Jose Antonio Hernández Martínez

SISTEMAS Y SOLUCIONES PARA EL REGADIO

TESIS DE MÁSTER

dirigida por el Dr. Joan Fons i Cors

Departamento

de Sistemas Informáticos y Computación

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Valencia

2012

Agradecimientos

Especialmente a mi familia, sobre todo a mis padres que me han mostrado siempre su cariño y apoyo.

A Joan Fons por su afecto, dedicación y confianza.

A mi cuadrilla de amigos por los grandes momentos vividos y por los que aún nos quedan.

A mis compañeros de Tragsa por el apoyo y el clima de trabajo. También agradecer a la dirección de la delegación de Valencia las facilidades otorgadas para realización del máster y su programa de I+D.

Tabla de contenido

1. INTRODUCCIÓN 11

1.1. MOTIVACIÓN 11

1.2. OBJETIVOS DE LA TESIS 12

1.3. ESTRUCTURA DE LA TESIS 12

2. CITRICULTURA VALENCIANA 15

2.1. INTRODUCCIÓN 15

2.1.1. Territorio 17

2.1.1.1. Clima 17

2.1.1.2. Suelo 24

2.1.2. Taxonomía 25

2.1.2.1. Naranjo dulce 26

2.1.2.2. Mandarinos 28

2.1.2.3. Híbridos 30

2.1.2.4. Pomelos 32

2.1.2.5. Limoneros 33

2.1.2.6. Limeros 33

2.1.2.7. Especies y variedades de interés ornamental 34

2.2. ANÁLISIS DEL SECTOR 36

2.2.1. Situación actual 36

2.2.1.1. Organismos 36

2.2.1.1.1. Autonómicos 36

2.2.1.1.2. Estatales 37

2.2.1.1.3. Europeos 37

2.2.1.2. Modelo de negocio 40

2.2.1.2.1. Cooperativismo 40

2.2.1.2.2. Compra venta 41

2.2.1.2.3. Comercio electrónico 42

2.2.1.3. Tecnología actual 43

2.2.2. Futuro a corto y medio plazo 48

2.3. MOTIVACIÓN 52

3. CONTEXTO TECNOLÓGICO 53

3.1. SISTEMAS DE INFORMACIÓN UBICUOS 53

3.1.1. Introducción 53

3.1.2. Origen *54*
3.1.3. Características *54*
3.1.4. Contexto *55*
3.1.5. Arquitecturas *57*
3.1.6. Sistemas de inteligencia ambiental *59*
3.2. ARDUINO 60
3.2.1. Introducción *60*
3.2.2. Orígenes *60*
3.2.3. Características *61*
3.2.3.1. Código abierto *62*
3.2.3.2. Multiplataforma *62*
3.2.3.3. Coste *62*
3.2.3.4. Consumo *63*
3.2.3.5. Extensibilidad *63*
3.2.3.6. Comunidad *63*
3.2.3.7. Entorno de desarrollo *64*
3.2.3.8. Hardware *65*
3.2.3.8.1. Arduino UNO *66*
3.2.3.8.2. Arduino Mega 2560 ..... *69*
3.2.3.8.3. Arduino Mega ADK *70*
3.2.3.8.4. Arduino Nano *71*
3.2.3.8.5. Arduino Mini *71*
3.2.3.8.6. Arduino LilyPad *73*
3.2.3.8.7. Arduino Pro *74*
3.2.3.8.8. Arduino Pro Mini *75*
3.2.3.8.9. Arduino Ethernet *76*
3.2.3.8.10. Comparativa *77*
3.2.3.8.11. Shields *79*
3.2.4. Futuro *82*
3.3. MOTIVACIÓN 86
4. SYSREG: SISTEMAS Y SOLUCIONES PARA EL REGADÍO 89
4.1. INTRODUCCIÓN 89
4.2. DESCRIPCIÓN 90
4.3. CARACTERÍSTICAS FUNCIONALES 90
4.3.1. Aprovechamiento de materias primas *91*
4.3.2. Productividad *91*

4.3.3. Tiempo real ... *92*
4.3.4. Movilidad ... *92*
4.3.5. Procesamiento ... *92*
4.3.6. Inteligencia ... *92*
4.4. REQUISITOS NO FUNCIONALES ... 93
4.4.1. Extensibilidad ... *93*
4.4.2. Coste ... *94*
4.4.3. Open-source ... *94*
4.4.4. Multiplataforma ... *94*
5. ANÁLISIS Y DISEÑO DEL PROYECTO SYSREG ... 95
5.1. INTRODUCCIÓN ... 95
5.2. ANÁLISIS ... 95
5.2.1. Diagrama de contexto ... *95*
5.2.2. Fase 1: RegAdmin ... *96*
5.2.3. Fases restantes ... *97*
5.3. DISEÑO ... 98
5.3.1. Arquitectura ... *98*
5.3.2. Fase 1: RegAdmin ... *99*
5.3.2.1. Arquitectura hardware ... *100*
5.3.2.2. Arquitectura Software ... *101*
5.3.2.1. Diagrama de clases ... *103*
5.3.2.2. Casos de uso ... *103*
6. IMPLEMENTACIÓN DE LA FASE REGADMIN ... 105
6.1. PLANIFICACIÓN ... 105
6.2. ACTIVAR/DESACTIVAR RIEGO ... 106
6.2.1. Hardware ... *106*
6.2.2. Software ... *110*
6.2.2.1. AlReg ... *110*
6.2.2.2. RegAdmin ... *111*
6.3. OBTENER TEMPERATURA ... 112
6.3.1. Hardware ... *112*
6.3.2. Software ... *114*
6.3.2.1. Alreg ... *114*
6.3.2.2. RegAdmin ... *115*
6.4. OBTENER HUMEDAD RELATIVA DEL AIRE ... 117

6.4.1. Hardware ... *117*
6.4.2. Software ... *118*
6.4.2.1. Alreg ... *118*
6.4.2.2. RegAdmin ... *119*
6.5. OBTENER PRESIÓN ATMOSFÉRICA ... 120
6.5.1. Hardware ... *120*
6.5.2. Software ... *121*
6.5.2.1. Alreg ... *121*
6.5.2.2. RegAdmin ... *123*
6.6. OBTENER HUMEDAD DEL SUELO ... 124
6.6.1. Hardware ... *124*
6.6.2. Software ... *126*
6.6.2.1. Alreg ... *126*
6.6.2.2. RegAdmin ... *126*
6.7. PROGRAMAR RIEGO ... 127
6.7.1. Software ... *127*
6.7.1.1. RegAdmin ... *127*
6.8. ESTABLECER ALARMAS ... 133
6.8.1. Software ... *133*
6.8.1.1. Alreg ... *133*
6.8.1.2. RegAdmin ... *136*
6.9. OTRAS CARACTERÍSTICAS ... 138
6.9.1. Evitar reseteo automático ... *138*
6.9.2. Indicador led ... *140*
6.9.3. Actualización automática de sensores ... *141*
6.9.4. Generación de versiones personalizadas para los Windows y OS X ... *142*
6.9.4.1. OS X ... *143*
6.9.4.1. Windows ... *144*

7. DESPLIEGUE ... 147
7.1. INTRODUCCIÓN ... 147
7.2. ESCENARIO JARDÍN ... 149

8. PROYECTO SIGAM ... 153
8.1. INTRODUCCIÓN ... 153
8.2. GRUPO TRAGSA Y TRAGSATEC ... 153
8.3. PROYECTOS ANTERIORES ... 156

8.3.1. SIAR ... *157*
8.3.1.1. Objetivos ... *157*
8.3.1.2. Descripción ... *157*
8.3.1.2.1. Estaciones agroclimáticas. ... *159*
8.3.1.2.2. Evapotranspiración de referencia según FAO Penman-Monteith ... *160*
8.4. SIGAM ... 179
8.4.1. Descripción técnica ... *179*
8.4.2. Presupuesto ... *180*
9. ASPECTOS TECNOLÓGICOS UTILIZADOS ... 183
9.1. RATIONAL UNIFIED PROCESS ... 183
9.2. SUBVERSIÓN ... 185
9.3. GOOGLE CODE ... 188
9.4. FRITZING ... 190
9.5. ENTORNO DE DESARROLLO ... 193
9.6. MICROCONTROLADORES: AVR Y PIC ... 195
9.7. USB *HOST* Y USB *DEVICE* ... 197
9.8. LIBRERÍAS ... 197
9.8.1. I²C ... *197*
9.8.2. OneWire ... *199*
9.8.3. FreqCounter ... *200*
9.8.4. Time ... *201*
9.8.5. TimeAlarms ... *202*
9.8.6. RXTX ... *203*
9.8.7. Google API y google-api-java-client ... *204*
10. CONCLUSIONES ... 207
11. REFERENCIAS ... 209

1. INTRODUCCIÓN

1.1. Motivación

En los últimos años estamos asistiendo a un abandono de las parcelas minifundistas de la Comunidad Valenciana. Este desvanecimiento del sector se debe a la elongación de la cadena de distribución (con un alto número de intermediarios que originan un alto precio para el consumidor y pérdidas para el agricultor), el estancamiento tecnológico del sector y la actual coyuntura económica y política, cuyos recortes han acentuado los problemas.

Parte de esta situación está originada por el envejecimiento de la población activa (tan solo el 2% son jóvenes agricultores), reacia a introducir nuevos métodos y tecnologías.

Además el aumento de los costes de producción, en materias primas como el petróleo, el agua de riego y los abonos, ha provocado que los márgenes económicos se vean reducidos aún más.

Por si esto fuera poco el cambio climático ha agudizado fenómenos meteorológicos extremos, como la gota fría, las heladas y las olas de calor, que provocan perdidas de producciones.

En este contexto tan difícil se enmarca la presente tesis, que pretende desarrollar un sistema agrario, extensible, remoto e inteligente cuyos objetivos son:

- Aumentar la productividad de los agricultores, mejorando así la competitividad de las parcelas agrarias.
- Acceso en tiempo real, monitorizando de una forma rápida y precisa las condiciones climáticas y activando instantáneamente actuadores a distancia.
- Optimizar el uso de las materias primas.
- Incrementar el porcentaje de jóvenes agricultores de la mano de las tecnologías de la información, con la utilización de ordenadores personales, dispositivos móviles e Internet. Esta motivación no excluye al resto de agricultores, sino todo lo contrario, la implantación de estos sistemas favorecerá el uso de nuevas tecnologías.

- Uso de tecnologías libres y de bajo coste. El sistema que se presenta se fundamenta en tecnologías libres y baratas: el proyecto de hardware libre Arduino y un conjunto de aplicaciones *Open Source*.

1.2. Objetivos de la tesis

Esta tesina tiene los siguientes objetivos:

- Analizar las carencias tecnológicas de la citricultura valenciana.
- Ensamblar un dispositivo totalmente funcional que cubra las necesidades detectadas.
- Diseñar e implementar el software necesario para interactuar con el dispositivo remotamente.
- Demostrar la mejora productiva conseguida con la utilización de este sistema.
- Preparar el sistema para futuras extensiones a diferentes variedades y sectores.

1.3. Estructura de la tesis

A continuación se detalla la estructura de la tesis:

- Capítulo 1: La presente introducción.
- Capítulo 2: Estudio del sector citrícola valenciano, en primer lugar definiendo la citricultura, particularizando a continuación, el estado del arte en la Comunidad Valenciana, y en último lugar, analizando la situación actual.
- Capítulo 3: Se presentan las principales tecnologías utilizadas en el proyecto, haciendo hincapié en los conceptos técnicos de Arduino.
- Capítulo 4: Descripción de las principales características del proyecto Sysreg.
- Capítulo 5: Análisis y diseño de los requisitos funcionales, profundizando en las características incluidas en la primera fase del proyecto.
- Capítulo 6: Implementación de los casos de uso de la fase RegAdmin, detallando tanto los componentes hardware como software.
- Capítulo 7: Mediante un despliegue en un escenario puntual, se realiza un caso de estudio del sistema desarrollado.

- Capítulo 8: Se detalla una propuesta de investigación y desarrollo presentada en la empresa Tragsatec.
- Capítulo 9: Descripción de otros aspectos tecnológicos utilizados en el proyecto.
- Capítulo 10: Conclusiones y valoración personal de la experiencia.

2. CITRICULTURA VALENCIANA

En esta sección analizaremos en profundidad el sector agrícola valenciano. Para ello definiremos la citricultura, estudiaremos las características territoriales de la Comunidad Valenciana, su adecuación al cultivo de agrios y clasificaremos las principales variedades de cítricos. Además estudiaremos los organismos, públicos y privados, involucrados en el sector para poder terminar identificando las oportunidades de negocio.

Para el desarrollo de este capítulo nos hemos basado en el libro especializado en citricultura de M. Agustí [1], así como en la documentación extraída de la Conselleria de Agricultura [2] y del IVIA [3].

2.1. Introducción

La citricultura es la rama de la fruticultura que estudia el cultivo y características de un grupo de plantas denominadas cítricos. Este nombre se debe a que la mayor parte de los frutales incluidos forman parte del género *Citrus*, aunque también existen especies del género *Fortunella, Ponciurus trifoliata* y diversos híbridos.

El origen de los cítricos se localiza en Asia oriental (China, Tailandia, Malasia e Indonesia). Los datos más antiguos datan del siglo XXIII a. C.. A España los primeros cítricos en llegar fueron los naranjos y los limoneros a manos de los árabes allá por el siglo XI. En cambio el mandarino se introdujo mucho más tarde, en 1856.

Los cítricos, también llamados agrios, se cultivan en regiones tropicales y subtropicales, siendo actualmente los frutos de mayor producción en el mundo (unas 85 millones de toneladas, frente a las 75 del plátano y 50 de manzanas). España es el principal productor de cítricos del Mediterráneo con unos 5 millones de Tm (toneladas).

En la Comunidad Valenciana (CV) es la principal actividad agraria, con unas 170.000 ha (hectáreas) y una producción media de 3 millones de Tm. Las plantaciones en la CV se caracterizan por su reducido tamaño (una media de 0,97 ha para naranjos y 0,73 ha para mandarinos), el cultivo a tiempo parcial y la falta de mano de obra puntual.

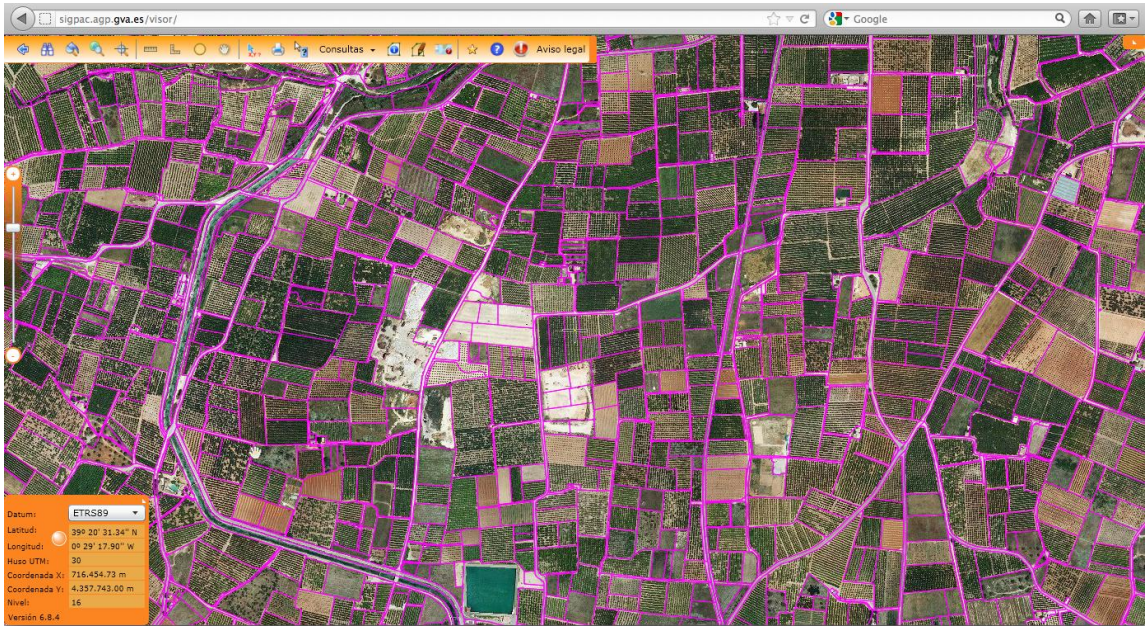


FOTO 2.1: ASPECTO MINIFUNDISTA DEL CAMPO VALENCIANO EN EL VISOR SIGPAC DE LA COMUNIDAD VALENCIANA

Si analizamos jerárquicamente el sector agrícola (que representa en el 77% de la producción final agraria, es decir, agricultura, pesca y ganadería) podemos estimar su valor de producción global en 2243 millones de euros. Esta actividad se desarrolla sobre 688.400 ha cultivables, el 30% del territorio disponible, distribuidos por un 51% en Valencia, un 27% en Alicante y un 22% en Castellón. El principal subsector son los cítricos con el 31%, seguido por un 24% de otros frutales, quedando por debajo los olivares, viñedo, cereales y hortalizas (incluyendo flores y plantas de vivero). Sin embargo proporcionalmente la aportación de los cítricos al valor final es mucho mayor: aportan el 50% de la producción.

Pero su importancia no recae únicamente en el área económica. La citricultura tiene un alto valor ecológico: los cítricos contribuyen a reducir el efecto invernadero y a combatir el cambio climático, puesto que son los árboles de hoja perenne que más gases absorben durante la fotosíntesis. Las estimaciones hablan de que los naranjos valencianos captan 900.000 Tm de dióxido de carbono (unos 150.000 vehículos consumiendo 300 millones de litro de gasóleo). Además ayudan al desarrollo del medio rural valenciano, ya que tienen en la actividad citrícola la principal fuente de ingresos.

Estos son los principales motivos por los cuales la presente tesis pretende mejorar el sector agrario.

En los siguientes puntos realizaremos un análisis del territorio valenciano para poder entender las condiciones climáticas y sus variables a estudiar, y repasaremos las principales variedades de agrios.

2.1.1. Territorio

El desarrollo de la citricultura en la región valenciana viene determinado por diversos factores. El más importante es el factor territorial, que le proporciona características climáticas y de suelo ideales para el cultivo de cítricos:

2.1.1.1. Clima

El clima es un factor crítico en el desarrollo de las plantas, es un limitante para su cultivo y es difícilmente modificable.

Los diferentes factores ambientales afectan al desarrollo de las plantas, pero su influencia es difícil de estudiar, ya que el estudio de un solo factor (por ejemplo la temperatura) es una simplificación, debido a que la respuesta real se debe a la interacción entre diferentes factores. Por ello a continuación detallaremos los factores climáticos más importantes y los compararemos con los presentes en la Comunidad Valenciana.

- Latitud

Los cítricos se desarrollan entre los 40º N y 40º S de latitud, aunque las plantaciones comerciales se encuentran en lar regiones subtropicales (entre los 20º y 40º de los hemisferios norte y sur) debido a que la temperatura es modulada por los vientos marinos.

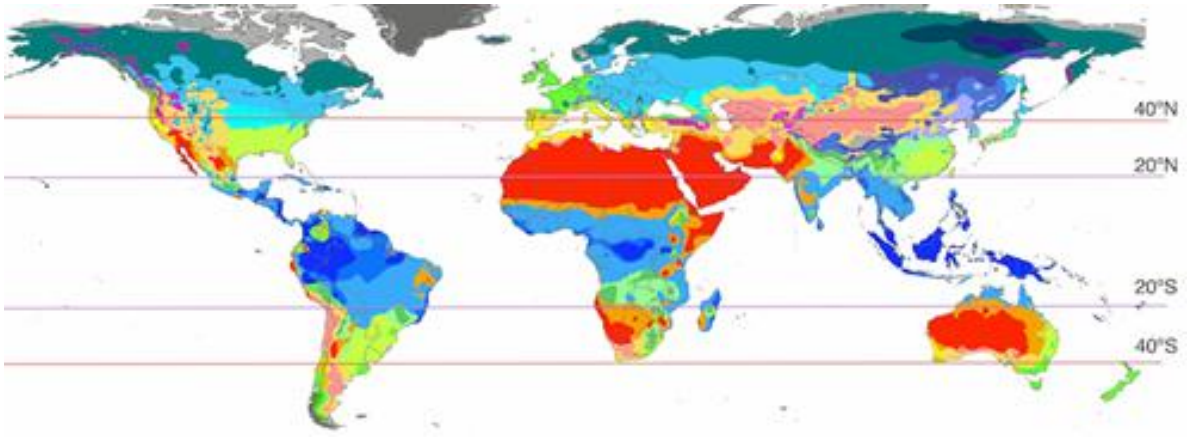


FIGURA 2.1: MAPA MUNDIAL DE TEMPERATURAS MEDIAS CON LAS ÁREAS VÁLIDAS PARA EL CULTIVO DE CÍTRICOS

Los agrios morfológicamente presentan una estructura mesofítica, con las yemas casi desnudas, hojas anchas y de poco espesor, estomas superficiales ausencia de pelos y cutícula fina.

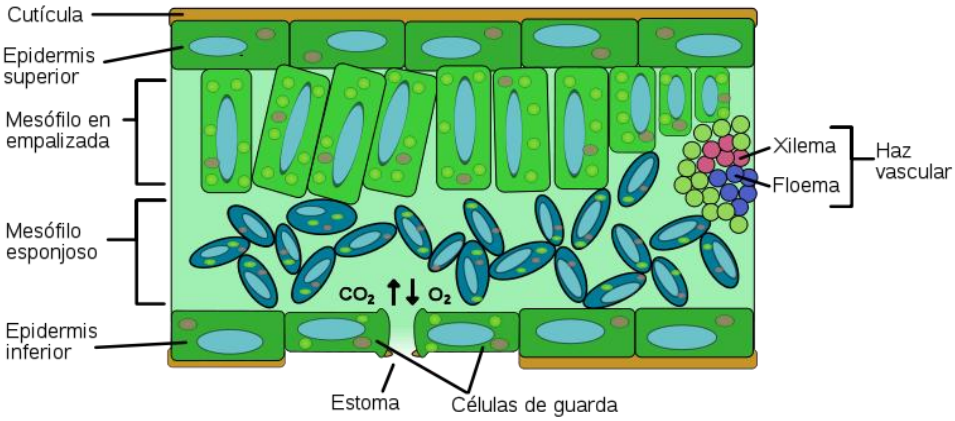


FIGURA 2.2: ESTRUCTURA FISIOLÓGICA DE LAS HOJAS DE LOS CÍTRICOS

Esta delicadeza estructural requiere de un medio húmedo, tanto en suelo como en atmósfera, con unas temperaturas cálidas pero suaves. Estas características se dan en las áreas subtropicales, mientras que en áreas tropicales los frutos no alcanzan la coloración plena.

La Comunidad Valenciana se encuentra entre los 40º y 37º del hemisferio norte, otorgándole unas condiciones climáticas determinantes para conseguir una buena producción y calidad del fruto.

- Altitud

La altura, en función de la latitud, es un factor limitante: en trópicos es posible cultivar cítricos a altitudes de 1500 m, aunque con corteza bastante basta. En cambio en las regiones subtropicales el límite se encuentra en los 500-600 m. La altitud afecta principalmente al fotoperiodo y las variaciones térmicas estacionarias.

En Valencia las principales comarcas de cultivo, como son *l'Horta* y *la Ribera*, se encuentran entre el nivel del mar y los 300 m. Incluso las zonas límites no superan 700 m.

- Temperatura

Es la variable climática más importante, determinante en el desarrollo vegetativo, la floración y el cuajado.

Las temperaturas entre 25 ºC y 30 ºC se consideran óptimas para la actividad fotosintética, ya que por encima de 35º ésta se reduce.

Las regiones tropicales del ecuador tienen periodos repetidos de floración mientras que las zonas subtropicales, con estaciones bien definidas, cuentan con un desarrollo controlado por los cambios estacionales. Así pues en las áreas subtropicales los agrios en invierno presentan un proceso de reposo y brotan uniformemente en primavera, con el aumento de temperatura.

La temperatura también afecta al cuajado de las flores. Un alto grado de floración no garantiza un alto grado de cosecha. Se considera que valores entre 11º y 22º favorecen la producción de polen y el cuajado.

Durante el desarrollo del fruto, temperaturas bajas (por debajo de 20º) y temperaturas superiores a 30º nocturnos dan lugar a frutos pequeños. El crecimiento del fruto tiende a su máxima expresión con temperaturas combinadas día/noche entre 20º y 25º. Es tan importante la temperatura en el desarrollo del fruto que su color viene determinado por las variaciones del fruto: El fruto crece con un color verde a temperaturas altas constantes (contiene niveles altos de clorofila) y no empieza a cambiar de color hasta estar por debajo de los 13º, es por esto que los cítricos empiezan a volverse naranja a partir de mediados de septiembre con la llegada del otoño.

Además del color, la temperatura afecta directamente a la acidez, de manera que cuanto más alto es el régimen térmico día/noche más baja en la concentración de ácidos:

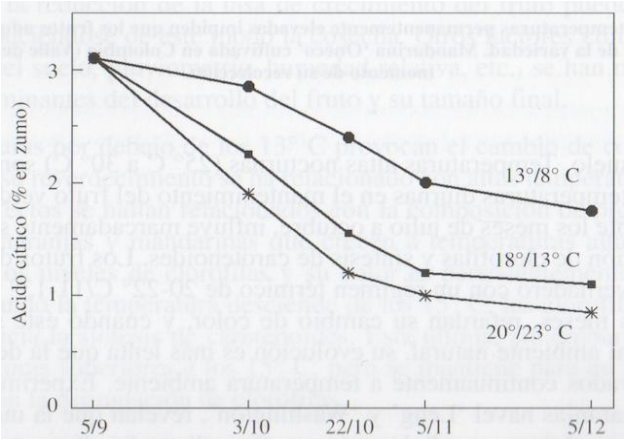


FIGURA 2.3: EFECTO DEL RÉGIMEN TÉRMICO DURANTE EL PERIODO DE RECOLECCIÓN SOBRE LA ACIDEZ

La rugosidad de la corteza y la forma del fruto también se encuentran relacionadas con la temperatura. Los climas secos y con grandes amplitudes térmicas producen frutos más rugosos y con cuello más alargado que los climas húmedos y con escasas oscilaciones.

En cuanto a límites de térmicos podemos afirmar que los cítricos pueden vivir sin sufrir daños importantes a temperaturas entre 0º y 50º, siendo por debajo de 0º la actividad vegetativa nula y únicamente por encima de los 50º dañina para el árbol. El rango óptimo se considera entre 23º y 34º y sin efectos secundarios entre 10º y 39º.

Debido a la situación geográfica de la Comunidad Valenciana el clima predominante es el clima mediterráneo, óptimo para el cultivo de cítricos, aunque con diferentes variantes según la zona:

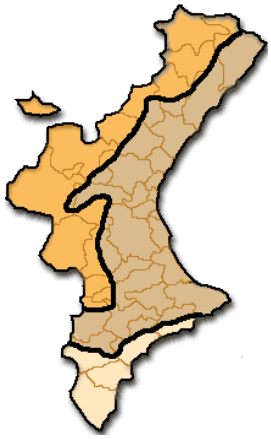


FIGURA 2.4: CLIMAS MEDITERRÁNEOS DE LA CV: MEDITERRÁNEO CONTINENTAL (NARANJA), MEDITERRÁNEO SECO (BLANCO) Y MEDITERRÁNEO TÍPICO (MARRÓN)

Por el litoral encontramos el clima Mediterráneo típico con inviernos suaves (13,5º) y veranos largos, secos y calurosos con máximas en torno a los 30º. Más hacia el interior encontramos el clima mediterráneo continentalizado que se diferencia al tener inviernos más fríos y veranos más cálidos, con máximas de 35º. Y por último en el sur de Alicante tenemos el clima mediterráneo seco con temperaturas similares al típico pero con escasas precipitaciones.

- Humedad relativa (HR)

Los cítricos se caracterizan por adaptarse moderadamente bien a diferentes valores de humedad.

Los estudios afirman que los descensos bruscos de la humedad relativa producen la caída fisiológica de frutos, considerándose el 37% como un valor crítico. Por otro lado la elevación de la temperatura y de la HR después del cambio de color del fruto provocan efectos como el *pixat* o *water spot* (aguado del fruto en condiciones de lluvia) y el *bufado* (separación de la pulpa y la corteza) y variaciones altas de estas estimulan el rajado o *splitting*. Bajo condiciones de elevada HR los frutos obtienen una mejor coloración y una forma más aplanada.

- Lluvia

Teniendo en cuenta las pérdidas por evapotranspiración, las necesidades hídricas de los agrios se encuentran entre 7.500 y 12.000 m^3 (metros cúbicos = kilolitros) por hectárea, lo equivalente a una pluviometría anual entre 750 y 1.200 mm (litros caídos por m^2), siendo necesaria una adecuada distribución.

La lluvia, además de afectar obviamente a la temperatura y la humedad relativa, ejerce su influencia en la humedad del suelo, aunque la falta de precipitaciones puede se sustituida por riego. Eso sí, los años lluviosos suelen tener mejores cosechas y frutos.

En épocas primaverales es importante la necesidad de tener periodos de estrés hídrico para favorecer la floración.

Parámetros que determinan la calidad del fruto, como la acidez y los sólidos solubles, se ven reducidos en épocas de copiosas lluvias:

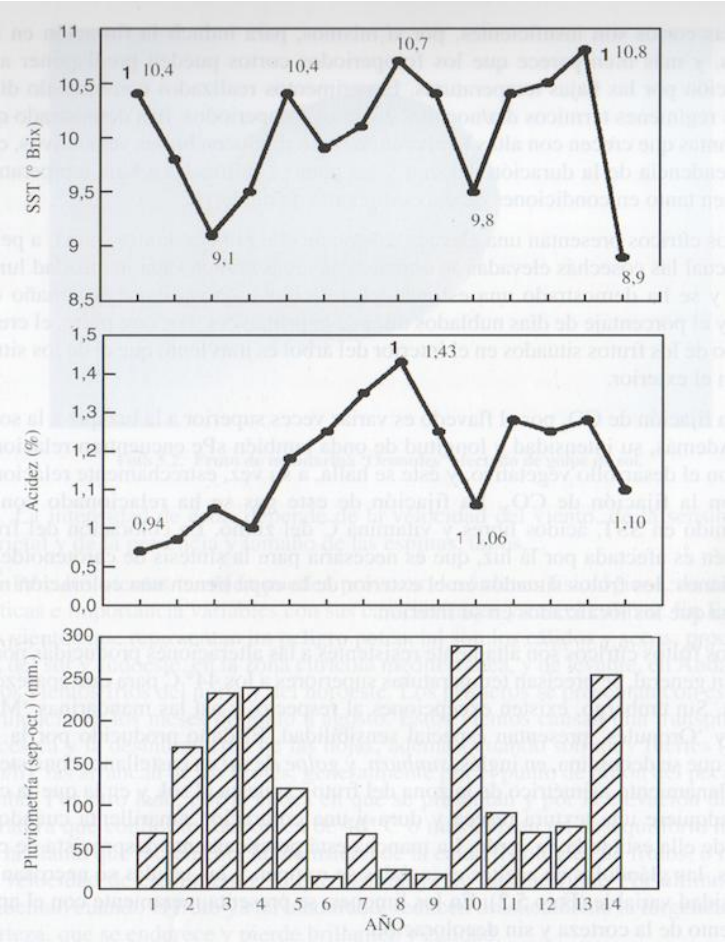


FIGURA 2.5: INFLUENCIA DE LA PLUVIOMETRÍA SOBRE EL CONTENIDO EN SÓLIDOS SOLUBLES Y LA ACIDEZ

En cuanto a aspectos negativos por la lluvia tenemos que el exceso de lluvia puede provocar daños en el árbol, por inundaciones del suelo, o sobre el fruto, con daños sobre la corteza.

Pero en la Comunidad Valenciana no se dan las condiciones ni de intensidad ni de distribución apropiada. Hablamos de precipitaciones en torno a una media de 700 mm anuales (con zonas áridas con 250 mm anuales y zonas lluviosas con más de 1.000 mm anuales). Además se caracterizan por ser irregulares y concentradas en otoño y primavera.

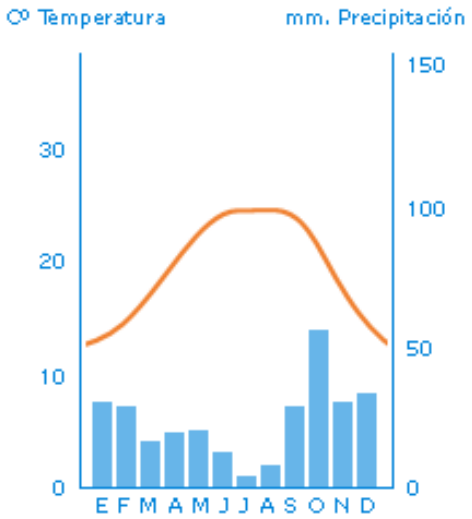


FIGURA 2.6: PRECIPITACIONES EN EL SUR DE ALICANTE

- Insolación

La intensidad y la duración de la luz (fotoperiodo) son determinantes para la extracción de energía, a través de la fotosíntesis, que posteriormente se acumula en los enlaces de carbono de los carbohidratos. Por ello las cosechas elevadas se obtienen en áreas con elevada intensidad luminosa, siendo este efecto patente en el escaso crecimiento del fruto del interior del árbol.

Además los altos periodos de luz favorecen el fijado de CO_2 y el aumento de vitamina C en el zumo.

En cuanto a problemas lumínicos solo encontramos problemas con determinadas variedades sensibles, como la Oronules, que pueden sufrir daños en el fruto con zonas de textura reseca, dura y con una coloración amarillenta (el llamado golpe de sol o *sunburn*).

- Viento

Es el factor abiótico (factor climático inerte) más importante, ya que produce daños mecánicos y lesiones en la corteza de los frutos.

Vientos por encima de una velocidad de 25 km/h se consideran dañinos, ya que producen heridas en las ramas, hojas y frutos. Por otro lado los vientos

cálidos y secos provocan una transpiración excesiva, secan la tierra y deshidratan las hojas. Y los vientos fríos, frecuentes entre noviembre y enero, provocan la caída de los frutos maduros (sobretodo de las naranjas).

En la Comunidad Valenciana los vientos más dañinos son los del sur, sudoeste y oeste en verano, por su fuerza y temperatura, y en los meses invierno los del norte y noreste

FIGURA 2.7: ROSA DE VIENTOS DE LA COMUNIDAD VALENCIANA

2.1.1.2. Suelo

El suelo aporta el soporte físico y los elementos esenciales (oxígeno y agua).

Los cítricos se pueden desarrollar bajo condiciones edáficas muy diferentes: desde suelos pedregosos (pobres) hasta suelos arcillosos y pesados. Sin embargo su desarrollo vegetativo y producción óptimos se dan en suelos arenosos (de textura gruesa: baja proporción de arcillas, permeables, buena circulación del aire y del agua) y suelos francos (con textura media: 45% de arena, 40% de limo y 15% de arcilla, considerados como los óptimos para el cultivo). En cambio los suelos arcillosos dificultan el crecimiento de las raíces.

Además el suelo ha de ser profundo, para facilitar un desarrollo que garantice un buen anclaje del árbol y una amplia exploración radicular.

En Valencia suelo es equilibrado, con colores oscuros y rojizos que denotan gran cantidad de materia orgánica, gran fertilidad y una alta proporción de óxidos de hierro.

2.1.2. Taxonomía

Las principales variedades comerciales de los cítricos se encuentran dentro de la familia de las *Rutaceas*, subfamilia *Aurantioideas.* Esta subfamilia se caracteriza por ser arboles o arbustos de hoja perenne (excepto el genero *Poncirus*, de hoja caduca). Concretamente los cítricos comerciales pertenecen a los géneros *Fortunella*, *Poncirus* y *Citrus:*

Familia: *Rutaceas*. Subfamilia: *Aurantioideas*

Tribu II: *Citreae.* 3 subtribus, 9 grupos subtribales, 28 géneros, 124 especies

Subtribu 2: *Citrinae.* Arboles cítricos. 3 grupos subtribales, 13 géneros, 65 especies

A: Cítricos primitivos: 5 gén., 14 sp.

XIV: *Severinia:* 6 especies.
XV: *Pleiospermium:* 5 especies
XVI: *Burkillanthus:* 1 especie
XVII: *Limnocitrus:* 1 especie
XVIII: *Hesperthusa:* 1 especie

B: Arboles cercanos a los cítricos: 2 gén., 22 sp.

XIX: *Citropsis:* 11 especies.
XX: *Atalantia:* 11 especies

C: Arboles cítricos verdaderos: 6 gén., 29 sp.

XXI: *Fortunella**: 4 especies.
XXII: *Eremocitrus:* 1 especie
XXIII: *Poncirus**: 1 especie
XXIV: *Clymenia:* 1 especie
XXV: *Microcitrus:* 6 especies
XXVI: *Citrus**: 16 especies

FIGURA 2.8: CLASIFICACIÓN BOTÁNICA DE LOS CÍTRICOS

Las especies del género *Citrus* son las más importantes agroalimentariamente hablando, ya que su cultivo representa la producción de frutos para consumo fresco y zumo. Las especies de mayor difusión son:

- *C. auroantifolia.* Lima mejicana.
- *C. latifolia.* Lima Tahití,
- *C. aurantium.* Naranjo amargo.
- *C. grandis. Pummelo*.
- *C. limon.* Limonero.
- *C. paradisi.* Pomelo.
- *C. reticulata.* Mandarina.
- *C. sinesis.* Naranjo dulce.

Este sistema taxonómico fue originado a finales del siglo XIX por Hooker y Engler, y revisado por Swingle a mediados de los años 60. Sin embargo, dada la complejidad y amplitud de los cítricos, esta no es la única clasificación botánica. Tanaka en 1977 propuso una clasificación más amplia del género *Citrus*. Actualmente la más utilizada es la de Swingle, aceptando la de Tanaka para algunas especies.

A continuación resumiremos las principales variedades de cítricos cultivadas en España, descendientes de las especies citadas anteriormente.

2.1.2.1. Naranjo dulce

De esta especie encontramos cultivadas en España tres grupos diferentes:

a) Grupo Navel

Se distinguen por presentar un segundo fruto muy pequeño incluido en la zona estilar. De esta característica toman su nombre, ya que se parece a un ombligo (*navel* en inglés significa ombligo).

FOTO 2.2: NARANJA NAVELATE

Destacan por no fecundar los óvulos y, por tanto, no presentar semillas.

Sus principales variedades son la *Navelina, Navelate, Lane late, Newhall, Washington navel, Navel Powell, Thompson navel.*

b) Grupo Blancas

Con una acidez baja, ausencia de *navel* en el fruto y alternancia de cosechas, fueron durante muchos años el cultivo más importante por sus posibilidades de industrialización en zumos.

Las únicas variedades que han persistidos han sido las que tienen calidad suficiente para el consumo fresco y entre estas están la *Salustiana* y *Valencia late.*

FOTO 2.3: NARANJA SALUSTIANA

c) Grupo Sanguíneas

De tamaño pequeño y alargado presentan un distintivo color rojo. Contienen semillas y un alto contenido en zumo.

Con una producción por debajo del 1% encontramos las variedades *Sanguinelli*, *Doblefina* y *Entrefina.*

FOTO 2.4: NARANJA SANGUINELLI

El periodo de recolección de las naranjas es el siguiente:

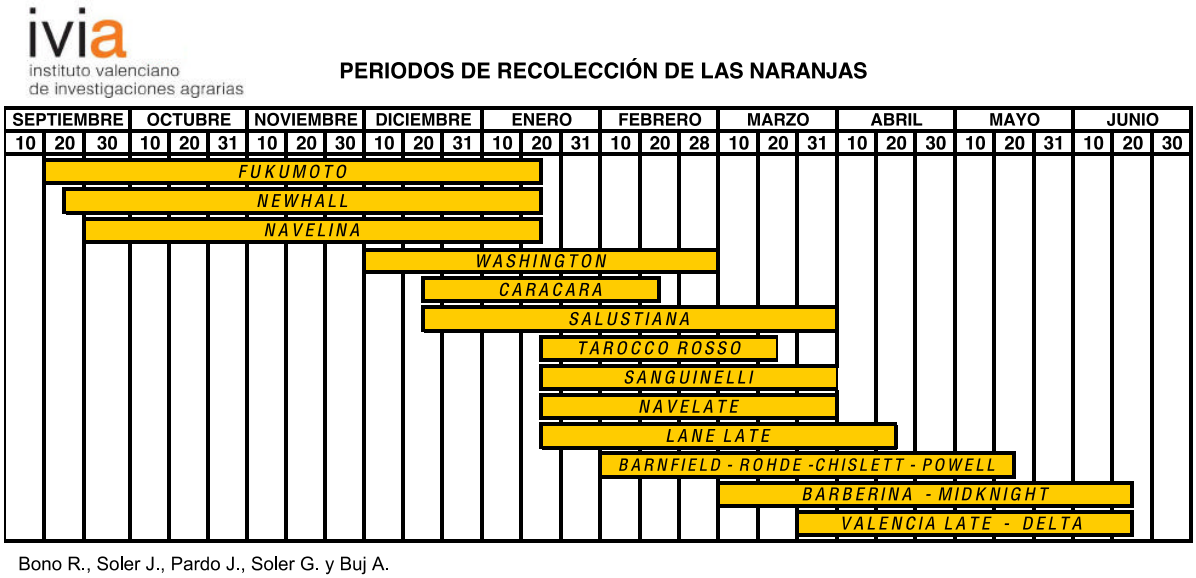


FIGURA 2.9: PERIODO DE RECOLECCIÓN DE LAS NARANJAS

2.1.2.2. Mandarinos

La compleja clasificación de las mandarinas, por su número de variedades, se puede resumir en 2 grandes grupos:

a) Mandarino Satsuma

Son mandarinas de un calibre grande, menor proporción de zumo y con semillas. Son más precoces y de peor calidad gustativa. Destacan por su facilidad para la conserva (e. g. gajos en almíbar) e industrialización.

Las principales variedades son la *Owari*, *Clausellina* y *Okitsu.*

FOTO 2.5: MANDARINA OKITSU

b) Mandarinos Clementinos

Poseen un sabor dulce, gran cantidad de zumo y tamaño entre pequeño y mediano.

El nombre Clementina tiene un significado más comercial que botánico, ya que se han originado por mutaciones espontaneas unas de otras, y cuyo origen inicial es el mandarino común (*Citrus reticulata* Blanco):

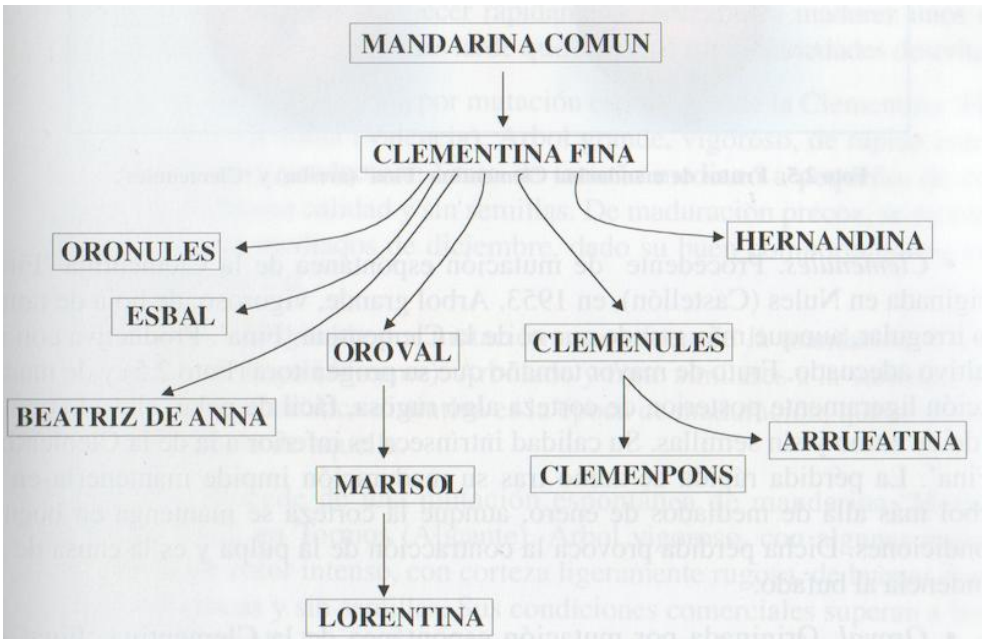


FIGURA 2.10: PRINCIPALES MUTACIONES DE MANDARINA CLEMENTINA APARECIDAS EN ESPAÑA

Entre las variedades más comerciales encontramos la *Clemenules, Marisol, Oronules, Esbal, Hernandina.*

FOTO 2.6: MANDARINA HERNANDINA

El periodo de recolección de los mandarinos es el siguiente:

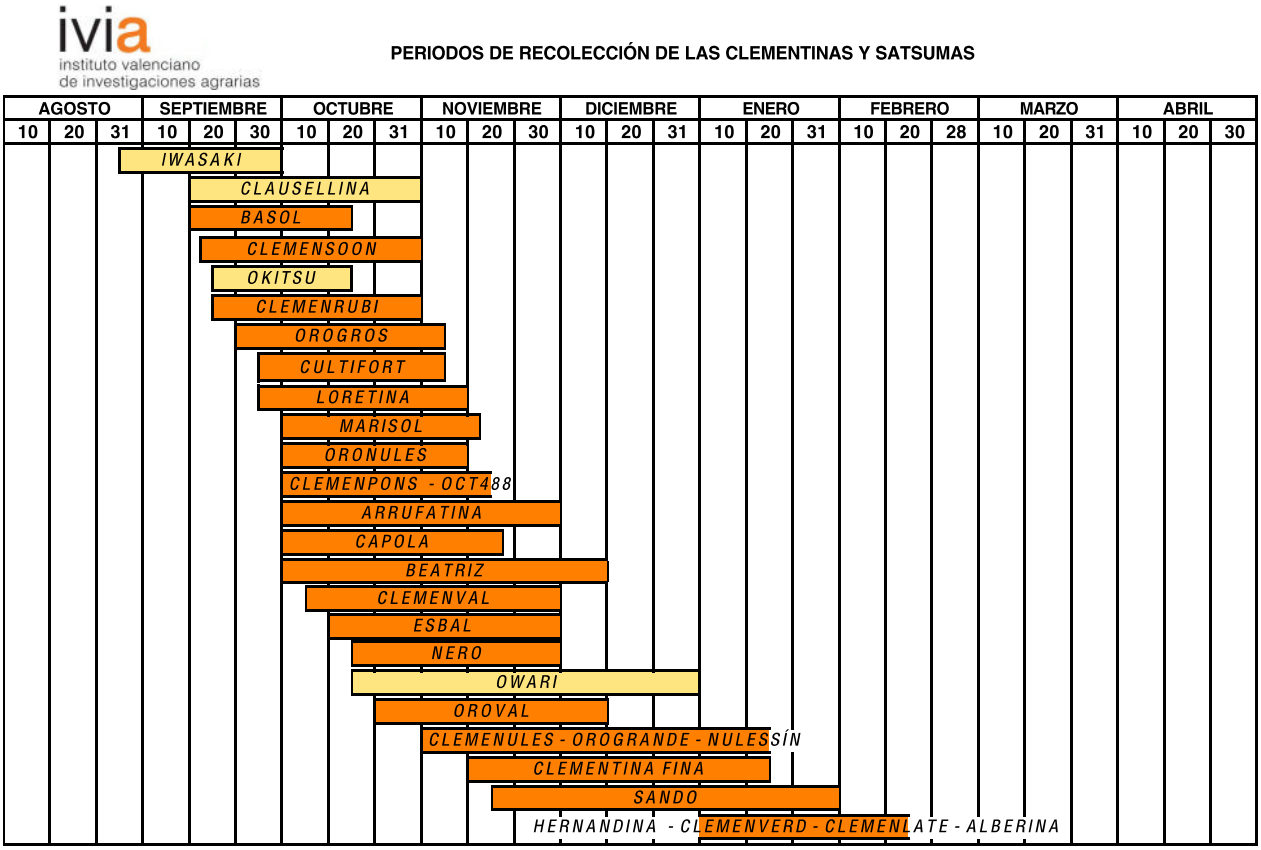


FIGURA 2.11: PERIODOS DE RECOLECCIÓN DE LAS CLEMENTINA Y SATSUMAS

2.1.2.3. Híbridos

Un híbrido es un organismo animal o vegetal procedente del cruce de dos géneros, especies o subespecies distintas.

