios reducen sus necesdades de equipamiento y espacio.
Los estudiantes pueden planificar su programa de prácticas en el tiempo.	Se reduce el número de casos prácticos.	El personal de laboratorio no tiene porqué estar muy especializado.
Es un ejemplo de aprendizaje autónomo y autosuficiente	Permite al profesor controlar el sistema de forma remota	Reduce e incluso elimina los accidentes

3.2 Tecnología

3.2.1 Microcontrolador Arduino.

Arduino es una herramienta para hacer que los ordenadores puedan sentir y controlar el mundo físico. Es una plataforma de desarrollo de computación física (physical computing) de código abierto, basada en una placa con un sencillo microcontrolador y un entorno de desarrollo para crear software para la placa.

Se puede usar Arduino para crear objetos interactivos, leyendo datos de una gran variedad de interruptores y sensores y controlar multitud de tipos de luces, motores y otros actuadores físicos. Los proyectos de Arduino pueden ser autónomos o comunicarse con un programa (software) que se ejecute en tu ordenador (ej. Flash, Processing, MaxMSP, Java). El software de desarrollo es abierto y se puede descargar gratis.

El lenguaje de programación de Arduino es una implementación de Wiring, una plataforma de computación física, que a su vez se basa en Processing, un entorno de programación multimedia.

3.2.2 Sensores.

Los sensores son dispositivos capaces de medir magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, movimiento, pH, etc. Una magnitud eléctrica puede ser una resistencia eléctrica, una capacidad eléctrica (como en un sensor de humedad), una tensión eléctrica (como en un termopar), una corriente eléctrica (como en un fototransistor), etc.

Un sensor está siempre en contacto con la variable de instrumentación con lo que puede decirse también que es un dispositivo que aprovecha una de sus propiedades con el fin de adaptar la señal que mide para que la pueda interpretar otro dispositivo. En este proyecto se realiza la lectura de los sensores de humedad del terreno, humedad y temperatura ambiente, luminosidad (fotorresistencia), concentración de oxígeno, concentración de dióxido de carbono y nivel de agua.

3.2.3 Actuadores.

Un actuador es un dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado. Éste recibe la orden de un regulador o controlador y en función a ella genera la orden para activar un elemento final de control como, por ejemplo, un ventilador.

En la realización de este proyecto, al estar controlados los actuadores por la placa Arduino, se han utilizado actuadores eléctricos. La estructura de un actuador eléctrico es simple en comparación con la de los actuadores hidráulicos y neumáticos, ya que sólo requieren de energía eléctrica para su activación/desactivación. Como se utilizan cables eléctricos para transmitir electricidad y señales, es altamente versátil y prácticamente no hay restricciones respecto a la distancia entre la fuente de energía y el actuador.

Como se comentaba en la introducción a este proyecto, se han utilizado tres actuadores:

- Una bomba de agua sumergible para la realización de la función del riego del invernadero.
- Un ventilador para el proceso tanto de extracción de humedad excesiva en el invernadero como de enfriamiento del mismo.
- Iluminación artificial para forzar un nivel de luminosidad determinado y controlar la temperatura.

3.3 Software usado

En esta sección se describen aquellas herramientas que han ayudado e intervenido en el desarrollo de este proyecto.

3.3.1 Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente. El lenguaje Java se creó con cinco objetivos principales:

- Debería usar el paradigma de la programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería incluir por defecto soporte para trabajo en red.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Las applet Java son programas incrustados en otras aplicaciones, normalmente una página Web que se muestra en un navegador. En la realización de este proyecto, la programación Java ha sido muy utilizada, ya que se ha usado para el diseño del servidor, del SRA Packet y del ScormRTE Packet.

3.3.2 EJS

Easy Java Simulations (simulaciones sencillas en Java), también conocido como EJS, es una herramienta de autor creada en Java que ayuda a crear simulaciones interactivas en Java, habitualmente con fines de enseñanza o aprendizaje. EJS ha sido creado por Francisco Esquembre y es parte del proyecto Open Source Physics (Física de código abierto).

EJS es un programa de descarga libre que ayuda a crear otros programas más precisamente, simulaciones científicas. Ha sido diseñado para permitir a usuarios trabajar a un alto nivel conceptual, usando un conjunto de herramientas simplificadas y concentrando la mayoría de su tiempo en los aspectos científicos de la simulación, y pidiendo al computador que realice automáticamente todas las otras tareas necesarias pero fácilmente automatizables.

EJS crea aplicaciones Java que son independientes y multiplataforma, o applets que se pueden visualizar usando cualquier navegador Web (y por tanto ser distribuidos a través de Internet), que pueden leer datos a través de la red y ser controlados usando scripts (conjuntos de instrucciones) incluidos en las páginas HTML.

3.3.3 MySQL

MySQL es el servidor de bases de datos relacionales más popular, desarrollado y proporcionado por MySQL AB. MySQL AB es una empresa cuyo negocio consiste en proporcionar servicios en torno al servidor de bases de datos MySQL.

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de datos. La información que puede almacenar una base de datos puede ser tan simple como la de una agenda, un contador o un libro de visitas, ó tan amplia como la de una tienda en línea, un sistema de noticias, un portal o la información generada en una red corporativa. Para agregar, accesar y procesar los datos almacenados en una base de datos, se necesita un sistema de administración de bases de datos, tal como MySQL.

MySQL es un sistema de administración de bases de datos relacionales. Una base de datos relacional almacena los datos en tablas separadas en lugar de poner todos los datos en un solo lugar. Esto agrega velocidad y flexibilidad. Las tablas son enlazadas al definir relaciones que hacen posible combinar datos de varias tablas cuando se necesitan consultar datos. La parte SQL de "MySQL" significa "Lenguaje Estructurado de Consulta", y es el lenguaje más usado y estandarizado para accesar a bases de datos relacionales.

MySQL es Open Source. Open Source significa que la persona que quiera puede usar y modificar MySQL. Cualquiera puede descargar el software de MySQL de Internet y usarlo sin pagar por ello. Inclusive, cualquiera que lo necesite puede estudiar el código fuente y cambiarlo de acuerdo a sus necesidades. MySQL usa la licencia GPL (Licencia Pública General GNU), para definir qué es lo que se puede y no se puede hacer con el software para diferentes situaciones. Sin embargo, si uno está incómodo con la licencia GPL o tiene la necesidad de incorporar código de MySQL en una aplicación comercial es posible comprar una versión de MySQL con una licencia comercial.

El servidor MySQL fue desarrollado originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes y ha estado siendo usado exitosamente en ambientes de producción sumamente exigentes por varios años. Aunque se encuentra en desarrollo constante, el servidor MySQL ofrece hoy un conjunto rico y útil de funciones. Su conectividad, velocidad, y seguridad hacen de MySQL un servidor bastante apropiado para accesar a bases de datos en Internet.

El software de bases de datos MySQL consiste de un sistema cliente/servidor que se compone de un servidor SQL multihilo, varios programas clientes y bibliotecas, herramientas administrativas y una gran variedad de interfaces de programación (APIs). Se puede obtener también como una biblioteca multihilo que se puede enlazar dentro de otras aplicaciones para obtener un producto más pequeño, más rápido y más fácil de manejar. Para obtener información técnica más detallada, es necesario consultar la guía de referencia de MySQL.

3.3.4 Sharable Content Object Reference Model

SCORM es un conjunto de estándares y especificaciones que permite crear objetos pedagógicos estructurados.

Los sistemas de gestión de contenidos en web originales usaban formatos propietarios para los contenidos que distribuían. Como resultado, no era posible el intercambio de tales contenidos. Con SCORM se hace posible crear contenidos que puedan importarse dentro de sistemas de gestión de aprendizaje diferentes, siempre que estos soporten la norma SCORM.

Los principales requerimientos que el modelo SCORM trata de satisfacer son:

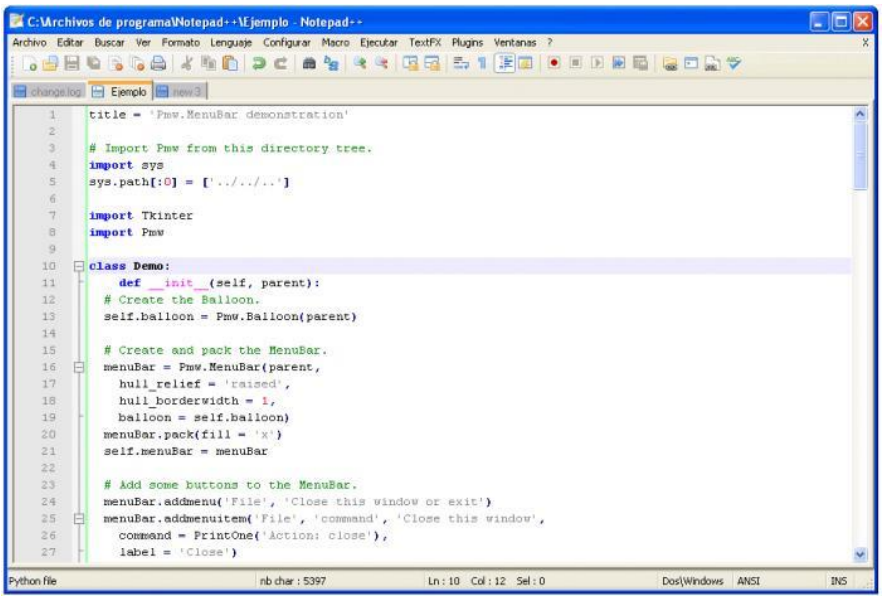
- Accesibilidad: Capacidad de acceder a los componentes de enseñanza desde un sitio distante a través de las tecnologías web, así como distribuirlos a otros sitios.
- Adaptabilidad: Capacidad de personalizar la formación en función de las necesidades de las personas y organizaciones.
- Durabilidad: Capacidad de resistir a la evolución de la tecnología sin necesitar una reconcepción, una reconfiguración o una reescritura del código.
- Interoperabilidad: Capacidad de utilizarse en otro emplazamiento y con otro conjunto de herramientas o sobre otra plataforma de componentes de enseñanza desarrolladas dentro de un sitio, con un cierto conjunto de herramientas o sobre una cierta plataforma. Existen numerosos niveles de interoperabilidad.
- Reusabilidad: Flexibilidad que permite integrar componentes de enseñanza dentro de múltiples contextos y aplicaciones.

3.3.5 Notepad++

Notepad++ es un editor de texto orientado a la edición de código fuente. Basado en Scintilla, está escrito en C++. Pretende ser una versión mucho más potente de notepad, pero manteniendo el máximo rendimiento y una interfaz simple y usable. Soporta pestañas, coloreado de sintaxis, búsqueda y reemplazo utilizando expresiones regulares, etc.

En el proyecto se ha utilizado para aquellas tareas y para aquellos momentos en los que abrir o utilizar un entorno de desarrollo completo no se ha considerado necesario, y era más simple editar el archivo sencillamente utilizando un editor de texto como Notepad++. A continuación se citan algunas características interesantes de Notepad++--:

- Identifica los lenguajes de programación más habituales y gracias a ello ofrece una presentación ordenada y clara del código.
- Permite abrir prácticamente todo: archivos con cualquier extensión. Si Notepad++ no lo abre es que el archivo está corrupto o no es editable.
- Permite trabajar con múltiples archivos abiertos en diferentes pestañas pero en una sola ventana.
- Reconoce las etiquetas y marca el principio, fin y elementos singulares de las mismas cuando se posiciona el cursor encima de ellas.



Código 3-1 Notepad++

3.3.6 SolidWorks

SolidWorks es un software CAD (diseño asistido por computadora) para modelado mecánico en 3D, desarrollado en la actualidad por SolidWorks Corp., una filial de Dassault Systèmes, S.A. (Suresnes, Francia), para el sistema operativo Microsoft Windows. Su primera versión fue lanzada al mercado en 1995 con el propósito de hacer la tecnología CAD más accesible.

El programa permite modelar piezas y conjuntos y extraer de ellos tanto planos técnicos como otro tipo de información necesaria para la producción. Es un programa que funciona con base en las nuevas técnicas de modelado con sistemas CAD. El proceso consiste en trasvasar la idea mental del diseñador al sistema CAD, "construyendo virtualmente" la pieza o conjunto. Posteriormente todas las extracciones (planos y ficheros de intercambio) se realizan de manera bastante automatizada.

Este software se ha usado para generar todos los planos necesarios a la hora de construir físicamente la maqueta del laboratorio remoto.

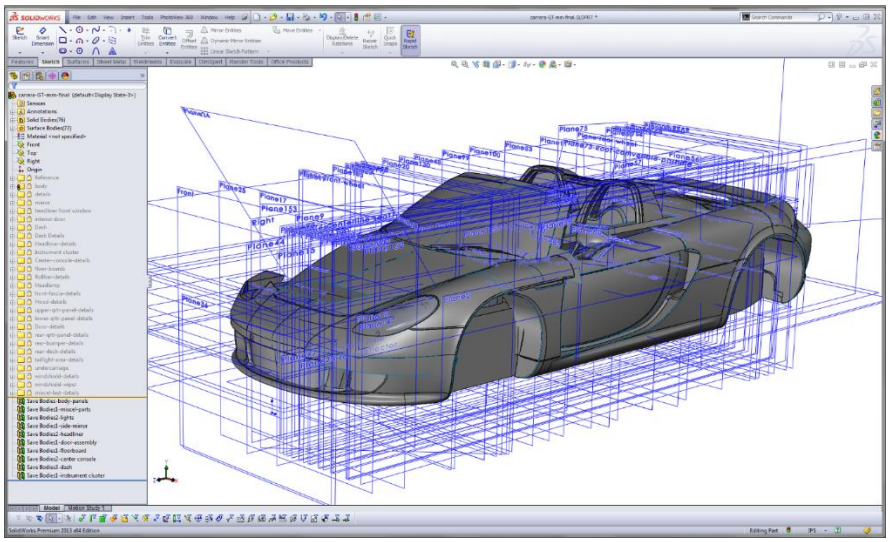


Figura 3-4 SolidWorks

3.3.7 Fritzing

Fritzing es un programa de automatización de diseño electrónico libre que busca ayudar a diseñadores para que puedan pasar de prototipos (usando, por ejemplo, placas de pruebas) a productos finales.

Fritzing fue creado bajo los principios de Processing y Arduino, y permite a los diseñadores, investigadores y aficionados documentar sus prototipos basados en Arduino y crear esquemas de circuitos impresos para su posterior fabricación. Además, cuenta con un sitio web complementario que ayuda a compartir y discutir bosquejos y experiencias y a reducir los costos de fabricación.

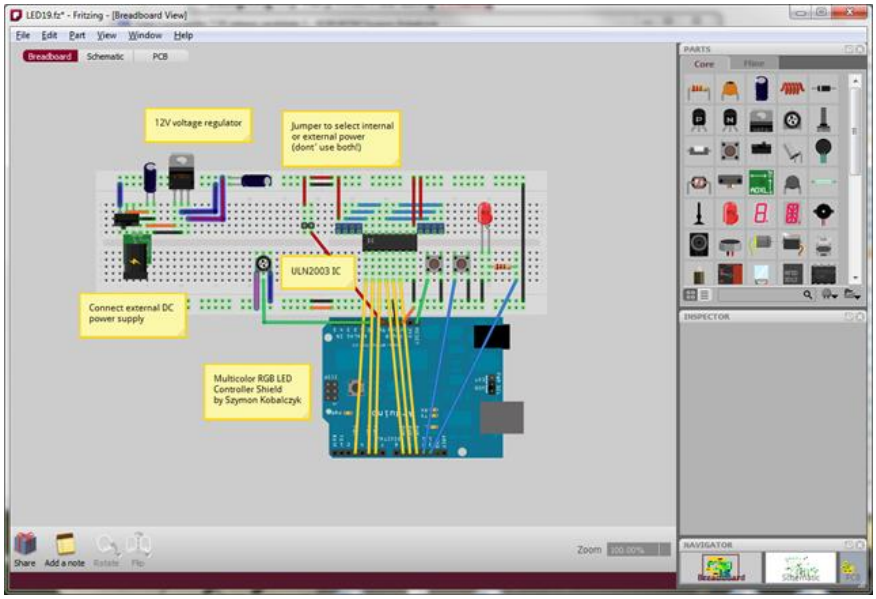


Figura 3-5 Fritzing

4 DESARROLLO DEL PROYECTO

4.1 Identificación de las partes del sistema

El proyecto consta de tres partes hardware controladas por las diferentes aplicaciones software:

- Hardware, en este apartado se incluyen:
    - La placa Arduino Mega, sensores y actuadores, como controladores principales del sistema.
    - Un PC (ordenador personal) en el papel del servidor, encargado de la toma y almacenamiento de datos de la placa Arduino en una base de datos y modificación de su forma de trabajo si el cliente lo pide.
    - Un PC (ordenador personal) en el papel de cliente, encargado de ejecutar el módulo de aprendizaje SCORM a través de la identifiación en el LMS. Una vez dentro del módulo SCORM se ejecutarán tanto el laboratorio virtual como el laboratorio remoto.
- Software, para el funcionamiento de cada una de las piezas hardware:
    - Un sketch para la configuración del funcionamiento de la placa Arduino Mega, así como para la configuración de sus pines y el puerto serial de comunicaciones USB.
    - Dos aplicaciones Java, realizando las labores de cliente y servidor, para controlar de forma directa (servidor) y de forma remota (cliente) el correcto funcionamiento del invernadero.
    - Con EJS (Easy Java Simulations), un cliente a través de un entorno gráfico de programación orientado al desarrollo de una simulación científica y técnica. Con EJS se realizará tanto la interfáz gráfica del applet del laboratorio virtual (simulación) como el applet para el laboratorio remoto.
    - Programación módulo de aprendizaje SCORM

4.2 Invernadero a escala

Se ha construido un invernadero a escala donde se alojarán todos los actuadores y sensores. El invernadero está compuesto por un depósito de agua externo para su uso en el riego y por la estructura propia que alberga la planta que se va a estudiar.

Dentro de este invernadero se ejecutarán diferentes acciones para ver su comportamiento en el tiempo con el fín de estudiar el crecimiento y la evolución de la planta que contenga en su interior.

El invernadero se adquirió ya fabricado en un establecimiento especializado. Se ha diseñado de tal manera que todos los sensores, actuadores y el controlador estén pegados a él para que así se pueda desplazar sin tener que desconectar nada.

Figura 4-1 Imagen renderizada del invernadero

4.3 Microcontrolador Arduino

4.3.1 Arduino Mega 2560

Arduino es una marca de microcontroladores mundialmente conocida por los amantes de la electrónica, la programación y la robótica. Es un proyecto Open Source que pone a disposición de sus usuarios una amplia gama de dispositivos basados en el microcontrolador AtMega. Es posible comprar una placa Arduino armada o bien conseguir las piezas para uno mismo desarrollar sus propios dispositivos.

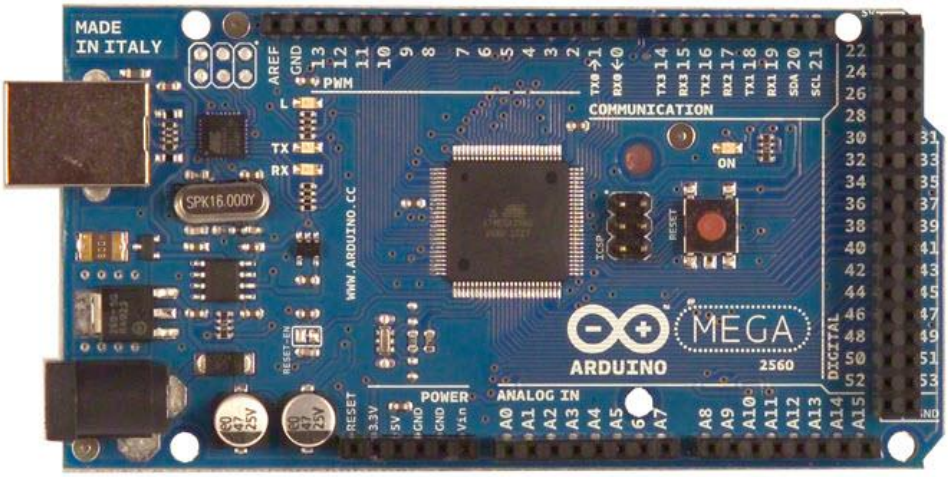


Figura 4-2 Placa Arduino Mega 2560

El Arduino Mega es probablemente el microcontrolador más potente de la familia Arduino. Posee 54 pines digitales que funcionan como entrada/salida, 16 entradas análogicas, un cristal oscilador de 16 MHz, una conexión USB, un botón de reset y una entrada para la alimentación de la placa.

La comunicación entre la computadora y Arduino se produce a través del puerto serie, sin embargo posee un convertidor USB-Serie, por lo que sólo se necesita conectar el dispositivo al PC utilizando un cable USB tipo A-B.

Figura 4-3 USB tipo A-B

Como se indica en la siguiente figura, la placa Arduino Uno está formada por:

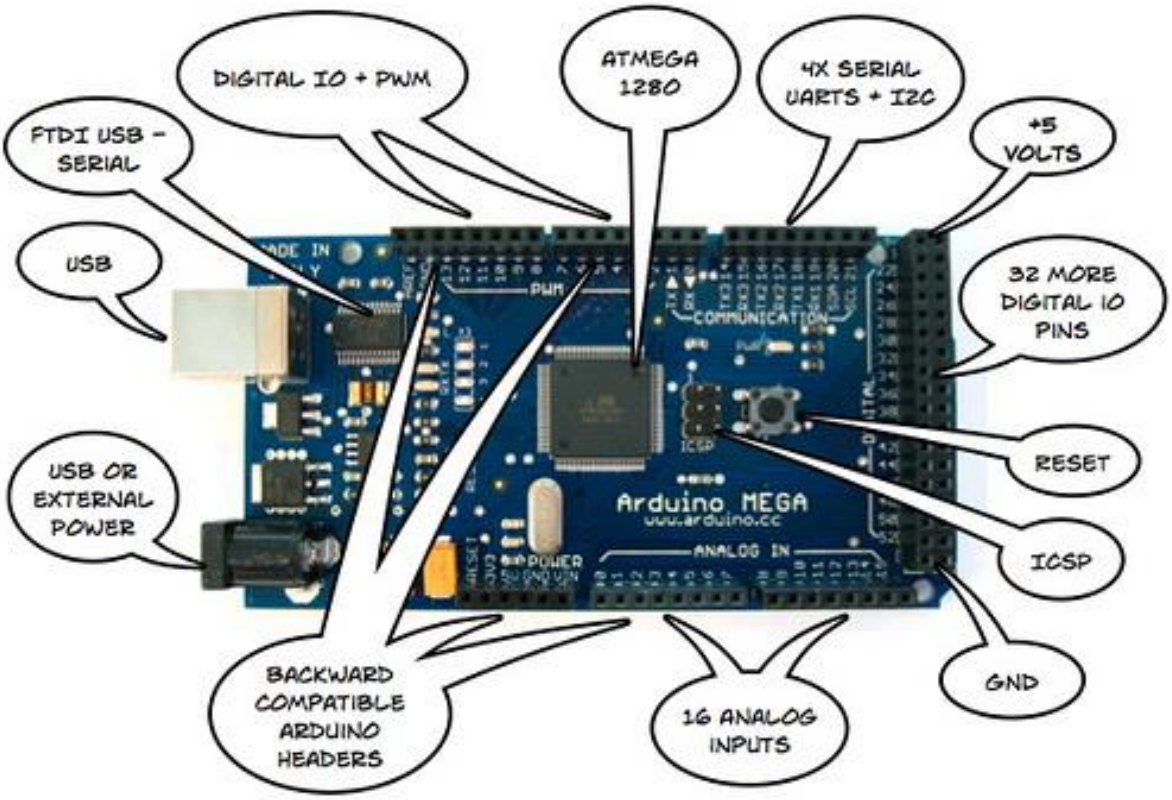


Figura 4-4 Principales componentes de Arduino

- Microcontrolador: ATmega2560.
- Voltaje Operativo: 5V.
- Voltaje de Entrada recomendado: 7-12V.
- Voltaje de Entrada (límites): 6-20V.
- Pines digitales de Entrada/Salida: 54 (de los cuales 15 proveen salida PWM).
- Pines análogos de entrada: 16.
- Corriente DC por cada Pin Entrada/Salida: 40 mA.
- Corriente DC entregada en el Pin 3.3V: 50 mA.
- Memoria Flash: 256 KB (8KB usados por el bootloader).
- SRAM: 8KB.

- EEPROM: 4KB.
- Clock Speed: 16 MHz.
- Botón de reset.
- Un LED que indica si la placa recibe alimentación o no.
- Dos LED's que indican si se está enviando o recibiendo datos por el puerto serie.
- Arduino Mega puede ser alimentado mediante el puerto USB o con una fuente externa. La alimentación es seleccionada de manera automática.
- Cuando se trabaja con una fuente externa se debe utilizar un convertidor AC/DC y regular dicho voltaje en el rango operativo de la placa. De igual manera se puede alimentar el micro mediante el uso de baterías. Preferiblemente el voltaje debe estar en el rango de los 7V hasta los 12V.
- Arduino Mega posee algunos pines para la alimentación del circuito aparte del adaptador para la alimentación:
    - VIN: A través de este pin es posible proporcionar alimentación a la placa.
    - 5V: Se puede obtener un voltaje de 5V y una corriente de 40mA desde este pin.
    - 3.3V: Se puede obtener un voltaje de 3.3V y una corriente de 50mA desde este pin.
    - GND: El ground (0V) de la placa.

A continuación se muestra un diagrama de pines para el controlador que se va a usar: Arduino Mega 2560.

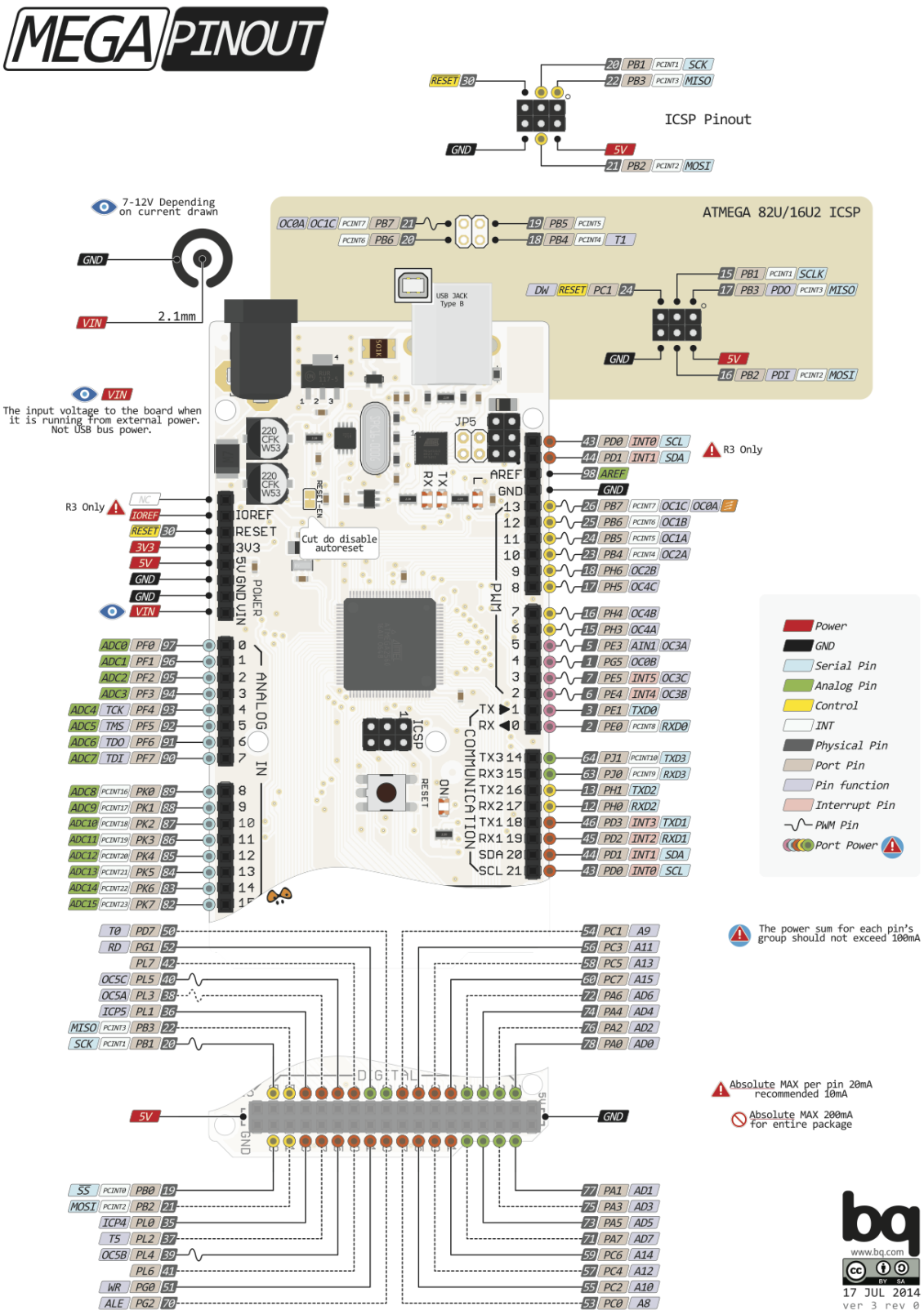


Figura 4-5 Diagrama de pines controlador Arduino Mega

4.4 Sensores y actuadores.

4.4.1 Sensor de Humedad y Temperatura Ambiente (DHT11).

El DHT11 es un sensor básico de humedad y temperatura de costo reducido. Usa un sensor de capacidad para medir la humedad y un termistor para medir la temperatura del aire que lo rodea. Está diseñado para medir temperaturas entre 0 y 50°C con una precisión de ±1°C y para medir humedad entre 20% y 80% con una precisión de 5% con periodos de muestreo de 1 segundo. El formato de presentación es una pequeña caja de plástico de 15.5mm x 12mm x 5.5mm con una cara en la cual tiene una rejilla que le permite obtener las lecturas del aire que lo rodea. Si se requiere mayor precisión se puede trabajar con su hermano, el sensor DHT22. El sensor tiene cuatro pines de los cuales solo se se van a usar el pin 1,2 y 4.

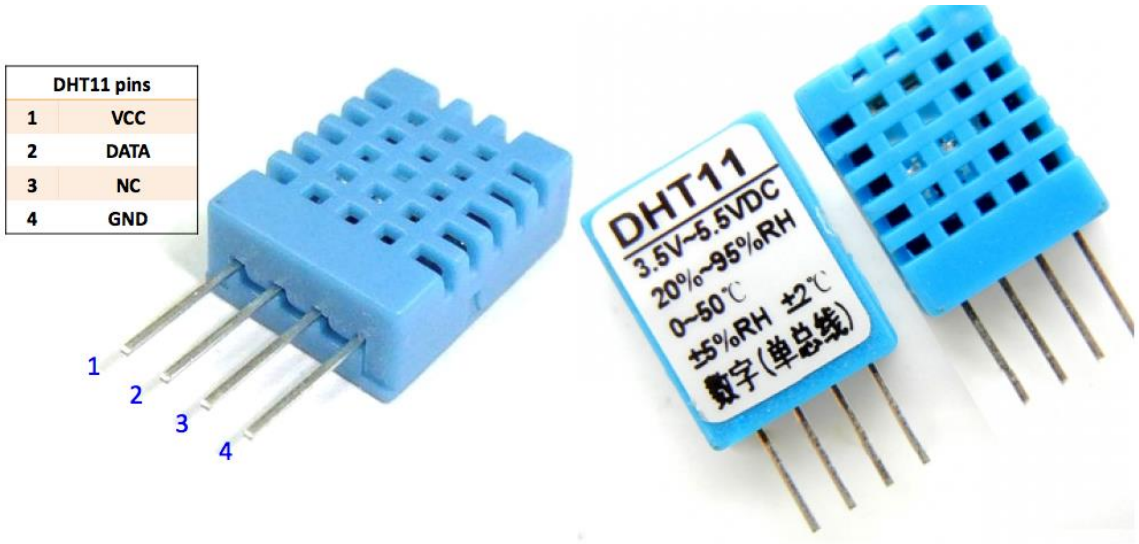


Figura 4-6 Sensor DHT11

A la hora de comprar el sensor existen por norma general tres variantes:

- El sensor suelto, con un encapsulado azul y cuatro pines disponibles para conectar.
- El sensor con una placa soldada, con tres pines disponibles para conectar, y una resistencia pull-up (normalmente de 4,7-10 kΩ).
- El mismo formato que el anterior, pero con un condensador de filtrado (normalmente de 100 nF).

Figura 4-7 Sensor DHT11 encapsulado

Las conexiones que se realizarán para obtener datos del sensor de humedad DHT11 se indican en la siguiente figura

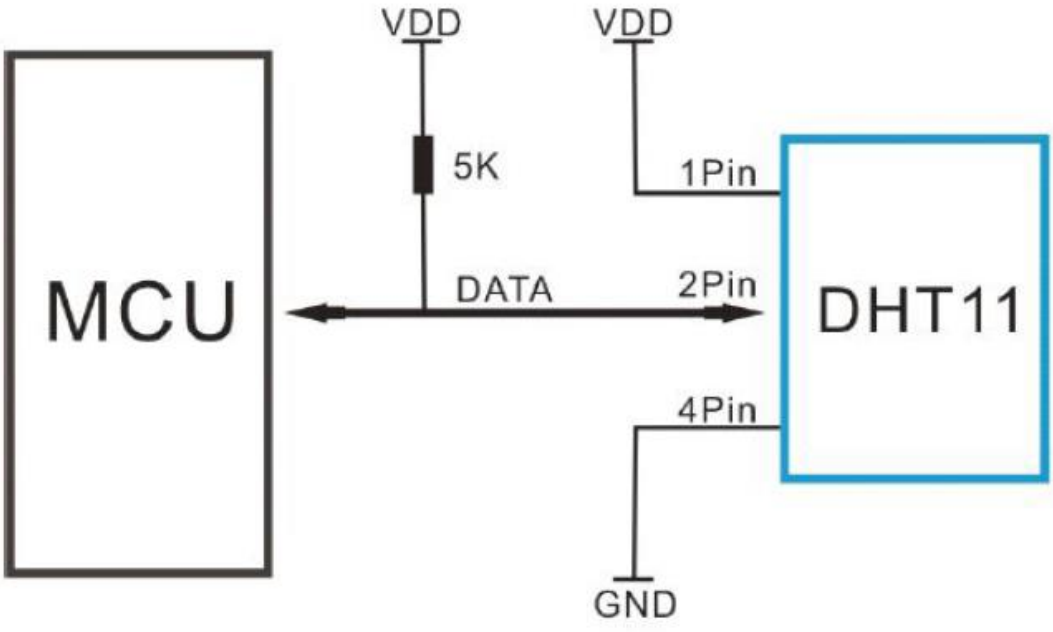


Figura 4-8 Diagrama de conexiones sensor DHT11

A continuación se muestra el datasheet del sensor.

Tabla 4-1 Especificaciones sensor DHT11

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

Para leer y transformar los datos que entrega el sensor es necesario utilizar dos librerías, ambas incluidas en el anexo. Para instalar las librerías en el ambiente de desarrollo Arduino (IDE) se debe bajar el archivo comprimido zip. Se descargará un archivo al PC que debe ser descomprimido, dentro del archivo zip se encuentra una carpeta, se debe copiar esta carpeta al directorio donde estén las librerías del ambiente de desarrollo Arduino (IDE) que en este ejemplo dónde se está utilizando Windows están en "*C:\Program Files (x86)\Arduino\libraries*", el contenido de la carpeta se copia dentro de "*C:\Program Files (x86)\Arduino\libraries\DHT*" y posteriormente se reinicia el IDE. Una vez montado el circuito y con la librería cargada es hora de comenzar a trabajar con el sketch para Arduino (véase Código 4-1):

```
#include "DHT.h"            // Librería para Sensores DHT
#define DHTPIN 2            // Pin del Arduino input
#define DHTTYPE DHT11       // DHT 11

DHT dht(DHTPIN, DHTTYPE);  // Inicializa el sensor
void setup()
{
  Serial.begin(9600);
  dht.begin();
}

void loop()
{
  delay(2000); // Espera dos segundos para realizar la primera medición.

  float h = dht.readHumidity();     // Obtiene la Humedad
  float t = dht.readTemperature();  // Obtiene la Temperatura en Celsius

  Serial.print("Humedad: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperatura: ");
  Serial.print(t);
  Serial.println(" °C ");
}
```

Código 4-1 Sensor DHT11

En el monitor serial se podrá ver como se ejecuta el script y los valores que obtiene al medir la temperatura y humedad del aire que lo rodea. Prueben a respirar cerca del sensor.

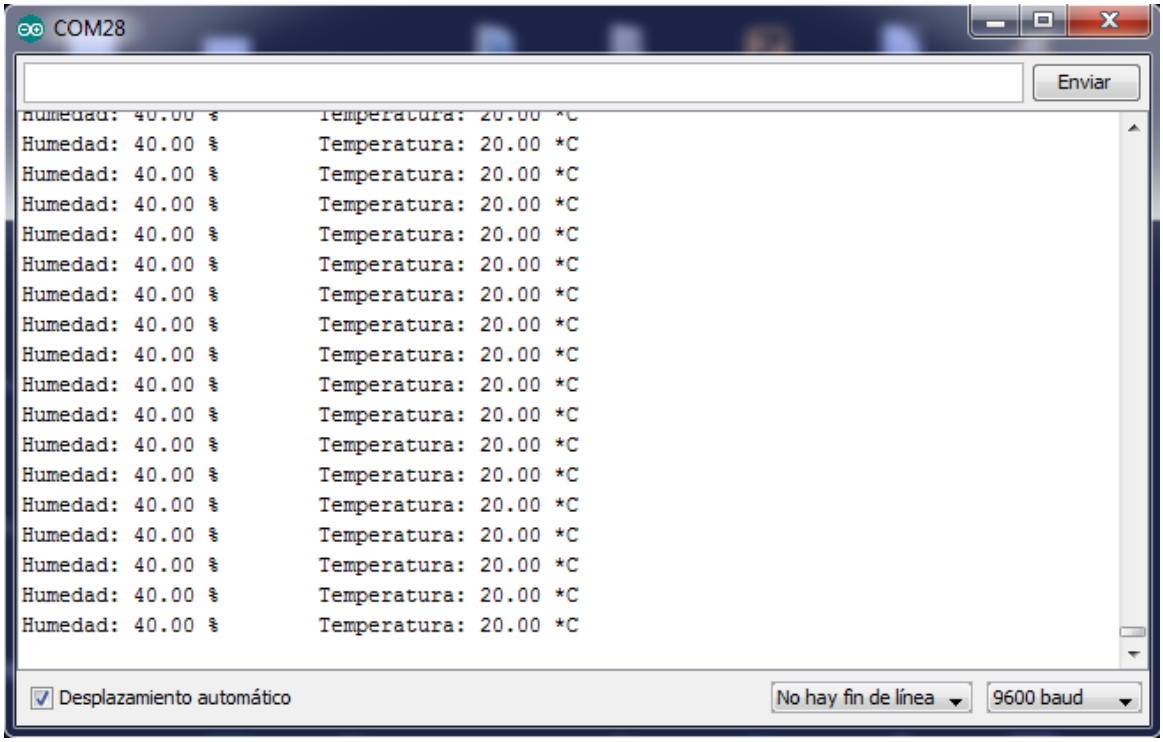


Figura 4-9 Monitor Serial

4.4.2 Sensor de Humedad del Suelo (HL-69).

El módulo HL-69 un sensor de humedad de suelo que utiliza la conductividad entre dos terminales para determinarla.

Antes se ha hablado de los sensores de humedad relativa, como el DHT11 y el DHT22. Ambos devuelven la humedad en el ambiente pero no son capaces de medir la humedad en el suelo. Para este propósito se usa el Módulo HL-69.

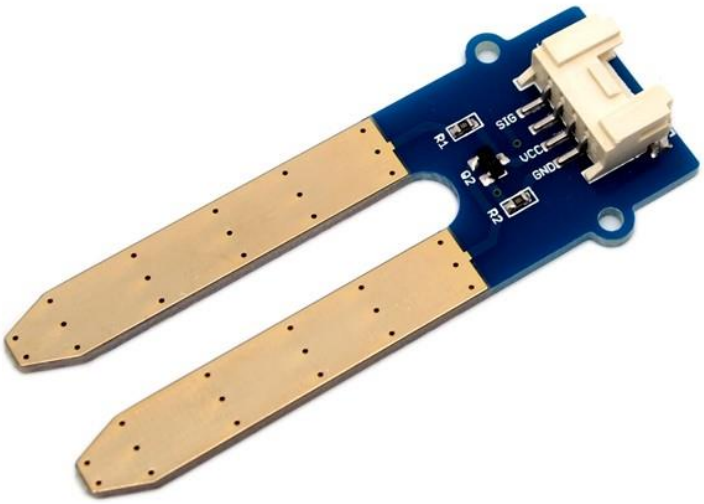


Figura 4-10 Sensor HL-69

El principio de funcionamiento de este dispositivo consiste en dos placas separadas entre sí por una distancia determinada. Ambas placas están recubiertas de una capa de material conductor. Si existe humedad en el suelo se creará un puente entre una punta y otra, lo que será detectado por un circuito de control con un amplificador operacional que será el encargado de transformar la conductividad registrada a un valor analógico que podrá ser leído por Arduino.

La programación de dicho sensor consiste simplemente en declarar el PIN al que está conectado como pin de entrada y posteriormente realizar la lectura analógica. El código para utilizar el sensor quedaría de la siguiente forma:

```
void setup()
{
        Serial.begin(9600);
        pinMode(10, INPUT);
}

void loop()
{
        Serial.println(analogRead(A0)); //lectura analógica
        delay(100);
}
```

Código 4-2 Sensor HL-69

En la salida analógica el nivel de voltaje dependerá directamente de cuanta humedad haya en el suelo. Es decir, dependiendo de cuanta conductividad (producto del agua en el suelo) haya entre las puntas del módulo, así variará el valor entregado por Arduino (entre 0 y 1023).

Al final, este módulo es muy útil en proyectos de control donde se requiera monitorizar las condiciones del suelo, especialmente si se está trabajando con plantas que necesitan cuidados especiales.

Las especificaciones de este sensor son las siguientes:

Tabla 4-2 Especificaciones sensor HL-69

	Condición	Min	Max
Voltaje		3.3V	5V
Corriente		0	35mA
Valor de salida	Sensor en suelo seco	0	300
	Sensor en suelo húmedo	300	700
	Sensor en agua	700	1000

4.4.3 Sensor LDR.

Una fotorresistencia LDR, aumenta su valor en función de la luminosidad ambiente. Se puede utilizar el módulo sensor de luz para detectar el nivel de luz ambiente (salida analógica), o para activar el disparo de un relé (salida digital), en infinidad de proyectos Arduino o PIC, así como proyectos electrónicos de iluminación.

Su funcionamiento se basa en el efecto fotoeléctrico. Un fotorresistor está hecho de un semiconductor de alta resistencia como el sulfuro de cadmio, CdS. Si la luz que incide en el dispositivo es de alta frecuencia, los fotones son absorbidos por las elasticidades del semiconductor dando a los electrones la suficiente energía para saltar la banda de conducción. El electrón libre que resulta, y su hueco asociado, conducen la electricidad, de tal modo que disminuye la resistencia. Los valores típicos varían entre 1 MΩ, o más, en la oscuridad y 100 Ω con luz brillante.

Las células de sulfuro del cadmio se basan en la capacidad del cadmio de variar su resistencia según la cantidad de luz que incide en la célula. Cuanta más luz incide, más baja es la resistencia. Las células son también capaces de reaccionar a una amplia gama de frecuencias, incluyendo infrarrojo (IR), luz visible, y ultravioleta (UV).

La fotorresistencia cambia su valor resistivo conforme a la intensidad de luz. Mayor luz, menor resistencia y viceversa como muestra la siguiente gráfica.

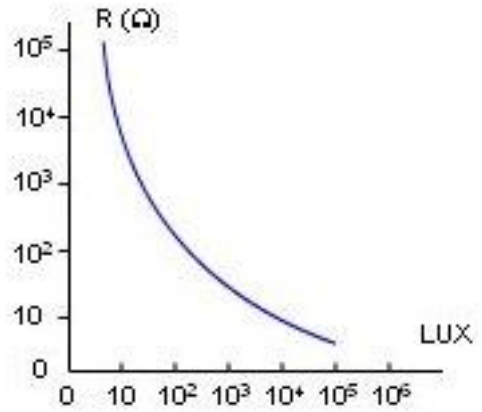


Figura 4-11 Gráfico comportamiento LDR

Características:

- Sensor de luz tipo fotorresistencia (LDR).
- Salida limpia de ruido a la salida del comparador.
- Potenciómetro de ajuste para regular el nivel de luz límite de detección.
- Tensión de trabajo de 3.3V a 5V.
- Dispone de salida tanto digital (0 y 1) y una patilla con salida analógica para poder medir los niveles de iluminación.
- Taladros para fácil instalación.
- Dimensiones de la PCB: 3.2 cm x 1.4 cm.
- Utiliza el comparador de tensión LM393.
- Indicación por led del estado de la salida y de alimentación.

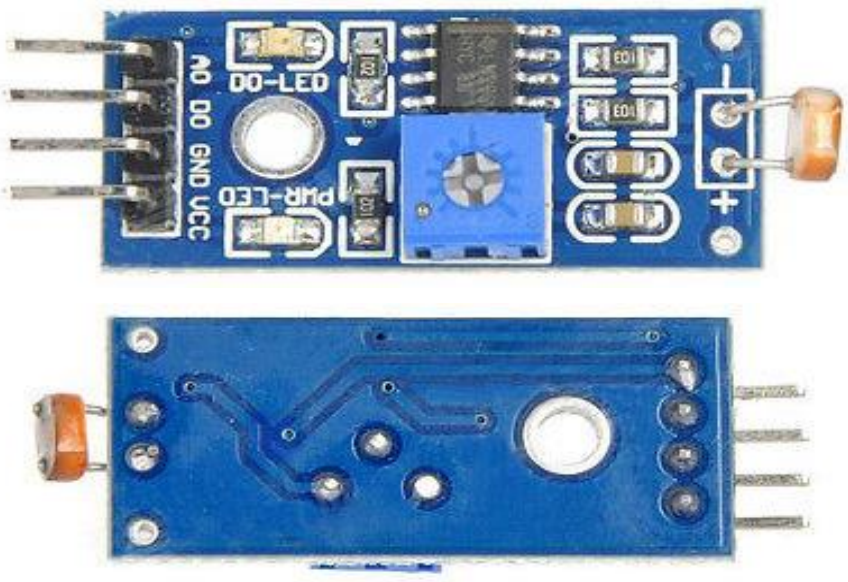


Figura 4-12 Sensor LDR

A continuación se muestra un ejemplo para controlar este sensor en Arduino.

```
const int analogInPin = A0; // Pin al que está conectado la LDR
int sensorValue = 0; // declaración en inicialización

void setup()
{
Serial.begin(9600);  // Se inicia la comunicación serial a 9600 bps
}
void loop()
{
sensorValue = analogRead(analogInPin);

// Se imprime el resultado
Serial.print("sensor = " );
Serial.print(sensorValue);
Serial.print("\t output = ");
Serial.println(outputValue);

delay(100);

}
```

Código 4-3 Sensor LDR

4.4.4 Módulo de 4 relés optoacoplados

El motivo de usar este módulo es poder trabajar con tensiones y potencias mucho mayores que las proporcionadas por el controlador Arduino Mega. Debido a que estos pequeños microcontroladores se mueven con corrientes y tensiones muy bajas son incapaces de controlar aparatos electrónicos que funcionan con corrientes y tensiones altas como por ejemplo una bombilla que utiliza 220 voltios. El funcionamiento es el mostrado en la siguiente figura: aplicando una tensión positiva de 5V desde Arduino se activa el contacto que permitirá la activación del dispositivo de mayor potencia.

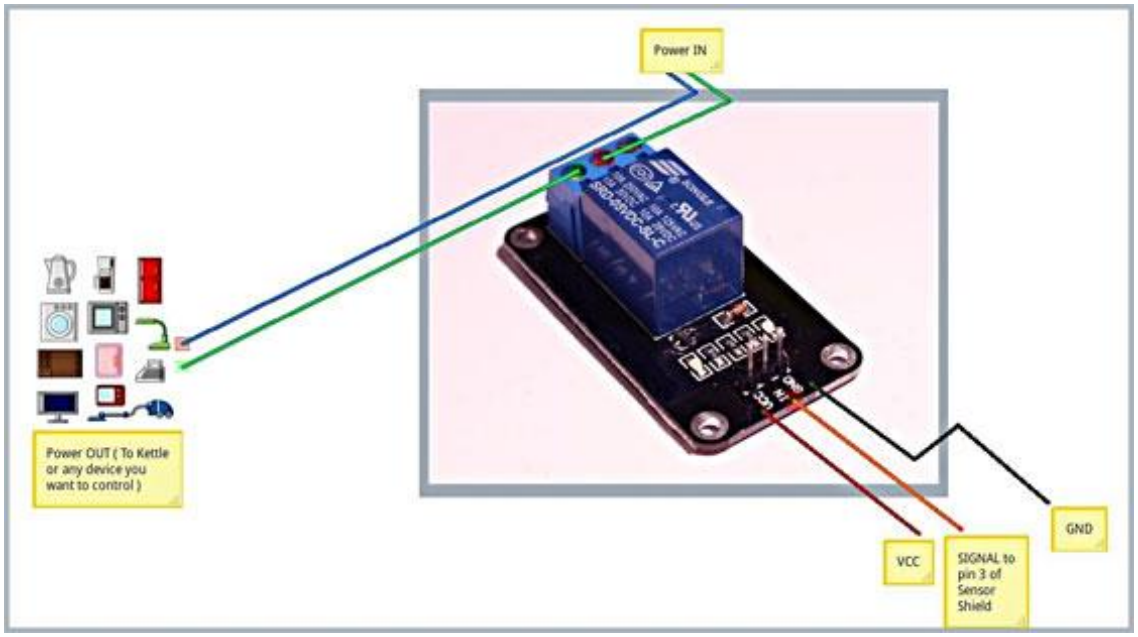


Figura 4-13 Diagrama conexión relé

El relé es un dispositivo electromecánico. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.

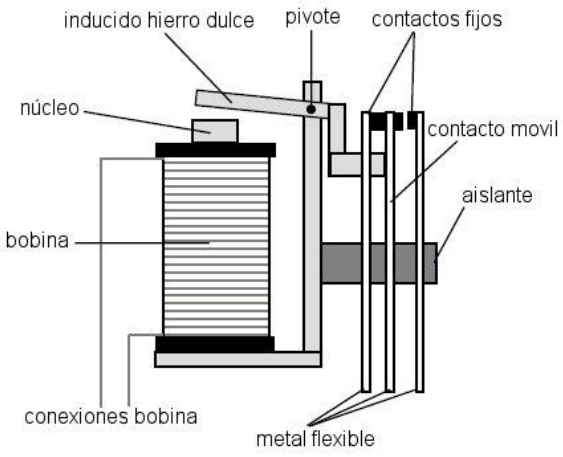


Figura 4-14 Partes de un relé

Hay que tener en cuenta que debido a la configuración de los contactos, los relés pueden ser NO (normalmente abiertos) y NC (normalmente cerrados). Los NO en ausencia de tensión en la bobina del relé estarán abiertos, es decir que no dejarán pasar intensidad. Por el contrario los NC se comportan de manera inversa. Esto es importante porque dependiendo de esta característica habrá que cambiar la forma de actuar con el microcontrolador Arduino. En este caso se usará la configuración NO.

Además del relé, se necesitan elementos adicionales para hacer el circuito posible, en este caso, los componentes que se van a utilizar son los siguientes:

- Relé sellado SPST-NO de 20 Amperios.
- Resistencia 1k Ohm.
- Resistencia 10k Ohm.
- Transistor BJT.
- Diodo rectificador – 1N4148.
- LED .

Para que el relé funcione correctamente hay que utilizar algunos componentes extras en el circuito. Es necesario usar un Transistor BJT debido a que el relé funciona con una

corriente de 80mA que es bastante más de lo que el pin GPIO del Arduino puede manejar, que son unos 20mA, entonces el transistor cuando funcione en zona activa permitirá elevar la corriente hasta unos 200mA, más que suficiente para controlar el relé y el Led indicador (este led se encenderá cuando el relé este activado).

También habrá que usar un diodo rectificador para que las corrientes no puedan moverse en sentido inverso y dañen el microcontrolador Arduino.

Conectado al pin de Arduino con el que se activará el relé se tendrá una resistencia de 1kΩ (R2), que a su vez se conectará a la base del transistor. El emisor del transistor va a masa (GND) y al colector va conectado un diodo rectificador (que como ya se ha mencionado se ocupa de que las corrientes solo se muevan en un sentido y no puedan dañar el microcontrolador) y en paralelo el relé, es decir, las patillas de control de la bobina del relé que serán las que activen o desactiven las salidas de éste (la bombilla). Además al colector se conectará una resistencia de 1kΩ (R4) seguido de un diodo led que indicará cuando el relé está funcionando.

El esquema del circuito sería el siguiente:

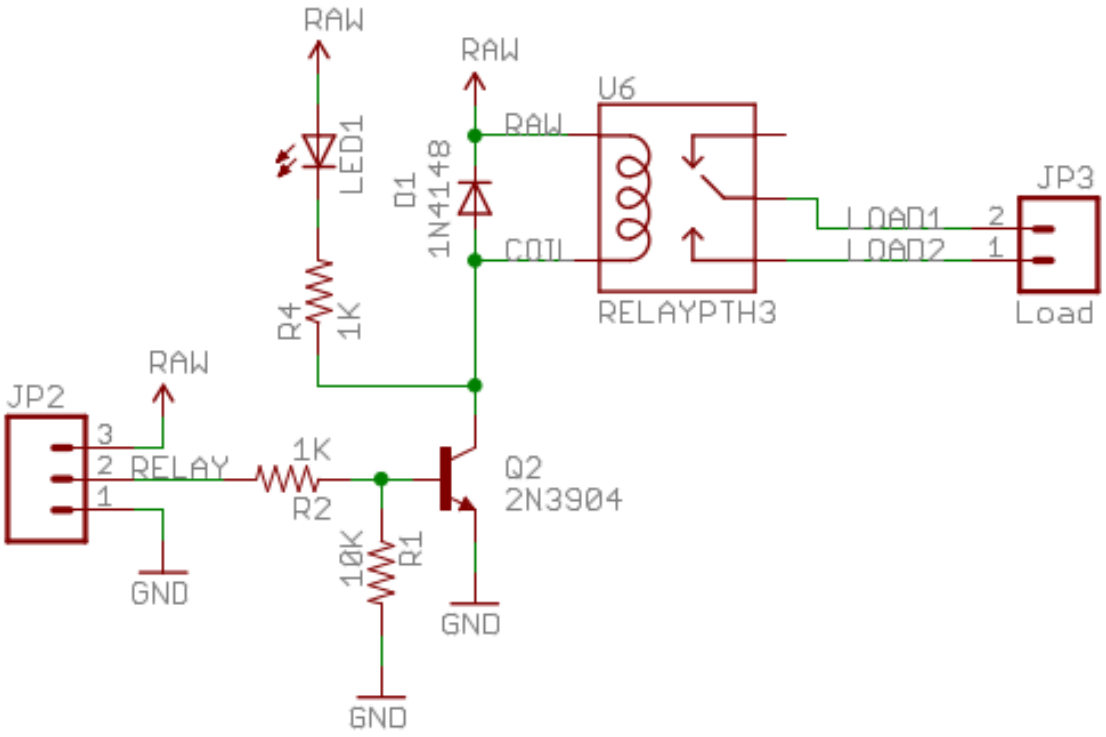


Figura 4-15 Esquema del circuito integrado en el relé

En este caso se ha utilizado una placa con las conexiones entre los elementos hechas, el motivo es que se facilita mucho las conexiones y todo queda mucho más ordenado, no habrá que utilizar ninguna protoboard y ocupa mucho menos espacio, su precio es muy bajo por lo que merece la pena. A pesar de todo esto la placa es prescindible y no hay por qué usarla, siempre se podrá realizar el circuito anteriormente indicado.

Figura 4-16 Relé individual

En este caso se necesitan 3 relés: ventilación, bombeo e iluminación. Se ha optado por comprar un módulo que incluye 4 relés para poder tenerlos todos agrupados en una única shield.

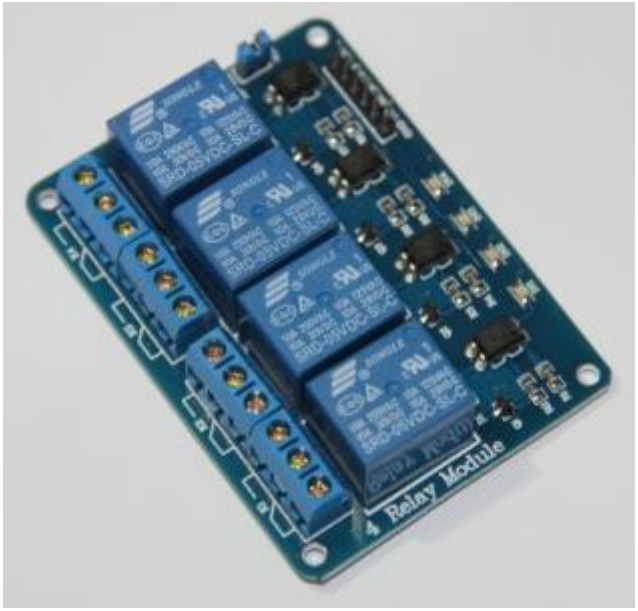


Figura 4-17 Módulo de 4 relés

A continuación se muestra un sencillo ejemplo de cómo programar el relé en Arduino, en este caso se enciende y apaga una bombilla con un intervalo de 1 segundo. El programa es muy sencillo:

```
int rele = 8;

void setup()

{
  pinMode(rele,OUTPUT);
}

void loop()

{
  digitalWrite(rele,HIGH);   // Se activa el relé y la bombilla
  delay(1000);               // 1 segundo
  digitalWrite(rele,LOW);    // Se desactiva el relé y la bombilla
  delay(1000);               // 1 segundo
}
```

Código 4-4 Actuador Relé

4.4.5 Sensor nivel de agua.

Por seguridad y para no quemar la bomba de agua es necesario detectar cuando no queda agua en el depósito. Para ello se usará el siguiente sensor:

-

Figura 4-18 Sensor nivel de agua

Es el denominado “tipo boya” debido a que funciona gracias a la diferencia de densidades entre el agua y el PVC. Hay que remarcar que este sensor solo proporcionará un dato de tipo booleano es decir, hay agua o no hay agua. Existen otro tipo de sensores que pueden llegar a decir la altura de agua en el depósito pero para este caso no es de interés.