Los agrios, como vimos al inicio del punto, son géneros descendientes de la familia *Rutaceas* que hibridan con mucha facilidad. Existen incluso híbridos intergenéricos (muy raros en el reino vegetal). Como podemos ver en la siguiente figura tenemos cruces entre géneros y entre especies:

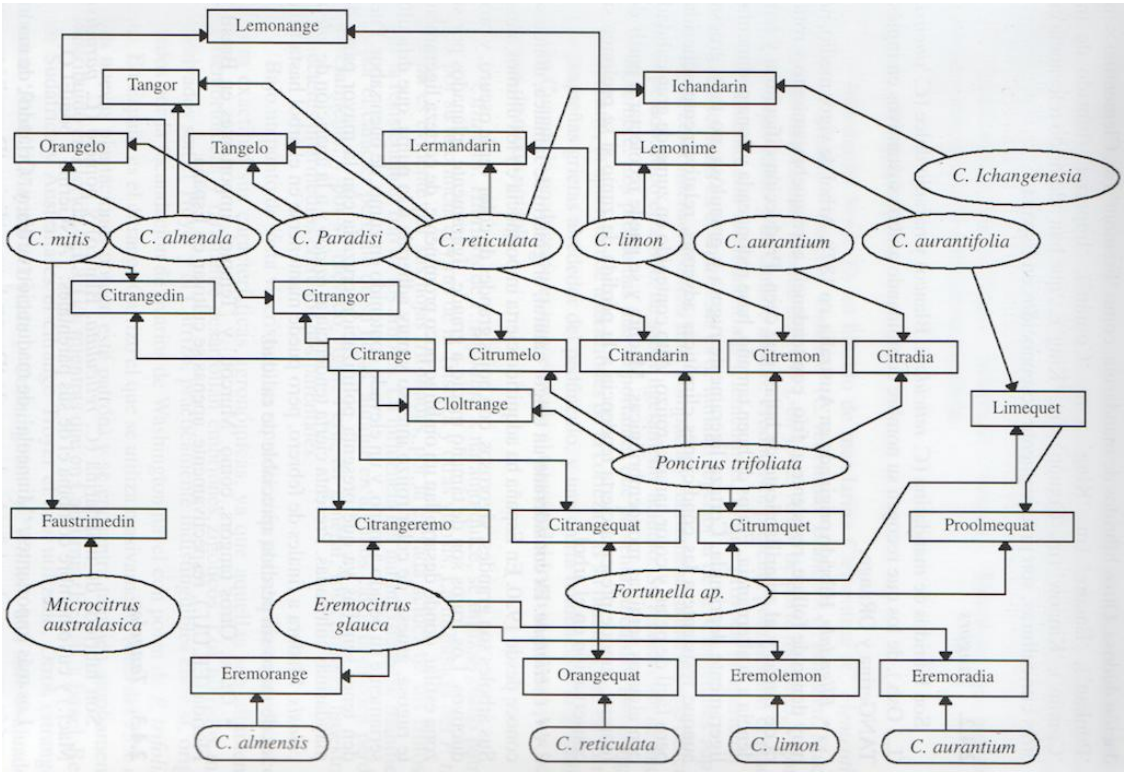


FIGURA 2.12: HÍBRIDOS INTERESPECÍFICOS E INTERGENÉRICOS DE LOS CÍTRICOS

Los más importantes son los híbridos de mandarino, donde destacan la *Fortune* (Clementina Fina y Mandarina Dancy) y la *Nardocott.*

FOTO 2.7: MANDARINA *FORTUNE*

También existen híbridos intergenéricos entre mandarinas y naranjos dulces, los Tangors (del inglés TANGerina y ORange) donde destaca la *Ortanique*, entre mandarinas y pomelos, los Tangelos (TANGerina y pomELO) e híbridos entre híbridos y especies, como la Mandarina Nova (Clementina y Tangelo).

FOTO 2.8: TANGOR ORTANIQUE

Su calendario de recolección es el siguiente:

FIGURA 2.13: PERIODO DE RECOLECCIÓN DE LOS HÍBRIDOS DE MANDARINO

2.1.2.4. Pomelos

Los pomelos, con una importancia limitada en España, son arboles vigorosos, de gran tamaño, con una resistencia similar al naranjo. Tiene flores grandes y producen un fruto muy grande. Tienen un sabor ligeramente amargo y refrescante.

Las variedades más comerciales son *Ruby, Rio red, Ray ruby* y *Stay Ruby:*

FOTO 2.9: POMELO STAR-RUBY

2.1.2.5. Limoneros

Los limoneros en general son arboles de tamaño medio o grande, vigorosos y muy productivos. Sus frutos tienen una elevada acidez, alto contenido en zumo y un número alto de semillas.

Las principales variedades de limones que se producen en España son el limón *Verna* y *Fino.*

FOTO 2.10: LIMÓN VERNA

2.1.2.6. Limeros

Las limas se producen en zonas tropicales ya que son sensibles a bajas temperaturas y requieren de zonas calurosas y húmedas. Son arboles muy vigorosos,

verticales y con muchas espinas. Sus frutos se caracterizan por volverse amarillas justo en el periodo de recolección en periodo de recolección.

Se dividen en limas ácidas y dulces, aunque solo las ácidas tienen interés, como es el caso de la variedad *Bearss.*

FOTO 2.11: LIMA BEARS

Los pomelos, limas y limones tienen un periodo de recolección más amplio que las naranjas y las mandarinas:

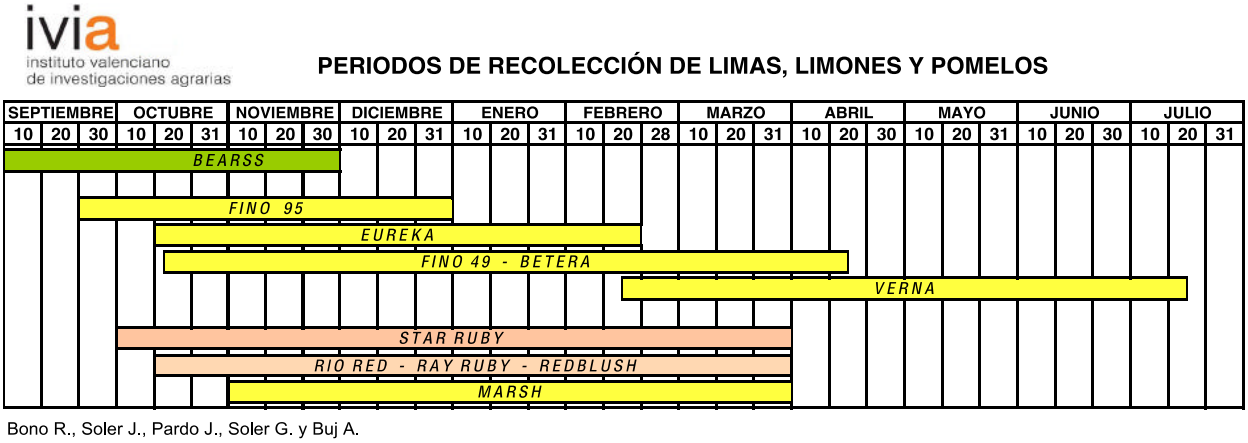


FIGURA 2.14: PERIODOS DE RECOLECCIÓN DE LIMAS, LIMONES Y POMELOS

2.1.2.7. Especies y variedades de interés ornamental

Además de estas variedades comerciales se utilizan mutaciones de los agrios con fines ornamentales y en jardinería, por motivos de belleza de follaje y colorido de los frutos.

Por un lado tenemos la utilización del naranjo amargo en calles y plazas, por ejemplo en Valencia, debido a su copa frondosa, hojas grandes, flores aromáticas y frutos grandes y coloridos.

También tenemos el Calamodín como un árbol cultivado como planta ornamental.

FOTO 2.12: ARBOLES DE CALAMONDÍN

Y otros como los *kumquats*, *citranges* y *limequats*:

FOTO 2.13: HOJAS Y FRUTOS DEL KUMQUAT NAGAMI

2.2. Análisis del sector

En este punto daremos una visión económica de la citricultura valenciana. En primer lugar analizaremos la situación actual, estudiando los organismos reguladores que influyen en el modelo de negocio y examinando la escasez tecnológica. Seguidamente, basándonos en las expectativas del sector, intentaremos esbozar el futuro agrícola en los próximos años, para así poder concluir cuales van a ser las líneas de trabajo con mayor posibilidad de éxito.

2.2.1. Situación actual

Como ya vimos al inicio de este capítulo, la citricultura tiene una gran importancia económica y de desarrollo rural en la Comunidad Valenciana. El cultivo de agrios representa el 31% de la superficie cultivable (213.404 ha) y en términos absolutos el valor de la producción asciende a 1.121,5 millones de euros (el 50% del todas las producciones agrícolas y ganaderas en la comunidad).

Sin embargo en los últimos años la citricultura valenciana está sufriendo una fuerte crisis que está provocando el abandono de las parcelas. Entre los problemas más importantes encontramos la caída de precios en origen, la apertura de fronteras con países suramericanos y africanos, la escasa diversificación varietal y la eliminación progresiva de ayudas.

Para entender esta decadencia repasaremos los organismos públicos y privados que intervienen en el proceso productivo y analizaremos los problemas para definir soluciones tecnológicas.

2.2.1.1. Organismos

En la cadena alimentaria intervienen un numeroso entramado de instituciones, empresas, intereses políticos y personas. Pero sin duda los organismos más influyentes son las entidades gubernamentales que regulan, mediante ayudas, el sector agrícola.

2.2.1.1.1. Autonómicos

El ente público más cercano lo encontramos a nivel autonómico en la Conselleria de Agricultura, Pesca, Alimentación y Agua (CAPAA). La Conselleria trata de aumentar la calidad de vida de los agricultores, mejorando la seguridad, promocionando productos, emitiendo ayudas, investigando y como instrumento mediador entre empresas agroalimentarias.

Entre sus organismos dependientes destacan el Consejo Regulador de Cítricos Valencianos, encargado de certificar y promocionar los cítricos que se cultivan en la CV, y el Comité de Agricultura Ecológica de la CV (CAECV), la autoridad de control de producción ecológica de la CV. Adscrito también a la Conselleria tenemos el Instituto Valenciano de Investigaciones Agrarias (IVIA) cuyo objetivo es impulsar la investigación científica y el desarrollo tecnológico de sector agroalimentario.

A nivel comarcal encontramos asociaciones como la Asociación Valenciana de Agricultores y la Asociación Agraria de Jóvenes Agricultores (AVA-ASAJA), que con más de 20.000 asociados, trata de reivindicar de una manera reformista y con un carácter empresarial los intereses de agricultores y ganaderos.

2.2.1.1.2. Estatales

En España la política agrícola es responsabilidad del Ministerio de Agricultura, Alimentación y Medio Ambiente (Magrama).

Dependientes del Magrama tenemos instituciones como el Fondo Español de Garantía Agraria (FEGA), cuya principal misión es velar para que las ayudas de la Política Agrícola Común (PAC) se apliquen estrictamente en sus dos principales fondos: el FEAGA (Fondo Europeo Agrícola de Garantía Agrícola), que en sus líneas incluye los pagos directos de la PAC, y el FEADER (Fondo Europeo Agrícola de Desarrollo Rural). También forman parte del Ministerio la Agencia Estatal de Meteorología (AEMET) y las Confederaciones Hidrográficas de cada cuenca (como la del Júcar).

2.2.1.1.3. Europeos

A pesar de la jerarquización estatal y autonómica, las grandes decisiones se toman a nivel europeo, debido a la importante dotación de fondos.

Uno de los primeros objetivos que se marcó la Unión Europea fue la de consensuar políticas agrícolas que consiguieran unos bajos precios en los alimentos, ya que consideran la alimentación como un bien común. Este y otros motivos, como el desarrollo rural, han propiciado que la Unión Europea dedique gran parte del presupuesto comunitario (cerca del 50%) en políticas y ayudas agrícolas.

Pero en la actual Europa de los 27 las decisiones no se toman en una única cámara:

- En primer lugar tenemos el Parlamento Europeo, formado por diputados Europeos elegidos cada 5 años por los votantes de cada país. Los diputados se encargan de debatir y aprobar, junto al Consejo, las leyes de la UE. Controlan las instituciones (especialmente la Comisión Europea), y debaten y aprueban el presupuesto de la UE.
- En segundo lugar tenemos al Consejo Europeo, que reúne a los líderes nacionales y donde se establecen las prioridades generales. Marca la política general, pero no tiene el poder de aprobar las leyes. Se reúnen cada 6 meses y su presidencia es rotatoria. No debe confundirse ni con el Consejo de la Unión Europea, donde se reúnen los ministros de cada país, ni con el Consejo de Europa, que no es un organismo de la UE.
- Y por último tenemos a la Comisión Europea cuyos miembros son nombrados por cada gobierno nacional y defiende los intereses conjuntos de la UE. Es una de las principales instituciones, ya que elabora las propuestas de nueva legislación, pone en práctica las políticas definidas y gestiona los fondos europeos. El Consejo Europeo designa a su presidente y a sus comisarios, entre los que se encuentra el Comisario de Agricultura y Desarrollo Rural encargado de la Política Agraria Común (PAC).

Las directrices políticas a nivel de la agricultura vienen determinadas principalmente por la Comisión Europea de Agricultura, que sería como el ministerio de agricultura europeo. Esta comisión es la que gestiona los fondos europeos para el desarrollo rural y garantía agraria, y los gobiernos y las comunidades autónomas son los encargados de vigilar el cumplimiento de esas políticas.

Así pues el funcionamiento del sector viene marcado por las decisiones políticas que se toman, principalmente, en dos frentes: la Política Agraria Común y el control fronterizo.

La PAC es una de las políticas más importantes de la Unión Europea. En sus orígenes pretendió incrementar la productividad, asegurar un nivel de vida equitativo en la población agrícola y garantizar precios de abastecimientos razonables. En la reforma de 1992 cambió de dirección buscando disminuir los precios ofreciendo ayudas compensatorias, política que se vio reforzada en la actual PAC (reformas del 2000, 2003

y 2007), introduciendo el pago de ayudas directas de apoyo a los precios y una fuerte política rural.

La actual PAC (2007-2013) tiene un fondo de más de 50.000 millones de euros anuales (40.000 millones en recursos financieros y otros 14.000 en desarrollo rural), lo que supone el 50% del presupuesto comunitario. Las principales ayudas que reciben los agricultores en la Comunidad Valenciana son, como ayuda directa, la solicitud única o pago único (unos 530 € por ha para naranjos y mandarinos), y otras subvenciones distribuidas en formación, jubilación anticipada, instalación de jóvenes agricultores y modernización de explotaciones (todas estas gestionadas por la agencia Valenciana de Fomento y Garantía Agraria), más los programas de desarrollo rural, Ruralter-Leader y Ruralter-Paisajes.

Pero para la Unión Europea el marco ha cambiado: el mercado mundial ha sufrido un fuerte crecimiento con la aparición de nuevas potencias económicas (como Brasil o China) y tras años de apoyo económico agrícola se encuentran zonas excesivamente intensas (que provocan sobreproducción, enfermedades, contaminación y otros problemas). Además la gestión se ha hecho un proceso centralizado y complejo, y los nuevos países del este no reciben el mismo porcentaje de ayudas.

De esta manera se prevé que para PAC del periodo 2014-2020 defina una nueva estrategia basada en la sostenibilidad (agricultura ecológica), reducción del volumen de ayudas respecto al PIB y una distribución más generosa con los nuevos países del este. Entre las medidas que pretende introducir la comisión europea es la de vincular el 30% de las pagos directos al cumplimiento de nuevas exigencias medioambientales.

Otra importante decisión política es la referente los pasos fronterizos. Hasta hace poco los países no pertenecientes a la UE tenían que pagar aranceles y sufrir fuertes controles fitosanitarios para evitar devaluar los productos europeos y controlar las plagas. Pero la política europea ha dado un giro aperturista y, auspiciada por países poderosos como Francia, esta abriendo fronteras con países no comunitarios, como es el caso de Marruecos. Estas aperturas repercuten negativamente en el sector agrícola español, ya que sus precios son inalcanzables y no están sometidos a controles.

2.2.1.2. Modelo de negocio

Pese a que existen comercios derivados de los cítricos, como la cosmética (la familia de las *Rutaceas* son muy importantes en el sector cosmético por ejemplo para producir perfumes) y la producción de bioetanol a partir de cáscaras de naranja, la estructura de producción de la citricultura valenciana es minifundista, monocultivo, a tiempo parcial y orientada a la producción de alimentos. Esto, unido al envejecimiento de población agrícola, ha provocado que el modelo de negocio se mantenga en torno al cooperativismo o al modelo de compra-venta entre agricultores y comercios.

2.2.1.2.1. Cooperativismo

El cooperativismo es la doctrina social que defiende la cooperación de sus integrantes en el rango económico y social como medio para lograr que sus participantes obtengan un mayor beneficio. Las cooperativas están formadas por personas unidas voluntariamente en una organización democrática cuya gestión debe ser acordada por sus socios. Los principios cooperativos son promovidos por la Alianza Cooperativa Internacional (ACI).

Las cooperativas agrarias son una de las más numerosas, aunque también existen cooperativas farmacéuticas, de transportes, de viviendas, de comercio…

FOTO 2.14: COOPERATIVA AGRÍCOLA SAN JOSÉ (ALCÁCER)

En la Comunidad Valenciana las cooperativas cuentan con una larga tradición, teniendo mayor presencia en pequeñas poblaciones donde dan trabajo a sus habitantes. Entre sus funciones destacan:

- Mantenimiento del medio rural: Son las grandes empresas situadas en las zonas rurales. Al estar compuestas y dirigidas por agricultores y ganaderos garantizan su propia existencia.
- Creación de empleo.
- Profesionalización de sus socios: Forman e informan a sus socios, introduciendo nuevas tecnologías y nuevos conocimientos, implantando además conceptos empresariales como la producción en cadena.
- Valor añadido en las operaciones comerciales: Concentran la oferta de productos agrarios y la demanda de suministros, consiguiendo productos de más calidad, con un mayor precio de salida y con una reducción de costes en suministros agrícolas.

El cooperativismo es el principal modelo de negocio utilizado en la citricultura valenciana. El agricultor socio de una cooperativa elige entre ceder el laboreo de la tierra a la propia cooperativa o bien gestionar sus propias tierras (siguiendo una política marcada por la cooperativa con controles sanitarios, técnicas de labrado y orientación varietal). Más tarde, en época de recolección, la cooperativa se encarga de recoger el fruto, procesarlo (limpieza, clasificación y empaquetado) y venderlo.

Además suele ser habitual que diferentes cooperativas locales se agrupen en asociaciones más grandes (cooperativas de cooperativas). Este es el caso del grupo Anecoop, una empresa cooperativa de segundo grado formada en 1975 y que en la actualidad es la primera empresa hortofrutícola del Mediterráneo.

2.2.1.2.2. Compra venta

El otro modelo importante de negocio es el llamado modelo de compra venta. Este modelo es el más sencillo y consiste en la gestión autónoma de las parcelas agrarias por parte del propietario y la posterior venta de la fruta, incluyendo los costes de recolección, a un comercio.

En la Comunidad Valenciana se da la peculiaridad de que este contrato de compra venta se realiza sin fijar un precio fijo definitivo, estimando los kilos de

producción y formalizando el acuerdo con un apretón de manos (*compra a ull*). De esta manera al inicio de cada campaña es el agricultor quien tiene que encontrar un comercio que cumpla con sus expectativas.

2.2.1.2.3. Comercio electrónico

Sin embargo, pese a la tradicionalidad de este sector, en los últimos años está teniendo un auge la venta de los cítricos directamente del agricultor al consumidor a través de portales web de venta de naranjas.

El reciente crecimiento de las Tecnologías de la Información y la Comunicación (TIC) ha permitido que los agricultores, sin un alto conocimiento de nuevas tecnologías, hayan encontrado en el comercio electrónico una forma competitiva de vender su producto. La actual crisis de precios en origen, causada entre otras cosas por el alto número de intermediarios en la cadena de venta, ha provocado que los agricultores hayan decidido empezar a vender sus naranjas directamente por Internet.

Las ventajas de este modelo son:

- Incremento de los beneficios para el productor, ya que se eliminan los intermediarios.
- Reducción del precio de venta.
- Aumento de la calidad del producto: La fruta llega directamente de la huerta al hogar.
- Compra on-line: El cliente compra su producto a través de una página web.
- Envío a domicilio.
- Contacto directo entre productor y consumidor.
- Personalización del pedido.
- Portal de novedades y noticias.
- Introducción de nuevos cultivos y nuevas técnicas, como la agricultura ecológica.
- Modelo con grandes oportunidades de negocio.

Con una sencilla búsqueda en la red podemos encontrar decenas de enlaces a portales de venta de naranjas: naranjasche.com, naranjaslola.com, lamejornaranja.com,

quieronaranjas.com, naranjasinternet.com, naranjascasanova.com, naranjasdirect.com, naranjasecologicas.com… Y la mayoría de ellas con sede social en la Comunidad Valenciana.

2.2.1.3. Tecnología actual

Para analizar el estado tecnológico actual nos vamos a centrar en los avances a nivel parcelario, puesto que la actual tesis no pretende estudiar ni las grandes obras hidrográficas (ingeniería de caminos e ingeniería del agua) ni los complejos sistemas de información gubernamentales utilizados en la gestión parcelaria (ayudas, censos e investigaciones).

El primer gran avance tecnológico que se ha producido durante las dos últimas décadas ha sido la introducción del riego por goteo.

Los árabes, en el siglo VII, fueron los primeros que en la península Ibérica desarrollaron los sistemas de irrigación. Aunque heredaron algunos conceptos de la época romana, fueron ellos los que introdujeron la cultura del agua creando acequias, almenaras, aljibes, azuds…

Las acequias servían para conducir el agua desde los pozos, canales, ríos o pantanos a los campos. Una vez canalizada el agua hasta la parcela agrícola, ésta inundaba la tierra.

FOTO 2.15: ACEQUIA DE RIEGO A MANTA POR DESBORDE (IZQUIERDA) Y POR PALETA (DERECHA)

Este método de regadío, conocido como riego a manta, ha sido el método de regadío predominante hasta el siglo XX. Buen ejemplo de ello ha sido el mantenimiento

del tribunal de las aguas de Valencia, una de las instituciones encargadas de mantener el orden.

Sin embargo durante los últimos años hemos asistido a un proceso de modernización de riego. Esta modernización se ha llevado a cabo en toda la jerarquía de riego:

- Los canales y embalses se han informatizado y optimizado con el fin de eliminar pérdidas y mejorar la gestión.
- Los motores de cada comunidad de regantes han cambiado a válvulas controladas electrónicamente con filtros de limpieza y depósitos para incorporar abonos.
- Las acequias se han sustituido por tuberías enterradas y selladas que distribuyen el agua hasta cada parcela.
- En el campo se han instalado casetas de riego con contadores, válvulas de activación de riego, succionadores de abonos y programadores de riego. Además por toda la parcela se han desplegado, con una estructura esquelética, tuberías con goteros autocompensados.

FOTO 2.16: INSTALACIÓN DE RIEGO POR GOTEO

Con esta modernización la gestión agrícola ha dado un importante avance, pasando de un modelo arcaico, centralizado, dependiente y deficiente a un modelo moderno, industrializado, descentralizado y autónomo.

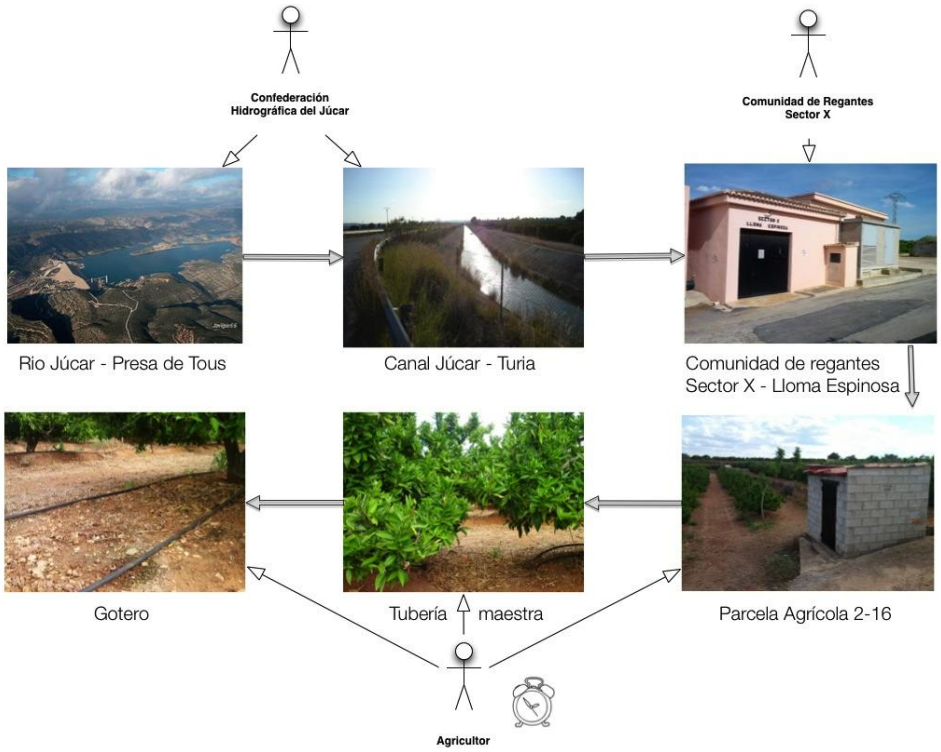


FOTO 2.17: ESQUEMA DE RIEGO DESCENTRALIZADO

Cabe decir que esta estructura descentralizada no es la única posible. En determinadas comunidades de regantes la programación de riego no es llevada a cabo directamente por el agricultor, sino que existen unas casetas comunitarias donde están instalados los filtros, contadores y válvulas maestras.

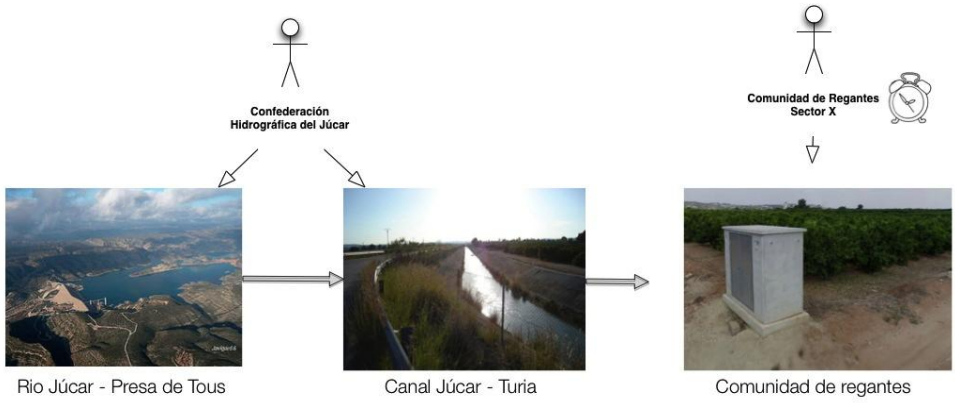


FOTO 2.18: ESQUEMA DE RIEGO CENTRALIZADO

Uno de los componentes más importantes de los sistemas de riego por goteo son los programadores de riego. Estos programadores son microcontroladores cerrados y privativos que permiten fijar la hora y programar la frecuencia y duración del riego.

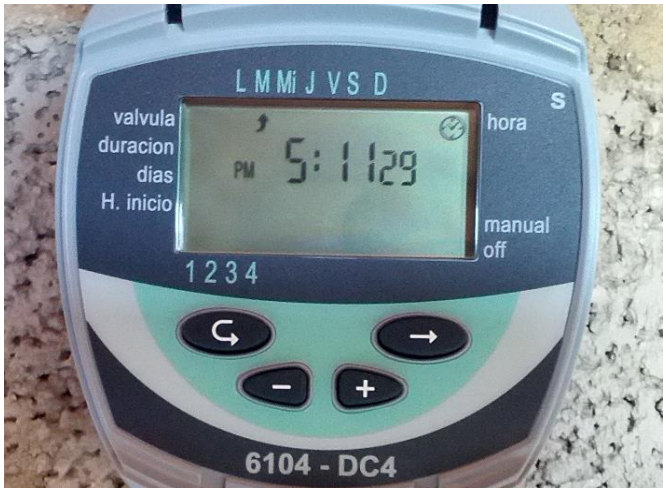


FOTO 2.19: PROGRAMADOR DE RIEGO GALCON

Son dispositivos sencillos (funcionan con pilas de 9V) que permiten controlar hasta 4 válvulas y establecer 4 alarmas por día. Disponen también de riego manual.

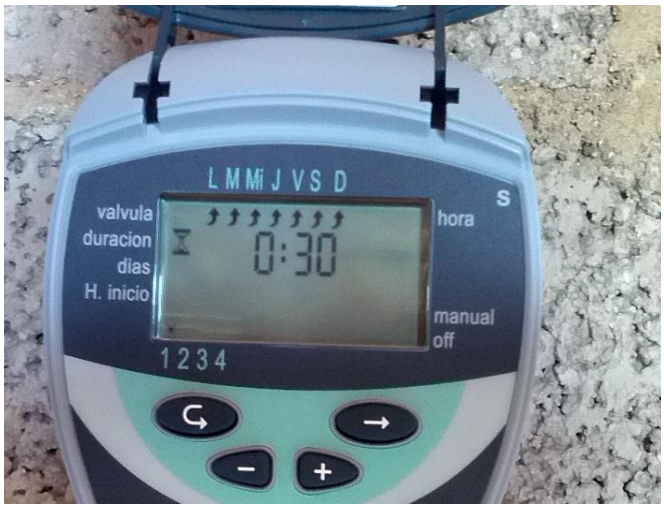


FOTO 2.20: PROGRAMACIÓN DE RIEGO DIARIA DE 30 MINUTOS

Requieren la utilización de válvulas activadas por pulsos y su precio ronda los 300 €,

A nivel de software agrario únicamente nos podemos referir a los sistemas puestos en marcha por las entidades gubernamentales para facilitar la gestión de ayudas y que de manera secundaria pueden complementar las tareas agrícolas. Hablamos de:

- SIGPAC: El Sistema de Información Geográfica de Parcelas Agrícolas (SIGPAC) es el software desarrollado por el ministerio de Agricultura, Alimentación y Medio Ambiente a través del FEGA, para identificar geográficamente las parcelas declaradas por los agricultores y ganaderos.

  Fue creado a finales del 2004 con los siguientes objetivos:

  - Facilitar a los agricultores la presentación de solicitudes con la producción de soportes gráficos, como los croquis.
  - Mejorar los controles administrativos digitalizando la información presentada por los agricultores.
  - Optimizar los controles sobre el terreno agilizando la localización de parcelas y permitiendo la realización de visitas rápidas tanto en controles clásicos como en controles por teledetección.

  El SIGPAC incluye ortofotos (fotos tomadas desde el aire y georrefenciadas) con una escala de, al menos 1:10.000, y capas vectoriales que delimitan parcelas y recintos.

  Disponemos de visores (para acceder a las ortofotos y a las diferentes capas almacenadas) en cada comunidad autónoma (Foto 2.1) y a nivel nacional en el FEGA.

- Tramites telemáticos: Las administraciones estatales y autonómicas están implantando durante los últimos años la administración electrónica. Apoyados en tecnologías como el DNI electrónico o los certificados digitales, las administraciones ponen a disposición del ciudadano portales web desde donde el interesado puede solicitar trámites de manera telemática. De esta forma la administración reduce costes, al eliminar los registros de entrada y los impresos, y agiliza los procesos de resolución; y para el ciudadano la burocracia se convierte en una tarea más fácil, ágil y barata.

A nivel autonómico, en la Conselleria de Agricultura, el usuario puede tramitar telemáticamente entre otros la gestión de alegaciones al SIGPAC (ALSI), la consulta de expedientes genéricos y el movimiento de ganado.

2.2.2. Futuro a corto y medio plazo

Como ya hemos esgrimido a lo largo de la presente sección, la agricultura mediterránea se enfrenta a una época de cambios:

Reforma de la PAC: Durante el año 2012 y principios 2013 se van a producir las negociaciones y acuerdos que van a definir la nueva Política Agraria Común. La actual Comisión Europea de Agricultura pretende eliminar ayudas a los países del sur para primar el desarrollo de los nuevos países del este, además de premiar los cultivos ecológicos. En esta negociación la estrategia española intenta conseguir reciprocidad de condiciones a los nuevos países, control aduanero e incrementar el importe de las ayudas.

Para el sector valenciano es imprescindible tener el apoyo de la Comisión Europea, ya que sus ayudas son vitales para el mantenimiento del sector. Por este motivo existe unanimidad en la posición frente a la nueva PAC: tanto asociaciones de agricultores, como organismos autonómicos, cooperativas y empresas del sector piden que las ayudas se ajusten a las condiciones de alta productividad del campo valenciano, que si lo comparamos con otros sectores agrícolas, como el olivo andaluz, tienen una productividad por hectárea mucho mayor, pese a recibir el mismo importe de ayudas. Además la Comunidad Valenciana quiere hacer valer la contribución medioambiental mediterránea en la mitigación del cambio climático, ya que incorpora unos valores como la fijación de CO_2, que en el caso de los cítricos se cifra en unas 900.000 toneladas anuales, el uso eficiente del agua o el cultivo de superficies con riesgo de salinización, que suponen una barrera protectora del medio por parte de los agricultores.

Diversificación varietal: La estructura productiva valenciana, unida a las características propias de los cítricos, ha provocado que el mercado se encuentre saturado en determinadas épocas del año. Por ello, desde los gobiernos autonómicos y cooperativas, se siguen poniendo en marcha planes de reconversión (mediante ayudas) citrícola con el objetivo de repartir las toneladas de fruto a lo largo del año.

Además en el ámbito de la investigación, instituciones como el IVIA, estudian la introducción de nuevas variedades de agrios, tanto de híbridos (como Moncalina y la Murta) como de mutaciones de Clementinas (variedades Nulessin, Nero, Clemenverd, IVIA-Man 19571), que permitan introducir frutos en periodos posteriores a Abril y mejorar la calidad (e. g. eliminando semillas)

Pero la diversificación varietal no solo afectará a los cítricos, sino que implicará la introducción de nuevos frutales. Tal es el caso de la introducción del kiwi en el campo valenciano. El kiwi o *Actinidia Deliciosa* es una planta trepadora propia de regiones templadas como Nueva Zelanda. Su estructura fisiológica requiere de la preparación de las parcelas con estructuras metálicas que permitan su desarrollo.

FOTO 2.21: PARCELA DE KIWIS

Además al ser plantas dioicas (poseen unidades reproductivas monosexuales que se manifiestan en diferentes individuos, es decir, tenemos plantas masculinas y femeninas) requieren de una disposición combinada y desproporcionada (solo un 20 o 30% de machos) para obtener una buena proporción entre polinización y calidad de frutos. El cultivo del kiwi supone un cambio sustancial en técnicas de cultivo.

FOTO 2.22: FLOR MASCULINA (IZQUIERDA, AMARILLA POR EL CENTRO) Y FLOR FEMENINA (DERECHA, BLANCO POR EL CENTRO) DEL KIWI

El motivo de la introducción del kiwi, y de otros frutos como el aguacate, es por que son cultivos muy sensibles al clima (mucho más que los cítricos) y la Comunidad Valenciana tiene un clima y suelo perfecto para su cultivo, siendo en regiones rivales, como Marruecos o el norte de Europa, incultivables por exceso de frio o de calor.

Agricultura ecológica: Una de las vertientes agrícolas que está teniendo un más auge durante los últimos años es la agricultura ecológica. La idea de este sistema de cultivo es la de utilizar de manera óptima los recursos naturales, sin utilizar productos químicos de síntesis u organismos genéticamente modificados (transgénicos).

En la agricultura ecológica la fertilización se lleva a cabo utilizando abono orgánico o *compost*, plantas que se descompongan en abono (el abono verde) o abonos minerales.

Las plagas y enfermedades dejan de tratarse con fitosanitarios para tratarse con la suelta de insectos útiles (depredadores y parásitos como el *Toxoptera aurantil* que ataca el pulgón), la utilización de productos naturales (como flores con propiedades bacterianas), atrayentes y repelentes (como el extracto de ajo para repeler la mosca blanca), feromonas que evitan la reproducción de plagas y la diversificación del cultivo (evitando así que aparezcan plagas y de paso enriqueciendo el medio).

La citricultura ecológica ya empieza a ser una realidad, tal y como demuestran la aparición de organizaciones como el CAECV (Comité de Agricultura Ecológica de la Comunidad Valenciana) y el incremento de comercios, por ejemplo, en grandes centros comerciales donde ya se pueden encontrar secciones específicas de agricultura ecológica. Además el precio de venta en sustancialmente superior debido a una producción más ajusta, la conciencia europea de respeto al medio ambiente y el enfoque a un público más selecto con un poder adquisitivo mayor.

Tecnificación del sector: La agricultura española, y en especial la citricultura valenciana, debe aceptar el reto de competir directamente con los países europeos líderes en tecnología agraria. Hablamos de países como Holanda en los que encontramos conceptos como los invernaderos inteligentes (que producen más energía de la que utilizan), la horticultura (cultivo de flores, verduras, frutas, champiñones, arboles, bulbos…), la agroindustria (empresas de alimentación y bebidas que trabajan directamente la tierra) y fuertes programas de investigación (como el *Food Valley*, un complejo donde trabajan cerca de 15.000 científicos mejorando el rendimiento, sostenibilidad y seguridad de los patrones).

Por tanto esta tecnificación tendrá como pilares la utilización de energías renovables (principalmente la energía solar), el cambio a la agricultura ecológica (productos que respeten con el medio ambiente), una cultura de vida más verde (cuidado de la naturaleza y de los animales) y la implantación de tecnologías de la información.

Comercio electrónico: Los comercios tradicionales deben empezar a explorar nuevas forma de negocio, en especial el comercio electrónico. Los agricultores que por su propio pie han decidido vender electrónicamente sus productos han demostrado que existe un mercado emergente con un margen de beneficio muy grande.

El comercio electrónico es un mercado que los productores deberán de empezar a asumir, siendo el caso de las cooperativas de vital importancia. Las cooperativas son un grupo de empresas con una gran cantidad de recursos y con un gran volumen de fruta, resultándoles una línea de negocio fácil de abrir, con rápidos beneficios, gran difusión, contacto directo con el cliente y una forma de crear una imagen de marca.

2.3. Motivación

Como hemos podido ver en este capítulo, la citricultura es una industria con un importante poder económico, cultural y ambiental en la Comunidad Valenciana. Por ello la agricultura no puede estar ligada de por vida al cultivo tradicional de agrios. En la actual situación económica los agricultores valencianos deben enfrentarse a un proceso de cambio, principalmente en dos frentes: tecnológico (con la incorporación de nuevas técnicas de cultivo y de tecnologías de la información) y varietal.

Además para acentuar este proceso de crisis, el contexto político europeo realizará reformas que implicarán una reducción de las ayudas tradicionales.

Por tanto esta tesis buscará beneficiarse de las oportunidades de negocio actuales (nuevas líneas de ayuda, inversiones públicas y nuevos mercados) para ofrecer soluciones tecnológicas que cubran las carencias del sector agrícola valenciano y español (hablamos de la escasez de jóvenes agricultores, el poco aprovechamiento de energías renovables y la falta de informatización).

3. CONTEXTO TECNOLÓGICO

En esta parte del proyecto estudiaremos los principales ingredientes tecnológicos utilizados para confeccionar un proyecto que se adapte a las necesidades del sector estudiado en la sección 2.

En primer lugar analizaremos los sistemas de información ubicuos como nuevo paradigma informático y a continuación presentaremos una tecnología abierta y pionera que dotará de los componentes necesarios para implementar la tesis.

Para realizar la primera sección nos hemos basado en la documentación de las asignaturas Sistemas de Información Ubicuos [26, 27, 28, 29, 30] y Sistemas de Inteligencia Ambiental [31].

3.1. Sistemas de información ubicuos

3.1.1. Introducción

La computación ubicua (CU) es el modelo de interacción hombre-máquina que pretende integrar la tecnología en la vida real, de manera que esta sea omnipresente. También se conoce con los términos anglosajones de *pervasive computing*, *context-aware computing* y *calm compunting*.

La computación pervasiva persigue evadir al usuario de configuraciones, uniendo software, hardware y redes para centrarse en la utilidad de los sistemas. Su objetivo es evolucionar la tecnología de manera que invada todos los ámbitos de la vida, haciéndola imperceptible al ser humano y eliminando la sobrecarga de información.

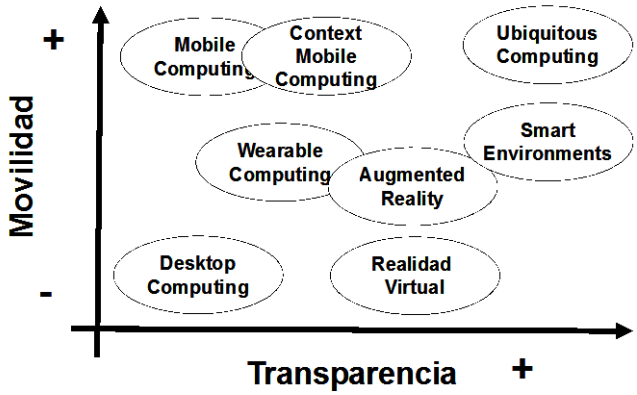


FIGURA 3.1: RELACIÓN ENTRE MOVILIDAD E INVISIBILIDAD DE LOS ACTUALES MODELOS DE COMPUTACIÓN

3.1.2. Origen

Su origen se remonta a finales de 1987 cuando en el centro de investigación PARC de Xerox coincidieron dos proyectos: los paneles interactivos de Bob Sprague y Richar Bruce, y el grupo de investigación de Lucy Suchman y Mark Weiser que buscaban un modelo computacional que se centrara en el uso. De la unión de estos tecnólogos surgió el primer proyecto de CU en 1988, un sistema compuesto por:

- El LiveBoard: Unos paneles interactivos.
- El ParcPad: Un computador del tamaño de un libro, con una interfaz dirigida con un lápiz y con diversas tecnologías que facilitaban la identificación, como el predecesor del actual sistema NFC (*Near Field Communication*).
- El ParcTab: Similar al Pad pero de tamaño más reducido (podía ser usado con una mano) y táctil. Llevaba toda una suite de herramientas para el trabajo en tiempo real (e-mail, calendario…). Además tenía una herramienta que permitía dibujar sobre el LiveBoard.
- El último dispositivo era una tarjeta de identificación que permitía identificar y posicionar al personal dentro de la empresa.

Todos estos dispositivos se diseñaron e implementaron teniendo en cuenta su uso colaborativo (con su infraestructura de comunicaciones) de tal manera que en 1994 estaba completamente funcionando en el PARC.

3.1.3. Características

El término de la computación ubicua es totalmente opuesto al de Realidad Virtual: en la RV nosotros nos sumergimos en un mundo virtual, mientras que en la CU integramos los computadores en nuestro mundo (*embodied virtuality*).

Para hacer invisible la tecnología el modelo paradigmático de una persona un ordenador ha de evolucionar al modelo de una persona cientos de ordenadores.

Un factor de vital importancia en la CU es la localización: los dispositivos deben tener la capacidad de identificarse de manera unívoca, para variar su funcionamiento dependiendo de su localización. Además debemos tener la tecnología necesaria para poder obtener dispositivos de todas las escalas, desde tarjetas identificadoras hasta pantallas incrustadas en la pared.

Los sistemas ubicuos deben implementar técnicas de criptografía que garanticen la seguridad de los programas y sus datos.

El hardware que forma los dispositivos debe ser barato y de bajo consumo.

El software pasará de sistemas monolíticos a sistemas de *microkernels* con una parte básica a la que se incorporan dinámicamente módulos funcionales.

Las redes de los SIU (Sistemas de Información Ubicuos) son redes inalámbricas, rápidas, baratas y capaces de albergar un cientos de dispositivos.

Los principales componentes de un sistema de información ubicuo son: el modelado del contexto y la arquitectura. A continuación detallaremos las principales características de estos conceptos.

3.1.4. Contexto

La computación ubicua debe ser capaz de variar su funcionamiento en función del contexto. El contexto son los elementos del entorno del usuario que el que el sistema sabe. Es de vital importancia que el modelado del contexto se realice de una manera formal evitando hacer uso de técnicas primitivas, esquemáticas y mal comunicadas.

Existes diferentes maneras de modelar el contexto:

- Clave-valor: La más simple utilizando pares calve valor.
- Esquema de marcado: Estructuras jerárquicas con etiquetas de marcado.
- Modelos gráficos: Como por ejemplo UML.
- Modelos orientados a objetos.
- Modelos basados en lógica.
- Modelos basados en ontologías.

Una de las tendencias actuales es modelar el contexto con ontologías, ya que son simples, flexibles, extensibles, generales y expresivas. Una ontología es una comprensión compartida sobre ciertos dominios, siendo el resultado un conjunto de entidades, relaciones, funciones, axiomas e instancias.

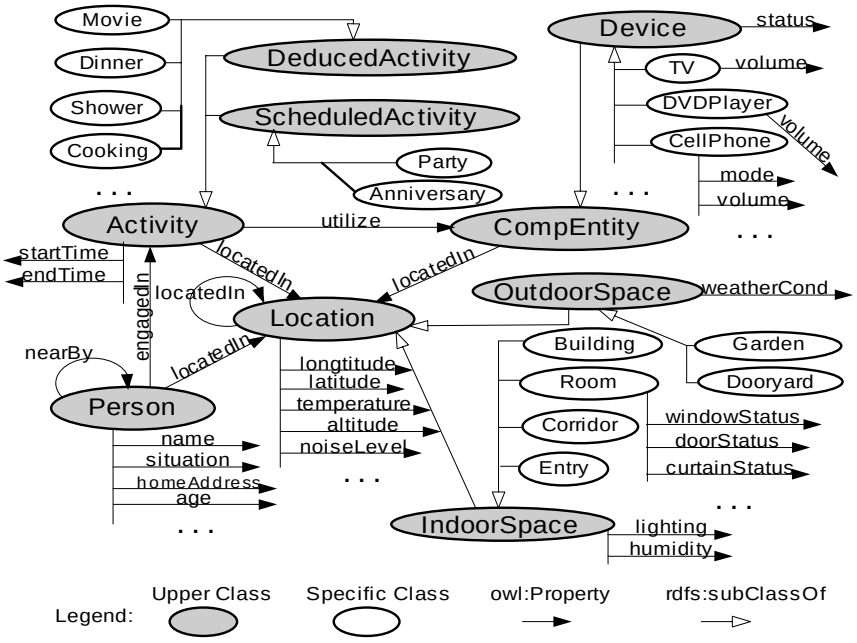


FIGURA 3.2: ONTOLOGÍA PARCIAL DEL DOMINIO DE UN HOGAR

En esta formalización se hace uso de predicados de primer orden para representar el contexto y facilitar la compartición del conocimiento (interoperabilidad). Ontologías como CONON sugieren dividir en dos niveles el conocimiento: uno superior, en el que están todos los términos compartidos, y diferentes subdominios compartidos específicos para cada problema.

Una vez modelado el contexto, podemos extraer el conocimiento explícito mediante propiedades de la lógica de primer orden (transitivas, inversas, subclases…) o mediante reglas definidas por el usuario.

Situation	Reasoning Rules
Sleeping	(?u locatedIn Bedroom) $\wedge$ (Bedroom lightLevel LOW) $\wedge$ (Bedroom drapeStatus CLOSED) $\Rightarrow$ (?u situation SLEEPING)
Shower-ing	(?u locatedIn Bathroom) $\wedge$ (WaterHeater locatedIn Bathroom) $\wedge$ (Bathroom doorStatus CLOSED) $\wedge$ (WaterHeater status ON) $\Rightarrow$ (?u situation SHOWERING)
Cooking	(?u locatedIn Kitchen) $\wedge$ (ElectricOven locatedIn Kitchen) $\wedge$ (ElectricOven status ON) $\Rightarrow$ (?u situation COOKING)

FIGURA 3.3: REGLAS DEFINIDAS POR EL USUARIO PARA LA ONTOLOGÍA DE HOGAR

3.1.5. Arquitecturas

Los SIU vienen marcados por la tecnología utilizada en el software, el hardware y las redes. Por tanto es interesante estudiar las principales arquitecturas utilizadas.

La mayoría de sistemas, independientemente de la arquitectura utilizada, responden a una aproximación por capas de 5 niveles: sensores (físicos, lógicos y virtuales), controladores, preprocesado (encapsulamiento de la información de diferentes sensores), API (preprocesado y gestión) y la capa de aplicación.

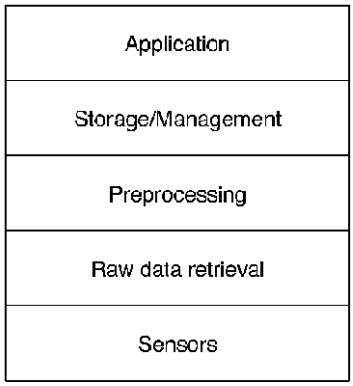


FIGURA 3.4: NIVELES CONCEPTUALES DE LOS SIU

En cuanto a arquitecturas tenemos:

- Acceso directo a los sensores: El cliente software accede al sensor directamente para obtener la información deseada, sin existir ninguna capa adicional que obtenga y procese la información. Son sistemas fuertemente acoplados, ya que incorporan los controladores de los sensores y penalizan la interoperabilidad.
- *Middleware*: Esta aproximación introduce una arquitectura por capas con la intención de esconder los detalles de bajo nivel de los sensores. Facilita la extensibilidad y la reusabilidad de los componentes.

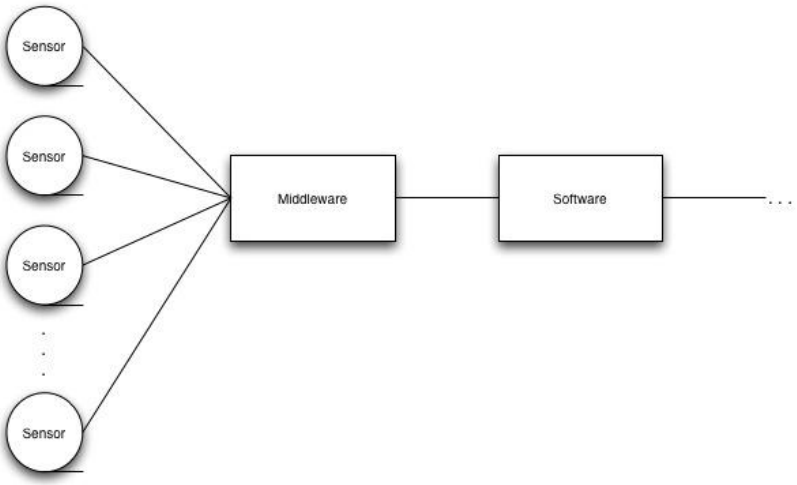


FIGURA 3.5: ARQUITECTURA MIDDLEWARE

- Servidor de contexto. Es una evolución de la arquitectura *middleware* que introduce un componente de gestión remota de acceso. El acceso a la información de los sensores se mueve a un servidor de contexto que sirve la información a cada cliente. Facilita el acceso concurrente e independiza el uso de la información del proceso de obtención. Requiere un mayor desarrollo de la red, ya que las comunicaciones toman mayor importancia.

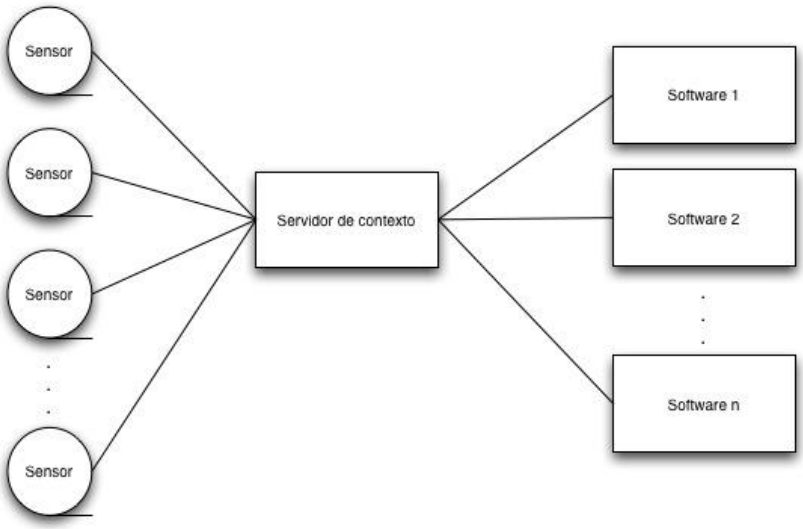


FIGURA 3.6: ARQUITECTURA CON SERVIDOR DE CONTEXTO

3.1.6. Sistemas de inteligencia ambiental

Dentro de los SIU podemos considerar a los Sistemas de Inteligencia Ambiental (SIA) como un subconjunto de ellos. Aunque algunas fuentes a menudo los definen como sinónimos de los sistemas ubicuos, para el autor de esta tesis los SIA se diferencian de los SIU en que están más centrados en adaptar su comportamiento en función de las características ambientales y tienen un marcado carácter de aprendizaje.

Los SIA crean entornos inteligentes que se adaptan a las necesidades, gustos e intereses de los humanos que viven en el entorno, ayudando además a realizar las tareas de la vida diaria.

La inteligencia ambiental esta basada en servicios como el aprendizaje automático, el reconocimiento de lenguaje natural, gestos y estados de ánimo. Por tanto focalizan más en tecnologías de interfaces naturales (HCI), inteligencia artificial e interacción con sensores físicos.

Dentro de los SIA encontramos el subgrupo de los sistemas domóticos, que son los encargados de proporcionar servicios relacionados con el hogar. Los principales servicios que ofrecen son:

- Automatización y control: Iluminación, electrodomésticos…
- Seguridad y vigilancia: Detección de intrusos, simulación de presencia, prevención de accidentes.
- Comunicaciones: Interacciones entre humanos y maquinas.
- Teleservicios y ocio: Telemedicina, juegos en red…

3.2. Arduino

3.2.1. Introducción

Arduino es una plataforma de computación física de código abierto, formada por una placa (básicamente un microcontrolador con entradas y salidas hardware) y un entorno de desarrollo basado en la plataforma Processing.

La computación física utiliza la electrónica para prototipar nuevos dispositivos que interaccionan con los humanos a través de sensores y actuadores, siendo éstos controlados por un software que corre dentro del microcontrolador.

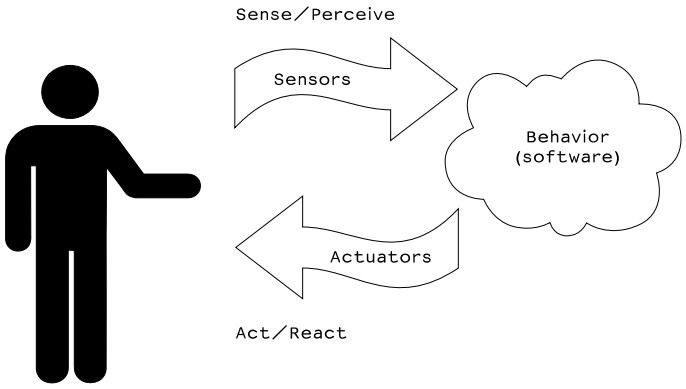


FIGURA 3.7: DISPOSITIVOS INTERACTIVOS

En el pasado la electrónica requería trabajar con ingenieros especializados, construir pequeños componentes y tener amplios conocimientos. Pero los microcontroladores han evolucionado a dispositivos más baratos y fáciles de usar, siendo mucho más accesibles a aficionados, artistas e ingenieros de otras ramas.

Arduino proporciona la infraestructura para que todo el mundo sea capaz construir sus propios objetos interactivos: las placas, que se pueden adquirir o construir a partir de los diagramas, y un entorno de desarrollo (el IDE más librerías) liberado como código abierto.

3.2.2. Orígenes

La plataforma Arduino surgió en el Instituto de Diseño Interactivo Ivrea en 2005, basándose en otros proyectos en los que estaban trabajando, como el lenguaje de programación Processing y la plataforma Wiring.

La idea era crear una herramienta para los estudiantes más barata y moderna que las soluciones existentes, como BASIC Stamps.

En el equipo de desarrollo de Arduino destacaban Massimo Banzi y el español David Cuartielles. Ellos fueron quienes, basándose en los experimentos de Banzi, desarrollaron la primera placa. Más tarde se incorporaron David Mellis para dedicarse en la parte del software, Tom Igoe como consejero y Gianluca Martilo como productor de las placas.

El proyecto empezó uniendo y reimplementando las tecnologías de Wiring y Processing, utilizando un microprocesador más barato y añadiendo compatibilidad con programas antiguos.

Tras el primer prototipo de Banzi y Cuartielles, el equipo paso ha producir un lote de 200 unidades con un diseño más profesional, vendiéndolas a diferentes universidades a precio de coste. Pero no fue hasta la llegada de la versión USB cuando diferentes distribuidores, como SparkFun y Adafruit, empezaron a comercializar la placa. Hoy en día se llevan vendidas más de 50.000 placas.

La placa más famosa es el modelo Arduino UNO, aunque como luego veremos disponemos de varias placas más.

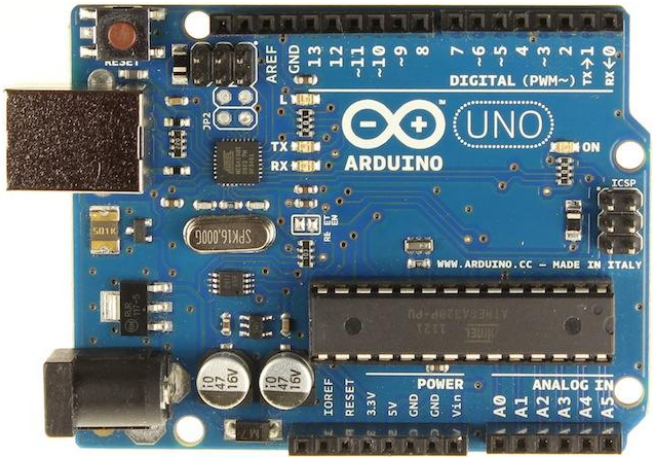


FOTO 3.1: ARDUINO UNO REVISIÓN 3

3.2.3. Características

El funcionamiento de Arduino consiste básicamente en obtener la información del entorno a través de sus entradas y afectar a aquello que lo rodea utilizando los pines de salida.

3.2.3.1. Código abierto

Arduino es el emblema del hardware *open-source*.

Las referencias de diseño de las placas, es decir, los ficheros Eagle CAD donde se detallan los diagramas de placa, están publicados bajo la licencia *Creative Commons Atrribution Share-Alike*. Esta licencia permite utilizar el material tanto para fines personales como comerciales, siempre y cuando se referencie a Arduino (*Attribution*) y el producto final se libere bajo la misma licencia (*Share-Alike*).

El entorno de desarrollo está implementado en Java y liberado bajo la licencia GPL (*General Public Licence*), una licencia estricta en términos de *copyleft* que obliga a, en caso de hacer una distribución pública del software, éste y sus versiones modificadas sean publicadas bajo GPL.

Y por último las librerías escritas C y C++ que permiten acceder al microcontrolador (la API de nivel más bajo) está publicado bajo la licencia LGPL (*Lesser General Public Licence*), una licencia más permisiva que permite utilizar el código en programas privativos. Esto implica que el firmware generado por el IDE puede ser incluido en software privativo.

3.2.3.2. Multiplataforma

Uno de los motivos por los cuales en los primeros días de vida del proyecto eligieron basarse los experimentos de Banzi para empezar a desarrollar la primera placa fue la interoperabilidad. Arduino utiliza librerías de código abierto, como RXTX, escritas en C y C++ y compiladas para diferentes sistemas operativos. Además el entorno de desarrollo está escrito en Java lo que permite tener un software multiplataforma.

3.2.3.3. Coste

El proyecto Arduino surgió con el objetivo de producir placas de muy bajo coste. Actualmente podemos comprar placas en multitud de distribuidores (http://arduino.cc/en/Main/Buy) y el precio para la placa básica Arduino UNO es de 22 €.

3.2.3.4. Consumo

Como luego veremos en el apartado arquitectura, la placa Arduino puede ser alimentada a través de una conexión USB o de una fuente de alimentación externa con voltajes entre 6 y 20 V.

Por cada pin la placa es capaz de sacar una corriente de unos 40 mA, lo que la convierte en una placa de bajo voltaje y bajo consumo.

3.2.3.5. Extensibilidad

Gran parte del éxito de Arduino se ha debido a la extensibilidad que le han otorgado los diferentes *shields* que han ido apareciendo. Un *shield* no es más que una placa que se acopla a Arduino u otros *shields* y que le otorga una cierta funcionalidad. En el apartado 3.2.3.8 realizaremos un estudio de los principales *shields* en el mercado.

3.2.3.6. Comunidad

Como todo proyecto de software libre que se precie, Arduino cuenta con una gran comunidad de desarrolladores que publican su código abiertamente.

El organigrama de la comunidad Arduino responde a una estructura oligárquica, en la que las decisiones las toman un grupo de expertos, formados actualmente por el equipo original del proyecto.

En este escalafón más alto se deciden que aportaciones y nuevas funcionalidades se añaden al proyecto principal. Así por ejemplo, el equipo Arduino decide que nuevas características incluirá el IDE o que microcontrolador dirigirá la placa. Su ámbito de actuación abarca todo el proyecto Arduino: software, hardware, documentación, marketing, ventas...

Sin embargo, pese a esta estructura centralizada, cualquier desarrollador puede realizar sus aportaciones de diferentes modos. En primer lugar los usuarios pueden contactar directamente con el equipo para realizar sugerencias o notificar errores. Por otro lado tenemos el Arduino Playground, una Wiki donde desarrolladores de todo el mundo publican su código, proponen nuevas características, aportan librerías, y documentan proyectos. Aunque el Playground se podría considerar el núcleo de la comunidad Arduino, la estructura colaborativa y descentralizada del software libre permite que diferentes portales web publiquen también sus herramientas para favorecer

el desarrollo. Hablamos de foros, Wikis y página de distribuidores SparkFun o la española Cooking Hacks (de la empresa Libelium).

3.2.3.7. Entorno de desarrollo

La plataforma Arduino está formada principalmente por dos piezas: las placas, que son la parte hardware, y el entorno de desarrollo, la parte software que incluye las herramientas de desarrollo y las librerías de acceso a la placa.

El entorno de desarrollo está basado en el proyecto de software libre Processing, un lenguaje de programación con un entorno de desarrollo basado en Java. Es un proyecto creado por artistas y diseñadores de fácil utilización, enfocado a la enseñanza y producción de proyectos multimedia e interactivos. Aunque su ámbito de aplicación es muy grande, destaca por su manejo sencillo de imágenes y gráficos, como podemos observar en su tablón de proyectos (http://processing.org/exhibition/).

En Arduino un programa o *sketch* es un fichero de texto con la extensión *.ino* en el que se definen dos funciones del lenguaje Processing: *setup()* y *loop()*.

```
void setup() {
  // Inicialización del pin digal
  pinMode(13, OUTPUT);
}
void loop() {
  digitalWrite(13, HIGH);   // Encendemos el LED
  delay(1000);              // Esperamos un segundo
  digitalWrite(13, LOW);    // Apagamos el LED
  delay(1000);              // Esperamos
}
```

La función *setup*() se ejecuta solo una vez, al conectar o resetear la placa, y la función *loop*() es un bucle infinito que se repite hasta que la placa se desconecta o se resetea. Además al principio del programa podemos definir directrices del compilador y variables globales.

El lenguaje de programación Processing está basado en Java, lo que le permite heredar todas sus funcionalidades, pero utiliza una sintaxis más simplificada y orientada a la programación gráfica. Cada *sketch* es realmente una subclase de la clase Java *PApplet*, convirtiendo cada clase dentro del *sketch* en clases Java.

En cuanto al software, encontramos el IDE Arduino (apartado 9.5).

3.2.3.8. Hardware

Aunque los orígenes de la placa se remontan a las investigaciones de Banzi y Cuartielles, Arduino ha tenido importantes influencias de proyectos de hardware como Wiring.

Wiring es también una plataforma de programación de microcontroladores de código abierto. Al igual que Arduino está orientado al público en general.

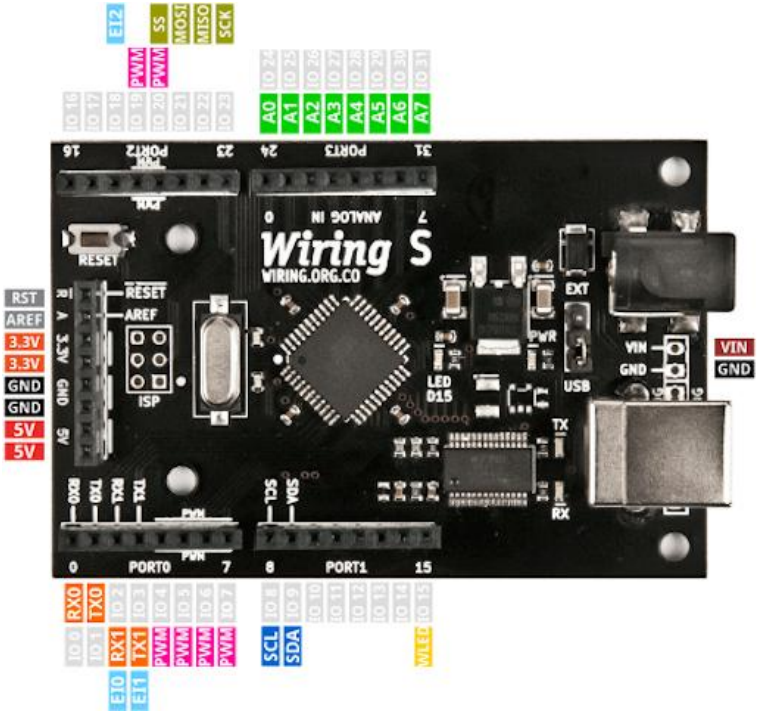


FOTO 3.2: PLACA WIRING

Su núcleo está formado por un microcontrolador de tipo AVR de la serie ATmega, aunque también soporta microchips PIC (más información en la sección 9.6). Incorpora 32 pines que se pueden configurar como entradas y salidas para conectar sensores y actuadores. Estos pines, como muestra la imagen, están agrupados por características.

El proyecto Arduino es una plataforma que incluye un conjunto de placas con una arquitectura similar a la placa Wiring. Todas ellas están basadas en los microcontroladores AVR de Atmel, utilizan *sketch* compilados con el compilador *avr-gcc* (librerías AVR *Libc*) e incorporan un *bootloader* (cargador de arranque) que permite la carga de software sin hardware adicional.

A continuación detallaremos las principales características de todos los modelos:

3.2.3.8.1. Arduino UNO

El buque insignia de la plataforma Arduino. Es la evolución de los modelos Duemilanove y Diecimilia (la versión original).

Dirigida por el micro ATmega328 cuenta con 32 KB de memoria flash (con 0,5 KB utilizados por el *bootlader*), 2 KB de memoria SRAM y 1 KB de memoria EEPROM (donde podemos guardar información no volátil con una librería específica).

La placa se conecta al PC a través del puerto USB. Esta comunicación se lleva a cabo por el chip ATmega16U2, un AVR (otro microcontrolador RISC) programado como conversor de USB a serie (remplazando el anterior *driver* FTDI USB a serial) que incluye además el controlador para los sistemas Windows.

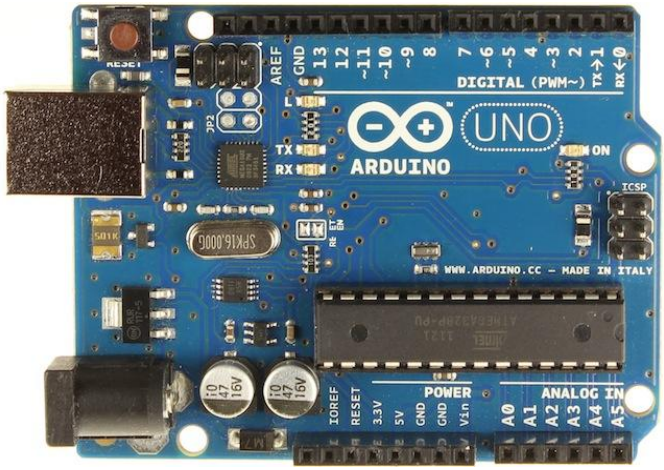


FOTO 3.3: ARDUINO UNO

Esta placa puede ser alimentada a través del USB, del conector *jack* o de los pines V_{in} y GND, admitiendo diferencias de potencial entre 6 y 20 V (se recomiendan voltajes entre 7 y 12 V), siendo la fuente de alimentación seleccionada automáticamente. Además incorpora un fusible de polímero reseteable que, junto a un conjunto de transistores y diodos, protege de sobretensiones y picos de corriente. Dispone de una salida de 3.3 V, con una corriente máxima de 50 mA, y de una salida de 5 V.

Cuenta con 14 pines digitales que pueden ser programados como entradas o salidas (a través de las funciones *pinMode(), digitalWrite()* y *digitalRead()*), de las cuales 4 son del tipo PWM si se utiliza la funcion *analogWrite()* (permiten modular el ancho de pulsos para enviar la información de manera analógica, e. g. voltajes de 2,7

V). Cada pin puede recibir y emitir hasta un máximo de 40 mA y tiene internamente una resistencia en *pull-up* (colector abierto) de entre 20 y 50 KOhms.

La placa UNO dispone de 6 entradas analógicas (de la A0 a la A6) que proporcionan 10 bits de resolución (1024 valores diferentes) para la lectura de datos mediante la función *analogRead().*

Otros pines relevantes son el 13, que incorpora un LED, el pin AREF, que permite variar la referencia de tierra (por defecto a 5V) de las entradas analógicas, IOREF, que informa a *shields* u otros dispositivos del voltaje proporcionado por la placa, pines para el protocolo I^2C que permiten interactuar con sensores y actuadores compatibles (pines A4, A5, y los pines SDA y SCL al lado del AREF), los pines 10, 11, 12 y 13 para el protocolo SPI (funcionalidad actualmente no soportada en el lenguaje Arduino) y el pin *reset*, que permite evitar el autoreseteo automático. En el borde derecho tenemos los pines ICSP (*In-Circuit Serial Programming*), conectores que permiten programar el micro sin pasar por el *bootloader* utilizando uno de los siguientes dispositivos, que transforman del protocolo serie RS232 a USB:

- Cables FTDI: No incorporan el controlador en un chip integrado por lo que hay que instalar software adicional y resetear la placa de manera manual.

FOTO 3.4: CABLE FTDI USB

- *Breadboard* FTDI de SparkFun: Incorpora el driver FTDI a USB, que facilita la comunicación entre dispositivos. Requiere instalar el driver para el SO destino.

FOTO 3.5: PLACA FTDI USB DE SPARKFUN

- Adaptador Arduino USB Serial: Utiliza el micro ATmega8U2 programado por como controlador facilitando proceso de subida (autoreseteo, *checksums*…) e incorpora el driver para sistemas Windows.

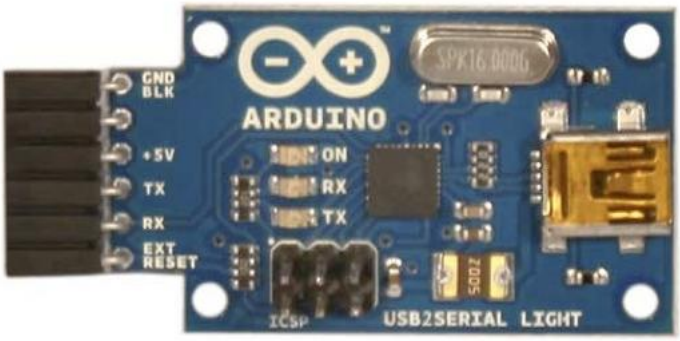


FOTO 3.6: ADAPTADOR ARDUINO DE SERIE A USB

Los pines 2 y 3, además del funcionamiento como E/S digital estándar, permiten definir interrupciones que actúan como disparadores (definidos con la función *attachInterrupt())* frente a eventos, como valores bajos de tensión, tensiones límites o cambios de valor.

Existe otra versión comercial, la llamada Arduino UNO SMD, que se diferencia del modelo estándar en que incorpora el microcontrolador directamente soldado a la placa. El motivo de llevar la placa estándar el micro en formato de pastilla y zócalo es el de permitir el intercambio de micros (por ejemplo con diferentes versiones del

software), sustituir componentes dañados o para probar otras versiones de la familia ATmega.

FOTO 3.7: ARDUINO UNO SMD

Su precio es de 22 €.

3.2.3.8.2. Arduino Mega 2560

Es la placa más grande y potente de Arduino. Incorpora un microcontrolador superior, el ATmega2560, soldado directamente a la placa.

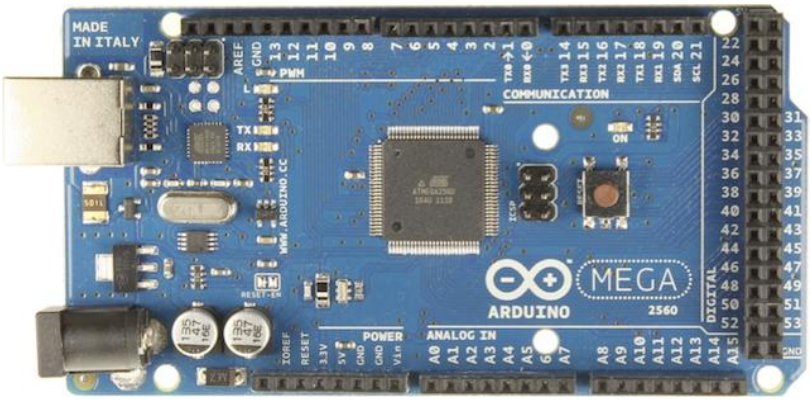


FOTO 3.8: ARDUINO MEGA 2560

La unidad de proceso ATmega2560 ejecuta instrucciones RISC a una velocidad de reloj de 16 MHz, y cuenta con una memoria flash de 256 KB (8 de los cuales son usados por el *bootloader*), una memoria RAM de 8 KB y 4 KB de memoria EEPROM.

Al igual que el modelo básico, se conecta por USB, admitiendo alimentación externa. Utiliza el microcontrolador ATmega16U2 como convertidor USB a serie e incorpora los pines extra SDA, SCL, AREF, IOREF, RESET_EN.

Sus diferencias respecto a la placa UNO son:

- La cantidad de pines. En esta placa tenemos 54 pines digitales, de los cuales 15 proveen una salida PWM y16 entradas analógicas.
- La incorporación de nuevos puertos de comunicación serie hardware (UART) que permiten a los pines 14, 15, 16 y 17 establecer comunicaciones de serie con otros dispositivos sin tener que utilizar librerías que emulen este protocolo (como la librería *SoftwareSerial*).
- Más interrupciones: Los pines 2, 3, 18, 19, 20 y 21 pueden definir *triggers*.

Su tamaño es más alargado, 10,1 cm, pero igual de ancho, lo que permite que sea compatible con los *shields* del modelo estándar. Tiene un precio de 41 €.

3.2.3.8.3. Arduino Mega ADK

Esta versión fue introducida en el año 2011 con motivo de la aparición del kit de desarrollo de accesorios para Android (Android ADK).

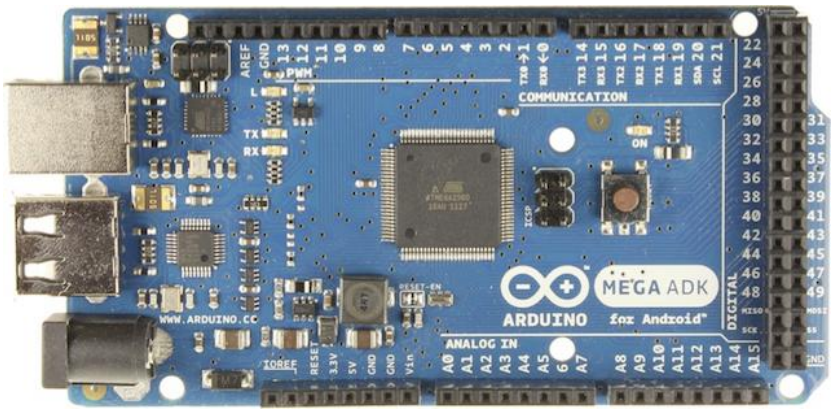


FOTO 3.9: ARDUINO MEGA ADK

Se trata de una placa Arduino Mega a la que han añadido el chip MAX3421e IC, que dota a la placa de una interfaz USB en modo *host* que permite conectar dispositivos como clientes USB (USB *devices*).

El resto son los mismos componentes que la placa Arduino Mega 2560.

Su precio es de 59 €.

3.2.3.8.4. Arduino Nano

Es un modelo pequeño pero completo, que incorpora el mismo controlador que la Arduino UNO, el ATmega328. Carece de adaptador de corriente *jack*, ya que se alimenta a través del cable mini-B USB, que sirve también de comunicación (llevada a cabo por el chip USB FTDI). Las características de memoria son las mismas que la placa UNO, exceptuando el tamaño del *bootloader* que se ve incrementado a 2 KB debido al uso del chip FTDI USB.

Podemos considerarla una placa Arduino UNO de reducido tamaño, 4,30 cm de largo por 1,85 cm de ancho.

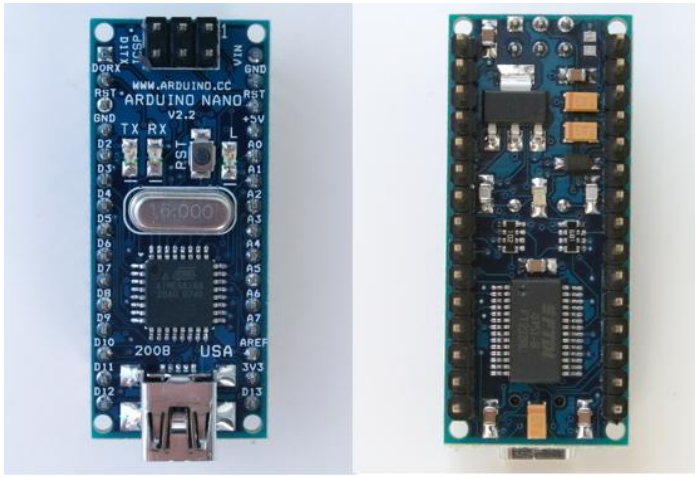


FOTO 3.10: ARDUINO NANO

Cuenta con 14 entradas digitales (4 de PWM) y 6 analógicas, además de pines I^2C, interrupciones y otros puertos.

Ha sido diseñada y producida por Gravitech, con un precio de venta al público de 33 €.

3.2.3.8.5. Arduino Mini

Es la versión más diminuta que incorpora el micro ATmega328 (3 cm de largo).

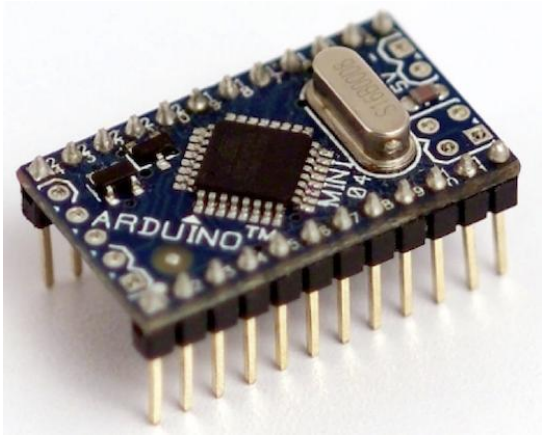


FOTO 3.11: ARDUINO MINI

Tiene los mismos pines que el modelo nano:

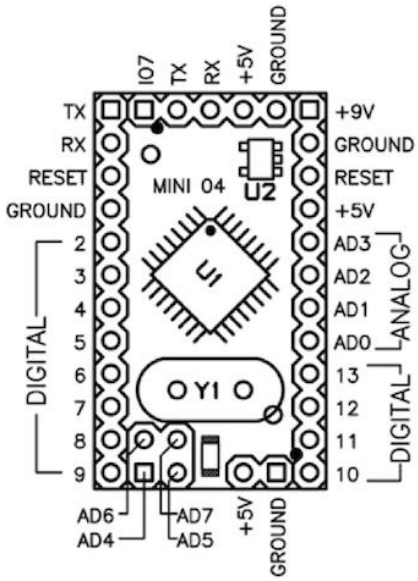


FOTO 3.12: CONEXIONES ARDUINO MINI

Pero no incorpora adaptador de serie a USB, por lo que debe ser programada con un adaptador.

Debido al tamaño no incorpora ni los fusibles, ni la circuitería necesaria para protegerla de picos de corriente y voltajes altos, por lo que el rango de entrada recomendado se reduce de 7 a 9 V. Su precio ronda los 26 €.

3.2.3.8.6. Arduino LilyPad

La placa LilyPad es un modelo diseñado para usar en prendas y textiles, con un estilo moderno, de color púrpura y lavable, perfecta para coser en la ropa. De su diseño y desarrollo se encarga SparkFun.

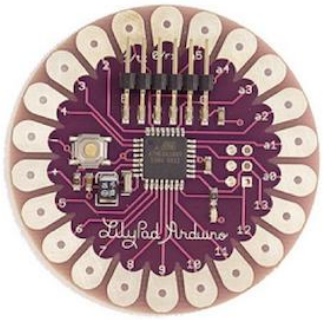


FOTO 3.13: LILYPAD ARDUINO

Utiliza el microcontrolador ATmega328V, la versión de bajo consumo del ATmega328. Este AVR funciona a una velocidad de 8 MHz, cuenta con una memoria Flash de 16 KB, 1 KB de RAM y 512 bytes de EEPROM. En cuanto a pines tenemos 16 de entrada y salida digitales (6 de ellos PWM) y 6 entradas analógicas.

Al ser un micro de bajo voltaje acepta una diferencia de potencial de entrada de 2,7 a 5,5 V, siendo cualquier voltaje por encima de 5,5 V dañino para la placa.

La placa lleva instalado el *bootloader* de Arduino pero, al carecer de puerto de conexión, requiere para la programación de la conexión de un accesorio, como los que hemos visto antes, que permita la comunicación.

Dispone de una amplia gama de *shields* personalizados que permiten conectar baterías, bocinas, tarjetas, etc. Tiene un precio de 17 €.

FOTO 3.14: ACCESORIOS LILYPAD

3.2.3.8.7. Arduino Pro

Es una placa pensada para ser instalada de manera permanente en los proyectos. Tiene un diseño minimalista, con un formato compatible con los *shields* estándar e incorpora conectores para baterías.

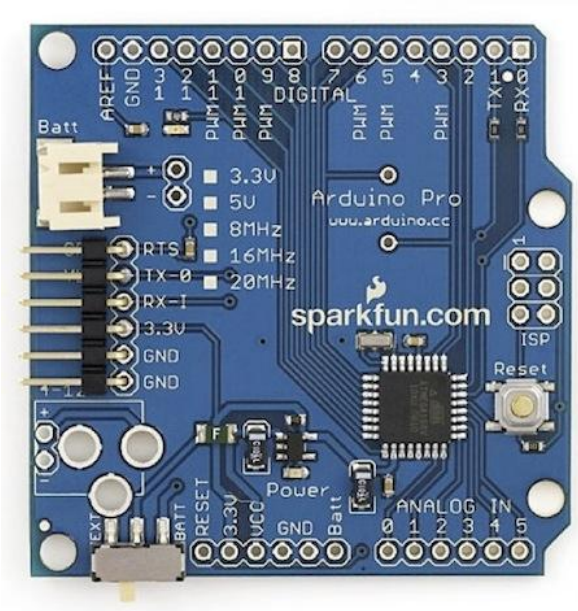


FOTO 3.15: ARDUINO PRO

Está optimizada para un entorno de explotación, eliminando componentes innecesarios, como el driver USB.

Consta del mismo AVR que la placa UNO, el chip ATmega328, soldado directamente a la placa, por lo que sus características son muy similares a la placa de referencia: 14 pines digitales (6 del tipo PWM), 6 entradas analógicas, interrupciones, SPI…

La placa se puede alimentar por baterías (a través del conector), por una fuente externa (conectando a los puntos marcados como + y - o soldando un conector macho *jack* en el borde izquierdo inferior) o por el pin VCC, siendo seleccionada de manera manual la fuente de entrada con un selector.

La placa ha sido diseñada por SparkFun, y se vende en dos versiones diferentes:

- 5,5 V: Con una velocidad de reloj de 16 MHz admite un voltaje de entrada de entre 5 y 12 V. Es la versión estándar con un consumo similar a la placa UNO.

- 3,3 V: Versión de bajo consumo a la que se le ha rebajado el voltaje de entrada hasta los 3,3 V. Su velocidad de reloj también se ha reducido hasta los 8 MHz pero manteniendo las características de memoria. Acepta valores de entrada de entre 3,35 y 12 V.

Sus dimensiones son menores que la placa UNO (5,2 x 5,3 cm) y su precio ronda los 18 € para la versión de 5 V y 20 para la de 3.3 V.

3.2.3.8.8. Arduino Pro Mini

Es la placa Arduino más pequeña. Se trata de la versión reducida de la placa Pro, también ha sido diseñada por el equipo de SparkFun.

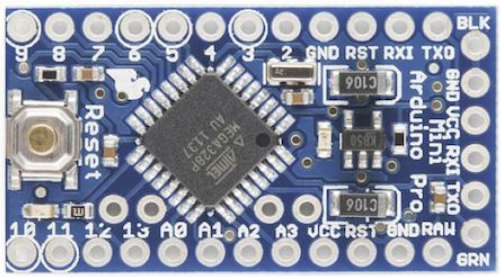


FOTO 3.16: ARDUINO MINI PRO

En su última revisión incorpora el microcontrolador ATmega328, en sus dos versiones, la de 3,3 V que funciona a 8 MHz, y la de 5 V a 16 MHz.

Esta placa destaca por sus delgadez (0,8 mm), sus reducidas dimensiones (3,3 cm de alto por 1,8 de ancho) y su peso de 2 gramos.

Pese a su tamaño incorpora el mismo número de conexiones (14 digitales y 6 analógicas) y reguladores de voltaje (admite entradas de entre 3,35 y 12 V para la versión de 3,3 V y entre 5 y 12 V par a la versión de 5 V).

Requiere también de un dispositivo controlador USB para realizar la programación y el auto reseteo.

Al igual que la placa Pro, todos los componentes que incorpora son del tipo SMD (*Superficial Mount Technology*) de dos capas, tecnología de montaje que reduce costes y permite embeber componentes en circuitos impresos de reducidas dimensiones.

Su precio está alrededor de 20 €.

3.2.3.8.9. Arduino Ethernet

La placa Arduino Ethernet está basada en microcontrolador ATmega328, con sus características estándar (16 MHz, 14 E/S digitales, 6 analógicas…).

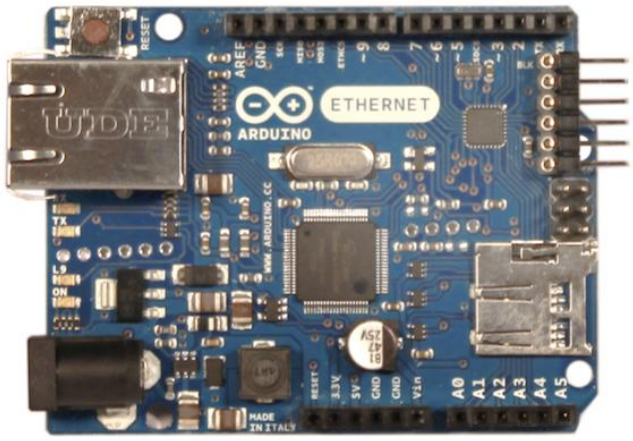


FOTO 3.17: ARDUINO ETHERNET

Sus diferencias con la placa UNO radican en que no incorpora un controlador USB, sino el controlador Wiznet Ethernet que dota a la placa de una interfaz Ethernet. Por este motivo los pines digitales 10, 11, 12 y 13 han sido reservados para realizar la comunicación a través de la clavija RJ45.

Así pues la programación se debe realizar bien utilizando los pines USB de la esquina noreste con un dispositivo controlador (cable FTDI, la placa USB a serial de SparkFun o el adaptador USB oficial de Arduino) o bien con un programador ISP externo a través de los pines ICSP.

La placa se puede alimentar mediante una fuente externa (puerto Jack o pines V_{in} y GND) con unos límites entre 6 y 20 V o bien a través de Ethernet con la tecnología PoE (*Power Over Ethernet*). Este protocolo, estandarizado por la norma IEEE 802.3af, es utilizado por algunos *routers* y *switches* para alimentar dispositivos de red utilizando el mismo cable Ethernet (RJ45). Al ser una tecnología con un alto voltaje requiere de la instalación de un módulo regulador adicional. Con el módulo la placa soporta tensiones de entre 36 y 57 V.

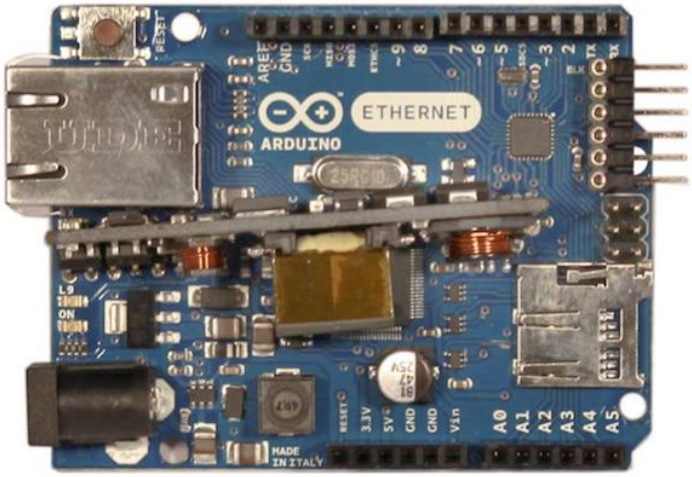


FOTO 3.18: ARDUINO ETHERNET W/ POE

Además, como podemos ver en la imagen, la placa incorpora un zócalo para tarjetas micro SD. En esta tarjeta podremos leer y almacenar datos con la librería SD. El puerto digital 4 se ha reservado para interactuar con la tarjeta SD.

Su tamaño es el mismo que la placa UNO (6,85 x 5,3 cm) y tiene un precio de 69 € (incluyendo el módulo PoE).

3.2.3.8.10. Comparativa

A continuación adjuntamos una tabla comparativa donde detallamos las características técnicas de cada placa:

	UNO	Mega	Mega ADK	Nano	Mini	LilyPad	Pro	Pro Mini	Ethernet
Microcontrolador	ATmega328	ATmega2560	ATmega2560	ATmega328	ATmega328	ATmega328V	ATmega328	ATmega168	ATmega328
Entradas/Salidas digitales	14	56	56	14	14	14	14	14	14
Salidas digitales PWM	6	15	15	6	6	6	6	6	4
Entradas analógicas	6	16	16	6	8	6	6	6	6
Puertos de serie HW (UART)	1	4	4	1	1	1	1	1	1
Frecuencia	16 MHz	16 MHz	16 MHz	16 MHz	16 MHz	8 MHz	8 MHz ó 16 MHz	8 MHz ó 16 MHz	16 MHz
Memoria Flash (KB)	32 (0,5 KB)	256 (8 KB)	256 (8 KB)	32 (2 KB)	32(2 KB b.)	16(2 KB b.)	32(2 KB b.)	16(2 KB boot.)	32 (0,5 KB b.)
SRAM	2 KB	8 KB	8 KB	2 KB	2 KB	1 KB	2 KB	1 KB	2 KB
EEPROM	1 KB	4 KB	4 KB	1 KB	1 KB	512 B	1 KB	512 B	1 KB
Conexión	USB	USB	USB	USB	No	No	No	No	Ethernet
Entrada de alimentación	Jack	Jack	Jack	No	Pin	Pin	Pin	Pin	Pin
ICSP	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Controlador USB Serial	Atmega16U2	Atmega16U2	Atmega16U2	FTDI FT232RL	No	No	No	No	No
Voltaje de funcionamiento	5 V	5 V	5 V	5 V	5 V	2,7 - 5,5 V	3,3 V ó 5 V	3,3 V ó 5 V	5 V
Voltaje de entrada (recomendado)	7 - 12 V	7 - 12 V	7 - 12 V	7 - 12 V	7 - 9 V	2,7 - 5,5 V	3,35 - 12 V ó 5 - 12V	3,35 - 12 V ó 5 - 12V	7 - 12 V
Voltaje de entrada (límite)	6 - 20 V	6 - 20 V	6 - 20 V	6 - 20 V	7 - 9 V	2,7 - 5,5 V	3,35 - 12 V ó 5 - 12V	3,35-12 V ó 5 - 12V	6 - 20 V
PoE (límite)	No	No	No	No	No	No	No	No	36 - 57 V
Corriente DC por pin E/S	40 mA	40 mA	40 mA	40 mA	40 mA	40 mA	40 mA	40 mA	40 mA
Corriente DC para pin 3,3V	50 mA	50 mA	50 mA	No	No	No	No	No	50 mA
Tamaño (largo/ ancho cm)	6,85 x 5,3 cm	10,1 x 5,3	10,1 x 5,3	4,31 x 1,85	3 cm	5 cm	5,2 x 5,3 cm	3,3 x 1,8 cm	6,85 x 5,3 cm
Precio	22 €	41 €	59 €	33 €	36 €	17 €	18 €	19 €	69 € (PoE)

3.2.3.8.11. Shields

En primer lugar tenemos los oficiales de Arduino:

- Arduino *Ethernet Shield*: Otorga a la placa Arduino conectividad a Internet a través de un cable RJ45. Está formada por un chip W5100 capaz de obtener velocidades de hasta 100 Megabits. Incluye además una ranura para tarjetas SD.

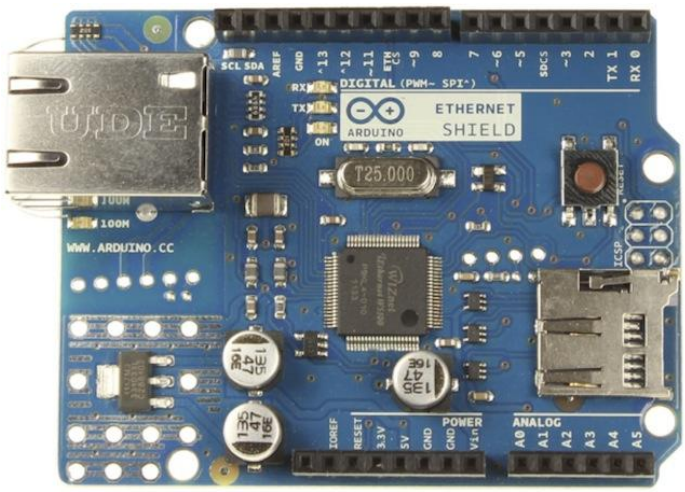


FOTO 3.19: ETHERNET *SHIELD*

- *Wireless* SD *Shield*: Permite a una placa Arduino comunicarse inalámbricamente usando un módulo inalámbrico XBEE (u otros módulos basados en éste), obteniendo un alcance de hasta 90 metros. También incluye un zócalo para una tarjeta SD.

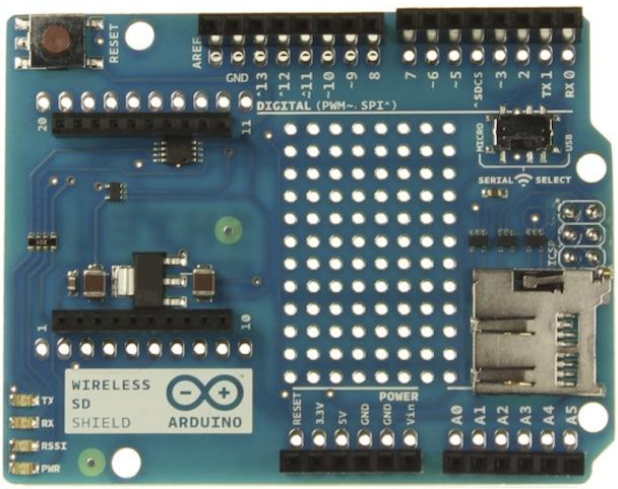


FOTO 3.20: WIRELESS* SD *SHIELD

- *Wireless Proto Shield*: Igual que la anterior placa ésta permite conectar módulos XBEE para realizar conexiones inalámbricas, pero la diferencia es que no tiene ranuras para tarjetas SD, sino que en lugar de esto tenemos un *grid* (pequeña placa de prototipado) ideal para realizar nuestros propios diseños.