Este sensor está diseñado para su montaje en el interior de depósitos. Están fabricados con polipropileno siendo aptos para agua y líquidos similares. Cuando el flotador magnético pivota al nivel adecuado, el sensor abrirá o cerrará sus contactos (según la posición de montaje). La sujeción se realiza mediante el pegado a una pared del depósito con pegamento termofusible. A continuación se muestran los datos técnicos del sensor

- Tipo contacto Reed.
- Características contactos 200 Vdc, 0,5ª (máx.10W).
- Resistencia contacto 15 Ω máx.
- Actuación 0,5ms.
- Liberación 0,1 ms.

- Temperatura de trabajo -20ºC a 90ºC.

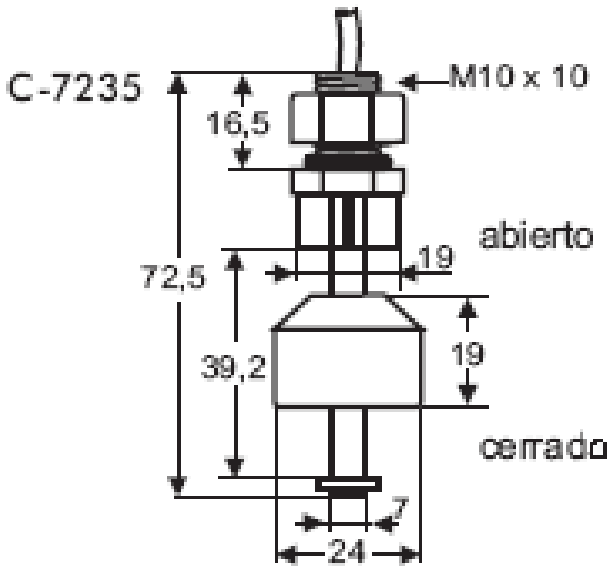


Figura 4-19 Plano sensor nivel de agua

Un ejemplo del código usado para leer este tipo de sensor digital (1/0) es el siguiente:

```
//Se conecta al pin 2 el sensor de agua
int pin = 2;


void setup() {
  //Se inicializa la comunicación serial
  Serial.begin(9600);
  pinMode(pin, INPUT);
}


void loop()
{

  int h2osensor = digitalRead(pin);
  Serial.println(h2osensor);
  delay(1000);
}
```

Código 4-5 Sensor nivel de agua

4.4.6 Ventilador.

Es el encargado de mantener la humedad en el interior del invernadero entre unos márgenes registrados para el cultivo. Se establece una humedad del invernadero máxima, cuando el sensor la detecta se pone en marcha el ventilador hasta alcanzar la humedad mínima establecida. La apariencia del dispositivo es la siguiente:

Figura 4-20 Ventilador

Debido a que requiere potencias elevadas que no son capaces de ser proporcionados por el propio Arduino, será necesario conectarlo a una fuente de alimentación externa y para ello se hará uso de un relé previamente descrito.

La programación en Arduino para su uso se reduce a la activación del relé.

Características técnicas:

- Dimensiones 45x45x10 mm.
- Tensión nominal 12V.
- Tensión nominal de funcionamiento 4.5~13.8 V.
- Velocidad 6000 rpm ±20%.
- Corriente 114 mA.

- Potencia 1.4 W.
- Nivel de ruido 33 dB(A).
- Temperatura de funcionamiento -10ºC ~70ºC.
- Peso 18.5 g.

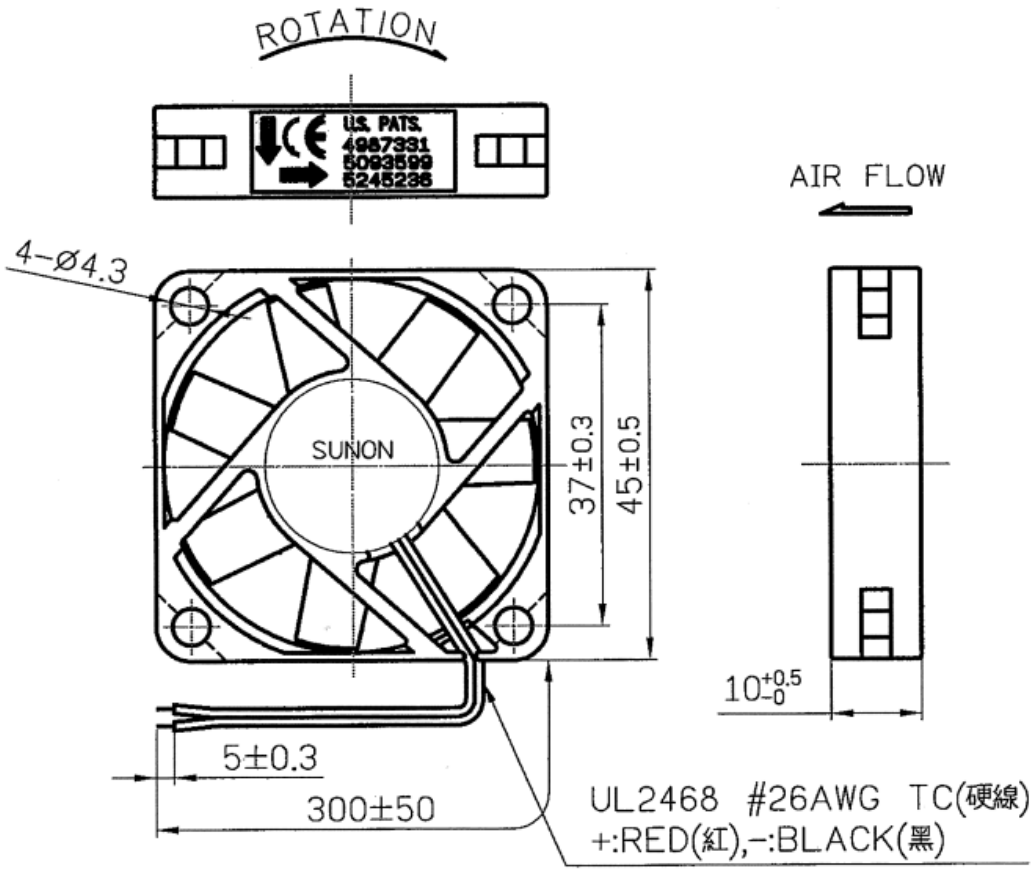


Figura 4-21 Plano del ventilador

4.4.7 Bomba de agua

Su función es la de regar el terreno, para ello, se configura la placa Arduino con unos márgenes entre los que se tiene que encontrar la humedad del terreno, cuando la humedad del terreno es menor que la mínima establecida se activa esta bomba de agua y riega hasta llegar al margen superior de humedad del terreno.

Debido a que requiere potencias elevadas que no son capaces de ser proporcionados por el propio Arduino, será necesario conectarlo a una fuente de alimentación externa y para ello se hará uso de un relé previamente descrito.

La programación en Arduino para su uso se reduce a la activación del relé.

Especificaciones:

- Caudal 8 l / min.
- Altura máxima de bombeo 5 m (0,5 bar).
- Tensión 12 V.
- Consumo 10 - 18 W.
- Dimensiones Ø38 x 104 mm.
- Cable 1 m.
- Módulo apropiado C-0166.

Figura 4-22 Bomba de agua

4.4.8 Sensor de O_2

Grove-Gas Sensor (O_2) es un tipo de sensor para medir la concentración de oxígeno en el aire, que se basa en el principio de la célula electroquímica. Se puede saber con claridad la actual concentración de oxígeno cuando la tensión de salida proporcional a los valores de la concentración de oxígeno y se refieren a la concentración de oxígeno gráfica característica lineal (ver tabla). Es muy adecuado para la detección de la concentración de oxígeno en la protección del medio ambiente.

Figura 4-23 Sensor de O_2

Entre las características del sensor se encuentran:

- Alta precisión.
- Alta sensibilidad.
- Amplia gama de linealidad.
- Capacidad anti-interferencia fuerte.
- Fiabilidad extraordinaria.

En el siguiente gráfico se puede comprobar perfectamente la linealidad de dicho sistema. Además el valor medido y el valor ideal prácticamente coinciden.

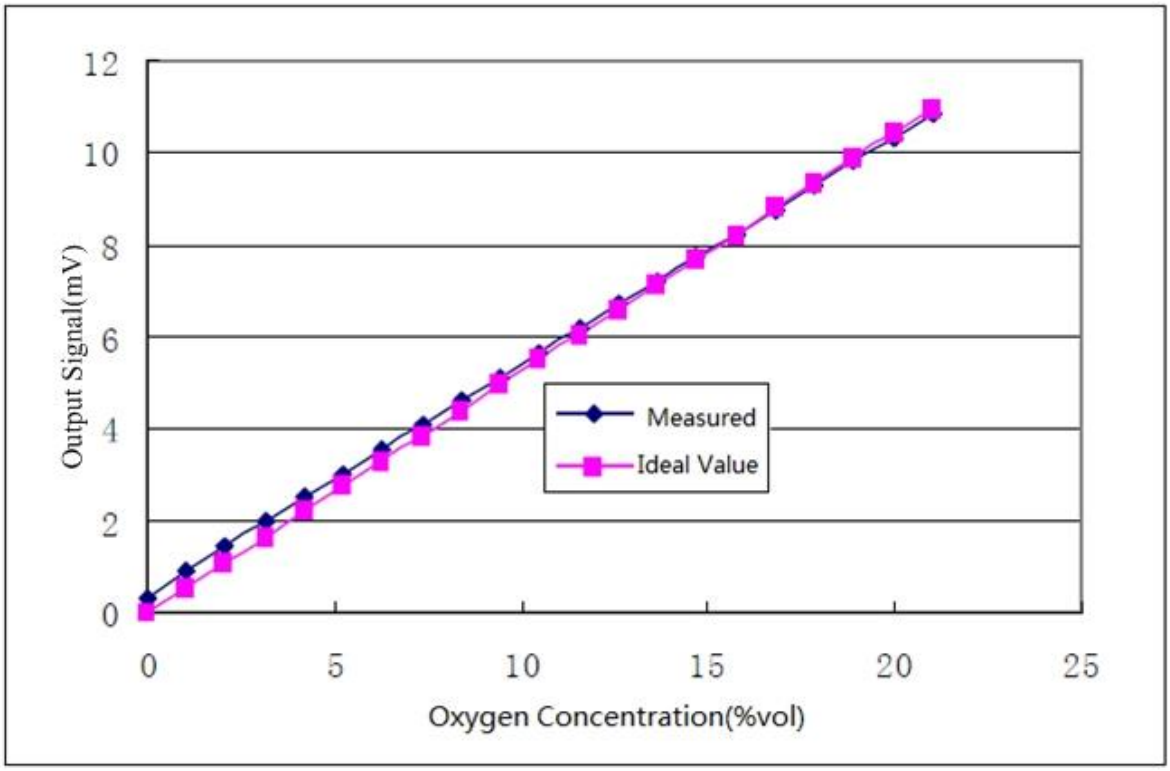


Figura 4-24 Comportamiento del sensor de 0_2

Especificaciones:

Tabla 4-3 Especificaciones sensor O_2

Items	Parameter
Measurement Range	0-30%vol
Detect Life	two years
Sensitivity	0.10～0.25 mA(in air)
Temperature Range	-20℃～50℃
Pressure range	QNE±10%
Response Time	≤15S
Humidity range	0～99%RH Non-condensing
Stability	<2%

El código para Arduino es el siguiente:

```
#include <math.h>

void setup()
{
  Serial.begin(9600);

}
void loop()
{

float sensorValue;
float sensorVoltage;
sensorValue = analogRead(A0);
sensorVoltage =(sensorValue/1024)*5.0;
sensorVoltage = sensorVoltage/201*1000;
Serial.println("the output voltage is:");

Serial.print(sensorVoltage);
Serial.println("mV");
delay(1000);
}
```

Código 4-6 Sensor O_2

Tras la carga del código se podrá ver en el monitor serial del IDE la salida de los datos buscados:

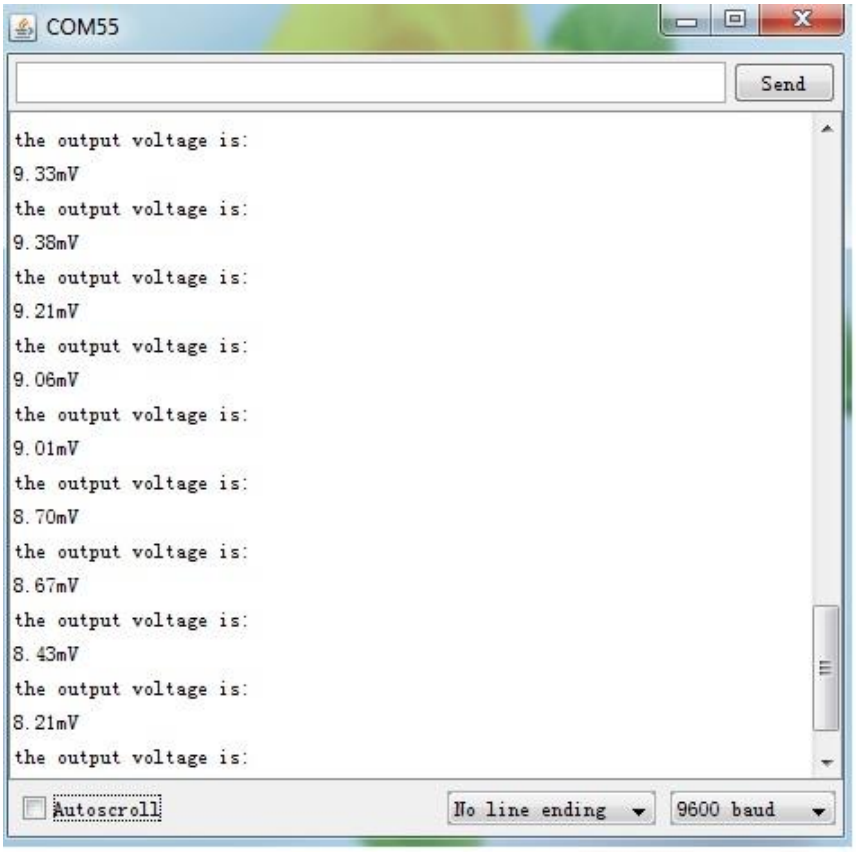


Figura 4-25 Ejecución del programa para el sensor de O_2

Esquemático del dispositivo

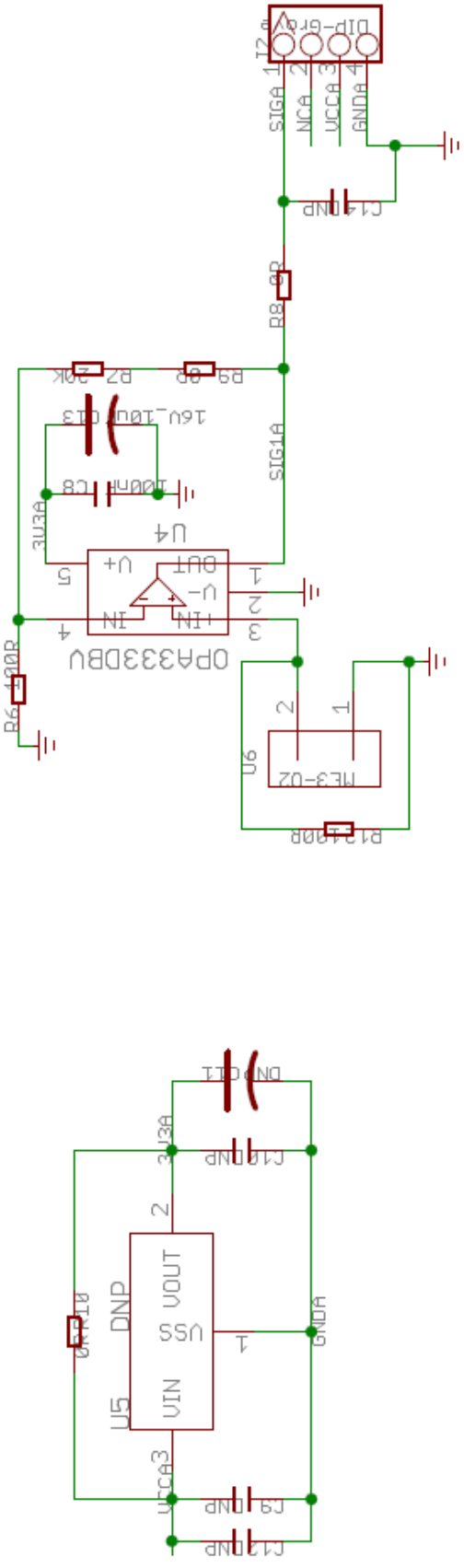


Figura 4-26 Esquemático del sensor de O_2

4.4.9 Sensor CO_2

El sensor de CO_2 ha sido diseñado por DFRobot. La tensión de salida del módulo cae con el aumento de concentracción de CO_2. El potenciómetro incluido está diseñado para establecer el umbral de voltaje. Mientras la concentración de CO_2 sea lo suficientemente alta (tensión es menor que el umbral), una señal digital (ON / OFF) será la salida.

Cuenta con un módulo sensor MG-811, que es muy sensible a cambios en la concentración de CO_2. Todos los componentes tienen calidad industrial que significa estabilidad y reproducibilidad. Además este sensor tiene un circuito accondicionador incluido para amplificar la señal de salida. El voltaje de operación de este sensor es de 5V. Una de las mayores ventajas de este dispositivo es que se proporciona tanto una salida analógica como digital. En este caso solamente interesa la salida analógica.

Figura 4-27 Sensor CO_2

El código para Arduino se adjunta en el anexo.

4.4.10 Módulo de pantalla LCD con ranura SD Arduino

La pantalla TFT Arduino es una pantalla LCD retroiluminada con encabezamientos. Puede representar texto, imágenes y formas en la pantalla utilizando la librería TFT. En la parte posterior de la pantalla hay una ranura de microSD integrada con la que es posible, entre otras cosas, grabar los datos en una tarjeta extraible. Los encabezamientos de la pantalla están diseñados para encajar en el conector hembra de la parte delantera de la Arduino Esplora, pero son compatibles con cualquier Arduino basado en AVR (Uno, Leonardo, etc.) o con el Arduino Due.

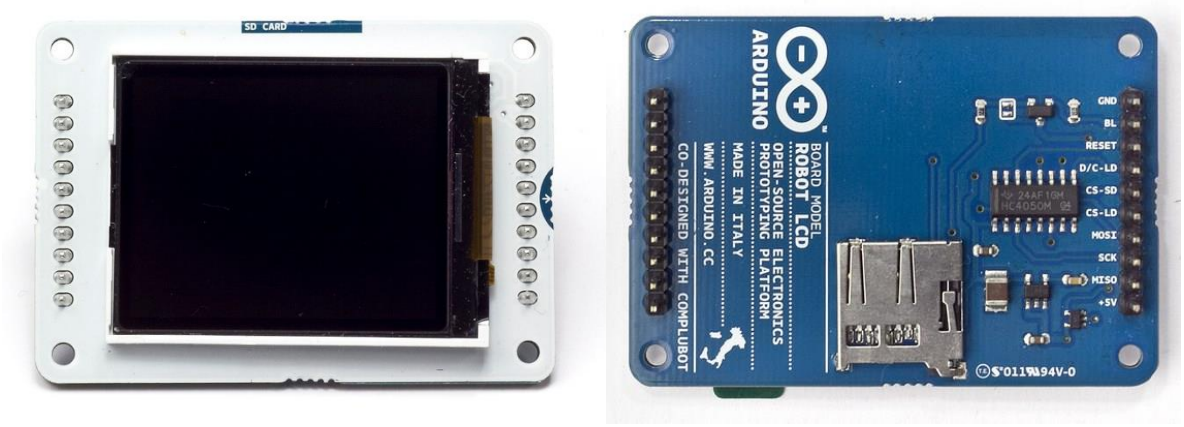


Figura 4-28 Apariencia frontal y trasera del módulo LCD

Especificaciones:

- 1.77 pulgadas.
- Display LCD con resolución de 160 x 128 píxeles.
- Interfaz: serie SPI.
- Tensión de funcionamiento: 5 V.
- Vdd= 2.7~3.3V.
- Conector para tarjeta microSD.
- La retroiluminación LED es atenuable y utiliza PWM.
- Dimensiones 7,6 x 5,1 x 2,5 cm.

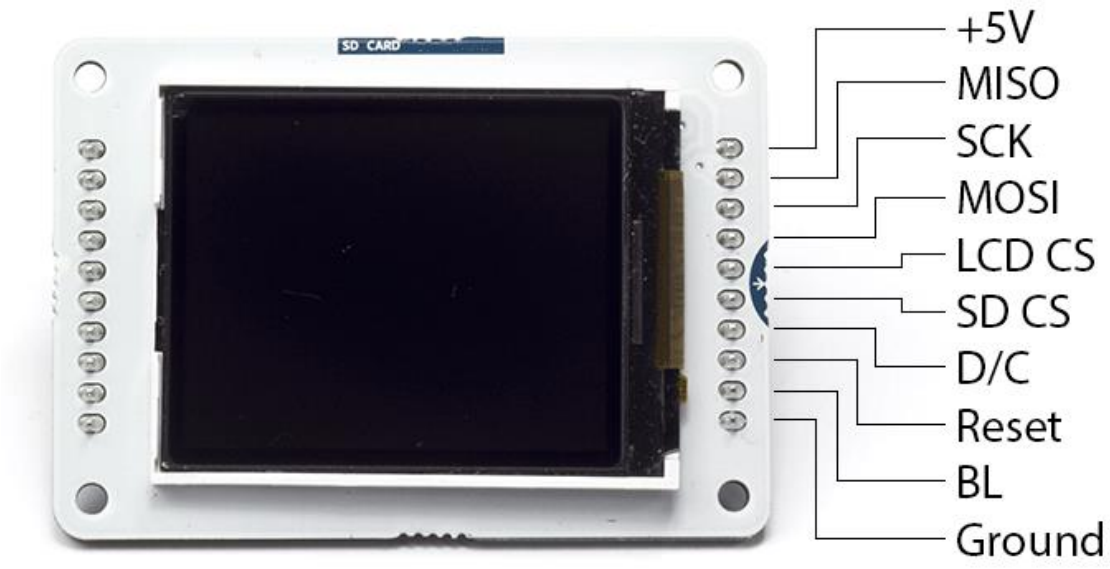
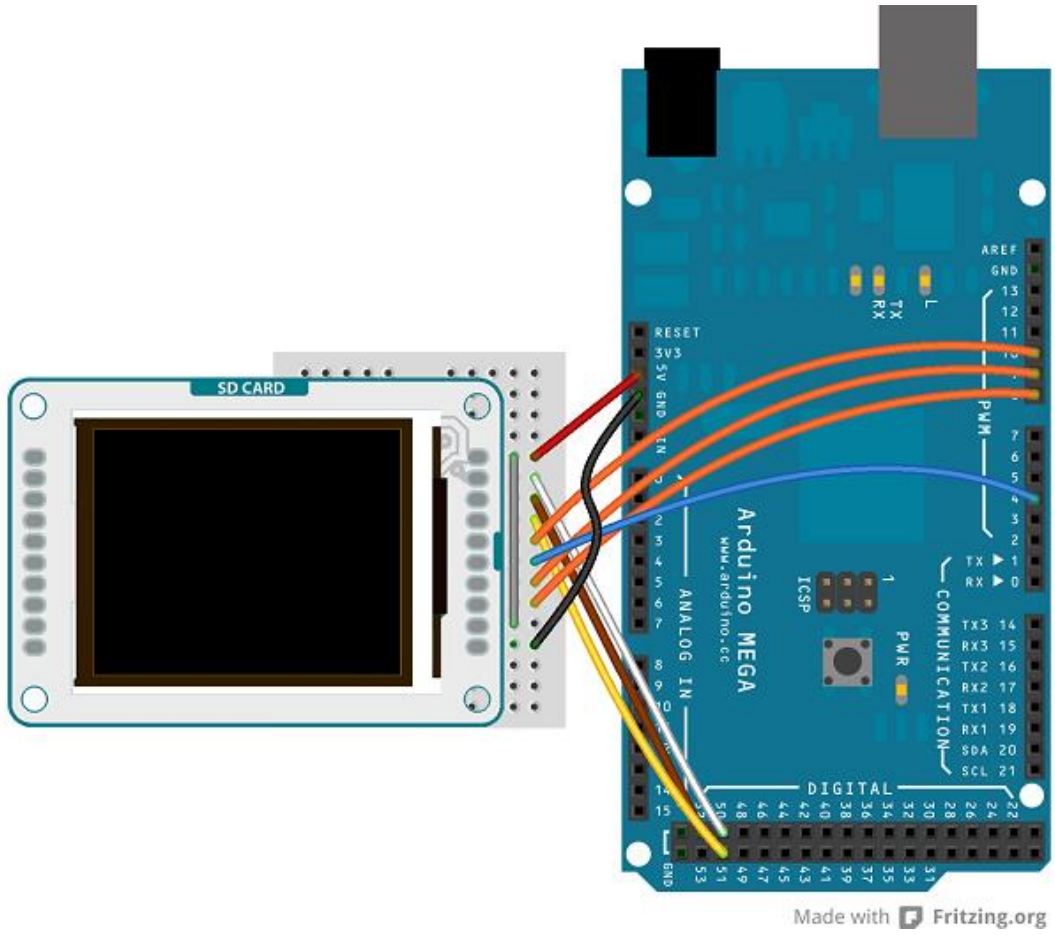


Figura 4-29 Conexiones del módulo LCD



Código 4-7 Esquema conexiones Pantalla LCD

A continuación se muestra un fragmento de código para Arduino con el que se podrá comprobar el funcionamiento del dispositivo.

```
#include <TFT.h>  // Arduino LCD library
#include <SPI.h>
#define cs   10   // pin definition for the Uno
#define dc   9
#define rst  8

// pin definition for the Leonardo
// #define cs   7
// #define dc   0
// #define rst  1

// create an instance of the library
TFT TFTscreen = TFT(cs, dc, rst);

// char array to print to the screen
char sensorPrintout[4];

void setup() {

  // Put this line at the beginning of every sketch that uses the GLCD:
  TFTscreen.begin();

  // clear the screen with a black background
  TFTscreen.background(0, 0, 0);

  // write the static text to the screen
  // set the font color to white
  TFTscreen.stroke(255,255,255);
  // set the font size
  TFTscreen.setTextSize(2);
  // write the text to the top left corner of the screen
  TFTscreen.text("Sensor Value :\n ",0,0);
  // ste the font size very large for the loop
  TFTscreen.setTextSize(5);
}

void loop() {

  // Read the value of the sensor on A0
  String sensorVal = String(analogRead(A0));

  // convert the reading to a char array
  sensorVal.toCharArray(sensorPrintout, 4);

  // set the font color
  TFTscreen.stroke(255,255,255);
  // print the sensor value
  TFTscreen.text(sensorPrintout, 0, 20);
  // wait for a moment
  delay(250);
  // erase the text you just wrote
  TFTscreen.stroke(0,0,0);
  TFTscreen.text(sensorPrintout, 0, 20);
}
```

Código 4-8 Pantalla LCD

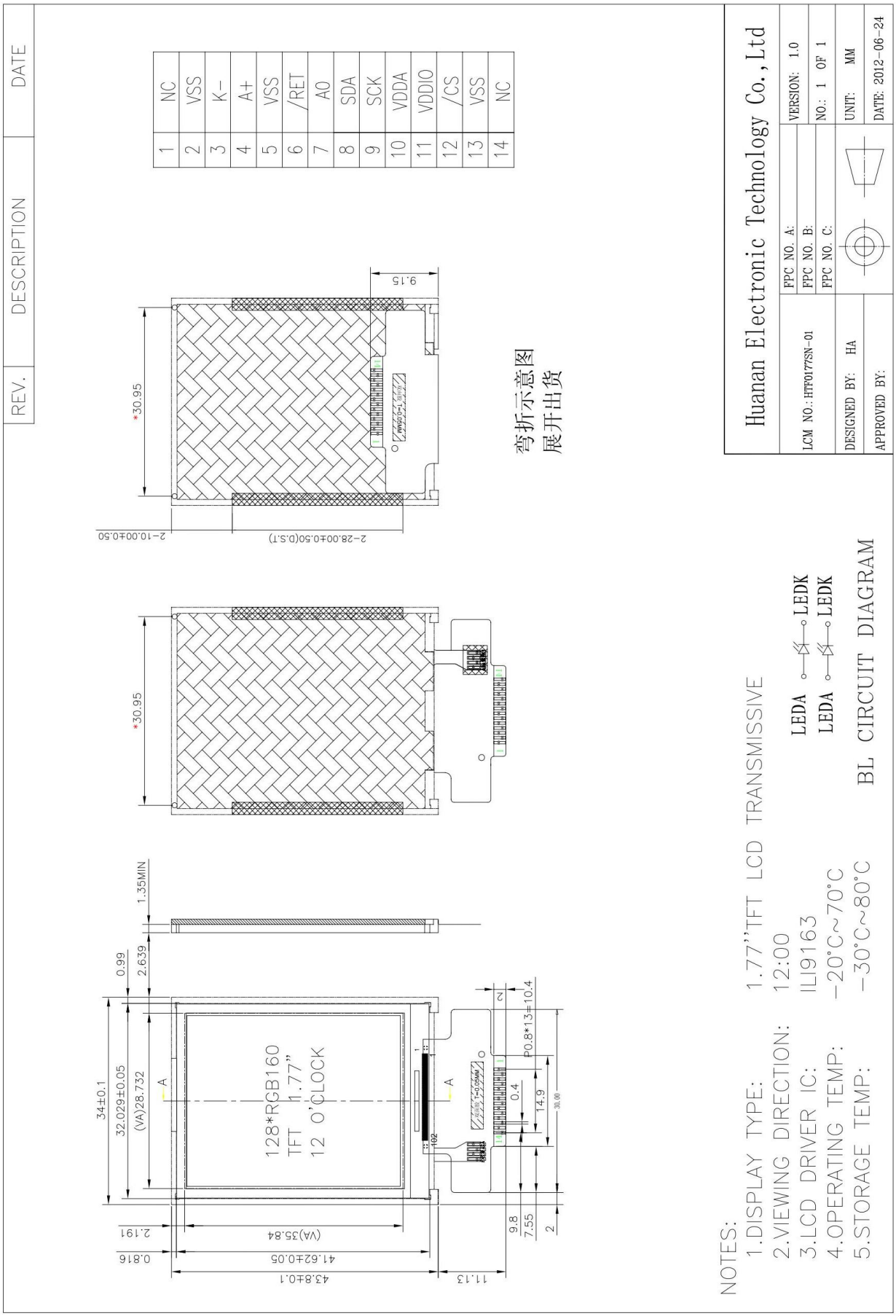


Figura 4-30 Plano del módulo LCD

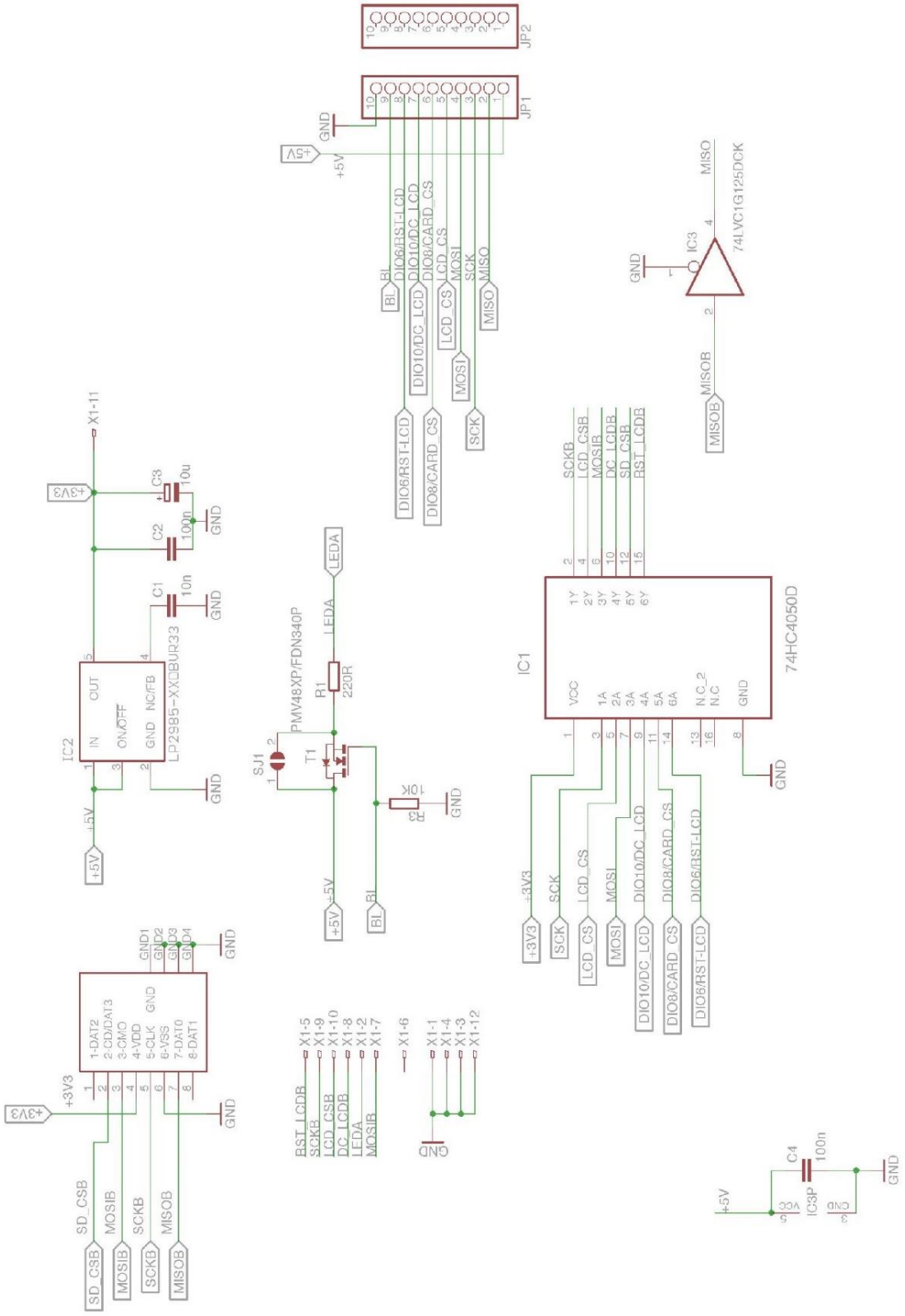


Figura 4-31 Esquemático del módulo LCD

4.4.11 Cámara IP Conceptronic CIPCAMPTIWL

Con el objetivo de poder ver el estado del laboratorio remoto se ha instalado una cámara IP accesible a través de una IP previamente asignada. De esta manera se podrá ver el estado de todos los actuadores a través de la imagen.

Figura 4-32 Cámara IP

Las principales características del dispositivo son:

- CMOS progresivo de 1/5", 640 x 480.
- Admite las funciones de panorámica de 340° e inclinación de 110°.
- Indicadores LED infrarrojos incorporados para visión nocturna.
- Micrófono y altavoz incorporados para comunicación de sonido 2-direccional.
- Admite detección de movimiento y notificación de incidentes por correo electrónico o FTP.
- Entrada/salida digitales (1DI/1DO) para sensor y alarma.
- Frencuencia de trabajo 2.4 GHz.
- Máxima velocidad soportada 150Mbps.
- Seguridad inalámbrica con cifrados WEP, WPA y WPA2.

- Conexión LAN cableada (10/100Mbps).
- Admite visualización remota desde iPhone/smartphone/PDA a través de Internet.

A continuación se muestra un esquema de cómo conectar esta cámara IP:

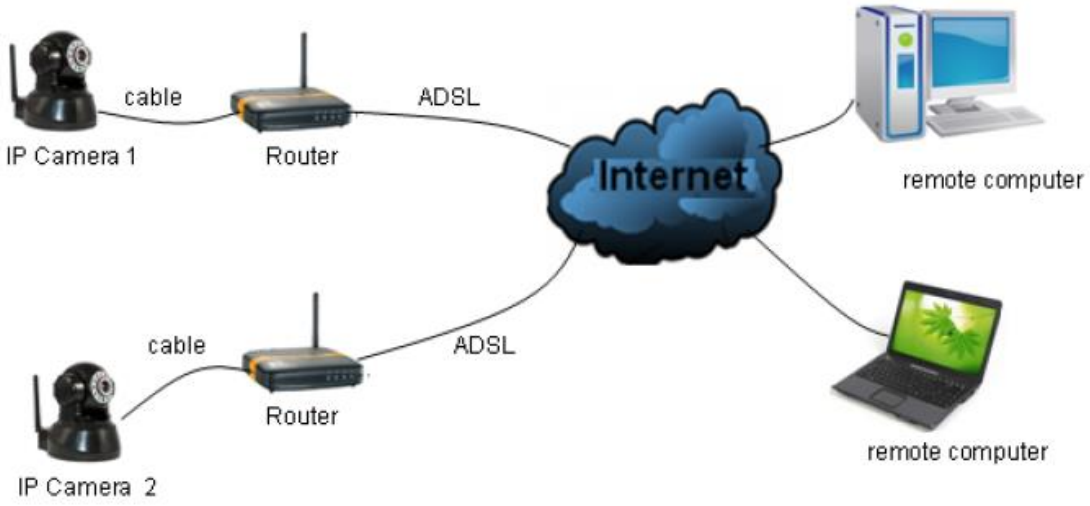


Figura 4-33 Conexión cámara IP

5 DESARROLLO DEL CÓDIGO PARA ARDUINO

5.1 El entorno de desarrollo (IDE).

Para poder subir el código con el que se programará el microcontrolador Arduino, se requiere descargar Arduino IDE. Para ello acceder a la siguiente URL: http://arduino.cc/en/Main/Software

Figura 5-1 Aspecto web oficial Arduino

Una vez descargado, se descromprimirá el contenido del archivo dando lugar a una carpeta donde se encuentran todos los archivos necesarios para que Arduino interactúe con el PC.

A veces se requiere la instalación de algunos drivers dentro de Windows, los cuales se encuentran en esta carpeta.

Es necesario copiar todo el contenido del archivo descomprimido en una carpeta en la ruta C:/Arduino/.

Para mayor comodidad se sugiere crear un acceso directo en el escritorio del archivo denominado Arduino.exe. Finalmente si se ejecuta el archivo Arduino.exe, aparecerá la siguiente ventana:

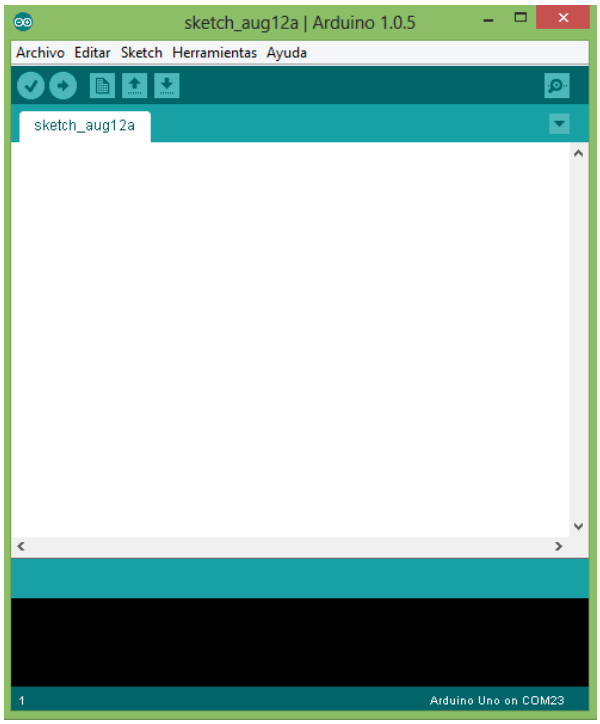


Figura 5-2 Aspecto IDE Arduino

Tras la conexión de la placa Arduino al puerto USB y si los drivers están el Sistema Operativo reconocerá el dispositivo. A continuación se debe ir al menú Herramientas/Tarjeta donde aparecerá toda la gama de productos Arduino. El usuario debe escoger el modelo que posee, en este caso Arduino Mega 2560.

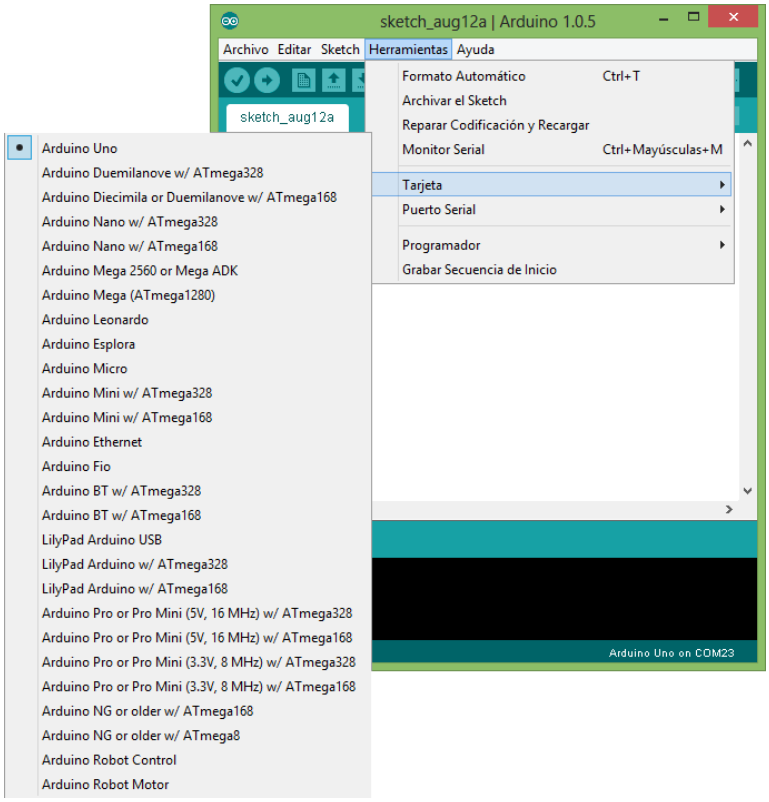


Figura 5-3 Selección tipo tarjeta

En el mismo menú de herramientas se debe escoger el puerto serie en el cual se encuentra el microcontrolador conectado. Para que esta opción se habilite es necesario que la placa esté conectada mediante USB y que los drivers funcionen correctamente.

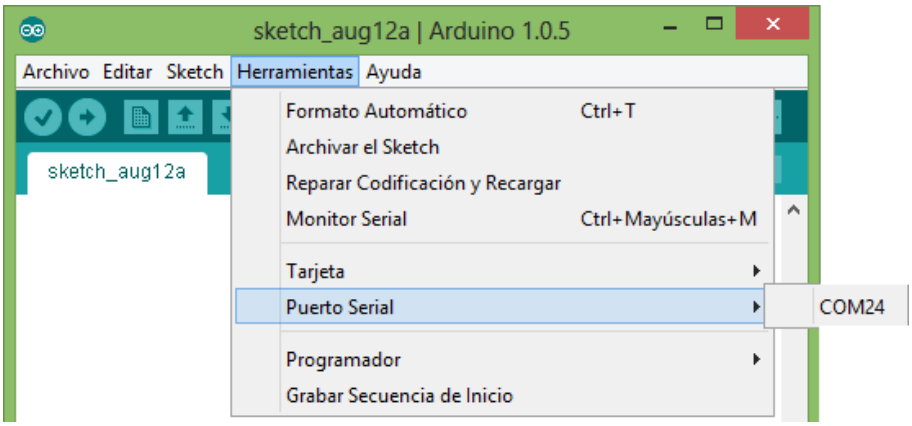


Figura 5-4 Selección Puerto Serial

En el menú Archivo hay ejemplos de código que pueden ser cargados a la placa para aprender sobre el funcionamiento de la misma. El entorno es muy amigable al usuario y no es complicado acostumbrarse a la interfaz y a sus funciones. Estas características hacen de Arduino el mejor de los dispositivos para aprender sobre el uso de microcontroladores.

Para comprobar que lo realizado hasta ahora es correcto y familiarizarse con el interfaz de desarrollo, se abre uno de los ejemplos. Es recomendable elegir el ejemplo “Blink”, ya que esta placa viene con un LED instalado en ella y existen buenas explicaciones del mismo, tanto en la página de Arduino, como en tutoriales de YouTube. Para abrir este ejemplo se tiene que acceder al menú Archivo->Ejemplos->Basics->Blink. Este ejemplo lo único que hace es hacer parpadear un diodo LED que está colocado en el pin 13 de la placa (véase Código 5-1).