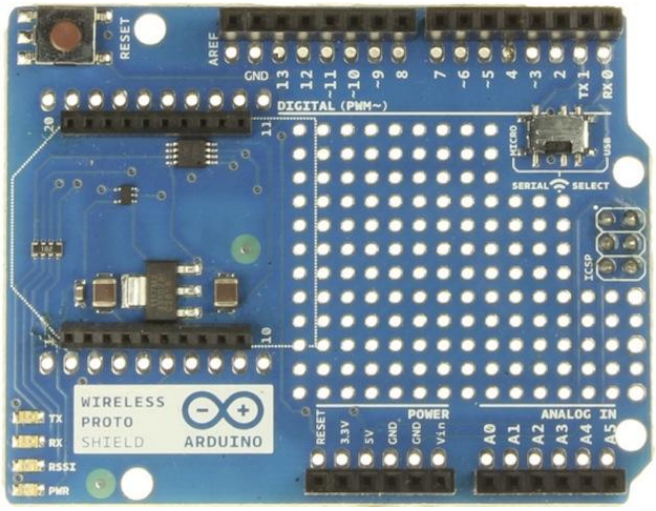


FOTO 3.21: PROTO *SHIELD*

- Arduino Motor *Shield*: *Shield* diseñado para controlar motores. Su principal componente es el chip L298 un controlador de doble puente (*dual full-bridge driver*) que nos permite controlar solenoides, motores de pulsos (*stepping motors*) y motores de corriente continua. Permite controlar la dirección (variando la polaridad) y velocidad dos motores independientemente.

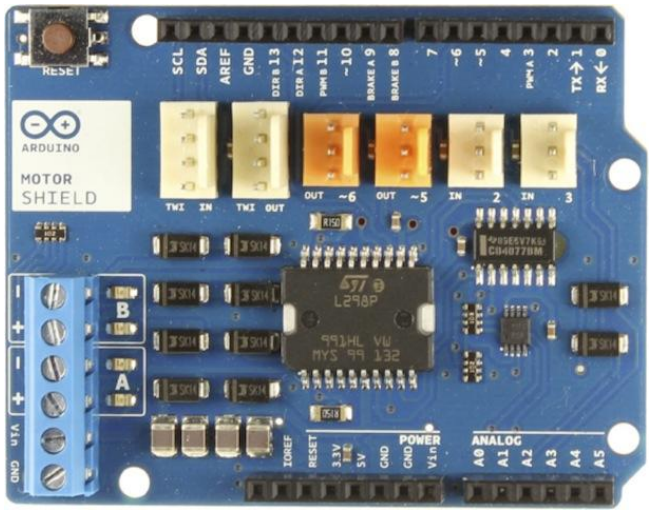


FOTO 3.22: ARDUINO MOTOR *SHIELD*

- Arduino Proto *Shield*: Esta placa es perfecta para realizar nuestros propios diseños. Básicamente se trata de una placa de prototipado con los pines necesarios para encajarla en Arduino.

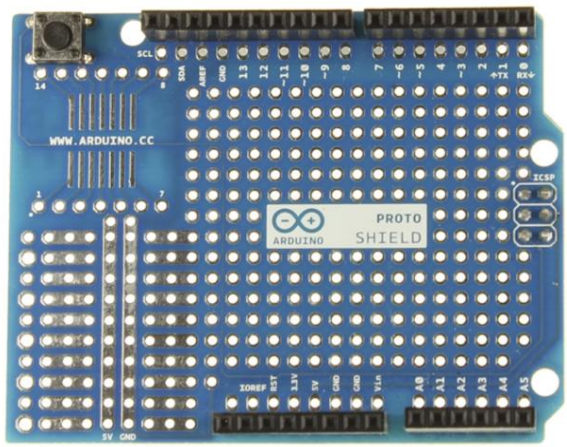


FOTO 3.23: ARDUINO PROTO *SHIELD*

Además de los *shields* oficiales, otros distribuidores como Libelium han lanzado sus propios *shields*:

- 3G/GPRS *Shield for Arduino* (3G + GPS): Se trata de uno de los *shields* más avanzados y caros (150 €) que encontramos. Este *shield* otorga conexiones de datos inalámbricas sobre redes 3G (WCDMA Y HSPA) y 2G (GPRS). Incluye un GPS interno que, ayudándose en triangulaciones, permite obtener la posición (A-GPS). Además dispone de un kit de audio y video para capturar video (640 x 480) y audio y emitir sonidos a través del altavoz.

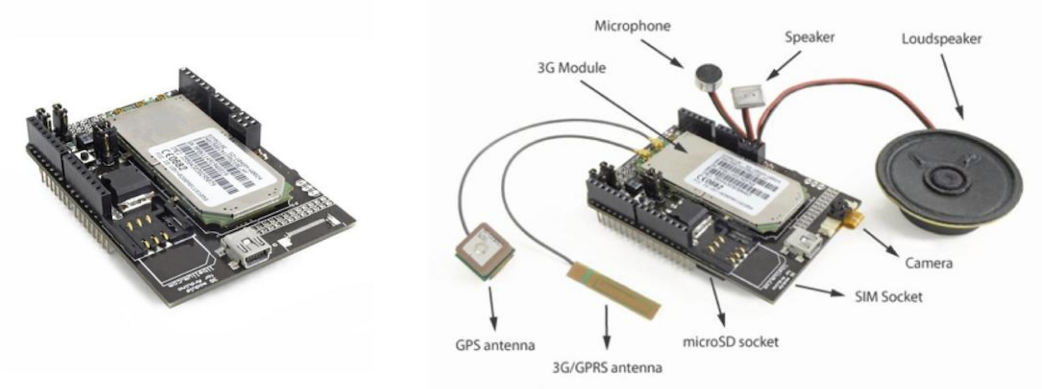


FOTO 3.24: *SHIELD* 3G/GPRS DE COOKING-HACKS

3.2.4. Futuro

La plataforma Arduino está en constante evolución. Los éxitos cosechados durante los últimos años han provocado que la popularidad de la placa se haya disparado. La mejor prueba de ello es que Google, en su conferencia de desarrolladores Google I/O del año 2011, presentó un entorno de desarrollo de accesorios para Android basado en Arduino, el Android Open Accessory Development Kit (ADK) cuya placa básicamente es un Arduino con un *shield* para funcionar en modo USB *host* y poder conectar con dispositivos cliente.

El equipo Arduino sigue trabajando para mejora tanto la parte software como hardware.

En la parte software el equipo ya ha liberado la versión 1.0 del IDE, congelando así un elevado número de librerías y *APIs*. Para las siguientes versiones, además de incluir las nuevas características de la plataforma Processing, el equipo quiere incluir de manera nativa nuevas funcionalidades, como son el protocolo SPI, nuevas formas de entrada, como de teclados, y la optimización de las librerías de bajo nivel, sobretodo a nivel de interrupciones.

Pero los mayores cambios se van a producir a nivel hardware. La plataforma Arduino incorporará en los próximos meses diversas placas nuevas:

- Arduino Leonardo: Será una placa Arduino de muy bajo coste basada en el micro Atmega32u4. Tendrá los mismos conectores que la placa UNO, pero con un diseño más simple y de menor coste. Esta nueva placa irá acompañada de nuevos drivers USB y el soporte para nuevos dispositivos de entrada, como ratones.

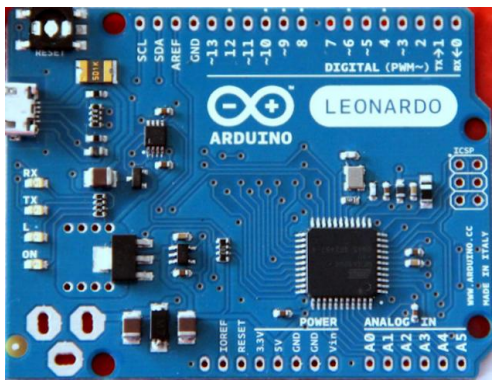


FOTO 3.25: ARDUINO LEONARDO

- Arduino Wifi *Shield*: Será un *shield* que dotará de comunicación wifi a cualquier placa Arduino. Utilizará un módulo producido por H&D Wireless que, unido a un potente procesador AVR32, permitirá implementar la pila TCP-IP. La idea es que la programación de protocolos de red siga la línea del *shield* Ethernet.

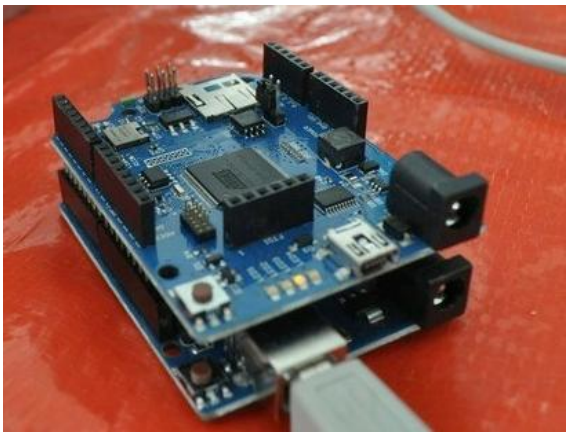


FOTO 3.26: ARDUINO WIFI *SHIELD*

- Arduino Lottie Lemon: Para potenciar el desarrollo de aplicaciones robóticas en la plataforma Arduino, Complubot (un grupo de aficionados españoles a la robótica) en colaboración con el equipo Arduino, ha desarrollado un robot circular con motores DC, pantallas, sensores y área de prototipado. Esta plataforma irá acompaña de diversas librerías que, utilizando el mismo IDE, facilitarán el desarrollo de pequeños robots Arduino.

FOTO 3.27: ARDUINO LOTTIE LEMON

- Arduino Due: Será uno de los mayores hitos de la plataforma Arduino ya que incorporará el microchip Atmel SAM3U, un micro con procesador ARM Cortex M3 a 96 MHz. Con esto tendremos una placa Arduino, del estilo Mega, con una arquitectura de instrucciones de 32 bits. Contará con 256 KB de memoria Flash, 50 KB de SRAM, 5 buses SPI, 2 interfaces I^2C, 16 entradas analógicas con una resolución de 12 bits y 56 pines digitales.

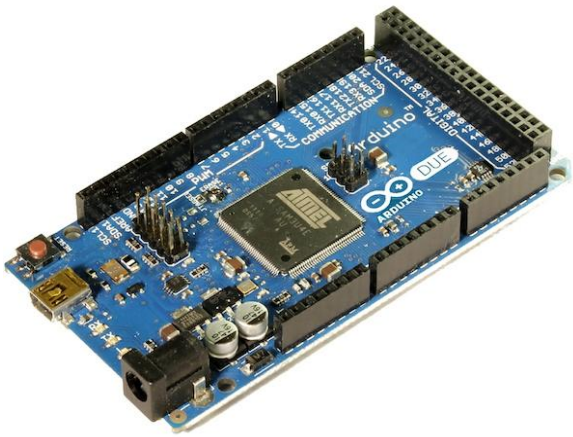


FOTO 3.28: ARDUINO DUE

Además la industria sigue aportando *shields* adicionales. Tal es el caso de la división de investigación y desarrollo de Movistar, Telefonica I+D, que está terminando un *shield* GSM para dotar de conectividad datos móviles a la placa Arduino. Este *shield* se diferenciará de otros similares en que aportará nuevas librerías para facilitar la creación de *sockets* y conexiones HTTP. Además, junto a Movistar, pretenden lanzar una tarifa de datos acorde a la filosofía de Arduino: de muy bajo y que impulse el internet de las cosas (*Internet of the things).*

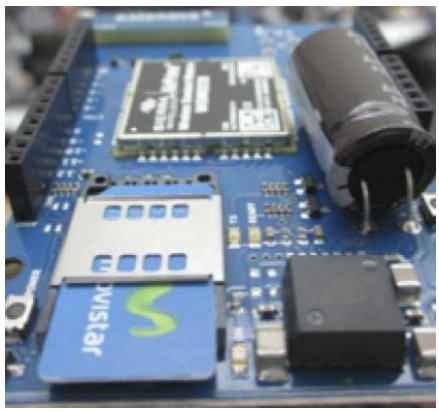


FOTO 3.29: TELEFÓNICA I+D GSM *SHIELD*

Ya por último destacar otras plataformas con tecnologías diferentes pero la misma mentalidad: ofrecer sistemas de prototipado abiertos y de bajo coste.

- Netduino: Es una plataforma electrónica abierta que utiliza el entorno de desarrollo .NET. Con un microcontrolador Atmel de 32 bits (el AT91SAM7X512, 48 MHz, 60 KB de RAM y 128 KB de Flash) el proyecto Netduino ofrece una placa 20 pines (que indiferentemente se pueden utilizar como digitales o analógicos), entre los cuales encontramos PWM, SPI, I^2C, RX, TX… El software requiere de la instalación de las librerías .NET 4.1 y el entorno de desarrollo de código abierto Netduino SDK.

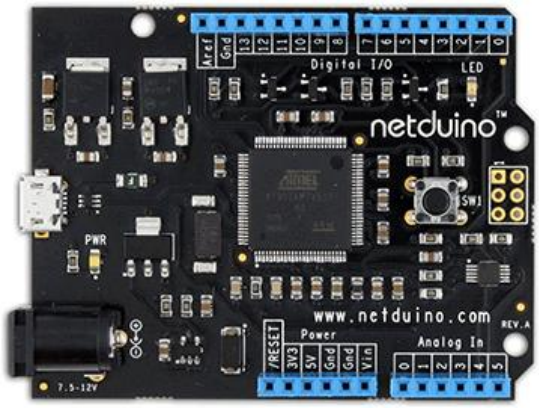


FOTO 3.30: NETDUINO

- Alternativas PIC: Como explicamos en la sección 9.6, el microcontrolador utilizado por Arduino es de la familia AVR, aunque también existen micros PIC. Para este tipo de microcontroladores están surgiendo diferentes proyectos, muy verdes todavía, que pretenden llevar el concepto de Arduino a los PIC. Hablamos de proyectos como Pingüino:

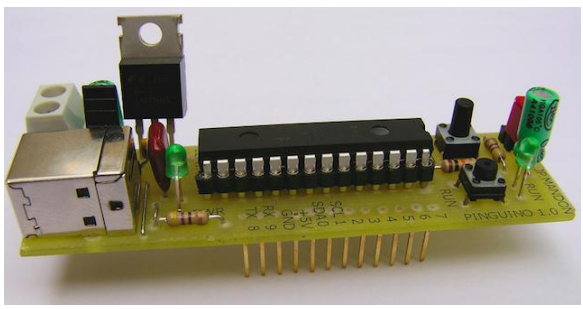


FOTO 3.31: PROYECTO PINGÜINO

O PowerJaguar:

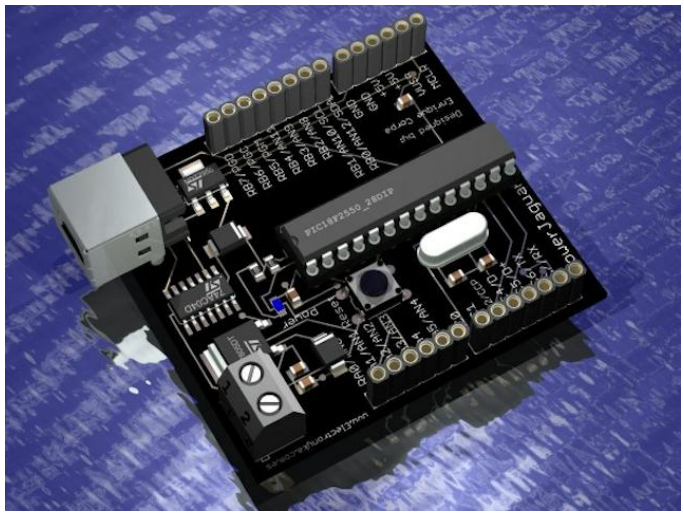


FOTO 3.32: POWERJAGUAR

3.3. Motivación

En este capítulo hemos podido estudiar el estado del arte de un par de disciplinas que se encuentran en un momento álgido: los sistemas de información ubicuos y las plataformas de prototipado.

Los SIU son el futuro inmediato de las tecnologías de la información. Es el paradigma computacional previo por el que las TIC deben pasar antes de llegar a la computación cuántica. La tecnología debe evolucionar de manera que lo cubra todo, siendo más importante abarcar todos los ámbitos de la vida real que conseguir una ingente potencia de cálculo.

Por otro lado estamos también aconteciendo al boom de las tecnologías móviles, con los *smartphones*, o teléfonos inteligentes, como referentes. En los próximos años los sistemas en movilidad (teléfonos inteligentes, tabletas y portátiles ultraligeros) coparan el tráfico de red y sustituirán a las estaciones convencionales de trabajo. Además las redes de datos móviles alcanzarán velocidades similares a las tecnologías cableadas y su cobertura rondará el 100% del territorio. Este crecimiento, unido a los nuevos sistemas operativos y procesadores de bajo consumo, ayudará a que plataformas como Arduino tengan un rápido crecimiento, tanto en número de dispositivos como en índice de adopción.

Esta tesis pretende aprovechar todas estas tecnologías emergentes para definir una línea de negocio poco explotada: ayudar a industrias tradicionales a dar un salto cualitativo y cuantitativo tecnológicamente hablando.

Además el proyecto que presentaremos a continuación pretende ser el inicio de un desarrollo que abarque múltiples de áreas de negocio, ya que tiene como principios la extensibilidad y aplicabilidad.

4. SYSREG: SISTEMAS Y SOLUCIONES PARA EL REGADÍO

Este capítulo presentará el proyecto Sysreg como una respuesta de las TIC a los problemas y desafíos presentados en los dos puntos anteriores. En primer lugar detallaremos como se germinó el proyecto, a continuación describiremos el proyecto de una manera global, marcando objetivos y prioridades, y finalmente detallaremos las características del proyecto, tanto funcionales como no funcionales.

4.1. Introducción

Nos encontramos en un ambiente tecnológico favorable que permite la creación de *startups* tecnológicas (empresas de nueva creación con un marcado carácter innovador): numerosos son los casos de nuevas empresas que con un equipo joven, una organización dinámica, poco capital y mucha ambición están obteniendo grandes resultados basándose en el crecimiento exponencial de clientes atraídos por la creatividad, el desarrollo continuo y abierto. Tal es caso de empresas como Twitter, Facebook, Instagram y Flickboard que ya tienen una valoración millonaria.

Es en esta situación donde cada uno debe de mirar a su alrededor y plantearse la idea de mejorar lo que tiene haciendo uso de nuevas tecnologías. No hablamos ni de importantes investigaciones ni de complejos desarrollos, sino de innovar.

Este es el objetivo de la presente tesis y en especial de proyecto Sysreg: utilizar todos los avances realizados durante los últimos años para aportar ideas innovadoras que mejoren un producto tradicional. Además si con este producto ayudamos a un sector en crisis, nuestra aportación será doble. Y el objeto de la tesis es el campo valenciano.

La idea de innovar en la agricultura no es nueva para el autor de esta tesis. Desde el inicio de su andadura en las ciencias de la computación el autor procuró utilizar sus conocimientos agrícolas para idear proyectos llamativos y creativos. Así en el año 2005 y 2006, en las asignaturas de Producción Multimedia, con el proyecto *Naranjas Arroba*, y en Estrategias y Nuevas Tecnologías con el proyecto *Taronline*, las ideas impulsar el comercio electrónico en cítricos triunfaron sobre otros modelos de

negocio. Esta idea se arrastró durante los siguientes años hasta el 2009, año en el que un equipo de 9 personas decidió seguir las riendas de esta idea para desarrollar un software de gestión de pedidos de naranjas (en Herramientas CASE y Métodos en Ingeniería del Software con la implementación del proyecto *Taronline*). Pero el carácter académico de estas iniciativas y el nulo soporte económico propiciaron que no se llevaran al mercado. Y el resultado ha sido que la industria citrícola que elegido este camino, el de la venta electrónica de cítricos, es la única que mantiene los beneficios.

La tesis que presentamos pretende ser un paso más en la idea de modernizar el campo valenciano. La idea surgió a finales del 2010, durante la realización del segundo módulo del Máster en el que se enmarca la tesis, como demuestra el proyecto *SAI* (Sistema Agrícola Inteligente) presentado en la asignatura Sistemas de Inteligencia Ambiental. Esta idea se ha plasmado en el proyecto Sysreg, una plataforma que pretende llegar a definir un sistema de información agrícola.

4.2. Descripción

Sysreg es un sistema de gestión agrícola remoto, inteligente y extensible que surge con la intención de introducir en el sector agrícola valenciano componentes tecnológicos.

El principal objetivo de este sistema es conseguir una mejora productiva, gracias a la monitorización en tiempo real y a la optimización de los recursos en función de las condiciones. Y todo ello utilizando únicamente dispositivos de muy bajo coste y de código abierto.

Aunque Sysreg fue ideado en el marco de la agricultura citrícola minifundista propia del Mediterráneo, el proyecto puede ser extrapolado para gestionar desde pequeños jardines hasta grandes industrias hortofrutícolas (e. g. viveros, semilleros…).

4.3. Características funcionales

El proyecto que presentamos tiene como prioridades 6 puntos. Estos pilares guiarán el desarrollo temprano y serán características básicas. El motivo de la definición de estas máximas es que son las carencias detectadas en la agricultura valenciana.

A continuación definiremos estos pilares:

4.3.1. Aprovechamiento de materias primas

El proyecto Sysreg surgió del estudio de las válvulas de riego a goteo instaladas en el campo valenciano. Cuando se empezó a instalar el riego a goteo se presumía de un ahorro de agua considerable: se hablaba de que con este método se focalizaba el riego y no se desperdiciaba agua.

Sin embargo esta modernización tiene lagunas importantes: los programadores son dispositivos simples, cerrados, alimentados con pilas, con pantallas LCD en los que programamos estáticamente el riego y caros. Además en caso de cualquier avería o situación anómala (i. e. lluvia, viento, cambios de temperatura) requieren de la intervención manual. Además el agricultor está poco familiarizado con el dispositivo, por lo que no se realiza un aprovechamiento total.

Por tanto el sistema ideado debe cumplir con el objetivo de sus antecesores: optimizar recursos. Sysreg aporta al riego por goteo dinamismo: el dispositivo está preparado para manejar el riego según las condiciones climáticas, identificar averías y actuar de una manera inmediata.

Esta optimización no es solo aplicable a la gestión de agua. Los sistemas de riego incorporan válvulas con efecto Venturi que permiten succionar otros líquidos por la fuerza de la presión del agua. Así mediante de estas válvulas podemos controlar abonos, productos fitosanitarios y productos de limpieza.

4.3.2. Productividad

A lo largo de la historia las grandes crisis se han superado con mejoras productivas y, en la actual sociedad de la información, las TIC aportan ese progreso productivo. Sin embargo en el ámbito agrario estas mejoras no se están introduciendo.

Con Sysreg se pretenden integrar los avances tecnológicos con el objetivo de mejorar la productividad agrícola. Esta mejora se consigue de la siguiente manera:

- Aumentado el número de parcelas gestionables: Con este sistema el agricultor puede gestionar un número mayor de parcelas, ya que tendrá identificadas cada una de ellas con sus diferentes sensores y actuadores.
- Reduciendo los tiempos de actuación: Con la monitorización en tiempo real y la actuación a distancia.

- Automatizando procesos: La mayoría de actuaciones de campo suelen ser repetitivas y estar basadas en buenas prácticas. Siendo así, es sencillo imaginar un software capaz de integrar este conocimiento y definir reglas rápidas y sencillas.
- Analizando resultados: El procesamiento y análisis de los resultados fortalece y optimiza los procesos de actuación.

4.3.3. Tiempo real

Los sistemas de riego modernos permiten al usuario actuar directamente sin requerir de otros servicios o personas. Además disponemos sensores meteorológicos de todo tipo que aportan datos precisos en tiempo real. Sin embargo estás dos herramientas no siempre van unidas de la mano, como muestran las estaciones de riego agrícolas.

Sysreg pretende relacionar los sensores con los actuadores, sin perder de vista el acceso inmediato a los dispositivos conectados a la placa.

4.3.4. Movilidad

Otro pilar muy importante de nuestro proyecto es la capacidad de interactuar a distancia con el sistema. Para ello, en las diferentes fases, incorporaremos mecanismos de acceso remoto. Estos mecanismos irán evolucionando en complejidad y funcionalidad.

4.3.5. Procesamiento

Sysreg incorporará paulatinamente características de los sistemas de información clásicos. Hablamos de procesos de captura, almacenamiento, procesado y presentación de datos que ayudarán al usuario a optimizar tareas y recursos. Para ello se definirán una lista de valores a estudiar (como producción, agua, abonos) que se almacenarán en una base de datos para, posteriormente, presentarlas en indicadores y gráficos.

4.3.6. Inteligencia

El proyecto del que habla la tesis implementará un sistema de reglas que automatice la toma de decisiones. Este tipo de reglas abarcan desde simples reglas de desactivación de riego (e. g. si está lloviendo → Desactivar el riego y la programación) a reglas más complejas en función del contexto (e. g. si estamos en invierno, ha habido una bajada brusca de la temperatura por debajo de los 0 grados y la humedad es baja → Activamos el riego para evitar el congelamiento del fruto).

Como hemos visto en el capítulo 2 las reglas que definen el contexto estarán definidas de una manera simple e intuitiva utilizando un lenguaje lógico y serán ejecutadas por un motor de inferencia.

4.4. Requisitos no funcionales

Además de las características anteriores, el sistema deberá cumplir con unos requisitos no funcionales que lo doten de versatilidad y lo diferencien de otros productos ya existentes:

4.4.1. Extensibilidad

La principal ventaja de nuestro proyecto es que, pese a que va a ser desarrollado y probado en un entorno citrícola, es capaz de gestionar cualquier parcela agraria que utilice elementos como válvulas de riego, variables meteorológicas, sensores analógicos y digitales, relés, transistores, servomotores… y aquí entran industrias como la floricultura, la horticultura, industrias agroalimentarias…

Sin necesidad de excesivos cambios nuestro hardware podrá funcionar con todo tipo de actuadores (relés, servomotores, altavoces, pantallas) y nuevos tipos de sensores.

Pero la extensibilidad de nuestro producto no se reduce solo al ámbito de aplicación, sino también es extensible a nivel software y hardware.

Nuestra placa ha sido desarrollada en un entorno colaborativo, como es la comunidad Arduino, en el que diariamente aparecen nuevas invenciones. Así por ejemplo para placas Arduinio existen ya módulos para *loggers* GPS, *displays* LCD, sensores de ultra sonidos, cuadracópteros teledirigidos…

Por otro lado el software está desarrollado en una arquitectura de 3 capas en la que encontramos a nivel más bajo el código de comunicación con la placa, en el medio la lógica de negocio y por arriba la interfaz gráfica. De este modo tenemos un software desacoplado en el que introducir nuevos cambios solo afecta a la capa correspondiente. Por ejemplo si quisiéramos interaccionar con otra placa, como Netduino, solo tendríamos que modificar la capa Arduino.

4.4.2. Coste

El proyecto Sysreg pretende ser una solución de bajo coste. Para ello utiliza componentes sencillos, funcionales y tecnológicamente independientes.

En el uso de sensores se recurre a dispositivos producidos en serie, que utilizan protocolos estándar y cuentan con un extenso ámbito de aplicación.

4.4.3. Open-source

El proyecto Sysreg se ha nutrido de multitud de proyectos libres. Por ello, y porque necesita de la aportación de la comunidad, será un proyecto libre.

Todo componente de la tesis se liberará bajo la misma licencia que su componentes, es decir, la documentación bajo la licencia *Creative Commons Atrribution Share-Alike* y el resto de software bajo la licencia de código abierto GNU GPL v3.

4.4.4. Multiplataforma

Para que un sistema sea portable y remoto es necesario que sea capaz de ejecutarse en la mayor cantidad posible de sistemas. La parte software de Sysreg será desarrollado en un lenguaje de programación multiplataforma, Java, con versiones personalizadas para cada sistema operativo.

Por otro lado el componente hardware, AlReg, pese a ser definido para la plataforma Arduino utilizará el entorno de desarrollo Processing, lo que lo dota de la flexibilidad necesaria para portarlo a otras arquitecturas.

5. ANÁLISIS Y DISEÑO DEL PROYECTO SYSREG

5.1. Introducción

Dadas las características del proyecto que lo hacen un sistema mixto (hardware y software), con tecnologías primerizas y numerosas funcionales, hemos decidido utilizar una metodología iterativa e incremental, que defina una arquitectura sólida sobre la que poder evolucionar y su desarrollo esté dirigido por las funcionalidades (casos de uso). Concretamente seguiremos una aproximación al paradigma de desarrollo de software RUP (*Rational Unified Process*, sección 9.1).

Para realizar el análisis definiremos un diagrama de contexto que enumerará los casos de uso. A continuación, especificaremos una arquitectura sobre la qué cimentar el proyecto, para terminar con un diagrama de clases y una descripción más profunda de los casos de uso. Todo ello siguiendo un diseño jerárquico, en el cual en primer lugar realizaremos el estudio a nivel global y posteriormente bajar al nivel de la primera fase.

5.2. Análisis

5.2.1. Diagrama de contexto

Para analizar las funcionalidades del sistema utilizaremos un diagrama de contexto (UML) agrupando los casos de uso por módulos o paquetes.

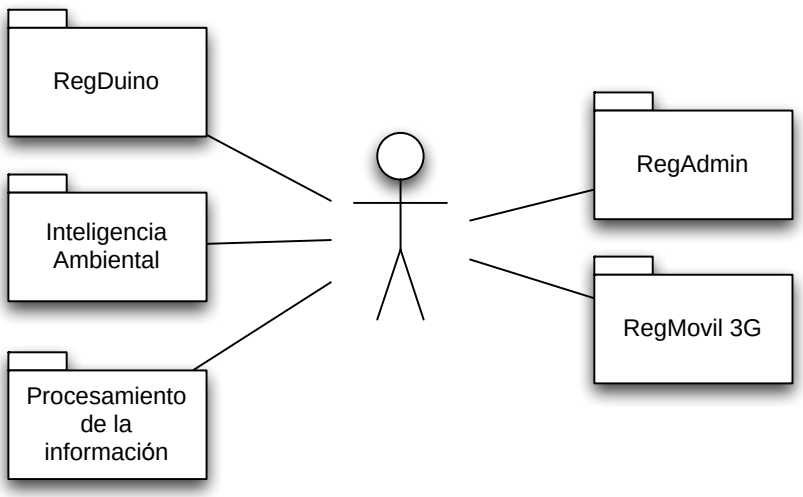


FIGURA 5.1: DIAGRAMA DE CONTEXTO

Al ser un proyecto grande hemos decidido que el desarrollo se realice de una manera evolutiva, implementando por fases las características de Sysreg. Además cada paquete incluirá componentes software y hardware, quedando relegado los detalles de cada componente al diseño específico de cada módulo.

5.2.2. Fase 1: RegAdmin

En la primera fase del proyecto englobaremos las funcionalidades básicas del sistema que, por motivos de descubrimiento y desarrollo incremental, el autor ha considerado punto de partida del proyecto. Esta fase corresponde al módulo RegAdmin, cuyos casos de uso son:

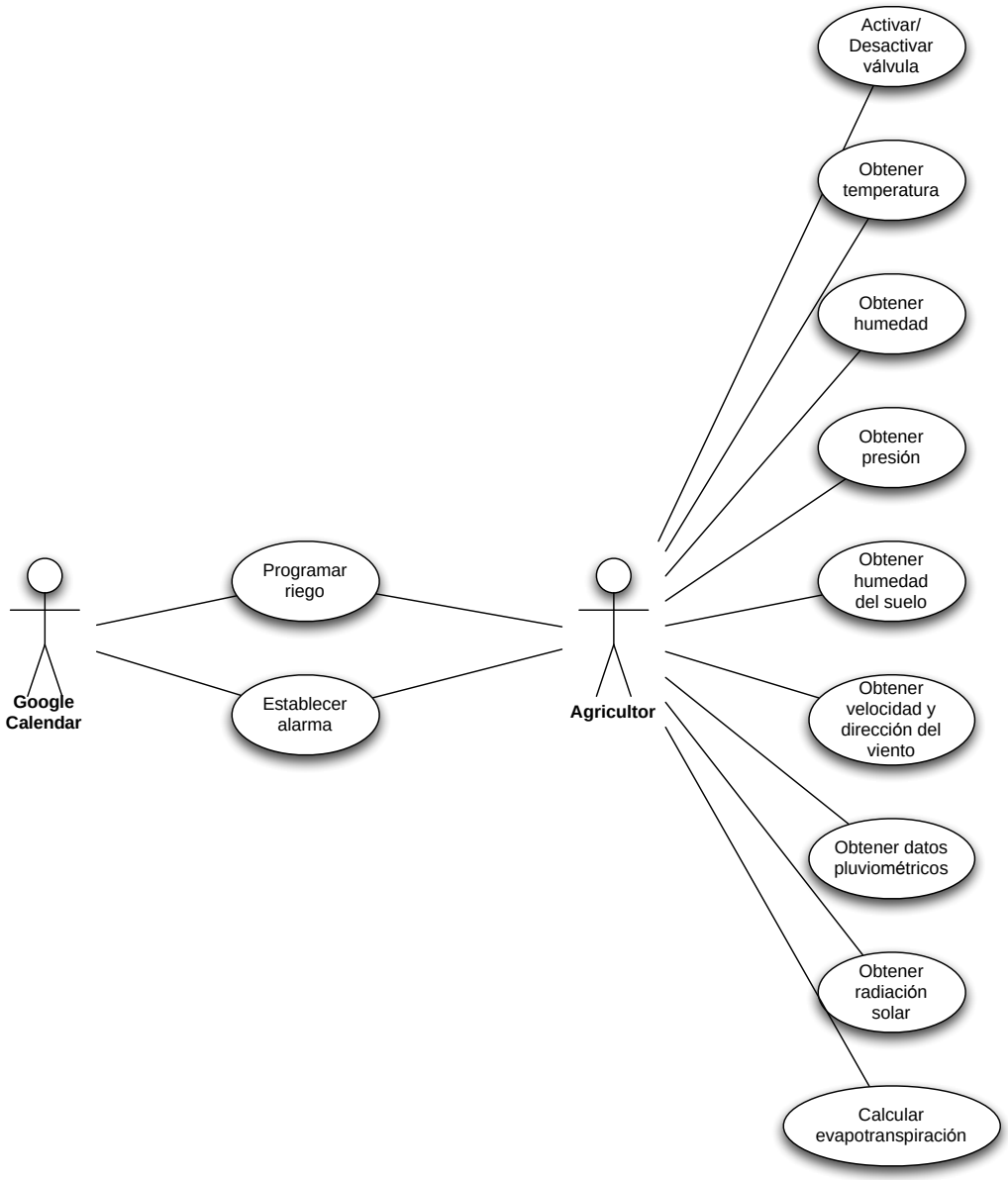


FIGURA 5.2: DIAGRAMA DE CONTEXTO DE LA FASE REGADMIN

5.2.3. Fases restantes

El resto de módulos son:

- RegDroid: Permite la conexión de la placa a dispositivos móviles con sistema operativo Android. Incluye cambios sustanciales a nivel hardware ya que necesita incorporar un controlador USB *host* que permita conectar dispositivos cliente USB (USB *device*). El motivo de este controlador adicional es que el kit de desarrollo de accesorios de Android (el Android ADK) solo permite conectar en la actualidad dispositivos que actúen como USB *host,* y el *driver* USB de Arduino trabaja en modo *device.* Además en la parte software se migrará la aplicación RegAmin al entorno de desarrollo Android y se ampliará el código de la placa Arduino para que sirva datos por el nuevo controlador USB.

- RegMovil 3G: El segundo gran hito del sistema Sysreg es la incorporación de conectividad de datos móviles a la placa Arduino. Con está funcionalidad podremos conectarnos a la placa desde cualquier lugar y de manera inalámbrica. En la parte física incorporaremos un *shield* de datos móviles, cuya elección dependerá de un análisis entre las diferentes opciones comerciales (apartado 3.2.3.8.11). A nivel de aplicación se diseñará una nueva API en código Arduino que permita establecer conexiones a través de Internet (mediante *sockets*, conexiones HTTP, FTP o SMTP). También se modificará el módulo RegAdmin para que permita conectar a través de red con la placa y proporcionar una solución Web para conectar desde el navegador.

- Procesamiento e inteligencia ambiental: Es el módulo más grande. Incluye las funcionalidades de procesamiento de la información y de inteligencia ambiental. El mayor esfuerzo en esta parte será el desarrollo software, ya que necesitaremos incluir multitud de componentes (bases de datos, motores de inferencia, servicios web…) e implementar el software capaz de aprovecharlos. Será la parte más costosa y su desarrollo se subdividirá en la parte del sistema de información agrario y la parte del sistema de inteligencia ambiental.

El análisis, diseño e implementación de estas fases quedan fuera del alcance de la tesis por motivos de tiempo. Su desarrollo se realizará en el orden descrito durante los próximos meses teniendo como objetivo ofrecer al final un sistema completo de gestión agraria, remoto e inteligente.

5.3. Diseño

El diseño en un paradigma de desarrollo de software como RUP viene determinada por definir una arquitectura robusta y especificar los casos de uso para poder seleccionar las funcionalidades a desarrollar.

5.3.1. Arquitectura

La arquitectura que vamos a describir pretende cimentar las bases de un desarrollo incremental y resistente a cambios. Hablamos de una arquitectura con componentes poco acoplados, alto grado de interoperabilidad, definición de hitos, separación de componentes, uso de estándares y uso de tecnologías estables y de código abierto.

A continuación detallaremos la arquitectura global del sistema, que incluirá todas las funcionalidades del sistema, tanto a nivel hardware como software.

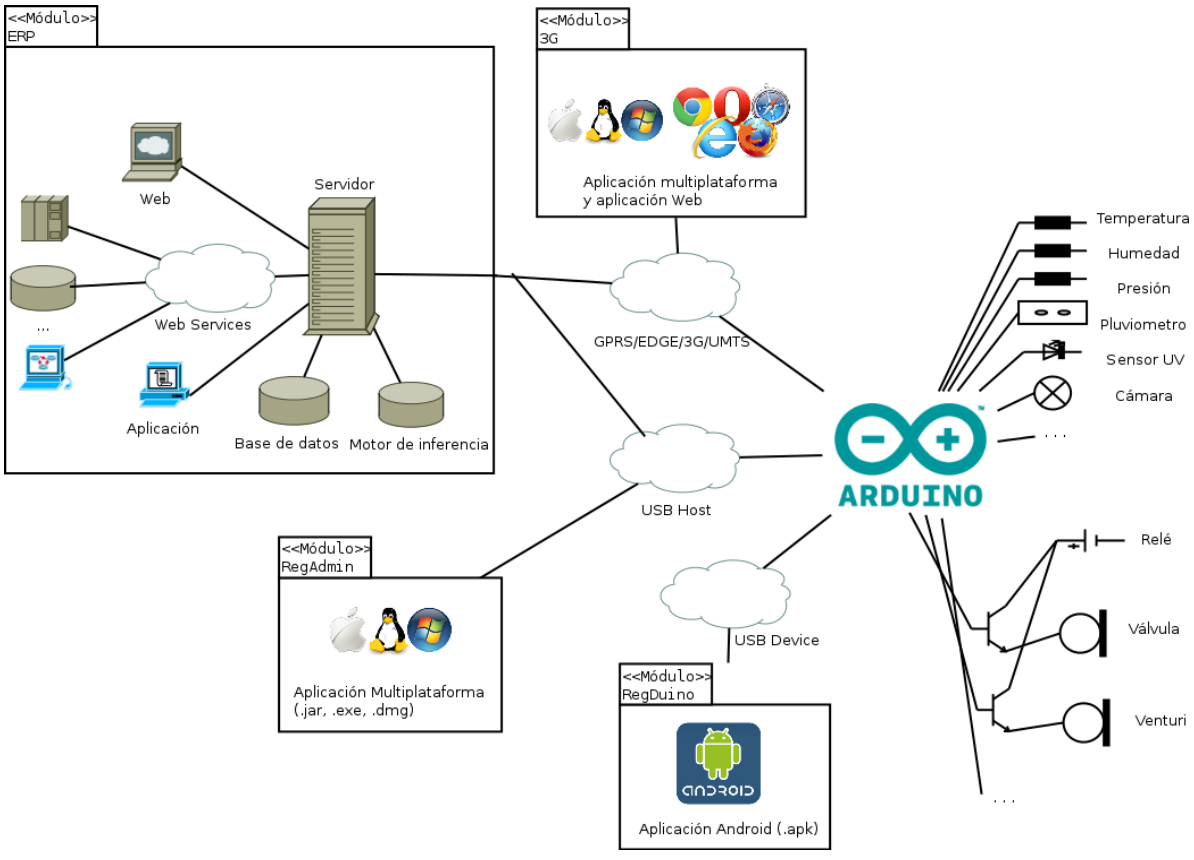


FIGURA 5.3: ARQUITECTURA

Como podemos ver se trata de una arquitectura dirigida por un dispositivo *middleware*, la plataforma Arduino. Independientemente de la placa seleccionada,

conectaremos a esta placa diferentes sensores y actuadores. La comunicación se realizará de 3 posibles formas, en función del protocolo de comunicación utilizado:

- USB *host*: Es el modo por defecto establecido para comunicarse con la plataforma Arduino. Utilizando el controlador USB de la placa conectaremos mediante librerías de conexión de serie, como por ejemplo la librería RXTX, o programas terceros como el IDE de Arduino.
- USB *device*: Mediante un *shield* que incorpore un controlador USB *host* o con la placa Arduino Mega ADK dotaremos a la placa del modo USB anfitrión. Esto principalmente nos permitirá conectar móviles Android en modo accesorio.
- GPRS/3G: Definiendo una arquitectura de red (mediante *sockets,* conexiones HTTP, FTP o SMTP) utilizaremos una placa con zócalo para tarjetas SIM que permita establecer conexiones inalámbricas en redes 2G ó 3G

En la parte software del sistema definiremos diferentes aplicativos para cada interfaz de comunicación:

- Aplicación multiplataforma para interaccionar con la placa por USB
- Aplicación Android que trabaje en modo accesorio con la plataforma Arduino
- Aplicación web y de escritorio para conectarse a través de redes de datos móviles.
- Aplicaciones de inteligencia ambiental y procesamiento de datos.

5.3.2. Fase 1: RegAdmin

La primera fase del proyecto tiene como objetivo desarrollar una placa que permita gestionar una parcela agrícola a través de sensores y actuadores, e implementar un software que establezca la conexión con la placa y gestionar de una manera visual las funcionalidades básicas.

Esta fase incluye el desarrollo de una parte software (el aplicativo RegAdmin y el *sketch* AlReg) y de los componentes hardware necesarios para implementar los casos de uso.

5.3.2.1. Arquitectura hardware

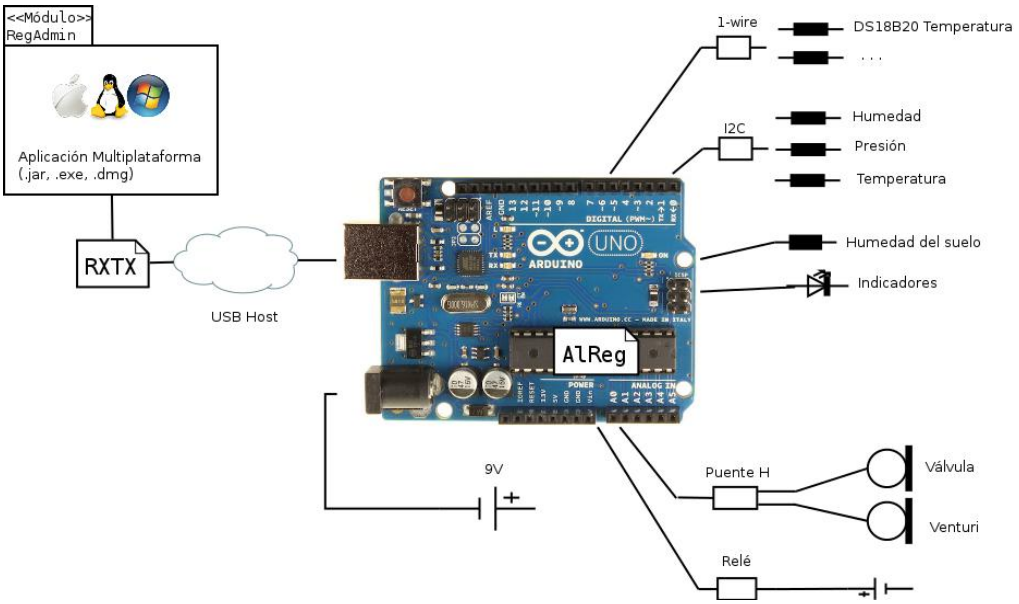


FIGURA 5.4: ARQUITECTURA HARDWARE DE LA FASE REGADMIN

La placa escogida para esta fase es la Arduino UNO, principalmente por ser una placa versátil, con suficientes entradas y salidas, bajo consumo, tamaño contenido y de bajo coste.

A esta placa conectaremos dos grupos de dispositivos: sensores y actuadores. Pero para facilitar la adición de nuevos componentes y mejorar la interoperabilidad, se van a utilizar protocolos de bajo nivel. Concretamente se van a utilizar, siempre que sea posible, dispositivos que trabajen con los protocolos I^2C y 1-Wire, relegando el uso de componentes fuertemente acoplados a dispositivos específicos, como el sensor de humedad del suelo.

Por tanto los sensores escogidos han sido:

- Termómetro digital Maxim DS18B20 (1-Wire).
- Barómetro BMP085 (I^2C).
- Humedímetro HH10D (I^2C).
- Humedímetro del suelo.
- Anemómetro direccional (1-Wire).
- Pluviómetro (1-Wire).

- Piranómetro.

Mientras que como actuadores tendremos:

- Relés: Para activar y desactivar dispositivos conectados a corriente alterna.
- Transistores: Interruptores digitales de corriente continua
- Puentes H: Conjunción de transistores que permiten variar el sentido de la corriente. Útiles para controlar motores y activar o desactivar válvulas de riego con solenoides de tipo *latch*.

La placa será alimentada por la toma de corriente de la placa Arduino con un voltaje de 9V. Así podremos alimentar la placa independientemente de la conexión USB, con un voltaje recomendado y de paso tener una salida suficiente para activar las válvulas de riego.

5.3.2.2. Arquitectura Software

A nivel software la aplicación del módulo RegAdmin seguirá una arquitectura por capas, concretamente de 3 niveles: interfaz de usuario, lógica de negocio y acceso a datos.

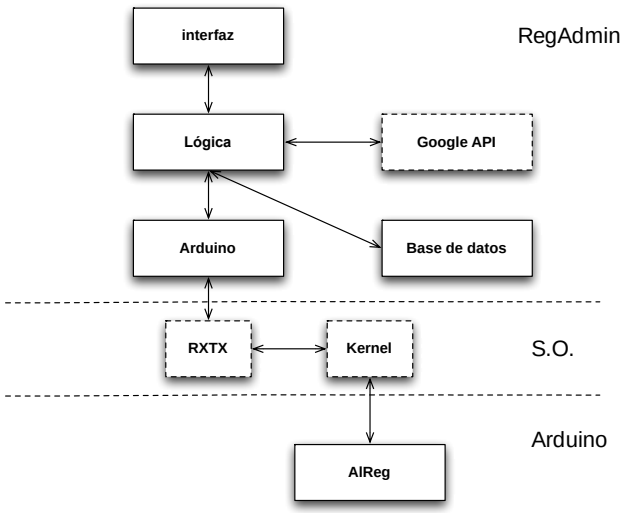


FIGURA 5.5: ARQUITECTURA SOFTWARE DE LA FASE REGADMIN

El lenguaje de programación elegido será Java, por ser un entorno de programación multiplataforma (la aplicación será compatible con sistemas Windows, OS X y Linux además de ser fácilmente migrable a Android), con un amplio soporte

(tanto de forma oficial por Oracle como por la comunidad de software libre) y con una gran cantidad de librerías. Se utilizará Eclipse como entorno de desarrollo (sección 9.5).

La parte visual se desarrollará con las librerías Swing de Java.

En capa negocio se implementará la lógica de los casos de uso, utilizando la API de Google para poder implementar la programación de riegos, ya que estableceremos los riegos a través de eventos de calendario con Google Calendar.

En la parte de acceso a datos tendremos por una parte la clase de acceso a la placa Arduino y por otra la de acceso a bases de datos.

De especial interés es la comunicación entre la placa y el módulo RegAdmin. Como se ha visto en la arquitectura global del sistema, la comunicación con el módulo RegAdmin se realizará definiendo una interfaz USB *host*. Para ello se utilizará la librería nativa RXTX que, mediante llamadas al sistema operativo, establecerá la comunicación con el dispositivo. Luego, en la parte Arduino, se definirá un protocolo de comunicación en lenguaje Processing para intercambiar datos con el programa. Este protocolo se implementará en el *sketch* AlReg y los mensajes se enviarán de la siguiente forma:

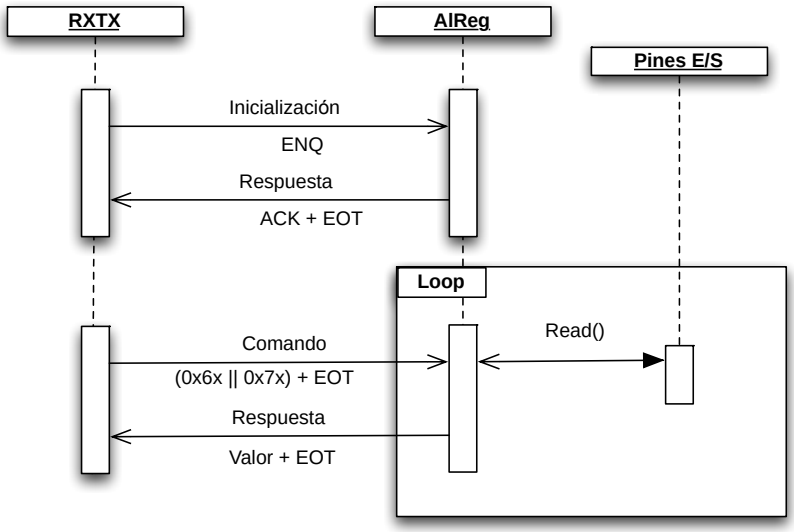


FIGURA 5.6: DIAGRAMA DE SECUENCIA DE LA COMUNICACIÓN ENTRE REGADMIN Y ALREG

Todo este desarrollo ser realizará utilizando la plataforma Google Code (anexo 9.3) para la gestión del proyecto, utilizando un servidor de control de versiones Subversion como repositorio.

5.3.2.1. Diagrama de clases

Las principales clases que encontramos en el sistema Sysreg hacen referencia a la gestión de alarmas y a la identificación de sensores.

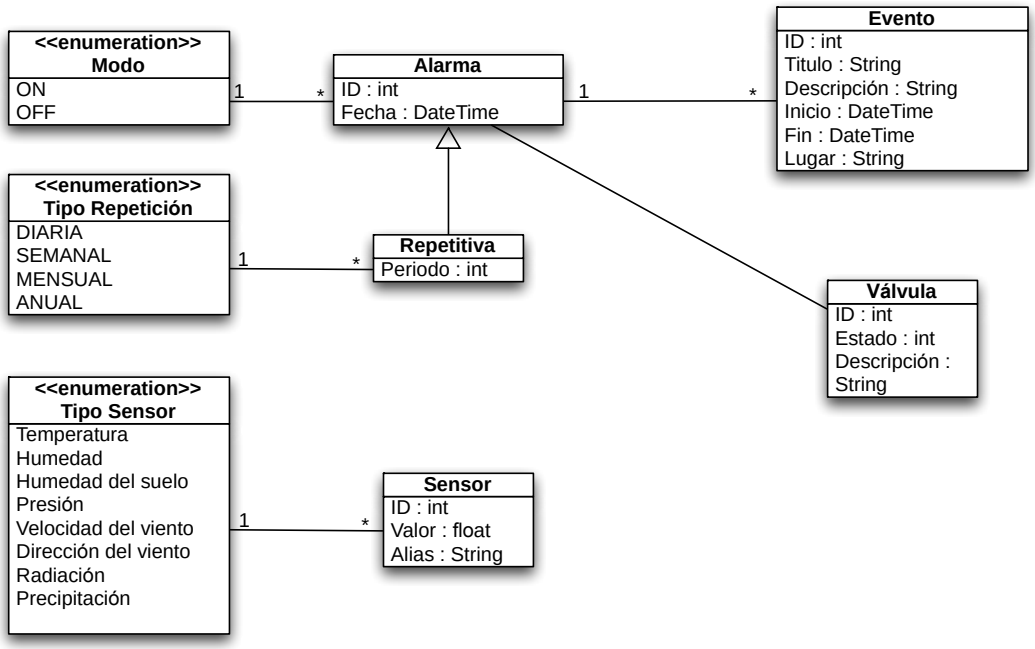


FIGURA 5.7: DIAGRAMA DE CLASES DE LA FASE REGADMIN

Las clases *Sensor* y *Válvula* se incluyen para definir pseudónimos a los sensores. Las clases *Alarma* y *Evento* se utilizarán para gestionar los eventos de calendario que dirigen el riego: como se va a utilizar un calendario en la nube para gestionar el riego, son necesarias operaciones sobre los eventos (modo *online*) y transformaciones a alarmas (modo *offline).*

5.3.2.2. Casos de uso

Las funcionalidades básicas que implementa la primera fase del proyecto Sysreg vienen definidas por los siguientes casos de uso:

- Activar/desactivar válvula: El usuario podrá seleccionar una válvula y controlarla en función del contexto, teniendo en consideración si existe algún proceso (e. g. una alarma) que haya modificado su comportamiento. Deberá existir también la posibilidad de forzar la activación o desactivación.

- Obtener temperatura: Identificando el termómetro el usuario obtendrá la temperatura en grados centígrados con una precisión de al menos un decimal.
- Obtener humedad: Devolverá la humedad relativa del aire del humedímettro seleccionado. Su resultado se expresará en tanto por cien.
- Obtener presión: Calculará la presión atmosférica en pascales (Pa). Además indicando la altura sobre el nivel del mar se obtendrá una estimación del tiempo.
- Obtener humedad del suelo: Obtendrá porcentaje de agua en el suelo.
- Obtener velocidad y dirección del viento: Retornará la velocidad, en kilómetros por hora, y la dirección del viento.
- Obtener datos pluviométricos: Consulta que devolverá la cantidad de litros por metros cuadrados que se están recogiendo en un instante dado.
- Obtener radiación solar: A través de un piranómetro calculará el índice de radiación.
- Calcular evapotranspiración de referencia: Aportando los datos necesarios (temperatura, velocidad y dirección del viento, humedad del aire y radiación solar) obtendremos las necesidades de riego independientes del cultivo. Para realizar este cálculo se hará uso de la fórmula de la evapotranspiración de referencia (ET_0) detallada en el apartado 8.3.1.2.2.
- Programar riego: Utilizando un calendario, el agricultor podrá programar eventos de riego, indicando la válvula, la hora de inicio y la hora de fin.
- Establecer alarma: El usuario podrá establecer alarmas de encendido y apagado. Estas alarmas se pueden programar de manera repetitiva, con un periodo diario, semanal, mensual o anual. La generación de alarmas también podrá ser de manera automática, exportando los eventos de riego a alarmas, definiendo un modo sin conexión de gestión del riego.

6. IMPLEMENTACIÓN DE LA FASE REGADMIN

A continuación se detalla como se ha implementado el módulo RegAdmin. En primer lugar definiremos una hoja de ruta basándonos en el diseño incremental del capítulo anterior.

El resto de la sección lo dedicaremos a detallar, especificando solo las partes más importantes del código, como se han implementado los casos de uso.

6.1. Planificación

Como vimos en el capítulo 6, Sysreg es un proyecto grande que se va a diseñar de una manera incremental. De los 4 grandes módulos que agrupan las funcionalidades, se tiene definida la siguiente hoja de ruta:

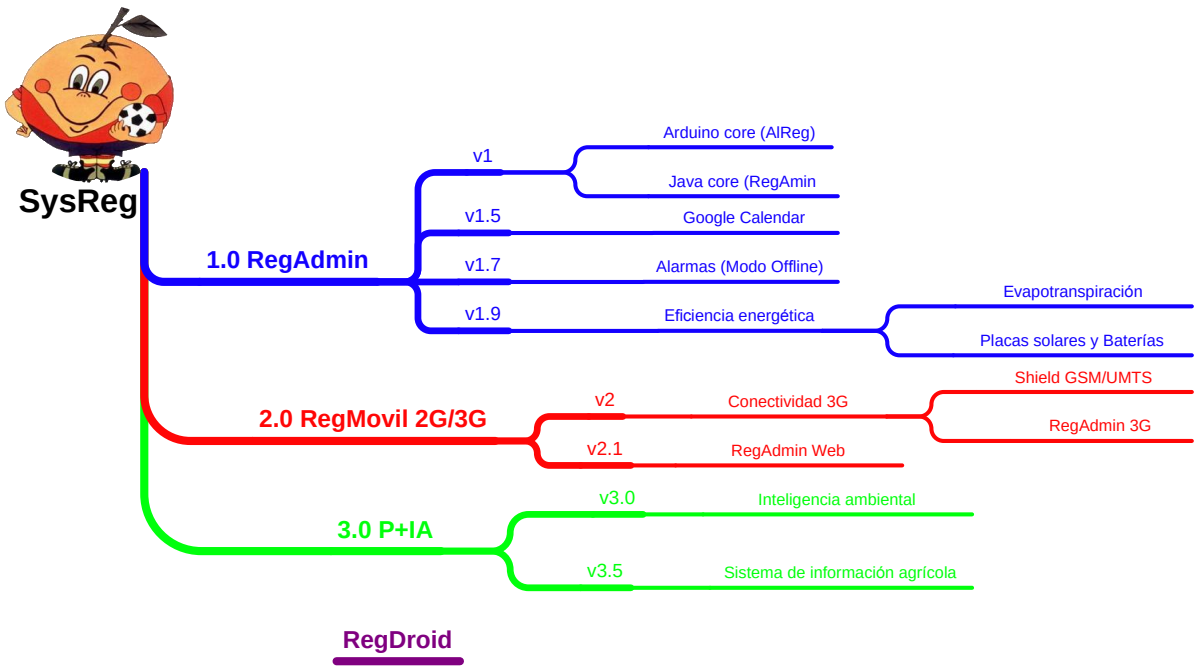


FIGURA 6.1: HOJA DE RUTA DEL PROYECTO SYSREG

Sin embargo la actual tesis solo va a cubrir una parte de la fase 1 del proyecto que engloba de las funcionalidades primarias, que permitirán construir una unidad funcional básica.

Además los casos de uso se van a definir para un número reducido pero suficiente de sensores y actuadores, los necesarios para poder gestionar un primer escenario de despliegue: la primera fase del proyecto permitirá gestionar una válvula de riego y acceder a la información de los sensores de temperatura, humedad relativa del aire, humedad del suelo y presión atmosférica.

Por tanto quedan fuera de la tesis la implementación de los casos de uso relacionados con los datos del viento, los pluviométricos, la radiación solar y, por extensión, el cálculo de la evapotranspiración de referencia.

A continuación detallaremos la implementación realizada en orden cronológico de cada caso de uso, diferenciando la implementación física, en la que presentaremos los componentes y los diagramas de conexión, y la implementación lógica, con sus algoritmos y librerías utilizadas.

6.2. Activar/desactivar riego

El primer caso de uso implementado ha sido la gestión de las válvulas. Con esta funcionalidad el agricultor podrá activar y desactivar las válvulas de riego de su parcela agrícola. En esta primera iteración se permitirá la gestión de una única válvula de riego.

6.2.1. Hardware

Dadas las alternativas comerciales en cuanto a tipos de válvulas de riego disponibles, tenemos válvulas de activación por pasos, por corriente y por pulsos, hemos decidido seleccionar el tipo que más se ajuste a las características de nuestro proyecto: bajo coste, bajo consumo y sencillez.

Por estos motivos para la gestión de válvulas hemos optado por electroválvulas de corriente continua activadas por pulsos de corriente. Estas válvulas están formadas por dos componentes principales:

- Válvula: Componente, generalmente de material plástico, que conduce el liquido en el sentido deseado. Como muestra la imagen, en función del estado de una pequeña obertura en la parte superior, la válvula permite el flujo a través del conducto, es decir, si se ocluye el orificio de la válvula, la membrana se cerrará e impedirá la comunicación.

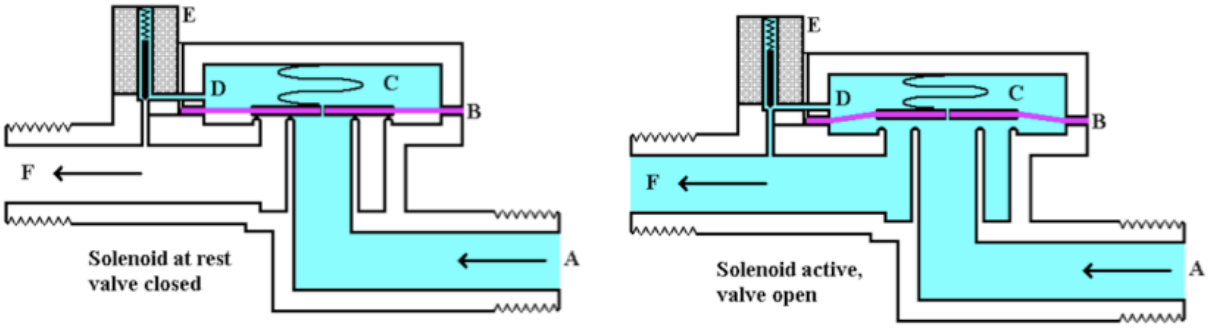


FIGURA 6.2: FUNCIONAMIENTO DE LAS ELECTROVÁLVULAS

- Solenoide: Se trata de una bobina que convierte la energía eléctrica en energía mecánica para actuar sobre la válvula.

FOTO 6.1: ELECTROVÁLVULA CON SOLENOIDE *LATCH* 9 V DE HUNTER

Si el solenoide es de tipo *latch* (pulsos) con la aplicación de un pequeño pulso en un sentido o en otro podremos abrir o cerrar la válvula. Además estas válvulas permiten funcionar también en modo continuo, es decir, existe un sentido en el que si existe corriente la válvula se abre, y en el momento en que ésta cesa la válvula se cierra (este sentido coincide con el del pulso de cierre).

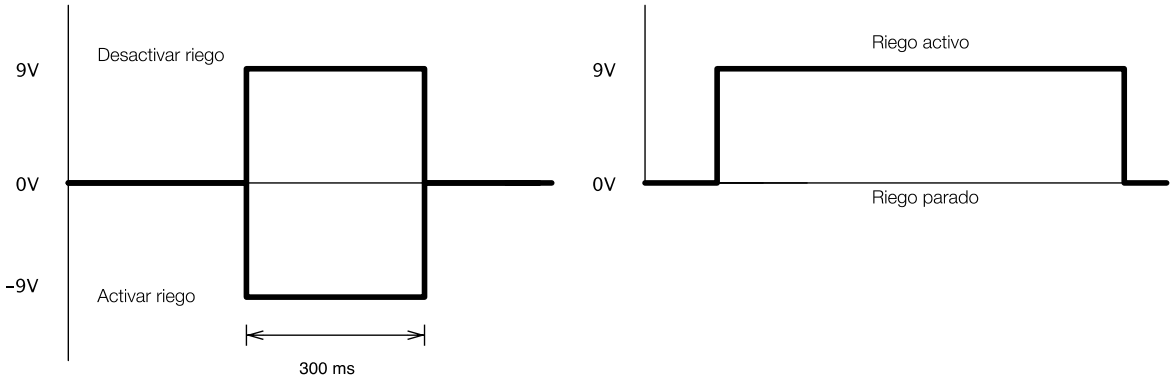


FIGURA 6.3: MODOS DE FUNCIONAMIENTO DE LA ELECTROVÁLVULA *LATCH*

Para gestionar este tipo de válvulas tenemos dos opciones: utilizar transistores o utilizar puentes H. El uso de transistores (*switch* digitales) condiciona el funcionamiento de la electroválvula a su modo continuo, es decir, con el transistor solamente podremos encender y apagar el flujo de electricidad en un sentido, de manera que cuando esté a nivel alto la válvula permitirá el riego.

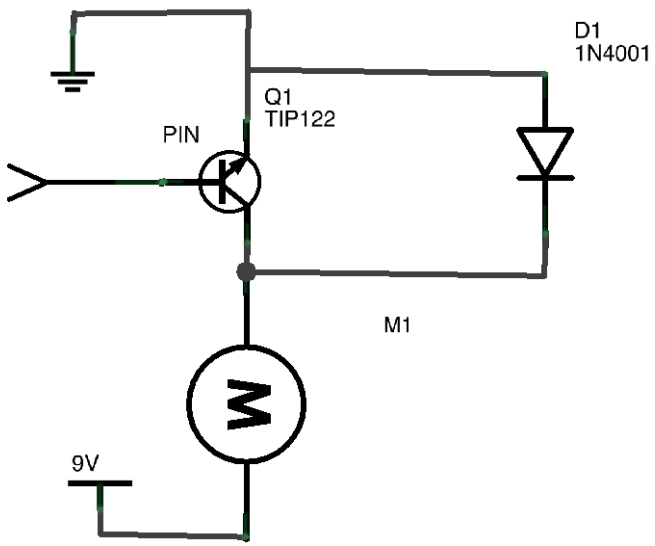


FIGURA 6.4: ESQUEMA DEL SOLENOIDE CONTROLADO CON UN TRANSISTOR

El uso del transistor simplifica el circuito, ya que solamente tenemos que utilizar un transistor, por ejemplo el TIP122 y conectar los componentes como muestra el esquema. Pero en cambio tiene un importante inconveniente: el modo continuo implica un consumo eléctrico elevado.

La utilización de puentes H es mucho más eficiente. Permite trabajar en modo *latch*. Los puentes H se utilizan en la electrónica principalmente para girar un motor eléctrico en varios sentidos. Está formado internamente por 4 transistores que, según su estado, modifican el flujo de corriente.

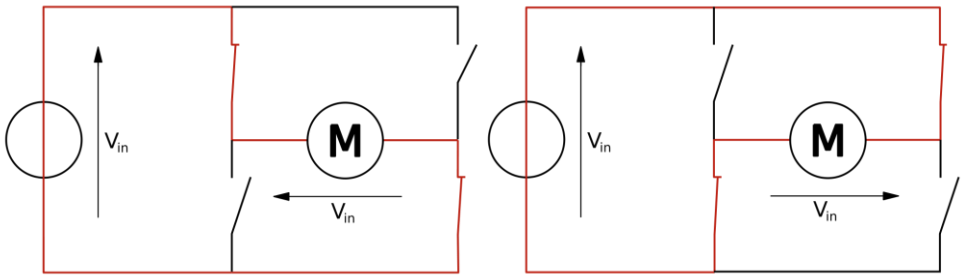


FIGURA 6.5: MODOS DE FUNCIONAMIENTO DEL PUENTE H

Así pues conectaremos a nuestra placa Arduino un puente H, concretamente el *driver* L293NE, que permita gestionar dos válvulas de riego. El esquema de conexiones es el siguiente:

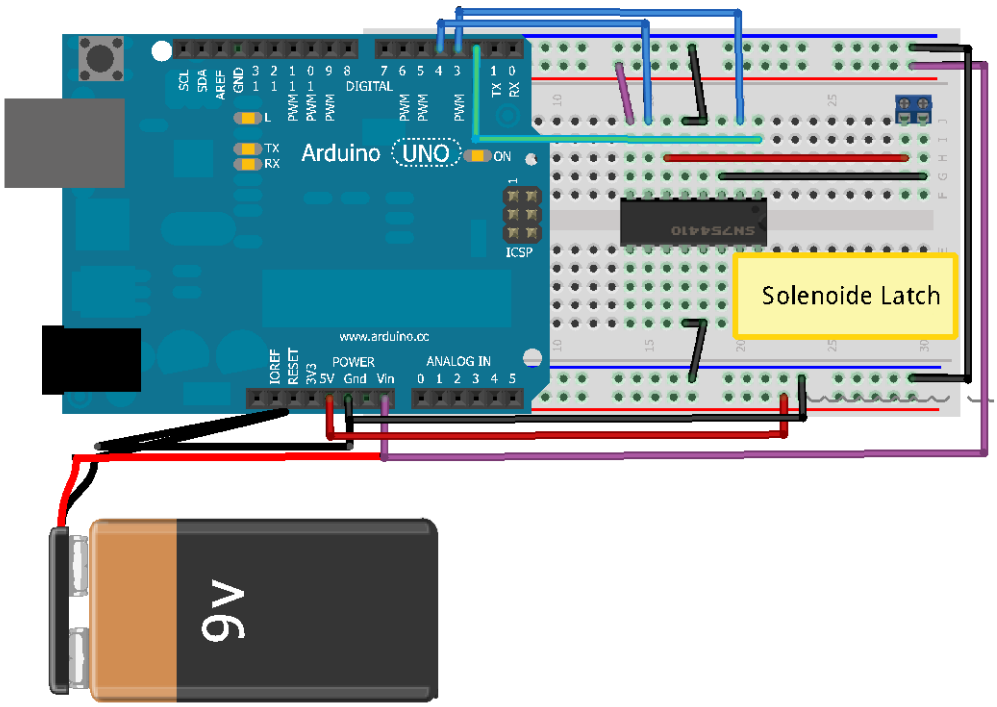


FIGURA 6.6: DIAGRAMA DE CONEXIONES DEL CASO DE USO ACTIVAR/DESACTIVAR RIEGO

6.2.2. Software

6.2.2.1. AlReg

Una vez realizadas la conexiones físicas pasaremos a programar el microcontrolador con un *sketch* que defina una pequeña API entre el micro y el aplicativo, de manera que al leer la placa Arduino una orden codificada en caracteres ASCII, ésta devuelva el resultado o actué en consecuencia. Además esta API deberá incluir el código de inicialización para establecer la conexión.

Hexadecimal	Decimal	ASCII	Descripción
0x05	5	ENQ	Establecer conexión
0x64	100	d	Activar riego
0x65	101	e	Desactivar riego

TABLA 6.1: COMANDOS DEL CASO DE USO ACTIVAR/DESACTIVAR RIEGO

Como podemos ver en el siguiente fragmento de código, para abrir o cerrar la válvula activamos los pines del puente H durante 300 milisegundos y después desactivamos la corriente. Así enviamos un pulso de 9 V al solenoide.

```
void loop(void){
byte present;
byte data[12];
byte addr[8]; // Identificador del sensor
int count = Serial.available();
if(count > 0){
        command=Serial.read();
        switch (command){
        case 0x05: // Solicitud de conexión: ENQ
                Serial.write(6);// Código de inicialización, ACK
                Serial.write(4);// EndOfTransmission (ASCII)
                break;
        case 0x64: // activarRiego: d 100 0x64
                riego = 1;
                digitalWrite(enable, HIGH);  // Establecer leg 1 del H-bridge a high
                digitalWrite(motor1Pin, LOW);   // Establecer leg 1 a low
                digitalWrite(motor2Pin, HIGH);  // Establecer leg 2 a high
                delay(300);
                digitalWrite(enable, LOW);  // Establecer leg 1 a low
                break;
        case 0x65:  // desactivarRiego: e 101 0x65
                riego = 0;
                digitalWrite(enable, HIGH);  // Establecer leg 1 del H-bridge a high
                digitalWrite(motor1Pin, HIGH);  // Establecer leg 1 a high
                digitalWrite(motor2Pin, LOW);   // Establecer leg 2 a low
                delay(300);
                digitalWrite(enable, LOW);  // Establecer leg 1 a low
                break;
}
```

6.2.2.2. RegAdmin

Por la parte del software multiplataforma RegAdmin nos encontramos en primer lugar con un prerrequisito: debemos establecer una comunicación serie con la placa para enviar los comandos establecidos. Por ello en la capa Arduino incluiremos las llamadas necesarias para, utilizando la librería RXTX, conectarnos por USB con la placa Arduino.

```
Enumeration portEnum;
portEnum = CommPortIdentifier.getPortIdentifiers();
while (portEnum.hasMoreElements()) { // Iteramos buscando el puerto
	CommPortIdentifier currPortId = (CommPortIdentifier) portEnum.nextElement();
	if (currPortId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
		String curr_port = currPortId.getName();
		System.out.println(curr_port);
		serialPort = (SerialPort) currPortId.open("RegAdmin",TIME_OUT);
		serialPort.setSerialPortParams(DATA_RATE,
				SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
				SerialPort.PARITY_NONE);
		// Abrimos los flujos de comunicación
		input = serialPort.getInputStream();
		output = serialPort.getOutputStream();
		serialPort.addEventListener(this);
		serialPort.notifyOnDataAvailable(true);
		output.write(0x05);
		byte[] res = this.leerArduinoBytes();
		if (res != null && res[0] == 6) {// Responde correctamente
			return 0;
		} else {// Timeout o error de comunicación
		System.out.println("No se ha podido encontrar un puerto de
	comunicación válido");
	return -1;
	}
```

Una vez establecida la comunicación solamente tendremos que enviar el comando indicado a la placa.

```
public boolean startReg() {
	try {
		output.write(0x64);
		return true;
	} catch (IOException e) {… }
}
public boolean stopReg() {
	try {
		output.write(0x65);
		return true;
	} catch (IOException e) {…}
}
```

En la capa superior, el paquete *logic*, definiremos la lógica en la clase *Negocio.java* para que la activación/desactivación del riego se produzca en función del contexto, es decir, si ya está activo que no se vuelva a enviar el pulso o si hay un riego automático programado que se avise al usuario para que fuerce o no el riego.

Y en la interfaz definiremos dos botones que permitan ejecutar las órdenes.

FOTO 6.2: INTERFAZ DE USUARIO DEL CASO DE USO ACTIVAR/DESACTIVAR RIEGO

6.3. Obtener temperatura

Para poder gestionar de una manera óptima los recursos agrarios la primera variable que se debe de estudiar es la temperatura.

6.3.1. Hardware

En el mercado tenemos multitud de sensores de temperatura, pero los que realmente nos interesan son los digitales, ya que no requieren ni de una calibración ni de un postprocesamiento de los datos para obtener la temperatura real. Estos termómetros son un poco más caros y difíciles de encontrar que los analógicos, pero con ellos obtenemos datos con una mayor precisión y de una manera más fácil.

Siguiendo las directrices de diseño del capítulo 5, nos hemos decantado por un termómetro digital que implementa un estándar de bajo nivel, el protocolo 1-Wire. Utilizaremos el modelo DS18B20 de la empresa Maxim. Este termómetro digital tiene una resolución de hasta 12 bits que le otorga una precisión de 0,125 ºC. Su margen de error es de ± 0.5 °C y es capaz de medir temperaturas entre – 55 ºC y 125 ºC.

FIGURA 6.7: TERMÓMETRO DIGITAL MAXIM DS18B20

Como muestran las siguientes figuras, el termómetro puede ser alimentado por una fuente externa (con voltajes entre 3 y 5.5V) o través del propio bus 1–Wire en el llamado modo parásito.

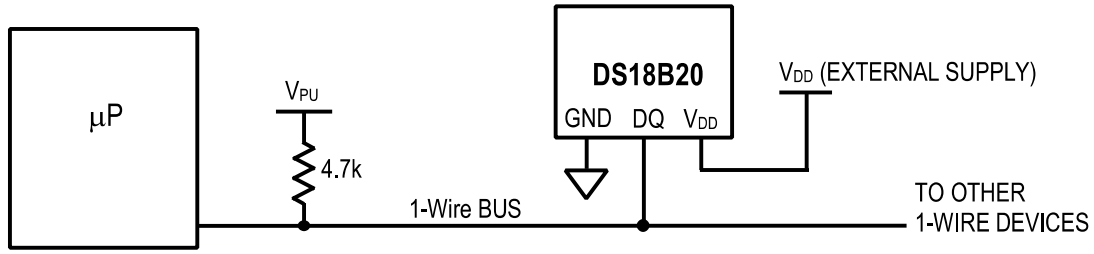


FIGURA 6.8: TERMÓMETRO DS18B20 ALIMENTADO CON UNA FUENTE EXTERNA

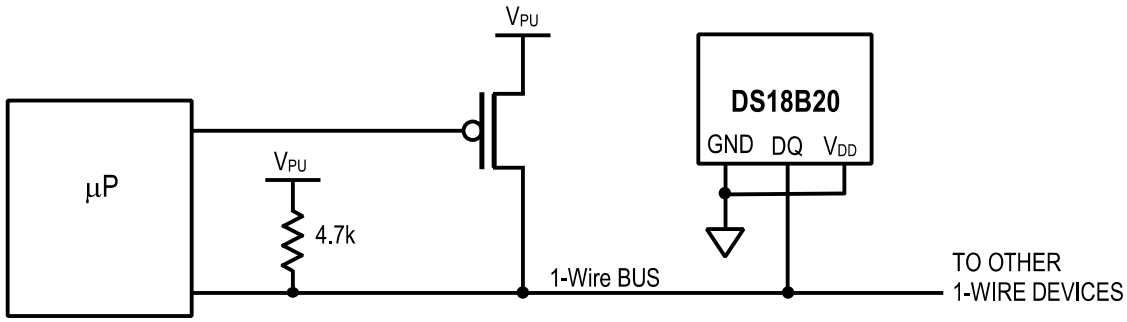


FIGURA 6.9: TERMÓMETRO DS18B20 ALIMENTADO EN MODO PARÁSITO

El DS18B20 es capaz de devolver la temperatura en una palabra de 12 bits en un tiempo máximo 750 ms. Sin embargo el modo parasito puede ocasionar que consultas demasiado consecutivas devuelvan valores inválidos, debido a los tiempos de carga del condensador interno. Por este motivo se ha decidido utilizar el modo de alimentación externa.

Además gracias a la utilización del protocolo 1-Wire tenemos la posibilidad de conectar a dicho bus tantos dispositivos 1-Wire como queramos, ya que este bus funciona con una arquitectura maestro-esclavo, donde el microcontrolador (que actúa como maestro) selecciona con que dispositivo 1-Wire quiere conectar (identificado con un número unívoco). Esto en la práctica se refleja en que podremos conectar tantos termómetros (y otros sensores) como queramos al mismo pin de datos 1-Wire.

El diagrama de conexiones de nuestra placa AlReg es el siguiente:

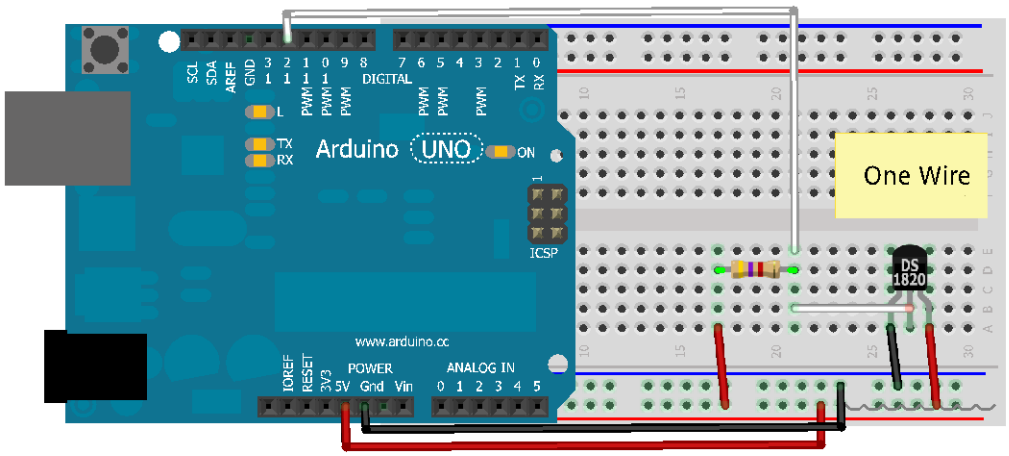


FIGURA 6.10: DIAGRAMA DE CONEXIONES DEL CASO DE USO OBTENER TEMPERATURA

6.3.2. Software

6.3.2.1. Alreg

Al ser el protocolo 1-Wire un protocolo complejo, requiere que a nivel de programación definamos diferentes comandos para seleccionar el dispositivo esclavo.

Hexadecimal	Decimal	ASCII	Descripción
0x6A	106	j	Contar sensores
0x6B	107	k	Resetear búsqueda
0x6C	108	l	Siguiente sensor
0x6D	109	m	Seleccionar cursor
0x6E	110	n	Obtener temperatura

TABLA 6.2: COMANDOS DEL CASO DE USO OBTENER TEMPERATURA

En el *sketch* Arduino utilizaremos la librería *OneWire.h* para contactar con los dispositivos 1-Wire. Con esta librería solo tendremos que indicar el pin del bus y a continuación enviar las órdenes de acuerdo con la especificación del dispositivo DS18B20. Luego incluiremos estos comandos en el bucle principal, añadiendo el código EOT al final de la comunicación:

```
#include <OneWire.h>
OneWire ds(12);
case 0x6A: // contarSensores: j 106 0x6A
    ds.reset_search();
    while(ds.search(addr))
      count++;
  [[…]]
```

```
case 0x6B: // resetBusqueda: k 107 0x6B
    ds.reset_search();
    break;
  case 0x6C: // siguienteSensor: l 108 0x6C
    if(ds.search(addr))
      for(int i=0;i<8;i++)
        Serial.write(addr[i]);
    break;
  case 0x6D: // seleccionarCursor: m 109 0x6D
    delay(40);
    contador=0;
    for (int i=0;i<8;i++){
      if(Serial.available()>0){
        sensor_id[i]=Serial.read();
        contador++; }}
    if(contador==8)
      if ( OneWire::crc8( sensor_id, 7) != sensor_id[7]) {// CRC no válido
        Serial.print(0,DEC);
        Serial.write(4);
        return;       }
      // Sensor válido
      Serial.print(1,DEC);
      Serial.write(4);
      ds.reset();
     ds.select(sensor_id);
    [[…]]
  case 0x6E: // Obtener Tª del sensor seleccionado: n 110 0x6E
    ds.write(0x44,1);    // Comando para empezar la conversión
    delay(1000);     // 750ms pueden ser suficientes para realizar la conexión
    present = ds.reset();
    ds.select(sensor_id);
    ds.write(0xBE);         // Leer Scratchpad
    for (int i = 0; i < 9; i++)    // Necesitamos 9 bytes
      data[i] = ds.read();
    LowByte = data[0];
    HighByte = data[1];
    TReading = (HighByte << 8) + LowByte;
    SignBit = TReading & 0x8000;  // Comprobamos el MSB
    if (SignBit) // negative
      TReading = (TReading ^ 0xffff) + 1; // 2's comp
    Tc_100 = (6 * TReading) + TReading / 4;    // Multiplicar por (100 * 0.0625)
    Whole = Tc_100 / 100;  //Separamos la parte entera de la fraccionaria
    Fract = Tc_100 % 100;
    Serial.print(Fract);
    [[…]]
Serial.write(4); //EndOfTransmission (ASCII)
```

Con este código podremos leer todos los sensores DS18B20 conectados al bus 1.-Wire.

6.3.2.2. RegAdmin

En el programa RegAdmin tendremos que añadir a la capa Arduino las funciones que permitan consultar todos estos comandos:

```
public int contarSensoresT() {
       output.write(0x6A);
       String res = leerArduino();
       return Integer.parseInt(res);
}
```

```
public byte[][] listarSensoresT() {
        this.contarSensoresT();
        this.resetearBusquedaT();
        this.sensores_t = new byte[this.n_sensores_t][8];
        for (int i = 0; i < this.n_sensores_t; i++) {
                output.write(0x6C);
                byte res[] = leerArduinoBytes();
                this.sensores_t[i] = res;
        }
        this.resetearBusquedaT();
        return this.sensores_t;
}
public Float obtenerTemperatura(byte[] sensor) {
        if (seleccionarSensorT(sensor) != 1)
                return null;
        else {
                output.write(0x6E);
                String res = leerArduino();
                Float res_f = Float.parseFloat(res);
                return res_f;
        }
}
```

Mientras que en la lógica de negocio manejaremos los sensores como arreglos de cadenas que contendrán el identificador, así seleccionaremos el sensor correspondiente con su identificador único:

```
private String[] sensores_t;
public Float obtenerTemperatura(String sensor) {
        for (int i = 0; i < sensores_t.length; i++)
                if (sensor.compareTo(sensores_t[i]) == 0)
                        break;
        if (i == sensores_t.length)// No se ha encontrado el sensor
                return null;
        else {// Los dos índices coinciden
                Float res = Arduino.obtenerTemperatura(sensores_t_raw[i]);
                return res;
        }
}
```

La interfaz la prepararemos también para que cargue automáticamente todos los sensores de temperatura, incluyendo un botón para actualizar los valores.

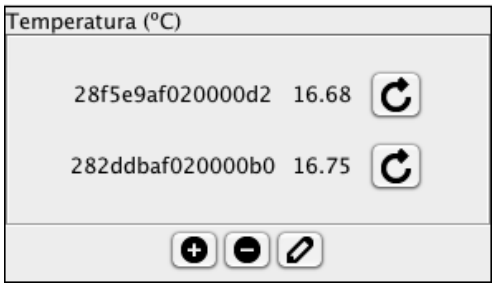


FOTO 6.3: INTERFAZ DE USUARIO DEL CASO DE USO OBTENER TEMPERATURA

6.4. Obtener humedad relativa del aire

La humedad es una variable de vital importancia en la gestión del agua, ya que indica la cantidad de vapor de agua presente en el aire. En el caso de uso que nos atañe nos interesa conocer la humedad relativa, es decir, el porcentaje de la totalidad de agua que podría contener el aire a una cierta temperatura.

6.4.1. Hardware

La mayoría de sensores de humedad comerciales que encontramos son analógicos, lo que quiere decir que necesitamos en primer lugar calibrar el sensor y en segundo realizar una conversión analógico-digital de los datos obtenidos. En general estos sensores son menos precisos que los digitales pero son más baratos, más fáciles de conseguir y existen para casi todas las medidas.

Para este caso de uso hemos elegido el sensor de humedad HH10D de la compañía Hope Microelectronics. Este humedímetro analógico implementa el protocolo I^2C con voltaje de entrada de 3.3V. Su consumo ronda los 150uA y en un rango entre -10ºC y 60 ºC es capaz de obtener la humedad relativa con un 3% de precisión. Una característica de este sensor es que incorpora un sensor de temperatura interno, por lo que no es necesario pasar la temperatura como dato de entrada.

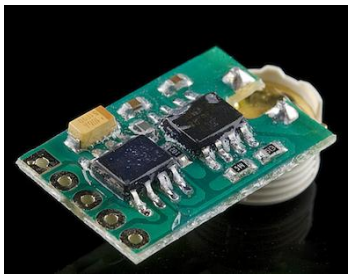


FOTO 6.4: SENSOR DE TEMPERATURA HH10D

Los dispositivos que cumplen con la especificación I^2C se comunican a través de dos buses compartidos: una línea de datos (SDA) y una de reloj (SDL). Igual que el protocolo 1-Wire responde a una arquitectura maestro-esclavo. Los diferentes dispositivos esclavos se identifican con un código familiar, lo que implica que si queremos conectar más de un sensor del mismo tipo deberemos optar por cambiar su identificador, utilizar un multiplexor para seleccionar el sensor o diseñar un circuito que encienda y apague los sensores.

Nuestro sensor tiene la peculiaridad de que utiliza estos puertos solo para recibir órdenes, enviando la información por una línea auxiliar (F_{out}) a través de frecuencias.

Para realizar las conexiones utilizaremos los pines SDA y SDL por defecto incluidos en la placa Arduino UNO (pines analógicos 4 y 5), más el pin digital 5 para leer el resultado:

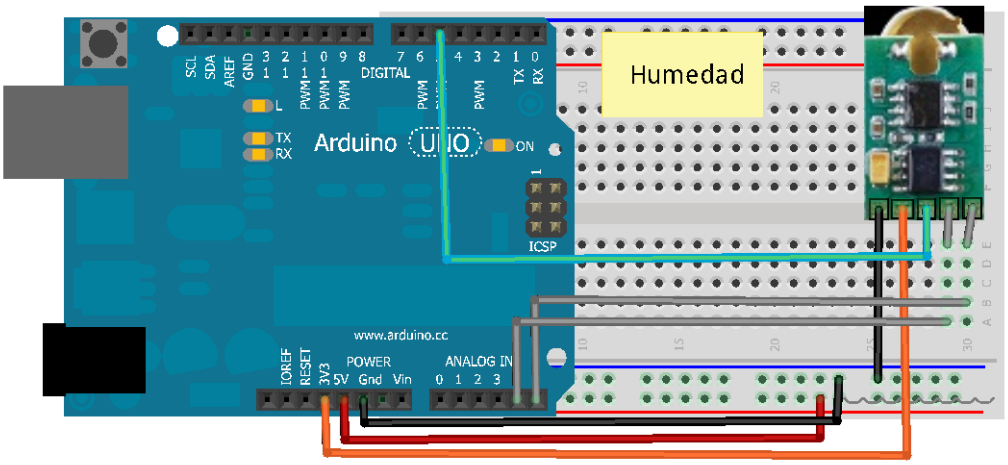


FOTO 6.5: DIAGRAMA DE CONEXIONES DEL CASO DE USO OBTENER HUMEDAD RELATIVA DEL AIRE

6.4.2. Software

6.4.2.1. Alreg

Dadas las características de nuestro sensor necesitaremos utilizar en el código Processing dos librerías: la librería *Wire.h* y la *FreqCounter.h.*

La primera de ellas es utilizada para comunicarse con los dispositivos a través del protocolo I^2C. Con ella establecemos la conexión entre el dispositivo maestro (Arduino) y el esclavo, enviando los comandos particulares de cada sensor. Además nuestro sensor requiere de una calibración previa.

```
#include <Wire.h>
Wire.begin();
void hh10dCalibration(){
        sens    =  i2cRead2bytes(81, 10); // Leemos sensibilidad de la EEPROM
        offset =  i2cRead2bytes(81, 12); // Leemos el offset
        // Las dos primera lecturas son erróneas
        hh10dReadHumidity();
        hh10dReadHumidity();
}
```

```
int i2cRead2bytes(int deviceaddress, byte address){
          Wire.beginTransmission(deviceaddress);
          Wire.write(address);
          Wire.endTransmission();
          Wire.requestFrom(deviceaddress, 2);
          int rv = 0;
          for (int c = 0; c < 2; c++ )
            if (Wire.available()) rv = rv * 256 + Wire.read();
          return rv;
}
```

Por otro lado como nuestro humedímetro devuelve el resultado en frecuencias, necesitaremos de una librería auxiliar que cuente el valor de los pulsos transformándolos a un valor numérico:

```
#include <FreqCounter.h>
float hh10dReadHumidity(){
  FreqCounter::f_comp= 8; // Establecemos la compensación a 12
  FreqCounter::start(1000); // Comenzamos a contar con gatetime de 1000ms
  while (FreqCounter::f_ready == 0) // Esperamos a que esté preparado
    freq=FreqCounter::f_freq; // Leemos el resultado
  float RH =  (offset-freq)*sens/4096;
  return RH;
}
```

Todo esto lo incorporaremos a nuestra API para ofrecer la siguiente orden:

Hexadecimal	Decimal	ASCII	Descripción
0x75	117	u	Obtener humedad HH10D

TABLA 6.3: COMANDOS DEL CASO DE USO OBTENER HUMEDAD RELATIVA DEL AIRE

6.4.2.2. RegAdmin

En la aplicación Java tendremos que crear las funciones de comunicación con la placa (capa Arduino):

```
public Float obtenerHumedadHH10D() {
       output.write(0x75);
       String res = leerArduino();
       Float res_f = Float.parseFloat(res);
       return res_f;
}
```

Y las funciones necesaria en la capa de la lógica y interfaz para ofrecer la humedad relativa:

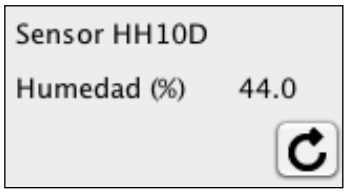


FOTO 6.6: INTERFAZ DE USUARIO DEL CASO DE USO OBTENER HUMEDAD RELATIVA DEL AIRE

6.5. Obtener presión atmosférica

La presión atmosférica es un parámetro muy útil para conocer las condiciones meteorológicas. Obteniendo el valor de la presión y conociendo la altura respecto al nivel del mar (derivando de ella la presión de referencia) podemos estimar si el tiempo es soleado (presión atmosférica alta), está nublado o está lloviendo (presión baja).

6.5.1. Hardware

Los sensores de presión tienen un coste superior a los sensores vistos hasta el momento. Esto es debido a que sus componentes son más complejos y requieren de otros sensores auxiliares incorporados.

En nuestro caso seguiremos las reglas de diseño y escogeremos un sensor que funcione sobre el bus I^2C, sea de pequeñas dimensiones y consuma poco.

El dispositivo elegido el sensor Bosch BMP086 ensamblado por SparkFun. Se caracteriza por funcionar a 3.3 V y trabajar en un rango de entre 300 Pa y 1100 hPa, con una precisión de 0.03 Pa. Viene de serie calibrado e incluye medición de la temperatura.