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

```
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Código 5-1 Ejemplo Blink

Para pasar el código del programa a la placa Arduino primero se comprueba que es correcto, para ello se pulsa el botón de verificación de código que tiene la siguiente forma (✓) como se puede comprobar en la figura de más abajo. Si todo va bien aparecerá el mensaje "Compilación terminada". Una vez se tiene el código verificado se procede a cargarlo en la placa, para ello se tiene que pulsar el botón en forma de flecha (→) junto al botón de verificación. Durante la carga del programa a la placa se encenderán los LEDs de la placa Arduino que indican que se está enviando y recibiendo información por el puerto serie: TX/RX. Si la carga se ha realizado correctamente aparecerá el mensaje "Carga terminada" y transcurridos unos segundos se comprobará el correcto funcionamiento del código cargado, en este caso, se verá como el propio LED que tiene la placa conectado en el pin 13 se apaga y enciende.

Figura 5-5 Compilar y cargar

5.2 Código de programación Invernadero.ino

A continuación se describen todos los procesos dentro del sketch cargardo en el microcontrolador. En el void setup() se produce la carga de la librería DHT.h necesaría e imprescindible para el uso del sensor DHT11. Asi mismo se produce la llamada a dicho sensor indicando el tipo del mismo. Posteriormente se realiza la definición tanto de pines de entrada como de pines de salida.

Por otra parte se lleva a cabo tanto la declaración como la inicialización de las variables que intervienen en el control. Posteriormente se indica el periodo con el que se registran los datos de los sensores.

A partir de este momento el programa ya está totalmente configurado y listo para el envio/recepción de datos. Tras realizar la lectura de todos los sensores y su acondicionamiento se produce el envio de todos estos valores a través de comunicación serial para que así puedan ser escuchados por cualquier otra aplicación.

Por otro lado se comprueba si hay algún dato disponible para ser recibido, en caso positivo se realiza la lectura de todos los parámetros recibidos y según el valor de éstos se lleva a cabo un control manual o automático de la planta de proceso.

Para ver más detalladamente todos los procesos consultad el anexo incluido en esta memoria donde se describe todo el código programado.

6 INSTALACIÓN Y DESARROLLO DEL SERVIDOR XAMPP

XAMPP es un servidor independiente de plataforma y de software libre que está compuesto principalmente de la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script, como son PHP y Perl. Su curioso nombre no es más que un acrónimo que proviene de: X (multiplataforma), Apache, MySQL, PHP y Pearl. El hecho de que sea un servidor multiplataforma, viene a decir que funciona igual para cualquiera de los sistemas operativos existentes en el mercado.

En este caso, el servidor donde se ha instalado la base de datos funciona bajo el sistema operativo de Windows y el paquete XAMPP para dicho sistema operativo incluye lo siguiente:

- Servidor Web Apache: La base de datos del servidor web.
- MySQL: Motor de base de datos muy popular.
- PHP: Uno de los lenguajes de programación más populares.
- phpMyAdmin: Aplicación web para gestionar bases de datos desde el navegador.
- FilleZilla FTP Server: Servidor de FTP.
- Tomcat: Servidor web y contenedor de servlets Java.
- Strawberry Perl: intérprete de Pearl para Windows.
- XAMPP Control Panel: Panel de control propio desde el cual se controla todo lo anteriormente descrito.

Para su instalación, se debe acceder a la página de descarga oficial https://www.apachefriends.org/index.html y escoger el sistema operativo adecuado, posteriormente se iniciará la descarga.

Tras la descarga se ejecutará el paquete instalador. Saltará una notificación de la presencia de antivirus (si es que hay alguno instalado) en el PC y de la posibilidad de que interfiera en la instalación, pero aun así se debe continuar con la instalación.

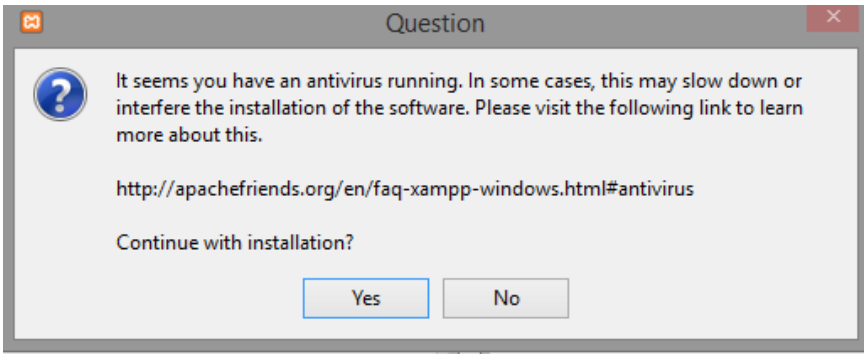


Figura 6-1 Aviso de antivirus

A continuación, aparece el asistente de instalación que ofrece la posibilidad de seleccionar los componentes que se desean instalar pero no es necesario cambiar nada, ya que por defecto se instalan todos. Sin embargo, se permite la posibilidad de prescindir de alguno de ellos.

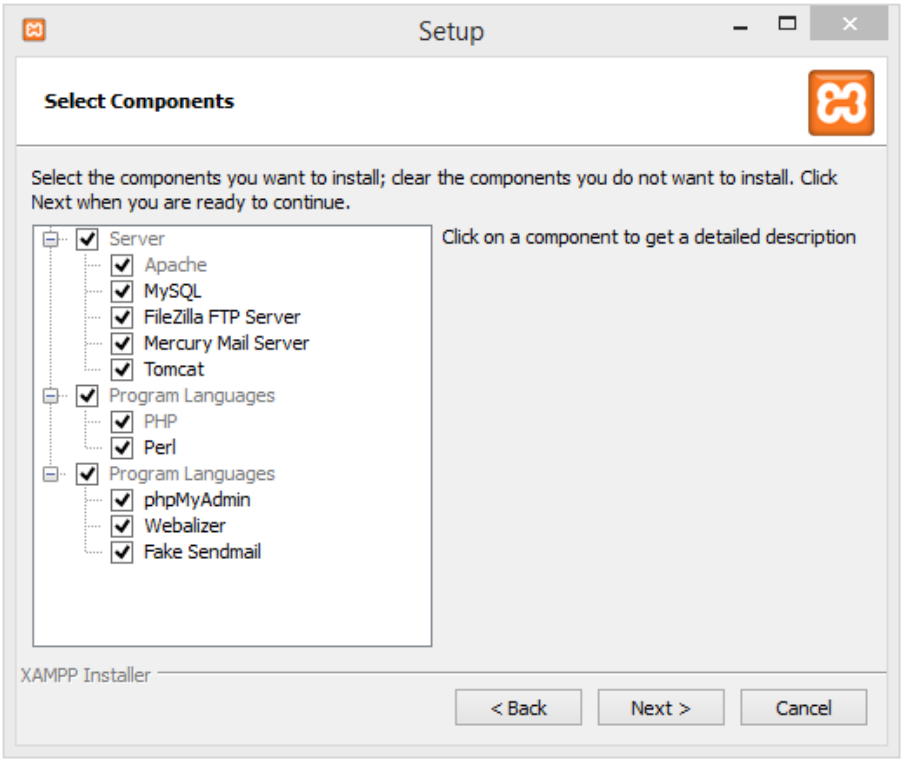


Figura 6-2 Asistente de instalación

Por último, se debe seleccionar el directorio donde se instalará el contenido y pulsar *Next* en las ventanas posteriores hasta comenzar con la instalación.

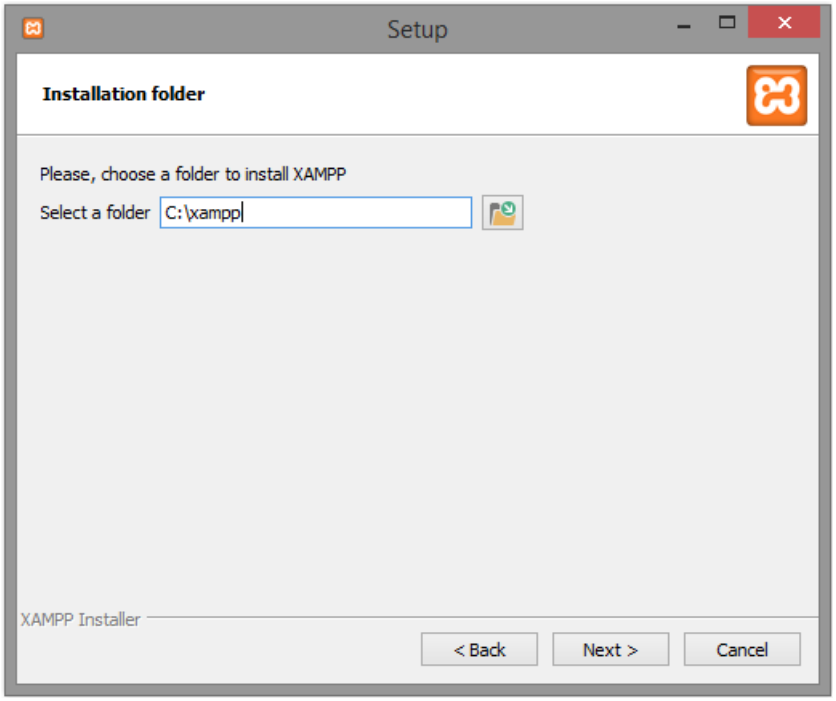


Figura 6-3 Selección de la carpeta de destino

Una vez finalizada la instalación, se abrirá automáticamente el panel de control de XAMPP (de no abrirse, se deberá de hacer de manera manual).

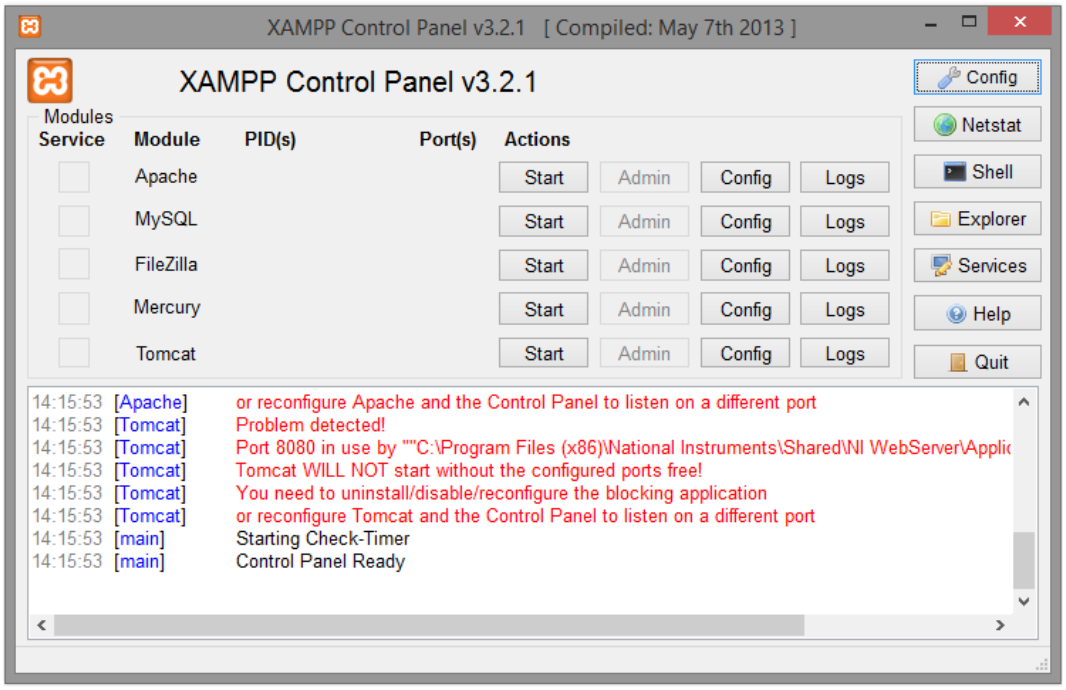


Figura 6-4 Panel de control de XAMPP

Como se puede apreciar, cada fila hace referencia a un módulo: Apache, MySQL, FilleZIlla, Mercury o Tomcat. Para el uso requerido (bases de datos) solamente será necesario activar Apache y por supuesto, MySQL haciendo clic en el botón Start.

Si esto último se ha hecho de manera correcta, el color de dichos servicios habrá pasado a color verde lo cual significa que el servidor está listo.

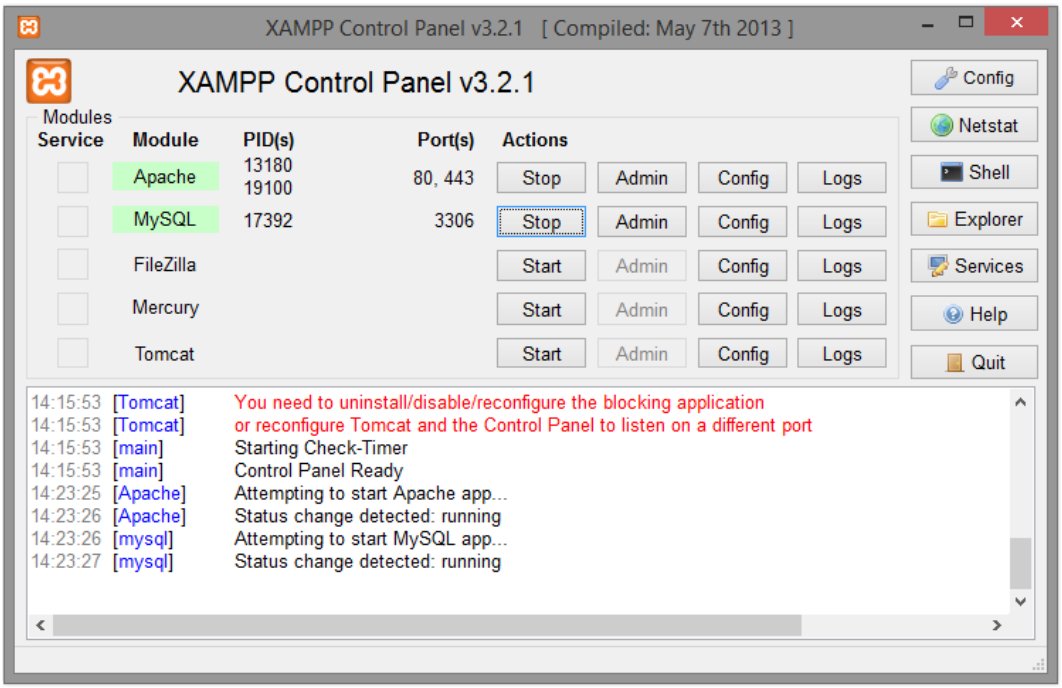


Figura 6-5 Módulos activos a través del panel de control de XAMPP

6.1 Configuración de XAMPP

Con el fin de evitar futuros accesos indeseados al servidor instalado, la mejor solución es configurarlo con contraseñas. Para realizar esto se debe abrir el navegador y escribir lo siguiente:http://localhost. Con esto se conseguirá acceder a la pantalla principal (seleccionando previamente el idioma deseado).

Figura 6-6 Pantalla principal de XAMPP

En primer lugar, se deben inspeccionar las diferentes opciones ubicadas en el menú de la izquierda que son útiles para la configuración. Tras clicar en el apartado *Estado* se podrá observar una lista de los componentes instalados en el equipo así como el estado de cada uno de ellos. Esta pestaña es únicamente informativa, por lo que no servirá para mucho más.

Figura 6-7 Pestaña de estado

A continuación se clica en la opción de *Chequeo de seguridad* la cual tiene más importancia. Se debe tener en cuenta que en el momento en que el equipo esté conectado a Internet, cualquiera podría acceder a él debido al reciente servidor web creado..

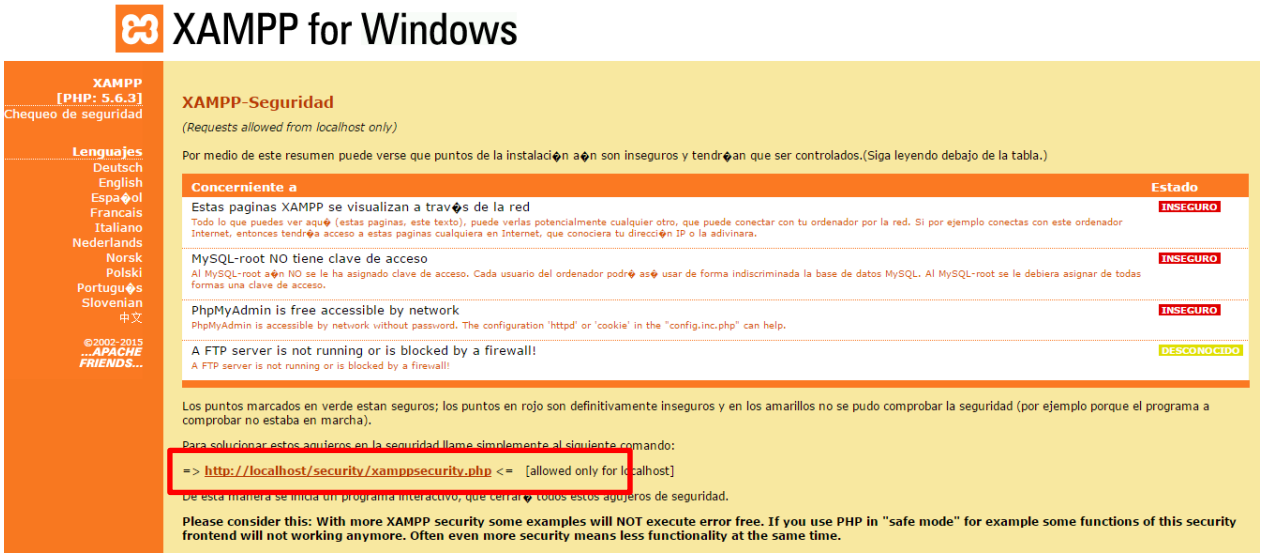


Figura 6-8 Pestaña Chequeo de seguridad

Para proteger el equipo de esta posible amenaza, se debe entrar en esta sección donde se encuentra al final de la misma un enlace: http://localhost/security/xamppsecurity.php, tras acceder a él se podra fijar una clave para restringir el acceso al equipo.

Figura 6-9 Pantalla de seguridad

Una vez dentro, se recomienda implantar la contraseña para el usuario *Root* (predefinido) de la base de datos. Debido a la posibilidad del olvido de la contraseña existe una opción para guardar ésta en un archivo de texto.

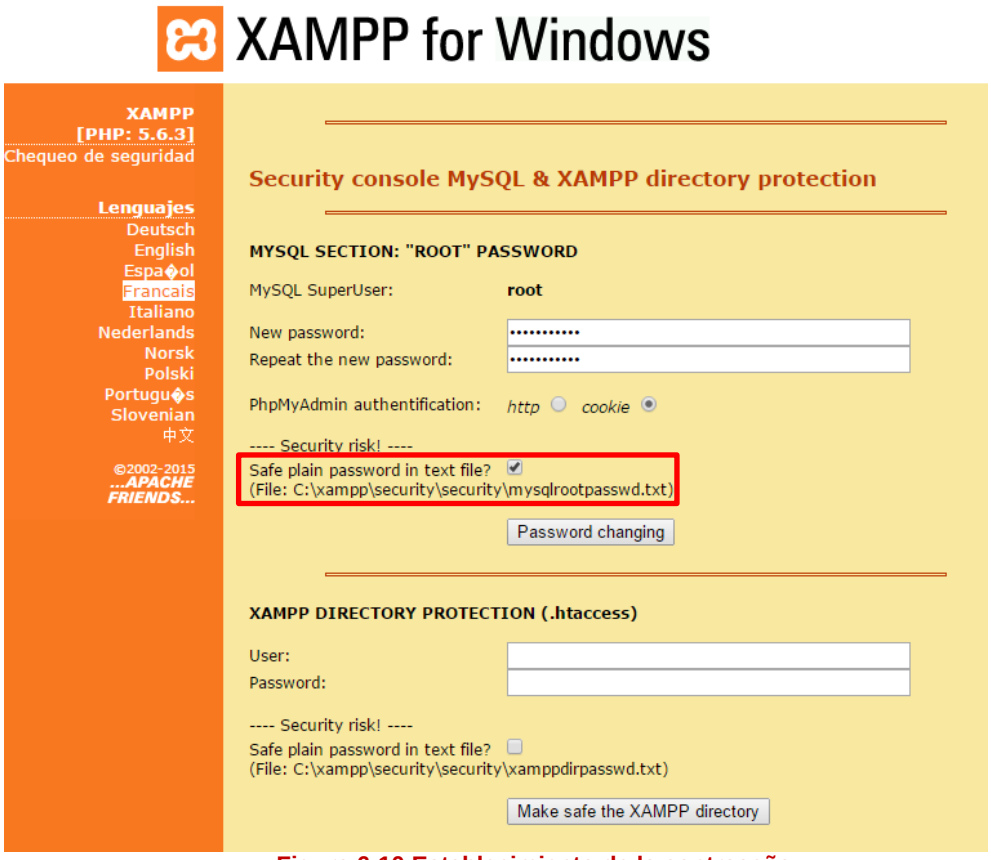


Figura 6-10 Establecimiento de la contraseña

Una vez que la contraseña haya sido escrita, se procederá a pinchar en *Password changing* y acto seguido se podrá observar en la misma pantalla que la contraseña ha sido cambiada con éxito.

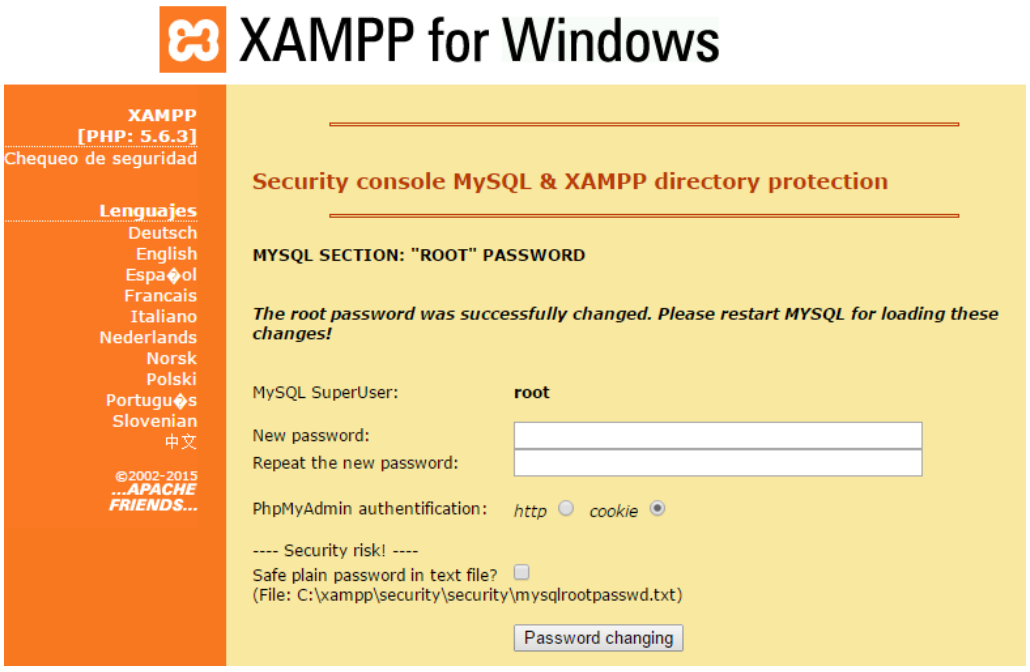


Figura 6-11 Mensaje de comprobación

A continuación y sin cambiar de pantalla, se procederá a configurar el archivo .htaccess, mediante el cual se protegerá el acceso al sitio web local. Para ello escribir un nombre de usuario (el que se desee), una contraseña y hacer click en el botón de *Make safe the XAMPP directory*.

Figura 6-12 Configuración del archivo .htaccess

Tras haber realizado estos sencillos pasos se tendrá el servidor totalmente asegurado, prueba de ello son los cambios que ahora se pueden apreciar en la pestaña de *Chequeo de seguridad*.

Figura 6-13 Cambios en la pestaña Chequeo de seguridad

6.2 Creación de una base de datos con XAMPP

Con XAMPP ya instalado y funcionando correctamente en el servidor se procederá ahora a la creacción de una base de datos. Tras abrir el Control Panel inicialmente instalado se debe ejecutar tanto Apache como MySQL. Posteriormente se debe pulsar el botón admin en la línea correspondiente a MySQL. Se abrirá el navegador para llevar a cabo la identificación

Figura 6-14 Identificación

Posteriormente se mostrará la página principal de phpMyAdmin.

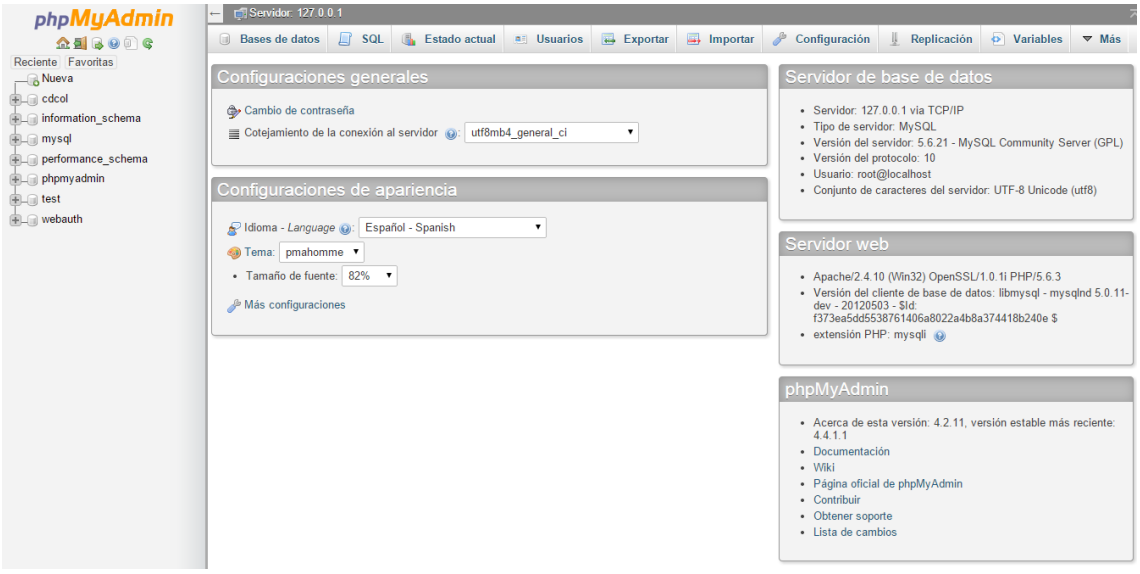


Figura 6-15 Página principal phpMyAdmin

Para crear una base de datos se debe pinchar en la pestaña de Bases de datos donde se podrá observar algo similar a la siguiente imagen.

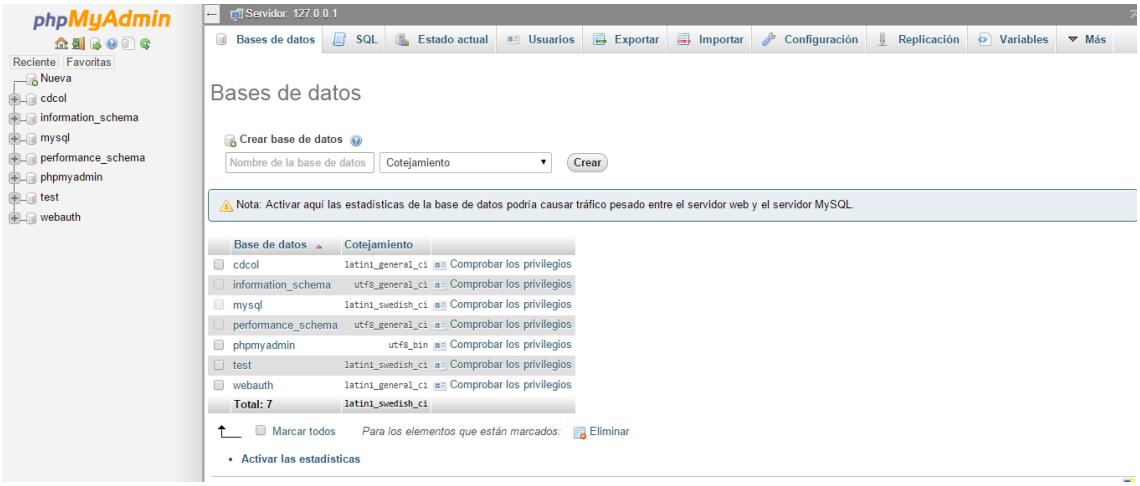


Figura 6-16 Pestaña de base de datos

Se debe introducir el nombre de la base de datos (“cajadecristal”) y acto seguido pulsar sobre el botón de Crear.

Se podrá observar en el menú de la parte izquierda como aparece dicha base de datos, pinchar sobre ella para dirigirse ahora a la creación de las tablas que serán útiles para este proyecto.

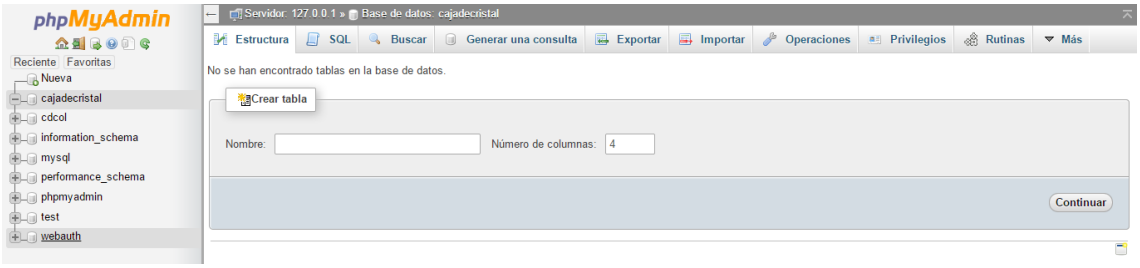
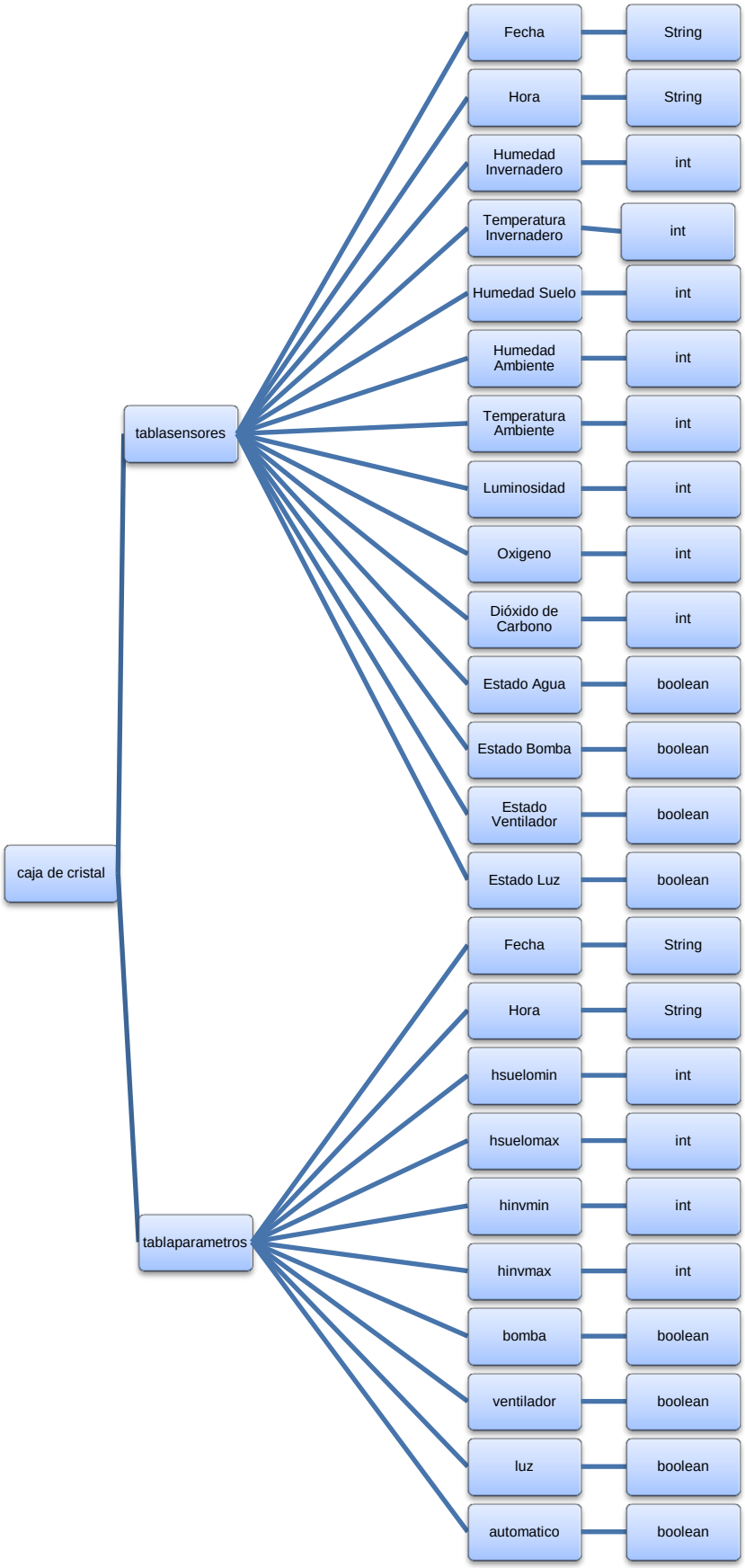


Figura 6-17 Creación de tablas

Son necesarias dos tablas, una que recoja los datos de los distintos sensores llamada tablasensores y otra tabla que almacena los parámetros de trabajo y que se llamará tablaparametros. Es importante tener en cuenta que el entorno MySQL no admite las mayúsculas.

Durante el proceso de creación de las tablas, se debe introducir la opción utf8_spanish_ci en el campo de cotejamiento. De esta manera, se selecciona una codificación que admite símbolos españoles como las tildes o la ‘ñ’.

6.3 Estructura de la base de datos



6.4 Configuración acceso remoto a la base de datos

Ahora mismo, se tiene una base de datos a la que solo se puede acceder de manera local, pero en este proyecto se requiere que cualquier persona pueda acceder a la base de datos anteriormente creada. Para ello, se debe configurar un usuario remoto que permita este acceso.

Antes de nada, es necesario modificar algunos archivos de texto que restringen el acceso remoto. Hay que acceder al *Control Panel* y en la fila de *MySQL* pinchar en el botón de *Config* y en la opción de *my.ini*.

Se abrirá el archivo de texto correspondiente donde habrá que fijarse en dos aspectos importantes: *bind-adress* y *skip-networking*. Referente al primero, es necesario cambiar la dirección IP que incorpora por defecto (127.0.0.1) por 0.0.0.0 para habilitar a cualquier IP y eliminar la # que incorpora, para habilitar dicha opción.

Por otro lado, en la fila donde se mencione el *skip-networking* habrá que asegurarse de que la # está presente.

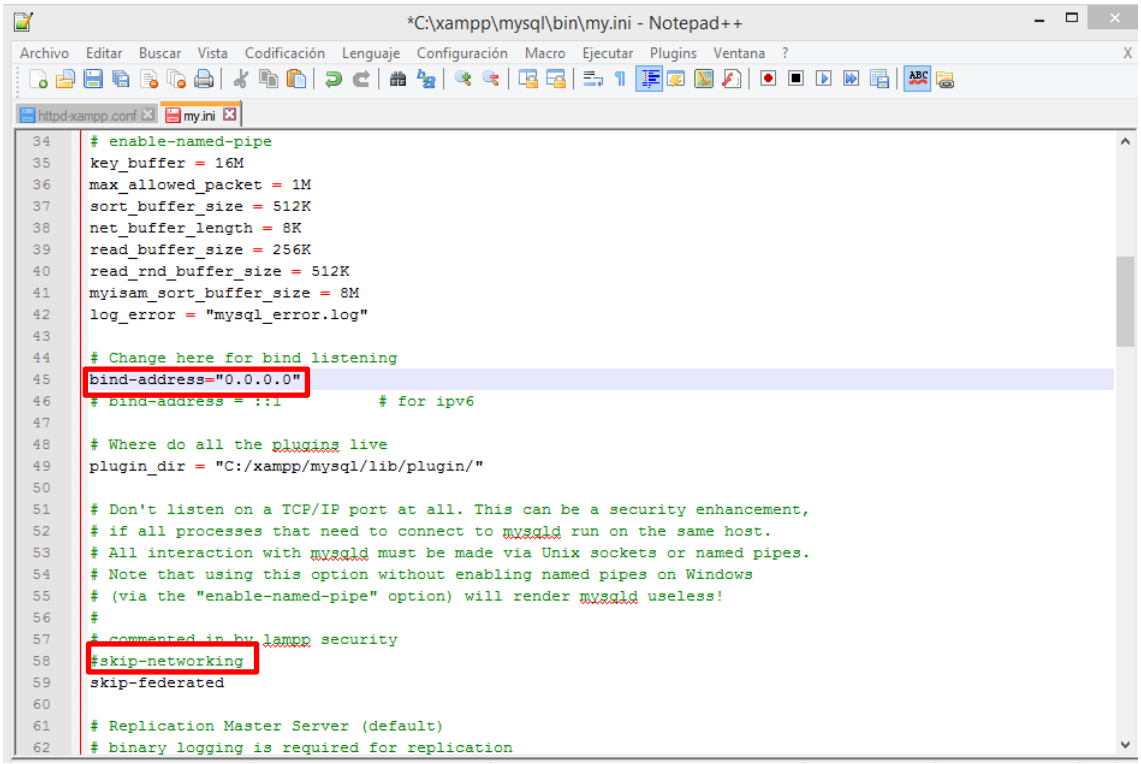


Figura 6-18 Modificación del archivo my.ini

Otro archivo que es necesario modificar está en la ruta C:\xampp\apache\conf\extra y se llama *httpd-xampp.conf*, al abrirlo dirigirse al final del archivo donde pone *Require Local* y colocar una # delante con el fin de ignorar esta instrucción.

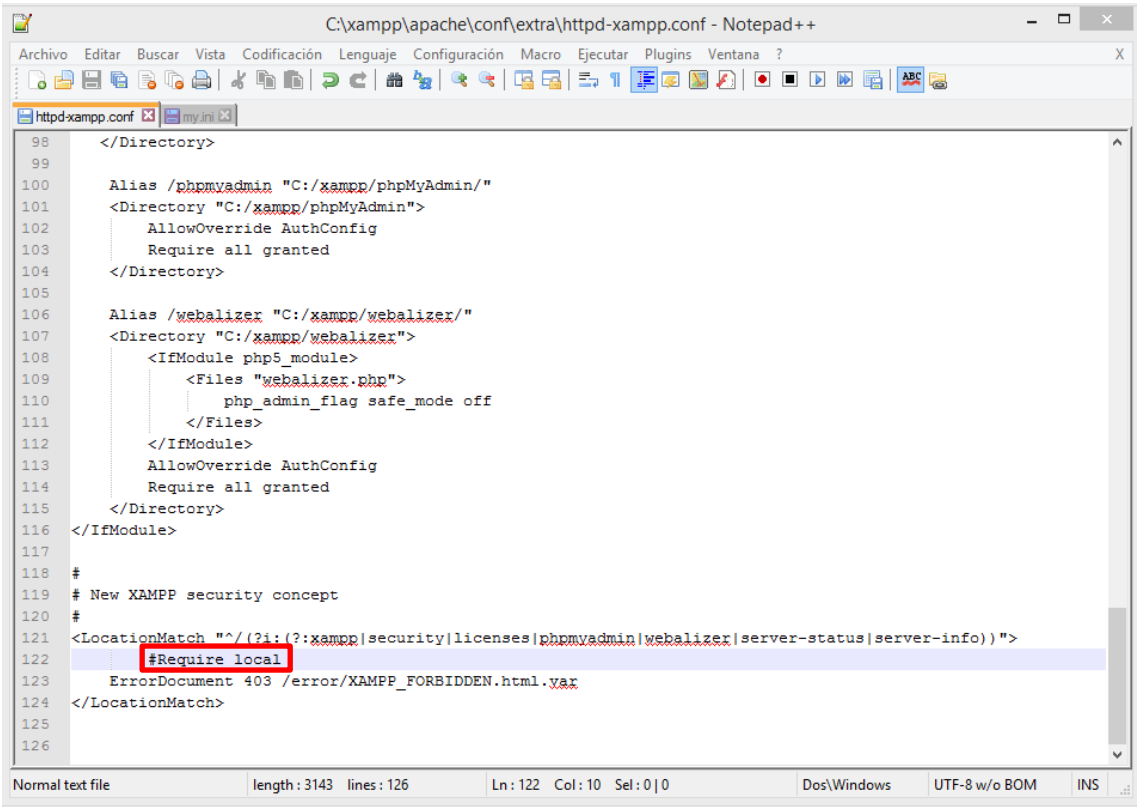


Figura 6-19 Configuración del archivo httpd-xampp.conf

Una vez modificados estos dos archivos, hay que pasar a la creación del usuario que actuará como remoto dentro de *MySQL*. Para ello, dentro de la pantalla principal de *phpMyAdmin* pinchar en la pestaña *SQL* y escribir las siguientes líneas:

```
CREATE USER 'remoto'@'%' IDENTIFIED BY 'invernadero';
GRANT ALL ON cajadecristal.* TO 'remoto'@'%';
```

Código 6-1 Creación usuario MySQL

Los campos de *nombre_usuario* y *contraseña* se elegirán según cada uno, mientras que el campo *nombre_BD* hace referencia a la base de datos a la cual se da acceso a este usuario.

Es importante destacar el símbolo % que indica que cualquier IP podrá acceder a este usuario.

Por último, se podrá comprobar que el usuario creado tiene todos los privilegios para trabajar en la base de datos a la que se le ha dado el acceso.

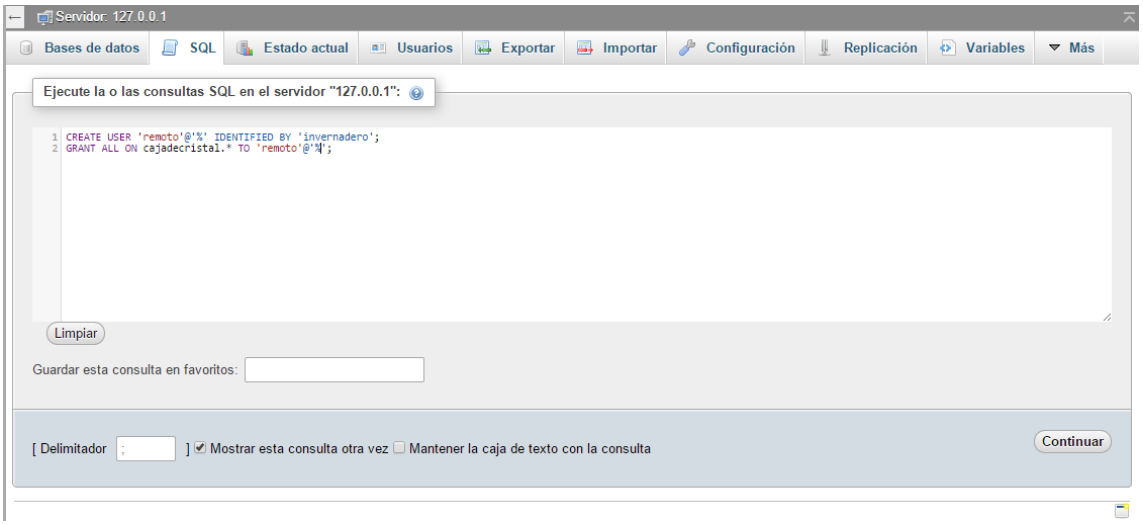


Figura 6-20 Creación de usuario remoto

El puerto por defecto que tiene asignado mySQL es el 3306 sin embargo para realizar el acceso correctamente se cambiarña al 4001 para ello hay que modificar el fichero my.ini.

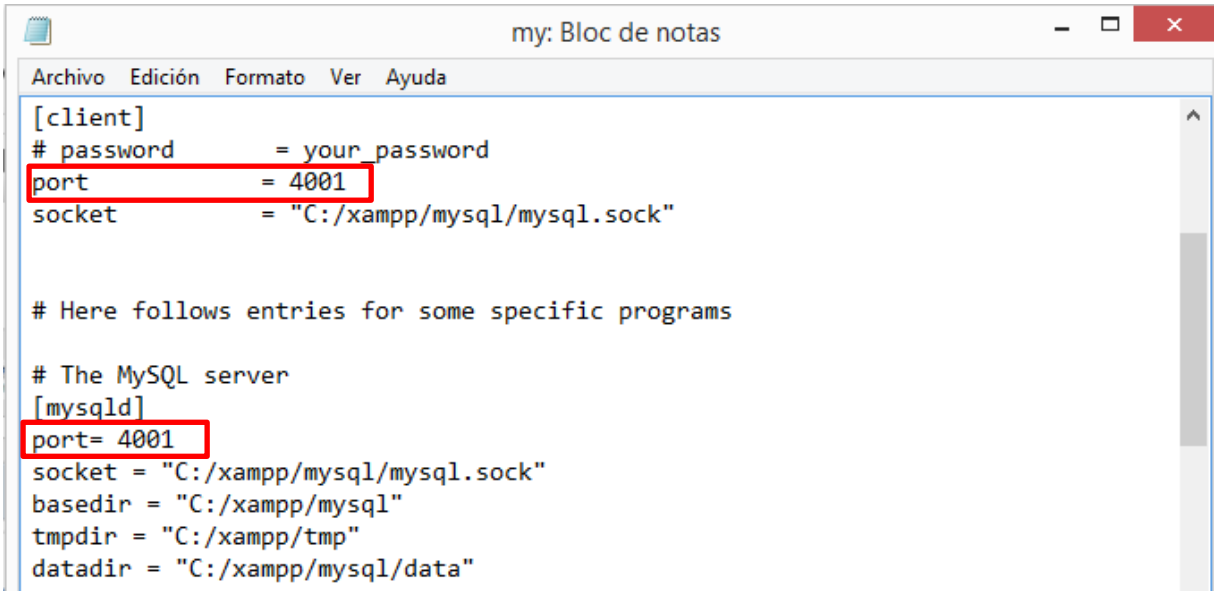


Figura 6-21 Modificación puerto

7 DESARROLLO DE LOS CÓDIGOS SERVIDOR.JAR Y SRAV.JAR

7.1 Instalación del entorno de desarrollo (IDE) NetBeans.

Para comenzar con el proceso de implementación de las aplicaciones cliente y servidor, en primer lugar se debe elegir el Integrated Development Environment o IDE. Cuando se habla de IDE, se habla de un programa que permite desarrollar código en un lenguaje y que incorpora habitualmente:

- Un espacio para la escritura de código con cierta ayuda interactiva para generar código y para indicar los errores de sintaxis que se cometan por parte del programador.
- La posibilidad de compilar y ejecutar el código escrito.
- La posibilidad de organizar los proyectos de programación.
- Herramientas auxiliares para programadores para detección de errores o análisis de programas (debuggers).
- Otras opciones como utilidades para pruebas, carga de librerías, etc.

Se ha optado por NetBeans porque es un entorno de desarrollo muy completo, cómodo de usar, y que dispone de muchas funcionalidades. Se podría haber usado otros IDEs como: Eclipse, BlueJ, JBuilder, JCreator.

Para el correcto funcionamiento de este entorno de desarrollo, previamente tiene que estar instalado el JDK (Java Development Kit) que es la herramienta básica para crear programas usando el lenguaje Java. Es gratuito y se puede descargar desde la página oficial de Java, en el sitio web de Oracle (el actual propietario de esta tecnología, tras haber adquirido Sun, la empresa que creó Java: http://www.oracle.com/technetwork/java/javase/downloads.

Se adjunta un anexo con todos los pasos para la instalación de este software.

7.2 Librería mySQLconnector

La librería para java de MySQL (mysql-connector-java-5.1.34) permite realizar una conexión a cualquier base de datos de mySQL desde java para realizar consultas o incluir nuevos datos en ella.

En el software desarrollado en java, todos los imports necesarios para manejar la base de datos están en java.sql.*. Puesto que casi todos los métodos relativos a base de datos pueden lanzar la excepción SQLException, se debe meter todo el programa en un try-catch.

Además, se necesita la clase org.gjt.mm.mysql.Driver que viene con el driver de MySQL. Por ello se debe incluir el jar que contiene el driver MySQL (mysql-connector-java-5.1.34) o la versión más moderna y compatible con la versión usada de MySQL.

7.2.1 Instalar el driver

Lo primero que se tiene que hacer es asegurarse que el Driver se inicializa y se registra, para ello se procede a escribir el código que se muestra a continuación:

```
try
{
   Class.forName("com.mysql.jdbc.Driver");

}
catch (ClassNotFoundException ex)
{
    Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE, null, ex);
}
```

Código 7-1 Instalación driver MySQL

7.2.2 Establecer la conexión con la base de datos

Se debe tener el servidor de MySQL arrancado. Si se ha instalado y dejado esa opción como estaba, cada vez que se encienda el ordenador, se arrancará el servidor de MySQL, por lo que no hay que preocuparse por ello.

El servidor de MySQL abre por defecto el puerto 3306 para aceptar conexiones de posibles clientes, de programas que quieran conectarse y acceder a la base de datos. El programa java, si quiere consultar la tabla de base de datos que se ha creado, deberá conectarse a este servidor.

Para establecer la conexion, la clase DriverManager tiene el método getConnection().

```
Connection conectar = DriverManager.getConnection ("jdbc:mysql://localhost/
nombrebasededatos" ,"user","password");
```

Código 7-2 Establecimiento de la conexión

El primer parámetro del método getConnection() es un String que contiene la url de la base de datos:

- jdb:mysql porque se está utilizando un driver jdbc para MySQL, que es el que se ha descargado.
- localhost porque el servidor de base de datos está en el mismo ordenador en el que voy a ejecutar el programa java. Aquí se puede poner una IP o un nombre de máquina que esté en la red.
- nombrebasededatos es el nombre de la base de datos que he creado dentro de MySQL. Se debe poner la base de datos dentro del servidor de MySQL a la que se quiere realizar la conexión. Es el nombre asignado cuando desde SQL se ejecutó “create database cajadecristal”.
- “user” String que corresponde al nombre de usuario.
- “password” String que corresponde al password para acceder a la base de datos.

Si todo va bien, en conectar se tendrá una conexión a la base de datos.

Esta conexión es en realidad un socket entre java y la base de datos. Lo que sí es importante, es saber que si varios hilos comparten esta conexión, deben usarla sincronizadamente. Si no se hace así, los mensajes que van por el socket se pueden entremezclar y los hilos pueden leer fragmentos de mensaje destinados al otro hilo. Otra opción es que cada hilo cree su propia conexión.

7.2.3 Realizar una consulta

Para realizar cualquier acción sobre la base de datos (consulta, insertar nuevos registros, modificar los existentes o borrar), es necesario usar una clase Statement. Para obtenerla, se le pide dicha clase a la conexión. La forma de hacerlo, para una consulta, es la siguiente:

```
Statement stm=cn.createStatement();
ResultSet rset=stm.executeQuery("SELECT * FROM nombre_tabla");
```

Código 7-3 Creación consulta

La parte de createStatement() no tiene ningún secreto, salvo que puede lanzar una excepción que hay que capturar. El Statement obtenido tiene un método executeQuery(). Este método sirve para realizar una consulta a la base de datos.

El parámetro que se pasa es un String en el que está la consulta en lenguaje SQL. En este caso "SELECT * FROM nombre_tabla" siendo nombre_tabla el nombre que se ha puesto a la tabla existente en la base de datos.

El resultado lo devuelve el método como un ResultSet. Este ResultSet no es más que una clase java similar a una lista en la que está el resultado de la consulta. Cada elemento de la lista es uno de los registros de la base de datos. En realidad, ResulSet no contiene todos los datos, sino que los va consiguiendo de la base de datos según se van pidiendo. Por ello, el método executeQuery() puede tardar un poco, pero al recorrer los elementos del ResultSet no es tan rápido. De esta forma se evita que una consulta que dé muchos resultados tarde mucho tiempo y llene la memoria del programa java.

7.2.4 Leer los resultados

El ResultSet contiene dentro los registros leídos de la base de datos. Inicialmente, tal cual lo devuelve el Statement.executeQuery(), tiene internamente un "puntero" apuntando justo delante del primer registro. El método next() del ResultSet hace que dicho puntero avance al siguiente registro, en este caso, al primero. Si lo consigue, el método next() devuelve true. Si no lo consigue (no hay siguiente registro que leer), devuelve false. El método absolute(fila) sitúa el “puntero” en la fila deseada. Existe otro método last() que desplaza el “puntero” al último registro.

Por tanto, una forma de ir leyendo los registros es mediante un while (véase Código 7-4).

```
while(rset.next())   //Hago la lectura de todas las filas
{
datoString=rset.getString(1);
datoInt=rset.getInt(2);
datoBoolean=rset.getBoolean(3);
}
```

Código 7-4 Lectura de datos

Una vez que el "puntero" está apuntando a un registro, los métodos getInt(), getString(), getBoolean(), etc van devolviendo los valores de los campos de dicho registro. Tambien se puden pasar a estos métodos un índice (que comienza en 1) para indicar qué columna de la tabla de la base de datos es la deseada.

7.2.5 Escribir valores en una tabla

Cuando se trabaja con una base de datos es posible que haya sentencias SQL que se tengan que ejecutar varias veces durante la sesión, aunque sea con distintos parámetros. Por ejemplo, durante una sesión con base de datos se puede querer insertar varios registros en una tabla. Cada vez los datos que se insertan serán distintos, pero la sentencia SQL será la misma: Un INSERT sobre determinada tabla que será siempre igual, salvo los valores concretos a insertar.

Una vez establecida la conexión, es necesario crear el PreparedStatement llamando al método prepareStatement() de la Connection. Puesto que los PreparedStatement son importantes cuando utilizan varias veces para ganar en eficiencia, es importante guardar este PreparedStatement en algún sitio al que se pueda acceder cuando sea necesario. Si cada vez que se usa se crea un PreparedStatement nuevo, no se conseguirá la mejora de eficiencia. Un buen sitio para guardar este PreparedStatement puede ser un atributo de la clase.

Una vez creado y guardado se podrá usar siempre que sea necesario. Para ello, primero hay que darle valor a los parámetros que están como interrogantes. Se usarán los método set() de que dispone PreparedStatement. Una vez introducidos todos los datos se ejecuta executeUpdate(). La inserción quedaría entonces:

```
try
{
      PreparedStatement pstm=cn.prepareStatement("INSERT INTO nombre_tabla
      VALUES(?,?,?)");

      pstm.setString(1, datoString);
      pstm.setInt(2, datoInt);
      pstm.setBoolean(3, datoBoolean);

      pstm.executeUpdate();
 }

catch (SQLException ex)
{
    System.out.println("Error de escritura en mySQL");
    Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE, null, ex);
}
```

Código 7-5 Escritura de datos

7.2.6 Borrar una tabla

Una tarea habitual con bases de datos será el tener que borrar cierta información. A continuación se describe cómo realizar el borrado de ciertos registros que cumplan una condición y el borrado completo de todos los registros (filas) de una tabla. Para ello se utilizarán dos expresiones para operaciones (consultas) sobre bases de datos que se describen a continuación: delete y truncate.

La sentencia DELETE se suele usar para borrar unos registros de una tabla que cumplen una o varias condiciones. Se va a utilizar una sintaxis de este tipo:

```
("DELETE FROM nombre_Tabla WHERE columna (>, <, =) valorEspecificado ");
```

Código 7-6 Delete

En este caso se está empleando la cláusula WHERE que sirve para indicar una condición. Por ejemplo DELETE FROM tablasensores WHERE luminosidad > 50 significa "borrar todas las filas de la tabla tablasensores en las que en la columna luminosdiad exista un valor mayor que 50". Fíjese que al indicar mayor (y no mayor o igual) una fila donde la luminosidad sea exactamente 50 no será borrada. La cláusula WHERE también puede ser aplicada cuando se hacen consultas de tipo SELECT. Por ejemplo anteriormente se usó SELECT * FROM tablasensores como consulta que devolvía todas las filas de la tabla tablasensores. Si se escribiese SELECT * FROM tablasensores WHERE luminosidad <50 se obtendría como resultado todas las filas de la tabla tablasensores donde la columna

luminosidad contiene un valor menor a 50. En una cláusula where se puede establecer una condición de igualdad (=) pero también se pueden usar otras condiciones como mayor (>), menor (<), mayor o igual (>=), menor o igual (<=), y también condiciones múltiples y condiciones más complejas, pero no se va a entrar en detalles sobre esto ahora.

Igual que se puede hacer una consulta para obtener todas las filas de una tabla, también se puede borrar todos los registros de una tabla en concreto y, para ello, sólo basta con omitir las condiciones, es decir, hacer una consulta escribiendo lo siguiente: DELETE FROM nombre_Tabla

Obviamente es peligroso hacer consultas de borrado de datos con bases de datos importantes, ya que un error a la hora de escribir la consulta puede dar lugar a la pérdida de datos.

La sentencia TRUNCATE sirve para borrar todos los registros de una tabla, al igual que se hacía con la función DELETE sin condiciones, pero tiene algunas diferencias con ésta que serán explicadas más adelante. La sintaxis a emplear es:

```
("TRUNCATE TABLE nombre_Tabla");
```

Código 7-7 Truncate

Al igual que las operaciones de tipo DELETE, esta operación es peligrosa en el sentido de que si se ejecuta erróneamente puede dar lugar a la pérdida de datos.

Las principales diferencias entre DELETE y TRUNCATE son:

- Ambas eliminan los datos, no la estructura de la tabla.
- Sólo DELETE permite la eliminación condicional de los registros (es decir, borrar sólo ciertas filas), TRUNCATE no lo permite.
- TRUNCATE es más rápida que DELETE.
- TRUNCATE reiniciará el contador para una tabla que contenga una clave autoincrementada. Si en la tabla tablasensores se tuviera un campo id autoincremental 1, 2, 3, 4, 5 … n (hasta el número de registros existentes) al hacer TRUNCATE el contador volverá a empezar en 1. En cambio DELETE mantendrá el contador de la

tabla para una clave autoincrementada. Es decir, si se borran todos los registros de una tabla que tenía un campo contador autoincremental cuyo último valor era 3257, al insertar un dato después del borrado el valor del contador será 3258 en lugar de 1.

- TRUNCATE recrea una tabla, es decir, la tabla desaparece completamente y luego es creada de nuevo, mientras que DELETE no hace que desaparezca la tabla, sólo elimina sus registros.

En este caso finalmente se ha optado por usar la operación TRUNCATE por lo que el comando queda de la siguiente manera:

```
try
{
Statement stm=cn.createStatement();
stm.executeUpdate("TRUNCATE nombre_tabla1");
stm.executeUpdate("TRUNCATE nombre_tabla2");
}
catch (SQLException ex)
{
Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE, null, ex);
}
```

Código 7-8 Borrado de tablas

De esta manera se borra todo el contenido de las tablas: nombre_tabla1 y nombre_tabla2.

7.2.7 Cerrar la conexión

Una vez que se termina de usar la conexión, se debería cerrar, o bien terminar el programa, con lo que se cierra automáticamente (véase Código 7-9).

```
conectar.close();
```

Código 7-9 Cerrar la conexión

7.3 Librería PanamaHitek_Arduino

Arduino y Java son dos entes independientes. Arduino es una plataforma de hardware libre con su propio lenguaje, mientras que Java es un lenguaje de programación de alto nivel. Java necesita para su ejecución la Máquina Virtual de Java, la cual, hasta ahora no puede ser ejecutada desde Arduino.

Sin embargo esto no significa que no se puedan crear programas en Java que le envíen instrucciones a Arduino en el que se entrega el control de circuitos electrónicos desde una interfaz en la computadora.

La comunicación serial es la forma más sencilla de comunicación que existe entre Arduino y la computadora, ya que aprovecha el puerto USB por medio de un convertidor USB-Serie.

Para poder comunicar las aplicaciones en Java con Arduino se debe aprovechar la comunicación serial, lo cual es perfectamente posible si se implementan las librerías adecuadas.

Para solucionar el problema que conlleva conectarse a Arduino sin usar su propio IDE se usará una librería que permite acceder a la comunicación serial desde cualquier entorno de desarrollo (IDE) para Java, en este caso Netbeans 8.0.2

Se usará la nueva versión de la librería PanamaHitek_Arduino, anteriormente conocida como librería Arduino para Java. Esta librería se puede descargar desde el siguiente enlace: http://sourceforge.net/projects/arduinoyjava/files/v2.7.0/

Antes de esta versión era necesario tener instalados los drivers rxtxSerial.dll en la ruta de JAVA_HOME. A partir de la versión 2.7.0, en cada ejecución la librería verifica si los drivers están instalados en la ruta C:/JavaRXTX.

Si no existen dichos ficheros, la librería los crea. Si el directorio no existe, se encarga de armar la estructura para el almacenamiento de los archivos *.dll (Windows) necesarios para que el programa cargue sin problemas.

La librería incluye dos clases: la clase PanamaHitek_Arduino y la clase PanamaHitek_MultiMessage. La clase PanamaHitek_Arduino es la encargada de manejar todas las conexiones y la comunicación con Arduino. La clase PanamaHitek_MultiMessage incluye las herramientas necesarias para recibir múltiples mensajes de forma simultánea en Java. También se incluyen las clases que pertenecen a la librería RXTX, sobre la cual está construida la librería PanamaHitek_Arduino.

7.3.1 Principales métodos de la librería PanamHitek_Arduino

Los principales métodos de esta librería se describen a continuación:

ArduinoRX(String COM#, int time_out, int baud_rate, SerialPortEventListener evento)

Este método se utiliza para iniciar la conexión de Java con Arduino solamente para la recepción de datos. En el nombre de puerto se coloca el COM#, o sea el puerto COM donde esté conectado Arduino, el time out es el tiempo de espera (normalmente se usa 2000), el baud rate debe ser el mismo que se usa en Arduino IDE (generalmente 9600) y el Serial PortEventListener debe ser una variable declarada antes de utilizar este método.

ArduinoTX(String COM#, int time_out, int baud_rate)

Este método se utiliza para iniciar la conexión de Java con Arduino solamente para la transmisión de datos.

ArduinoRXTX(StringCOM# ,int time_out,int baud_rate,SerialPortEventListener evento)

Este método se utiliza para iniciar la conexión de Java con Arduino para la transmisión y recepción de datos.

sendData(String data)

Método utilizado para enviar datos a Arduino. Los datos se deben enviar como cadena de texto (String). En este caso, en el que se envia datos de tipo entero, será necesario usar la función (char).

receiveData()

Devuelve un dato recibido a través del puerto serie. Este dato será numérico en formato ASCII por lo que se debe traducir de carácter a decimal.

MessageAvailable()

Devuelve un valor boolean que indica si hay algún mensaje disponible para imprimir. Dicho mensaje DEBE ser enviado desde Arduino utilizando Serial.println()

printMessage()

Devuelve una cadena de caracteres que contiene el mensaje que ha sido enviado desde Arduino, pero traducido a caracteres. Se debe utilizar dentro de una estructura condicional utilizando MessageAvailable(). Cuando haya un mensaje disponible, se imprime utilizando este método.

killConnection()

Permite finalizar la conexión entre Arduino y el PC sin tener que finalizar la aplicación que se está ejecutando.

7.3.2 Posibles errores en esta librería

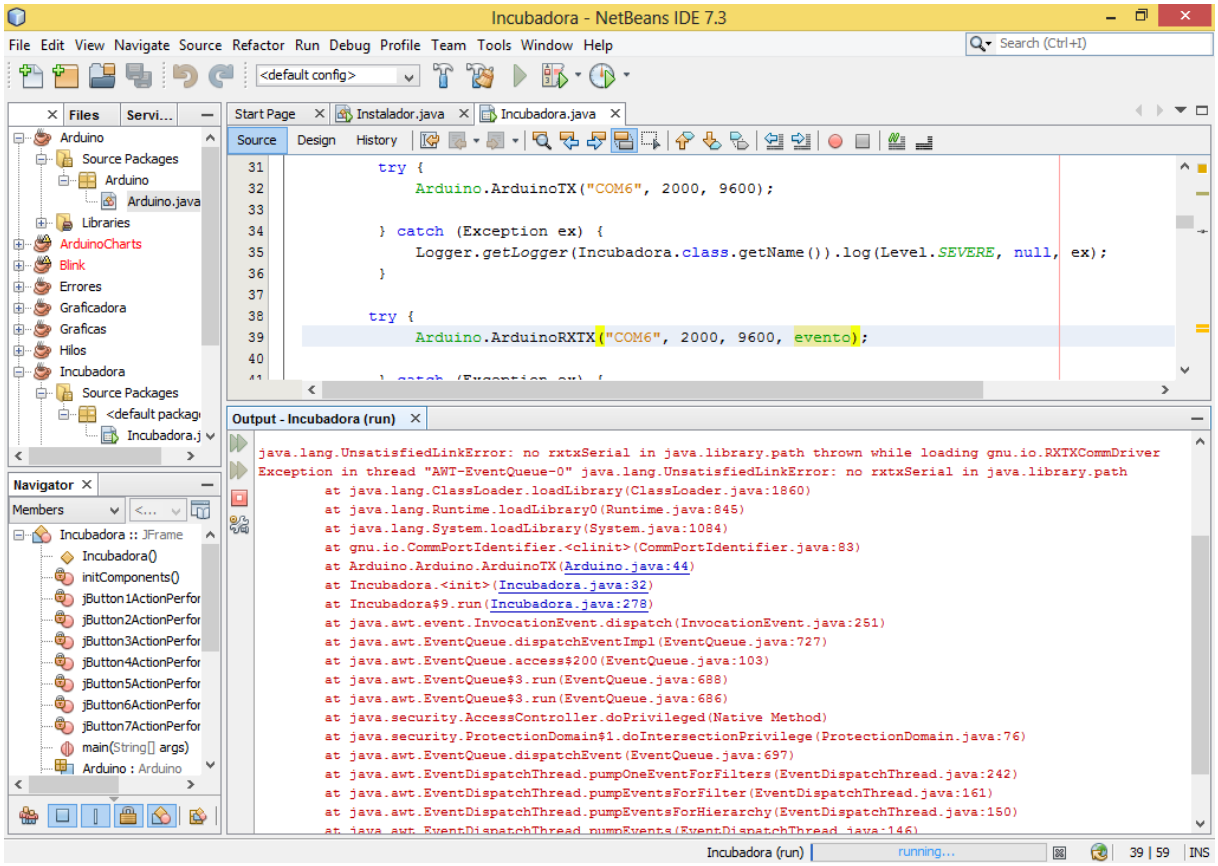


Figura 7-1 Ventana que muestra el error

Uno de los errores más comunes es el siguiente: *java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver*

Este error se produce cuando se intenta iniciar una aplicación en Java que se va a comunicar con Arduino. La causa de esto es que no se ha instalado correctamente la librería RXTX la cual necesita 2 drivers para trabajar: rxtxSerial.dll y rxtxParallel.dll

De hecho, el driver Parallel no será necesario ya que se está usando comunicación serial, no por puerto paralelo.

Otro error muy común es que la aplicación java lance la siguiente ventana:

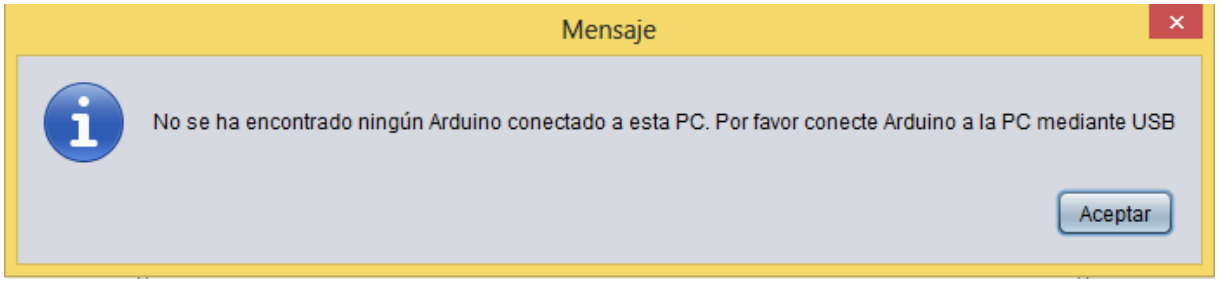


Figura 7-2 Ventana de alerta

Se produce cuando se trata de iniciar un programa sin tener conectado Arduino al PC. También suele pasar que aún estando conectado el PC no lo reconoce. Desconectando y volviendo a conectar el Arduino se soluciona este problema. Si el monitor serial de Arduino IDE está activado tampoco se podrá iniciar las aplicaciones en Java.

Otra ventana que puede aparecer es la siguiente:

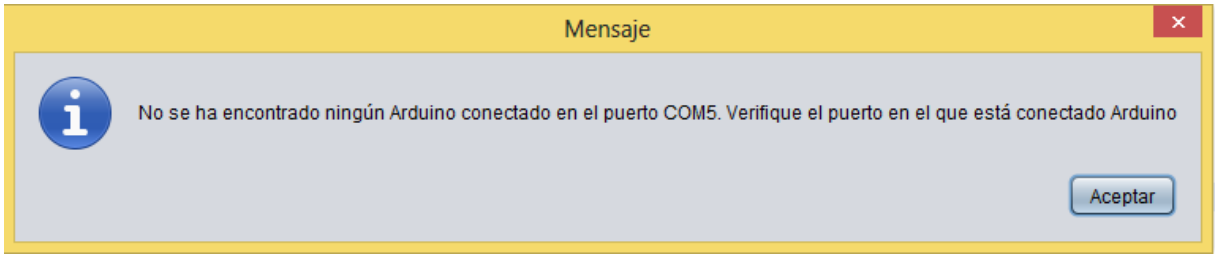


Figura 7-3 Ventana de alerta

Este error se produce cuando se tiene el Arduino conectado pero el puerto COM es el equivocado.

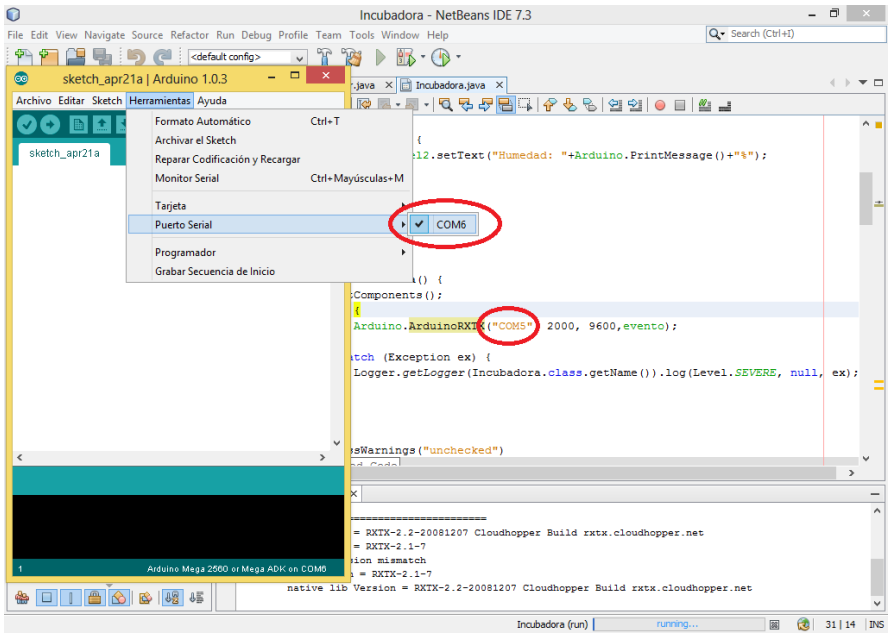


Figura 7-4 Origen del problema

Arduino IDE indica que la placa que usada está conectada en el COM6 pero en Java se ha indicado que se usará el COM5. Debido a esto no es posible iniciar la aplicación. El COM que aparece en Arduino IDE debe ser igual al colocado en Java.

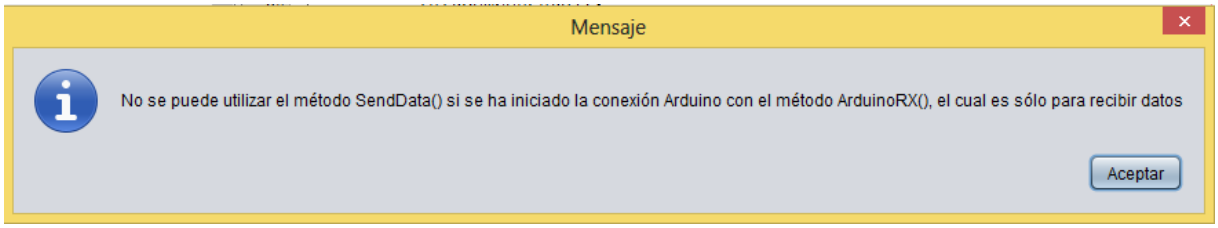


Figura 7-5 Ventana de alerta

Este error se produce cuando se trata de enviar datos tras haber utilizado como conexión el método ArduinoRX que solamente permite recibir datos. Si desea transmitir y recibir datos a la vez debe usar el método ArduinoRXTX.

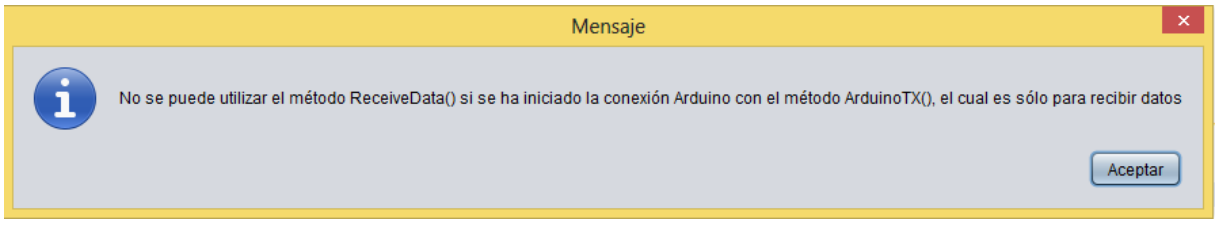


Figura 7-6 Ventana de alerta

Este error se produce cuando se trata de recibir datos y se ha iniciado la conexión con Arduino utilizando el método ArduinoTX que solamente permite transmitir datos. Si desea transmitir y recibir datos a la vez debe usar el método ArduinoRXTX.

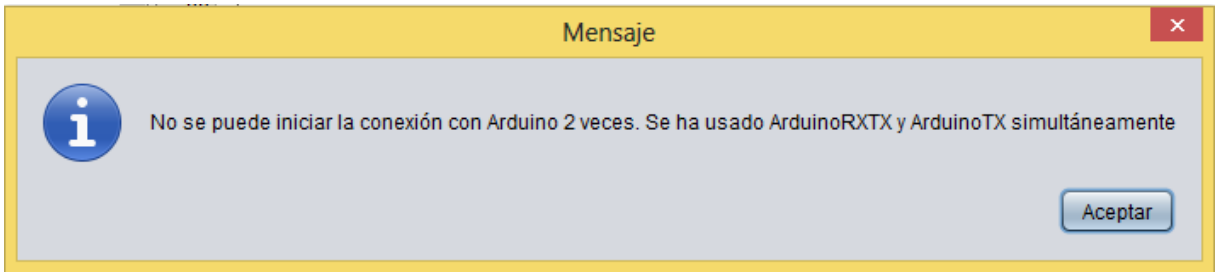


Figura 7-7 Ventana de alerta

Este error se produce cuando se inicializa una conexión con Arduino en 2 partes diferentes del código. Esta librería solamente permite iniciar la conexión una sola vez, ya sea utilizando el método ArduinoRX, ArduinoTX o ArduinoRXTX.

7.4 Librería POI

En este apartado se describe el uso de la librería POI. El primer paso es descargar los ficheros jar correspondientes a la última versión desde la web de Apache POI http://poi.apache.org/

Es importante tener en cuenta que cada archivo de Excel representa un LIBRO, dentro de cada libro existen HOJAS, dentro de cada HOJA existen FILAS, y, finalmente, en cada FILA existen CELDAS. Se hace mención de esto porque ayudará a ver cómo se organiza la información en el archivo.

Primero es necesario crear un LIBRO haciendo uso de la clase HSSFWorkbook.