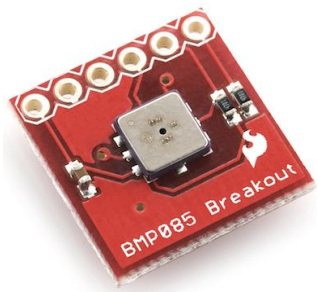


FOTO 6.7: SENSOR DE TEMPERATURA BMP085

El barómetro BMP085 cuenta con 4 modos de funcionamiento (ultra bajo consumo, bajo consumo, alta precisión y ultra alta precisión) y se comunica con el microcontrolador a través de los pines SDA y SCL del protocolo I^2C:

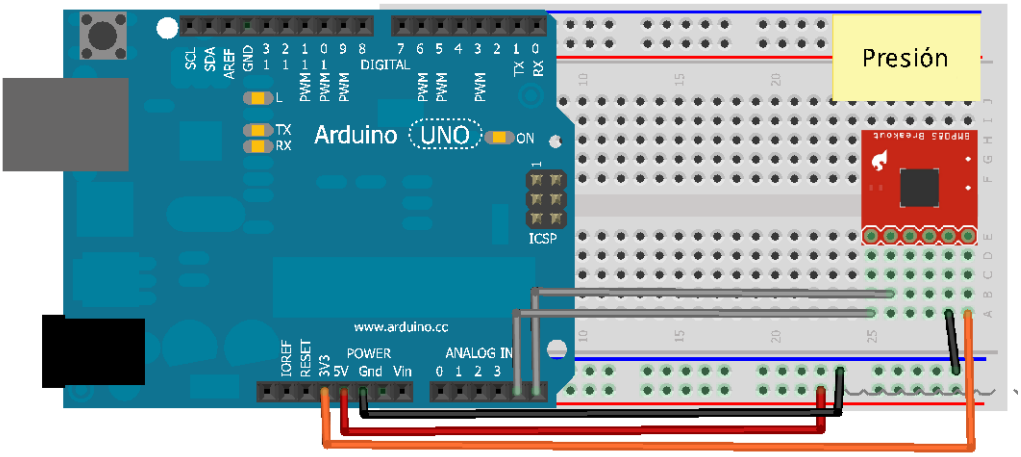


FIGURA 6.11: DIAGRAMA DE CONEXIONES DEL CASO DE USO OBTENER PRESIÓN ATMOSFÉRICA

6.5.2. Software

6.5.2.1. Alreg

En el código Arduino vamos a definir los siguientes comandos, que ofrecen las funcionalidades del sensor:

Hexadecimal	Decimal	ASCII	Descripción
0x70	112	p	Obtener temperatura BMP085
0x71	113	q	Obtener presión BMP085
0x72	114	r	Obtener altura BMP085
0x73	115	s	Obtener previsión del tiempo BMP085

TABLA 6.4: COMANDOS DEL CASO DE USO OBTENER PRESIÓN ATMOSFÉRICA

Para consultar los valores brutos de temperatura y de presión simplemente utilizaremos la librería *Wire* enviando las órdenes correspondientes:

```
unsigned int bmp085ReadUT(){
  unsigned int ut;
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF4);
  Wire.write(0x2E);
  Wire.endTransmission();
  delay(5); //Wait
  ut = bmp085ReadInt(0xF6);// Leemos dos bytes de los registros 0xF6 and 0xF7
  return ut;}

unsigned long bmp085ReadUP() {// Leemos el valor de presión descompensada
  unsigned char msb, lsb, xlsb;
  unsigned long up = 0;
  Wire.beginTransmission(BMP085_ADDRESS);
  Wire.write(0xF4);
  Wire.write(0x34 + (OSS<<6));
  Wire.endTransmission();
  delay(2 + (3<<OSS)); // Esperamos a la conversión
  Wire.beginTransmission(BMP085_ADDRESS);
```

```
  Wire.write(0xF4);
  Wire.write(0x34 + (OSS<<6));
  Wire.endTransmission();
  delay(2 + (3<<OSS)); // Esperamos a la conversión
  Wire.beginTransmission(BMP085_ADDRESS); // Leemos los registros 0xF6 (MSB), 0xF7
(LSB), and 0xF8 (XLSB)
  Wire.write(0xF6);
  Wire.endTransmission();
  Wire.requestFrom(BMP085_ADDRESS, 3);
  while(Wire.available() < 3)// Esperamos a que la información esté disponible
    ;
  msb = Wire.read();
  lsb = Wire.read();
  xlsb = Wire.read();
  up = (((unsigned long) msb << 16) | ((unsigned long) lsb << 8) | (unsigned long)
xlsb) >> (8-OSS);
  return up;
}
```

Al venir el sensor calibrado de fábrica necesitamos normalizar los valores obtenidos con las constantes de calibración:

```
case 0x70: // Obtener Tª del sensor de presión I2C: p 112 0x70
       temperature = bmp085GetTemperature(bmp085ReadUT());
       Whole = temperature / 10;
       Fract = temperature % 10;
       [[...]]
short bmp085GetTemperature(unsigned int ut) {
       long x1, x2;
       x1 = (((long)ut - (long)ac6)*(long)ac5) >> 15;
       x2 = ((long)mc << 11)/(x1 + md);
       b5 = x1 + x2;
       return ((b5 + 8)>>4);
}
void bmp085Calibration(){
         ac1 = bmp085ReadInt(0xAA);
         ac2 = bmp085ReadInt(0xAC);
         [[..]]
}
```

Para obtener la presión, además de normalizar, precisamos utilizar la temperatura para calcular la relación que define la presión relativa:

```
case 0x71: // Obtener Presión del sensor I2C: q 113 0x71
       temperature = bmp085GetTemperature(bmp085ReadUT());
       pressure = bmp085GetPressure(bmp085ReadUP());
       [[..]]

long bmp085GetPressure(unsigned long up){
       long x1, x2, x3, b3, b6, p;
       unsigned long b4, b7;
       b6 = b5 - 4000;
       x1 = (b2 * (b6 * b6)>>12)>>11; // b5 from bmp085GetTemperature(...)
       x2 = (ac2 * b6)>>11;
       x3 = x1 + x2;
       b3 = (((((long)ac1)*4 + x3)<<OSS) + 2)>>2;
       x1 = (ac3 * b6)>>13;
       x2 = (b1 * ((b6 * b6)>>12))>>16;
       x3 = ((x1 + x2) + 2)>>2;
       b4 = (ac4 * (unsigned long)(x3 + 32768))>>15;
       b7 = ((unsigned long)(up - b3) * (50000>>OSS));
       if (b7 < 0x80000000)
              p = (b7<<1)/b4;
       else
```

```
                p = (b7/b4)<<1;
        x1 = (p>>8) * (p>>8);
        x1 = (x1 * 3038)>>16;
        x2 = (-7357 * p)>>16;
        p += (x1 + x2 + 3791)>>4;
        return p;
}
```

Ya por último, conociendo la altura sobre el nivel del mar en la que se encuentra el sensor podemos calcular la presión de referencia y en función de ella calcular el estado tiempo.

```
float currentAltitude = 34; // Altitud en metros de Alcácer
float ePressure = p0 * pow((1-currentAltitude/44330), 5.255); //100917,669916678 =
presión esperada (en Pa) para la altitud
[[..]]
case 0x73: // Obtener estimación del tiempo del sensor I2C: s 115 0x73
    temperature = bmp085GetTemperature(bmp085ReadUT());
    pressure = bmp085GetPressure(bmp085ReadUP());
    weatherDiff = pressure - ePressure;
    if(weatherDiff > 250)
      Serial.print(Sunny);
    else if ((weatherDiff <= 250) || (weatherDiff >= -250))
      Serial.print(Partly_Cloudy);
    else if (weatherDiff > -250)
      Serial.print(Rain);
    [[...]]
```

6.5.2.2. RegAdmin

Dispuesta toda la lógica del barómetro en el microcontrolador Arduino, desde la aplicación RegAdmin solo queda promocionar el resultado desde la clase Arduino a la interfaz de usuario:

```
[[Arduino]]
public Long obtenerPresionBMP085() {
        output.write(0x71);
        String res = leerArduino();
        long res_l = Long.parseLong(res);
        return res_l;
        }
[[...]]

[[Lógica de negocio]]
public Long obtenerPresionBMP085() {
        return Arduino.obtenerPresionBMP085();
}
[[...]]
```

Mostrando el resultado de la siguiente manera:

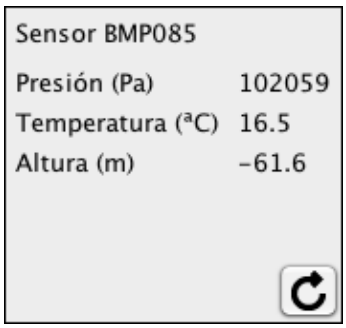


FOTO 6.8: INTERFAZ DE USUARIO DEL CASO DE USO OBTENER PRESIÓN ATMOSFÉRICA

6.6. Obtener humedad del suelo

El último sensor de la fase RegAdmin que va a tener cabida en la tesis es el sensor de humedad del suelo. Con este dispositivo vamos a poder conocer el grado de humedad existente en la tierra, obteniendo así un indicador directo de las necesidades de riego.

6.6.1. Hardware

Los sensores de humedad del suelo son un producto especializado, de difícil acceso, dependientes de plataforma y caros. Por estos motivos hemos decidido implementar nuestro propio sensor de humedad de la tierra basándonos en el conocimiento difundido por las comunidades DIY (*Do It Yourself*).

El método resistivo que vamos a utilizar consiste en enterrar dos sondas en la tierra, aplicar una diferencia de potencial (mediante los pines digitales 8 y 9) y analizar el voltaje resultante (capturado por el pin analógico 0). De esta manera si nos encontramos en un medio acuoso el valor obtenido por el pin analógico será alto (hasta 1024) debido a la alta humedad de la tierra, mientras que si no existe prácticamente humedad en el medio su valor será cercano a 0.

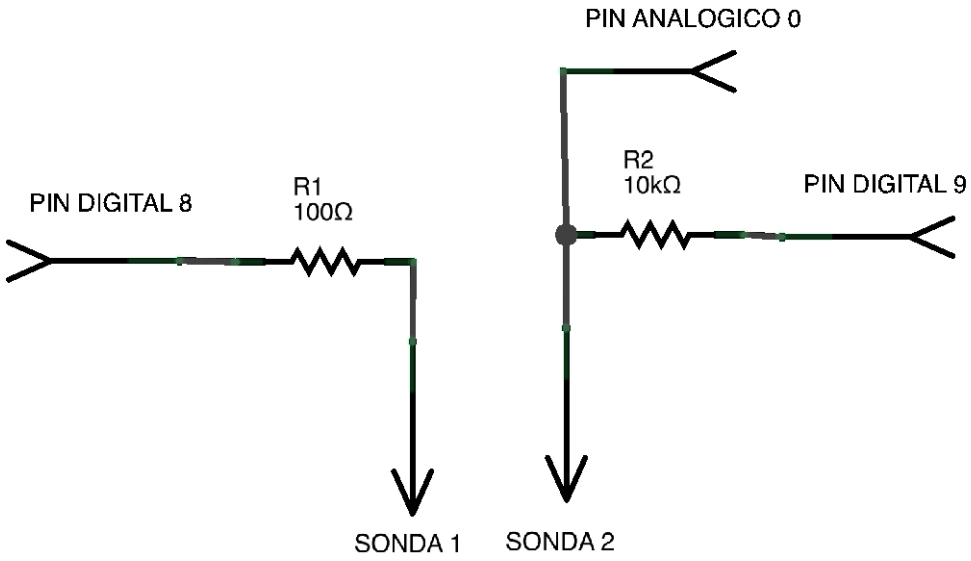


FIGURA 6.12: DIAGRAMA DE CONEXIONES DEL CASO DE USO OBTENER HUMEDAD DEL SUELO

Pero este análisis puede ser demasiado simplista debido a la corrosión que sufren las sondas por la electrólisis (proceso de separación de materiales que puede producir desgaste y resultados demasiado constantes), a las características de la tierra y a las interferencias exteriores. Por estos motivos nuestro sensor alternará el sentido de la corriente, utilizará sondas galvanizadas recubiertas con material aislante y calibrará las resistencias en función de la dureza de la tierra (10 KOhm para superficies arenosas).

FOTO 6.9: SENSOR RESISTIVO DE HUMEDAD DEL SUELO

6.6.2. Software

Una vez construido el sensor, el trabajo restante consiste es implementar el método en el código Arduino .

6.6.2.1. Alreg

Para obtener unos datos fiables necesitamos analizar los resultados variando el sentido de la corriente. Esto lo lograremos programando en la placa Arduino los pulsos de las salidas digitales de manera alternativa. Seguidamente leeremos los valores de entrada y calcularemos la media aritmética:

```
#define voltageFlipPin1 8
#define voltageFlipPin2 9
case 0x76: // Obtener humedad del suelo 0x76 118 v
        setSensorPolarity(true);
        delay(flipTimer);
        int val1 = analogRead(sensorPin);
        delay(flipTimer);
        setSensorPolarity(false);
        delay(flipTimer);
        int val2 = 1023 - analogRead(sensorPin); // Invertimos la lectura
        int avg = (val1 + val2) / 2;// Calculamos el valor de humedad en % truncando
        avg = avg * 0.09765625; // En porcentaje (*100/1024)

void setSensorPolarity(boolean flip){
        if(flip){
                digitalWrite(voltageFlipPin1, HIGH);
                digitalWrite(voltageFlipPin2, LOW);
        }else{
                digitalWrite(voltageFlipPin1, LOW);
                digitalWrite(voltageFlipPin2, HIGH);
        }
}
```

Incluyendo este código en el bucle principal podremos definir el nuevo comando:

Hexadecimal	Decimal	ASCII	Descripción
0x76	118	v	Obtener humedad del suelo

TABLA 6.5: COMANDO DEL CASO DE USO OBTENER HUMEDAD DEL SUELO

6.6.2.2. RegAdmin

Al igual que en los sensores de humedad relativa y de presión, como la mayor carga algorítmica se ha realizado en la parte del microcontrolador, en el programa RegAdmin solamente tendremos que añadir las llamadas a la placa:

```
[[Arduino]]
public int obtenerHumedadSuelo() {
        output.write(0x76);
        String res = leerArduino();
        Integer res_i = Integer.parseInt(res);
        return res_i; }

[[Lógica de negocio]]
public int obtenerHumedadSuelo() {
        return ino.obtenerHumedadSuelo();}
```

Y definir una interfaz de usuario:

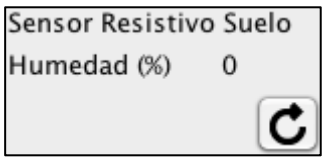


FOTO 6.10: INTERFAZ DE USUARIO DEL CASO DE USO OBTENER HUMEDAD DEL SUELO

6.7. Programar riego

Tras la implementación de este caso de uso el agricultor podrá planificar el riego mediante la inserción de eventos en un calendario. Así cuando nuestro dispositivo detecte que se encuentra dentro del intervalo de un evento activará el riego, y cuando este termine o se elimine la cita el riego se detendrá.

Pero debido al desarrollo evolutivo, en esta primera fases no estamos incluyendo un objetivo básico del sistema: la movilidad. Por este motivo se ha decidido utilizar como herramienta de gestión de riego un servicio de calendario en la nube. De las diferentes alternativas hemos escogido Google Calendar. Con este servicio obtendremos un método de entrada universal, accesible desde cualquier lugar, multiplataforma (clientes para SO móviles como Android o iOS y para PC como Windows, OS X y Linux), gratuito y con facilidades para el desarrollo.

La incorporación de este caso de uso solamente tendrá componente software, ya que utilizando una librería externa consultará el calendario y realizará las llamadas de bajo nivel para activar o desactivar el riego.

6.7.1. Software

En la parte software solo estará implicado el programa RegAdmin.

6.7.1.1. RegAdmin

El acceso a los datos del calendario será llevado a cabo por la librería *google-api-java-client*. Esta librería, utilizando la API independiente del lenguaje de Google, realiza las llamadas JSON a los servicios de Google. Además para la identificación del usuario ofrece un método a través del navegador que evita la introducción de datos y contraseñas en el programa cliente. Para más información acudir al anexo 9.8.7.

Así pues en la lógica de negocio lo que en primer lugar se realizará será la conexión con el calendario a través de la URL de autorización de Google:

```
import com.google.api.client.*;
private int abrirCalendario() {
        HttpTransport httpTransport = new NetHttpTransport();
        JacksonFactory jsonFactory = new JacksonFactory();
        String clientId = "797624020242.apps.googleusercontent.com";
        String clientSecret = "2YlbP9Pt90AUF029Dj4vCHiE";
        String redirectUrl = "urn:ietf:wg:oauth:2.0:oob";
        String scope = "https://www.googleapis.com/auth/calendar";
        // Paso 1: Autorización
        String authorizationUrl = new GoogleAuthorizationRequestUrl(
                        clientId, redirectUrl, scope).build();
        // Redirección para usar la url de autorización
        System.out.println("Vaya al siguiente enlace en su navegador:");
        System.out.println(authorizationUrl);
        // Abrimos directamente el navegador
        ImageIcon icono = new ImageIcon(Negocio.class.getResource("/imágenes
/Naranjito/Naranjito 64.png"));
        int res=JOptionPane.showConfirmDialog(null, "¿Desea iniciar sesión en Google
Calendar?", "Iniciar sesión", JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE, icono);
        if(res == JOptionPane.YES_OPTION)
        Desktop.getDesktop().browse(new URI(authorizationUrl));
        else //NO_OPTION
                return -4;
        // Lectura del código a través de un JOptionPane
        String code = (String) JOptionPane.showInputDialog(null,
                        "Introduzca el código de autorización:",
                        "Autorización para acceder a Google Calendar",
                        JOptionPane.QUESTION_MESSAGE, icono, null, null);
        if (code == null) // Se ha pulsado la Cancelar
                return -4;
        // Paso 2: Intercambio de códigos
        AccessTokenResponse response = new GoogleAuthorizationCodeGrant(
                        httpTransport, jsonFactory, clientId, clientSecret, code,
                        redirectUrl).execute();
        GoogleAccessProtectedResource accessProtectedResource = new
GoogleAccessProtectedResource(
                        response.accessToken, httpTransport, jsonFactory, clientId,
                        clientSecret, response.refreshToken);
        googleCalendar = com.google.api.services.calendar.Calendar
                        .builder(httpTransport, jsonFactory)
                        .setApplicationName("SysReg")
                        .setHttpRequestInitializer(accessProtectedResource).build();
        return 0;
        }
}
```

Este método nos facilitará una URL que, introduciéndola en el navegador, nos llevará a una pantalla de *login* en la que podremos permitir el acceso a nuestro calendario. Si el acceso es correcto nos devolverá una clave pública que copiaremos y

pegaremos en RegAdmin, estableciendo así la conexión de nuestro cliente con el calendario.

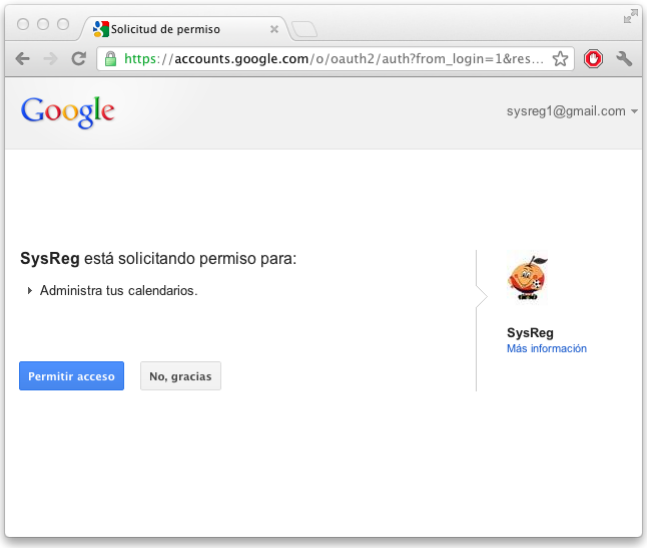


FOTO 6.11: SOLICITUD DE PERMISOS A LA APLICACIÓN REGADMIN

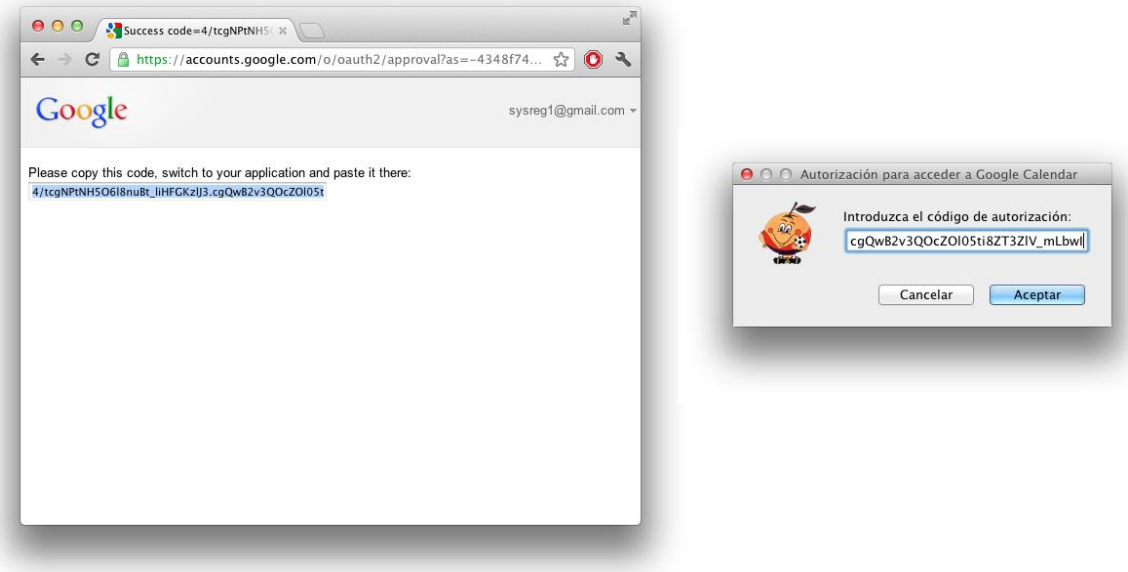


FOTO 6.12: CLAVE DE AUTORIZACIÓN

Una vez realizada la conexión, realizaremos una consulta de los eventos activos en las próximas 48 horas y los insertaremos en nuestra clase *Evento*, para posteriormente ordenarlos y colorearlos según el estado:

```
public List<Evento> cargarCalendario() {
	com.google.api.services.calendar.Calendar.Events.List consulta =
googleCalendar
			.events().list("primary");
	DateTime dateInicio = obtenerHoy();  // DateTime.parseRfc3339("2011-11-
23T00:00:00Z");
	DateTime dateFin = obtenerPasadoManyana();
	String inicio = dateInicio.toString();
	String fin = dateFin.toString();
	consulta.setTimeMin(inicio);
	consulta.setTimeMax(fin);
	// Para que nos retorne los eventos repetidos como eventos simples
	consulta.setSingleEvents(true);
	Events events = consulta.execute();
	sortedEvents = new ArrayList<Evento>();
	while (true) {
		for (Event event : events.getItems())
			if (event.getStatus().equals("confirmed")) {
				Evento e = new Evento(event.getSummary(),
					event.getDescription(), event.getStart()
					.getDateTime(), event.getEnd()
					.getDateTime(), event.getId());
				e.setLugar(event.getLocation());
				sortedEvents.add(e);
			}
		String pageToken = events.getNextPageToken();
		if (pageToken != null && !pageToken.isEmpty()) {
			events = googleCalendar.events().list("primary")
					.setPageToken(pageToken).execute();
		} else
			break;
	}
	// Ordenamos y coloreamos (según el estado) los eventos
	Collections.sort(sortedEvents);
	for (int i = 0; i < sortedEvents.size(); i++) {
		Evento e = sortedEvents.get(i);
		DateTime now = new DateTime(new Date(),
				TimeZone.getTimeZone("Europe/Madrid"));
		e.colorear(now);
	}
}
```

Por último simplemente nos quedará comprobar si debemos activar o desactivar el riego en función los eventos programados:

```
public List<Evento> comprobarRiego() {
	boolean activo = false;
	if (sortedEvents != null) {
		for (int i = 0; i < sortedEvents.size(); i++) {
			Evento e = sortedEvents.get(i);
			DateTime now = new DateTime(new Date(),
					TimeZone.getTimeZone("Europe/Madrid"));
			e.colorear(now);
			if (e.getEstado() == State.VERDE)
				activo = true;}
		if (activo)
			iniciarRiego();
		else
			pararRiego();
	}
	return sortedEvents;
}
```

Por otro lado en la parte de la interfaz de usuario (capa *iu)* nos encargaremos de:

- Mostrar los eventos del periodo establecido coloreándolos en función del estado.

FOTO 6.13: INTERFAZ DE USUARIO DEL CASO DE USO PROGRAMAR RIEGO (EVENTOS)

- Proveer un enlace a la edición del calendario: Desde el programa RegAdmin no vamos a editar el calendario de riegos, simplemente ofreceremos la posibilidad de, pulsando sobre el listado, acceder al editor web del calendario.

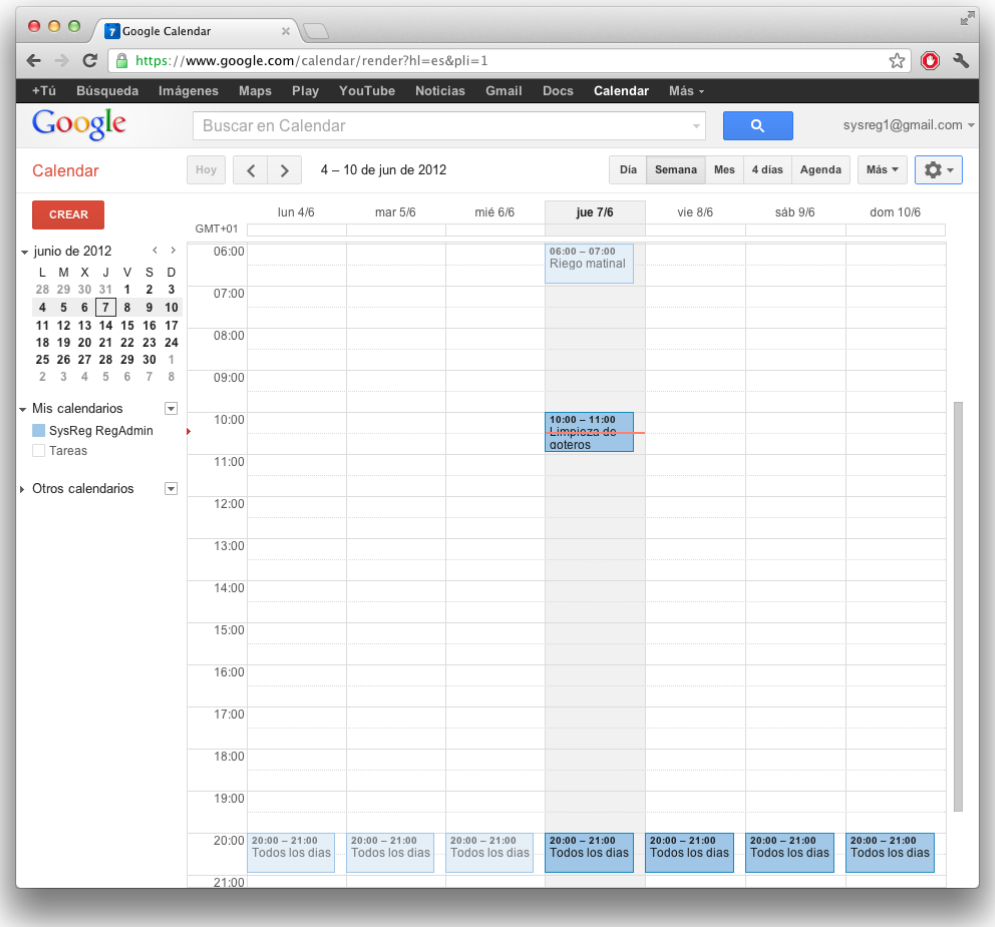


FOTO 6.14: GOOGLE CALENDAR

- Ejecución del hilo de comprobación de riego: Incluiremos el código necesario para ejecutar en segundo plano un proceso que compruebe cada minuto si debemos activar o desactivar un riego (función *comprobarRiego()*) y cada cinco descargue de nuevo los eventos del calendario (función *cargarCalendario()*).

```
private class RelojRiego extends Thread {
	private boolean activo = true;
	public RelojRiego() {
		this.setName("Hilo de comprobación del riego");
	}
	public void detener() {
		this.activo = false;
		if (!this.isInterrupted())
			this.interrupt();
	}
	public void run() {
		try {
			System.out.println("Inicio hilo actualizar reloj");
			int i = 0;
			while (activo) {
				Thread.sleep(60000);
				if (i < 5)// Cada 5 minutos actualizamos el calendario,
					i++;
				else {
					actualizarCalendario();
					i = 0;
				}
				System.out.println("Iteración hilo comprobar riego");
				eventos = logica.comprobarRiego();
				pintarEventos();
				// Actualizar la hora
				DecimalFormat entero = new DecimalFormat("00");
				Calendar now = Calendar.getInstance();
				l_hora.setText("Hora: "
					+
entero.format(now.get(Calendar.HOUR_OF_DAY))
					+ ":" +
entero.format(now.get(Calendar.MINUTE)));
			}
			System.out.println("Fin hilo actualizar reloj");
		} catch (InterruptedException e) {
			System.out.println("Sincronización desactivada (hilo reloj)");
		}
	}
}
```

- Activación y desactivación de la sincronización: Mediante un *checkbox* y un botón de actualizar permitiremos al usuario gestionar la conexión con el calendario. Además la gestión del riego manual interrumpirá automáticamente la sincronización, es decir, cuando pulsemos manualmente activar/desactivar riego cancelaremos la sincronización del riego para evitar conflictos.

FOTO 6.15: INTERFAZ DE USUARIO DEL CASO DE USO PROGRAMAR RIEGO (SINCRONIZACIÓN)

6.8. Establecer alarmas

La idea de este caso de uso es la de ofrecer un modo sin conexión para gestionar el riego. Con las funcionalidades implementadas hasta el momento podemos considerar Sysreg un sistema funcional para la optimización de recursos, pero hemos establecido una dependencia entre el hardware y el software de manera que uno no tiene sentido sin el otro.

Debido a esto hemos definido este caso de uso que pretende ofrecer al agricultor una forma de programar riegos, ya sea directamente en la placa o a través del aplicativo RegAdmin. Además nos aprovecharemos del caso de uso anterior permitiendo transformar eventos de calendario en alarmas en la placa.

6.8.1. Software

6.8.1.1. Alreg

El objetivo es dejar la gestión de alarmas en la parte del microcontrolador. Así conseguiremos que la implementación de las alarmas sea más precisa, consuma menos recursos y sea más independiente.

Esta implementación se llevará a cabo en el código Processing de la placa Arduino. Utilizando las librerías *Time.h* y *TimeAlarm.h* (detalladas en el capítulo 10) definiremos una serie de órdenes que serán implementadas a bajo nivel con interrupciones.

Hexadecimal	Decimal	ASCII	Descripción
0x41	65	A	Establecer hora
0x42	66	B	Establecer alarma de encendido
0x43	67	C	Establecer alarma de apagado
0x44	68	D	Establecer alarma repetitiva de encendido
0x45	69	E	Establecer alarma repetitiva de apagado
0x46	70	F	Eliminar alarma
0x47	71	G	Eliminar todas las alarmas

TABLA 6.6: COMANDOS DEL CASO DE USO ESTABLECER ALARMAS

En primer paso obligatorio para definir alarmas es establecer la hora actual. La librería *Time* ofrece una serie de funciones que periten sincronizar un reloj en la placa con la hora que le pasamos por parámetro en formato UNIX (segundos desde el 1 de Enero de 1970):

```
#include <Time.h>
case 0x41: // establecerHora: A 65 0x41¡
        delay(40);
        pctime = 0;
        n_digitos = 0; // Recogemos los 10 siguientes caracteres ASCII que será la
hora en tiempo UNIX
        for (int cont=0;cont<10;cont++){
                if(Serial.available()>0){
                        digito = Serial.read();
                        if( digito >= '0' && digito <= '9'){
                                pctime = (10 * pctime) + (digito - '0') ; //
convertimos los dígitos a números
                                n_digitos ++;
        }       }       }
        if(n_digitos == 10){
                setTime(pctime);   // Sincronizamos el reloj Arduino
                adjustTime(3600); // Tiempo peninsular (GMT +1)
                if(timeStatus() == timeSet){
                        Serial.print(1,DEC); //return 1
                }
        }
[[...]]
```

Para la definición de alarmas hemos optado por diferenciar entre alarmas puntuales y alarmas repetidas periódicamente. Las alarmas puntuales se implementarán como temporizadores, es decir, definiremos los segundos restantes hasta ejecutar la orden mediante la función *timerOnce(segundos_restante, función_a_ejecutar)* :

```
#include <TimeAlarms.h>
case  0x42: // establecerAlarmaON: B 66 0x42
        delay(40);
        alarmtime = 0;
        n_digitos = 0;
        for (int cont=0;cont<10;cont++){
                if(Serial.available()>0){
                        digito = Serial.read();
                        if( digito >= '0' && digito <= '9'){
                                alarmtime = (10 * alarmtime) + (digito - '0') ; //
convertimos los dígitos a números
                                n_digitos ++;
        }       }       }
        alarmtime += 3600;// Le sumamos 1 hora para estar en la franja horaria GMT +1
        if(n_digitos == 10 && timeStatus() == timeSet  && alarmtime > now()){
//Condiciones
                AlarmID_t a1 = Alarm.timerOnce(alarmtime - now(), alarmaOn);
                if(a1 == dtINVALID_ALARM_ID)
                        Serial.print(-1,DEC);
                Serial.print(a1,DEC);
        }
[[...]]
```

Donde *alarmaOn* se corresponde con el código del caso de uso activar riego. El código de *establecerAlarmaOFF* será igual pero cambiando la función *alarmaOn* por *alarmaOff* (con el código del caso de uso desactivar alarma).

El otro tipo de alarmas son las repetitivas. Tal y como vimos en el diagrama de clases, estas alarma se repiten con un cierto periodo cada día, semana, mes o año. En nuestra implementación, por simplificación, las hemos definido todas diarias:

```
case  0x44: // establecerAlarmaRepOn: D 68 0x44
       delay(40);
       horas = 0;
       minutos = 0;
       segundos = 0;
       n_digitos = 0;
       for (int cont=0;cont<6;cont++){ // Leemos los datos
          if(Serial.available()>0){
               digito = Serial.read();
               if( digito >= '0' && digito <= '9'){
                      if( cont < 2) // hora
                             horas = (10 * horas) + (digito - '0') ;
                      else if(cont < 4)
                             minutos = (10 * minutos) + (digito - '0') ;
                      else if(cont < 6)
                             segundos = (10 * segundos) + (digito - '0') ;
                      n_digitos ++;
               }
          }
       }
       if(n_digitos == 6 && timeStatus() == timeSet && horas < 24 && minutos < 60 &&
segundos < 60){ // Condiciones para que se pueda establecer la alarma
               AlarmID_t a1 =  Alarm.alarmRepeat(horas, minutos, segundos, alarmaOn);
               if(a1 == dtINVALID_ALARM_ID)
                      Serial.print(-1,DEC);
               Serial.print(a1,DEC);
       }
[[...]]
```

Como hemos visto en el código, al crear una alarma nos retorna un identificador único. Con este identificador podremos eliminar una alarma en concreto:

```
case  0x46: // eliminarAlarma(xxx): F 70 0x46
       delay(40);
       alarma_id = 0;
       n_digitos = 0;
       for (int cont=0;cont<3;cont++){
               if(Serial.available()>0){
                      digito = Serial.read();
                      if( digito >= '0' && digito <= '9'){
                             alarma_id = (10 * alarma_id) + (digito - '0'); //
convertimos los dígitos a números
                             n_digitos ++;
       }      }      }
       if(n_digitos > 0  && alarma_id >= 0 && alarma_id < 255){ //Condiciones
               Alarm.free(alarma_id); //Este método deshabilita y libera el id para
que se pueda utilizar
               Serial.print(1,DEC); //return 1
[[...]]
```

También dispondremos de orden para eliminar la totalidad de las alarmas, aunque por motivos de implementación de la librería, cuando establezcamos la hora se borrarán todas las alarmas.

```
case  0x47: // eliminarAlarmas: G 71 0x47
       for (int i=0;i<dtNBR_ALARMS;i++){
              Alarm.free(i);
       }
       break;
```

6.8.1.2. RegAdmin

Una vez en definidos todos los comandos en el código Arduino, en la aplicación multiplataforma RegAdmin podremos establecer las alarmas desde una ventana propia (clase interfaz):

FOTO 6.16: INTERFAZ DE USUARIO DEL CASO DE USO ESTABLECER ALARMAS

Cuando una vez definidas las alarmas pulsemos el botón subir, llamaremos a la función *alarmasAPlaca* (en la capa de la lógica de negocio). Esta función establece la hora, elimina todas las alarmas anteriores, transforma las fechas al formato requerido y, a través de las funciones de comunicación ofertadas por capa Arduino, propaga los comandos a nuestra placa:

```
public boolean AlamarmasAPlaca(List<Alarma> alarmas) {
       if (!Arduino.eliminarAlarmas())
              return false;
       if (!Arduino.establecerHora(null))// Hora actual
              return false;
       for (Alarma a : alarmas) {
          if (a instanceof AlarmaRepetitiva) {
              AlarmaRepetitiva arep = (AlarmaRepetitiva) a;
```

```
                Calendar fecha = new GregorianCalendar(
                        TimeZone.getTimeZone(("Europe/Madrid")));
                fecha.setTimeInMillis(a.getFecha().getValue());

                if (arep.getModo() == Alarma.Modo.ON) {
                        Arduino.establecerAlarmaRepOn(fecha.get(Calendar.HOUR_OF_DAY),
                                        fecha.get(Calendar.MINUTE),
                                        fecha.get(Calendar.SECOND));
                } else // Si no es de encendido es de apagado
                      Arduino.establecerAlarmaRepOff(fecha.get(Calendar.HOUR_OF_DAY),
                           fecha.get(Calendar.MINUTE),fecha.get(Calendar.SECOND));
        } else  // Si es una alarma puntual
                if (a.getModo() == Alarma.Modo.ON)
                        Arduino.establecerAlarmaOn(a.getFecha().getValue() / 1000);
                else
                        Arduino.establecerAlarmaOff(a.getFecha().getValue() / 1000);
                }
        }
}
```

La otra gran funcionalidad que incluye este caso de uso es la posibilidad de migrar los eventos del calendario a alarmas internas de la placa. De esta forma podremos desconectar la placa del PC y mantener la programación de riego. Esta migración requiere de una transformación (realizada en la capa negocio) de los objetos de la clase *Evento* a la clase *Alarma* y su especialización *Alarma Repetitiva:*

```
public List<Alarma> CalendarioAAlarma(DateTime inicio, DateTime fin) {
        if (googleCalendar == null) //Si no se ha establecido conexión lo intentamos
                if (abrirCalendario() != 0)
                        return null;
        // Leemos todos los eventos en el periodo indicado
        com.google.api.services.calendar.Calendar.Events.List consulta =
                googleCalendar.events().list("primary");
        consulta.setTimeMin(inicio.toString());
        consulta.setTimeMax(fin.toString());
        Events events = consulta.execute();
        List<Alarma> alarmas = new ArrayList<Alarma>();
        while (events.getItems() != null) {
           for (Event event : events.getItems()) {
              if (event.getStatus().equals("confirmed")) {
                if (event.getRecurrence() == null) { // Eventos simples
                        Alarma on = new Alarma(event.getStart(),
                                .getDateTime(), event.getId(),Alarma.Modo.ON);
                        Alarma off = new Alarma(event.getEnd().getDateTime(),
                                event.getId(),Alarma.Modo.OFF);
                        alarmas.add(on);
                        alarmas.add(off);
                } else { // Eventos repetidos
                        AlarmaRepetitiva on = new AlarmaRepetitiva(event.getStart().
                                getDateTime(),event.getId(),Alarma.Modo.ON,
                                AlarmaRepetitiva.Tipo.DIARIA,AlarmaRepetitiva.Periodo.
                                TODOS);
                        AlarmaRepetitiva off = new AlarmaRepetitiva(event.getEnd()
                                .getDateTime(), event.getId(),Alarma.Modo.OFF,
                                AlarmaRepetitiva.Tipo.DIARIA, AlarmaRepetitiva.Periodo.
                                TODOS);
                        alarmas.add(on);
                        alarmas.add(off);
        }       }       } // Si hay más resultados terminamos
           String pageToken = events.getNextPageToken();
           if (pageToken != null && !pageToken.isEmpty()) {
                events = googleCalendar.events().list("primary")
                        .setPageToken(pageToken).execute();
           }else
```

```
        }       }       } // Si hay más resultados terminamos
           String pageToken = events.getNextPageToken();
           if (pageToken != null && !pageToken.isEmpty()) {
                events = googleCalendar.events().list("primary")
                              .setPageToken(pageToken).execute();
           }else
              break; }
        return alarmas; }
```

Esta función se activará desde la pantalla anterior de creación de alarmas y mostrará los resultados como alarmas para evaluar el resultado de la migración y posteriormente subir los cambios (*AlarmasAPlaca()*)

6.9. Otras características

A continuación se presentan diversas funcionalidades que, aunque no tienen el peso especifico para desarrollar un caso de uso propio, han sido incluidas en la primera unidad funcional del sistema Sysreg.

6.9.1. Evitar reseteo automático

Al estar la plataforma Arduino enfocada al prototipado, cada vez que conectamos la placa a un computador se produce un reseteo para cargar el *bootloader* y, si el usuario lo desea, subir un *sketch* compilado.

Pero este comportamiento no es el deseado en una placa preparada para producción, ya que cada vez que conectemos la placa por USB el *sketch* finalizará y empezará de nuevo por la función *setup()*. Esto provocará además la perdida de todos las variables internas, siendo en nuestro caso especialmente grave, por que se perderán las alarmas, las calibraciones y el estado del riego.

Por este motivo en todos los modelos Arduino disponemos de unos pines etiquetados como *RESET-EN* unidos por una pequeña pista. Si queremos deshabilitar el auto-reseteo deberemos cortar dicha pista, siendo posible restablecer esta conexión con una pequeña soldadura.

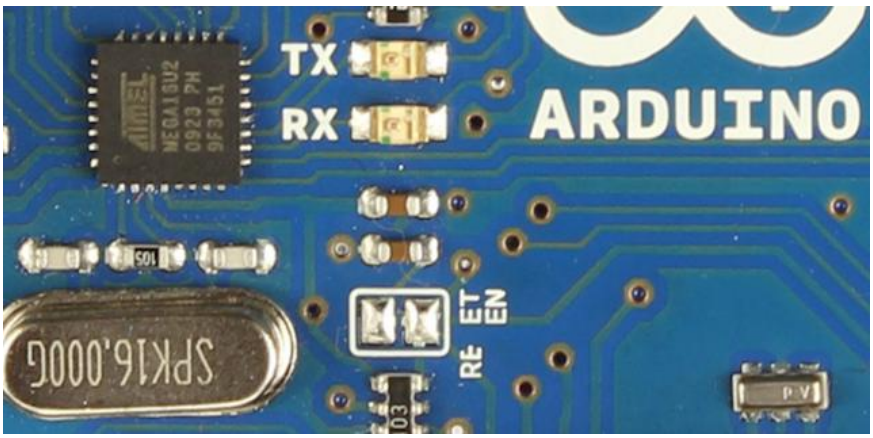


FOTO 6.17: CONEXIÓN *RESET-EN*

Pero si estamos en un entorno de prueba o de preproducción y queremos a subir unas veces código a la placa y otras probarla sin el reseteo, disponemos también del pin *reset* que, al conectarlo a bajo nivel (0 lógico), deshabilita el reseteo. Para poder mantener este valor de tensión debemos conectar una resistencia de 120 Ohm entre el pin de 5 V y el pin de *reset*:

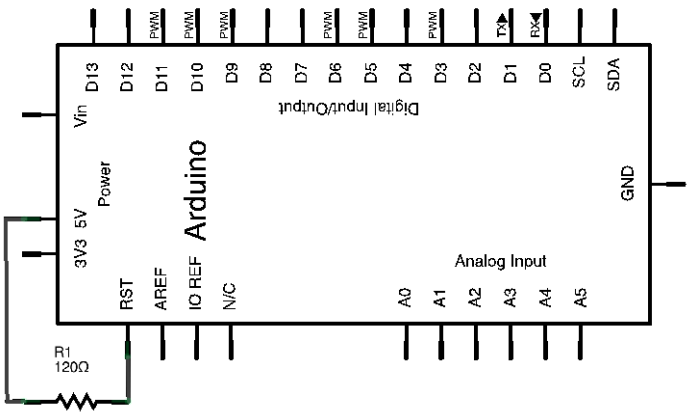


FIGURA 6.13: DIAGRAMA DE CONEXIONES PARA DESHABILITAR EL RESETEO AUTOMÁTICO

Sin embargo, debido a los cambios introducidos internamente en la placa Arduino UNO, el uso de una resistencia no resulta efectivo. En lugar de ésta tenemos que conectar un condensador electrolítico de 10 uF con el ánodo (pata corta, negativo) conectado a la tierra y el cátodo (pata larga, positivo) al pin de *reset:*

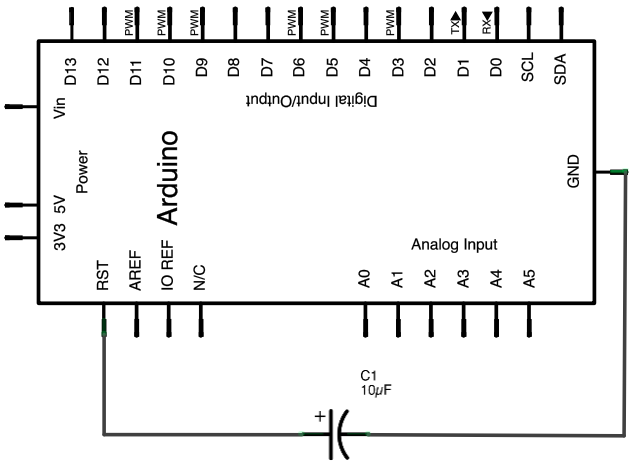


FIGURA 6.14: DIAGRAMA DE CONEXIONES PARA DESHABILITAR EL RESETEO AUTOMÁTICO EN ARDUINO UNO

6.9.2. Indicador led

Resulta interesante añadir un indicador led a nuestra placa para que cuando activemos el riego éste se encienda. Esto es especialmente útil cuando las válvulas de riego no están cercanas a la placa (las características de la corriente continua permiten enviar pulsos sin problemas entre distancias de hasta 30 metros).

Por una parte tenemos la opción de utilizar un pequeño led que se encuentra conectado a la placa en el pin 13. Pero este led es de pequeño tamaño y de muy poca intensidad, por lo que se ha optado por añadir un led rojo entre el pin 6 y la tierra, con una pequeña resistencia de 220 Ohm que le otorga una intensidad alta:

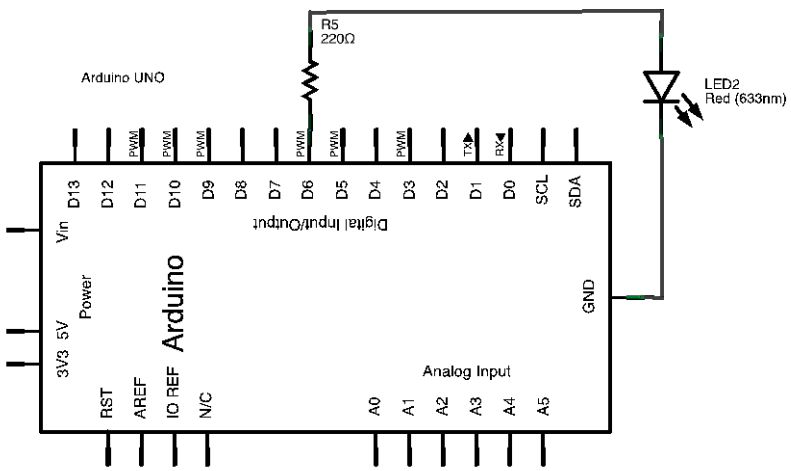


FIGURA 6.15: DIAGRAMA DE CONEXIONES DEL INDICADOR LED DE RIEGO

Además en el *sketch* Arduino incluiremos el código para activar y desactivar el led cuando se inicie y se apague el riego:

```
case 0x64: // activarRiego: d 100 0x64
       riego = 1;
       digitalWrite(enable, HIGH);
       digitalWrite(motor1Pin, LOW);
       digitalWrite(motor2Pin, HIGH);
       digitalWrite(6, HIGH);
[[...]]
case 0x65:  // desactivarRiego: e 101 0x65
       riego = 0;
       digitalWrite(enable, HIGH);
       digitalWrite(motor1Pin, HIGH);
       digitalWrite(motor2Pin, LOW);
       digitalWrite(6, LOW);
[[...]]
```

6.9.3. Actualización automática de sensores

En esta primera fase del proyecto se ha decidido incluir la idea de poder refrescar automáticamente los valores ambientales obtenidos a través de los sensores.

Así en lugar de tener que actualizar manualmente cada uno de los sensores cuando se esté esperando un cambio, el usuario podrá marcar una casilla y seleccionar la frecuencia de refresco para que el programa lo haga automáticamente.

Esta nueva funcionalidad solo afectará a la capa interfaz.

En primer lugar se definirá un *checkbox*, por defecto desmarcado, y un *spinner* para seleccionar la frecuencia del muestreo:

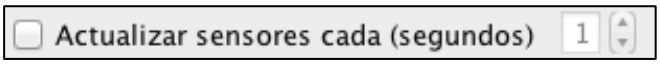


FOTO 6.18: INTERFAZ DE USUARIO DE LA ACTUALIZACIÓN AUTOMÁTICA DE SENSORES

Además incluiremos un nuevo hilo, que se creará al marcar el *checkbox*, que en segundo plano consultará los sensores y actualizará los valores. Este hilo funcionará como un refresco individual de cada sensor:

```
private class RelojSensores extends Thread {
   private boolean activo = true;
   public RelojSensores() {
        this.setName("Hilo de actualización de los sensores");
   }
   public void detener() {
        this.activo = false;
   }
   public void run() {
        System.out.println("Inicio hilo actualizar sensores");
        int i = 0;
        while (activo) {
                System.out.println("Iteración hilo actualizar sensores");
                // Sensores de temperatura
                String[] sensores = logica.listarSensoresT();// Sensores Temperatura
                eliminarSensoresT();
                for (String sensor : sensores) {
                        Float res = logica.obtenerTemperatura(sensor);
                        anyardirSensor(sensor, res);}
                Long presion = logica.obtenerPresionBMP085();// Sensor BMP085
                l_Presion.setText(presion.toString());
                Float temp = logica.obtenerTemperaturaBMP085();
                l_Temperatura.setText(temp.toString());
                Float alt = logica.obtenerAlturaBMP085();
                l_Altura.setText(alt.toString());
                Float humedad = logica.obtenerHumedadHH10D(); // Sensor HH10D
                l_humedad.setText(humedad.toString());
                int humedadSuelo = logica.obtenerHumedadSuelo();//Sensor H del suelo
                l_humedadSuelo.setText(String.valueOf(humedadSuelo));
                Thread.sleep((Integer)spiActualizarSensores.getValue()*1000);
        }
        System.out.println("Fin hilo actualizar sensores");
   }
}
```

El motivo por el cual el hilo se ha creado en la capa interfaz es debido a la estructura por niveles de la aplicación: al consultar a la capa negocio los valores y escribir en la interfaz debemos considerarlo como un evento de la capa interfaz.

6.9.4. Generación de versiones personalizadas para los Windows y OS X

La aplicación RegAdmin se ha desarrollado en el lenguaje de programación Java con el objetivo de obtener una aplicación multiplataforma.

Descargando el código fuente del repositorio SVN (https://sysreg.googlecode.com/svn/trunk/src) y las librerías utilizadas (RXTX y *google-java-client*) podemos generar un fichero empaquetado ejecutable en sistemas operativos Windows, Linux y OS X que tengan instalada la máquina virtual Java 6 o superior.

Además esta compatibilidad ha sido refinada a nivel de código introduciendo directrices como el uso de la codificación UTF-8 (para preservar acentos y caracteres hispanos) y opciones dependientes del SO como la barra de menús superior en OS X.

Sin embargo por culpa del uso de la librería de bajo nivel RXTX, cuyos binarios están compilados para cada SO, se requiere para ejecutar el *jar* de la instalación de

dicha librería. Esta instalación, documentada en la rama de librerías del repositorio (*svn/branches/librerias/rxtx-2.1-7-bins-r2)*, consiste en:

- OS X: Copiar en */Library/Java/Extensions* los ficheros *RXTXcomm.jar*, *librxtxSerial.jnilib* y ejecutar en la terminal:

```
sudo mkdir /var/lock
sudo chmod a+wrx /var/lock
```

- Windows: En el directorio del JRE (usualmente *C:\Archivos de programa\Java\jre6*) copiar los ficheros *rxtxSerial.dll* y *RXTXcomm.jar*.
- Linux: Instalar con un gestor de paquetes (e. g. el *apt-get*) el componente *librxtx-java*.

Así pues para facilitar la instalación y por otros motivos estéticos, como la utilización de iconos propios y la aparición en el ejecutable del nombre del proyecto, hemos decidido lanzar además dos versiones personalizadas para los sistemas Windows y OS X.

6.9.4.1. OS X

Para generar una aplicación OS X utilizaremos el programa *Empaquetador de archivos JAR* incluido en la suite de desarrollo Xcode.

En este programa seleccionaremos el *jar* creado con Eclipse, elegiremos un icono en formato *.icns* (que lo podemos crear a partir de imágenes con la aplicación *Icon Composer*, también en Xcode) y en el apartado *Classpath and Files* incluiremos el fichero *librxtxSerial.jnilib* , tal y como muestran las siguientes imágenes:

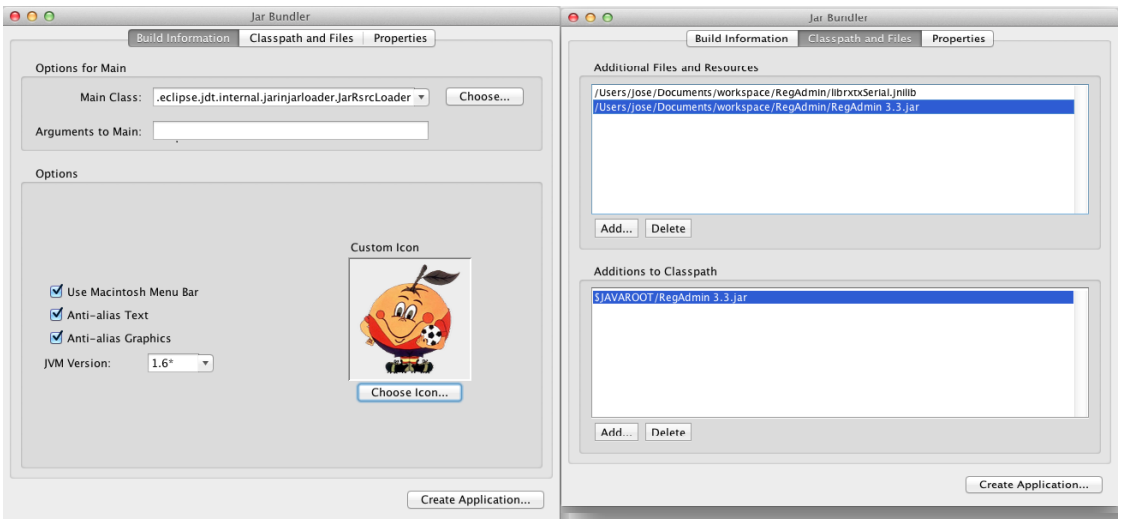


FOTO 6.19: CONFIGURACIÓN DEL *JAR BUNDLER*

Además generaremos un fichero *.dmg* (*Apple Disk Image*) que comprimirá la aplicación y de una manera visual nos guiará en la instalación. Esta imagen la podemos crear a través de comandos en una terminal o bien con la aplicación gratuita DMG Architect.

FOTO 6.20: DMG ARCHITECT

6.9.4.1. Windows

Pese a todo lo contado al principio de la sección, para ejecutar la aplicación en Windows no es del todo imprescindible instalar la librería RXTX. Simplemente con dejar los ficheros *rxtxSerial.dll*, compilado para correspondiente la arquitectura (32 o 64 bits), y *RXTXcomm.jar* en la misma ruta, nuestro ejecutable multiplataforma funcionará.

Pero si queremos personalizar el icono y otras cosas, como el nombre, tenemos que recurrir a empaquetadores o *wrappers*.

Estos programas permiten añadir opciones dependientes de los sistemas operativos a las aplicaciones Java. Hablamos de modificar las variables de entorno, los argumentos de la máquina virtual, realizar comprobaciones de versiones, mostrar mensajes al iniciar el programa, establecer iconos…

En nuestro caso construiremos un fichero ejecutable *.exe* a partir del *jar* que incluya el icono, el nombre de la aplicación y la comprobación de que la máquina virtual instalada es como mínimo la 6. Para ello utilizaremos el programa Launch4j.

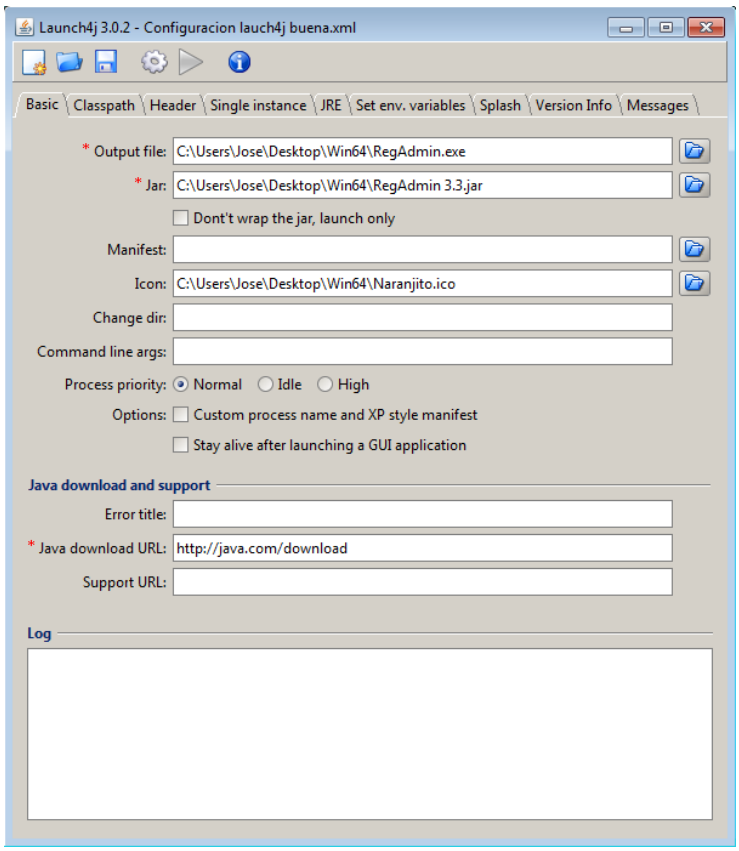


FOTO 6.21: EMPAQUETADOR LAUNCH4J

Sin embargo este programa no es capaz de incorporar ficheros al *classpath*, por lo que no podremos incluir las librerías RXTX y el *jar* en un mismo ejecutable. En su lugar tendremos que generar una carpeta con el ejecutable y los ficheros necesarios, que podremos copiar a modo de instalación, por ejemplo, en la carpeta archivos de programa.

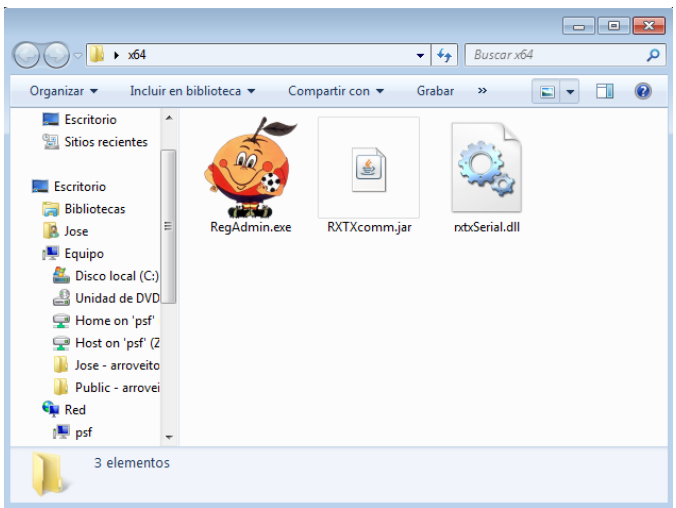


FOTO 6.22: REGADMIN PARA WINDOWS X64

7. DESPLIEGUE

En el presente capítulo describiremos el despliegue realizado de la primera fase del proyecto Sysreg en un escenario real.

Primeramente revisaremos la integración de todas las fases de implementación en el primer prototipo funcional, para después estudiar el despliegue en un escenario ideal (cercano, pequeño y con todos los medios necesarios): un jardín.

Al final analizaremos los resultados obtenidos en este contexto para terminar con las conclusiones de la fase RegAdmin.

7.1. Introducción

En primer lugar debemos ensamblar nuestra primera unidad funcional a partir de todo el trabajo realizado anteriormente.

La placa resultante de la fase uno ha quedado plasmada en el documento Fritzing *Alreg.fzz* (disponible en la ruta del repositorio */trunk/Alreg/*) cuya vista de *Protoboard* es la siguiente:

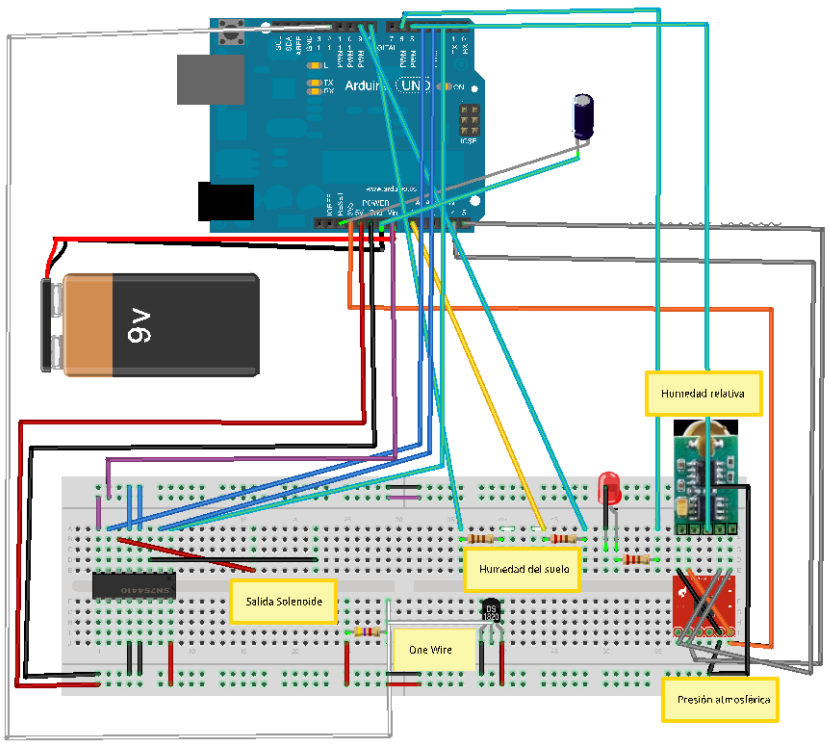


FIGURA 7.1: DIAGRAMA DE CONEXIONES DE LA PRIMERA FASE DEL PROYECTO SYSREG

Ensamblando todos los componentes en una placa Arduino UNO el resultado hardware final ha sido:

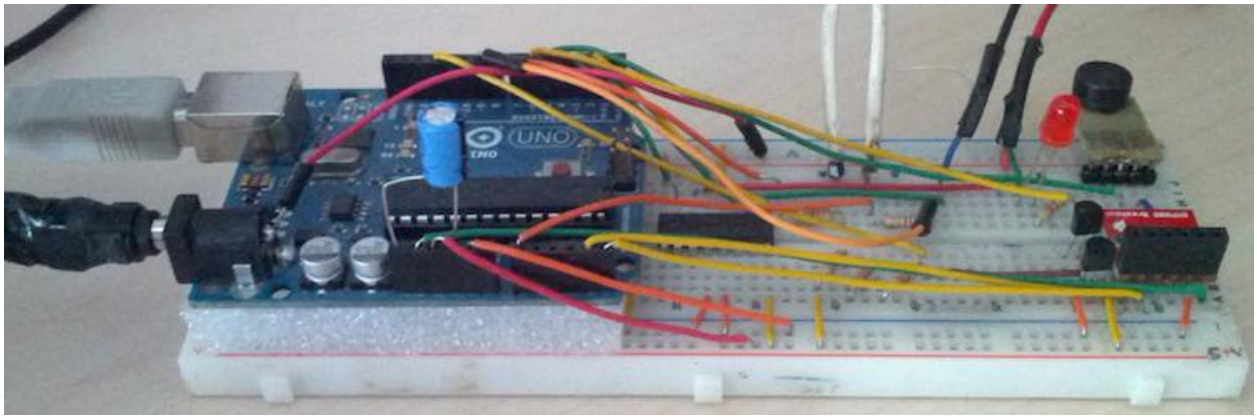


FOTO 7.1: UNIDAD FUNCIONAL DE LA FASE REGADMIN

Por la parte software se ha compilado y subido a la placa Ardurino el *sketch* AlReg (disponible en */trunk/AlReg/Alreg.ino*) y generado la aplicación RegAdmin que incluye las características implementadas (disponible en */trunk/src*).

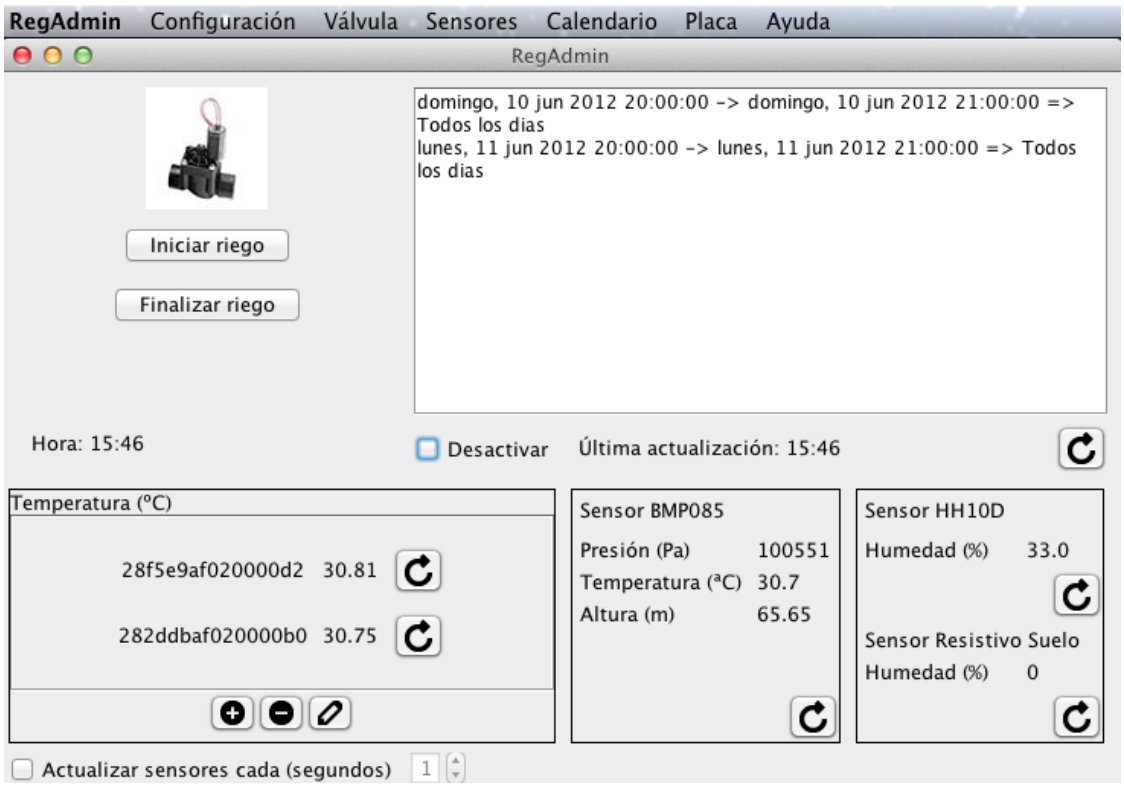


FOTO 7.2: REGADMIN OS X

7.2. Escenario jardín

El escenario de estudio está formado por un recinto de 30 m^2 con unas 30 plantas entre las que encontramos plantas ornamentales (adelfas, dientes de león, lavanda, romero, albahaca…), palmeras, pinos y frutales.

Dispone de un grifo de agua caliente y fría, toma de corriente doméstica (220 V de corriente alterna con un máximo de 25 A) y de un servidor OS X versión 10.7.4 (Lion) a 7 metros con acceso a la red (incluida IP pública) disponible las 24 horas del día.

Para llevar a cabo el despliegue de la fase RegAdmin han sido necesarias realizar las siguientes instalaciones:

- Riego por goteo: Se ha desplegado una tubería maestra que distribuye el agua a cada maceta con goteros autocompensados.

FOTO 7.3: INSTALACIÓN DE RIEGO POR GOTEO

- Válvula de riego: Para automatizar el riego se ha sustituido el grifo por un codo de ¾ compatible con el tamaño de la válvula de riego. Esta válvula estará controlada por un solenoide de tipo *latch* con posibilidad accionamiento manual.

FOTO 7.4: INSTALACIÓN DE LA VÁLVULA DE RIEGO

- Sensores: Para monitorizar el escenario se ha instalado en el exterior un sensor de temperatura y un sensor de humedad del suelo, en la maceta con mayor volumen. En el interior, junto a la placa, encontraremos el sensor de presión, el de humedad y otro de temperatura.

FOTO 7.5: INSTALACIÓN DEL SENSOR DE HUMEDAD DEL SUELO

- Instalación de la placa: Al servidor OS X se ha conectado la placa por USB con un transformador de 9 V alimentándola por la entrada *jack*.

Para poder demostrar las ventajas de nuestra solución se realizó un estudio de 2 semanas de duración entre el 14 y el 28 de Mayo.

Durante la primera semana de este estudio se llevó a cabo el riego manualmente todos los días a las 8 de la noche. Se utilizaron 12 litros de agua para cubrir las necesidades hídricas de este periodo primaveral, haciendo un total de 84 litros, y requiriendo de 10 minutos de trabajo (en total una hora y diez minutos).

FOTO 7.6: ESCENARIO

En cambio utilizando el prototipo funcional programamos una alarma de riego diaria de 5 minutos de duración a las 8 de la tarde que consumió un total de 17 litros y medio de agua (100 ml por cada gotero los 7 días de la semana). Además durante los días 25 y 26 de Mayo, en los que se produjo un aumento considerable de las temperaturas, mediante el calendario en la nube se programaron dos riegos puntuales de 3 minutos a las 12 de la mañana, consumiendo 4 litros más de agua. En cuanto a gasto de personal solamente se pueden contabilizar los dos minutos empleados en definir la alarmas puntuales.

Por tanto podemos concluir que para este ejemplo los resultado obtenidos han revelado un importante ahorro de agua del 71% (se ha evitado la evaporación de agua, el excedente puntual y se ha precisado más la dosis) y se ha eliminado la intervención

humana en la gestión parcelaria. Además hemos conseguido indicadores precisos del estado actual, que han ayudado a actuar en situaciones puntuales.

También se ha producido una mejora a nivel de comodidad y de movilidad, que permitirán el cultivo de productos más sensibles, como verduras.

Aunque este escenario pueda parecer un análisis simplista, lo que quiere poner en relieve es que los avances son notorios a nivel de productividad y de eficiencia. Aunque por falta de tiempo no se haya podido proyectar dos escenarios más (como iban a ser una parcela citrícola y un invernadero) los resultados son fácilmente extrapolables: con esta versión ya obtenemos unos tiempos de acceso menores, un aprovechamiento de materias mayor, la obtención de nuevos indicadores y una mejora productiva.

8. PROYECTO SiGAM

8.1. Introducción

En esta parte se realizará un estudio de una rama de la tesis que ha permitido proponer un proyecto de investigación y desarrollo en el Grupo Tragsa.

En primer lugar se detallarán las líneas de actuación de Tragsatec, la filial tecnológica de Tragsa, para ver en que áreas se podrían desarrollar proyectos derivados de la tesis, y a continuación se mostrará un proyecto similar (SIAR) del cual exportaremos la formalización del cálculo de necesidad hídrica mediante la evapotranspiración de referencia (la ET_0). Por último detallaremos la propuesta I+D.

8.2. Grupo Tragsa y Tragsatec

El Grupo Tragsa está formado por empresas de capital íntegramente público: Tragsa (Empresa de Trasnformación Agraria, SA) y su filial Tragsatec (Tecnologías y Servicios Agrarios, SA) participadas mayoritariamente por la Sociedad Estatal de Participaciones Industriales (SEPI).

El Grupo se configura como medio propio instrumental y de servicio técnico de la Administración. En esta condición, está obligado a realizar, con carácter exclusivo, los trabajos que le encomiende la Administración General del Estado.

Tragsatec fue constituida en 1989 como empresa filial de la matriz Tragsa y tiene como principales campos las actuaciones de ingeniería.

Dispone de un soporte informático que utiliza medios técnicos avanzados, como sistemas de información geográficos o grandes explotaciones de bases de datos.

Tiene 17 delegaciones autonómicas, 41 delegaciones provinciales y más de 5.500 empleados. Está presente también en otros países a través de la cooperación internacional.

La actividad de Tragsatec queda desglosada en diferentes líneas de actuación:

- Calidad y evaluación ambiental: El desarrollo sostenible requiere una concepción integral del medio ambiente y de su interacción con la actividad humana. En Tragsatec, se realizan tareas planificación ambiental, incorporando

la variable ambiental a la actividad productiva. Hablamos pues de proyectos de mitigación del cambio climático, evaluación de productos fitosanitarios, implantación de sistemas de calidad ambiental, tratamientos de plagas…

- Biodiversidad: La presión que ejerce la actividad humana sobre los ecosistemas puede provocar una extrema simplificación de los mismos si no se desarrollan estrategias de uso sostenible de la diversidad biológica. Tragsatec cuenta con medios científico-técnicos cualificados aportando soluciones a medida acordes a las circunstancias concurrentes en cada caso. Se realiza un inventariado y seguimiento de las especies protegidas, se siguen las especies invasoras y se realiza cartografía, seguimiento y conservación de hábitats.
- Desarrollo rural y ordenación del territorio: El desarrollo rural constituye uno de los pilares básicos de las grandes estrategias territoriales, tanto en su vertiente socioeconómica como ambiental e institucional. El desarrollo rural incluye estudios sociológicos, socioeconómicos y ambientales para identificar las zonas a las que irán dedicadas las ayudas. Por otro lado la ordenación del territorio incluye tanto la concentración parcelaria (reorganización de la propiedad para la mejora de explotaciones) y la gestión patrimonial.
- Políticas agrarias: La utilización de las tecnologías de la información es hoy en día un elemento ineludible en la planificación y gestión de las políticas de cualquier ámbito. Esta es una de las líneas más importantes en Tragsatec donde se utilizan tecnologías punteras para ofrecer un servicio de calidad a las Administraciones Públicas. En esta línea de actuación destaca el Sistema de información de parcelas agrícolas (SIGPAC), utilizado para la gestión de ayudas de la PAC, y el mantenimiento de datos agrícolas (como registros vitícolas, citrícolas…) utilizados en muchas aplicaciones como orígenes de datos.
- Servicios agrícolas: Las nuevas tecnologías permiten desarrollar una agricultura intensiva pero que necesita de un mejor control de las plagas y enfermedades, así como de una mejora en los sistemas de producción agroalimentaria. Tragsatec ofrece apoyo a las Administraciones Públicas en la prestación de todos aquellos servicios que se consideren necesarios tanto en materia fitosanitaria como agroalimentaria. Se trata de trabajos como la vigilancia fitosanitaria, estudios sectoriales, control de calidad comercial, funciones de promoción agroalimentaria…

- Servicios ganaderos: Tragsatec realiza actividades relacionadas con la sanidad y la producción animal. Se realizan tareas de gestión de alertas sanitarias, programas de control y erradicación de enfermedades animales, actuaciones en fauna silvestre, estudios epidemiológicos, recogida de animales muertos, apoyo a la exportación importación de animales…
- Pesca y asuntos marítimos: España es un país con una importante cultura marítima y Tragsatec se encarga de realizar tareas científicas, trabajos de gestión y labores educativas que logren compaginar la conservación de los recursos y valores naturales de los ecosistemas marinos con un uso y aprovechamiento sostenible de los mismos.
- Calidad y seguridad alimentaria: Tragsatec cuenta con equipos especializados en el desarrollo, implantación y mantenimiento de mecanismos de control en todas las fases de la cadena productiva. Hablamos de calidad alimentaria (denominaciones de origen) y seguridad alimentaria (control de alimentos, mataderos, residuos y laboratorios agroalimentarios).
- Salud pública: Aquí se agrupan las líneas estratégicas de la política sanitaria, que se articulan a través de acciones destinadas a proteger, promover y restaurar la salud del conjunto de la población promoviendo, además, estilos de vida saludables. Se controlan los riesgos químicos (sustancias fitosanitarias, biocidas…), riesgos biológicos, riesgos ambientales (como plagas) además de diseñar sistemas de control de aguas y de medicamentos.
- Infraestructuras, instalaciones y equipamientos: Desde el inicio Tragsatec realiza proyectos de ingeniería civil (redactando proyectos, edificando, rehabilitando estructuras) y de arquitectura, edificación y urbanismo.
- Tecnologías de la información: Destacan los proyectos de Sistemas de Información Geográfica (con aplicaciones SIG a medida, servidores cartográficos Web y visores Web) colaborando activamente en la utilización de estándares abiertos e interoperables OpenGIS dentro del OGC (Open Geospatial Consortium). Se desarrollan también aplicaciones para dispositivos móviles en diferentes sistemas operativos (Windows Mobile, Windows Phone 7, Symbian, Android) utilizando tecnologías inalámbricas como GPRS/UMTS, Bluetooth, WiMAX y posicionamiento por GPS. Otras áreas de trabajo son los sistemas de

información de gestión administrativa, teledetección y fotogrametría, interoperabilidad entre servicios y consultoría.

- Gestión documental: Tragsatec ofrece servicios de desarrollo e implantación de soluciones informatizadas que faciliten la gestión de los expedientes administrativos desde su registro hasta su archivo. Estas soluciones, basadas en la implantación de herramientas de gestión documental y herramientas BPM (*Bussiness Process Management*), ayudan a disminuir el consumo de papel, mejoran los tiempos de tramitación y eliminan posibles errores derivados de la falta de trazabilidad de un expediente.
- Gestión y tecnologías del agua: Una de las líneas de actuación más importantes en los últimos años han sido la gestión y tecnología del agua. Esta línea se centra en prestar consultoría, asesoramiento y servicios técnicos para apoyar a las funciones de planificación y gestión del agua que recaen en las administraciones hídricas, así como en las entidades encargadas del abastecimiento, saneamiento y depuración de las aglomeraciones urbanas. Por otra parte procura las soluciones tecnológicas más adecuadas para garantizar la utilización sostenible del recurso, así como la prevención y defensa frente a inundaciones u otras situaciones adversas. Los trabajos se realizan principalmente en áreas de planificación hidrológica (planes de sequia, ordenación de acuíferos…), gestión hidrológica (ordenación territorial, censo de vertidos…) e ingeniería del agua (regadíos, cuencas, embales…).

El proyecto Sysreg tiene un marco de aplicación tan amplio que se puede incluir en cualquiera de las líneas detalladas. Sin embargo, el tratamiento del agua un tema tan delicado y actual que ha motivado que la primera línea de investigación salida de esta tesina vaya enfocadas a enriquecer la ingeniería del agua, siempre desde el punto de vista de las tecnologías de la información.

8.3. Proyectos anteriores

Como hemos visto en las líneas de actuación, los proyectos en gestión hídrica y tecnologías de la información son muy variados. Pero si alguno de ellos destaca por la similitud en objetivos, herramientas y ámbito, ese es SIAR.

8.3.1. SIAR

El Sistema de Información Agroclimática para el Regadío (SIAR) es un proyecto iniciado por el Ministerio de Agricultura, Pesca y Alimentación en el año 1998. Este proyecto promueve la creación de una infraestructura de captura, registro y trasmisión de datos para el posterior cálculo de la demanda hídrica de zonas bajo riego.

Como parámetros de cálculo utiliza la temperatura, humedad del aire, velocidad del viento, radiación solar y precipitación.

Utilizando instrumentos de calidad y acceso en tiempo real consigue una fiabilidad de datos que, unida a la aplicación de la fórmula de cálculo de la evapotranspiración de referencia más ajustada para cada zona, proporciona una mayor exactitud en la determinación de las necesidades de riego en los cultivos.

8.3.1.1. Objetivos

Los objetivos a alcanzar por SIAR son los siguientes:

- Obtener los datos agroclimáticos representativos de las zonas bajo riego.
- Determinar la evapotranspiración de referencia (ET_0) para cada zona de riego.
- Asesorar a los regantes: Programación y dosis de riego.
- Instrumentación para la toma de decisiones en la optimización del uso del agua.
- Control fitosanitario.
- Investigación.

8.3.1.2. Descripción

El Ministerio de Agricultura puso el proyecto SIAR en las comunidades autónomas con riesgos de sequia: Andalucía, Canarias, Castilla-La Mancha, Castilla y León, Extremadura, Murcia y Valencia.

Para gestionar esta red distribuida de sensores se estructuró el sistema en 3 niveles: 361 estaciones agroclimáticas automáticas, 12 centros zonales y un centro nacional. Los centros zonales corresponden a las CCAA y obtienen diariamente y de forma automática los datos capturados en las estaciones de cada CA, siendo posteriormente transmitida esa información al centro nacional, donde se explota la información con programas de gestión y cálculo.

Este sistema, que ha sido el punto de partida de los servicios de asesoramiento al regante ofrecidos por las CCAA, esta estructurado de la siguiente manera:

- Estaciones agroclimáticas: 361 estaciones repartidas en 12 CCAA, 45 de ellas en la Comunidad Valenciana. Diseñadas para obtener los datos necesarios para calcular la evapotranspiración potencial.
- Centro zonal: Ubicado y gestionado por cada comunidad autónoma. Cada centro obtiene diariamente y de forma automática (a través de redes GSM) los datos de las estaciones de dicha comunidad. Está compuesto por un servidor que corre la aplicación VRIA, desarrollada específicamente para adquirir los datos, almacenarlos y calcular los diferentes parámetros y variables. Con todo esto se pueden realizar tareas de asesoramiento al regante, de publicación y de distribución de datos.
- Centro nacional: Instalado en la Dirección General de Desarrollo Rural (dependiente del Ministerio de Agricultura) aglutina los datos de todas las estaciones, pudiendo sustituir a los centros zonales en caso de fallo. Ejerce funciones de coordinación general, para mantener la homogeneidad, estableciendo el modelo lógico y conceptual mínimo que cada centro zonal debe cumplir (independiente de su modelo de datos). También realiza los cálculos de la evapotranspiración de referencia (ET_0) y de las demandas hídricas para planificaciones de cuencas.
- Comité técnico: Formado por representantes de las CCAA y de la administración central. Controla la homogeneidad de los datos y discute y consensua los cambios estructurales.

Esta infraestructura permite obtener en tiempo real los datos necesarios para calcular la demanda hídrica de cada zona. Hablamos de la temperatura, humedad, velocidad y dirección del viento, radiación solar y precipitación. Con estás variable se aplica la fórmula de cálculo de la evapotranspiración con el objetivo de proporcionar información relativa a las necesidades de riego.

A continuación mostraremos los dos componentes más importantes del proyecto, las estaciones agroclimáticas y la fórmula ET_0.

8.3.1.2.1. Estaciones agroclimáticas.

Diseñadas para medir y registrar de modo automático y continuo variables meteorológicas de aplicación agrícola.

Están ubicadas en puntos con interés meteorológico (según el Instituto Nacional de Meteorología).

Consta de una anemoveleta (para medir la velocidad y dirección del viento), un piranómetro (para medir la radiación), un pluviómetro y un sensor de temperatura y humedad del aire embebidos en una sonda.

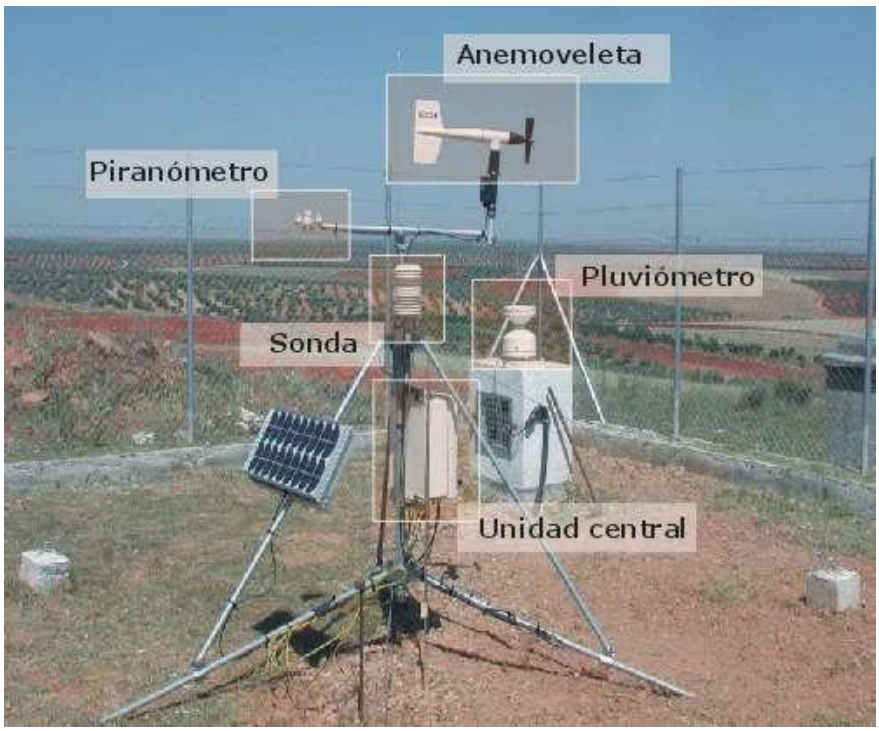


FOTO 8.1: ESTACIÓN AGROCLIMÁTICA DEL PROYECTO SIAR

En la unidad central encontramos los componentes que permiten la recogida y envió de datos. Sus principales componentes son:

- *Datalogger*: Se encarga de recoger la información de los sensores con pines analógicos, realizar los cálculos necesarios, convertir datos de analógico a digital, establecer la frecuencia de muestreo y alimentar los dispositivos.
- *Memoria:* Tenemos por un lado la memoria *flash* donde se carga el sistema operativo y el programa principal del *datalogger*, y por otro la memoria RAM para la ejecución de programas.

- *Panel de conexiones*: Se conectan las entradas analógicas, las tomas de 12 V, 5 V y de tierra.
- *Alimentación*: Para la alimentación del *datalogger* y del módulo de comunicaciones se emplea una batería de 12 V conectada a un panel solar de 20 W

FOTO 8.2: UNIDAD CENTRAL DEL PROYECTO SIAR

8.3.1.2.2. Evapotranspiración de referencia según FAO Penman-Monteith

La evapotranspiración (ET) es la combinación de dos procesos separados por los que se pierde agua: la evaporación a través del suelo y la transpiración del cultivo.

Por un lado la evaporación convierte el agua liquida en vapor eliminándola de la superficie evaporante. Está afectada principalmente por la radiación solar, la temperatura del aire, la humedad atmosférica y la velocidad del viento.

Y por otro, la transpiración consiste en la vaporización del agua contenida en los tejidos de la planta a través de los estomas, y depende de los mismos factores que la evaporación.

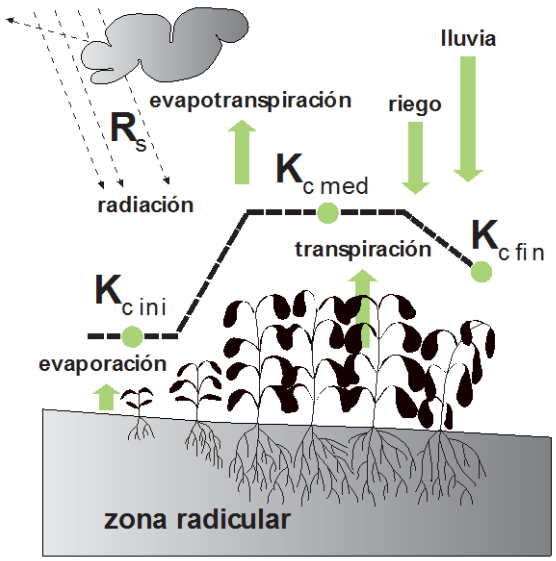


FIGURA 8.1: EVAPOTRANSPIRACIÓN DEL CULTIVO

Estos dos procesos ocurren simultáneamente. En las primeras etapas de cultivo se pierde principalmente por evaporación directa del suelo y a medida que la cosecha crece aumenta el factor de transpiración.

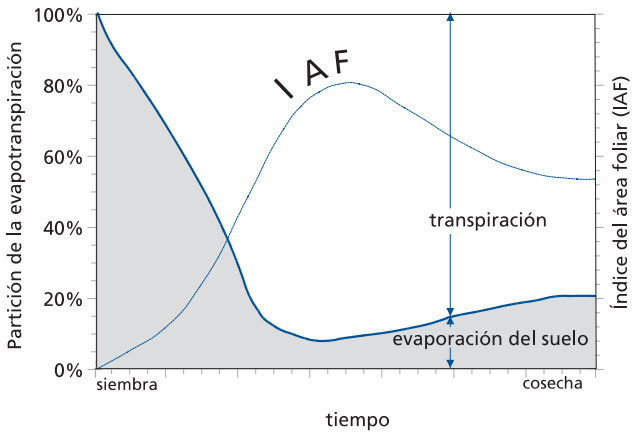


FIGURA 8.2: EVAPOTRANSPIRACIÓN DURANTE EL CICLO DE VIDA VEGETAL

Se expresa en milímetros (mm) por unidad de tiempo (hora, día, mes) y refleja la cantidad de agua perdida por una superficie cultivada en unidades de altura de agua, por ejemplo, en una superficie de una hectárea (10.000 m^2) una perdida de 1 mm de agua corresponde a una pérdida de 10 m^3 de agua (1 mm día $^{-1}$ = 10 m^3 ha^{-1} $día^{-1}$ = 10.000 l ha^{-1} $día^{-1}$).

La ET ha sido estudiada por multitud de expertos y de organizaciones, pero ha sido la FAO (Organización de las Naciones Unidas para la Agricultura y la Alimentación) la principal organización que ha promovido su investigación y estandarización como medida para determinar las necesidades de riego.

Sin embargo la ET es un termino complejo en el que encontramos tres definiciones: la evapotranspiración del cultivo de referencia (ET_0), la evapotranspiración del cultivo bajo condiciones estándar (ET_c) y la evapotranspiración del cultivo bajo varias condiciones de manejo y ambientales ($ET_{c\ aj}$).

La ET_0 se introdujo para estudiar la ET de la atmósfera, de manera independiente del tipo de cultivo. Así pues los únicos factores que afectan a la ET_0 son parámetros climáticos. Este es el valor de ET_0 para un cultivo ideal.

Regiones	Temperatura promedio durante el día (°C)		
	Templada ~10°C	Moderada 20°C	Caliente > 30°C
Trópicos y subtrópicos			
- húmedos y subhúmedos	2 - 3	3 - 5	5 - 7
- áridos y semiáridos	2 – 4	4 - 6	6 - 8
Regiones templadas			
- húmedas y subhúmedas	1 - 2	2 - 4	4 - 7
- áridas y semiáridas	1 – 3	4 - 7	6 - 9

FIGURA 8.3: VALORES TÍPICOS DE LA ET_0

La ET_c es la ET de cualquier cultivo exento de enfermedades, con buena fertilización, en condiciones óptimas de suelo y agua. Básicamente es una ET_0 adecuada al cultivo con el Coeficiente del cultivo K_c ($ET_c = K_c * ET_0$). Requiere de datos propios del cultivo, como resistencia del cultivo y albedo.

Y la $ET_{c\ aj}$ se refiere a la ET de cultivos bajo condiciones ambientales diferentes a las estándares, es decir, condiciones no óptimas (plagas, enfermedades, salinidad del suelo, exceso de agua…). Requiere de datos como el estrés hídrico (K_s), las limitaciones ambientales…

Así pues la fórmula más utilizada es la ET_0, ya que permite determinar la frecuencia y dosis de riego a partir de variables atmosféricas simples. Se utiliza en estaciones meteorológicas por su independencia de cultivo, por ser un factor de referencia y por permitir derivar fórmulas más ajustadas.

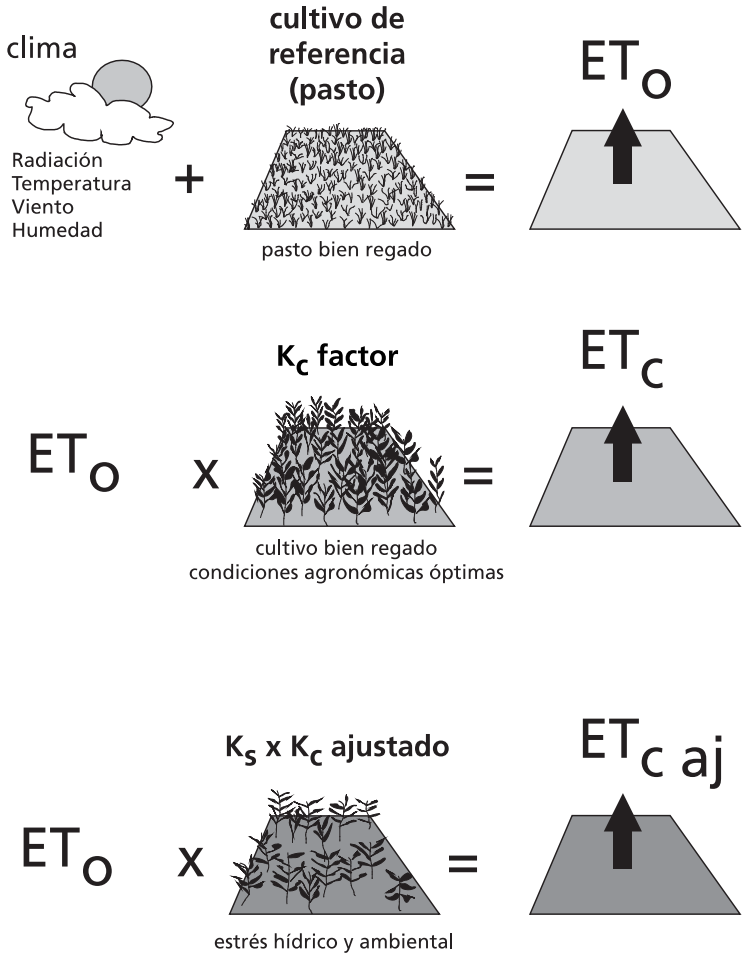


FIGURA 8.4: RELACIONES ENTRE LAS FÓRMULAS DE LA EVAPOTRANSPIRACIÓN

Durante los últimos 50 años se han desarrollado una gran cantidad de métodos, más o menos empíricos, con el fin de estimar la evapotranspiración de referencia (en un pasto ideal, similar a una superficie de altura uniforme, crecimiento activo, completamente sombreado y adecuadamente regado) a partir de variables climáticas como son el método Blanye-Criddle (por radiación) y el de Penman (el método del tanque de evaporación). Pero en 1990 un comité de expertos organizado por la FAO consideró que el que ofrecía mejores resultados con el mínimo error era el de Penman modificado (método Penman-Monteith).

En el método original, Penman derivó una ecuación para calcular la evaporación en una superficie abierta a partir de datos climáticos estándar: horas de sol, temperatura, humedad atmosférica y velocidad de viento.

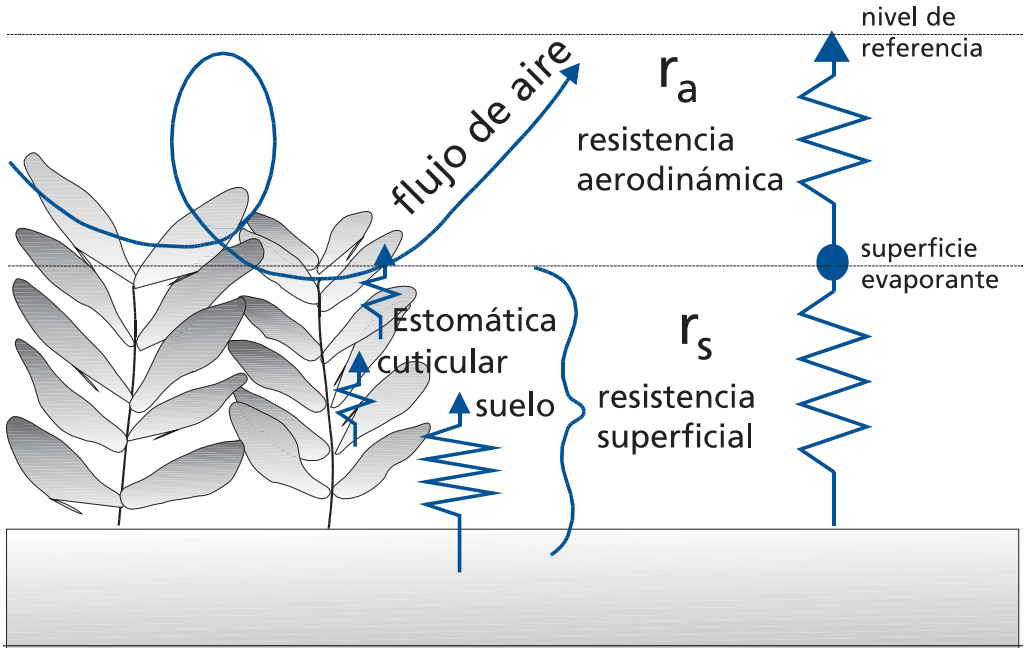


FIGURA 8.5: FACTORES DE RESISTENCIA SUPERFICIAL Y AERODINÁMICA QUE INFLUYEN EN EL PROCESO DE ET

La ecuación original era:

$$\lambda ET = \frac{\Delta(R_n - G) + \rho_a c_p \dfrac{(e_s - e_a)}{r_a}}{\Delta + \gamma\left(1 + \dfrac{r_s}{r_a}\right)}$$

ECUACIÓN 8.1: ECUACIÓN ORIGINAL DEL MÉTODO PENMAN

Donde R_n es la radiación neta, G es el flujo del calor en el suelo, $(e_s - e_a)$ representa el déficit de presión de vapor del aire, ρ_a es la densidad media del aire a presión constante, c_p es el calor específico del aire, Δ representa la pendiente de la curva de presión de vapor de saturación, γ es la constante psicrométrica, y r_s y r_a son las resistencias superficial (total) y aerodinámica.

Pero en el citado congreso de la FAO, además de seleccionar este método como estándar para el cálculo de la evapotranspiración de referencia, el método se revisó, estableciendo la definición de cultivo de referencia como un cultivo de una altura de 0,12 cm, resistencia superficial de 70 s m^{-1} y un albedo de 0,23, con el objetivo de reducir las imprecisiones y producir valores más consistentes.