```
HSSFWorkbook objWB = new HSSFWorkbook();
```

Código 7-10 Creación Libro

A continuación se crea la hoja con la clase HSSFSheet.

```
HSSFSheet hoja1 = objWB.createSheet("hoja 1");
```

Código 7-11 Creación Hoja

Posteriormente se crea la fila con HSSFRow. Notese que el valor que se envía al método encargado de crear las filas es de tipo short, el mismo que indica el número correspondiente a la fila que se ha de trabajar. El índice de las filas empieza en “0”, aunque ello no impide trabajar directamente con otras filas (véase Código 7-12).

```
HSSFRow fila = hoja1.createRow((short)1);
```

Código 7-12 Creación Fila

Una vez creada la fila, se empieza a trabajar con las celdas. Como es apreciable en el código se tiene la posibilidad de establecer estilos mediante las clases HSSFFont y HSSFCellStyle.

La primera, permite establecer el tipo de fuente que se empleará para la celda utilizada. Para ello se cuenta con los métodos setPointHeightInPoints que recibe un valor de tipo short que representa el tamaño de la fuente; el método setFontName el mismo que recibe una constante de la misma clase que permite establecer la fuente que se ha de emplear, y, otros métodos como: setBoldweight y setUnderline, entre otros, que permitirán aplicarle otros estilos y efectos al valor que ocupe dicha celda.

La segunda, es la clase que, finalmente, ayudará a aplicar el estilo a la celda. Se puede acomodar y alinear el texto mediante los métodos setWrapText, setAlignment y setVerticalAlignment; aplicar la fuente trabajada, con el método setFont; configurar los bordes mediante los métodos: setBorderBottom, setBorderLeft, setBorderRight y setBorderTop, para el tipo; y, setBottomBorderColor, setLeftBorderColor, setRightBorderColor y setTopBorderColor para establecer el color de los bordes; y, establecer el sombreado de las celdas mediante los métodos setFillForegroundColor, setFillBackgroundColor y setFillPattern (véase Código 7-13).

```
// Primero, se establece el tipo de fuente
HSSFFont fuente = objLibro.createFont();
fuente.setFontHeightInPoints((short)11);
fuente.setFontName(fuente.FONT_ARIAL);
fuente.setBoldweight(HSSFFont.BOLDWEIGHT_BOLD);

// Luego se crea el objeto que se encargará de aplicar el estilo a la celda
HSSFCellStyle estiloCelda = objLibro.createCellStyle();
estiloCelda.setWrapText(true);
estiloCelda.setAlignment(HSSFCellStyle. ALIGN_JUSTIFY);
estiloCelda.setVerticalAlignment(HSSFCellStyle.VERTICAL_TOP);
estiloCelda.setFont(fuente);

// Se establecen los bordes
estiloCelda.setBorderBottom(HSSFCellStyle.BORDER_MEDIUM);
estiloCelda.setBottomBorderColor((short)8);
estiloCelda.setBorderLeft(HSSFCellStyle.BORDER_MEDIUM);
estiloCelda.setLeftBorderColor((short)8);
estiloCelda.setBorderRight(HSSFCellStyle.BORDER_MEDIUM);
estiloCelda.setRightBorderColor((short)8);
estiloCelda.setBorderTop(HSSFCellStyle.BORDER_MEDIUM);
estiloCelda.setTopBorderColor((short)8);

// Se establece el tipo de sombreado de la celda
estiloCelda.setFillForegroundColor((short)22);
```

```
estiloCelda.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);

// Se crea la celda, se aplica el estilo y se define
// el tipo de dato que contendrá la celda
HSSFCell celda = objFila.createCell((short)0);
celda.setCellStyle(estiloCelda);
celda.setCellType(HSSFCell.CELL_TYPE_STRING);

// Finalmente, se establece el valor

     celda.setCellValue("Un valor");
```

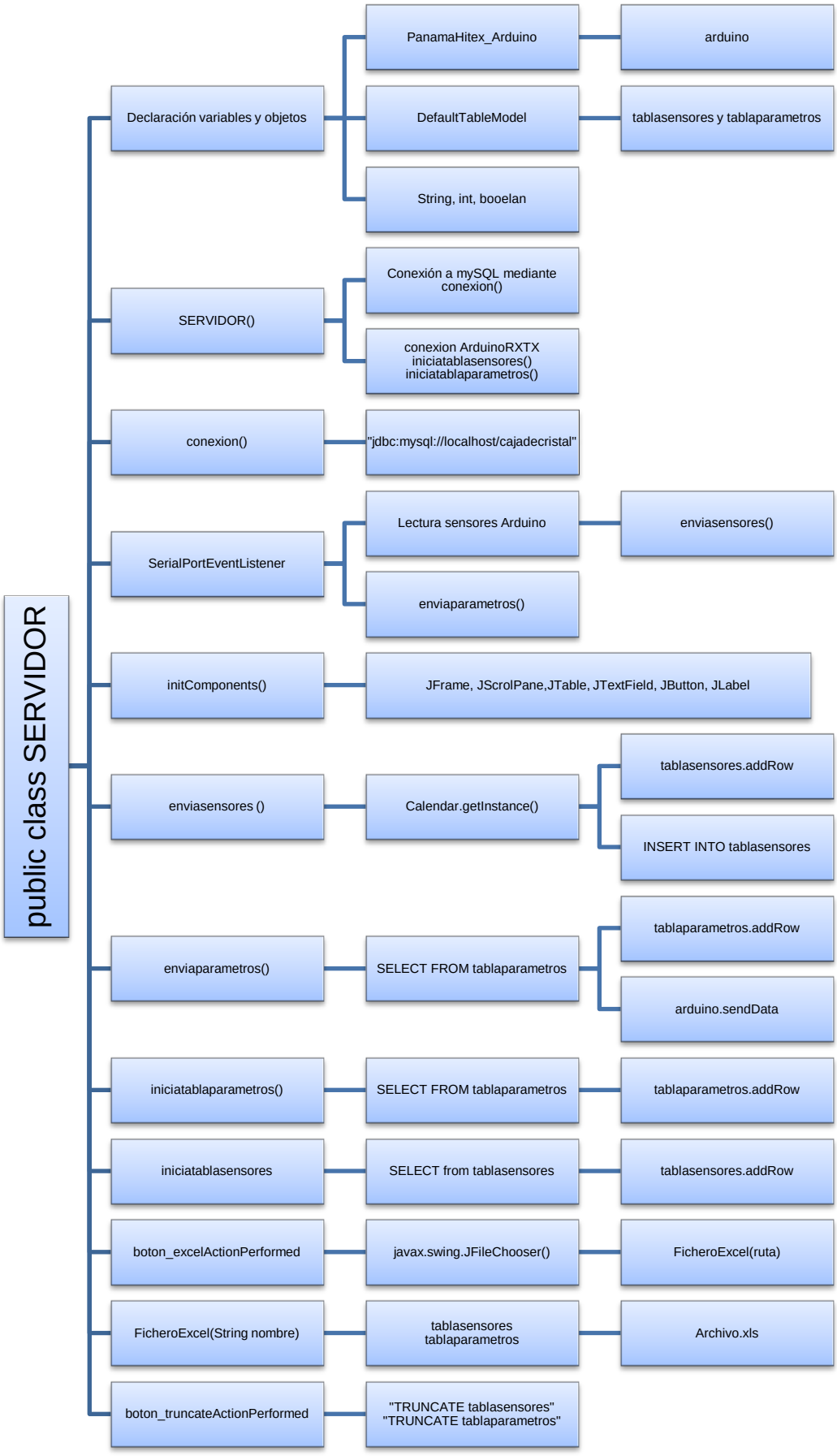
Código 7-13 Creación Celda

Finalmente, se vuelca el libro a un archivo de la siguiente forma:

```
String strNombreArchivo = "C:/libro1.xls";
File objFile = new File(strNombreArchivo);
FileOutputStream archivoSalida = new FileOutputStream(objFile);
objWB.write(archivoSalida);
archivoSalida.close();
```

Código 7-14 Creación archivo.xls

7.5 Código de programación servidor.jar



A continuación se procede al análisis de todas las funciones programadas y de como interactuan entre ellas. Para consultar el código programado acudir al Anexo.

7.5.1 Declaración de variables y objetos

En esta parte inicial del código se realiza tanto la declaración como la inicialización de las variables y los objetos más usados a lo largo de la aplicación Java. En primer lugar se produce la creacción y llamada al constructor para crear un objeto de tipo PanamaHitek_Arduino. A partir de ahora se usará arduino para referenciar a este objeto.

Posteriormente se realiza la declaración de las tablas, en este caso solo es necesario usar dos: tablasensores y tablaparametros. El DefaultTableModel es una clase que implementa TableModel que contiene todos los métodos necesarios para modificar datos en su interior, añadir filas o columnas y darle a cada columna el nombre que se desee. Para utilizar DefaultTableModel es necesario importarla (import javax.swing.JTable , import javax.swing.table.DefaultTableModel) y luego declararla para posteriormente poder usar la clase JTable.

A continuación se declaran todas las variables que están incluidas en la base de datos. En este caso existen variables de tipo String, int y boolean.

7.5.2 public SERVIDOR() throws SQLException

Se cargan todos los componentes de la interfaz gráfica mediante la llamada initComponents(). Posteriormente se enlanzan las tablas virtuales (DefaultTableModel) con las tablas visuales (JTable) que aparecerán en la ventana gráfica (JFrame). Se asignan los identificadores a ambas tablas.

Por otra parte se realiza la comunicación con Arduino tanto de escritura como de lectura (RXTX).

Finalmente se llama a las funciones iniciatablasensores() e iniciatablaparametros() que se encargan de volcar en el programa Java todos los registros existentes en la base de datos de mySQL. Es decir mediante estas funciones se incorporan todos los registros históricos (grabados anterioremente a la ejecución del servidor) a las tablas.

7.5.3 public Connection conexion () throws SQLException

Esta función se encarga de realizar la conexión con la base de datos mySQL. Esta función devuelve una variable de tipo Connection. Las variables de conexión son un objeto del tipo "Connection" (paquete java.sql). La implementación de la interfaz "Connection" administra la conexión entre el programa y la base de datos. Los objetos "Connection" permiten a los programas crear instrucciones de SQL para manipular bases de datos. El programa inicializa a conexión con el resultado de una llamada al método "static getConnection" de la clase "DriverManager" (paquete java.sql), el cual trata de conectarse a la base de datos especificada mediante su URL. El método "getConnection" recibe tres argumentos: un objeto String que especifica el URL de la base de datos, un objeto String que especifica el nombre de usuario y un objeto String que especifica la contraseña.En resumen son aquellas que permiten realizar la conexión con la base de datos para posteriormente poder enviar instrucciones o sentencias SQL.

Para realizar la conexión hay que proporcionar la URL donde se encuentra dicha base de datos. En este caso esta aplicación java se ejecutará en el PC que hará la función de servidor y por tanto será en este mismo PC donde se encuentre almacenada dicha base de datos. La URL será la siguiente: `"jdbc:mysql://localhost/cajadecristal:4001".`

7.5.4 public void serialEvent

Esta función se esta ejecutando cíclicamente. Se trata de un listener que esta constantemente a la espera de que haya datos disponibles para su envío desde Arduino al PC. Para ello se recurre a arduino.MessageAvailable() que devuelve un true cada vez que Arduino envia los datos de los sensores a través de la comunicación serial del PC.

Estos datos son almacenados en un primer lugar en un Array de nombre DatosSensores mediante el comando arduino.PrintMessage(). Posteriormente se asignan cada uno de los valores de los sensores a las variable declaradas para ello. En este caso hinv, tinv, hsuelo, hamb, tamb, lumi, o2 y co2. En el caso de las variables h2o, estadobomba, estadoventilador y estadoluz en un primer momento son leídas como variables de tipo int. Como lo que se necesitan son variales de tipo boolean es necesario realizar una comparación de tipo if para así poder crear las variables h2oboolean, estadobombaboolean, estadoventiladorboolean y estadoluzboolean.

Una vez realizada la lectura, asignación y tratamiento de todas las variables de los sensores, se ejecuta la función enviasensores() que añadirá una nueva fila tanto al DefaultTableModel tablasensores como a la base de datos en mySQL.

Desde aquí también se realiza el envío de los parámetros de funcionamiento. Para ello se llama a la función enviaparametros().

7.5.5 private void initComponents()

Para conseguir el aspecto de la interfaz gráfica se ejecuta el siguiente código, asociado a la parte Design. Este código se autogenera gracias al IDE que permite programar el entorno gráfico mediante una herramienta de gran comodidad.

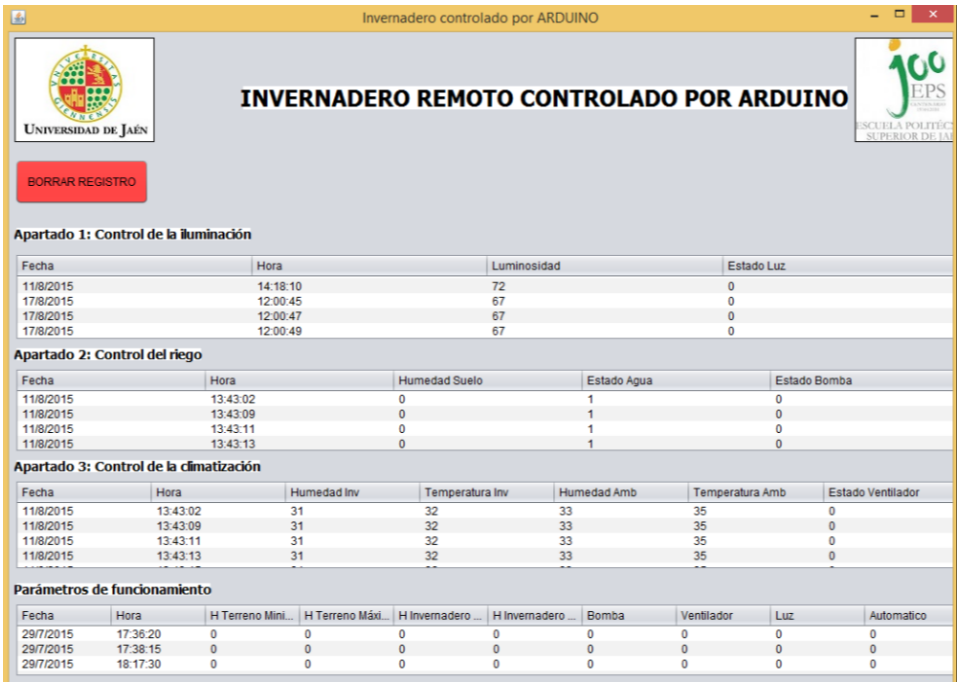


Figura 7-8 Aspecto de la Interfaz Gráfica

7.5.6 public void enviasensores()

Una vez recibidos todos los datos de los sensores mediante el listener hay que almacenarlos tanto en tablasensores como en mySQL. Como las tablas llevan asociadas una columna en la que se graba la fecha y hora de registro, lo primero es obtener la fecha y hora ctual. Para ello se recurre a Calendar.getInstance(), posteriormente se realiza el tratamiento de las variables para que la fecha y hora aparezca en el formato deseado. En este caso sería 02/05/2015 18:25:23. El único problema encontrado ha sido que en Java para el primer mes,

es decir Enero se le asigna un valor 0. Para solucionar el problema se ha convertido el dato a entero para poder sumarle una unidad y posteriormente convertirlo a String que es su tipo de dato original.

7.5.7 public void enviaparametros();

Esta parte del programa se centra en la tablaparametros. En este caso los datos son obtenidos de la base de datos mySQL que es el mecanismo vinculante entre cliente y servidor. Se comprueba el último registro guardado en tablaparametros(mySQL), para ello se ejecuta rset.last() que permite situar el cursor en la última fila registrada. Se asignan los datos a las variables cuyo nombre comienzan con new: newhsuelomin, newhsuelomax, newhinvmin, newhinvmax, newbomba, newventilador, newluz, newautomatico. A continuación se comprueba si alguna de las nuevas variables cambia con respecto a los actuales parámetros de funcionamiento del controlador. En caso positivo de que el cliente desde EJS haya modificado algún parámetro, se actualiza el funcionamiento del controlador y se añade a la tablaparametros dichos valores.

La comunicación Servidor-Arduino se realiza mediante el envío de una cadena de caracteres. En el caso de variables de tipo int, se envia el carácter equivalente ascii. Es decir si hay que mandar el valor 122 lo que se mandaría sería el carácter ‘z’. Para variables de tipo boolean , simplemente se envían uno de los dos caracteres: ‘1’ o ‘0’. No confundir el carácter ‘1’ con el número 1. A continuación se muestra la lista de caracteres ASCII.

33	!	54	6	75	K	96	`	117	u	138	Š	159	Ÿ	180	´	201	É	222	Þ	243	ó
34	"	55	7	76	L	97	a	118	v	139	‹	160		181	µ	202	Ê	223	ß	244	ô
35	#	56	8	77	M	98	b	119	w	140	Œ	161	¡	182	¶	203	Ë	224	à	245	õ
36	$	57	9	78	N	99	c	120	x	141	□	162	¢	183	·	204	Ì	225	á	246	ö
37	%	58	:	79	O	100	d	121	y	142	□	163	£	184	¸	205	Í	226	â	247	÷
38	&	59	;	80	P	101	e	122	z	143	□	164	¤	185	¹	206	Î	227	ã	248	ø
39	'	60	<	81	Q	102	f	123	{	144	□	165	¥	186	º	207	Ï	228	ä	249	ù
40	(	61	=	82	R	103	g	124	|	145	‘	166	¦	187	»	208	Ð	229	å	250	ú
41	)	62	>	83	S	104	h	125	}	146	’	167	§	188	¼	209	Ñ	230	æ	251	û
42	*	63	?	84	T	105	i	126	~	147	“	168	¨	189	½	210	Ò	231	ç	252	ü
43	+	64	@	85	U	106	j	127	□	148	”	169	©	190	¾	211	Ó	232	è	253	ý
44	,	65	A	86	V	107	k	128	€	149	•	170	ª	191	¿	212	Ô	233	é	254	þ
45	-	66	B	87	W	108	l	129	□	150	–	171	«	192	À	213	Õ	234	ê	255	ÿ
46	.	67	C	88	X	109	m	130	‚	151	—	172	¬	193	Á	214	Ö	235	ë	256	
47	/	68	D	89	Y	110	n	131	ƒ	152	˜	173	-	194	Â	215	×	236	ì		
48	0	69	E	90	Z	111	o	132	„	153	™	174	®	195	Ã	216	Ø	237	í		
49	1	70	F	91	[	112	p	133	…	154	š	175	¯	196	Ä	217	Ù	238	î		
50	2	71	G	92	\	113	q	134	†	155	›	176	°	197	Å	218	Ú	239	ï		
51	3	72	H	93	]	114	r	135	‡	156	œ	177	±	198	Æ	219	Û	240	ð		
52	4	73	I	94	^	115	s	136	ˆ	157	□	178	²	199	Ç	220	Ü	241	ñ		
53	5	74	J	95	_	116	t	137	‰	158	□	179	³	200	È	221	Ý	242	ò		

Figura 7-9 Lista de caracteres ASCII

Una vez formada la cadena de caracteres que contiene toda la información de los parámetros codificada en ascii, se ejecuta arduino.sendData(cadena). Posteriormente en el código programado para Arduino habrá que realizar el proceso justo al contrario. Se reciben caracteres y se convierten a valores numéricos.

7.5.8 public void iniciatablaparametros();

Esta función se ejecuta una solo vez, cada vez que se inicia el servidor. Se ocupa de volcar toda la información existente en mySQL dentro de la tabla tablaparametros. Es decir, actualiza la aplicación java con todos los datos existentes en la base de datos. A partir de este momento ya se pueden añadir nuevos parámetros.

Para ello se inicia la conexión con mySQL mediante la sentencia "SELECT FROM tablaparametros" .Una vez dentro de la base de datos se ejecuta una sentencia cíclica de tipo while. De esta manera se pasa por todos los registros existentes dentro de mySQL. Esto se consigue gracias a rset.next() que devuelve true siempre que existen más filas por leer. Una vez el cursor está situado en la última fila, rset.next() devolverá false.

Una vez finalizada la acción de esta función, se encuentran los mismos registros tanto en mySQL como en el DefaultTableModel existente dentro del servidor.

7.5.9 public void iniciatablasensores();

Esta función se ejecuta una sola vez, cada vez que se inicia el servidor. Se ocupa de volcar toda la información existente en mySQL dentro de la tabla tablasensores. Es decir, actualiza la aplicación java con todos los datos existentes en la base de datos. A partir de este momento ya se pueden añadir nuevos datos recibidos de los sensores.

Para ello inicia la conexión con mySQL mediante la sentencia "SELECT FROM tablasensores" .Una vez dentro de la base de datos se ejecuta una sentencia cíclica de tipo while. De esta manera se pasan por todos los registros existentes dentro de mySQL. Esto se consigue gracias a rset.next() que devuelve true siempre que existen más filas por leer. Una vez el cursor esta situado en la última fila, rset.next() devolverá false.

Una vez finalizada la acción de esta función, se tendrán los mismos registros tanto en mySQL como en el DefaultTableModel existente dentro del servidor.

7.5.10 private void boton_excelActionPerformed

Se trata del código asociado al boton_excel (JButton). Un botón es un componente en el que el usuario hace clic para desencadenar cierta acción. Una aplicación de Java puede utilizar varios tipos de botones, incluyendo botones de comando, casillas de verificación, botones interruptores y botones de opción.

Todos los tipos de botones son subclases de AbstractButton (paquete javax.swing), la cual declara las características comunes para los botones de Swing.

Un botón de comando genera un evento ActionEvent cuando el usuario hace clic en él. Los botones de comando se crean con la clase JButton. El texto de la cara de un objeto JButton se llama etiqueta del botón. Una GUI puede tener muchos objetos JButton, pero cada etiqueta de botón debe generalmente ser única en las partes de la GUI en que se muestre.

Figura 7-10 Botón Excel

Al pulsar sobre el botón de excel se ejecuta este código que se encarga de lanzar una ventana para así poder seleccionar la ruta donde se desea guardar el fichero. Posteriormente se llama a FicheroExcel(ruta) que es la función que se encarga de volcar toda la información de las DefaultTableModel en un fichero excel.

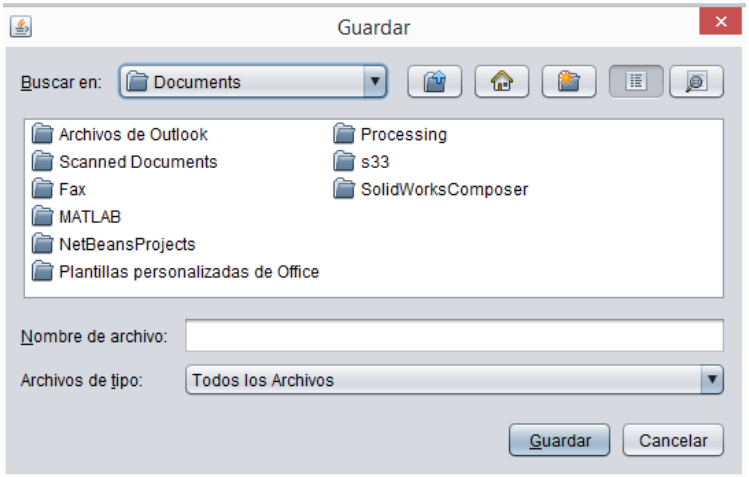


Figura 7-11 Ventana para seleccionar la ruta

7.5.11 public void FicheroExcel(String nombre)

Esta función se encarga de generar un archivo de tipo nombre.xls , tiene un parámetro String de entrada que contiene la ruta del archivo que se desea generar. El objetivo de esta función es conseguir que el servidor pueda proporcionar toda la información generada en una hoja excel. De esta manera se podrá tratar los datos de los sensores obtenidos con cualquier otro programa informático.

Para ello se crea un libro (Workbook) que va a contener dos hojas (Sheet) con el nombre de sensores y parámetros. Posteriormente se identifica todas las columnas añadiendo su nombre.

Para realizar todo el volcado de la información es necesario recorrer toda la tabla (DefaultTableModel) mediante dos sentencias enlazadas de tipo for. Una se encargará de ir pasando por todas las filas y la otra irá recorriendo las diferentes columnas.

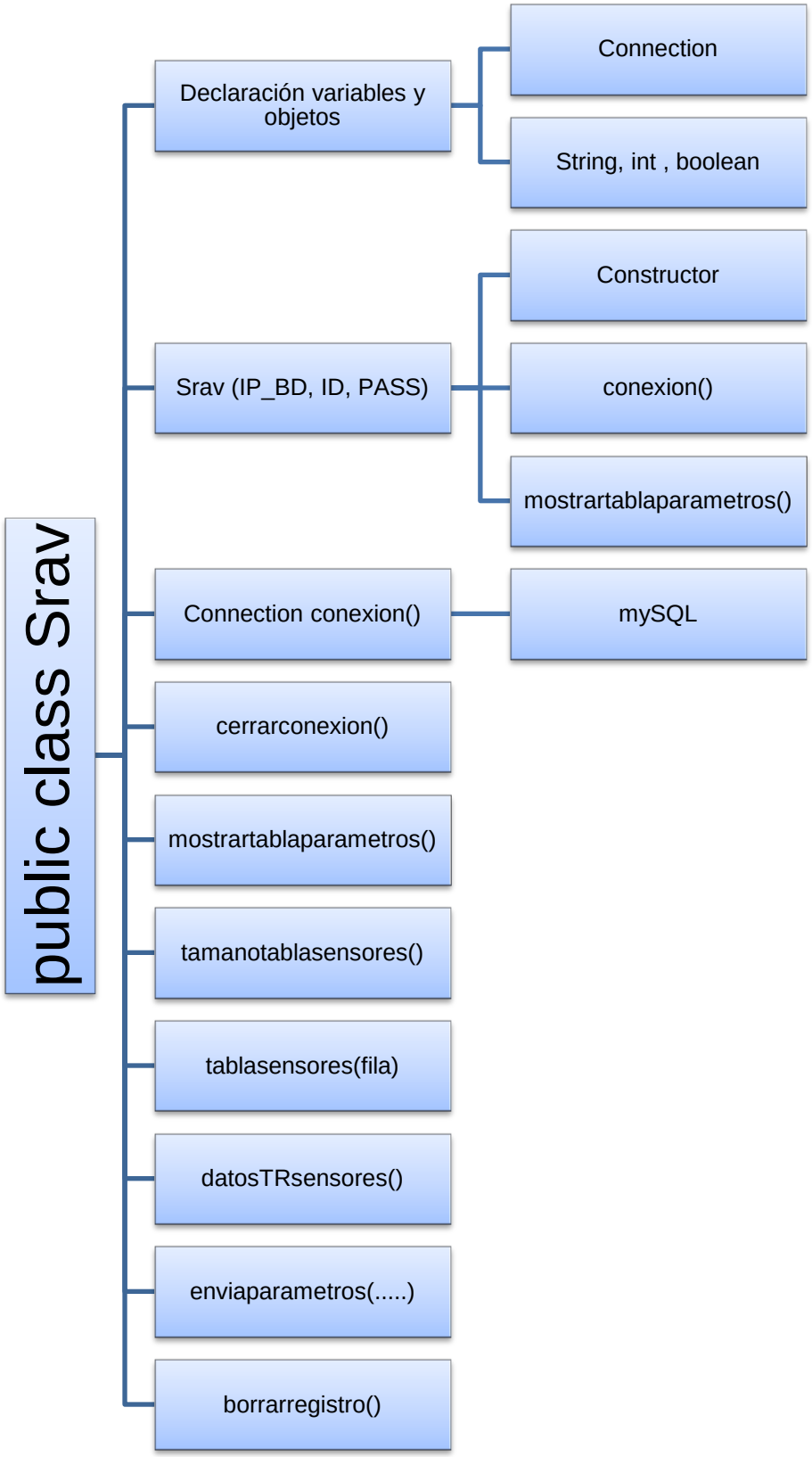
Después de haber generado cada hoja de datos, se volverá a recorrer para adjuntar la celda al contenido mediante el comando: autoSizeColumn(j). Por último se guarda y se cierra el fichero generado. Se podrá comprobar como se ha generado el archivo deseado en la ruta seleccionada.

7.5.12 private void boton_truncateActionPerformed

Al pulsar el botón BORRAR REGISTRO se borra toda la información contenida en la base de datos de mySQL dentro de la tablasensores.

Figura 7-12 Botón BORRAR REGISTRO

7.6 Código de programación srav.jar



7.6.1 Declaración de variables y objetos

Lo primero de todo en cualquier programa escrito en Java es declarar e inicializar todas las variables relevantes en el código. En este caso se comienza con la declaración de la variable de tipo Connection a la que se nombra cn. Las variables de conexión son un objeto del tipo "Connection" (paquete java.sql). La implementación de la interfaz "Connection" administra la conexión entre el programa y la base de datos. Los objetos "Connection" permiten a los programas crear instrucciones de SQL para manipular bases de datos. En resumen son aquellas que te permiten realizar la conexión con la base de datos para posteriormente poder enviar instrucciones o sentencias SQL.

A continuación se realiza la declaración todas las variables relacionadas con los parámetros de funcionamiento del microcontrolador. Son un total de 4 parámetros de tipo int y otros 4 de tipo boolean.

Con respecto a las variables relacionadas con los datos proporcionados por los sensores hay que diferenciar entre las variables usadas para recorrer el histórico y las variables que proporcionan el dato en tiempo real.

7.6.2 public Srav (String IP_BD, String ID, String PASS)

En este fragmento se encuentran el constructor de la clase Srav. Los constructores de una clase son fragmentos de código que sirven para inicializar un objeto a un estado determinado. Una clase puede carecer de constructor, pero esto no es lo más habitual. En este caso se ha optado por realizar este tipo de constructor con tres parámetros IP_BD, ID, PASS.

Dentro del constructor se llama a la función propia conexión y a mostrartablaparametros.

7.6.3 public Connection conexion(String IP_bd, String ID_usuario, String pass)

Este método ha sido creado con la única función de realizar la conexión de manera remota (desde el pc cliente al servidor) a la base de datos SQL. El programa inicializa la conexión con el resultado de una llamada al método "static getConnection" de la clase "DriverManager" (paquete java.sql), el cual trata de conectarse a la base de datos especificada mediante su URL. El metodo "getConnection" recibe tres argumentos: un objeto String que

especifica el URL de la base de datos, un objeto String que especifica el nombre de usuario y un objeto String que especifica la contraseña. En este caso hay que recordar que al ser la conexión de tipo remoto habrá que especificar la IP y el puerto, ej:192.138.1.33:4001

7.6.4 public void cerrarconexion()

Esta función simplemente cierra la conexión SQL si es ejecutada.

7.6.5 public void mostrartablaparametros()

Esta función es ejecutada dentro del constructor con el único objetivo de actualizar el valor de los parámetros de Arduino una vez ejecutado el cliente. Es decir lee el último valor registrado en tablaparametros dentro de mySQL.

Para ello se realiza la conexión a la tabla (“SELECT * FROM tablaparametros) donde habrá que desplazarse al último registro mediante el comando rs.last().

7.6.6 public int tamanotablasensores()

El único objetivo de esta función es devolver un dato de tipo entero que contiene el número de registros existente dentro de la tablasensores. Este dato será necesario posteriormente para poder graficar en EJS los valores históricos de los sensores.

Para ello se recorre toda la tabla mediante rs.next y se va incrementando el valor a devolver (filas).

7.6.7 public void tablasensores(int fila)

Esta función se encarga de refrescar los valores a todas las variables relacionados con la lectura de sensores. Tiene un parámetro de entrada que es la fila en la que se posicionará dentro de tablasensores. Gracias al comando rs.absolute(fila) se podrá desplazarse al registro deseado sin ningún problema.

Esta función será posteriormente ejecutada por EJS dentro de un bucle para graficar todo el histórico.

7.6.8 public void datosTRsensores()

Esta función es ejecutada desde EJS para obtener los datos proporcionados por los sensores instalados en tiempo real. Para llevar a cabo este objetivo se realiza la conexión a la tablasensores dentro de mySQL(SELECT * FROM tablasensores). Se desplaza el cursor a la última fila registrada mediante el comando rs.last(). En un primer momento se lee solo la fecha y hora de ese registro y se almacena en las variables de tipo String nFechas y nHoras. Por otro lado se realiza una comparación de estas dos variables con las variables de tipo String Fechas y Horas que contienen la fecha y hora de el último registro graficado en EJS. Para comparar dos valores de tipo String se usa el comando nFechas.Matches(Fechas) si estos dos valores no coinciden significa que el tiempo ha avanzado y que por lo tanto se está leyendo datos en tiempo real. Una vez comprobado esto se realiza la lectura de todos los sensores.

7.6.9 public void enviaparametros(int hsuelomin,int hsuelomax, int hinvmin, int hinvmax, boolean bomba, boolean ventilador, boolean luz, boolean automatico)

Esta función es llamada por EJS con la finalidad de cambiar uno o varios parámetros de funcionamiento en el controlador. Los parámetros controlables son los siguientes:

- int hsuelomin= valor mínimo para el intervalo de control automático de la humedad del suelo.
- int hsuelomax= valor máximo para el intervalo de control automático de la humedad del suelo.
- int hinvmin= valor mínimo para el intervalo de control automático de la humedad del invernadero.
- int hinvmax= valor máximo para el intervalo de control automático de la humedad del invernadero.
- boolean bomba= valor que controla el encendido manual de la bomba.
- boolean ventilador= valor que controla el encendido manual del ventilador.
- boolean luz= valor que controla el encendido manual de la luz.

- boolean automatico= valor que determina el tipo de funcionamiento del invernadero (manual o automático).

Lo primero es obtener la fecha y hora actual. Para ello se recurre a Calendar.getInstance(), posteriormente se realizará el tratamiento de las variables para que la fecha y hora aparezca en el formato deseado. En este caso sería 02/05/2015 18:25:23. El único problema que ha ocurrido ha sido que en Java para el primer mes, es decir Enero se le asigna un valor 0. Para solucionar el problema se ha convertido el dato a entero para poder sumarle una unidad y posteriormente convertirlo a String que es su tipo de dato original.

Una vez llamada la función desde EJS que se encarga de proporcionarle todos los parámetros, solo queda grabar esta infomación en mySQL (tablaparámetros).

7.6.10 public void borrarregistro()

Esta función es muy simple, es llamada desde EJS para vaciar la tabla llamada tablasensores de la base de datos que se muestra en el servidor. De esta manera, se consigue evitar una acumulación de datos en el servidor que pudiera ralentizar el funcionamiento. El código es muy sencillo, simplemente se ejecuta la opción TRUNCATE ya comentada anteriormente.

8 DESARROLLO DEL SOTWARE DE LA INTERFAZ EN EJS

8.1 Instalación del software EJS (Easy Java Simulations)

Easy Java Simulations es una herramienta de software diseñada para la creación de simulaciones discretas por computador. Una simulación discreta por computador, o simplemente una simulación por computador, es un programa de computador que intenta reproducir, con fines pedagógicos o científicos, un fenómeno natural a través de la visualización de los diferentes estados que éste puede presentar. Cada uno de estos estados está descrito por un conjunto de variables que cambia en el tiempo debido a la iteración de un cierto algoritmo.

Todo esto significa que EJS es un programa que ayuda a crear otros programas; más precisamente, simulaciones científicas. Ha sido diseñado para permitir a sus usuarios trabajar a un alto nivel conceptual, usando un conjunto de herramientas simplificadas y concentrando la mayoría de su tiempo en los aspectos científicos de la simulación, y pidiendo al computador que realice automáticamente todas las otras tareas necesarias pero fácilmente automatizables.

En particular, EJS crea aplicaciones Java que son independientes y multiplataforma, o applets que se pueden visualizar usando cualquier navegador Web (y por tanto ser distribuidos a través de Internet), que pueden leer datos a través de la red y ser controlados usando scripts (conjuntos de instrucciones) incluidos en las páginas HTML.

Easy Java Simulations ha sido escrito completamente en lenguaje Java y, por tanto, puede ejecutarse en cualquier plataforma que soporte Java (se requiere la versión 1.5 o posterior). Funciona exactamente igual en todos los casos, con pequeñas variaciones debidas a cómo cada sistema produce los gráficos. Sin embargo, el proceso de instalación varía un poco según el sistema operativo.

Para su instalación, se debe acudir a la página de descarga http://www.um.es/fem/EjsWiki/Main/Download donde se encuentran tanto la versión más actualizada como otras versiones anteriores.

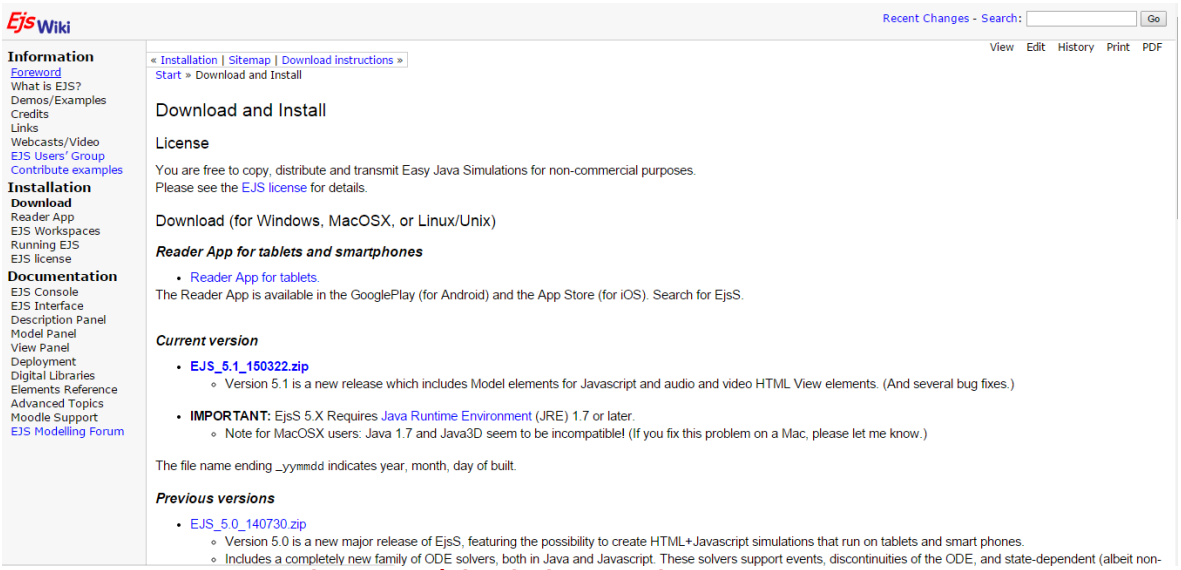


Figura 8-1 Página de descarga de EJS

Una vez descargado, la instalación es muy básica pero por si hubiese alguna duda acerca de la misma, en la página de descarga se encuentran unas instrucciones de instalación.

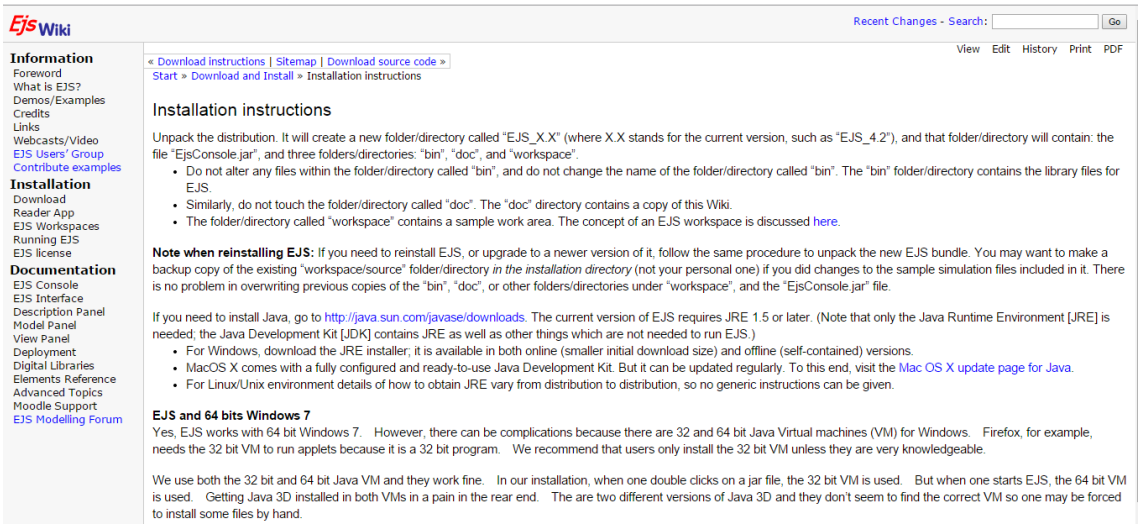


Figura 8-2 Instrucciones de instalación de EJS

Cuando ya se haya realizado satisfactoriamente la instalación, es recomendable crear un acceso directo al archivo *EjsConsole.jar* en el escritorio para tenerlo más a mano y facilitar su ejecución.

8.2 Introducción a EJS

Esta sección va destinada a explicar brevemente las partes de las que consta EJS. Cuando se inicia el programa, se abren dos ventanas:

- La consola de EJS, que consta de tres pestañas: Opciones básicas, Opciones avanzadas y Área de mensajes. Ésta última pestaña es la que aparece por defecto. Es importante comprobar en la pestaña de Opciones avanzadas que el JDK previamente descargado para el uso de NetBeans se encuentre asignado en la opción Java (JDK).

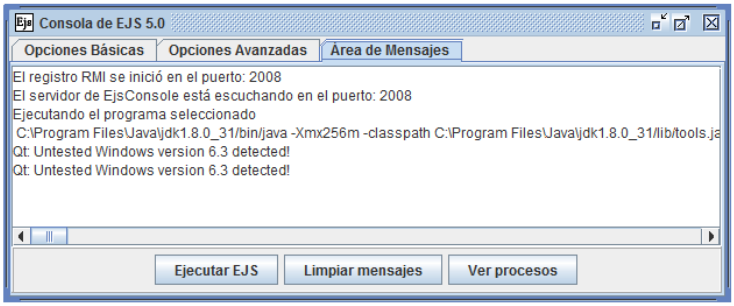


Figura 8-3 Consola de EJS

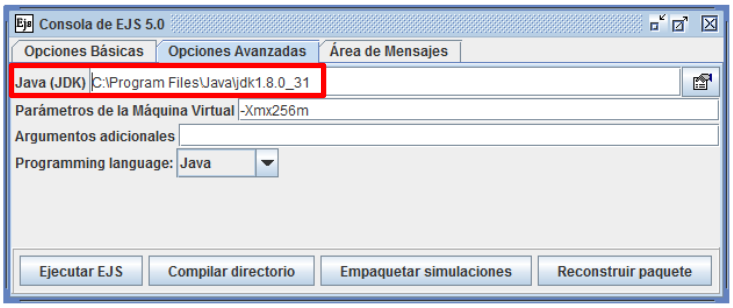


Figura 8-4 Pestaña de Opciones avanzadas

- La ventana de edición de EJS, encabezada por tres botones de radio: Descripción, Modelo y Vista. La parte de Descripción va destinada a recopilar una breve introducción acerca de la simulación en cuestión. En cuanto a Modelo, es la sección donde se generan las variables necesarias, se inicializan y se ejecutan las funciones de manera cíclica que permite el funcionamiento de la simulación. Y la sección de Vista es la que se usará para la creación de la interfaz de simulación.

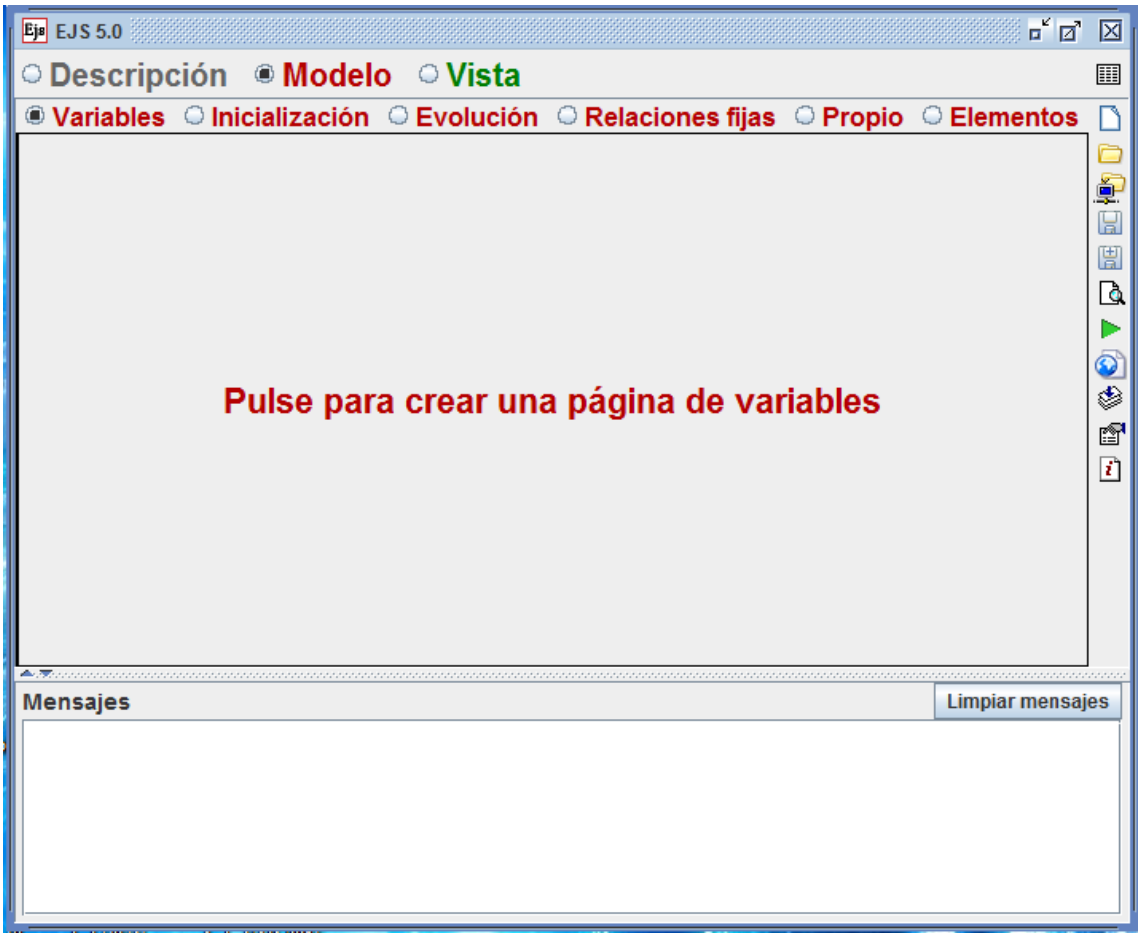


Figura 8-5 Ventana de edición de EJS

Para obtener un poco de habilidad en el manejo de EJS es aconsejable utilizar manuales que se pueden encontrar en Internet, la gran mayoría están en inglés pero en este enlace se puede encontrar en español:

http://fem.um.es/Download/ModelingScience/EjsIntro_es.pdf

Cuando se termine una sesión de trabajo con EJS, la forma más ordenada de terminar es cerrar la ventana de edición de EJS en la forma habitual del sistema operativo. EJS pedirá que se confirme que se desea salir. Otros modos de terminar la sesión con EJS pueden ser más abruptos (por ejemplo, no guardarán el estado de EJS para una futura sesión). Si no ha guardado los últimos cambios, EJS preguntará si se desea hacerlo.

Es posible ejecutar una segunda, o más, copias de EJS, si se desea. Para ello, hay que utilizar los botones *Ejecutar Easy Java Simulations* o *Ejecutar EJS* de la consola de EJS. No es recomendable intentar ejecutar una segunda copia de EJS haciendo de nuevo doble-clic en

EjsConsole.jar (acceso directo) cuando la consola ya está en ejecución. De hecho, si se intenta, EJS preguntará si realmente se desea hacer.

La consola no puede cerrarse directamente. Se cerrará de manera automática cuando la última copia de EJS se cierre.

8.3 -Estructura y configuración del laboratorio remoto

EJS utiliza el concepto de espacio de trabajo para organizar su cometido. El espacio de trabajo es un directorio en el disco duro donde EJS almacena los archivos de las simulaciones de un proyecto determinado, pudiendo almacenar un número ilimitado de éstos. Dentro de un espacio de trabajo, EJS crea cuatro subdirectorios:

- *Config*: Es el directorio donde EJS guarda la configuración determinada por el usuario y otros archivos de opciones.
- *Export*: Es el directorio de destino propuesto por EJS cuando se le generan archivos listos para su distribución.
- *Output:* Es el directorio usado por EJS para situar los archivos temporales generados cuando se compila una simulación.
- *Source*: Es el directorio donde se deben colocar todos los archivos necesarios para la simulación (tanto los fuente como los auxiliares).

En este caso, el proyecto se encontrará ubicado en Dropbox\GIEI\4\UJA\TFG\EJS\EJS_5.0\workspace\source\NEW SRA junto con una *carpeta* aparte que contiene diversas imágenes usadas para el apartado de *Descripción* y un par de archivos .JAR necesarios para el objeto de este proyecto. Uno de ellos está realizado por el desarrollador de este proyecto y es de vital importancia para el correcto funcionamiento del mismo (*srav.jar*) y el otro scormRTE se usa para comunicarse con el LMS.

Al iniciar EJS se abren dos ventanas, hay que centrarse en la ventana de edición, que es donde se trabajará.

En primer lugar, en el apartado de *Descripción* se puede elaborar una presentación explicando qué es lo que hace la simulación.

A continuación se entra de lleno en la programación de la simulación, pasando a la sección de *Modelo*. Se comienza por las variables, las cuales se han organizado en seis tablas para facilitar su identificación y distinguirlas en función de su cometido.

La primera tabla recoge aquellas variables correspondientes a los valores de los sensores almacenados en la base de datos, los parámetros de trabajo del invernadero, una serie de variables que generan un cambio de color en tiempo real y dos variables tipo *String* que almacenan los valores de fecha y hora.

Figura 8-6 Tabla Variables

La tabla siguiente alberga las variables relativas a la conexión con la base de datos remota e informan del éxito de la operación.

Figura 8-7 Tabla Conexión

A continuación en la tabla sucesiva, se encuentran aquellas variables que se encargan de almacenar los datos de los sensores a tiempo real.

Figura 8-8 Tabla Tiempo Real

La tabla *Cadenas* contiene una serie de variables tipo *String* que serán útiles a la hora de visualizar los datos de los sensores a tiempo real, los almacenados en la base de datos, información acerca de los actuadores presentes así como las preguntas y respuestas de este applet.

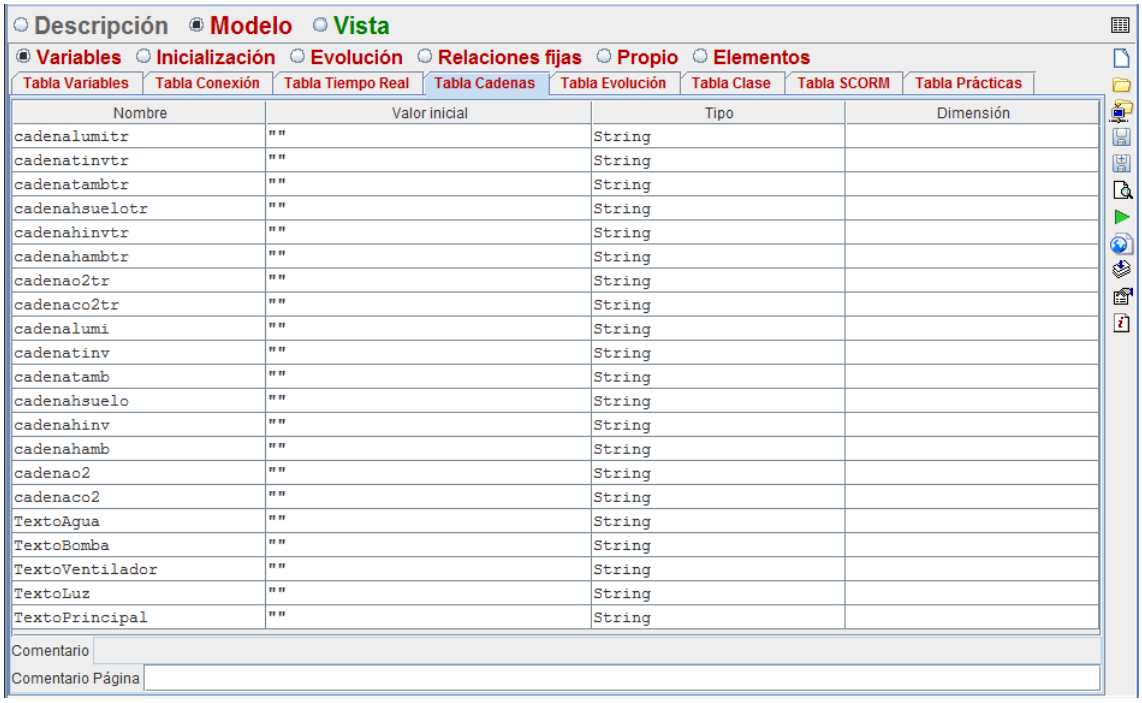


Figura 8-9 Tabla Cadenas

En la tabla *Evolución* están contenidas las variables que se utilizarán para la pestaña *Evolución* propiamente dicha.

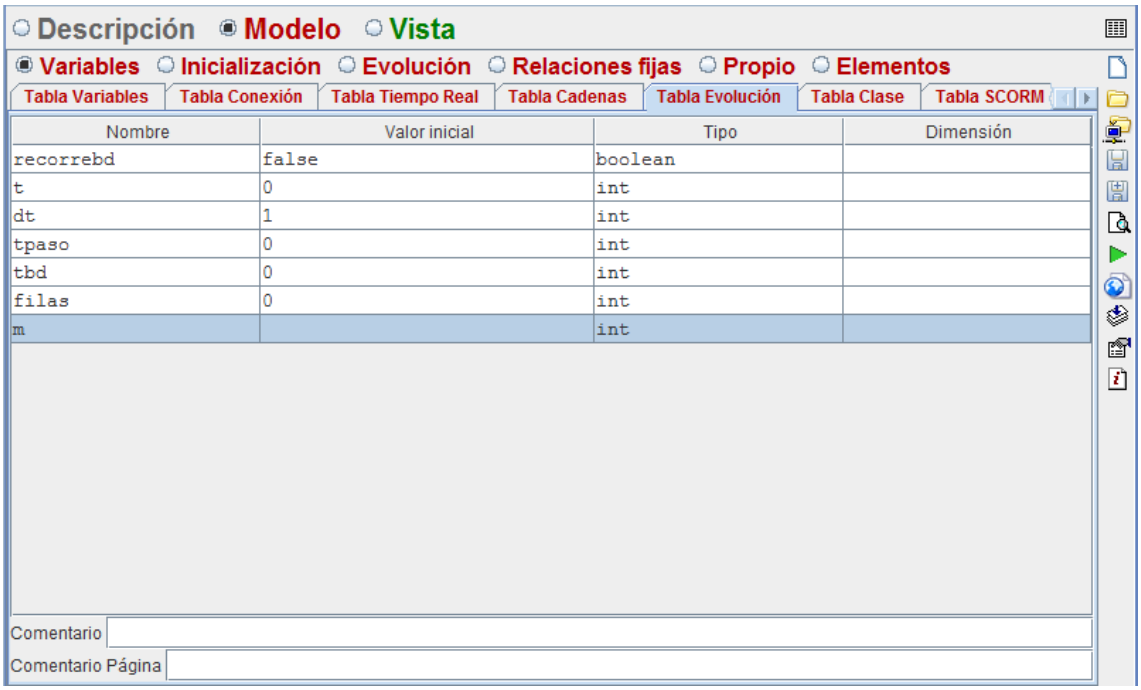


Figura 8-10 Tabla Evolución

En la tabla *Clase* se define un objeto de la clase generada específicamente para el desarrollo de este proyecto.

Figura 8-11 Tabla Clase

Se puede apreciar que el tipo de estas últimas variables no es uno de los usuales que tiene EJS de manera predeterminada, de hecho es uno creado por el autor del proyecto. Su función principal es asociarse con los archivos .JAR, los cuales comunican al simulador con el invernadero de manera remota.

Para que EJS permita realizar esta asignación, se debe tener el archivo .JAR guardado en una carpeta dentro del directorio de trabajo (como ya se ha explicado anteriormente). Generar un archivo .JAR es sencillo, se debe hacer click derecho en el proyecto de NetBeans que se desee convertir a .JAR, ir a *Properties* y en la ventana que aparece pulsar en *Packaging* y marcar la opción que dice *Compress JAR File*.

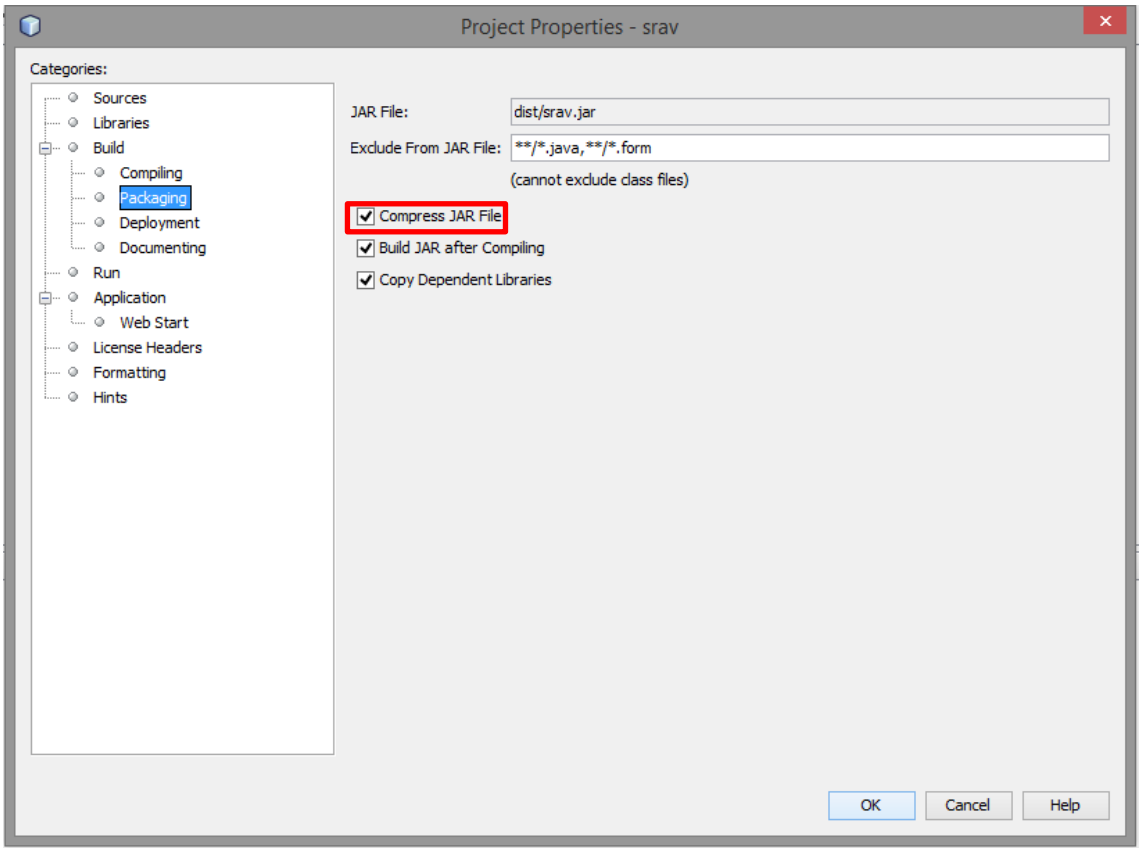


Figura 8-12 Obtención de un archivo .JAR en NetBeans

Una vez hecho esto, se debe pulsar en la opción de Build Project (símbolo de un martillo) y generará un archivo .JAR que se encontrará dentro de la carpeta dist del proyecto.

Aparte de en el directorio de trabajo, tanto este archivo .JAR ,como el conector de MySQL (*mysql-connector-java-5.1.34.jar*) y como scormRTE deberán copiarse en el directorio *extensions* que se encuentra en la siguiente raíz: *C:\EJS_5.0\bin\extensions* para la compilación en EJS.

Volviendo dentro del entorno de EJS, es necesario realizar unos pequeños ajustes para lograr la comunicación con el archivo .JAR por lo que hay que acceder al panel de información de la consola (situado en la parte superior a la derecha).

Dentro de este panel, dentro de la segunda pestaña llamada *Opciones de ejecución* hay que importar todos los archivos generados para la comunicación remota y por último, incluir en el apartado de librerías JAR el conector a la base de datos que habilitará la declaración del objeto Srav en la Tabla Clase y la utilización del paquete scormRTE.

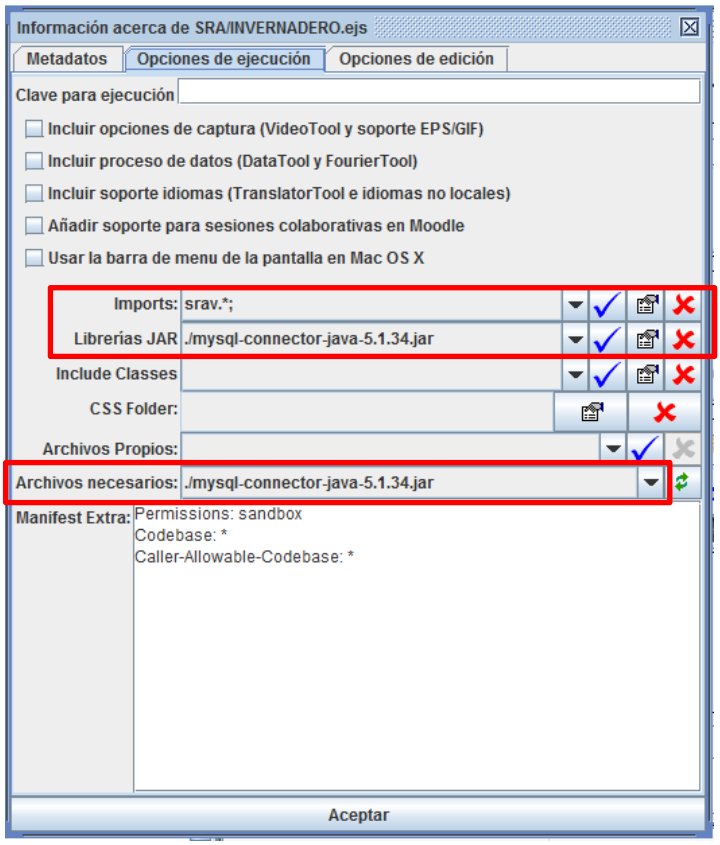


Figura 8-13 Panel de información de EJS

La siguiente tabla hace referencia a los elementos SCORM que se debe asociar con esta simulación:

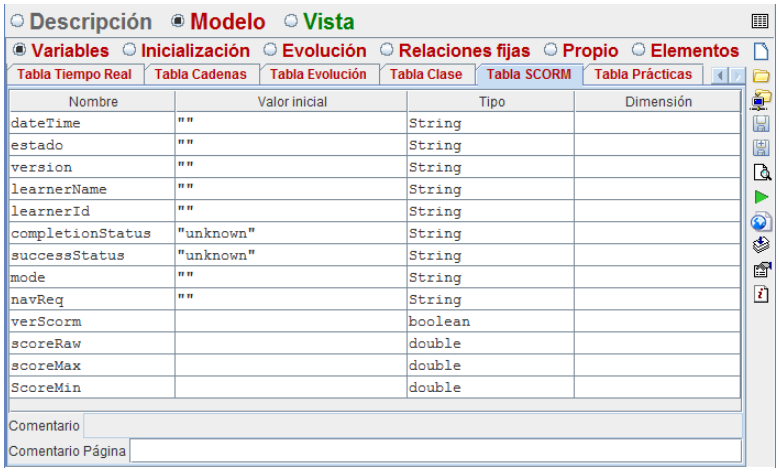


Figura 8-14 Tabla SCORM

La última tabla alberga aquellas variables relacionadas con el apartado de cuestiones que posee la simulación:

Figura 8-15 Tabla Prácticas

Realizado todo esto, ya se puede decir que el sistema EJS está correctamente configurado y continuar con el paso siguiente, la inicialización. Dentro del apartado *Modelo* hay una sección que se encarga de esto puesto que tiene el mismo nombre, pero para este proyecto se ha optado por otra solución.

El acceso a la base de datos se realizará en uno de los botones que tiene la simulación, más concretamente en el botón *Conectar* y en él se realiza la asignación de diversas variables, la muestra de datos de los sensores conectados y recoge los últimos parámetros de trabajo albergados en la base de datos.

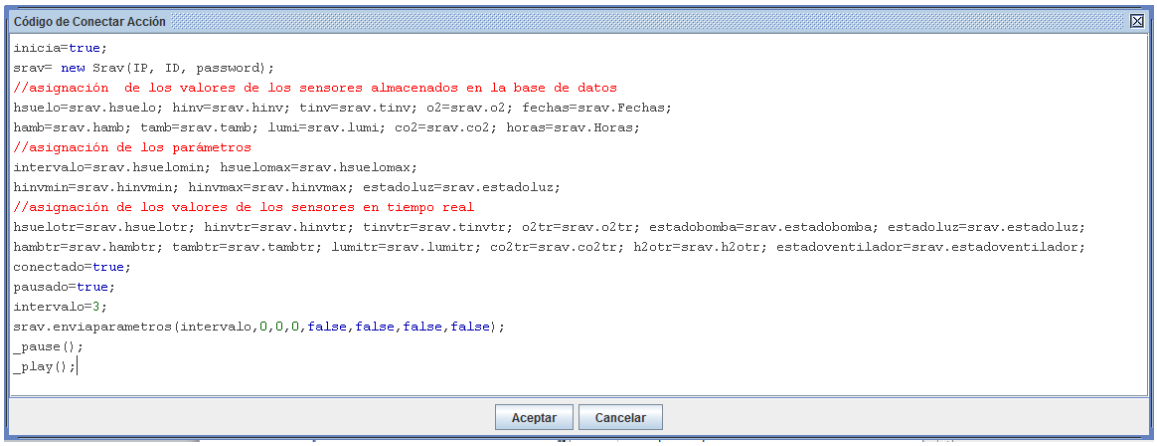


Figura 8-16 Código referente al botón Conectar

A continuación se muestra más detalladamente el código de la imagen anterior:

```
inicia=true;
srav= new Srav(IP, ID, password);
//asignación  de los valores de los sensores almacenados en la base de datos
hsuelo=srav.hsuelo; hinv=srav.hinv; tinv=srav.tinv; o2=srav.o2;
fechas=srav.fechas; hamb=srav.hamb; tamb=srav.tamb; lumi=srav.lumi;
co2=srav.co2; horas=srav.horas;
//asignación de los parámetros
hsuelomin=srav.hsuelomin; hsuelomax=srav.hsuelomax;
hinvmin=srav.hinvmin; hinvmax=srav.hinvmax;
//asignación de los valores de los sensores en tiempo real
hsuelotr=srav.hsuelotr; hinvtr=srav.hinvtr; tinvtr=srav.tinvtr;
o2tr=srav.o2tr; estadobomba=srav.estadobomba; estadoluz=srav.estadoluz;
hambtr=srav.hambtr; tambtr=srav.tambtr; lumitr=srav.lumitr;
co2tr=srav.co2tr; h2otr=srav.h2otr; estadoventilador=srav.estadoventilador;
conectado=true;
pausado=true;
intervalo=3;
srav.enviaparametros(intervalo,0,0,0,false,false,false,false);
_pause();
_play();
```

Código 8-1 Botón Conectar

El paso siguiente, dentro de *Modelo* es el apartado de *Evolución*. En EJS, la evolución es un bucle infinito que se repite una y otra vez, lo cual es perfecto para poder obtener en todo momento los valores que recogen los sensores del invernadero.

El código que recoge lo explicado anteriormente se muestra a continuación:

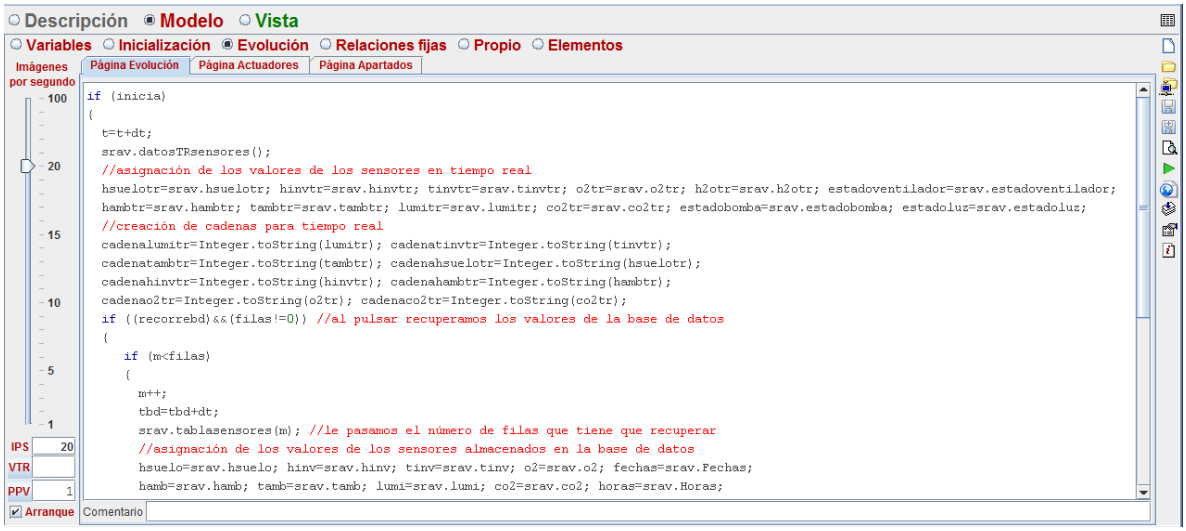


Figura 8-14 Código referente a Evolución

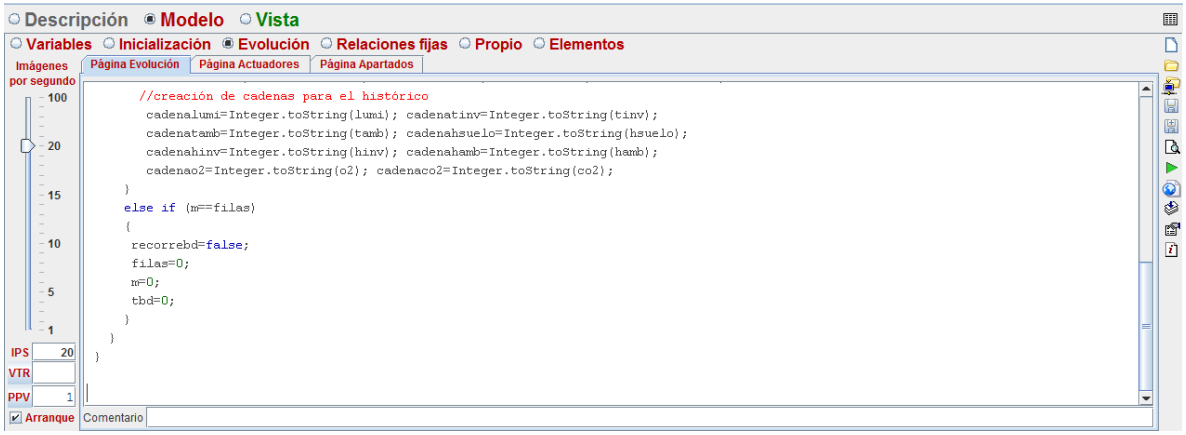


Figura 8-158 Código referente a Evolución

Para visualizar mejor el código referente a la pestaña Evolución, se muestra a continuación:

```
if (inicia)
{
  t=t+dt;
  srav.datosTRsensores();
  //asignación de los valores de los sensores en tiempo real
  hsuelotr=srav.hsuelotr; hinvtr=srav.hinvtr; tinvtr=srav.tinvtr;
o2tr=srav.o2tr; h2otr=srav.h2otr; estadoventilador=srav.estadoventilador;
hambtr=srav.hambtr; tambtr=srav.tambtr; lumitr=srav.lumitr;
co2tr=srav.co2tr; estadobomba=srav.estadobomba; estadoluz=srav.estadoluz;
  //creación de cadenas para tiempo real
  cadenalumitr=Integer.toString(lumitr);
cadenatinvtr=Integer.toString(tinvtr);
  cadenatambtr=Integer.toString(tambtr);
cadenahsuelotr=Integer.toString(hsuelotr);
  cadenahinvtr=Integer.toString(hinvtr);
cadenahambtr=Integer.toString(hambtr);
  cadenao2tr=Integer.toString(o2tr); cadenaco2tr=Integer.toString(co2tr);
  if ((recorrebd)&&(filas!=0)) //al pulsar se recupera los valores de la
base de datos
  {
     if (m<filas)
     {
       m++;
       tbd=tbd+dt;
       srav.tablasensores(m); //se le pasa el número de filas que tiene que
recuperar
       //asignación de los valores de los sensores almacenados en la base
de datos
       hsuelo=srav.hsuelo; hinv=srav.hinv; tinv=srav.tinv; o2=srav.o2;
       hamb=srav.hamb; tamb=srav.tamb; lumi=srav.lumi; co2=srav.co2;
       //creación de cadenas para el histórico
        cadenalumi=Integer.toString(lumi);
        cadenatinv=Integer.toString(tinv);
        cadenatamb=Integer.toString(tamb);
        cadenahsuelo=Integer.toString(hsuelo);
        cadenahinv=Integer.toString(hinv);
        cadenahamb=Integer.toString(hamb);
        cadenao2=Integer.toString(o2);
        cadenaco2=Integer.toString(co2);
     }
```

```
      else if (m==filas)
      {
       recorrebd=false;
       filas=0;
       m=0;
       tbd=0;
      }
    }
}
```

Código 8-2 Evolución

A continuación se muestra la página que modifica las variables tipo *String* que posee la interfaz:

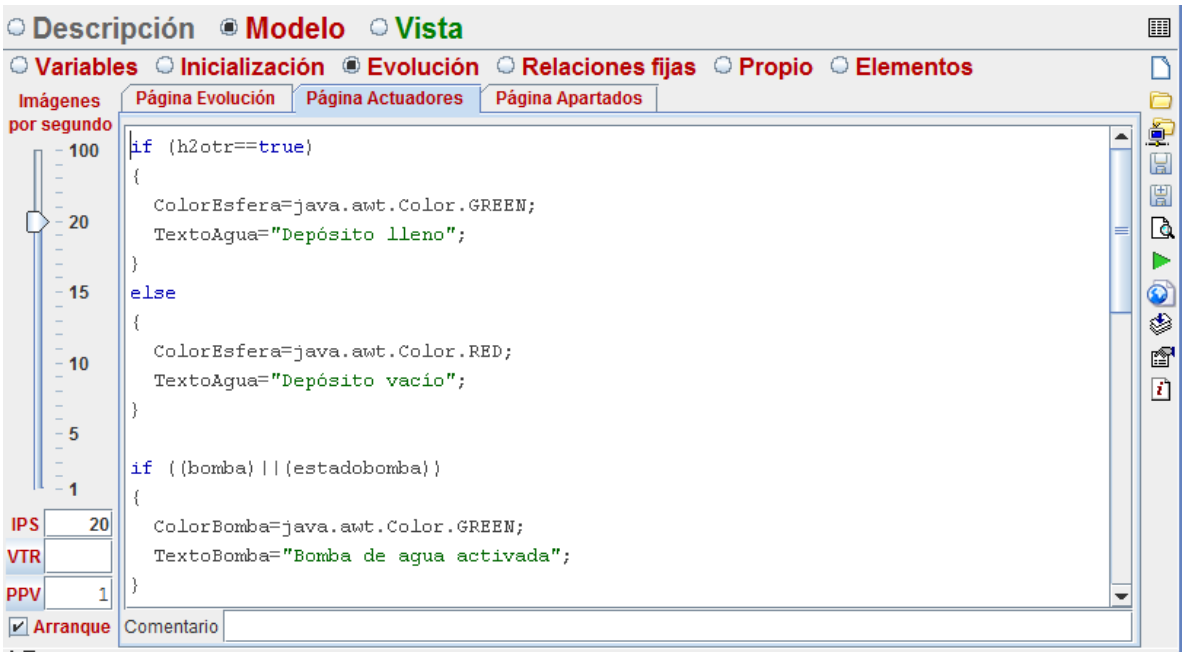


Figura 8-19 Código referente a Actuadores

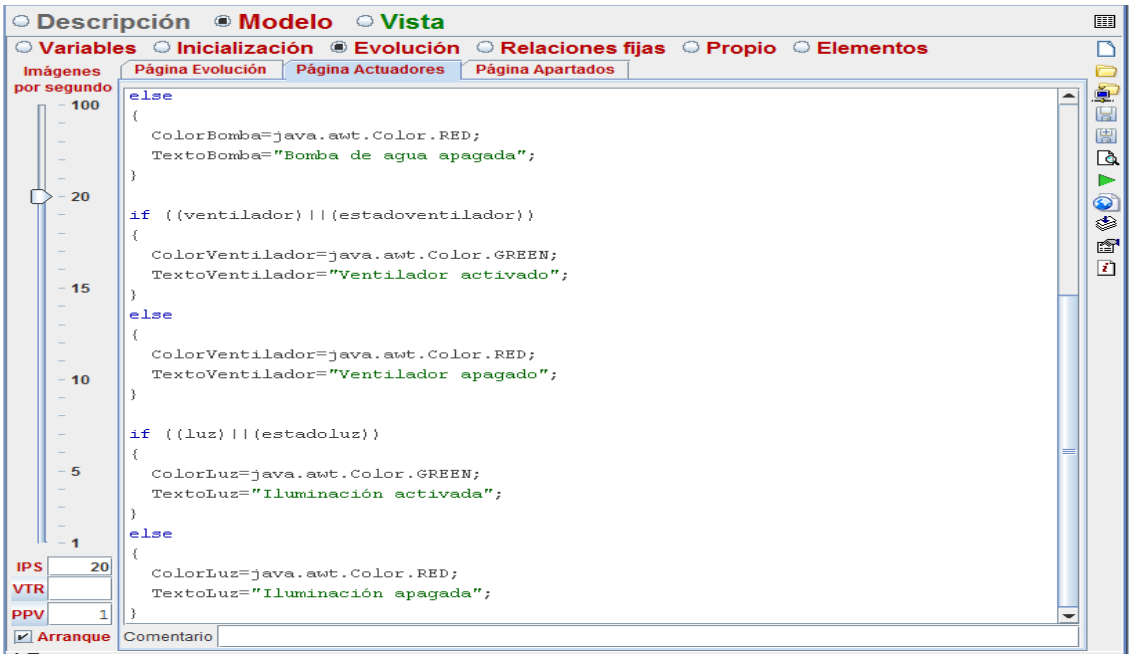


Figura 8-20 Código referente a Actuadores

De igual manera que en el caso anterior, a continuación se recoge el código referente a la pestaña Actuadores:

```
if (h2otr==true)
{
  ColorEsfera=java.awt.Color.GREEN;
  TextoAgua="Depósito lleno";
}
else
{
  ColorEsfera=java.awt.Color.RED;
  TextoAgua="Depósito vacío";
}

if ((bomba)||(estadobomba))
{
  ColorBomba=java.awt.Color.GREEN;
  TextoBomba="Bomba de agua activada";
}
else
{
  ColorBomba=java.awt.Color.RED;
  TextoBomba="Bomba de agua apagada";
}

if ((ventilador)||(estadoventilador))
{
  ColorVentilador=java.awt.Color.GREEN;
  TextoVentilador="Ventilador activado";
}
else
{
  ColorVentilador=java.awt.Color.RED;
  TextoVentilador="Ventilador apagado";
}

if ((luz)||(estadoluz))
{
  ColorLuz=java.awt.Color.GREEN;
  TextoLuz="Iluminación activada";
}
else
{
  ColorLuz=java.awt.Color.RED;
  TextoLuz="Iluminación apagada";
}
```

Código 8-3 Strings

Por último, existe una página dedicada a la creación de preguntas y respuestas en función de la ventana en la que se encuentre el cliente:

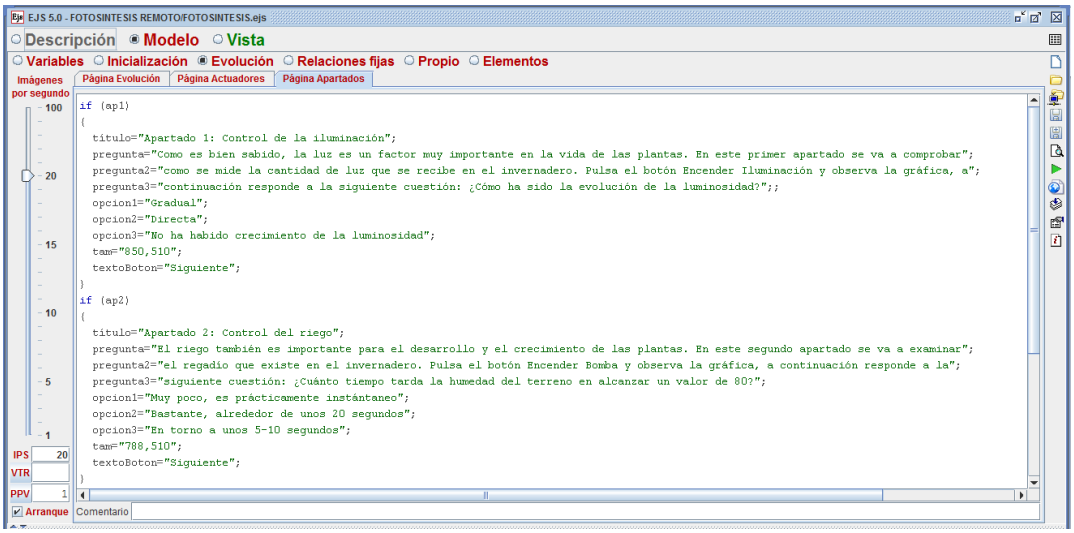


Figura 8-21 Código referente a Apartados

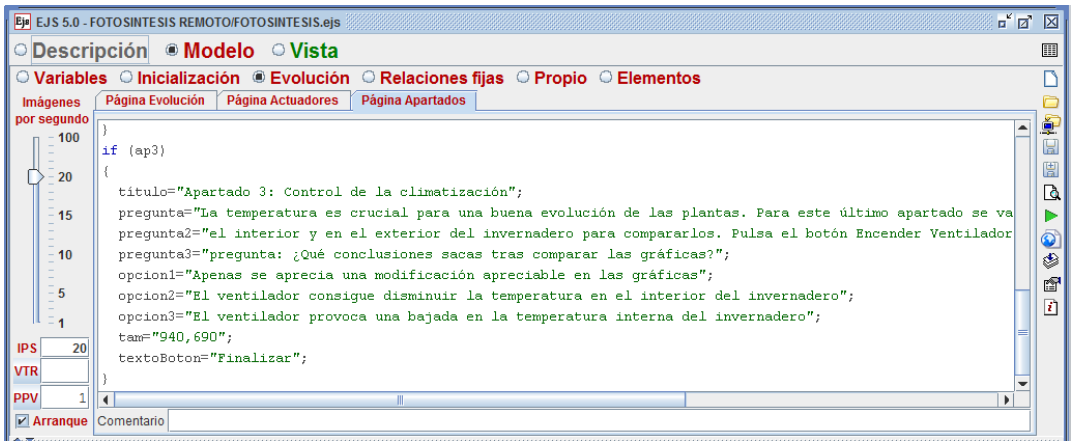


Figura 8-22 Código referente a Apartados

A continuación aparece el código referente a la pestaña de Apartados:

```
if (ap1)
{
  título="Apartado 1: Control de la iluminación";
  pregunta="Como es bien sabido, la luz es un factor muy importante en la
vida de las plantas. En este primer apartado se va a comprobar";
  pregunta2="como se mide la cantidad de luz que se recibe en el
invernadero. Pulsa el botón Encender Iluminación y observa la gráfica, a";
  pregunta3="continuación responde a la siguiente cuestión: ¿Cómo ha sido
la evolución de la luminosidad?";
  opcion1="Gradual";
  opcion2="Directa";
  opcion3="No ha habido crecimiento de la luminosidad";
  tam="850,510";
  textoBoton="Siguiente";
}
if (ap2)
{
  título="Apartado 2: Control del riego";
  pregunta="El riego también es importante para el desarrollo y el
crecimiento de las plantas. En este segundo apartado se va a examinar";
  pregunta2="el regadío que existe en el invernadero. Pulsa el botón
Encender Bomba y observa la gráfica, a continuación responde a la";
```

```
  pregunta3="siguiente cuestión: ¿Cuánto tiempo tarda la humedad del
terreno en alcanzar un valor de 80?";
  opcion1="Muy poco, es prácticamente instantáneo";
  opcion2="Bastante, alrededor de unos 20 segundos";
  opcion3="En torno a unos 5-10 segundos";
  tam="788,510";
  textoBoton="Siguiente";
}
if (ap3)
{
  título="Apartado 3: Control de la climatización";
  pregunta="La temperatura es crucial para una buena evolución de las
plantas. Para este último apartado se va a observar los valores de
temperatura y humedad en";
  pregunta2="el interior y en el exterior del invernadero para compararlos.
Pulsa el botón Encender Ventilador y observa las 4 gráficas, después
contesta a la siguiente";
  pregunta3="pregunta: ¿Qué conclusiones sacas tras comparar las
gráficas?";
  opcion1="Apenas se aprecia una modificación apreciable en las gráficas";
  opcion2="El ventilador consigue disminuir la temperatura en el interior
del invernadero";
  opcion3="El ventilador provoca una bajada en la temperatura interna del
invernadero";
  tam="940,690";
  textoBoton="Finalizar";
}
```

Código 8-4 Apartados

Después de esto, no será necesario programar en ninguno de los otros apartados que tiene *Modelo* debido al paquete srav.JAR que se ha creado.

Llegados a este punto, tan solo queda el panel de *Vista* en el que se configura la interfaz de la simulación. Para este proyecto se han usado dos pantallas: La primera muestra la evolución de los valores de los sensores a tiempo real, la segunda muestra el histórico de los valores de los sensores recogido en la base de datos y en ambas se visualiza la cámara IP.

La creación de la interfaz es muy sencilla ya que simplemente se ha de seleccionar los elementos deseados entre los existentes en la parte derecha y colocarlos sobre *Vista de simulación*. En los manuales que se mencionaron en apartados anteriores se explica de manera más detallada el uso y manejo de la sección *Vista*.

En la siguiente imagen se puede ver como se han insertado todos los elementos que sean necesarios (botones, gráficas, textos, paneles, etc)

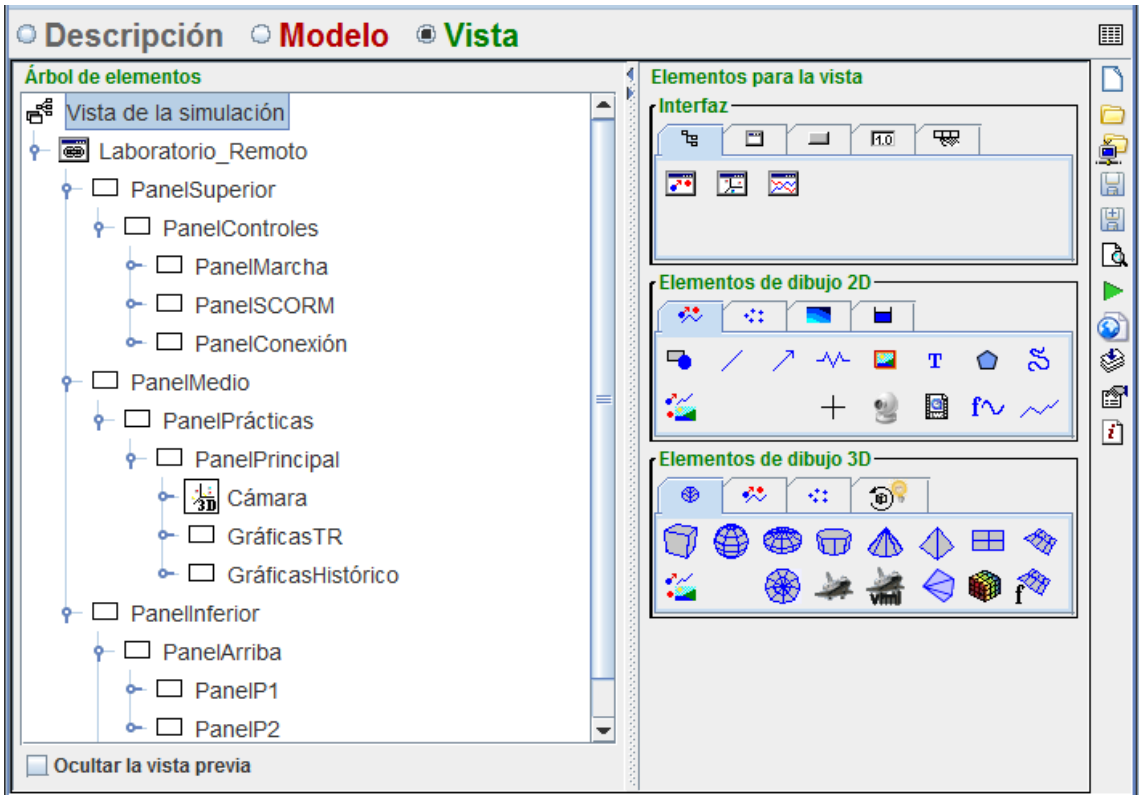


Figura 8-23 Panel Vista de EJS

Una vez que toda la interfaz esté creada, es momento de aportar funcionalidad a algunos elementos como por ejemplo el botón *Desconectar:*

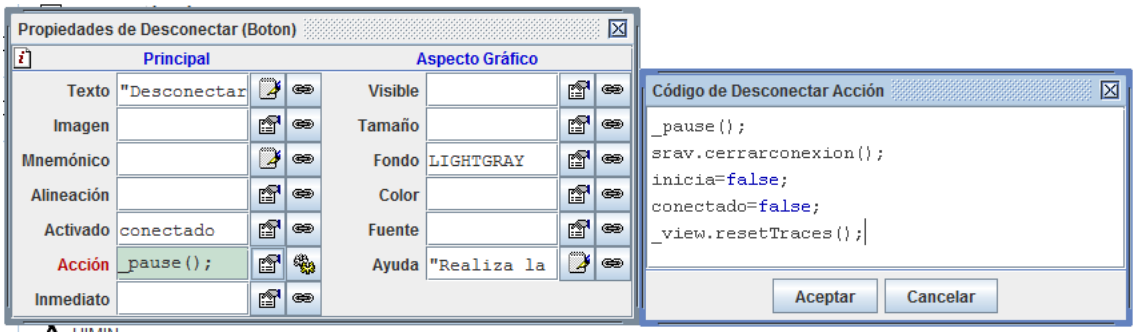


Figura 8-24 Código referente al botón Desconectar

El código que se muestra a continuación va referido a la imagen anterior:

```
_pause();
srav.cerrarconexion();
inicia=false;
conectado=false;
_view.resetTraces();
```

Código 8-6 Desconectar

El código implementado en la imagen anterior indica que cuando sea pulsado debe pausar la aplicación, desconectarse de la base de datos, resetear las gráficas e informar de su estado actual.

Los botones de *Play* y *Pause* son más sencillos ya que simplemente tienen la acción de pausar o reanudar la simulación.

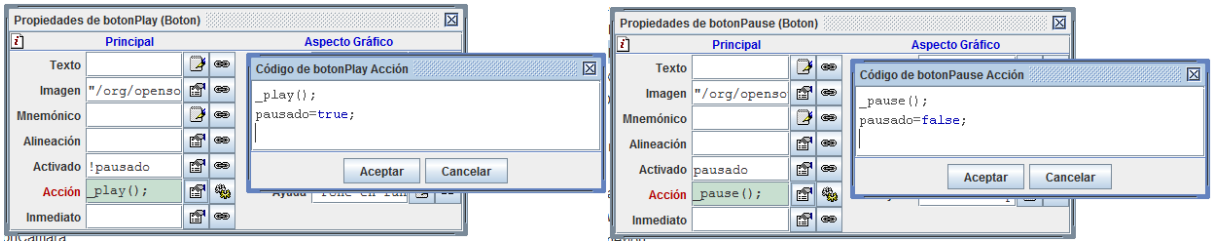


Figura 8-25 Botón Play y Pause

Otros botones importantes son aquellos que activan (o desactivan) distintos actuadores como el ventilador, la bomba de agua o la iluminación. Son botones de dos estados y su configuración es la siguiente:

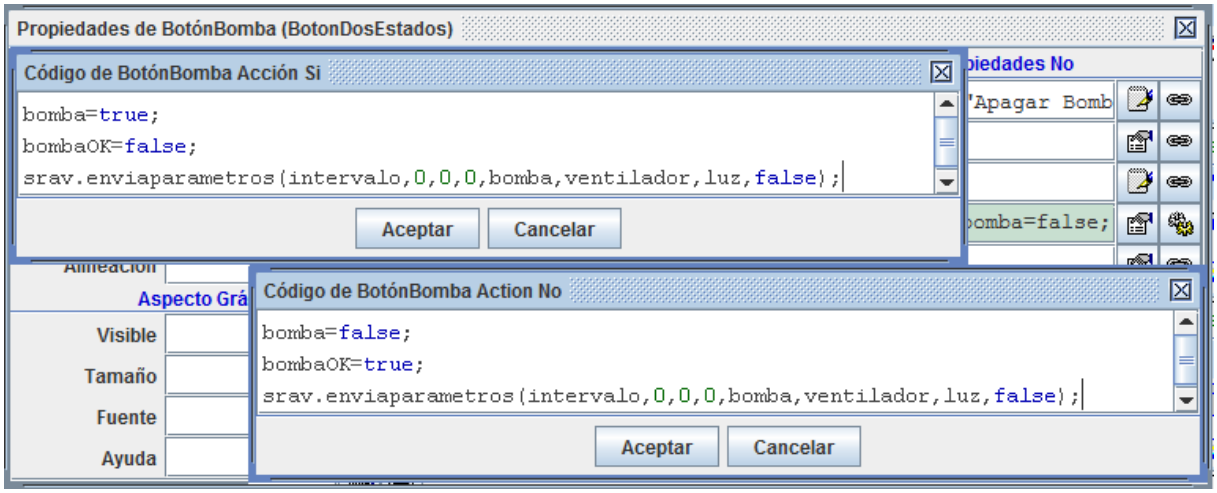


Figura 8-26 Propiedades referentes al botón Bomba

Para el caso de los otros dos botones que activan el ventilador y la iluminación, el código es exactamente el mismo que en la imagen anterior exceptuando el nombre de la variable en sí.

A continuación se comenta la parte perteneciente a las gráficas de tiempo real, en este proyecto hay ocho gráficas (una para cada sensor) divididas en tres apartados que muestran los datos obtenidos por los sensores en tiempo real.

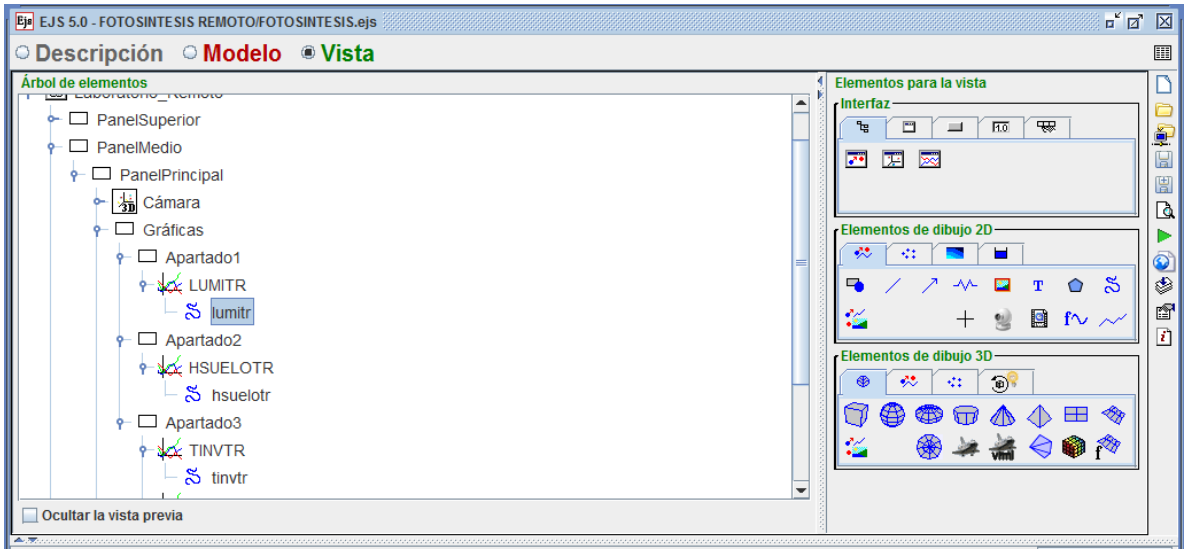


Figura 8-27 Panel de gráficas

Cada una de ellas contiene un rastro, al que se le asocia la variable adecuada que hace referencia al sensor correspondiente. Se tiene una pantalla por apartado, este laboratorio remoto está dividido en función de cada actuador (es decir, 3 en total). Con esta estructuración se busca un orden más lógico.

En la parte baja de la pantalla se ubican los botones de dos estados mencionados con anterioridad y varias esferas que cambian de color en función de unas variables booleanas y que avisan del nivel del depósito de agua o de la activación o desactivación de los actuadores. Además, también se incluyen las preguntas con sus tres opciones, que son requisito indispensable para avanzar al siguinte apartado.

Las esferas son dibujos 2D con un tamaño limitado por el usuario y cuyo color va asociado a una variable de tipo *Object* la cual se modifica en función del valor de una variable booleana como se puede observar en la página Actuadores de la *Evolución* (Ilustraciones 8-19 y 8-20).

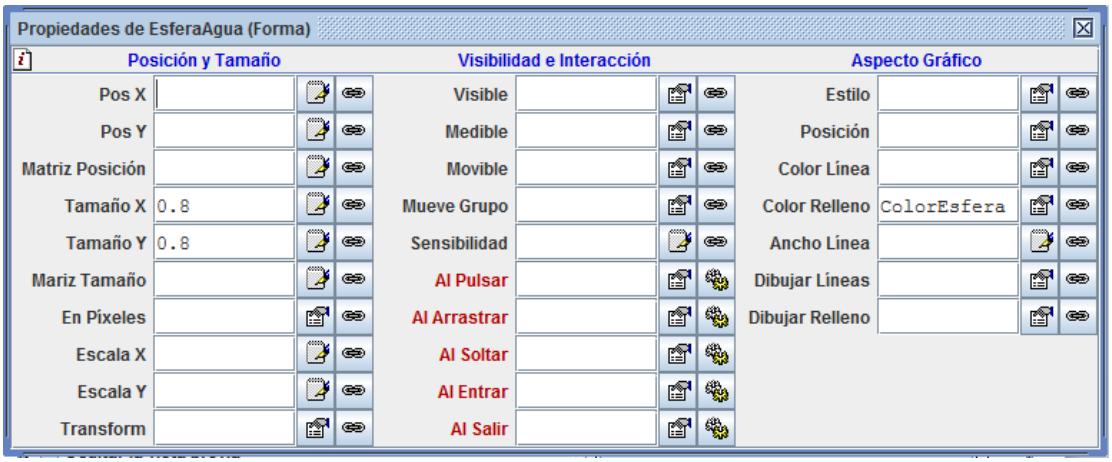


Figura 8-28 Propiedades referentes a EsferaAgua

La anterior Figura sirve para mostrar la configuración de todas los dibujos 2D utilizados puesto que lo único que cambia es la variable tipo *Object* que afecte al color de relleno.

Otros botones importantes en este laboratorio se encuentran en la parte superior de la pantalla. Los botones *Borrar registro* y *Enviar* se encargan de comunicarse con el servidor para mandar una serie de instrucciones.

El botón *Borrar registro* llama a una función del paquete srav citada con anterioridad que se encarga de vaciar una de las tablas de la base de datos.

En cuanto al botón *Enviar* se encarga de alterar el valor de la variable intervalo, que modifica el número de segundos con el que la placa de Arduino recoge y almacena información de los sensores.

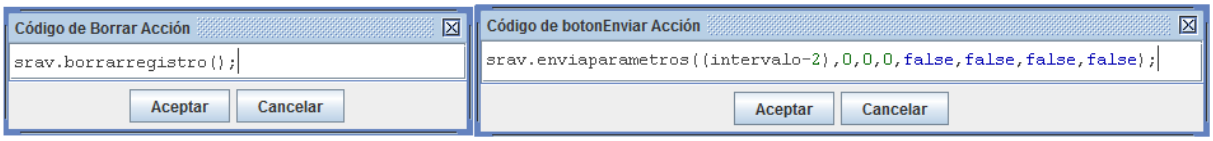


Figura 8-169 Código referente a los botones Borrar registro y Enviar

Por último, el panel dedicado a la cámara contiene un elemento de dibujo 3D llamado *remoteARSystem* en el que se configura su URL y otros datos importantes como el usuario y la contraseña.

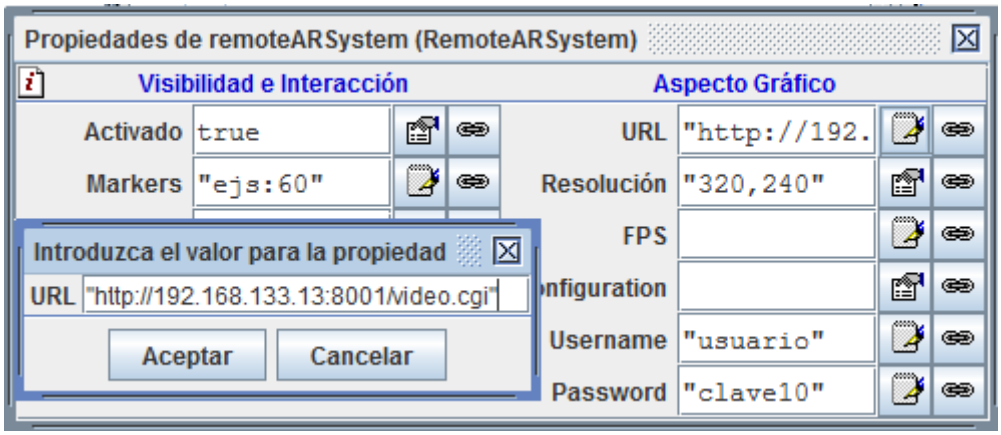


Figura 8-30 Propiedades referentes a remoteARSystem

Una vez configurados todos los botones, gráficas y el resto de elementos de la simulación ha llegado el momento de ejecutarlo, realizando previamente un guardado de seguridad. La puesta en marcha se realiza pulsando sobre el botón verde de la ventana principal de EJS.

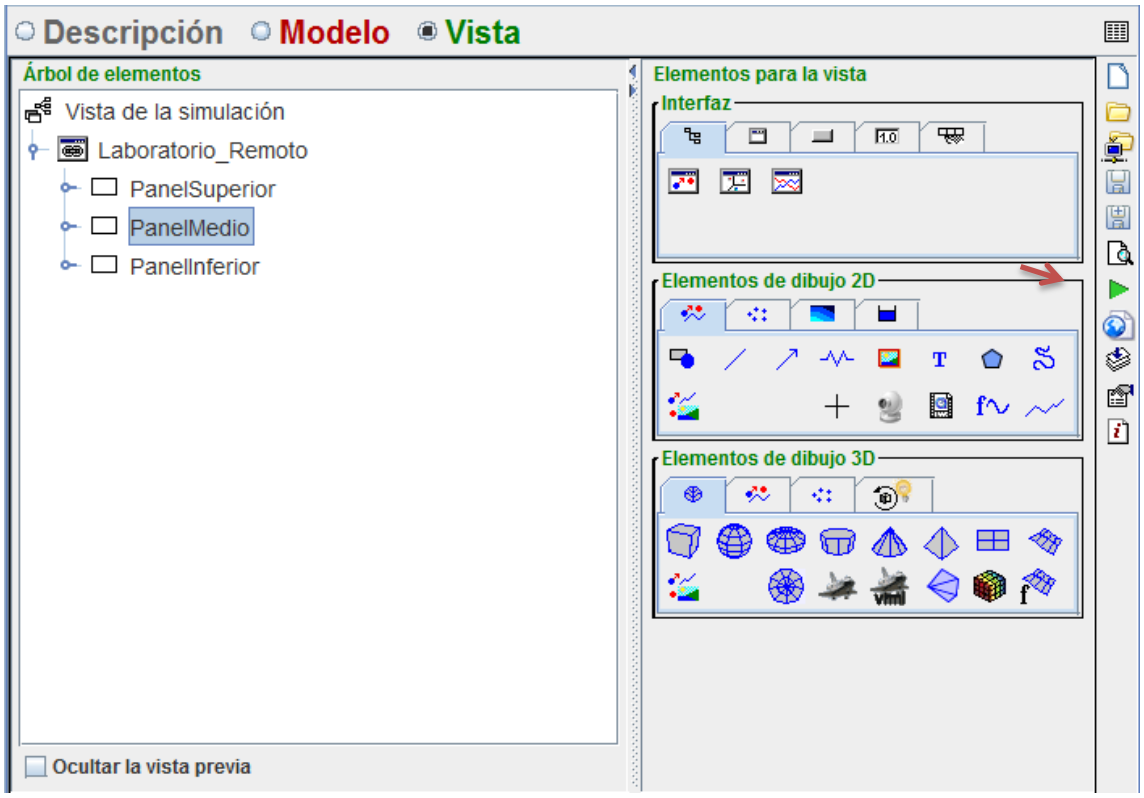


Figura 8-31 Botón play para poner en marcha la simulación

Si se ejecuta de manera correcta, el botón pasará a convertirse en y por lo tanto no habrá ningún error en la ventana de mensajes. De lo contrario, es necesario revisar la programación realizada para solventar los errores que la ventana muestra.

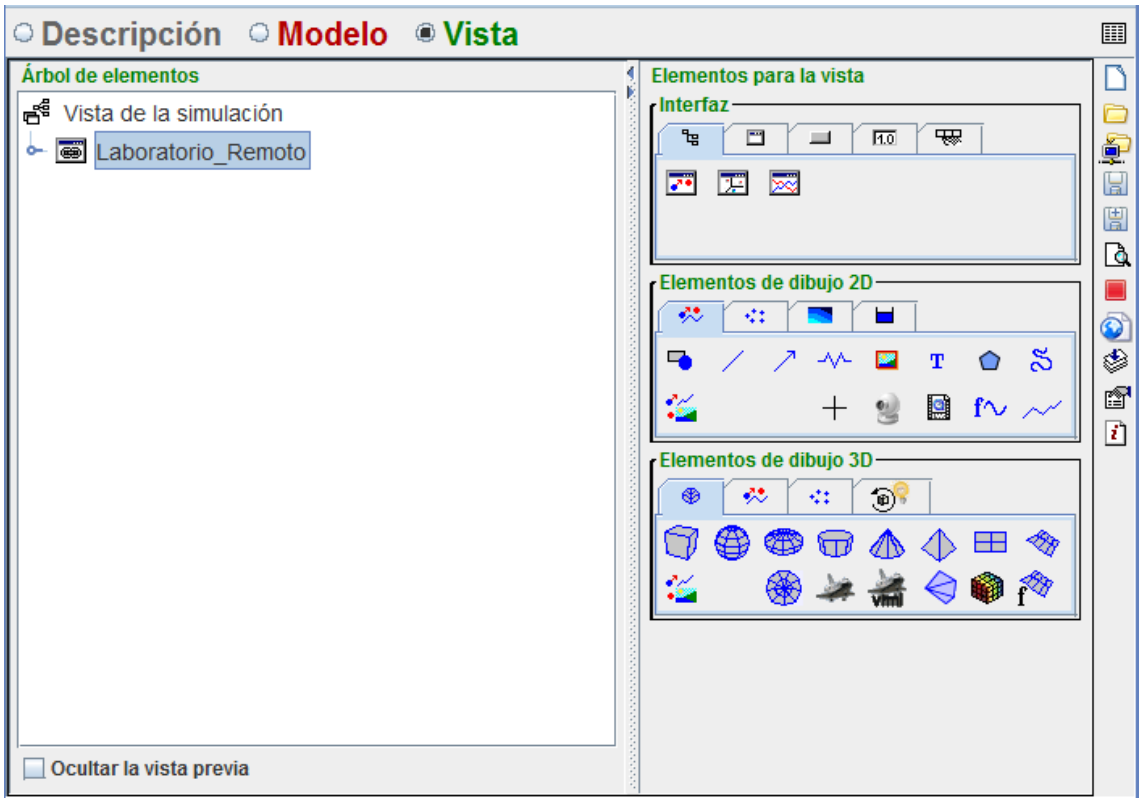


Figura 8-32 Ejecución de la simulación

Además de la propia ventana de mensajes, es importante mirar la ventana de la consola de EJS, especialmente la pestaña de *Área de mensajes* ya que es donde aparecerán los errores una vez que la simulación haya arrancado de manera correcta.

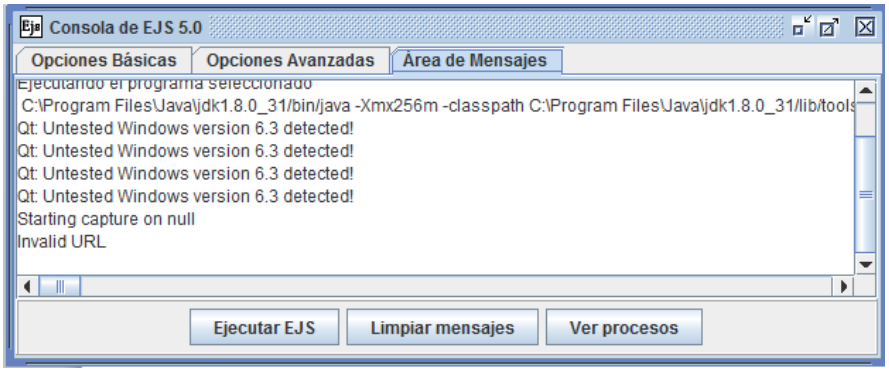


Figura 8-33 Consola de EJS

8.4 Estructura y configuración del laboratorio virtual.

En primer lugar comenzar con la sección *Variables* de la pestaña *Evolución* en la que existen varios apartados. A continuación se va a hablar con más detalle de las variables presentes en dichos apartados.

La Tabla Variables almacena una serie de valores que permiten visualizar la imagen y el texto correspondiente a la planta seleccionada, controlar el movimiento del deslizador y otras variables importantes para crear un bucle cíclico.

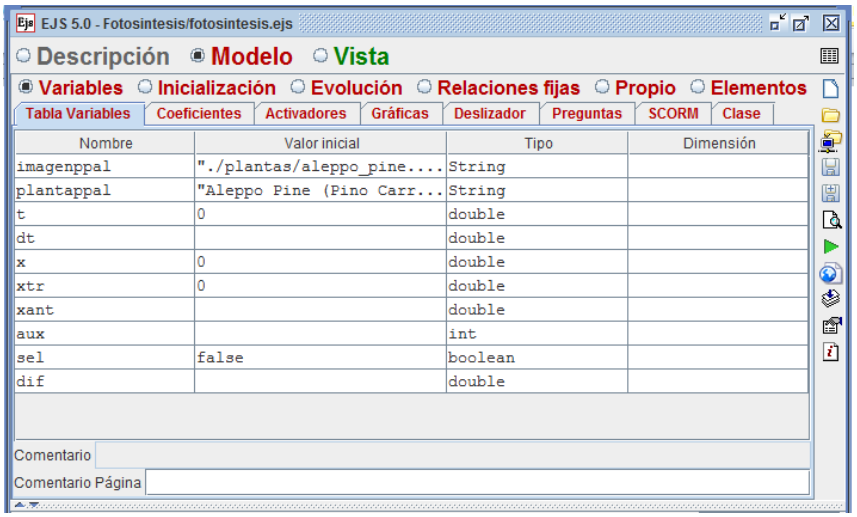


Figura 8-3417 Tabla Variables

En Coeficientes se recoge aquellas variables útiles que servirán como coeficientes de las ecuaciones que se representan en el gráfico.

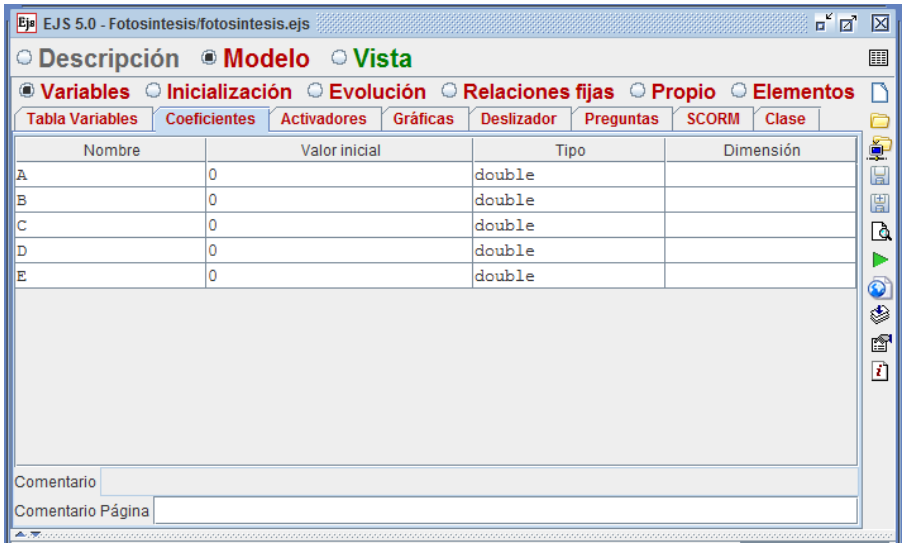


Figura 8-3518 Coeficientes

En el apartado Activadores, se encuentran las variables referentes a la selección de las imágenes para su representación.

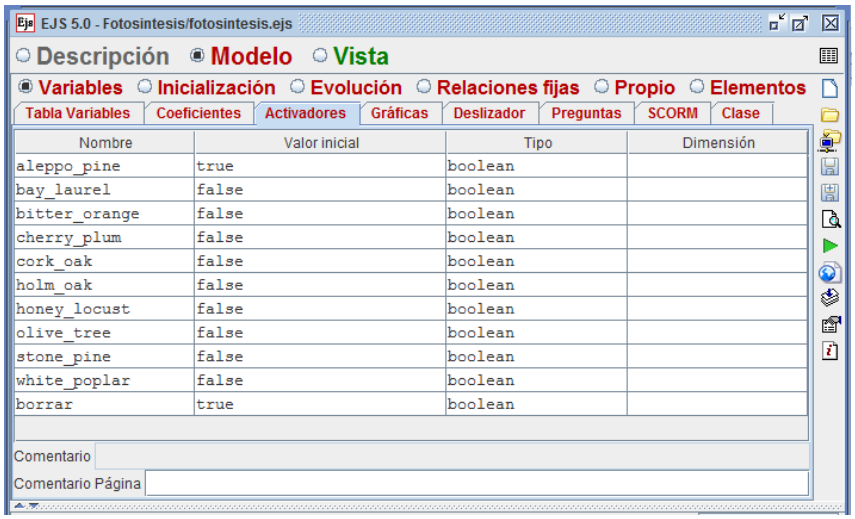


Figura 8-36 Activadores

En Gráficas, se almacenan las 10 trazas (una por planta) que se utilizan para graficar la evolución de la eficiencia fotosintética de cada una de ellas.

Figura 8-37 Gráficas

En el apartado Deslizador tan solo se encuentra la variable que provoca un cambio de color en función del aumento de la variable asociada a dicho deslizador.

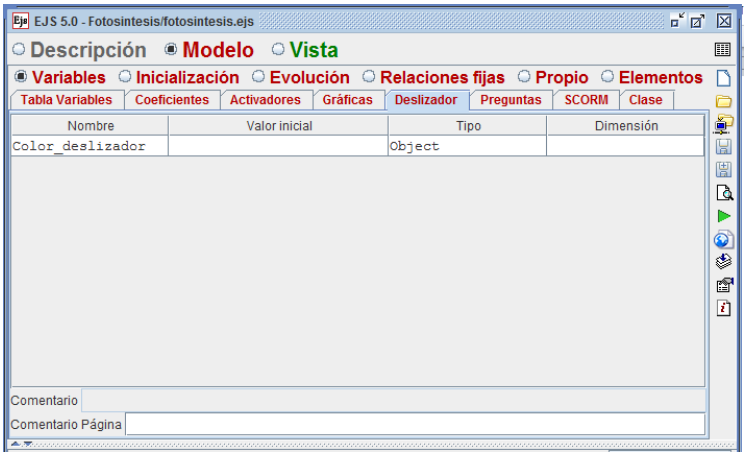


Figura 8-38 Deslizador

En Preguntas, se recogen aquellas variables referentes a la modificación de las cuestiones y a las distintas opciones posibles para contestar.

Figura 8-39 Preguntas

En SCORM, se encuentran todas aquellas variables relacionadas con la comunicación SCORM así como las puntuaciones que se otorgan en función de las respuestas escogidas.

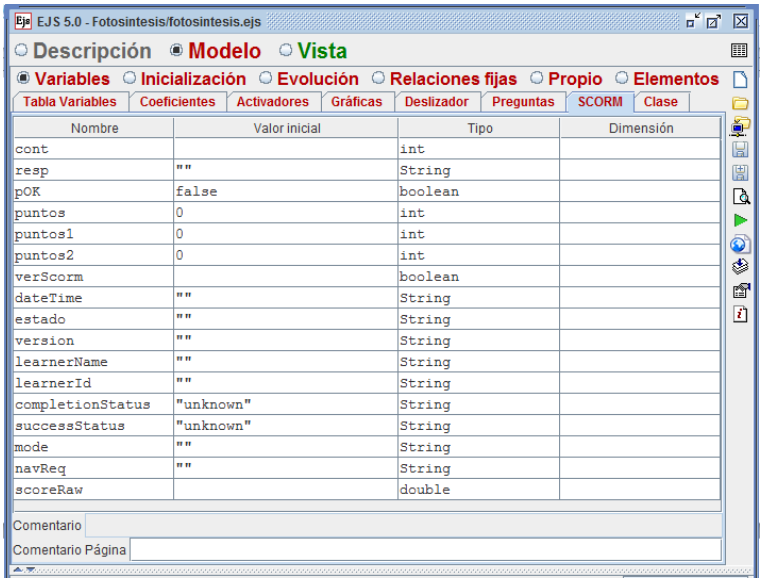


Figura 8-40 SCORM

Por último, en el apartado Clase se encuentra la variable que se asocia con el paquete scormRTE que se debe incluir para conseguir la comunicación del applet con el SCORM.

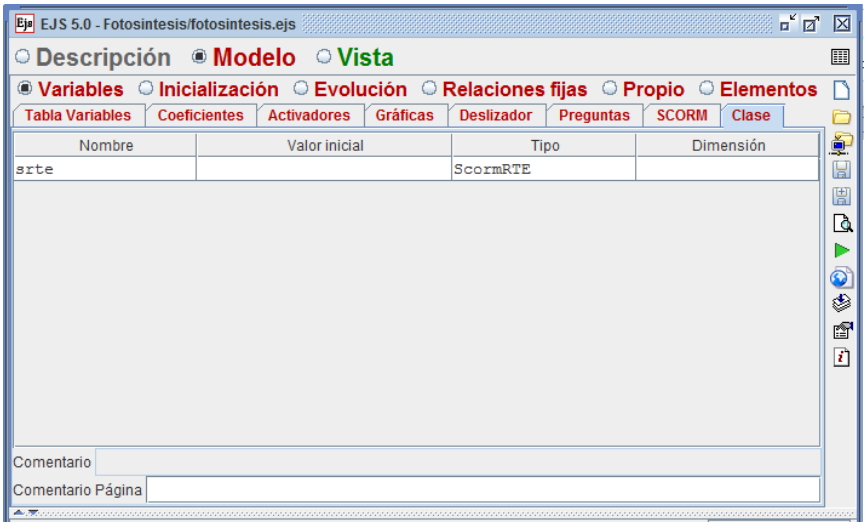


Figura 8-41 Clase

En la parte correspondiente a Inicialización está todo lo necesario para poder realizar la comunicación con SCORM y LMS:

```
_pause();
if (_isApplet())
{
  verScorm=true;
  estado="Es Applet";
  srte = new ScormRTE (_getApplet());
  dateTime=srte.fechaHoraScorm();
  version=srte.version;
  learnerName=srte.learnerName;
  learnerId=srte.learnerId;
  scoreRaw=srte.rteGetScoreRaw();
  mode=srte.rteGetMode();
  if ((scoreRaw > 0)&& (scoreRaw < 31))
  {
    _view.alert("Ventana","Puntuación previa","Puntuación de última
ejecución: " + scoreRaw + "\nSi  trabajas de nuevo en el laboratorio, esta
puntuación se borrará");
  }
  else
  {
    scoreRaw=srte.rteSetScoreRaw(puntos);
  }
}
else
{
  verScorm=false;
  estado="No es Applet";
}
```

Código 8-7 Inicialización

El apartado de Evolución está compuesto de cuatro pestañas independientes que se encargan de realizar tareas importantes: El cambio de color del deslizador en función de su posición, la evolución de las gráficas en función del deslizador, la traza que se grafica en función de la planta seleccionada y el texto que se muestra en función de la pregunta que sea.

Se comienza mostrando el cambio de color del deslizador:

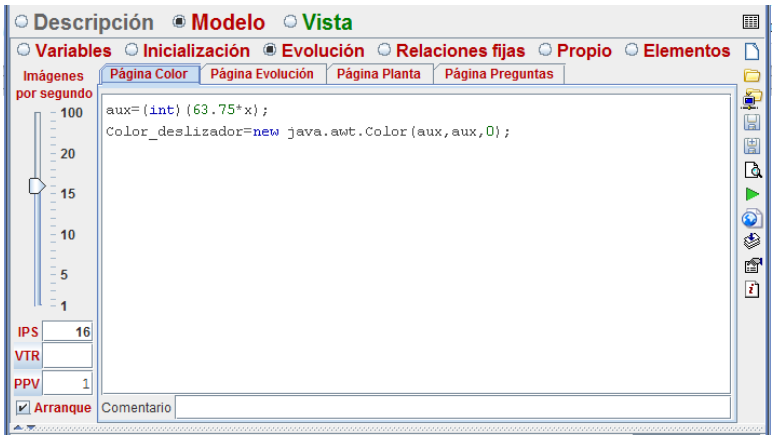


Figura 8-42 Página Color

En la pestaña siguiente se encuentra recogido el proceso cíclico que experimentan las gráficas. Básicamente se calcula la posición anterior de deslizador y se compara con la actual para pintar o borrar las gráficas seleccionadas.

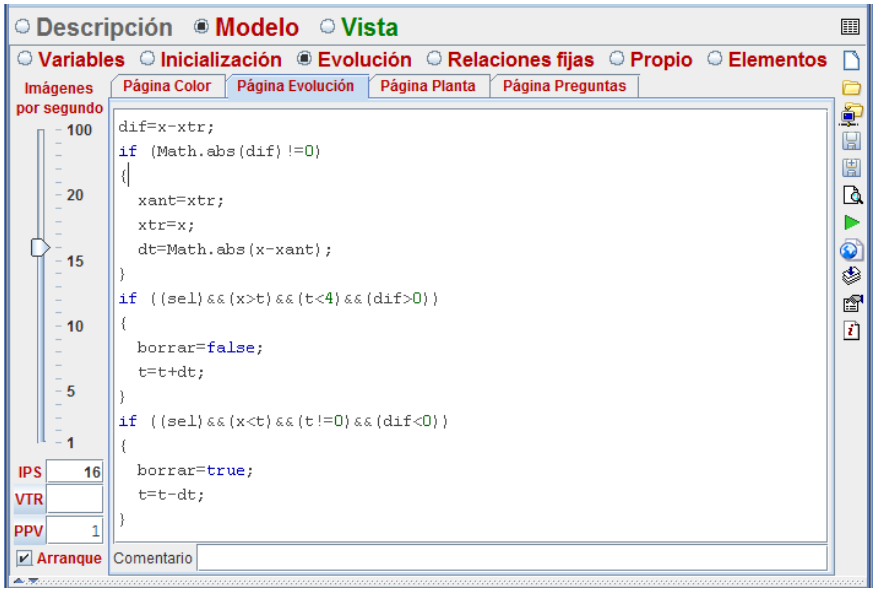


Figura 8-43 Página Evolución

En la pestaña llamada Página Planta están los valores de los coeficientes previamente calculados en Excel en función de la planta seleccionada así como la ecuación que responde a la evolución de la eficiencia fotosintética en función de la intensidad lumínica.

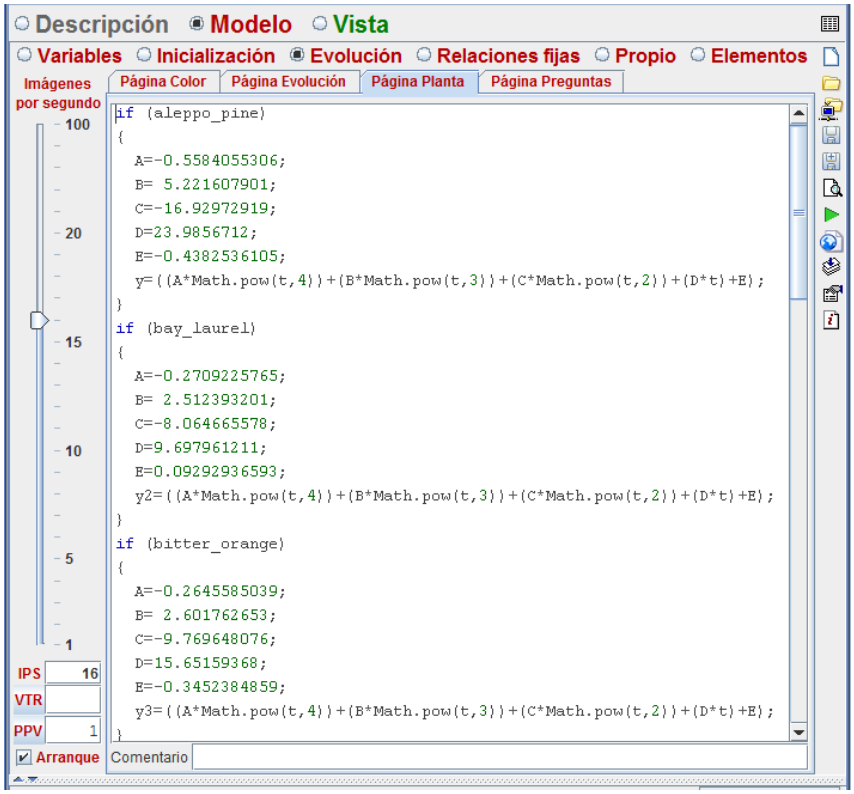


Figura 8-44 Página Planta (1º parte)

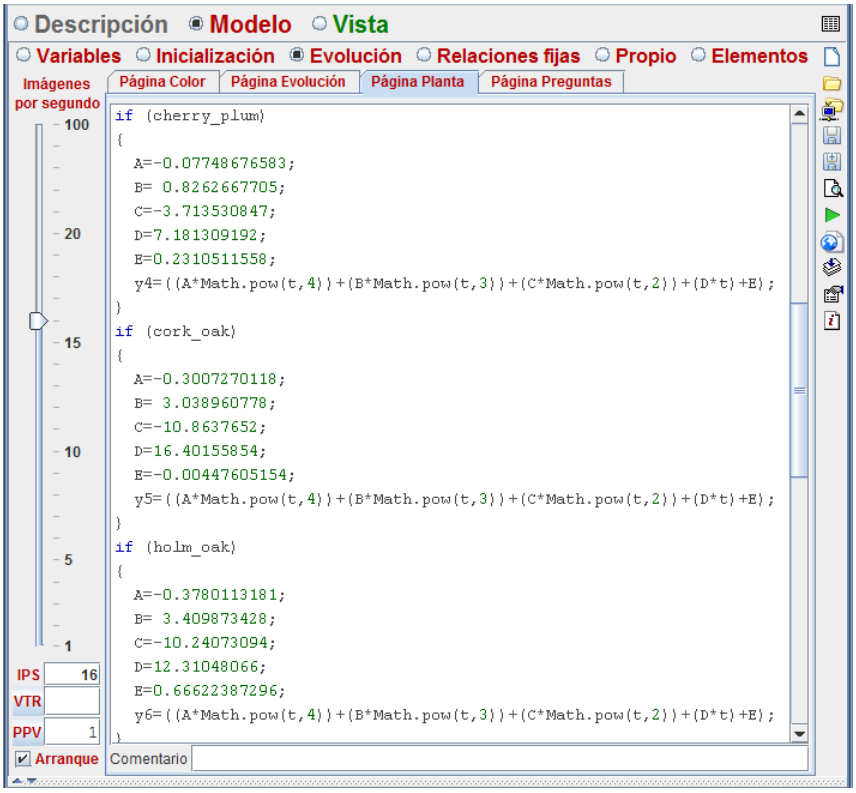


Figura 8-45 Página Planta (2º parte)

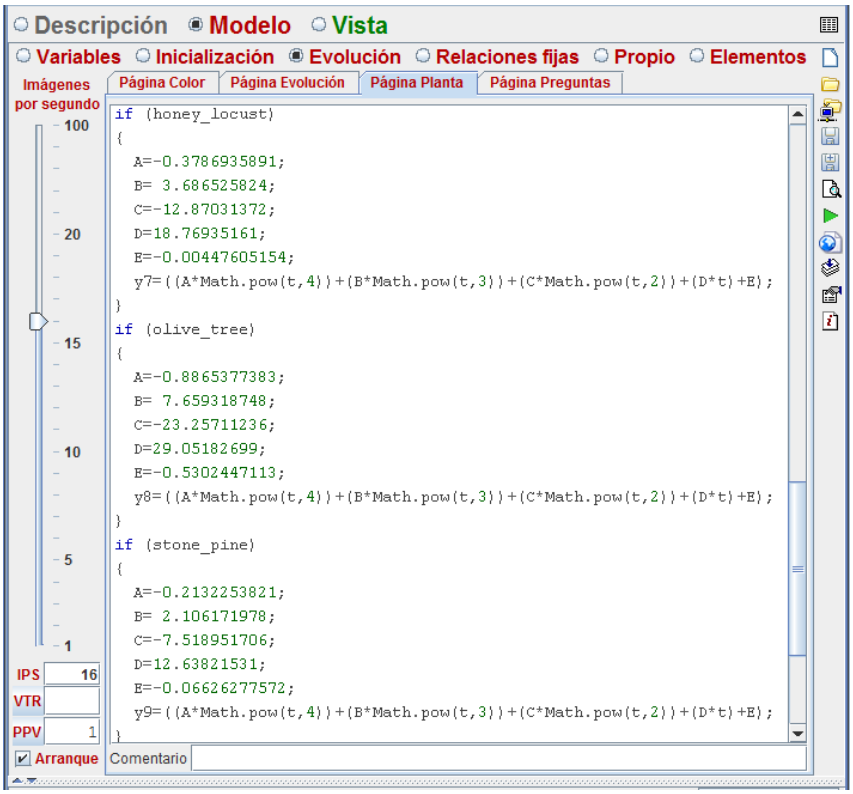


Figura 8-46 Página Planta (3º parte)

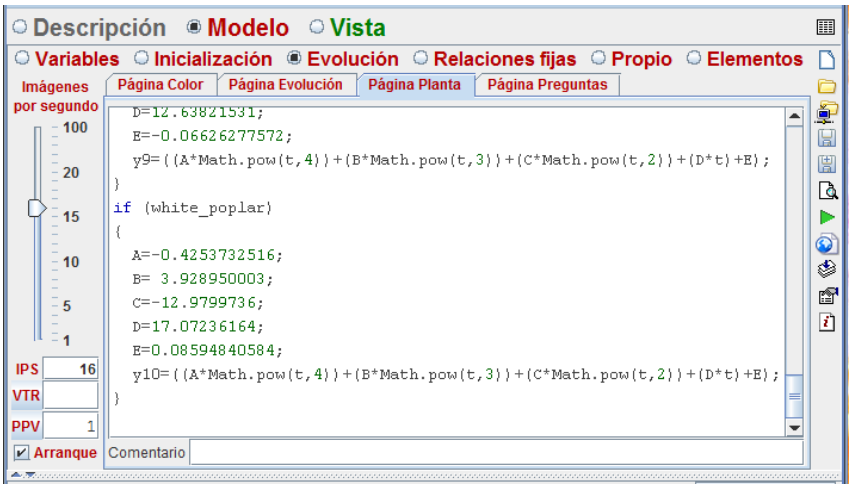


Figura 8-47 Página Planta (4º parte)

Por último se describe el apartado destinado a generar las preguntas que se hacen en este laboratorio virtual.

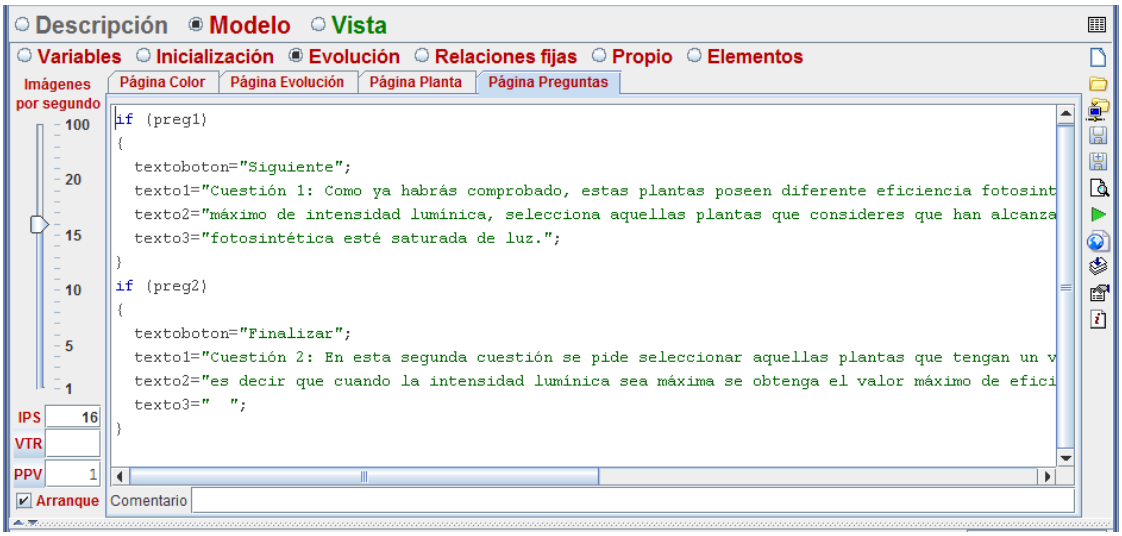


Figura 8-48 Página Preguntas

A continuación se reazlia la interfaz gracias a la sección *Vista*. Tras haber creado una interfaz apropiada para la ocasión, la sección *Vista* quedaría de la siguiente manera:

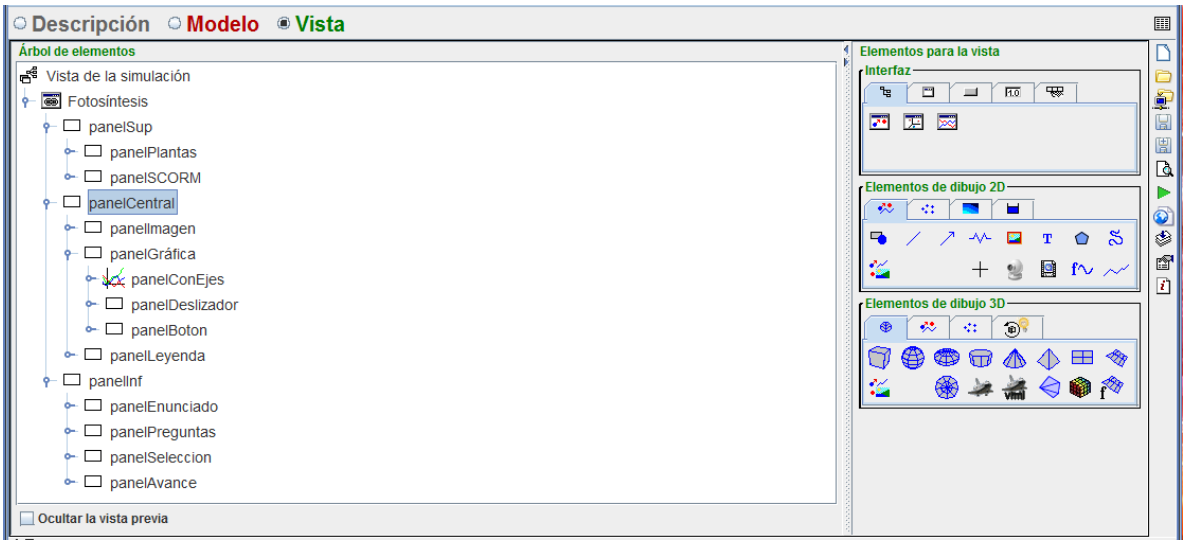


Figura 8-49 Sección Vista

Una vez ya desarrollado se pueden explicar más detenidamente aquellos elementos que resulten más interesantes.

A continuación se comentan los botones principales que se sitúan en la parte superior. El selector asociado a cada planta permite marcar (o desmarcar) la visualización de su gráfica. El código que conlleva es muy sencillo:

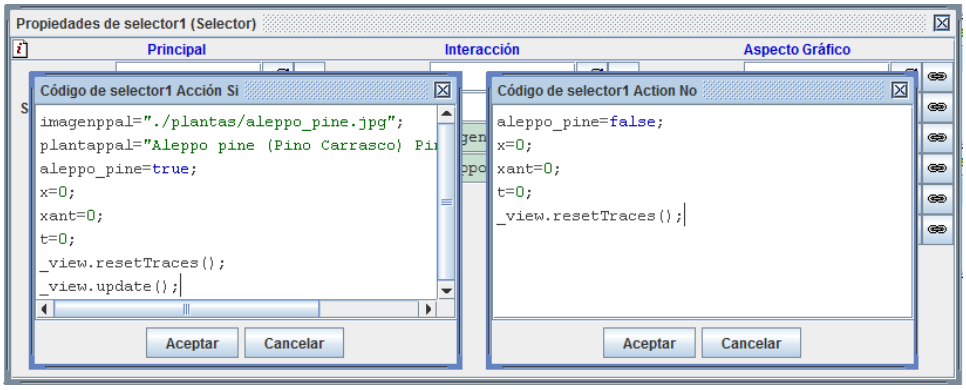


Figura 8-50 Selector Planta

Además de seleccionar la planta con el método indicado anteriormente, también se puede pulsar en la imagen pequeña que hace referencia a la planta para conseguir el mismo objetivo que con el selector. El código es bastante parecido al anterior pero presenta unas pequeñas diferencias:

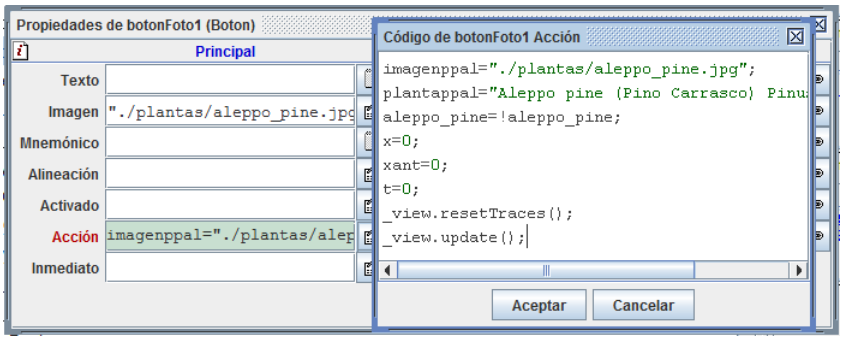


Figura 8-51 Propiedades del botón Foto

En cuanto a la parte central de la interfaz es importante destacar la presencia de una imagen aumentada de la última planta con la que se ha interactuado y cuyas propiedades son:

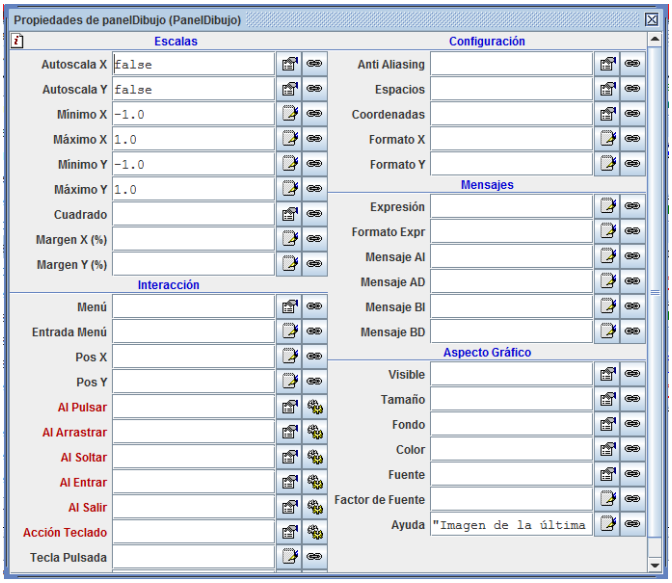


Figura 8-52 Propiedades de Imagen

También cabe destacar la presencia de un deslizador (intensidad lumínica) que cambia de color en función de la posición en la que se encuentre. Las propiedades de este deslizador son las siguientes:

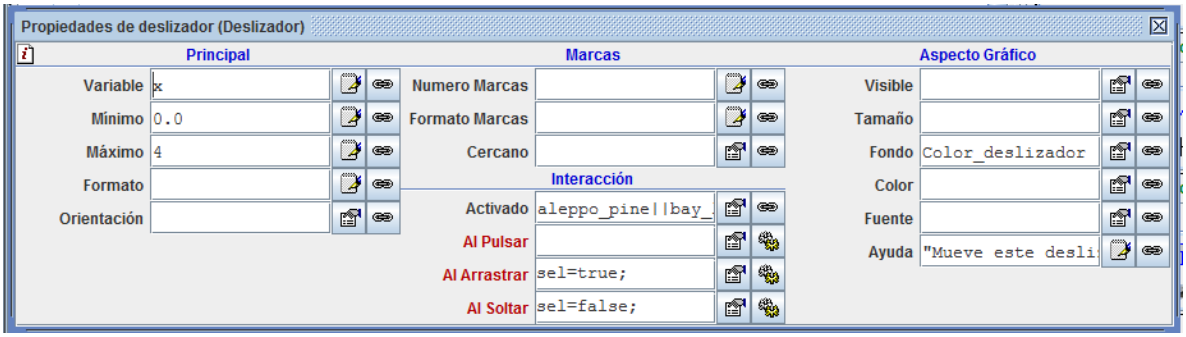


Figura 8-53 Propiedades del deslizador

Justo debajo de este deslizador se encuentra el botón *Reiniciar* que se encarga de reestablecer los parámetros a su estado original para comenzar de nuevo con la simulación. Su código es el siguiente:

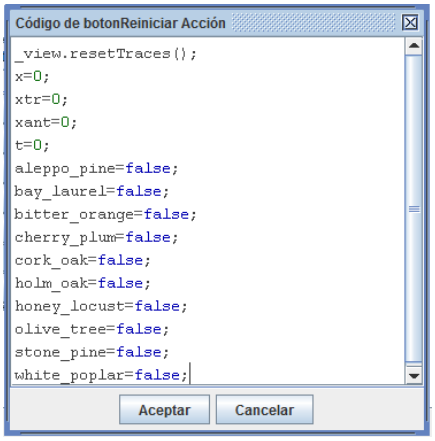


Figura 8-54 Propiedades del botón Reiniciar

La parte más importante de este parte central es una gráfica que muestra las distintas trazas de las plantas (eficiencia fotosintética) en función del valor de intensidad lumínica deseado (controlado por el deslizador).

Por último, para facilitar el reconocimiento de las distintas trazas se incluye una leyenda con la información necesaria para comprender el gráfico.

Como anteriormente se ha mencionado, hay un apartado de preguntas (panel inferior) que los alumnos deberán contestar para completar esta simulación.

La codificación las preguntas y respuestas es bastante sencilla ya que solo es añadir etiquetas de texto. Lo verdaderamente importante es el botón *Avance* que puntúa la pregunta en función de las respuestas marcadas. Es importante mostrar la codificación asociada a este botón ya que es el más complejo que presenta la interfaz:

```
cont=cont+1;
resp="#" + cont/2;
if (preg1)
{
  if (aleppopine)
  {
    puntos1=puntos1-1;
  }
  if (!aleppopine)
  {
    puntos1=puntos1+1;
  }
  if (baylaurel)
  {
```

```
    puntos1=puntos1+1;
  }
  if (!baylaurel)
  {
    puntos1=puntos1-1;
  }
  if (bitterorange)
  {
    puntos1=puntos1+1;
  }
  if (!bitterorange)
  {
    puntos1=puntos1-1;
  }
  if (cherryplum)
  {
    puntos1=puntos1+1;
  }
  if (!cherryplum)
  {
    puntos1=puntos1-1;
  }
  if (corkoak)
  {
    puntos1=puntos1-1;
  }
  if (!corkoak)
  {
    puntos1=puntos1+1;
  }
  if (holmoak)
  {
    puntos1=puntos1-1;
  }
  if (!holmoak)
  {
    puntos1=puntos1+1;
  }
  if (honeylocust)
  {
    puntos1=puntos1+1;
  }
  if (!honeylocust)
  {
    puntos1=puntos1-1;
  }
  if (olivetree)
  {
    puntos1=puntos1-1;
  }
  if (!olivetree)
  {
    puntos1=puntos1+1;
  }
  if (stonepine)
  {
    puntos1=puntos1-1;
  }
  if (!stonepine)
  {
    puntos1=puntos1+1;
```

```
  }
  if (whitepoplar)
  {
    puntos1=puntos1+1;
  }
  if (!whitepoplar)
  {
    puntos1=puntos1-1;
  }
  aleppopine=false;
  baylaurel=false;
  bitterorange=false;
  cherryplum=false;
  corkoak=false;
  holmoak=false;
  honeylocust=false;
  olivetree=false;
  stonepine=false;
  whitepoplar=false;
  if (puntos1>=5)
  {
    resp= resp + ":" + "- Cuestión 1 OK ("+puntos1+"/10)";
    _view.alert("Ventana","Pregunta superada", resp + "\nHas completado
correctamente esta pregunta.\nContinúa ahora con la siguiente cuestión");
    preg2=true;
    preg1=false;
  }
  else
  {
    resp= resp + ":" + "- Cuestión 1 no OK ("+puntos1+"/10)";
    _view.alert("Ventana","Pregunta no superada", resp + "\nNo has
contestado correctamente.\nDeberás volver a intentarlo");
    puntos1=0;
  }
}
else
{
  if (aleppopine)
  {
    puntos2=puntos2-1;
  }
  if (!aleppopine)
  {
    puntos2=puntos2+1;
  }
  if (baylaurel)
  {
    puntos2=puntos2-1;
  }
  if (!baylaurel)
  {
    puntos2=puntos2+1;
  }
  if (bitterorange)
  {
    puntos2=puntos2-1;
  }
  if (!bitterorange)
  {
    puntos2=puntos2+1;
  }
```

```
  if (cherryplum)
  {
    puntos2=puntos2-1;
  }
  if (!cherryplum)
  {
    puntos2=puntos2+1;
  }
  if (corkoak)
  {
    puntos2=puntos2+1;
  }
  if (!corkoak)
  {
    puntos2=puntos2-1;
  }
  if (holmoak)
  {
    puntos2=puntos2+1;
  }
  if (!holmoak)
  {
    puntos2=puntos2-1;
  }
  if (honeylocust)
  {
    puntos2=puntos2-1;
  }
  if (!honeylocust)
  {
    puntos2=puntos2+1;
  }
  if (olivetree)
  {
    puntos2=puntos2-1;
  }
  if (!olivetree)
  {
    puntos2=puntos2+1;
  }
  if (stonepine)
  {
    puntos2=puntos2-1;
  }
  if (!stonepine)
  {
    puntos2=puntos2+1;
  }
  if (whitepoplar)
  {
    puntos2=puntos2-1;
  }
  if (!whitepoplar)
  {
    puntos2=puntos2+1;
  }
  preg2=true;
  aleppopine=false;
  baylaurel=false;
  bitterorange=false;
  cherryplum=false;
```

```
  corkoak=false;
  holmoak=false;
  honeylocust=false;
  olivetree=false;
  stonepine=false;
  whitepoplar=false;
  puntos=puntos1+puntos2;
  if (puntos2>=5)
  {
    resp= resp + ":" + "- Cuestión 2 OK ("+puntos2+"/10)";
    _view.alert("Ventana","Pregunta superada", resp + "\nHas completado
correctamente esta pregunta.\nEnhorabuena has contestado correctamente las
dos preguntas");
  }
  else
  {
    resp= resp + ":" + "- Cuestión 2 no OK ("+puntos2+"/10)";
    _view.alert("Ventana","Pregunta no superada", resp + "\nNo has
contestado correctamente.\nDeberás volver a intentarlo");
  }
  if ((puntos1>=5)&&(puntos2>=5))
   {
     pOK=true;
   }
   else
   {
     resp="Lab Virtual no OK ("+puntos+"/20)";
     _view.alert("Ventana","No superado", resp + "\nNo has contestado
correctamente a las dos preguntas planteadas.\nPuedes volver a intentarlo
de nuevo");
     pOK=false;
     puntos1=0;
     puntos2=0;
     preg1=true;
     preg2=false;
   }
   if ((cont>=6)&&(!pOK))
   {
     _view.alert("Ventana","Excedido", resp + "\nHas superado el número de
intentos posibles.\nDebes volver a empezar el laboratorio virtual desde el
principio");
     cont=0;
     puntos=0;
     preg1=true;
     preg2=false;
   }
}
if (verScorm&&pOK)
{
 pOK=false;
 resp="Lab Virtual OK ("+puntos+"/20)";
 _view.alert("Ventana","Superado", resp + "\nHas completado correctamente
este laboratorio virtual.\nContinúa ahora con el siguiente apartado:
Laboratorio Remoto");
 completionStatus="completed";
 successStatus="passed";
 completionStatus=srte.rteSetCompletionStatus(completionStatus);
 successStatus=srte.rteSetSuccessStatus(successStatus);
 scoreRaw=srte.rteSetScoreRaw(puntos);
 }
if (pOK)
```

```
{
 pOK=false;
 resp="Lab Virtual OK ("+puntos+"/20)";
 _view.alert("Ventana","Superado", resp + "\nHas completado correctamente
este laboratorio virtual.\nContinúa ahora con el siguiente apartado:
Laboratorio Remoto");
}
```

Código 8-8 Botón Avance

9 -MÓDULO DE APRENDIZAJE SCORM

El contenido SCORM puede ser proporcionado a los estudiantes mediante cualquier Sistema de Gestión del Aprendizaje (Learning Management System (LMS)) compatible con SCORM que use la misma versión. En este caso será proporcionado a través de la plataforma ILIAS. Para realizar la importación de dicho módulo simplemente acceder a ILIAS, pulsar sobre "Añadir nuevo elemento" y seleccionar "Módulo de aprendizaje SCORM/AICC". Finalmente seleccionar importar nuevo paquete de tipo SCORM 2004.

A través de este módulo el alumno será guiado a través de un conjunto de páginas donde encontrará contenidos teóricos, varios tests y la ejecución de los applets desarrollados en este Trabajo Final de Grado (laboratorio virtual y laboratorio remoto).

Se han desarrollado un conjunto de 5 páginas que se ejecutarán progresivamente. La interfaz de dicho módulo se podrá ver adjunta en el anexo. El código programado para generar dichas páginas no se encuentra incluido en esta memoria debido a su amplia extensión por lo que se adjunta en un fichero .zip.

La estructura de este módulo de aprendizaje es la mostrada en la siguiente figura:

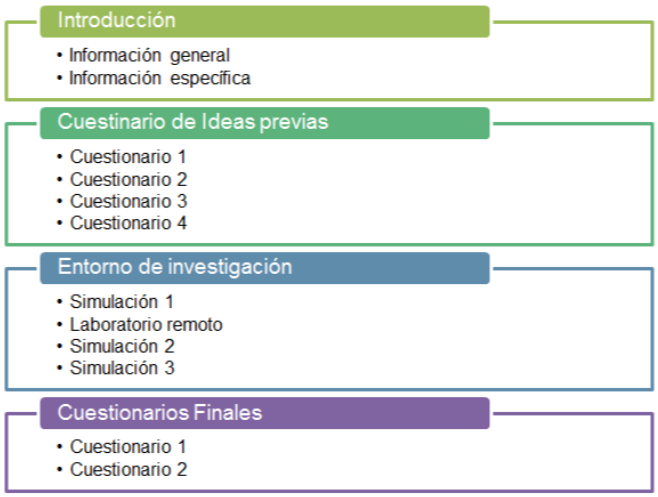


Figura 9-1 Estructura módulo de aprendizaje SCORM

A continuación se resumen el contenido de cada una de las cinco páginas:

- Introducción: Proporciona información general sobre el WebLab y se enmarca dentro de los estudios de grado.

- Cuestionario de ideas previas: Una serie de cuestionarios que los alumnos deben contestar para reflejar sus conocimientos antes de la realización de este WebLab.

- Entorno de investigación: (Para acceder a este sección es necesario haber superado los cuestionarios de la sección Cuestionarios de ideas previas) Proporciona los guiones de las prácticas que se deben realizar con la simulación (Virtual Lab) y el laboratorio remoto (Remote Lab) en los que se podrá trabajar con el fin de superar las prácticas propuestas. Tanto la simulación como el laboratorio remoto realizan una evaluación automática del trabajo realizado por cada alumno.

- Cuestionarios de ideas finales: (Para acceder a este sección es necesario haber superado la evaluación automática de la sección Entrono de investigación) Es exactamente igual que los cuestionarios del bloque anterior y su función es comparar las respuestas de uno y otro para comprobar el efecto del bloque Entorno de investigación.

- Teoría (Última página): (Para acceder a este sección es necesario haber contestado los cuestionarios de la página anterior) Muestra la teoría relacionada con el WebLab para que los alumnos conozcan la ciencia erudita y corrijan o comprueben sus ideas previas.

10 CONCLUSIONES Y TRABAJO FUTURO

El resultado del proceso de trabajo de este proyecto ha desembocado en la generación de los módulos software, hardware y documentación necesarios para desarrollar un nuevo WebLab híbrido (virtual + remoto) basados en principios físicos. Sin embargo, como parte de la continua investigación en laboratorios remotos aplicados a la educación queda mucho que mejorar todavía. Estos capítulos pretenden ofrecer una visión general de los objetivos que se han logrado, y el trabajo futuro de investigación y desarrollo que queda aún por hacer.

10.1 Resultado del proyecto

Repasando los objetivos técnicos, sociales y económicos del proyecto, tal como se describe en los primeros apartados de este trabajo final de grado, se podría decir que la realización del proyecto ha cumplido con las expectativas de manera exitosa. Se han cumplido los objetivos detallados para este proyecto a través del uso de diferentes herramientas y tipos de programas conocidos a lo largo de la realización de los estudios de esta titulación. De hecho, se ha implementado un WebLab híbrido para la experimentación con el modelado y control de parámetros sobre un invernadero.

Con la construcción de un invernadero a escala, se ha obtenido un laboratorio remoto, muy útil a efectos docentes, ya que además de haber sido útil para este proyecto, servirá para trabajar en otros o realizar diferentes experimentos o lecturas de diferentes tipos de sensores y actuadores. Para la consecución de estos objetivos el desarrollo del proyecto ha incluido ciertos módulos que pueden ser reutilizados en el futuro desarrollo de nuevos laboratorios remotos:

- Una aplicación de comunicaciones (srav.jar y servidor.jar) entre los diferentes subsistemas de un laboratorio remoto adaptable a las nuevas exigencias de futuros laboratorios remotos.Se trata de una herramienta muy útil y versátil para el trabajo con bases de datos: accesos, lecturas, modificaciones y otras acciones.

- Una base de datos que permite registrar todas las variables de los sensores aún y cuando el cliente no esté conectado.

- Un cliente EJS sencillo y reutilizable que permite la comunicación entre el laboratorio y los usuarios.

- Un laboratorio virtual que simula la evolución de la eficiencia fotosintética para 10 tipos de plantas en función de la cantidad de luz que reciban.

- Finalmente un módulo de aprendizaje SCORM en el que se insertan todos los demás applets.

El diseño del laboratorio, así como la arquitectura asociada, se han mantenido simples utilizando dispositivos hardware compatibles con Arduino que facilitan el desarrollo y que por lo general han sido validados por otros usuario de la plataforma Arduino. El proyecto también ha tratado de hacer uso de las tecnologías apropiadas y únicamente de los recursos de software necesarios.

El proyecto en sí ha sido considerablemente multidisciplinar, ya que abarca aspectos tales como las tecnologías de las comunicaciones, electrónica, gestión de base de datos mySQL, mecatrónica, diseño de software, programación java, tecnologías web y desarrollo web. Por otra parte, no se ha limitado al ámbito de la ingeniería, dado que el autor también ha llevado a cabo un trabajo relacionado con la preparación de documentos, el diseño de interfaces de usuario que mejoren la experiencia del usuario, etc.

10.2 Trabajo futuro.

Una vez que la estructura básica y la funcionalidad del experimento se han definido e implementado, se ha establecido la base para la mejora y la investigación futura de un nuevo WebLab híbrido. Los siguientes pasos de este proyecto se refieren a la ampliación de su funcionalidad, mejoras generales en la usabilidad, y su uso como punto de partida para la investigación pedagógica a mayor escala. En este sentido, el futuro trabajo más relevante se puede clasificar en cuatro áreas:

- Integración completa del laboratorio en otras plataformas LMS.
- Mejoras en la interfaz gráfica de usuario.
- Mejora en las comunicación del Arduino con el PC.
- Diseño de una nueva versión del laboratorio.

Cabe señalar que todas estas tareas persiguen los propósitos principales de aumentar la facilidad de uso y capacidad de aprendizaje desde la perspectiva del usuario, así como llegar a ofrecer una cantidad mayor de posibilidades educacionales.

10.2.1 Integración completa.

Otra línea de futuro interesante la constituye el estudio de las posibilidades de integración del software cliente EJS en otras plataformas LMS (Learning Management System) diferente a ILIAS.

10.2.2 Mejoras en la interfaz gráfica de usuario.

La actual interfaz gráfica de usuario desarrollada como parte del cliente es funcional y ofrece acceso a los principales controles necesarios para gestionar y controlar el laboratorio. Sin embargo, podría beneficiarse de algunas mejoras desde el punto de vista del diseño, ya que actualmente no es lo bastante intuitiva..

10.2.3 Mejora en la comunicación del Arduino con el PC.

Un aspecto muy a tener en cuenta sería trabajar con otro tipo de placa Arduino, ya que se podrían mejorar mucho las comunicaciones. Existen placas Arduino que se comunican vía Wifi (Arduino Yun), Bluetooth (Arduino Bluetooth) y Ethernet (Arduino Ethernet), con las que se podría diseñar sistemas de comunicación basados en la arquitectura cliente-servidor.

10.2.4 Diseño de una nueva versión del laboratorio

También sería interesante la incorporación de nuevos sensores y actuadores pues cada día surgen nuevos tipos de los mismos, que permitirían afinar más en la configuración del sistema desarrollado en este proyecto. Algunas ideas serían:

- Sensor analógico de pH para el agua.
- Sensor de nivel de agua de tipo escalonado.
- Caudalímetro digital.
- Iluminación artificial variable

- Posibilidad de ventilación natural no forzada.
- Electroválvulas para seleccionar diferentes métodos de riego.
- Colorímetro.
- Permitir la interacción con la cámara IP.
- Almacenaje de datos en tarjeta SD.

11 PLIEGO DE CONDICIONES TÉCNICAS

En este apartado se van a detallar las características hardware y software para que el presente TFG actúe correctamente partiendo de los equipos utilizados para su desarrollo y funcionamiento.

11.1 Hardware

El presente trabajo necesita las siguientes características hardware:

- Ordenador servidor equipado con tarjeta de red.
- Ordenador cliente equipado con tarjeta de red.
- Router multipuerto.
- Cámara IP.
- Microcontrolador Arduino Mega 2560.
- Cable USB.
- Fuente de alimentación 12V.
- Bomba de agua 12V.
- Ventilador 12V.
- Fuente de luz artificial 220V.
- Sensor humedad suelo HL-69.
- Pareja de sensores de temperatura y humedad DHT11.
- Sensor de luz (LDR).
- Sensor de líquidos.
- Conjunto de relés.

11.2 Software

El requerimiento software de los PC's utilizados en el desarrolo de este trabajo son:

- Windows 8.
- Suite ofimática Microsoft Office.
- Herramienta EJS.
- IDE de desarrollo NetBeans para programación Java.
- IDE de desarrollo Arduino para programar el microcontrolador.
- Administrador de base de datos mySQL.
- Editor de código Notepad++ para programación SCORM.
- Fritzing.
- Administrados LMS.

12 PLIEGO DE CONDICIONES LEGALES

Se recomienda el uso de programas con licencia original o no violar los derechos de autor que cada uno de los desarrolladores de software han establecido en las condiciones del contrato de uso orientado al usuario final.

13 PRESUPUESTO

En este capítulo se describen los costes económicos referidos a la elaboración de este Trabajo Final De Grado.

13.1 Coste Hardware

PRESUPUESTO Y MEDICIONES

Invernadero controlado por Arduino

CÓDIGO	RESUMEN	UDS	LONGITUD	ANCHURA	ALTURA	PARCIALES	CANTIDAD	PRECIO	IMPORTE
	CAPÍTULO C01 Hardware								
1.1	**Invernadero**								
	Estructura de metacrilato con unas dimensiones de 94x49x38cm.								
							1,00	250,00	250,00
1.2	**PC de sobremesa (servidor)**								
	Ordenador de sobremesa (Modelo: AWRDACPI) con un procesaro Intel Pentium 3.2GHz.								
							1,00	850,00	850,00
1.3	**Arduino Mega**								
	Arduinio ATMega2560 cuyo modelo es A000067.								
							1,00	29,90	29,90
1.4	**Display LCD**								
	Módulo de ampliación A000096. Pantalla LCD propia de Arduino.								
							1,00	20,80	20,80
1.5	**Sensor de temperatura y humedad**								
	Módulo DHT11 para Arduino. Capaz de detectar tanto temperatura como humedad relativa.								
							2,00	3,68	7,36
1.6	**Cables con terminal hembra**								
	Conjunto de 10 cables de 200mm.								
							2,00	3,19	6,38
1.7	**Cables con terminal macho**								
	Conjunto de 10 cables de 200mm.								
							2,00	3,19	6,38
1.8	**Ventilador fricción 12V**								
	Ventilador de dimesiones 45x45x10mm que funciona a 12V DC.								
							1,00	9,61	9,61
1.9	**Bomba de agua**								
	Bomba de agua C6000 que funciona 12V y que posee un caudal de 8L/min.								
							1,00	19,50	19,50
1.10	**Sensor de nivel de liquidos**								
	Sensor de nivel C7235 para liquidos. Requiere montaje vertical.								
							1,00	11,35	11,35
1.11	**Fuente aimentación 12V**								
	Fuente de alimentación de 12V, 24W y 2A (ALU1224) con clavija europea.								
							1,00	8,60	8,60
1.12	**Cable USB tipo A/B**								
	Cable USB para la conexión de Arduino con el PC servidor.								
							1,00	3,75	3,75
1.13	**Módulo sensor luminosidad**								
	Módulo compuesto por una LDR.								
							1,00	4,25	4,25
1.14	**Cámara IP**								
	Cámara Conceptronic cuyo modelo es CIPCAMPTIWL.								
							1,00	59,20	59,20

PRESUPUESTO Y MEDICIONES

Invernadero controlado por Arduino

CÓDIGO	RESUMEN	UDS	LONGITUD	ANCHURA	ALTURA	PARCIALES	CANTIDAD	PRECIO	IMPORTE
1.15	**Caja de plástico para Arduino**								
	Caja Hammond de plástico especial para Arduino.								
							1,00	14,25	14,25
1.16	**Relé para Arduino**								
	Relé especial para trabajar con Arduino.								
							2,00	6,00	12,00
1.17	**Sensor de oxígeno**								
	Sensor de oxígeno SEN00500P especial para Arduino.								
							1,00	137,98	137,98
1.18	**Sensor de dióxido de carbono**								
	Sensor de dióxido de carbono MG-811 apropiado para Arduino.								
							1,00	70,00	70,00
	TOTAL CAPÍTULO C01 Hardware.....								**1.521,31**

13.2 Coste Software

PRESUPUESTO Y MEDICIONES

Invernadero controlado por Arduino

CÓDIGO	RESUMEN	UDS	LONGITUD	ANCHURA	ALTURA	PARCIALES	CANTIDAD	PRECIO	IMPORTE
	CAPÍTULO C02 Software								
2.1	**Sistema operativo (servidor)**								
	En este caso se trata de Windows 7 Enterprise. Licencia de estudiante.								
							1,00	0,0	0,0
2.2	**Sistema operativo (cliente)**								
	En este caso se trata de Windows 7. Licencia de estudiante.								
							1,00	0,0	0,0
2.3	**JAVA**								
	Programa necesario para trabajar con el resto del software.								
							1,00	0,0	0,0
2.4	**IDE Arduino**								
	Programa necesario para poder operar con el microcontrolador Arduino.								
							1,00	0,0	0,0
2.5	**Fritzing**								
	Programa útil para realizar dibujos del conexionado de Arduino.								
							1,00	0,0	0,0
2.6	**IDE Netbeans**								
	Programa esencial para el desarrollo del servidor.								
							1,00	0,0	0,0
2.7	**XAMPP**								
	Programa que controla la base de datos MySQL.								
							1,00	0,0	0,0
2.8	**EJS**								
	Programa esencial para la creación de la interfaz con la que trabaja el cliente.								
							1,00	0,0	0,0
	TOTAL CAPÍTULO C02 Software								0,0

13.3 Coste Total

El precio total del presupuesto asciende a MIL QUINIENTOS VEINTIÚN EUROS CON TREINTA Y NUEVE CÉNTIMOS

14 BIBLIOGRAFÍA, WEBGRAFÍA Y REFERENCIAS

[1] Proyecto COMPASS. Disponible en http://www.compass-project.eu/

[2] Sznajdleder, P. A. (2013). Java a Fondo . Ed. Alfaomega

[3] Sikora, Zbigniew M. Java: practical guide for programmers.

[4] Schildt, Herbert; Coward, Danny. Java: the complete reference.

[5] Página principal de Java. Disponible en: https://www.java.com/es/

[6] Donahoo, Michael J; Speegle, Gregory D SQL: practical guide for developers.

[7] Tutorial sobre SQL. Disponible en: http://www.1keydata.com/es/sql/

[8] Easy Java Simulations Wiki. Disponible en http://www.um.es/fem/EjsWiki/

[9] Blog para programación con Arduino. Disponible en http://panamahitek.com/

[10] Canal Youtube con tutoriales de programación SQL. Disponible en https://www.youtube.com/channel/UCYetl3Yx9QP5MyRDTpmluDA/videos

[11] Web oficial Arduino. Disponible en http://www.arduino.cc/

[12] Web del proveedor Ariston .Disponible en: http://www.ariston.es/

[13] WebLab-Deusto. Disponible en: http://weblab.deusto.es/website/

[14] Oracle Help Center. Disponible en: http://docs.oracle.com/en/

[15] Scorm Standard. Disponible en http://scorm.com/scorm-explained/

[16] Gomes, L., Bogosyan, S. "Current Trends in Remote Laboratories", IEEE Transactions on Industrial Electronics. Vo. 56, Issue 12, pp. 4744-4756. ISSN: 0278-0046. Diciembre 2009.

[17] Corter, J.E., Nickerson, J.V.,Esche, S.K.,Chassapis, C. "Remote versus handson labs: a comparative study" in Proceedings of 34th Annual Frontiers in Education Conference, F1G - 17-21 Vol. 2, ISSN: 0190-5848, 2004.

[18] Müller, D., Erbe, H.H. " Collaborative remote laboratories in engineering education: Callenges and visions", in Advances on remote laboratories and elearningexperiences, Ed. by L. Gomes and J. Garcia Zubia, University of Deusto, Bilbao, 2007, pp. 35-59. ISBN: 978-84-9830-077-2

[19] Åkesson, H., Håkansson, L., Gustavsson, I., Zackrisson, J., Claesson, I., Lagö, T. "Vibration Analysis of Mechanical Structures over the Internet Integrated into Engineering Education", Proceedings of the 2006 ASEE Annual Conference, Chicago, USA, June 18 - 21, 2006.

[20] del Alamo, J. A. "iLabs: Performing Laboratory Experiments Across Continents," in Learning International Networks Consortium Workshop(LINC), MIT, March 24-25, 2004.

[21] Dormido, S., Sánchez, J., Morilla. F. “Laboratorios virtuales y remotos para la práctica a distancia de la Automática”, Proceedings del Congreso Online Educa Madrid, Madrid, Junio 2000.

[22] Gustavsson, I., Zackrisson, J., Håkansson, L., Claesson,I., Lagö. T.L., “The VISIR project – an Open Source Software Initiative for Distributed Online Laboratories”, in Proceedings of Remote Engineering & Virtual Instrumentation Conference (REV), Porto, Portugal, June 25 – 27, 2007.

[23] Casini, M., Prattichizzo, D., Vicino, A. "The automatic control telelab: userfriendly interface for distance learning", IEEE Transactions on in Education, Vol. 46, Issue 2, pp. 252. ISSN: 0018-9359. May 2003.

[24] Keeley, P. (2011). Chlorophyll. En C. Reinburg, J. Horak, A. Cooke & J. Cusick (Eds.) U*ncovering Students Ideas in Life Science. 25 New formative assessment Probes Vol 1 (p*p 51-56). USA: NSTA Press. National Science Teacher.

[25] Keeley, P. (2011). Light and Dark. En C. Reinburg, J. Horak, A. Cooke & J. Cusick (Eds.) U*ncovering Students Ideas in Life Science. 25 New formative assessment Probes Vol 1 (p*p 63-67). USA: NSTA Press. National Science Teacher Association.

[26] Keeley, P. , Eberle, F. and Dorsey, C. (2008). Respiration. En Keeley P. Uncovering Students Ideas in Science. 25 New formative assessment Probes (pp 131-139) USA: NSTA Press. National Science Teacher Association.

15 ANEXOS

15.1 Anexo I: SCORM

15.1.1 Teoría

En este apartado se engloba la parte más importante del SCORM, la teoría relacionada con los temas de fotosíntesis, clorofila, respiración y demás aspectos importantes relacionados con la materia. Se encuentra dividida en tres grandes bloques como se podrá ver a continuación:

15.1.1.1 Clorofila

Propósito

El propósito de esta sonda de evaluación es provocar ideas a los estudiantes sobre una palabra científica que encuentran frecuentemente en el instituto, *clorofila*. La prueba está diseñada para revelar si os estudiantes saben que la clorofila es más que un pigmento. Sus respuestas revelarán si también reconocen que la principal función de la clorofila es absorber energía lumínica para la fotosíntesis.

Explicación

Las dos respuesta correctas son las opciones E y I. La clorofila es un pigmento ubicado en los cloroplastos de las células vegetales (al igual que muchos organismos unicelulares). La función primaria de la clorofila es absorber energía lumínica (opción E). Los cloroplastos convierten la energía lumínica en energía química usando la clorofila. La clorofila además confiere a la mayoría de las plantas el color verde característico (opción I), aunque esta es una función pasiva ya que la molécula de clorofila simplemente refleja la luz verde más que absorberla y eso hace que las plantas parezcan verdes a nuestros ojos. Sin embargo, su

objetivo principal no es proporcionar color a las plantas sino más bien actuar como un fotorreceptor.

La clorofila es sólo una parte del orgánulo multicomponenente llamado cloroplasto. La estructura única de la molécula de clorofila le permite funcionar absorbiendo partículas de energía lumínica (fotones) de la luz solar y luego usarlos para excitar electrones. Estos electrones son pasados a otras moléculas en una larga y compleja reacción. Las otras moléculas presentes en la reacción convierten la energía obtenida de la luz solar en energía química, que es almacenada temporalmente en la primera parte de la fotosíntesis. Esta energía se usa luego cuando el carbono presente en el dióxido de carbono es asimilado hasta conseguir glucosa, proceso que ocurre durante la segunda parte de la fotosíntesis.

Consideraciones curriculares e instructivas

Estudiantes de primaria

En los cursos de primaria, los estudiantes se preguntan acerca de las diferencias entre las plantas y los animales y hacen preguntas como por ejemplo "*¿Cómo se alimentan las plantas?*". Ellos aprenden que las plantas necesitan nutrientes y se les introduce la idea de que las plantas fabrican su propia comida, más que adquirirla como los animales, pero los conceptos relacionados con la fotosíntesis no son desarrollados hasta los cursos de secundaria.

Estudiantes de secundaria

En los cursos de secundaria, a los estudiantes se les introduce los procesos básicos de la fotosíntesis y hacen conexiones entre el asunto principal e ideas relacionadas con la energía, que son parte del entendimiento de este proceso vital para las plantas. Los alumnos aprenden sobre las células vegetales y como diferenciarlas de las células animales. Aunque los detalles relacionados a orgánulos celulares pueden esperar hasta los estudios de bachillerato, los estudiantes son iniciados en el conocimiento de los cloroplastos y del pigmento conocido como clorofila.

Estos términos son conectados para su entendimiento de la hoja como una estructura de la planta que toma dióxido de carbono y absorbe luz solar para conseguir energía que se usará en el proceso de fotosíntesis. Los estudiantes examinan hojas a nivel microscópico y observan los cloroplastos dentro de las células de la hoja que llevan a cabo el proceso de fotosíntesis.

Estudiantes de bachillerato

En los cursos de bachillerato, los alumnos profundizan su conocimiento de la fotosíntesis, incluyendo las moléculas involucradas. En este curso, los estudiantes deberían estar familiarizados con el término *clorofila* y saber que es un pigmento ubicado en el cloroplasto que absorbe energía lumínica, que el cloroplasto la convierte en energía química. Además aprenden que existen dos tipos de clorofila en las plantas: clorofila A y clorofila B. También añaden a su entendimiento el espectro visible de las longitudes de onda de la luz absorbida por las moléculas de clorofila para comprender por qué las plantas son verdes. [24]

15.1.1.2 Luz y oscuridad

Propósito

El propósito de esta sonda de evaluación es provocar ideas a los estudiantes sobre cuando ocurren los procesos de fotosíntesis y de respiración. El sondeo está diseñado para revelar si los alumnos reconocen que las plantas respiran continuamente.

Explicación

La mejor respuesta es la siguiente: “La fotosíntesis ocurre cuando hay luz; la respiración ocurre tanto cuando hay luz como cuando hay oscuridad.” La palabra *fotosíntesis* significa “hacer con la luz.” Las plantas fotosintetizan durante el día cuando la luz solar está disponible (o en otros períodos cuando están expuestas a luz artificial). Las plantas capturan la energía lumínica procedente del Sol, que convierten posteriormente en energía química durante el proceso de fotosíntesis. Las células vegetales, como todas las demás células, necesitan energía para llevar a cabo sus procesos de vida.

La respiración celular es el proceso durante el cual las células toman oxígeno para romper azúcares y producir adenosín trifosfato (ATP) que puede ser utilizado por la célula. Este proceso no necesita luz, requiere una molécula de alimentos como glucosa y oxígeno (para la respiración aeróbica). El proceso de respiración celular ocurre siempre que la célula necesita energía para llevar a cabo sus procesos de vida, sea de día o de noche. Un concepto erróneo muy común es que las plantas fotosintetizan durante el día y realizan la respiración celular solo por la noche. Este error fue incluso perpetuado cuando los hospitales solían quitar

las plantas de las habitaciones de los pacientes por las noches ¡para asegurar un suplemento de oxígeno adecuado para el paciente!

La luz es necesaria para ejecutar uno de los dos principales procesos de reacción fotosintética. Esta reacción dependiente de la luz utiliza la energía proveniente del Sol para producir electrones de alta energía (que son almacenados en ATP y NADPH). Además esta reacción produce oxígeno como producto residual. Simultáneamente, un segundo conjunto de reacciones conocido como el ciclo de Calvin usa el ATP y el NADPH que se ha formado durante la reacción anterior para producir azúcares de alta energía. El ciclo de Calvin ocurre al mismo tiempo que la reacción dependiente de la luz dentro del cloroplasto. Las reacciones en el ciclo de Calvin pueden ocurrir con o sin luz, ya que no dependen de la luz. Por eso, las reacciones del ciclo de Calvin también pueden ser llamadas reacciones independientes de la luz o reacciones oscuras. Sin embargo, el proceso de fijación del carbono es dependiente de la luz en su totalidad porque sin la energía de la luz, los cloroplastos agotarían los ATP y los NADPH requeridos para la reacción y el ciclo de Calvin se detendría. Los alumnos que reconozcan que hay una reacción con luz y otra sin ella pueden creer erróneamente que la reacción sin luz solo puede ocurrir cuando haya oscuridad.

La fotosíntesis y la respiración celular ocurren en diferentes sitios dentro de la célula. La fotosíntesis ocurre en el cloroplasto y la respiración se da lugar en la mitocondria. El propósito principal de la fotosíntesis es fabricar la comida que las plantas necesitan para llevar a cabo sus procesos o almacenarla para su uso posterior por la planta. Por otro lado, el objetivo principal de la respiración celular es liberar energía de la comida de las plantas hechas para llevar a cabo sus procesos de vida. Las ideas que deben quedar claras en este sondeo son que la fotosíntesis requiere luz para que el proceso completo ocurra y que la respiración celular ocurre de manera continua independientemente de si hay o no luz.

Consideraciones curriculares e instructivas

Estudiantes de primaria

En los cursos de primaria, los alumnos aprenden que las plantas necesitan agua, luz solar, nutrientes y aire. Además se les introduce la idea de que las plantas fabrican su propia comida y necesitan oxígeno del aire. Sin embargo, los detalles de los procesos de la fotosíntesis y la respiración celular deberán esperar hasta los estudios de secundaria.

Estudiantes de secundaria

En el instituto, los alumnos construyen sobre su entendimiento básico de que las plantas necesitan luz solar, agua y aire para entender el nexo entre estas necesidades y los procesos de fotosíntesis y respiración. Ellos cualitativamente aprenden que las plantas toman dióxido de carbono y agua y que utilizan la energía procedente del Sol para producir azúcares y oxígeno. Los detalles sobre el proceso, incluyendo la formación de ATP y NADPH son explicados más adelante en los estudios de bachillerato.

Los alumnos también aprenden que la respiración es un proceso celular en el que los organismos toman oxígeno para romper azúcares y liberar energía. Sin embargo, el camino en el que estos procesos son presentados en el plan de estudios a menudo transmite el concepto erróneo de que la fotosíntesis es un proceso vegetal y la respiración es un proceso animal, o que los dos procesos en las plantas son opuestos (que uno ocurre durante el día y otro ocurre por la noche)

Estudiantes de bachillerato

Los estudiantes de bachiller construyen un entendimiento descriptivo y básico de la fotosíntesis y la respiración celular para entender el proceso celular y molecular implicado en la sintetización de comida y en la ruptura de la propia comida para liberar energía. Los alumnos pasan del entendimiento de las macroestructuras de la planta que toman agua y liberan gases y fabrican comida a un entendimiento de los orgánulos celulares incluyendo los cloroplastos y las mitocondrias.

Los estudiantes pueden profundizar su conocimiento examinando las reacciones específicas que ocurren en el interior de estas estructuras y la relación dentro de estas estructuras que facilita dichas reacciones, incluyendo las reacciones dependientes e independientes de la luz de la fotosíntesis y la respiración celular aerobia y anaerobia. [25]

15.1.1.3 Respiración

Propósito

El propósito de esta sonda de evaluación es provocar ideas a los estudiantes sobre la respiración. El sondeo está diseñado para averiguar si los alumnos reconocen la respiración

como un proceso que todo ser viviente realiza para obtener energía o si por el contrario, tienen un concepto más restrictivo y general de la respiración.

Explicación

Todo lo que hay en la lista utiliza el proceso de respiración. La respiración es un proceso esencial vital llevado a cabo por todos los organismos vivos (desde los unicelulares hasta los pluricelulares) para proveer la energía que los organismo necesitan para funcionar. La respiración aeróbica ocurre a dos niveles. A nivel de organismo, generalmente implica la toma de aire que contiene el oxígeno necesitado por las células y la eliminación del dióxido de carbono procedente del interior. A nivel celular, el oxígeno es usado para romper las moléculas de comida para liberar la energía necesitada por las células para funcionar. El dióxido de carbono es liberado de la célula como un producto residual.

La mayoría de la gente, incluidos los estudiantes, entienden que los animales con alguna forma de sistema respiratorio, aspiran oxígeno del aire hacia sus sistemas respiratorios y exhalan dióxido de carbono. Esta gente normalmente equipara el intercambio de gases ocurrido durante la respiración aeróbica con la respiración más que con un proceso celular. Todos los animales respiran, pero no son los únicos organismos que lo hacen porque todos aquellos organismos vivos que estén compuestos por al menos una célula también respirarán. Todas las células necesitan energía para funcionar, por lo que cada organismo debe llevar a cabo alguna forma de respiración celular independientemente de si tiene un sistema respiratorio que incluye órganos como pulmones o branquias.

Mientras que los diferentes tipos de organismos pueden realizar la respiración de diferentes maneras, todos los organismos utilizan la respiración para liberar energía mediante la ruptura de moléculas dentro de una célula. La respiración aeróbica trae consigo un intercambio de gases entre un organismo y su entorno. A veces este intercambio involucra a estructuras pluricelulares (por ejemplo, órganos) en un organismo que toma oxígeno y lo pone a disposición de las células. Por ejemplo, las plantas toman oxígeno a través de sus hojas y los animales toman oxígeno a través de sus pulmones o branquias donde es enviado el aire y usado posteriormente dentro de sus células para romper azúcares (alimento) y liberar energía. Los organismos unicelulares pueden absorber oxígeno en una célula directamente desde el entorno. La respiración también puede ocurrir en ausencia de oxígeno, este tipo de respiración

anaeróbica ocurre con algunos tipos de bacteria y hongos así como en las células musculares de los animales donde no hay oxígeno.

Los organismos en un estado precoz de desarrollo, como las larvas de mariposa dentro de su crisálida, huevos de rana y un pollo antes de nacer, dentro de su huevo, también respiran mediante la toma de oxígeno, poniéndolo a disposición de sus células y liberando energía de sus moléculas de alimentos. Todos ellos son organismos vivos que necesitan energía para desarrollarse. Bajo las condiciones adecuadas de temperatura y húmedas, las semillas respiran tomando oxígeno, aunque pueden estar inactivas durante largos períodos de tiempo antes de su germinación.

Las plantas utilizan el oxígeno en el proceso de respiración, además también toman dióxido de carbono y liberan oxígeno en el proceso de fotosíntesis explicado anteriormente. Sin embargo, estos dos procesos no son opuestos ni son mutuamente exclusivos. La respiración en las plantas también ocurre durante la fotosíntesis.

Consideraciones curriculares e instructivas

Estudiantes de primaria

En los cursos de primaria, los alumnos distinguen entre organismos vivos y aquellos que no tienen vida y además aprenden que la mayoría de los organismos vivos necesitan aire. La aspiración a este nivel es normalmente igualada con la respiración y centrada en estructuras familiares de animales o plantas que toman oxígeno, como los pulmones, las branquias y las hojas. Mientras los alumnos investigan organismos unicelulares, aprenden que estos organismos tan simples también necesitan aire.

Estudiantes de secundaria

En los cursos de secundaria, los alumnos continúan aprendiendo sobre varias estructuras que toman oxígeno y la ponen a disposición de sus células, incluyendo las estructuras de los insectos y de los organismos acuáticos. En este nivel, los alumnos empiezan a asociar la toma de oxígeno con la necesidad de las células, desarrollando así una comprensión básica de la respiración celular sin entrar en detalles de la estructura de la célula o procesos biomecánicos.

Los estudiantes conectan la necesidad de oxígeno de las células con su entendimiento progresivo de la oxidación como un proceso que libera energía de los alimentos dentro de las células. En esta etapa, los alumnos deberían empezar a desarrollar la generalización de que todos los organismos respiran, pues todos los organismos vivos necesitan energía.

Estudiantes de bachillerato

En las clases de biología de bachillerato, los alumnos se basan en su comprensión de la respiración celular, obtenida en cursos anteriores, para examinar dicho proceso a nivel celular y molecular, incluyendo las estructuras eucariotas involucradas como las mitocondrias además de la respiración celular procariota.

Los estudiantes aprenden acerca de estos conceptos y distinguen entre los procesos de respiración aeróbica y respiración anaeróbica. Sin embargo, en este nivel, mientras que los alumnos aprenden sobre los procesos de la fotosíntesis con mayor detalle, algunos de ellos pueden creer que solamente los animales respiran y que la fotosíntesis es el proceso opuesto en las plantas. [26]

15.2 Anexo II: Manuales

15.2.1 Manual de la interfaz del servidor.

Para iniciar el servidor simplemente hay que ejecutar el fichero java adjuntado con el nombre de servidor.jar

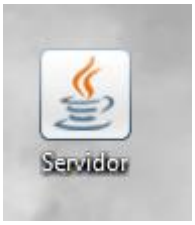


Figura 15-1 Icono

A continuación se muestra la interfaz del servidor.

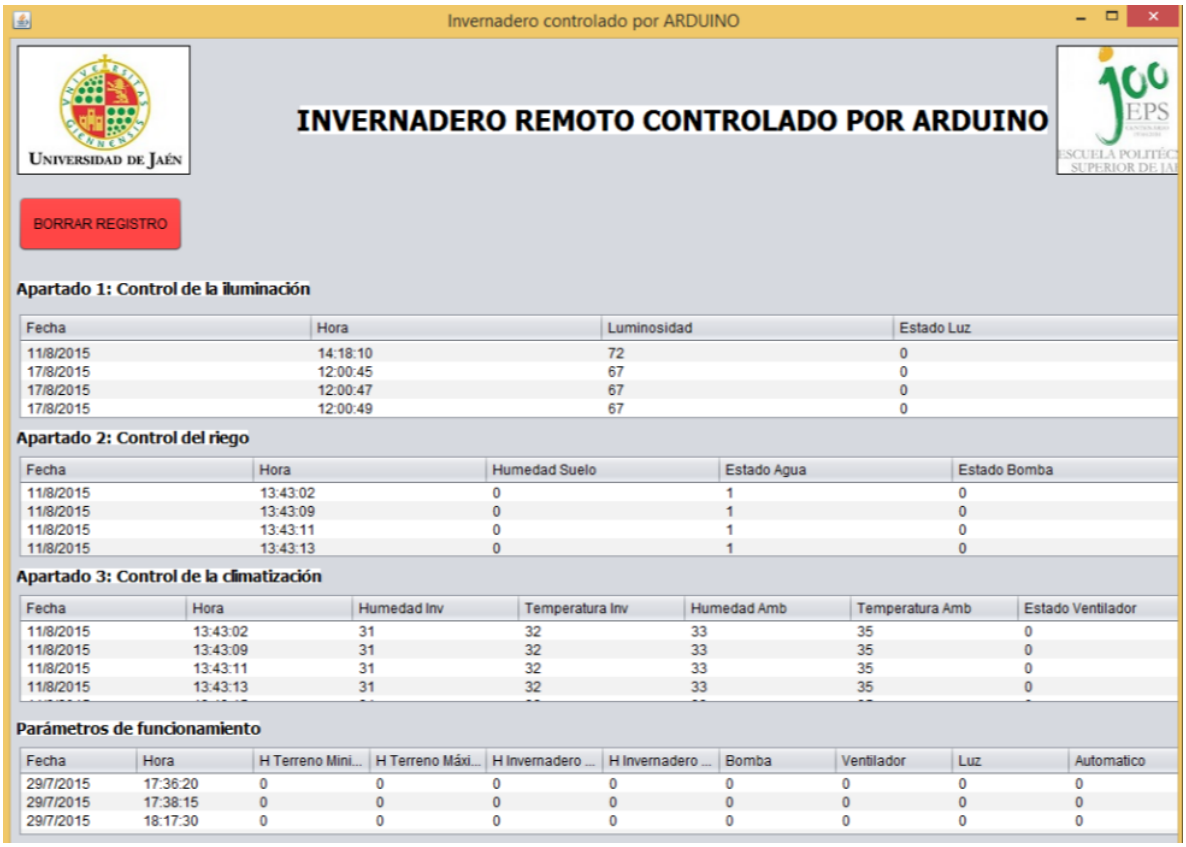


Figura 15-2 Interfaz del servidor

Como se puede comprobar se mostrarán cuatro tablas: las tres primeras guardan todos los datos proporcionados por los sensores (divididos por apartados, como en el laboratorio remoto) y la tabla restante almacena el registro de todos los parámetros mandados desde el cliente.

Adicionalmente se puede pulsar el botón de excel para exportar todos los datos o pulsar el botón de borrar registro para eliminar toda la información almacenada.

15.2.2 Manual de la interfaz del laboratorio virtual.

La interfaz con la que el cliente trabajaría sería la que se muestra a continuación:

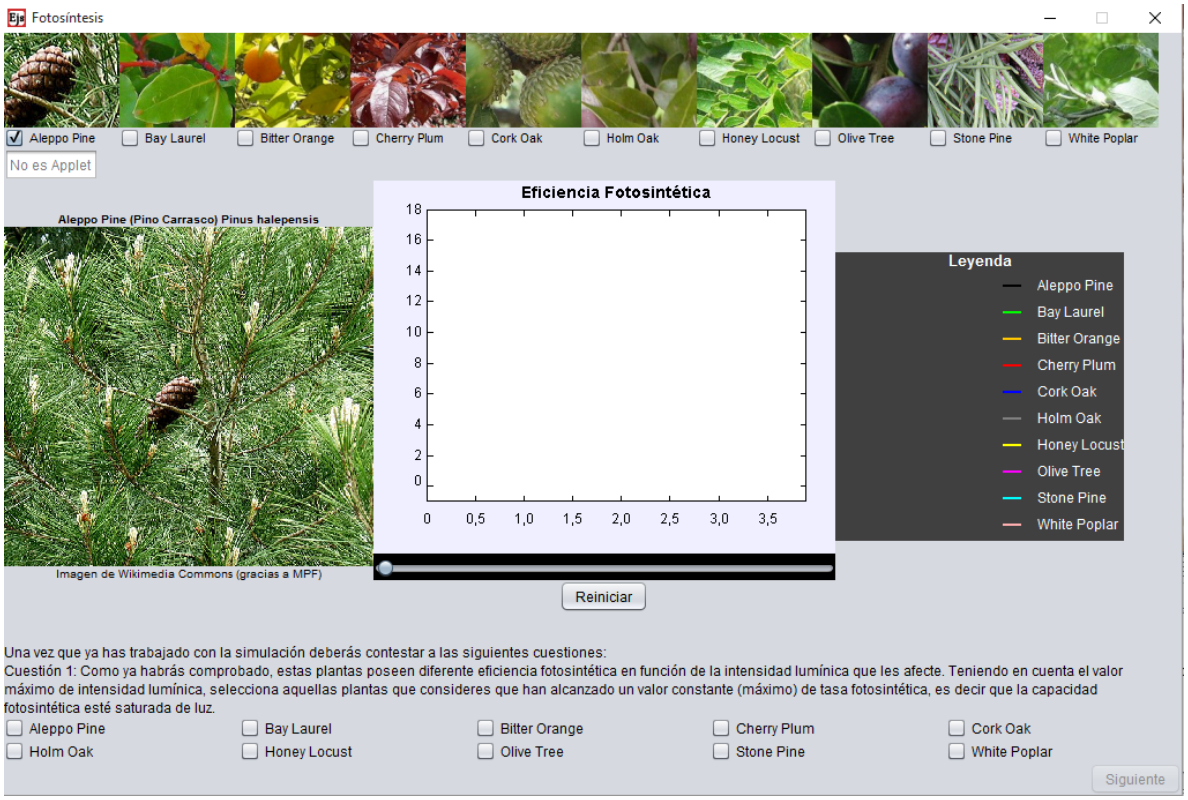


Figura 15-3 Interfaz principal

En la parte superior se observan diez selectores con diez fotografías, uno por cada tipo de planta que se ha escogido previamente para esta simulación. Para elegir una de las plantas se puede pulsar tanto en el selector como en la imagen. Si se pulsa una segunda vez, se desmarcará dicha opción.

En la parte central del interfaz, se encuentra una imagen de la última planta seleccionada, una gráfica que muestra la evolución de la eficiencia fotosintetica en aquellas plantas que se han seleccionado y una leyenda para reconocer fácilmente las trazas que se grafican. También hay que destacar la presencia de un deslizador y un botón llamado *Reiniciar*.

En la parte inferior, se aprecia un apartado dedicado a las preguntas obligatorias para superar esta simulación. Está compuesto por la pregunta y las diversas respuestas posibles.

En primer lugar, el alumno debe escoger aquellas plantas cuya evolución quiera observar. Acto seguido debe utlizar el deslizador, que representa la intensidad lumínica, y conforme se va aumentando el valor, en la gráfica se va dibujando la eficiencia fotosintética de las plantas seleccionadas.

En el caso de querer borrar la gráfica y realizar de nuevo la simulación, se debe pulsar el botón *Reiniciar*. Una vez que ya se ha visualizado la evolución de todas las plantas, se puede contestar a las preguntas que se plantean. Para ello, tras leer la cuestión el alumno escoge la respuesta (o respuestas) correctas y pulsa en el botón *Siguiente* para comprobar la puntuación que ha obtenido en esa pregunta y responder a otra nueva cuestión.

Después de contestar a esta segunda pregunta de manera correcta, se dará por concluido el trabajo en esta simulación y se permitirá avanzar al laboratorio remoto. Es importante añadir que el alumno debe obtener una puntuación mínima de 5/10 en las preguntas planteadas.

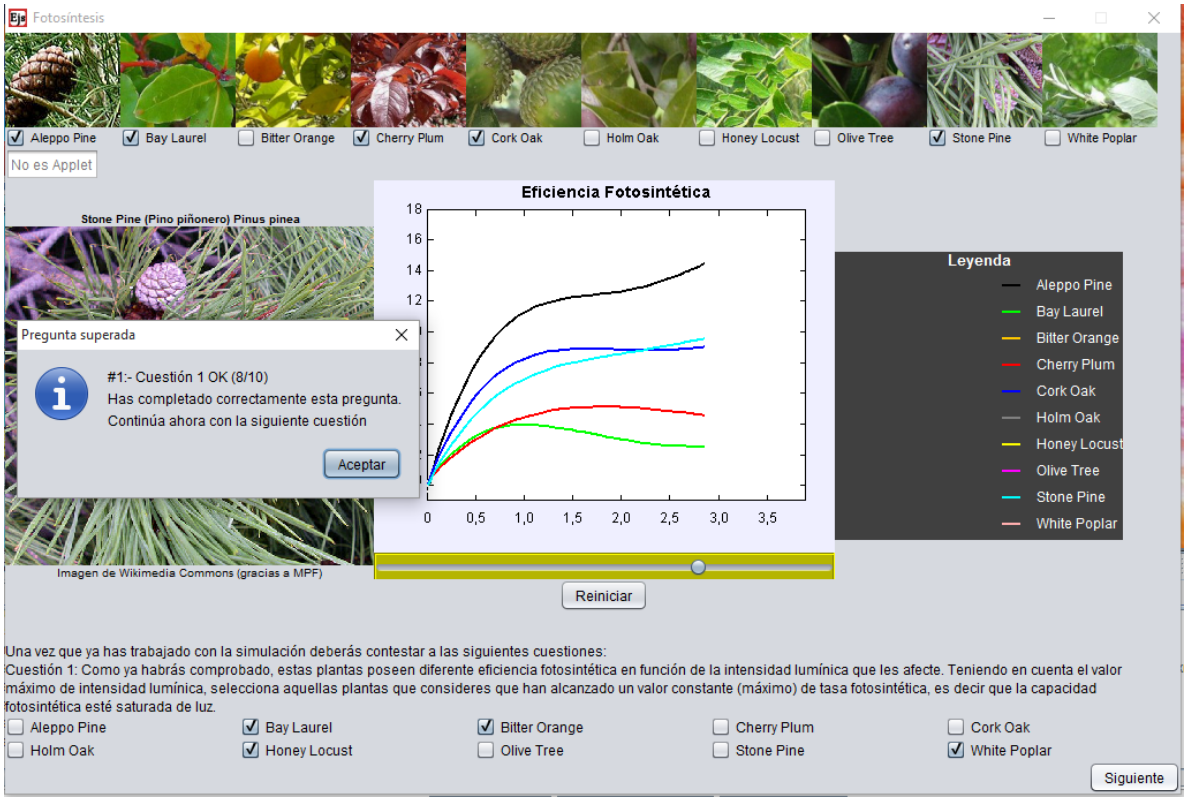


Figura 15-4 Funcionamiento del laboratorio virtual

15.2.3 Manual de la interfaz del laboratorio remoto.

El uso de la interfaz por parte del cliente es bastante sencillo, no obstante y para aclarar las posibles dudas que surjan se expone un breve manual a continuación:

1) En primer lugar hay que decir que el applet está dividido en tres apartados. Para comenzar se debe pulsar el botón *Conectar* para conseguir una comunicación de manera remota con el servidor situado en el laboratorio.
2) Una vez conectados, se estará recibiendo información de manera constante reflejada en las gráficas del interfaz. Además se podrá visionar en directo el estado del laboratorio mediante una pantalla que muestra la imagen en vivo.

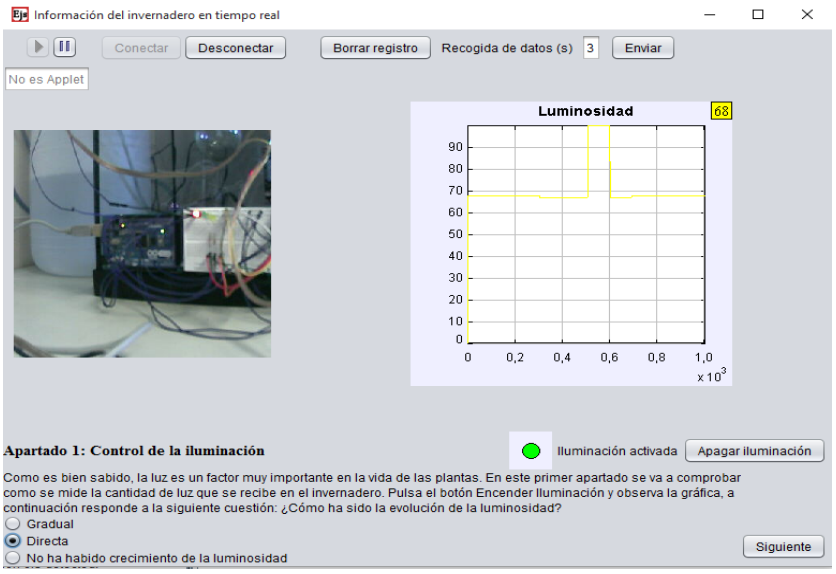


Figura 15-5 Apartado 1

3) En la parte superior izquierda, se observan dos botones, uno para pausar la aplicación y otra para volver a funcionar. Es importante destacar que ambos botones dependen del otro para su activación, es decir si la aplicación se está ejecutando de manera normal, el botón  no podrá ser pulsado y de igual manera ocurrirá con el botón 
4) De igual manera que en el caso anterior, ocurre con los botones *Desconectar* y *Conectar*. La habilitación de dichos botones depende de cuál de ellos esté activado.
5) En la parte superior derecha existe un campo numérico donde se puede cambiar el valor con el que Arduino recoge y almacena los datos de los sensores, este cambio se produce cuando se pulsa el botón *Enviar*. El botón *Borrar registro* vacía la base de datos que muestra el servidor para evitar problemas de acumulación de datos.

6) En la parte derecha se aprecia un botón que enciende uno de los actuadores (hay tres en total, uno por apartado). Todos ellos interactúan directamente con el laboratorio enviando información para encender (o apagar) actuadores como la bomba, el ventilador o la iluminación.

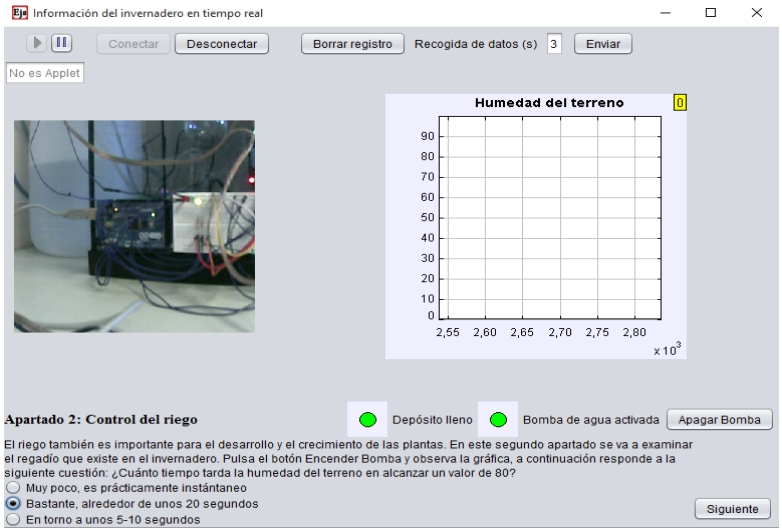


Figura 15-6 Apartado 2

7) Por último, hay una pregunta para cada apartado relacionada con la sección en la que se encuentre el alumno. Dicha pregunta tiene 3 opciones posibles y solo una es la correcta, se debe contestar correctamente la pregunta para poder avanzar al siguiente apartado.
8) En la parte inferior derecha se encuentra el botón *Siguiente* que se debe pulsar cuando se haya seleccionado la respuesta que el cliente crea que es la correcta. Una ventana informará al alumno de la nota obtenida y si es correcta, dicha puntuación se envía al LMS.

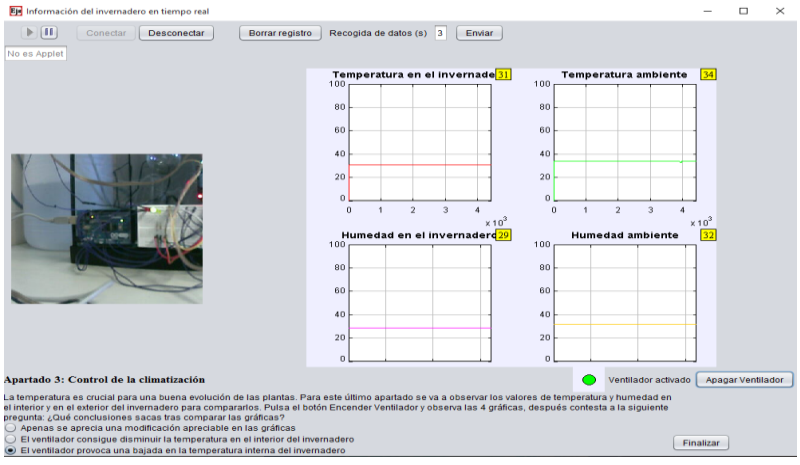


Figura 15-7 Apartado 3

15.3 Anexo III: Manuales de instalación

15.3.1 NetBeans IDE

A continuación, se van a describir los pasos a seguir para la instalación del JDK. Esta instalación se ha realizado en el sistema operativo Windows.

Figura 15-8 Aspecto web de descarga

En primer lugar, se debe escoger el sistema operativo y (leer y) aceptar las condiciones de la licencia:

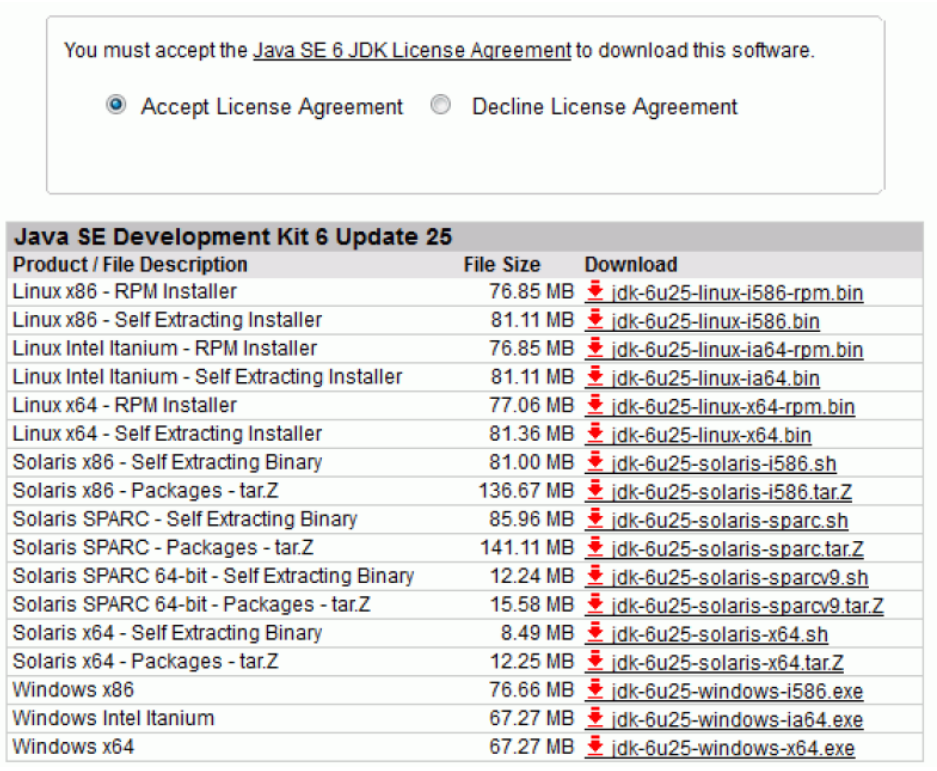


Figura 15-9 Selección del SO

Entonces se comenzará a recibir un único fichero de gran tamaño (cerca de 70 Mb, según versiones):

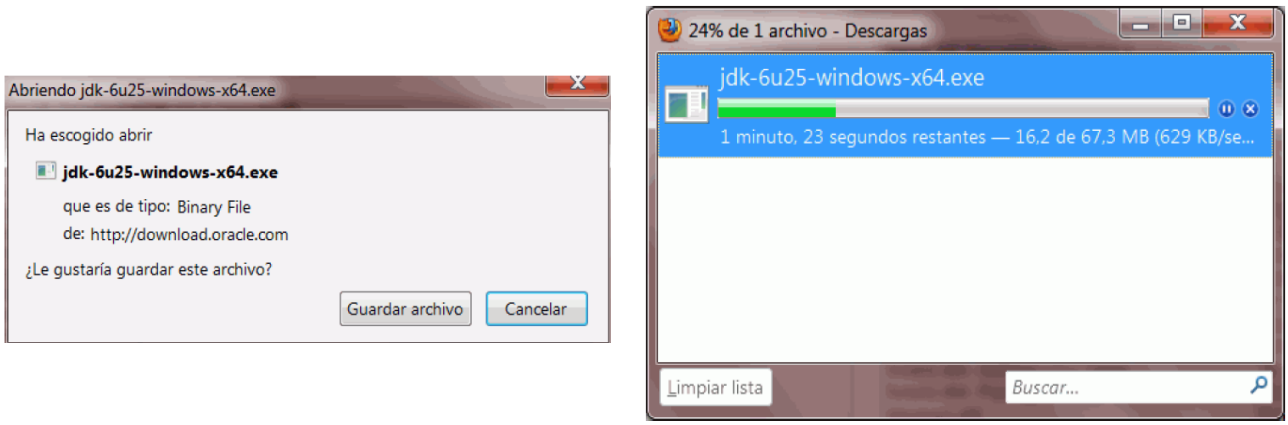


Figura 15-10 Inicio y desarrollo de la descarga

Cuando se haya descargado, se hará doble clic en el fichero, para comenzar la instalación propiamente dicha:

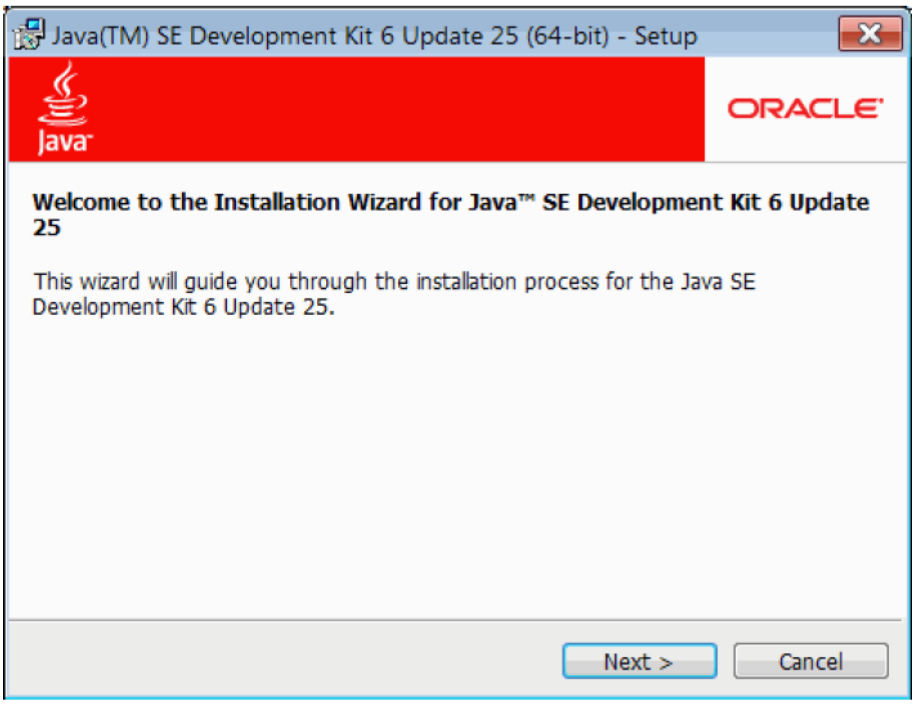


Figura 15-11 Instalación

Se podrán afinar detalles como la carpeta de instalación, o qué partes no se quieren instalar (por ejemplo, se podría optar por no instalar los ejemplos). Si se tiene suficiente espacio (posiblemente unos 400 Mb en total), se puede hacer una instalación típica, sin cambiar nada:

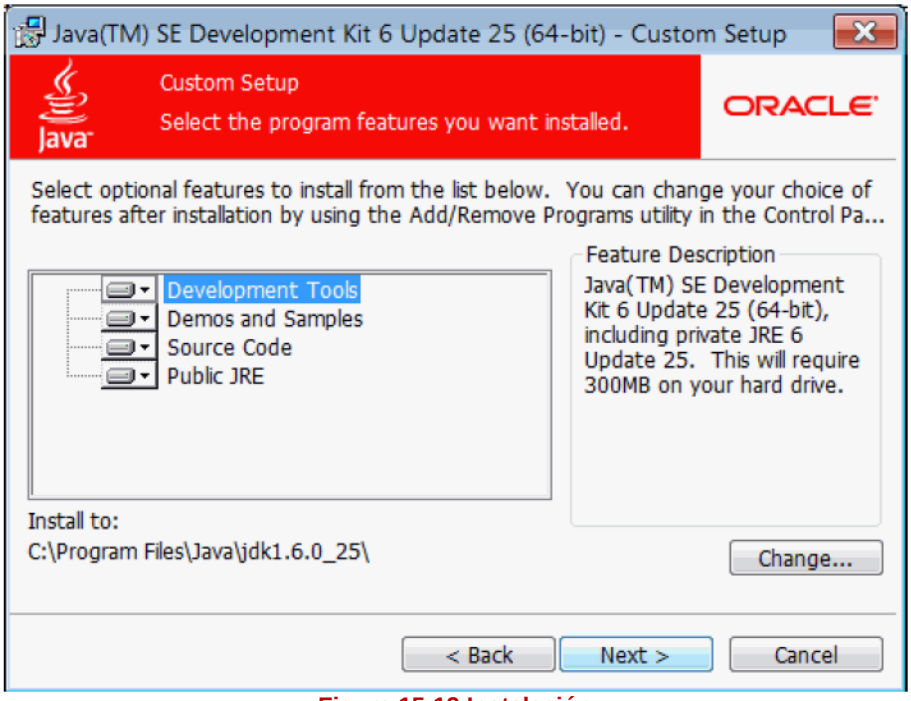


Figura 15-12 Instalación

En este momento se descomprime e instala todo:

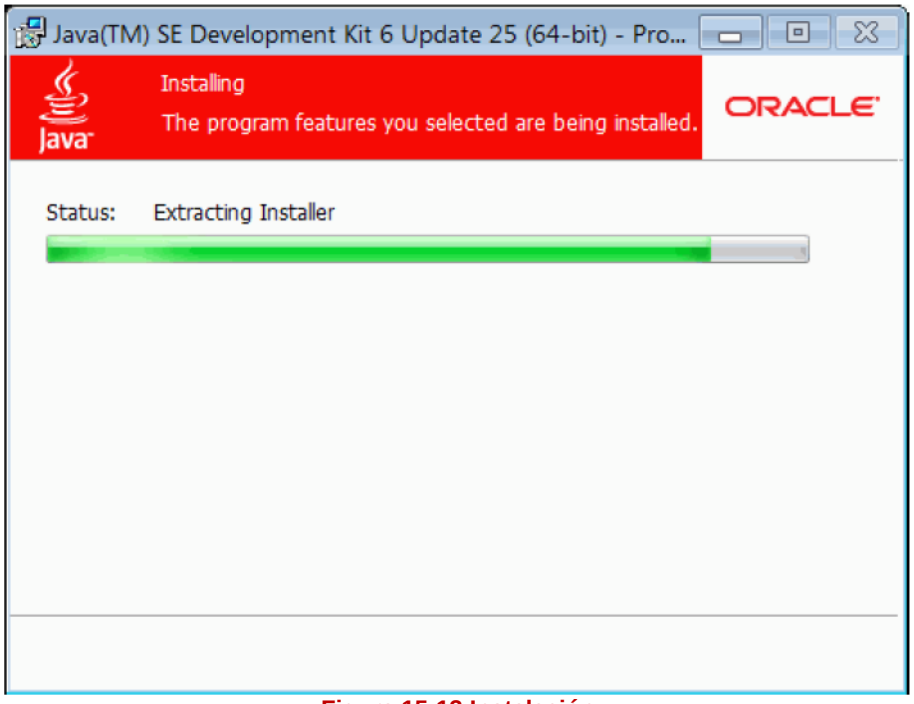


Figura 15-13 Instalación

En cierto punto se preguntará si se quiere instalar la máquina virtual Java (Java Runtime Environment, JRE). Lo razonable será responder que sí:

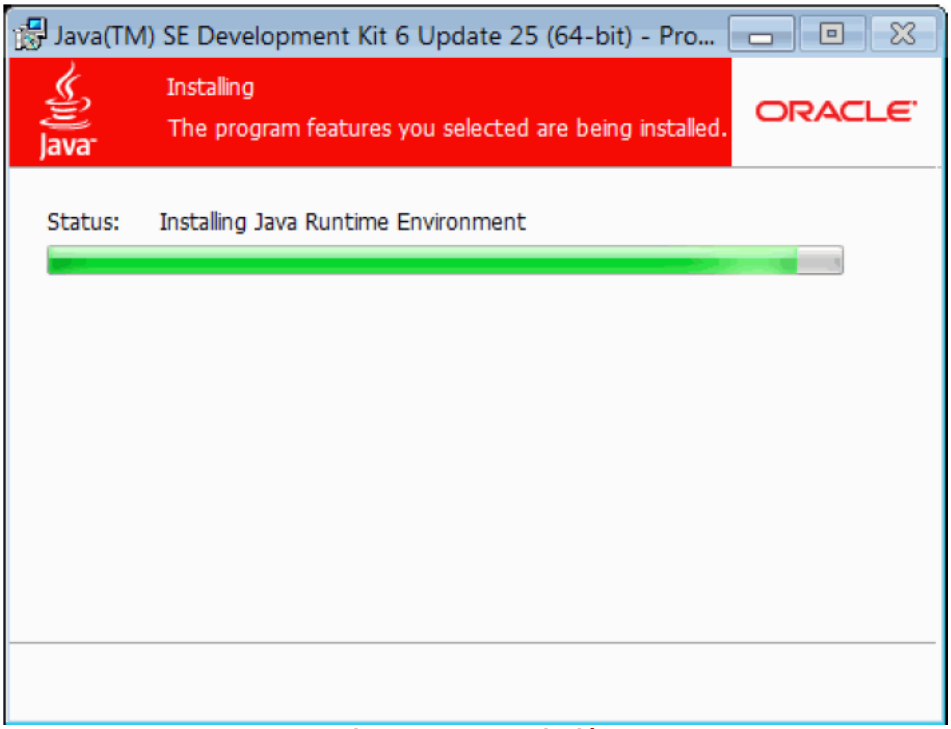


Figura 15-14 Instalación

Igual que para el JDK, se podría cambiar la carpeta de instalación:

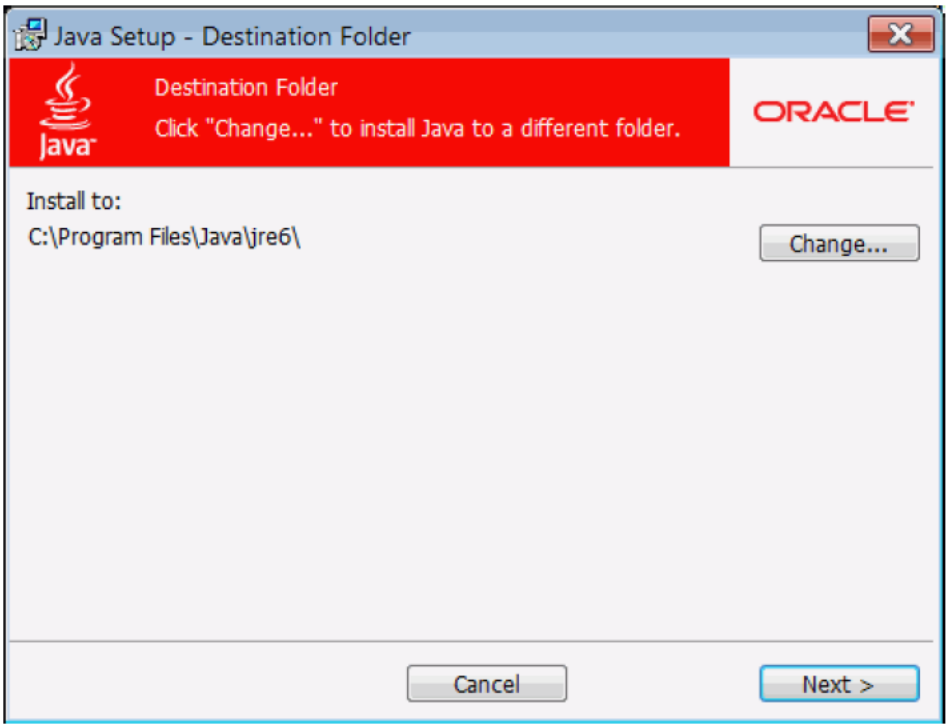


Figura 15-15 Instalación

Habrá que esperar otro momento.

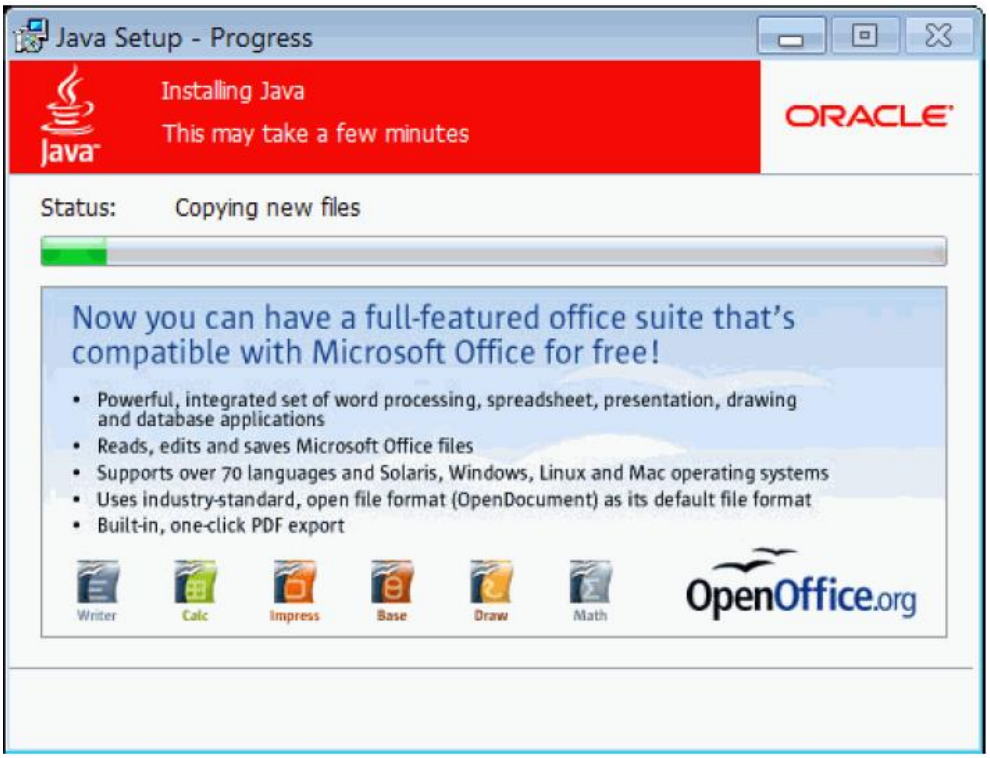


Figura 15-16 Instalación

Si todo ha ido bien, se obtendrá un mensaje de confirmación:

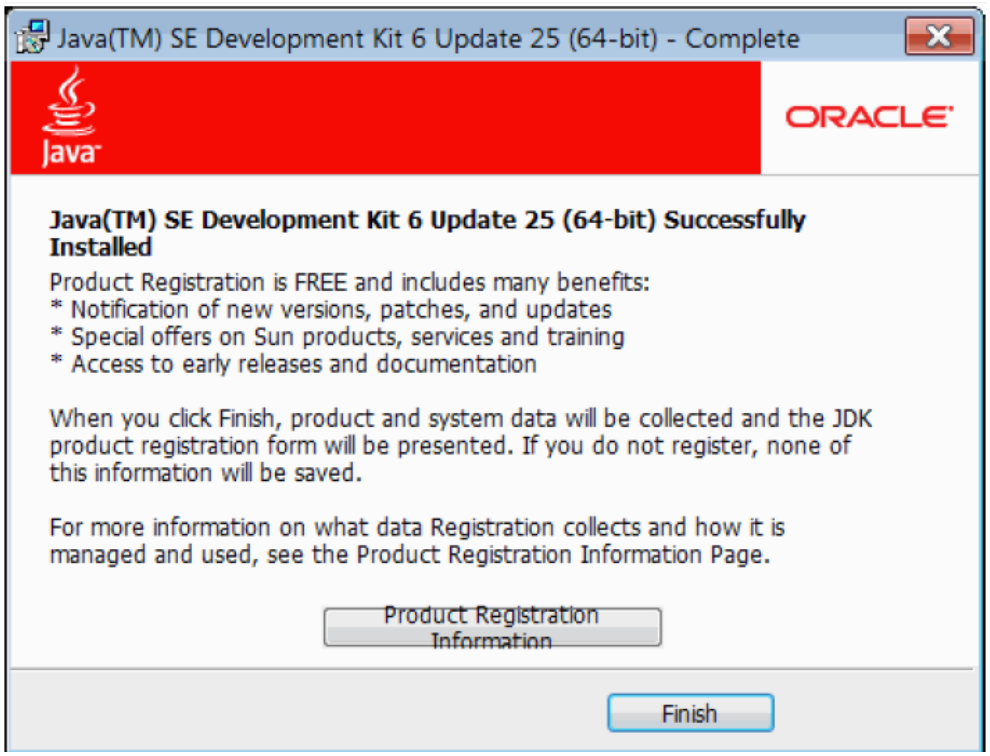


Figura 15-17 Instalación

Con eso ya se tiene instalada la herramienta básica. Una vez instalado el JDK, es hora de ir a la página de Netbeans: http://www.netbeans.org.

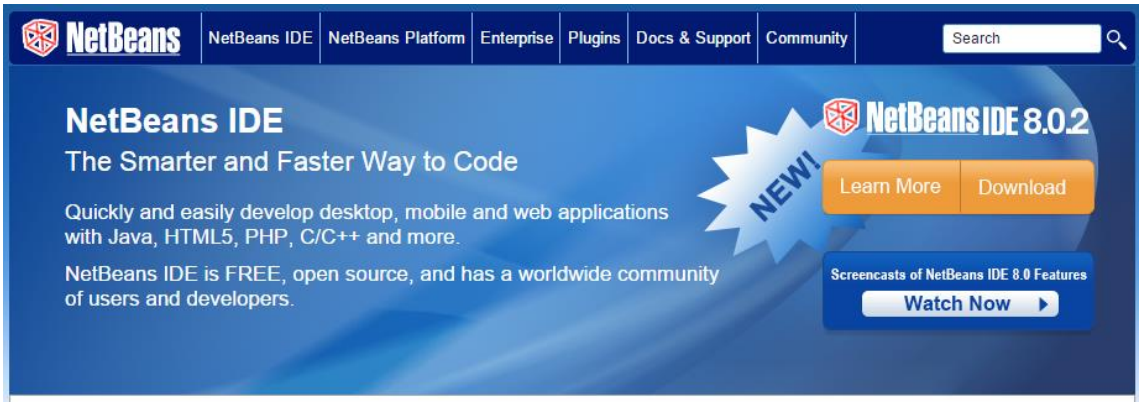


Figura 15-18 Aspecto de la web de NetBeans

Al hacer clic en "Download", se enlazará a la página de descargas, en la que existen varias versiones para elegir. Se ha elegido la versión “All” para así poder programar en un futuro en cualquier lenguaje.

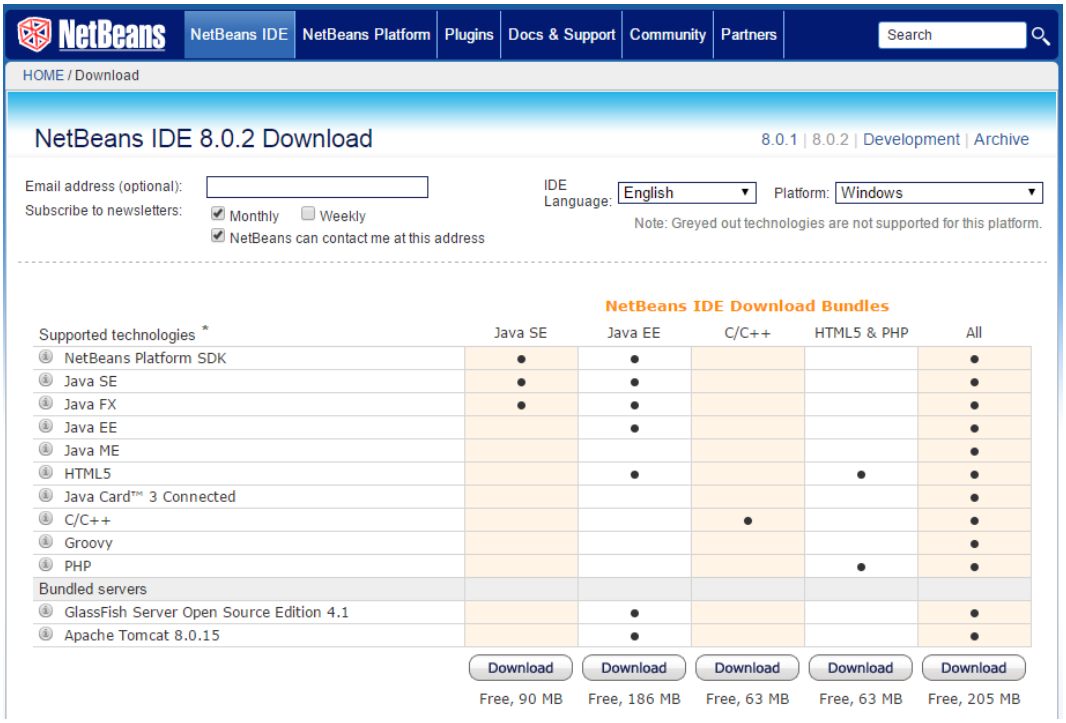


Figura 15-19 Versiones disponibles

Una vez elegida la versión, pulsar sobre “Donwload”.

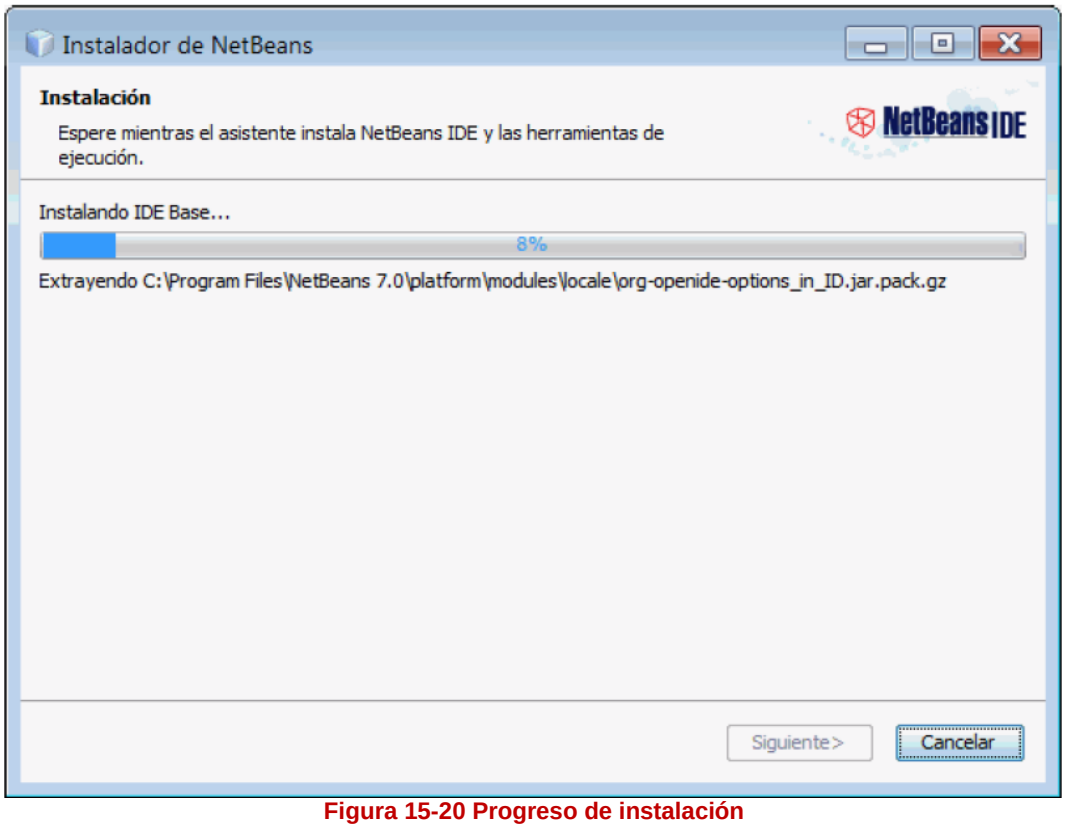


Figura 15-20 Progreso de instalación

Y al final quizá pregunte si se quiere permitir que se recopilen estadísticas del uso:

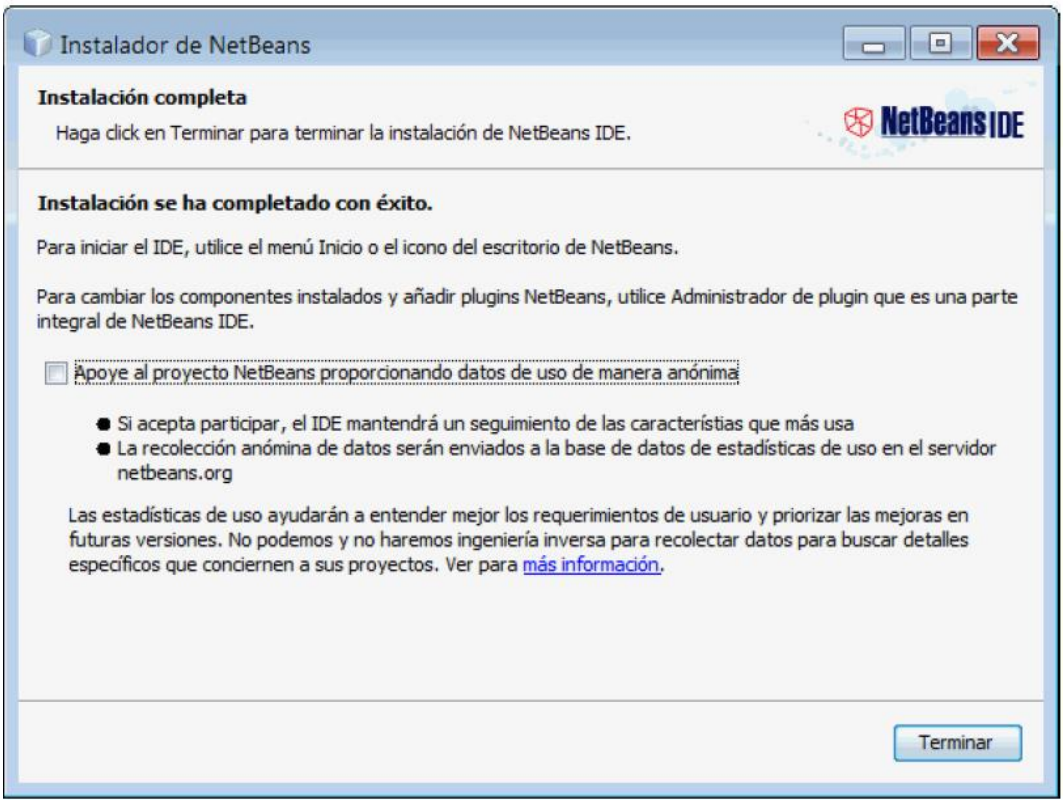


Figura 15-21 Instalación completa

A partir de este momento, la instalación ha finalizado. Ya se puede iniciar el IDE que tiene el siguiente aspecto:

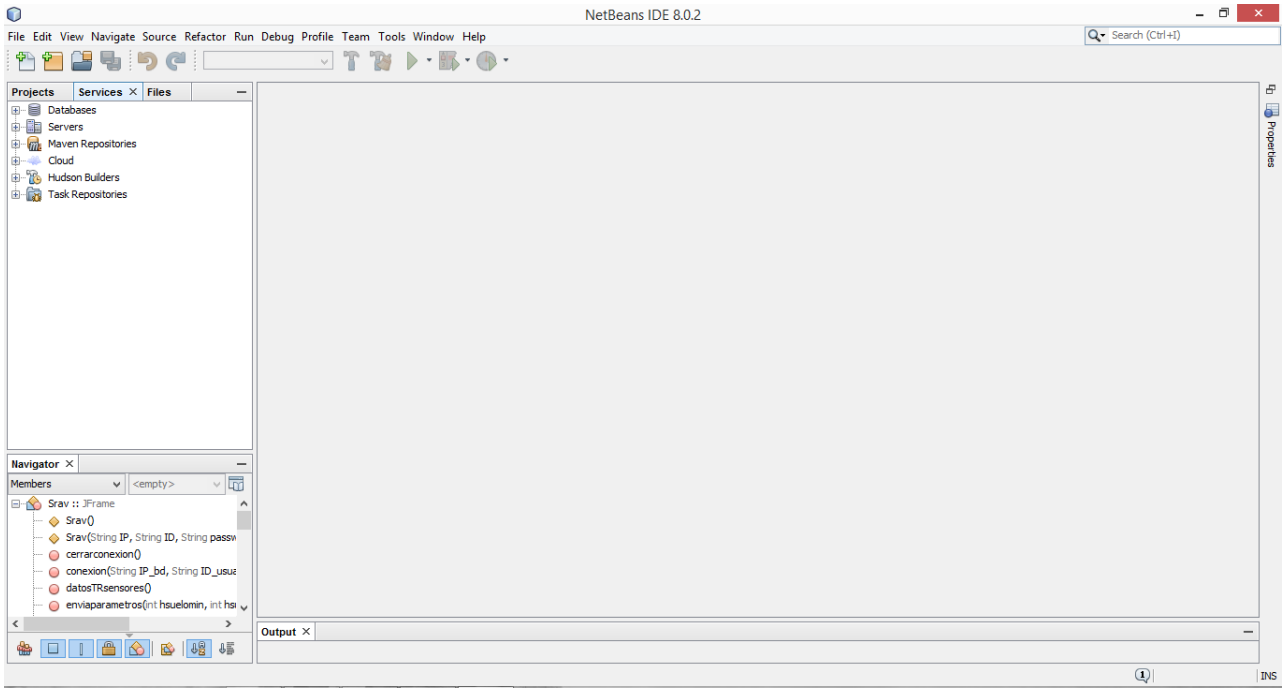


Figura 15-22 Entorno de la aplicación NetBeans

15.4 Anexo IV: Códigos de programación

15.4.1 Código de programación microcontrolador Arduino Mega

```
//SENSORES DHT11
// Se carga la librería DHT (sensor de temperatura y humedad)
#include "DHT.h"
#define DHTTYPE DHT11           //indicación tipo del sensor DHT

#define DHT_PIN_EXT 5           //pin al que se conecta el DHT11 exterior
#define DHT_PIN_INT 6           //pin al que se conecta el DHT11 interior

DHT dhtext(DHT_PIN_EXT, DHTTYPE);   //DHT11 exterior
DHT dhtint(DHT_PIN_INT, DHTTYPE);   //DHT11 interior


//SENSOR HL-69
int PIN_sensorsuelo=A2;

//SENSOR LDR
int PIN_sensorluminosidad=A3;

//SENSOR o2
int PIN_sensoro2=A4;

//SENSOR co2
int PIN_sensorco2=A5;

//SENSOR nivel de agua
int PIN_sensoragua=13;

// DECLARACIÓN E INICIALIZACIÓN VARIABLES INPUT
int hinv,tinv,hsuelo,hamb,tamb,lumi,o2,co2;
int estado_agua=1;
int estado_bomba=0;
int estado_ventilador=0;
int estado_luz=0;

//realiza la cuenta del intervalo de envío de datos al servidor
long tinicio = 0;
//intervalo en milisegundos de envío de datos al servidor
long intervalo = 2000;


// DECLARACIÓN E INICIALIZACIÓN VARIABLES OUTPUT
int PIN_bomba=10;       // Pin para actuar sobre la bomba de agua
int hsuelomin=300;      // Límite inferior para la humedad del terrero
int hsuelomax=400;      // Límite superior de humedad del terreno

int PIN_ventilador=11; // Pin para actuar sobre el ventilador
int hinvmin=70;         // Límite inferior para la humedad del invernadero
int hinvmax=85;         // Límite superior para la humedad del invernadero

int PIN_luz=12;

char bomba;
char ventilador;
char luz;
char automatico;
```

```
unsigned char buffer[9];
unsigned char i=0, ok=0;
int input;

//Se configura el puerto serie a 9600 baudios
void setup()
{
  Serial.begin(9600);

  dhtint.begin(); // Se inicia el sensor DHT del invernadero
  dhtext.begin(); // Se inicia el sensor DHT del exterior

  pinMode(PIN_bomba, OUTPUT);
  pinMode(PIN_ventilador, OUTPUT);
  pinMode(PIN_luz, OUTPUT);
  pinMode(PIN_sensoragua, INPUT);
}

void loop()
{

  //ENVIO VALORES DE LA LECTURA POR EL PUERTO SERIE

  unsigned long tactual = millis();
  if(tactual-tinicio > intervalo)
  {
      hinv=dhtint.readHumidity();    //Lee humedad interior
      tinv=dhtint.readTemperature(); //Lee temperatura interior

      hsuelo=analogRead(PIN_sensorsuelo);
      hsuelo=map(hsuelo, 0, 1024, 0, 100);

      hamb=dhtext.readHumidity();    //Lee humedad ambiente
      tamb=dhtext.readTemperature(); //Lee temperatura ambiente

      lumi = analogRead(PIN_sensorluminosidad);
      lumi = constrain(lumi, 100, 900); // Se normaliza el valor.
      lumi = map(lumi, 100, 900, 0, 100);

      o2=analogRead(PIN_sensoro2);
      o2=map(o2, 0, 1024, 0, 100);

      co2=analogRead(PIN_sensorco2);
      co2=map(co2, 0, 1024, 0, 100);

      estado_agua = digitalRead(PIN_sensoragua);



      Serial.print(hinv);
      Serial.print(" ");
      Serial.print(tinv);
      Serial.print(" ");
      Serial.print(hsuelo);
      Serial.print(" ");
      Serial.print(hamb);
      Serial.print(" ");
      Serial.print(tamb);
      Serial.print(" ");
      Serial.print(lumi);
```

```
      Serial.print(" ");
      Serial.print(o2);
      Serial.print(" ");
      Serial.print(co2);
      Serial.print(" ");
      Serial.print(estado_agua);
      Serial.print(" ");
      Serial.print(estado_bomba);
      Serial.print(" ");
      Serial.print(estado_ventilador);
      Serial.print(" ");
      Serial.println(estado_luz);

      tinicio=tactual;
  }


  //Si se recibió un dato entra
  if (Serial.available()>0)
  {
    input=Serial.read(); //Leo dato
    if(input!='#')
    {
      buffer[i]=input;
      buffer[i+1]='\0';
      i++;
    }
    else
    {
      ok=1;
    }
  }

  //Si ya llego toda la trama entra aquí
  if(ok==1)
  {
    //leo el buffer, proceso información y borro el buffer

    //Asigna los nuevos valores recibidos
    intervalo=(int)buffer[0]*1000;
    hsuelomax=(int)buffer[1]*4;
    hinvmin=(int)buffer[2];
    hinvmax=(int)buffer[3];
    bomba=buffer[4];
    ventilador=buffer[5];
    luz=buffer[6];
    automatico=buffer[7];


    ok=0; //bajo bandera
    i=0;  //borro el contador
  }

  if (automatico=='1')
  {
   if(hsuelo<hsuelomin && estado_agua==1)
   {
     digitalWrite(PIN_bomba,HIGH);
     estado_bomba=1;
   }
   else if(hsuelo>hsuelomax || estado_agua==0)
```

```
    {
      digitalWrite(PIN_bomba,LOW);
      estado_bomba=0;
    }

    if(hinv>hinvmax)
    {
      digitalWrite(PIN_ventilador,HIGH);
      estado_ventilador=1;
    }
    else if(hinv>hinvmin)
    {
      digitalWrite(PIN_ventilador,LOW);
      estado_ventilador=0;
    }
  }

  else if (automatico=='0')
  {
     if(bomba=='1' && estado_agua==1)
    {
      digitalWrite(PIN_bomba,HIGH);
      estado_bomba=1;
    }
    else if (bomba=='0' || estado_agua==0)
    {
      digitalWrite(PIN_bomba,LOW);
      estado_bomba=0;
    }

    if(ventilador=='1')
    {
      digitalWrite(PIN_ventilador,HIGH);
      estado_ventilador=1;
    }
    else if (ventilador=='0')
    {
      digitalWrite(PIN_ventilador,LOW);
      estado_ventilador=0;
    }

    if(luz=='1')
    {
      digitalWrite(PIN_luz,HIGH);
      estado_luz=1;
    }
    else if (luz=='0')
    {
      digitalWrite(PIN_luz,LOW);
      estado_luz=0;
    }
  }
}
```

15.4.2 Librería DHT.cpp

```
#include "DHT.h"

DHT::DHT(uint8_t pin, uint8_t type, uint8_t count) {
  _pin = pin;
  _type = type;
  _count = count;
  firstreading = true;
}

void DHT::begin(void) {
  // set up the pins!
  pinMode(_pin, INPUT);
  digitalWrite(_pin, HIGH);
  _lastreadtime = 0;
}

//boolean S == Scale.  True == Farenheit; False == Celcius
float DHT::readTemperature(bool S) {
  float f;

  if (read()) {
    switch (_type) {
    case DHT11:
      f = data[2];
      if(S)
            f = convertCtoF(f);

      return f;
    case DHT22:
    case DHT21:
      f = data[2] & 0x7F;
      f *= 256;
      f += data[3];
      f /= 10;
      if (data[2] & 0x80)
      f *= -1;
      if(S)
      f = convertCtoF(f);

      return f;
    }
  }
  Serial.print("Read fail");
  return NAN;
}

float DHT::convertCtoF(float c) {
      return c * 9 / 5 + 32;
}

float DHT::readHumidity(void) {
  float f;
  if (read()) {
    switch (_type) {
    case DHT11:
      f = data[0];
      return f;
    case DHT22:
    case DHT21:
```

```
      f = data[0];
      f *= 256;
      f += data[1];
      f /= 10;
      return f;
    }
  }
  Serial.print("Read fail");
  return NAN;
}


boolean DHT::read(void) {
  uint8_t laststate = HIGH;
  uint8_t counter = 0;
  uint8_t j = 0, i;
  unsigned long currenttime;

  // pull the pin high and wait 250 milliseconds
  digitalWrite(_pin, HIGH);
  delay(250);

  currenttime = millis();
  if (currenttime < _lastreadtime) {
    // ie there was a rollover
    _lastreadtime = 0;
  }
  if (!firstreading && ((currenttime - _lastreadtime) < 2000)) {
    return true; // return last correct measurement
    //delay(2000 - (currenttime - _lastreadtime));
  }
  firstreading = false;
  /*
    Serial.print("Currtime: "); Serial.print(currenttime);
    Serial.print(" Lasttime: "); Serial.print(_lastreadtime);
  */
  _lastreadtime = millis();

  data[0] = data[1] = data[2] = data[3] = data[4] = 0;

  // now pull it low for ~20 milliseconds
  pinMode(_pin, OUTPUT);
  digitalWrite(_pin, LOW);
  delay(20);
  cli();
  digitalWrite(_pin, HIGH);
  delayMicroseconds(40);
  pinMode(_pin, INPUT);

  // read in timings
  for ( i=0; i< MAXTIMINGS; i++) {
    counter = 0;
    while (digitalRead(_pin) == laststate) {
      counter++;
      delayMicroseconds(1);
      if (counter == 255) {
        break;
      }
    }
    laststate = digitalRead(_pin);
```

```
    if (counter == 255) break;

    // ignore first 3 transitions
    if ((i >= 4) && (i%2 == 0)) {
      // shove each bit into the storage bytes
      data[j/8] <<= 1;
      if (counter > _count)
        data[j/8] |= 1;
      j++;
    }
  }

  sei();

  /*
  Serial.println(j, DEC);
  Serial.print(data[0], HEX); Serial.print(", ");
  Serial.print(data[1], HEX); Serial.print(", ");
  Serial.print(data[2], HEX); Serial.print(", ");
  Serial.print(data[3], HEX); Serial.print(", ");
  Serial.print(data[4], HEX); Serial.print(" =? ");
  Serial.println(data[0] + data[1] + data[2] + data[3], HEX);
  */
  // check we read 40 bits and that the checksum matches
  if ((j >= 40) &&
      (data[4] == ((data[0] + data[1] + data[2] + data[3]) & 0xFF)) ) {
    return true;
  }

  return false;

}
```

15.4.3 Librería DHT.h

```
#ifndef DHT_H
#define DHT_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif

#define MAXTIMINGS 85

#define DHT11 11
#define DHT22 22
#define DHT21 21
#define AM2301 21

class DHT {
 private:
  uint8_t data[6];
  uint8_t _pin, _type, _count;
  boolean read(void);
  unsigned long _lastreadtime;
  boolean firstreading;

 public:
  DHT(uint8_t pin, uint8_t type, uint8_t count=6);
  void begin(void);
  float readTemperature(bool S=false);
  float convertCtoF(float);
  float readHumidity(void);

};
#endif
```

15.4.4 Código de programación para el sensor de CO_2

```
/************************Hardware Related Macros************************/
//define which analog input channel you are going to use
#define         MG_PIN                          (0)
#define         BOOL_PIN                        (2)
//define the DC gain of amplifier
#define         DC_GAIN                         (8.5)


/***********************Software Related Macros************************/
//define how many samples you are going to take in normal operation
#define         READ_SAMPLE_INTERVAL            (50)
/*define the time interval(in milisecond) between each samples in normal
operation*/
#define         READ_SAMPLE_TIMES               (5)


/**********************Application Related Macros**********************/
/*These two values differ from sensor to sensor. user should derermine this
 value.Define the output of the sensor in volts when the concentration of
CO2 is 400PPM*/
#define         ZERO_POINT_VOLTAGE              (0.324)
/*define the voltage drop of the sensor when move the sensor from air into
1000ppm CO2*/
#define         REACTION_VOLTGAE                (0.020)


/*****************************Globals***********************************/
/*two points are taken from the curve. With these two points, a line is
formed which is "approximately equivalent" to the original curve*/

/*data format:{ x, y, slope}; point1: (lg400, 0.324), point2: (lg4000,
0.280) */

//slope = ( reaction voltage ) / (log400 –log1000)

Float CO2Curve[3]={2.602,ZERO_POINT_VOLTAGE,(REACTION_VOLTGAE/(2.602-3))};

void setup()
{
    //UART setup, baudrate = 9600bps
    Serial.begin(9600);
    //set pin to input
    pinMode(BOOL_PIN, INPUT);
    //turn on pullup resistors
    digitalWrite(BOOL_PIN, HIGH);
    Serial.print("MG-811 Demostration\n");
}

void loop()
{
    int percentage;
    float volts;

    volts = MGRead(MG_PIN);
    Serial.print( "SEN0159:" );
    Serial.print(volts);
    Serial.print( "V           " );

    percentage = MGGetPercentage(volts,CO2Curve);
```

```
    Serial.print("CO2:");
    if (percentage == -1)
    {
        Serial.print( "<400" );
    }
    else
    {
        Serial.print(percentage);
    }
    Serial.print( "ppm" );
    Serial.print( "       Time point:" );
    Serial.print(millis());
    Serial.print("\n");

    if (digitalRead(BOOL_PIN))
    {
        Serial.print( "=====BOOL is HIGH======" );
    }
    else
    {
        Serial.print( "=====BOOL is LOW======" );
    }

    Serial.print("\n");

    delay(200);
}

/****************************  MGRead *********************************
Input:   mg_pin - analog channel
Output:  output of SEN-000007
Remarks: This function reads the output of SEN-000007
************************************************************************/
float MGRead(int mg_pin)
{
    int i;
    float v=0;

    for (i=0;i<READ_SAMPLE_TIMES;i++)
    {
        v += analogRead(mg_pin);
        delay(READ_SAMPLE_INTERVAL);
    }
    v = (v/READ_SAMPLE_TIMES) *5/1024 ;
    return v;
}

/****************************  MQGetPercentage ************************
Input:   volts   - SEN-000007 output measured in volts
         pcurve  - pointer to the curve of the target gas
Output:  ppm of the target gas
Remarks: By using the slope and a point of the line.The x(logarithmic value
of ppm)of the line could be derived if y(MG811 output) is provided.As it is
a logarithmic coordinate, power of 10 is used to convert the result to non-
logarithmic value.
************************************************************************/
int  MGGetPercentage(float volts, float *pcurve)
{
   if ((volts/DC_GAIN )>=ZERO_POINT_VOLTAGE)
   {
      return -1;
```

```
    }
    else
    {
        return pow(10, ((volts/DC_GAIN)-pcurve[1])/pcurve[2]+pcurve[0]);
    }
}
```

15.4.5 Código de programación servidor.jar

```
package servidor;

import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Calendar;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.table.DefaultTableModel;
import panamahitek.Arduino.PanamaHitek_Arduino;



public class SERVIDOR extends javax.swing.JFrame
{
    PanamaHitek_Arduino arduino= new PanamaHitek_Arduino();

    DefaultTableModel tablauno;
    DefaultTableModel tablados;
    DefaultTableModel tablatres;
    DefaultTableModel tablaparametros;

    //Declaración variables para tablasensores
    String Fecha, Hora;
    int hinv, tinv, hamb, tamb, hsuelo, lumi, o2, co2, h2o,
estadoluz,estadobomba,estadoventilador;
    boolean h2oboolean, estadoluzboolean, estadobombaboolean,
estadoventiladorboolean;
    String DatosSensores;

    //Declaración variables para tablaparametros
    String FECHA, HORA;
    int hsuelomin, hsuelomax, hinvmin, hinvmax;
    int newhsuelomin, newhsuelomax, newhinvmin, newhinvmax;

    boolean luz, bomba, ventilador, automatico;
    boolean newluz, newbomba,newventilador, newautomatico;


    Connection cn=null;


////////////////////////////////////////////////////////////////////////////
/

////////////////////////////////////////////////////////////////////////////
/

    public SERVIDOR() throws SQLException
    {
        this.cn=conexion();
        initComponents();
        this.setTitle("Invernadero controlado por ARDUINO");
```

```
        tablauno=(DefaultTableModel) tabla1.getModel();
        tablauno.setColumnIdentifiers(new Object[]{"Fecha", "Hora",
"Luminosidad","Estado Luz"});

        tablados=(DefaultTableModel) tabla2.getModel();
        tablados.setColumnIdentifiers(new Object[]{"Fecha", "Hora",
"Humedad Suelo","Estado Agua", "Estado Bomba"});

        tablatres=(DefaultTableModel) tabla3.getModel();
        tablatres.setColumnIdentifiers(new Object[]{"Fecha", "Hora",
"Humedad Inv","Temperatura Inv", "Humedad Amb", "Temperatura Amb", "Estado
Ventilador"});



        tablaparametros=(DefaultTableModel) tablap.getModel();
        tablaparametros.setColumnIdentifiers(new Object[]{"Fecha","Hora","H
Terreno Minima","H Terreno Máxima","H Invernadero Mínima","H Invernadero
Máxima","Bomba","Ventilador","Luz","Automatico"});


        //Realizo la conexión RXTX con Arduino
        try
        {
            arduino.ArduinoRXTX("COM4", 2000, 9600, evento);
        }
        catch (Exception ex)
        {
            Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE,
null, ex);
            System.out.println("No se pudo realizar la conexión con
Arduino");
        }

        /* A continuación obtengo el histórico. Vacio el contenido
        de la base de datos MySQL en mis tablas*/
        iniciatablaparametros();
        iniciatablasensores();
    }

    public Connection conexion () throws SQLException
    {   //Realiza la conexión con MySQL

        Connection conectar=null;  // Declaración variable conectar
        try
        {
            Class.forName("com.mysql.jdbc.Driver");

conectar=DriverManager.getConnection("jdbc:mysql://localhost:4001/cajadecri
stal","root","invernadero");
        }
        catch (ClassNotFoundException ex)
        {
            Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE,
null, ex);
        }

        return conectar;
```

```
    }

    SerialPortEventListener evento=new SerialPortEventListener()
    {
        @Override
        public void serialEvent(SerialPortEvent spe)
        {
            if (arduino.MessageAvailable())
            {   //Realizo la lectura desde Arduino
                DatosSensores=arduino.printMessage();

                //Separo la lectura en un array
                String[] ArrayDatosSensores=DatosSensores.split(" ");

                //Realizo la asignación
                hinv=Integer.parseInt(ArrayDatosSensores[0]);
                tinv=Integer.parseInt(ArrayDatosSensores[1]);
                hsuelo=Integer.parseInt(ArrayDatosSensores[2]);
                hamb=Integer.parseInt(ArrayDatosSensores[3]);
                tamb=Integer.parseInt(ArrayDatosSensores[4]);
                lumi=Integer.parseInt(ArrayDatosSensores[5]);
                o2=Integer.parseInt(ArrayDatosSensores[6]);
                co2=Integer.parseInt(ArrayDatosSensores[7]);

                h2o=Integer.parseInt(ArrayDatosSensores[8]);
                if (h2o==1)
                {
                    h2oboolean=true;
                }
                else
                {
                    h2oboolean=false;
                }

                estadobomba=Integer.parseInt(ArrayDatosSensores[9]);
                if (estadobomba==1)
                {
                    estadobombaboolean=true;
                }
                else
                {
                    estadobombaboolean=false;
                }

                estadoventilador=Integer.parseInt(ArrayDatosSensores[10]);
                if (estadoventilador==1)
                {
                    estadoventiladorboolean=true;
                }
                else
                {
                    estadoventiladorboolean=false;
                }

                estadoluz=Integer.parseInt(ArrayDatosSensores[11]);
                if (estadoluz==1)
                {
                    estadoluzboolean=true;
                }
                else
                {
```

```
                    estadoluzboolean=false;
                }


                /*Escribo una nueva fila en la tabla tablasensores
                y en la tabla MySQL tablasensores. Para ello llamo
                a la función enviasensores() */
                enviasensores();
            }

            try
            {
                /* Envio los parámetros de funcionamiento a Arduino.
                Para ello llamo a la función enviaparametros()*/
                enviaparametros();
            }
            catch (SQLException ex)
            {

Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE, null, ex);
            }



        }
    };

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTextField1 = new javax.swing.JTextField();
        jScrollPane1 = new javax.swing.JScrollPane();
        tabla1 = new javax.swing.JTable();
        jTextField2 = new javax.swing.JTextField();
        jScrollPane2 = new javax.swing.JScrollPane();
        tabla2 = new javax.swing.JTable();
        jTextField3 = new javax.swing.JTextField();
        jScrollPane3 = new javax.swing.JScrollPane();
        tabla3 = new javax.swing.JTable();
        jTextField4 = new javax.swing.JTextField();
        jScrollPane4 = new javax.swing.JScrollPane();
        tablap = new javax.swing.JTable();
        escudo_uja = new javax.swing.JLabel();
        escudo_eps = new javax.swing.JLabel();
        titulo = new javax.swing.JTextField();
        boton_truncate = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setBackground(new java.awt.Color(102, 102, 255));

        jTextField1.setEditable(false);
        jTextField1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jTextField1.setText("Apartado 1: Control de la iluminación");
        jTextField1.setBorder(null);
        jTextField1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });
```

```
        jScrollPane1.setAutoscrolls(true);

        tabla1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {

            }
        ));
        tabla1.setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        tabla1.setRowSelectionAllowed(false);
        jScrollPane1.setViewportView(tabla1);
        tabla1.getAccessibleContext().setAccessibleDescription("");

        jTextField2.setEditable(false);
        jTextField2.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jTextField2.setText("Apartado 2: Control del riego");
        jTextField2.setBorder(null);

        tabla2.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {

            }
        ));
        tabla2.setAutoscrolls(false);
        tabla2.setRowSelectionAllowed(false);
        jScrollPane2.setViewportView(tabla2);

        jTextField3.setEditable(false);
        jTextField3.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jTextField3.setText("Apartado 3: Control de la climatización");
        jTextField3.setBorder(null);
        jTextField3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField3ActionPerformed(evt);
            }
        });

        tabla3.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {

            }
        ));
        tabla3.setAutoscrolls(false);
        tabla3.setRowSelectionAllowed(false);
        jScrollPane3.setViewportView(tabla3);

        jTextField4.setEditable(false);
        jTextField4.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jTextField4.setText("Parámetros de funcionamiento");
        jTextField4.setBorder(null);
        jTextField4.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField4ActionPerformed(evt);
            }
        });

        tablap.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {

            }
        ));
        tablap.setAutoscrolls(false);
        tablap.setRowSelectionAllowed(false);
        jScrollPane4.setViewportView(tablap);

        escudo_uja.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/servidor/escudo_uja.jpg")));
// NOI18N
        escudo_uja.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        escudo_eps.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/servidor/logo.jpg"))); //
NOI18N
        escudo_eps.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));

        titulo.setEditable(false);
        titulo.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N
        titulo.setText("INVERNADERO REMOTO CONTROLADO POR ARDUINO");
        titulo.setBorder(null);

        boton_truncate.setBackground(new java.awt.Color(255, 51, 51));
        boton_truncate.setText("BORRAR REGISTRO");
        boton_truncate.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                boton_truncateActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING)
                    .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 1123, Short.MAX_VALUE)
                    .addComponent(jScrollPane1)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING)
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(escudo_uja,
javax.swing.GroupLayout.PREFERRED_SIZE, 150,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(93, 93, 93)
                                    .addComponent(titulo,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 99,
Short.MAX_VALUE)
                            .addComponent(escudo_eps,
javax.swing.GroupLayout.PREFERRED_SIZE, 129,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addComponent(jScrollPane3)
                        .addComponent(jScrollPane4)
                        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING)
                                .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jTextField3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(boton_truncate)
                                .addComponent(jTextField4,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(0, 0, Short.MAX_VALUE)))
                    .addContainerGap())
            );
            layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING)
                        .addGroup(layout.createSequentialGroup()
                            .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAI
LING)
                                .addComponent(escudo_eps,
javax.swing.GroupLayout.PREFERRED_SIZE, 112,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                            .addComponent(escudo_uja,
javax.swing.GroupLayout.PREFERRED_SIZE, 112,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(56, 56, 56)
                        .addComponent(titulo,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(18, 18, 18)
                .addComponent(boton_truncate,
javax.swing.GroupLayout.PREFERRED_SIZE, 49,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(23, 23, 23)
                .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 94,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 91,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE, 97,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jTextField4,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 80,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        pack();
```

```
    }// </editor-fold>

    public void enviasensores ()
    {
        /*Escribe en la tablasensores (virtual) y también en
        mySQL una fila con los últimos datos leídos del Arduino*/

        Calendar Calendario = Calendar.getInstance();
        String ano = Calendario.get(Calendar.YEAR) + "";
        String mes=
Integer.toString(1+Integer.parseInt(Calendario.get(Calendar.MONTH) + ""));
        String dia = Calendario.get(Calendar.DAY_OF_MONTH) + "";
        Fecha = dia + "/" + mes + "/" + ano;

        String hora = Calendario.get(Calendar.HOUR_OF_DAY) + "";
        String minuto = Calendario.get(Calendar.MINUTE) + "";
        String segundos = Calendario.get(Calendar.SECOND) + "";

        //Para conseguir  el formato correcto se realiza lo siguiente:
        if(Integer.parseInt(hora)<10)
        {
            hora = "0" + hora;
        }
        if(Integer.parseInt(minuto)<10)
        {
            minuto = "0" + minuto;
        }
        if(Integer.parseInt(segundos)<10)
        {
            segundos = "0" + segundos;
        }
        Hora = hora + ":" + minuto + ":" + segundos;

        tablauno.addRow(new Object[]{Fecha,Hora,lumi,estadoluzboolean?
1:0});
        tablados.addRow(new Object[]{Fecha,Hora,hsuelo,h2oboolean?
1:0,estadobombabooleanboolean? 1:0});
        tablatres.addRow(new
Object[]{Fecha,Hora,hinv,tinv,hamb,tamb,estadoventiladorboolean? 1:0});

        try
        {
            PreparedStatement pstm=cn.prepareStatement("INSERT INTO
tablasensores VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?)");
            pstm.setString(1, Fecha);
            pstm.setString(2, Hora);
            pstm.setInt(3, hinv);
            pstm.setInt(4, tinv);
            pstm.setInt(5, hsuelo);
            pstm.setInt(6, hamb);
            pstm.setInt(7, tamb);
            pstm.setInt(8, lumi);
            pstm.setInt(9, o2);
            pstm.setInt(10, co2);
            pstm.setBoolean(11, h2oboolean);
            pstm.setBoolean(12, estadobombabooleanboolean);
            pstm.setBoolean(13, estadoventiladorboolean);
            pstm.setBoolean(14, estadoluzboolean);

            pstm.executeUpdate();
        }
```

```
        catch (SQLException ex)
        {
            System.out.println("Error de escritura en mySQL");
            Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    public void enviaparametros() throws SQLException
    {
        /*Comprueba el último registro de tablaparametros(mySQL)
        y lo comprueba con el último guardado en la
tablaparametros(NetBeans)
        si son diferentes se mandan a Arduino*/

        //Realizo la conexión para lectura desde mySQL tablaparametros
        Statement stm=cn.createStatement();
        ResultSet rset=stm.executeQuery("SELECT * FROM tablaparametros");

        //Voy a la última fila y hago la lectura
        rset.last();
        FECHA=rset.getString(1);
        HORA=rset.getString(2);
        newhsuelomin=rset.getInt(3);
        newhsuelomax=rset.getInt(4);
        newhinvmin=rset.getInt(5);
        newhinvmax=rset.getInt(6);
        newbomba=rset.getBoolean(7);
        newventilador=rset.getBoolean(8);
        newluz=rset.getBoolean(9);
        newautomatico=rset.getBoolean(10);


        if
((hsuelomin!=newhsuelomin)||(hsuelomax!=newhsuelomax)||(hinvmin!=newhinvmin
)||(hinvmax!=newhinvmax)||(bomba!=newbomba)||(ventilador!=newventilador)||(
luz!=newluz)||(automatico!=newautomatico))
        {
            /*Si desde el cliente EJS se cambia algún parámetro en
            tablaparametros(mySQL) lo registro en la tabla virtual
            tablaparametros(NetBeans) y actualizo los parámetros
            de funcionamiento*/
            hsuelomin=newhsuelomin;
            hsuelomax=newhsuelomax;
            hinvmin=newhinvmin;
            hinvmax=newhinvmax;
            bomba=newbomba;
            ventilador=newventilador;
            luz=newluz;
            automatico=newautomatico;


            tablaparametros.addRow(new Object[]{FECHA, HORA,hsuelomin,
hsuelomax, hinvmin, hinvmax,bomba? 1:0,ventilador? 1:0,luz? 1:0,automatico?
1:0});

            String
cadena=""+((char)(hsuelomin+50))+((char)(hsuelomax/4))+((char)(hinvmin))+((
char)(hinvmax))+(bomba? '1':'0')+(ventilador? '1':'0')+(luz?
'1':'0')+(automatico? '1':'0')+'#'+"";
```

```
            /*A continuación se mandan los datos al Arduino*/
            try
            {
               arduino.sendData(cadena);
               System.out.println("Dato enviado: "+cadena);

               Thread.sleep(300);
            }
            catch (Exception ex)
            {

Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

    }

    public void iniciatablaparametros()
    {    /*Copia todo los registros de tablaparametros(mySQL)
          a tablaparametros(NetBeans)*/
        try
        {
            //Comienza la lectura desde mySQL tablaparametros
            Statement stm=cn.createStatement();
            ResultSet rset=stm.executeQuery("SELECT * FROM
tablaparametros");

            while(rset.next())   //Hago la lectura de todas las filas
            {
                FECHA=rset.getString(1);
                HORA=rset.getString(2);
                hsuelomin=rset.getInt(3);
                hsuelomax=rset.getInt(4);
                hinvmin=rset.getInt(5);
                hinvmax=rset.getInt(6);
                bomba=rset.getBoolean(7);
                ventilador=rset.getBoolean(8);
                luz=rset.getBoolean(9);
                automatico=rset.getBoolean(10);
                //Escribo en la tablaparametros
                tablaparametros.addRow(new Object[]{FECHA,
HORA,hsuelomin,hsuelomax,hinvmin,hinvmax,bomba? 1:0,ventilador? 1:0,luz?
1:0,automatico? 1:0});
            }
        }
        catch (SQLException ex)
        {
            Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    public void iniciatablasensores()
    {   /*Copia todo los registros de tablasensores(mySQL)
        a tablasensores(NetBeans)*/
        try
        {
            //Comienza la lectura desde mySQL tablasensores
            Statement stm=cn.createStatement();
            ResultSet rset=stm.executeQuery("SELECT * FROM tablasensores");
            while(rset.next())   //Hago la lectura de todas las filas
```

```
            {
                Fecha=rset.getString(1);
                Hora=rset.getString(2);
                hinv=rset.getInt(3);
                tinv=rset.getInt(4);
                hsuelo=rset.getInt(5);
                hamb=rset.getInt(6);
                tamb=rset.getInt(7);
                lumi=rset.getInt(8);
                o2=rset.getInt(9);
                co2=rset.getInt(10);
                h2oboolean=rset.getBoolean(11);
                estadobombaboolean=rset.getBoolean(12);
                estadoventiladorboolean=rset.getBoolean(12);
                estadoluzboolean=rset.getBoolean(13);
                //Escribo en la tablasensores

                tablauno.addRow(new
Object[]{Fecha,Hora,lumi,estadoluzboolean? 1:0});
                tablados.addRow(new Object[]{Fecha,Hora,hsuelo,h2oboolean?
1:0,estadobombaboolean? 1:0});
                tablatres.addRow(new
Object[]{Fecha,Hora,hinv,tinv,hamb,tamb,estadoventiladorboolean? 1:0});

            }
        }
        catch (SQLException ex)
        {
            Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    private void boton_truncateActionPerformed(java.awt.event.ActionEvent
evt) {
        try
        {
            Statement stm=cn.createStatement();
            stm.executeUpdate("TRUNCATE tablasensores");
        }
        catch (SQLException ex)
        {
            Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void jTextField4ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }
```

```
    public static void main(String args[]) {
        try
        {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels())
            {
                if ("Nimbus".equals(info.getName()))
                {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        }
        catch (ClassNotFoundException | InstantiationException |
IllegalAccessException | javax.swing.UnsupportedLookAndFeelException ex)
        {

java.util.logging.Logger.getLogger(SERVIDOR.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
        }

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable()
        {
            public void run()
            {
                try
                {
                    new SERVIDOR().setVisible(true);
                }
                catch (SQLException ex)
                {

Logger.getLogger(SERVIDOR.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton boton_truncate;
    private javax.swing.JLabel escudo_eps;
    private javax.swing.JLabel escudo_uja;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTable tabla1;
    private javax.swing.JTable tabla2;
    private javax.swing.JTable tabla3;
    private javax.swing.JTable tablap;
    private javax.swing.JTextField titulo;
    // End of variables declaration

}
```

15.4.6 Código de programación srav.jar

```
package srav;

import java.sql.*;
import java.util.Calendar;
import java.util.logging.Level;
import java.util.logging.Logger;


public class Srav extends javax.swing.JFrame
{
    // Declaración de la variable de tipo Connection
    public Connection cn;

    // Declaración parámetros de trabajo de Arduino//

    public int hsuelomin, hsuelomax, hinvmin,hinvmax;
    public boolean luz, bomba, ventilador,automatico;

    // Declaración datos sensores//

    // Valores de los sensores almacenados en la base de datos
    public int hinv, tinv, hsuelo, hamb, tamb, lumi,o2,co2,paso=0;
    // Valores de los sensores en tiempo real
    public int hinvtr, tinvtr, hsuelotr, hambtr, tambtr, lumitr, o2tr,
co2tr;
    public boolean  h2otr,estadoluz,estadobomba,estadoventilador;

    public String Fechas, Horas, IP, ID_nombre, password;


    // Probar a eliminar esto
    public Srav()
    {
    }

    public Srav (String IP_BD, String ID, String PASS)
    {
        IP=IP_BD;
        ID_nombre=ID;
        password=PASS;

        this.cn=conexion(IP, ID_nombre, password);

        mostrartablaparametros();
    }

    public Connection conexion(String IP_bd, String ID_usuario, String
pass)
    {
        Connection conectar=null;
        try
        {
            Class.forName("com.mysql.jdbc.Driver");

conectar=DriverManager.getConnection("jdbc:mysql://"+IP_bd+"/cajadecristal"
, ID_usuario, pass);
        }
        catch (Exception e)
```

```
        {
            System.out.print(e.getMessage());
        }

        return conectar;
    }

    public void cerrarconexion()
    {
        //Cierro la conexión con la base de datos

        try
        {
            cn.close();
        }
        catch (SQLException ex)
        {
            Logger.getLogger(Srav.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    public void mostrartablaparametros()
    {
        /* Lee desde mySQL TABLA tablaparametros (muestra ultima entrada)
        para saber en qué estado se dejó el sistema*/

        try
        {
            Statement st=cn.createStatement();
            ResultSet rs=st.executeQuery("SELECT * FROM tablaparametros ");

            //Se sitúa el cursor al final
            rs.last();
            hsuelomin=rs.getInt(3);
            hsuelomax=rs.getInt(4);
            hinvmin=rs.getInt(5);
            hinvmax=rs.getInt(6);
        }
        catch (SQLException ex)
        {
            Logger.getLogger(Srav.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    public int tamanotablasensores()
    {
        /* Devuelve el número de filas de la TABLA tablasensores*/

        int filas=0;

        try
        {
            Statement st = cn.createStatement();
            ResultSet rs= st.executeQuery("SELECT * FROM tablasensores");

            while(rs.next())
            {
                filas++;
            }
```

```
        }
        catch (SQLException ex)
        {
            Logger.getLogger(Srav.class.getName()).log(Level.SEVERE, null,
ex);
        }
        return filas;
    }

    public void tablasensores(int fila)
    {
        /* Lee desde mySQL TABLA tablasensores (muestra la fila deseada)*/
        try
        {
            Statement st=cn.createStatement();
            ResultSet rs=st.executeQuery("SELECT * FROM tablasensores");
            if(rs.absolute(fila))
            {
               Fechas=rs.getString(1);
               Horas=rs.getString(2);
               hinv=rs.getInt(3);
               tinv=rs.getInt(4);
               hsuelo=rs.getInt(5);
               hamb=rs.getInt(6);
               tamb=rs.getInt(7);
               lumi=rs.getInt(8);
               o2=rs.getInt(9);
               co2=rs.getInt(10);
            }
        }
        catch (SQLException ex)
        {
            Logger.getLogger(Srav.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    public void datosTRsensores()
    {
        /* Leer mySQL TABLA tablasensores
        (muestra la última fila solo si está actualizada)*/
        String nFechas;
        String nHoras;

        try
        {
            Statement st=cn.createStatement();
            ResultSet rs=st.executeQuery("SELECT * FROM tablasensores");

            /*Muevo el puntero a la última fila
            para leer la fecha y hora del último registro*/
            rs.last();
            nFechas=rs.getString(1);
            nHoras=rs.getString(2);

            /*A continuación comparo la última fecha y hora
            leída con la existente en el último registro. Si no coinciden
es
            debido a que el registro del tiempo ha cambiado y por
            lo tanto se está en tiempo real*/
           /*if (!nFechas.matches(Fechas)||(!nHoras.matches(Horas)))
```

```
            {*/
                 Fechas=nFechas;
                 Horas=nHoras;
                 hinvtr=rs.getInt(3);
                 tinvtr=rs.getInt(4);
                 hsuelotr=rs.getInt(5);
                 hambtr=rs.getInt(6);
                 tambtr=rs.getInt(7);
                 lumitr=rs.getInt(8);
                 o2tr=rs.getInt(9);
                 co2tr=rs.getInt(10);
                 h2otr=rs.getBoolean(11);
                 estadobomba=rs.getBoolean(12);
                 estadoventilador=rs.getBoolean(13);
                 estadoluz=rs.getBoolean(14);
             //}
        }
        catch (SQLException ex)
        {
            Logger.getLogger(Srav.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    public void enviaparametros(int hsuelomin,int hsuelomax, int hinvmin,
int hinvmax, boolean bomba, boolean ventilador, boolean luz, boolean
automatico) // Escribe en mySQL TABLA paramsArdu
    {
        Calendar Calendario=Calendar.getInstance();
        String ano = Calendario.get(Calendar.YEAR) + "";
        String mes=
Integer.toString(1+Integer.parseInt(Calendario.get(Calendar.MONTH) + ""));
        String dia = Calendario.get(Calendar.DAY_OF_MONTH) + "";
        String Fecha = dia + "/" + mes + "/" + ano;
        String hora = Calendario.get(Calendar.HOUR_OF_DAY) + "";
        String minuto = Calendario.get(Calendar.MINUTE) + "";
        String segundos = Calendario.get(Calendar.SECOND) + "";

         //Para que aparezca en el formato correcto
        if(Integer.parseInt(hora)<10)
        {
            hora = "0" + hora;
        }
        if(Integer.parseInt(minuto)<10)
        {
            minuto = "0" + minuto;
        }
        if(Integer.parseInt(segundos)<10)
        {
            segundos = "0" + segundos;
        }
        String Hora = hora + ":" + minuto + ":" + segundos;

        try
        {
            PreparedStatement pst=cn.prepareStatement("INSERT INTO
tablaparametros VALUES(?,?,?,?,?,?,?,?,?,?)");
            pst.setString(1, Fecha);
            pst.setString(2, Hora);
            pst.setInt(3, hsuelomin);
            pst.setInt(4, hsuelomax);
```

```
            pst.setInt(5, hinvmin);
            pst.setInt(6, hinvmax);
            pst.setBoolean(7, bomba);
            pst.setBoolean(8, ventilador);
            pst.setBoolean(9, luz);
            pst.setBoolean(10, automatico);

            pst.executeUpdate();
        }
        catch (SQLException ex)
        {
            Logger.getLogger(Srav.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
    public void borrarregistro()
    {
        try
        {
           Statement stm = cn.createStatement();
            stm.executeUpdate("TRUNCATE tablasensores");
        }
        catch (SQLException ex)
        {
            Logger.getLogger(Srav.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
}
```

15.5 Anexo VI: Fotografías

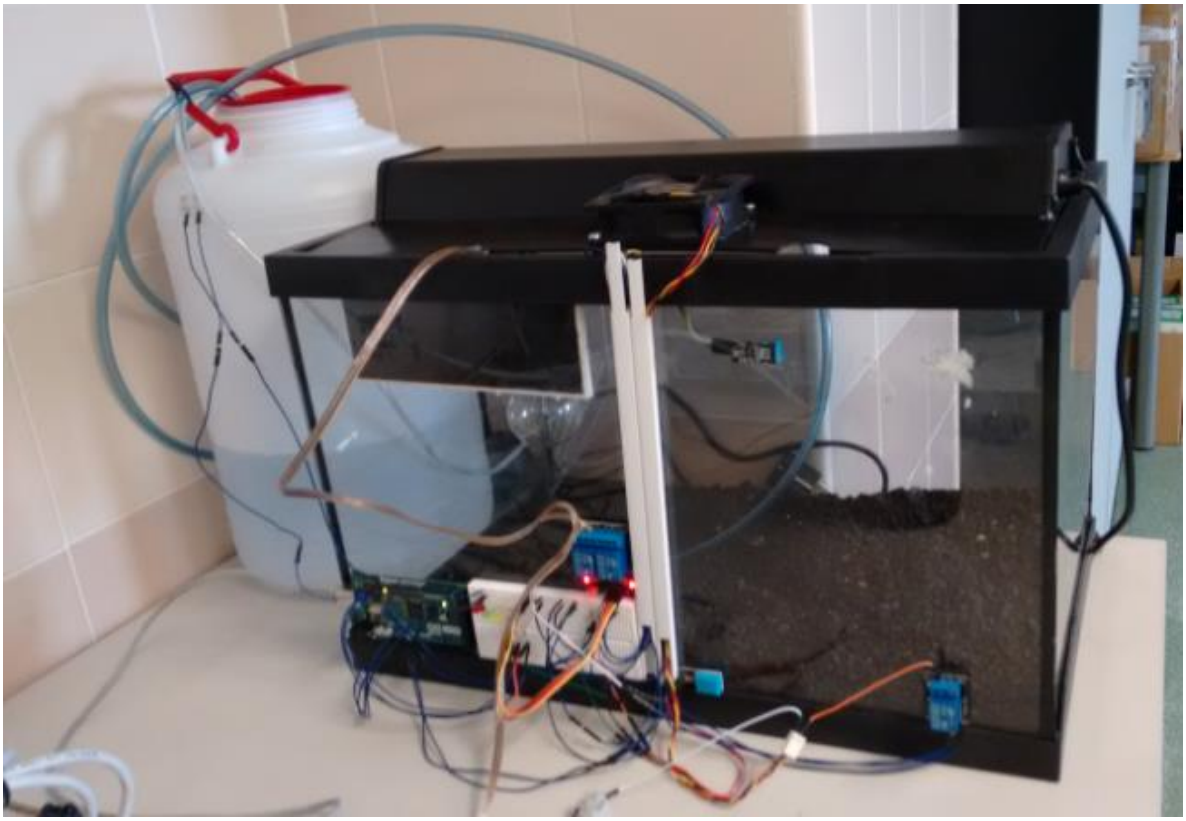


Figura 15-23 Fotografía 1

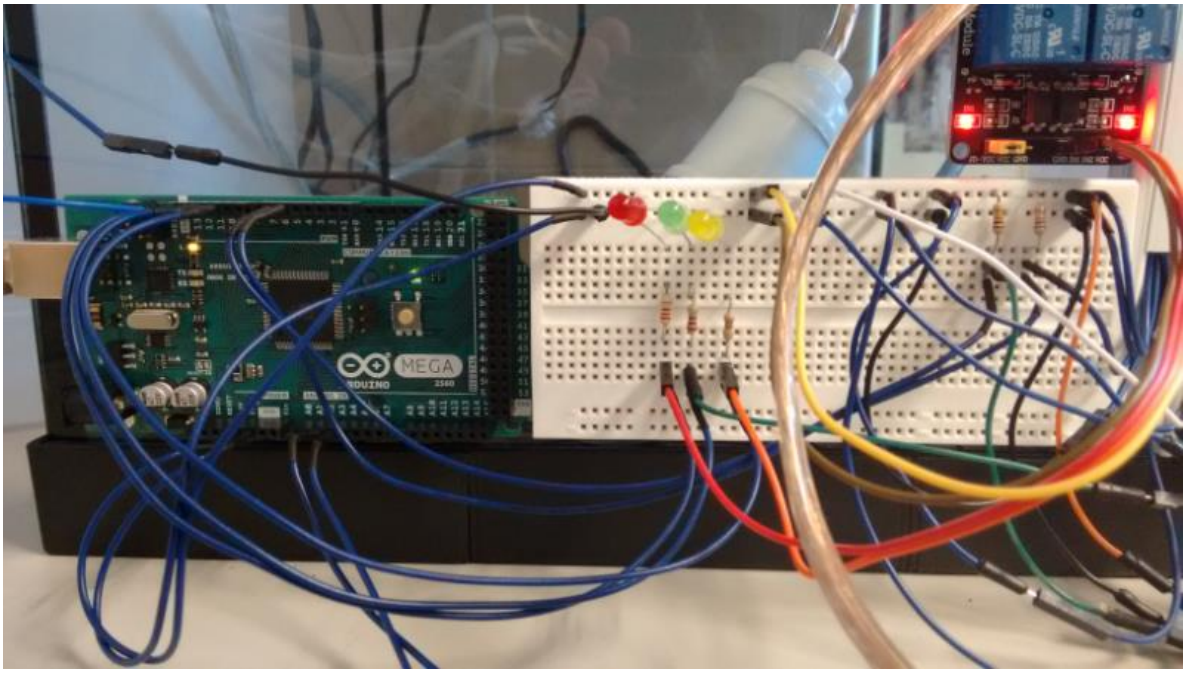


Figura 15-24 Fotografía 2