A continuación detallaremos la fórmula definida por la FAO para el cálculo de la ET_0:

$$ET_o = \frac{0{,}408\ \Delta (R_n - G) + \gamma \frac{900}{T + 273} u_2 (e_s - e_a)}{\Delta + \gamma (1 + 0{,}34\ u_2)}$$

ECUACIÓN 8.2: EVAPOTRANSPIRACIÓN DE REFERENCIA

donde:

ET_o	evapotranspiración de referencia (mm día^{-1})
R_n	radiación neta en la superficie del cultivo (MJ m^{-2} día^{-1})
R_a	radiación extraterrestre (mm día^{-1})
G	flujo del calor de suelo (MJ m^{-2} día^{-1})
T	temperatura media del aire a 2 m de altura (°C)
u_2	velocidad del viento a 2 m de altura (m s^{-1})
e_s	presión de vapor de saturación (kPa)
e_a	presión real de vapor (kPa)
$e_s - e_a$	déficit de presión de vapor (kPa)
Δ	pendiente de la curva de presión de vapor (kPa °C^{-1})
γ	constante psicrométrica (kPa °C^{-1})

Todas las variables se pueden obtener a partir de la radiación solar, la temperatura del aire, la humedad relativa y la velocidad del viento:

- Presión atmosférica:

$$P = 101{,}3 \left(\frac{293 - 0{,}0065\, z}{293} \right)^{5{,}26}$$

ECUACIÓN 8.3: PRESIÓN ATMOSFÉRICA

donde

P	Presión atmosférica [kPa]
z	Elevación sobre el nivel del mar [m]

- Constante psicrométrica:

$$\gamma = \frac{c_p P}{\varepsilon \lambda} = 0{,}665 * 10^{-3}$$

ECUACIÓN 8.4: CONSTANTE PSICROMÉTRICA

donde

γ	constante psicrométrica [kPa °C^{-1}],
P	presión atmosférica [kPa],
λ	calor latente de vaporización, 2,45 [MJ kg^{-1}],
c_p	calor específico a presión constante, 1,013 x 10^{-3} [MJ kg^{-1} °C^{-1}],
ε	cociente del peso molecular de vapor de agua /aire seco = 0,622.

- Temperatura media:

$$T_{media} = \frac{T_{max} + T_{min}}{2}$$

ECUACIÓN 8.5: TEMPERATURA MEDIA

- Presión media de vapor de la saturación:

$$e^o(T) = 0{,}6108 * \exp\left[\frac{17{,}27 * T}{T + 237{,}3}\right]$$

ECUACIÓN 8.6: PRESIÓN DE SATURACIÓN DE VAPOR

donde

$e^o(T)$	presión de saturación de vapor a la temperatura del aire, T [kPa]
T	temperatura del aire [°C]
exp [..]	2,7183 (base del logaritmo natural) elevado a la potencia [..]

$$e_s = \frac{e^o(T\,max) + e^o(T\,min)}{2}$$

ECUACIÓN 8.7: PRESIÓN MEDIA DE VAPOR DE SATURACIÓN

- Pendiente de la curva de presión de saturación de vapor:

$$\Delta = \frac{4098 * \left[0{,}6108 * \exp\left(\frac{17.27 * T}{T + 237{,}3}\right)\right]}{(T + 237{,}3)^2}$$

ECUACIÓN 8.8: PENDIENTE DE LA CURVA DE PRESIÓN DE SATURACIÓN DEL VAPOR

donde

Δ	pendiente de la curva de la presión de saturación de vapor a la temperatura del aire T [kPa °C^{-1}]
T	temperatura del aire [°C]
exp[..]	2,7183 (base del logaritmo natural) elevado a la potencia [..]

- Presión de vapor real:

$$e_a = \frac{e^o(T_{min})\frac{HR_{max}}{100} + e^o(T_{max})\frac{HR_{min}}{100}}{2}$$

ECUACIÓN 8.9: PRESIÓN DE VAPOR REAL

donde

e_a	presión real de vapor [kPa]
$e°(T_{min})$	presión de saturación de vapor a la temperatura mínima diaria [kPa
$e°(T_{max})$	presión de saturación de vapor a la temperatura máxima diaria [kPa
HR_{max}	humedad relativa máxima [%]
HR_{min}	humedad relativa mínima [%].

- Radiación (R_n – G)

La radiación es un cálculo muy complejo ya que incluye muchos componentes:

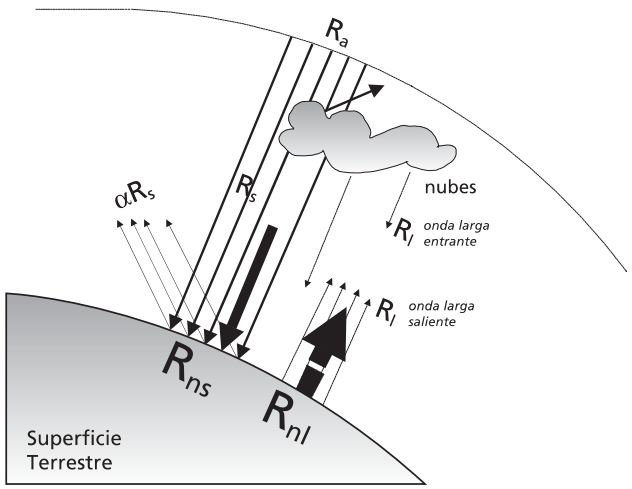


FIGURA 8.6: COMPONENTES DE LA RADIACIÓN SOLAR

- Radiación extraterrestre para periodos diarios (Ra):

$$R_a = \frac{24*60}{\pi} G_{sc} d_r \left[\omega_s \sin(\varphi)\sin(\delta) + \cos(\varphi)\cos(\delta)\sin(\omega)\right]$$

ECUACIÓN 8.10: RADIACIÓN EXTRATERRESTRE

donde

R_a radiación extraterrestre [MJ m^{-2} $día^{-1}$]
G_{sc} constante solar = 0,082 MJ m^{-2} min^{-1},
d_r distancia relativa inversa Tierra-Sol (Ecuación 23)
ω_s ángulo de radiación a la puesta del sol (Ecuaciones 25 o 26) [rad]
φ latitud [rad] (Ecuación 22)
δ declinación solar (Ecuación 24) [rad].

$$d_r = 1 + 0{,}033 * \cos\left(\frac{2\pi}{365} J\right)$$

$$\delta = 0{,}409 * \text{sen}\left(\frac{2\pi}{365} J - 1{,}39\right)$$

ECUACIÓN 8.11: DISTANCIA RELATIVA INVERSA TIERRA-SOL Y LATITUD

donde J es el número del día del año entre 1 y 365.

$$\omega_s = \frac{\pi}{2} - \arctan\left[\frac{-\tan(\varphi)\tan(\delta)}{X^{0,5}}\right]$$

ECUACIÓN 8.12: ÁNGULO DE RADIACIÓN

donde

$X = 1 - [\tan(\varphi)]^2 [\tan(\delta)]^2$
y $X = 0{,}00001$ si $X \leq 0$

- Duración máxima de la insolación (N):

$$N = \frac{24}{\pi} \omega_s$$

ECUACIÓN 8.13: DURACIÓN MÁXIMA DE LA INSOLACIÓN

- Radiación solar (R_s)

$$R_s = \left(a_s + b_s \frac{n}{N} \right) R_a$$

ECUACIÓN 8.14: RADIACIÓN SOLAR

donde

R_s	radiación solar o de onda corta [MJ m^{-2} $día^{-1}$],
n	duración real de la insolación [horas],
N	duración máxima posible de la insolación [horas],
n/N	duración relativa de la insolación[-],
R_a	radiación extraterrestre [MJ m^{-2} $día^{-1}$],
a_s	constante de regresión, que expresa la fracción radiación extraterrestre que llega a la tierra en días muy nublados (n = 0),
$a_s + b_s$	fracción de la radiación extraterrestre que llega a la tierra en días despejados (n = N).

- Radiación neta solar o de onda corta (R_{ns}):

$$R_{ns} = (1 - \alpha) R_s$$

ECUACIÓN 8.15: RADIACIÓN NETA DE ONDA CORTA

donde

R_{ns}	radiación neta solar o de onda corta [MJ m^{-2} $día^{-1}$],
α	albedo o coeficiente de reflexión del cultivo, que es 0,23 para el cultivo hipotético de referencia [adimensional],
R_s	radiación solar entrante [MJ m^{-2} $día^{-1}$].

- Radiación neta de onda larga (R_{ns})

$$R_{nl} = \sigma \left[\frac{T_{max,K}^{\ 4} + T_{min,K}^{\ 4}}{2} \right] \left(0{,}34 - 0{,}14 \sqrt{e_a} \right) \left(1{,}35 \frac{R_s}{R_{so}} - 0{,}35 \right)$$

ECUACIÓN 8.16: RADIACIÓN NETA DE ONDA LARGA

donde

R_{nl}	Radiación neta de onda larga [MJ m^{-2} $día^{-1}$],
σ	constante de Stefan-Boltzmann [4,903 x 10^{-9} MJ K^{-4} m^{-2} $día^{-1}$],
$T_{max,K}$	temperatura máxima absoluta durante un periodo de 24 horas [K = °C + 273,16],
$T_{min,K}$	temperatura mínima absoluta durante un periodo de 24 horas [K = °C + 273,16],
e_a	presión de vapor real [kPa],
R_s/R_{so}	radiación relativa de onda corta (valores ≤ 1,0),

- Radiación neta (R_n)

$$R_n = R_{ns} - R_{nl}$$

ECUACIÓN 8.17: RADIACIÓN NETA

- Flujo del calor del suelo (G)

$$G = c_s \frac{T_i - T_{i-1}}{\Delta t} \Delta z$$

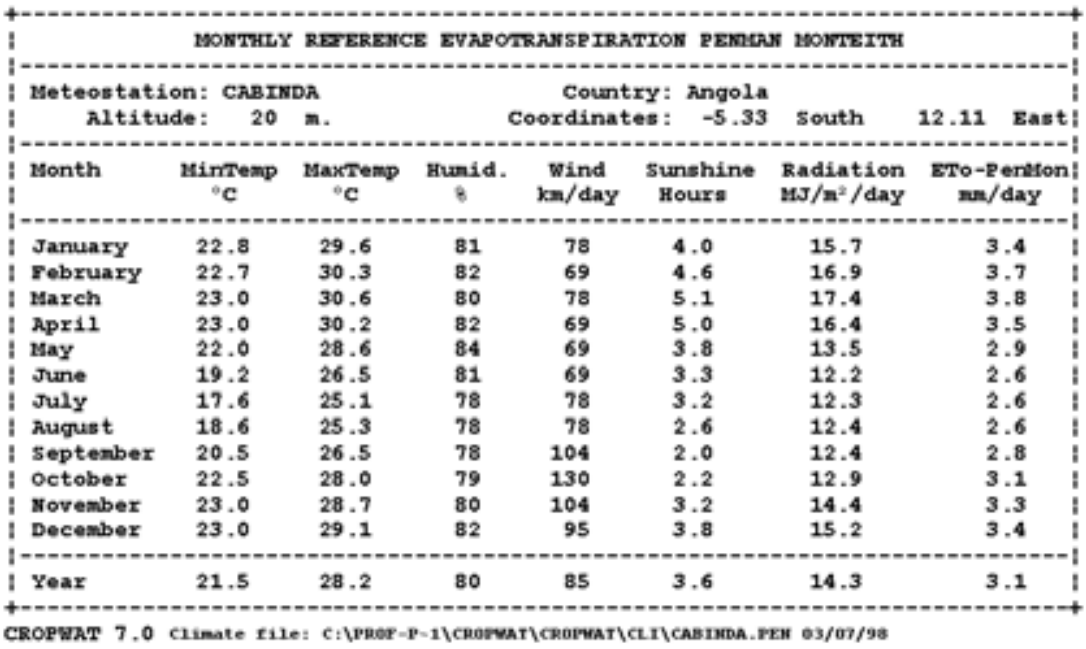
ECUACIÓN 8.18: FLUJO DE CALOR DEL SUELO

donde

G	flujo de calor del suelo [MJ m^{-2} $día^{-1}$],
c_s	capacidad calorífica del suelo [MJ m^{-3} $°C^{-1}$],
T_i	temperatura del aire en el tiempo i [°C],
T_{i-1}	temperatura del aire en el tiempo i-1 [°C],
Δt	intervalo de tiempo considerado [días],
Δz	profundidad efectiva del suelo [m].

Aunque el cálculo de la ET_0 se puede calcular computacionalmente con las fórmulas vistas, la FAO pone a disposición diferentes recursos para facilitar su cálculo:

- El software CROPWAT:



MONTHLY REFERENCE EVAPOTRANSPIRATION PENMAN MONTEITH

Meteostation: CABINDA Country: Angola
Altitude: 20 m. Coordinates: -5.33 South 12.11 East

Month	MinTemp °C	MaxTemp °C	Humid. %	Wind km/day	Sunshine Hours	Radiation MJ/m²/day	ETo-PenMon mm/day
January	22.8	29.6	81	78	4.0	15.7	3.4
February	22.7	30.3	82	69	4.6	16.9	3.7
March	23.0	30.6	80	78	5.1	17.4	3.8
April	23.0	30.2	82	69	5.0	16.4	3.5
May	22.0	28.6	84	69	3.8	13.5	2.9
June	19.2	26.5	81	69	3.3	12.2	2.6
July	17.6	25.1	78	78	3.2	12.3	2.6
August	18.6	25.3	78	78	2.6	12.4	2.6
September	20.5	26.5	78	104	2.0	12.4	2.8
October	22.5	28.0	79	130	2.2	12.9	3.1
November	23.0	28.7	80	104	3.2	14.4	3.3
December	23.0	29.1	82	95	3.8	15.2	3.4
Year	21.5	28.2	80	85	3.6	14.3	3.1

CROPWAT 7.0 Climate file: C:\PROF-P-1\CROPWAT\CROPWAT\CLI\CABINDA.PEN 03/07/98

FOTO 8.3: CAPTURA DEL PROGRAMA CROPWAT

- Una plantilla de cálculo con tablas de generadas:

Parámetros					
T_{max}		°C			
T_{min}		°C	$T_{media} = (T_{max}+T_{min})/2$		°C
T_{media}		°C	Δ (Cuadro 2.4 del Anexo 2)		kPa °C^{-1}
Altitud		m	γ (Cuadro 2.2 del Anexo 2)		kPa °C^{-1}
u_2		M s^{-1}	$(1 + 0{,}34\ u_2)$		
			$\Delta\ /\ [\Delta\ +\ \gamma\ (1 + 0{,}34\ u_2)]$		
			$\gamma\ /\ [\Delta\ +\ \gamma\ (1 + 0{,}34\ u_2)]$		
			$[\ 900\ /\ (T_{media}\ +\ 273)\ ]\ u_2$		

Déficit de presión de vapor					
T_{max}		°C	$e°(T_{max})$ (Cuadro 2.3)		kPa
T_{min}		°C	$e°(T_{min})$ (Cuadro 2.3)		kPa
Presión de saturación de vapor $e_s = [(e°(T_{max}) + e°(T_{min})]/2$					kPa
	e_a derivada de la temperatura del punto de rocío:				
$T_{rocío}$		°C	$e_a = e°(T_{rocío})$ (Cuadro 2.3)		kPa
O bien	**e_a derivada de la humedad relativa máxima y mínima:**				
HR_{max}		%	$e°(T_{min})\ HR_{max}/100$		kPa
HR_{min}		%	$e°(T_{max})\ HR_{min}/100$		kPa
			e_a: (promedio)		kPa
O bien	**e_a derivada de la humedad relativa máxima:** (recomendada si hay errores en HR_{min})				
HR_{max}		%	$e_a = e°(T_{min})\ HR_{max}/100$		kPa
O bien	**e_a derivada de la humedad relativa promedio:** (menos recomendada debido a la no linearidad)				
HR_{media}		%	$e_a = e_s\ HR_{media}/100$		kPa
Déficit de presión de vapor			**$(e_s - e_a)$**		kPa

Radiación

Latitud		°			
Día			R_a (Cuadro 2.6)		MJ m^{-2} $día^{-1}$
Mes			N (Cuadro 2.7)		Horas
n		horas	n/N		
Si no hay datos disponibles de R_s: $R_s = (0{,}25 + 0{,}50\ n/N)\ R_a$					MJ m^{-2} $día^{-1}$
$R_{so} = [0{,}75 + 2\ (Altitud)/\ 100\ 000]\ R_a$					MJ m^{-2} $día^{-1}$
R_s / R_{so}					
$R_{ns} = 0{,}77\ R_s$					MJ m^{-2} $día^{-1}$
T_{max}			$\sigma\ T_{max},K^4$ (Cuadro 2.8)		MJ m^{-2} $día^{-1}$
T_{min}			$\sigma\ T_{min},K^4$ (Cuadro 2.8)		MJ m^{-2} $día^{-1}$
$(\sigma T_{max},K^4 + \sigma T_{min},K^4)/2$					MJ m^{-2} $día^{-1}$
e_a		kPa	$(0{,}34 - 0{,}14\ \sqrt{e_a})$		
R_s/R_{so}			$(1{,}35\ R_s/R_{so} - 0{,}35)$		
$R_{nl} = (\sigma T_{max},K^4 + \sigma T_{min},K^4)/2\ \ (0{,}34 - 0{,}14\ \sqrt{e_a})\ \ (1{,}35\ R_s/R_{so} - 0{,}35)$					MJ m^{-2} $día^{-1}$
$R_n = R_{ns} - R_{nl}$					MJ m^{-2} $día^{-1}$
T_{mes}		°C	$G_{día}$ (asumir)	0	MJ m^{-2} $día^{-1}$
T_{mes-1}		°C	$G_{mes} = 0{,}14\ (T_{mes} - T_{mes-1})$		MJ m^{-2} $día^{-1}$
$R_n - G$					MJ m^{-2} $día^{-1}$
$0{,}408\ (R_n - G)$					mm

Evapotranspiración de referencia del pasto

$\left[\frac{\Delta}{\Delta + \gamma\ (1 + 0{,}34\ u_2)}\right]\ [0{,}408\ \ (R_n - G)]$		mm $día^{-1}$
$\left[\frac{\gamma}{\Delta + \gamma\ (1 + 0{,}34\ u_2)}\right]\left[\frac{900}{T + 273}\right]\ u_2\ [(e_s - e_a)]$		mm $día^{-1}$
$ET_o = \frac{0{,}408\ \Delta\ (R_n - G) + \gamma\ \frac{900}{T + 273}\ u_2\ (e_s - e_a)}{\Delta + \gamma\ (1 + 0{,}34\ u_2)}$		mm $día^{-1}$

FIGURA 8.7: PLANTILLA PARA EL CÁLCULO DE LA EVAPOTRANSPIRACIÓN DE REFERENCIA

Donde las fórmulas vistas anteriormente se pueden estimar partiendo de los valores atmosféricos de acuerdo con las siguientes tablas:

Pendiente de la curva de presión de vapor (Δ) para diferentes temperaturas (T)

$$\Delta = \frac{4098 * \left[0{,}6108 * \exp\left(\frac{17.27 * T}{T + 237{,}3} \right) \right]}{(T + 237{,}3)^2} \qquad \text{(Ec. 13)}$$

T °C	Δ kPa/°C	T °C	Δ kPa/°C	T °C	Δ kPa/°C	T °C	Δ kPa/°C
1,0	0,047	13,0	0,098	25,0	0,189	37,0	0,342
1,5	0,049	13,5	0,101	25,5	0,194	37,5	0,350
2,0	0,050	14,0	0,104	26,0	0,199	38,0	0,358
2,5	0,052	14,5	0,107	26,5	0,204	38,5	0,367
3,0	0,054	15,0	0,110	27,0	0,209	39,0	0,375
3,5	0,055	15,5	0,113	27,5	0,215	39,5	0,384
4,0	0,057	16,0	0,116	28,0	0,220	40,0	0,393
4,5	0,059	16,5	0,119	28,5	0,226	40,5	0,402
5,0	0,061	17,0	0,123	29,0	0,231	41,0	0,412
5,5	0,063	17,5	0,126	29,5	0,237	41,5	0,421
6,0	0,065	18,0	0,130	30,0	0,243	42,0	0,431
6,5	0,067	18,5	0,133	30,5	0,249	42,5	0,441
7,0	0,069	19,0	0,137	31,0	0,256	43,0	0,451
7,5	0,071	19,5	0,141	31,5	0,262	43,5	0,461
8,0	0,073	20,0	0,145	32,0	0,269	44,0	0,471
8,5	0,075	20,5	0,149	32,5	0,275	44,5	0,482
9,0	0,078	21,0	0,153	33,0	0,282	45,0	0,493
9,5	0,080	21,5	0,157	33,5	0,289	45,5	0,504
10,0	0,082	22,0	0,161	34,0	0,296	46,0	0,515
10,5	0,085	22,5	0,165	34,5	0,303	46,5	0,526
11,0	0,087	23,0	0,170	35,0	0,311	47,0	0,538
11,5	0,090	23,5	0,174	35,5	0,318	47,5	0,550
12,0	0,092	24,0	0,179	36,0	0,326	48,0	0,562
12,5	0,095	24,5	0,184	36,5	0,334	48,5	0,574

Constante psicrométrica (γ) para diferentes altitudes (z)

$$\gamma = \frac{c_p P}{\varepsilon \lambda} = 0{,}665 * 10^{-3} \qquad \text{(Ec. 8)}$$

z (m)	γ kPa °C^{-1}	z (m)	γ kPa/°C	z (m)	γ kPa/°C	z (m)	γ kPa/°C
0	0,067	1 000	0,060	2 000	0,053	3 000	0,047
100	0,067	1 100	0,059	2 100	0,052	3 100	0,046
200	0,066	1 200	0,058	2 200	0,052	3 200	0,046
300	0,065	1 300	0,058	2 300	0,051	3 300	0,045
400	0,064	1 400	0,057	2 400	0,051	3 400	0,045
500	0,064	1 500	0,056	2 500	0,050	3 500	0,044
600	0,063	1 600	0,056	2 600	0,049	3 600	0,043
700	0,062	1 700	0,055	2 700	0,049	3 700	0,043
800	0,061	1 800	0,054	2 800	0,048	3 800	0,042
900	0,061	1 900	0,054	2 900	0,047	3 900	0,042
1 000	0,060	2 000	0,053	3 000	0,047	4 000	0,041

basada en λ= 2,45 MJ kg^{-1} a 20°C.

Presión de saturación de vapor ($e^o(T)$) para diferentes temperaturas (T)

$$e^o(T) = 0{,}6108 * \exp\left[\frac{17{,}27 * T}{T + 237{,}3}\right] \qquad \text{(Ec.11)}$$

T °C	e°(T) kPa	T °C	e°(T) kPa	T °C	e°(T) kPa	T °C	e°(T) kPa
1,0	0,657	13,0	1,498	25,0	3,168	37,0	6,275
1,5	0,681	13,5	1,547	25,5	3,263	37,5	6,448
2,0	0,706	14,0	1,599	26,0	3,361	38,0	6,625
2,5	0,731	14,5	1,651	26,5	3,462	38,5	6,806
3,0	0,758	15,0	1,705	27,0	3,565	39,0	6,991
3,5	0,785	15,5	1,761	27,5	3,671	39,5	7,181
4,0	0,813	16,0	1,818	28,0	3,780	40,0	7,376
4,5	0,842	16,5	1,877	28,5	3,891	40,5	7,574
5,0	0,872	17,0	1,938	29,0	4,006	41,0	7,778
5,5	0,903	17,5	2,000	29,5	4,123	41,5	7,986
6,0	0,935	18,0	2,064	30,0	4,243	42,0	8,199
6,5	0,968	18,5	2,130	30,5	4,366	42,5	8,417
7,0	1,002	19,0	2,197	31,0	4,493	43,0	8,640
7,5	1,037	19,5	2,267	31,5	4,622	43,5	8,867
8,0	1,073	20,0	2,338	32,0	4,755	44,0	9,101
8,5	1,110	20,5	2,412	32,5	4,891	44,5	9,339
9,0	1,148	21,0	2,487	33,0	5,030	45,0	9,582
9,5	1,187	21,5	2,564	33,5	5,173	45,5	9,832
10,0	1,228	22,0	2,644	34,0	5,319	46,0	10,086
10,5	1,270	22,5	2,726	34,5	5,469	46,5	10,347
11,0	1,313	23,0	2,809	35,0	5,623	47,0	10,613
11,5	1,357	23,5	2,896	35,5	5,780	47,5	10,885
12,0	1,403	24,0	2,984	36,0	5,941	48,0	11,163
12,5	1,449	24,5	3,075	36,5	6,106	48,5	11,447

Radiación extraterrestre diaria (R_a) para diferentes latitudes para el día 15vo del mes[1]

$$R_a = \frac{24 * 60}{\pi} G_{sc} d_r \left[\omega_s \sin(\varphi)\sin(\delta) + \cos(\varphi)\cos(\delta)\sin(\omega)\right] \qquad \text{(Ec. 21)}$$

(valores en MJ m^{-2} $día^{-1}$)[2]

Hemisferio Norte												Lat. grad.	Hemisferio Sur											
Ene.	Feb.	Marzo	Abril	Mayo	Junio	Julio	Ago.	Sep.	Oct.	Nov.	Dic		Ene.	Feb.	Marzo	Abril	Mayo	Junio	Julio	Ago.	Sep.	Oct.	Nov.	Dic
0,0	2,6	10,4	23,0	35,2	42,5	39,4	28,0	14,9	4,9	0,1	0,0	70	41,4	28,6	15,8	4,9	0,2	0,0	0,0	2,2	10,7	23,5	37,3	45,3
0,1	3,7	11,7	23,9	35,3	42,0	38,9	28,6	16,1	6,0	0,7	0,0	68	41,0	29,3	16,9	6,0	0,8	0,0	0,0	3,2	11,9	24,4	37,4	44,7
0,6	4,8	12,9	24,8	35,6	41,4	38,8	29,3	17,3	7,2	1,5	0,1	66	40,9	30,0	18,1	7,2	1,5	0,1	0,5	4,2	13,1	25,4	37,6	44,1
1,4	5,9	14,1	25,8	35,9	41,2	38,8	30,0	18,4	8,5	2,4	0,6	64	41,0	30,8	19,3	8,4	2,4	0,6	1,2	5,3	14,4	26,3	38,0	43,9
2,3	7,1	15,4	26,6	36,3	41,2	39,0	30,6	19,5	9,7	3,4	1,3	62	41,2	31,5	20,4	9,6	3,4	1,2	2,0	6,4	15,5	27,2	38,3	43,9
3,3	8,3	16,6	27,5	36,6	41,2	39,2	31,3	20,6	10,9	4,4	2,2	60	41,5	32,3	21,5	10,8	4,4	2,0	2,9	7,6	16,7	28,1	38,7	43,9
4,3	9,6	17,7	28,4	37,0	41,3	39,4	32,0	21,7	12,1	5,5	3,1	58	41,7	33,0	22,6	12,0	5,5	2,9	3,9	8,7	17,9	28,9	39,1	44,0
5,4	10,8	18,9	29,2	37,4	41,4	39,6	32,6	22,7	13,3	6,7	4,2	56	42,0	33,7	23,6	13,2	6,6	3,9	4,9	9,9	19,0	29,8	39,5	44,1
6,5	12,0	20,0	30,0	37,8	41,5	39,8	33,2	23,7	14,5	7,8	5,2	54	42,2	34,3	24,6	14,4	7,7	4,9	6,0	11,1	20,1	30,6	39,9	44,3
7,7	13,2	21,1	30,8	38,2	41,6	40,1	33,8	24,7	15,7	9,0	6,4	52	42,5	35,0	25,6	15,6	8,8	6,0	7,1	12,2	21,2	31,4	40,2	44,4
8,9	14,4	22,2	31,5	38,5	41,7	40,2	34,4	25,7	16,9	10,2	7,5	50	42,7	35,6	26,6	16,7	10,0	7,1	8,2	13,4	22,2	32,1	40,6	44,5
10,1	15,7	23,3	32,2	33,8	41,8	40,4	34,9	26,6	18,1	11,4	8,7	48	42,9	36,2	27,5	17,9	11,1	8,2	9,3	14,6	23,3	32,8	40,9	44,5
113	16,9	24,3	32,9	39,1	41,9	40,6	35,4	27,5	19,2	12,6	9,9	46	43,0	36,7	28,4	19,0	12,3	9,3	10,4	15,7	24,3	33,5	41,1	44,6
12,5	18,0	25,3	33,5	39,3	41,9	40,7	35,9	28,4	20,3	13,9	11,1	44	43,2	37,2	29,3	20,1	13,5	10,5	11,6	16,8	25,2	34,1	41,4	44,6
13,8	19,2	26,3	34,1	39,5	41,9	40,8	36,3	29,2	21,4	15,1	12,4	42	43,3	37,7	30,1	21,2	14,6	11,6	12,8	18,0	26,2	34,7	41,6	44,6
15,0	20,4	27,2	34,7	39,7	41,9	40,8	36,7	30,0	22,5	16,3	13,6	40	43,4	38,1	30,9	22,3	15,8	12,8	13,9	19,1	27,1	35,3	41,8	44,6
16,2	21,5	28,1	35,2	39,9	41,8	40,8	37,0	30,7	23,6	17,5	14,8	38	43,4	38,5	31,7	23,3	16,9	13,9	15,1	20,2	28,0	35,8	41,9	44,5
17,5	22,6	29,0	35,7	40,0	41,7	40,8	37,4	31,5	24,6	18,7	16,1	36	43,4	38,9	32,4	24,3	18,1	15,1	16,2	21,2	28,8	36,3	42,0	44,4
18,7	23,7	29,9	36,1	40,0	41,6	40,8	37,6	32,1	25,6	19,9	17,3	34	43,4	39,2	33,0	25,3	19,2	16,2	17,4	22,3	29,6	36,7	42,0	44,3
19,9	24,8	30,7	35,5	40,0	41,4	40,7	37,9	32,8	26,6	21,1	18,5	32	43,3	39,4	33,7	26,3	20,3	17,4	18,5	23,3	30,4	37,1	42,0	44,1
21,1	25,8	31,4	36,8	40,0	41,2	40,6	38,0	33,4	27,6	22,2	19,8	30	43,1	39,6	34,3	27,2	21,4	18,5	19,6	24,3	31,1	37,5	42,0	43,9
22,3	26,8	32,2	37,1	40,0	40,9	40,4	38,2	33,9	28,5	23,3	21,0	28	43,0	39,8	34,8	28,1	22,5	19,7	20,7	25,3	31,8	37,8	41,9	43,6
23,4	27,8	32,8	37,4	39,9	40,6	40,2	38,3	34,5	29,3	24,5	22,2	26	42,8	39,9	35,3	29,0	23,5	20,8	21,8	26,3	32,5	38,0	41,8	43,3
24,6	28,8	33,5	37,6	39,7	40,3	39,9	38,3	34,9	30,2	25,5	23,3	24	42,5	40,0	35,8	29,8	24,6	21,9	22,9	27,2	33,1	38,3	41,7	43,0
25,7	29,7	34,1	37,8	39,5	40,0	39,6	38,4	35,4	31,0	26,6	24,5	22	42,2	40,1	36,2	30,6	25,6	23,0	24,0	28,1	33,7	38,4	41,4	42,6
26,8	30,6	34,7	37,9	39,3	39,5	39,3	38,3	35,8	31,8	27,7	25,6	20	41,9	40,0	36,6	31,3	26,6	24,1	25,0	28,9	34,2	38,6	41,2	42,1
27,9	31,5	35,2	38,0	39,0	39,1	38,9	38,2	36,1	32,5	28,7	26,8	18	41,5	40,0	37,0	32,1	27,5	25,1	26,0	29,8	34,7	38,7	40,9	41,7
28,9	32,3	35,7	38,1	38,7	38,6	38,5	38,1	36,4	33,2	29,6	27,9	16	41,1	39,9	37,2	32,8	28,5	26,2	27,0	30,6	35,2	38,7	40,6	41,2
29,9	33,1	36,1	38,1	38,4	38,1	38,1	38,0	36,7	33,9	30,6	28,9	14	40,6	39,7	37,5	33,4	29,4	27,2	27,9	31,3	35,6	38,7	40,2	40,6
30,9	33,8	36,5	38,0	38,0	37,6	37,6	37,8	36,9	34,5	31,5	30,0	12	40,1	39,6	37,7	34,0	30,2	28,1	28,9	32,1	36,0	38,6	39,8	40,0
31,9	34,5	36,9	37,9	37,6	37,0	37,1	37,5	37,1	35,1	32,4	31,0	10	39,5	39,3	37,8	34,6	31,1	29,1	29,8	32,8	36,3	38,5	39,3	39,4
32,8	35,2	37,2	37,8	37,1	36,3	36,5	37,2	37,2	35,6	33,3	32,0	8	38,9	39,0	37,9	35,1	31,9	30,0	30,7	33,4	36,6	38,4	38,8	38,7
33,7	35,8	37,4	37,6	36,6	35,7	35,9	36,9	37,3	36,1	34,1	32,9	6	38,3	38,7	38,0	35,6	32,7	30,9	31,5	34,0	36,8	38,2	38,2	38,0
34,6	36,4	37,6	37,4	36,0	35,0	35,3	36,5	37,3	36,6	34,9	33,9	4	37,6	38,3	38,0	36,0	33,4	31,8	32,3	34,6	37,0	38,0	37,6	37,2
35,4	37,0	37,8	37,1	35,4	34,2	34,6	36,1	37,3	37,0	35,6	34,8	2	36,9	37,9	38,0	36,4	34,1	32,6	33,1	35,2	37,1	37,7	37,0	36,4
36,2	37,5	37,9	36,8	34,8	33,4	33,9	35,7	37,2	37,4	36,3	35,6	0	36,2	37,5	37,9	36,8	34,8	33,4	33,9	35,7	37,2	37,4	36,3	35,6

[1] Los valores de R_g durante el día15^{avo} del mes, proveen una buena estimación (error <1%) de R_a promediada de todos los días del mes. Solamente en casos de latitudes muy elevadas (mayores a 55° N o S) y durante los meses invernales, las desviaciones podrían ser mayores al 1 %.

[2] Los valores pueden ser convertidos a sus equivalentes en mm día-1 si se dividen por Lambda = 2,45.

Insolación máxima diaria (N) para diferentes latitudes para el día 15vo del mes[1]

$$N = \frac{24}{\pi}\,\omega_s \qquad \text{(Ec. 34)}$$

Hemisferio Norte												Lat. grad.	Hemisferio Sur											
Ene.	Feb.	Marzo	Abril	Mayo	Junio	Julio	Ago.	Sep.	Oct.	Nov.	Dic.		Ene.	Feb.	Marzo	Abril	Mayo	Junio	Julio	Ago.	Sep.	Oct.	Nov.	Dic.
0,0	6,6	11,0	15,6	21,3	24,0	24,0	17,6	12,8	8,3	2,3	0,0	70	24,0	17,4	13,0	8,4	2,7	0,0	0,0	6,4	11,2	15,7	21,7	24,0
1,1	7,3	11,1	15,3	19,7	24,0	22,3	17,0	12,7	8,7	4,1	0,0	68	21,9	16,7	12,9	8,7	4,3	0,0	1,7	7,0	11,3	15,3	19,9	24,0
3,9	7,8	11,2	14,9	18,7	22,0	20,3	16,4	12,7	9,0	5,2	1,9	66	20,1	16,2	12,8	9,1	5,3	2,0	3,7	7,6	11,3	15,0	18,8	22,1
5,0	8,2	11,2	14,7	17,9	20,3	19,2	16,0	12,6	9,3	6,0	3,7	64	19,0	15,8	12,8	9,3	6,1	3,7	4,8	8,0	11,4	14,7	18,0	20,3
5,7	8,5	11,3	14,4	17,3	19,2	18,4	15,7	12,6	9,5	6,6	4,8	62	18,3	15,5	12,7	9,6	6,7	4,8	5,6	8,3	11,4	14,5	17,4	19,2
6,4	8,8	11,4	14,2	16,8	18,4	17,7	15,3	12,5	9,7	7,1	5,6	60	17,6	15,2	12,6	9,8	7,2	5,6	6,3	8,7	11,5	14,3	16,9	18,4
6,9	9,1	11,4	14,1	16,4	17,8	17,2	15,1	12,5	9,9	7,5	6,2	58	17,1	14,9	12,6	9,9	7,6	6,2	6,8	8,9	11,5	14,1	16,5	17,8
7,3	9,3	11,5	13,9	16,0	17,3	16,8	14,8	12,4	10,1	7,9	6,7	56	16,7	14,7	12,5	10,1	8,0	6,7	7,2	9,2	11,6	13,9	16,1	17,3
7,7	9,5	11,5	13,8	15,7	16,8	16,4	14,6	12,4	10,2	8,2	7,1	54	16,3	14,5	12,5	10,2	8,3	7,2	7,6	9,4	11,6	13,8	15,8	16,9
8,0	9,7	11,5	13,6	15,4	16,5	16,0	14,4	12,4	10,3	8,5	7,5	52	16,0	14,3	12,5	10,4	8,6	7,5	8,0	9,6	11,6	13,7	15,5	16,5
8,3	9,8	11,6	13,5	15,2	16,1	15,7	14,3	12,3	10,4	8,7	7,9	50	15,7	14,2	12,4	10,5	8,8	7,9	8,3	9,7	11,7	13,6	15,3	16,1
8,6	10,0	11,6	13,4	15,0	15,8	15,5	14,1	12,3	10,6	9,0	8,2	48	15,4	14,0	12,4	10,6	9,0	8,2	8,5	9,9	11,7	13,4	15,0	15,8
8,8	10,1	11,6	13,3	14,8	15,5	15,2	14,0	12,3	10,7	9,2	8,5	46	15,2	13,9	12,4	10,7	9,2	8,5	8,8	10,0	11,7	13,3	14,8	15,5
9,1	10,3	11,6	13,2	14,6	15,3	15,0	13,8	12,3	10,7	9,4	8,7	44	14,9	13,7	12,4	10,8	9,4	8,7	9,0	10,2	11,7	13,3	14,6	15,3
9,3	10,4	11,7	13,2	14,4	15,0	14,8	13,7	12,3	10,8	9,6	9,0	42	14,7	13,6	12,3	10,8	9,6	9,0	9,2	10,3	11,7	13,2	14,4	15,0
9,5	10,5	11,7	13,1	14,2	14,8	14,6	13,6	12,2	10,9	9,7	9,2	40	14,5	13,5	12,3	10,9	9,8	9,2	9,4	10,4	11,8	13,1	14,3	14,8
9,6	10,6	11,7	13,0	14,1	14,6	14,4	13,5	12,2	11,0	9,9	9,4	38	14,4	13,4	12,3	11,0	9,9	9,4	9,6	10,5	11,8	13,0	14,1	14,6
9,8	10,7	11,7	12,9	13,9	14,4	14,2	13,4	12,2	11,1	10,1	9,6	36	14,2	13,3	12,3	11,1	10,1	9,6	9,8	10,6	11,8	12,9	13,9	14,4
10,0	10,8	11,8	12,9	13,8	14,3	14,1	13,3	12,2	11,1	10,2	9,7	34	14,0	13,2	12,2	11,1	10,2	9,7	9,9	10,7	11,8	12,9	13,8	14,3
10,1	10,9	11,8	12,8	13,6	14,1	13,9	13,2	12,2	11,2	10,3	9,9	32	13,9	13,1	12,2	11,2	10,4	9,9	10,1	10,8	11,8	12,8	13,7	14,1
10,3	11,0	11,8	12,7	13,5	13,9	13,8	13,1	12,2	11,3	10,5	10,1	30	13,7	13,0	12,2	11,3	10,5	10,1	10,2	10,9	11,8	12,7	13,5	13,9
10,4	11,0	11,8	12,7	13,4	13,8	13,6	13,0	12,2	11,3	10,6	10,2	28	13,6	13,0	12,2	11,3	10,6	10,2	10,4	11,0	11,8	12,7	13,4	13,8
10,5	11,1	11,8	12,6	13,3	13,6	13,5	12,9	12,1	11,4	10,7	10,4	26	13,5	12,9	12,2	11,4	10,7	10,4	10,5	11,1	11,9	12,6	13,3	13,6
10,7	11,2	11,8	12,6	13,2	13,5	13,3	12,8	12,1	11,4	10,8	10,5	24	13,3	12,8	12,2	11,4	10,8	10,5	10,7	11,2	11,9	12,6	13,2	13,5
10,8	11,3	11,9	12,5	13,1	13,3	13,2	12,8	12,1	11,5	10,9	10,7	22	13,2	12,7	12,1	11,5	10,9	10,7	10,8	11,2	11,9	12,5	13,1	13,3
10,9	11,3	11,9	12,5	12,9	13,2	13,1	12,7	12,1	11,5	11,0	10,8	20	13,1	12,7	12,1	11,5	11,1	10,8	10,9	11,3	11,9	12,5	13,0	13,2
11,0	11,4	11,9	12,4	12,8	13,1	13,0	12,6	12,1	11,6	11,1	10,9	18	13,0	12,6	12,1	11,6	11,2	10,9	11,0	11,4	11,9	12,4	12,9	13,1
11,1	11,5	11,9	12,4	12,7	12,9	12,9	12,5	12,1	11,6	11,2	11,1	16	12,9	12,5	12,1	11,6	11,3	11,1	11,1	11,5	11,9	12,4	12,8	12,9
11,3	11,6	11,9	12,3	12,6	12,8	12,8	12,5	12,1	11,7	11,3	11,2	14	12,7	12,4	12,1	11,7	11,4	11,2	11,2	11,5	11,9	12,3	12,7	12,8
11,4	11,6	11,9	12,3	12,6	12,7	12,6	12,4	12,1	11,7	11,4	11,3	12	12,6	12,4	12,1	11,7	11,4	11,3	11,4	11,6	11,9	12,3	12,6	12,7
11,5	11,7	11,9	12,2	12,5	12,6	12,5	12,3	12,1	11,8	11,5	11,4	10	12,5	12,3	12,1	11,8	11,5	11,4	11,5	11,7	11,9	12,2	12,5	12,6
11,6	11,7	11,9	12,2	12,4	12,5	12,4	12,3	12,0	11,8	11,6	11,5	8	12,4	12,3	12,1	11,8	11,6	11,5	11,6	11,7	12,0	12,2	12,4	12,5
11,7	11,8	12,0	12,1	12,3	12,3	12,3	12,2	12,0	11,9	11,7	11,7	6	12,3	12,2	12,0	11,9	11,7	11,7	11,7	11,8	12,0	12,1	12,3	12,3
11,8	11,9	12,0	12,1	12,2	12,2	12,2	12,1	12,0	11,9	11,8	11,8	4	12,2	12,1	12,0	11,9	11,8	11,8	11,8	11,9	12,0	12,1	12,2	12,2
11,9	11,9	12,0	12,0	12,1	12,1	12,1	12,1	12,0	12,0	11,9	11,9	2	12,1	12,1	12,0	12,0	11,9	11,9	11,9	11,9	12,0	12,0	12,1	12,1
12,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0

[1] Los valores de N durante el día15avo del mes, proveen una buena estimación (error <1%) de N promediada sobre todos los días del mes. Solamente en casos de latitudes muy elevadas (mayores a 55° N o S) y durante los meses invernales, las desviaciones podrían ser mayores al 1 %.

σTK4 (de acuerdo a la ley de Stefan-Boltzmann) para diferentes temperaturas (T)

Con σ = 4,903 10^{-9} MJ K^{-4} m^{-2} $día^{-1}$
y T_K = T[°C] + 273,16

T (°C)	σTK⁴ (MJ m⁻² d⁻¹)	T (°C)	σTK⁴ (MJ m⁻² d⁻¹)	T (°C)	σTK⁴ (MJ m⁻² d⁻¹)
1,0	27,70	17,0	34,75	33,0	43,08
1,5	27,90	17,5	34,99	33,5	43,36
2,0	28,11	18,0	35,24	34,0	43,64
2,5	28,31	18,5	35,48	34,5	43,93
3,0	28,52	19,0	35,72	35,0	44,21
3,5	28,72	19,5	35,97	35,5	44,50
4,0	28,93	20,0	36,21	36,0	44,79
4,5	29,14	20,5	36,46	36,5	45,08
5,0	29,35	21,0	36,71	37,0	45,37
5,5	29,56	21,5	36,96	37,5	45,67
6,0	29,78	22,0	37,21	38,0	45,96
6,5	29,99	22,5	37,47	38,5	46,26
7,0	30,21	23,0	37,72	39,0	46,56
7,5	30,42	23,5	37,98	39,5	46,85
8,0	30,64	24,0	38,23	40,0	47,15
8,5	30,86	24,5	38,49	40,5	47,46
9,0	31,08	25,0	38,75	41,0	47,76
9,5	31,30	25,5	39,01	41,5	48,06
10,0	31,52	26,0	39,27	42,0	48,37
10,5	31,74	26,5	39,53	42,5	48,68
11,0	31,97	27,0	39,80	43,0	48,99
11,5	32,19	27,5	40,06	43,5	49,30
12,0	32,42	28,0	40,33	44,0	49,61
12,5	32,65	28,5	40,60	44,5	49,92
13,0	32,88	29,0	40,87	45,0	50,24
13,5	33,11	29,5	41,14	45,5	50,56
14,0	33,34	30,0	41,41	46,0	50,87
14,5	33,57	30,5	41,69	46,5	51,19
15,0	33,81	31,0	41,96	47,0	51,51
15,5	34,04	31,5	42,24	47,5	51,84
16,0	34,28	32,0	42,52	48,0	52,16
16,5	34,52	32,5	42,80	48,5	52,49

8.4. SiGAM

El Sistema de Gestión Agrícola en Movilidad, con el acrónimo SiGAM 1213, es un proyecto presentado en el departamento de I+D de Tragsatec como una rama de la tesis Sysreg, concretamente el módulo RegDuino.

El motivo de ramificar la tesis en un proyecto I+D y no presentar la propia tesis es preservar la independencia del proyecto Sysreg, de manera que ambos proyectos se complementen pero sean autónomos.

8.4.1. Descripción técnica

SiGAM es un sistema de gestión agrícola que pretende optimizar el trabajo del agricultor minifundista. Para ello pone al alcance de éste la tecnología más puntera e innovadora, formada por componentes físicos altamente extensibles y por dispositivos móviles de última generación.

La idea básica es tener un hardware capaz de gestionar parcelas agrícolas y poder acceder a este dispositivo a través del móvil.

El sistema está formado por dos componentes: un componente hardware y otro software:

- En primer lugar tenemos una placa Arduino a la que se conectan diferentes sensores y actuadores:
    - Sensores: Termómetros, humedímetros, barómetros, anemómetros, pluviómetros, sensores de luz UV, microcámaras, micrófonos.
    - Actuadores: Relés, servomotores, transistores, puentes H…
- En la parte software controlaremos el dispositivo con un *smartphone* con sistema operativo Android. Nuestro objetivo es conseguir un escenario en que el agricultor llegue a su parcela y, utilizando el móvil, sea capaz de gestionar la mayor parte de variables, como por ejemplo el riego.

  Para desarrollar el software se hará uso del kit de desarrollo de accesorios de Google, el Android ADK, utilizando la placa Arduino Mega ADK. Así pues se desarrollará un software para Android que al conectar la placa realice las siguientes funciones:

- Monitorización en tiempo real: Una vez conectados a la placa podremos obtener los datos de sensores y actuadores, así como observar el calendario de riegos programados.
- Activación/desactivación de los actuadores: Con nuestro móvil podremos actuar directamente sobre la parcela, por ejemplo encendiendo válvulas de riego.
- Programación *offline*: Utilizando el móvil seremos capaces de definir un calendario de riegos y subirlo a la placa.

8.4.2. Presupuesto

Articulo	Precio	Unidades	Total
Arduino Mega ADK	59	2	118
Arduino Ethernet	39,9	1	39,9
Arduino UNO	22	2	44
Arduino Mega	41	1	41
Arduino Cellular Shield	77,9	1	77,9
Arduino USB Host Shield	19,9	1	19,9
Conector 6 pines	0,5	5	2,5
Conector 8 pines	0,5	5	2,5
Arduino XBEE Shield (Sin XBEE)	16	2	32
XBEE 2MW Serie 2 Con Antena	23,5	2	47
Arduino Xbee Shield	42,9	2	85,8
Arduino GPS Shield	13,9	1	13,9
Conector 40 pines	2,05	3	6,15
Receptor GPS EM-406A	48	1	48
Lote diodos 25 uds (Color)	7,91	3	23,73
Rele 5V	2,3	3	6,9
Rele de estado solido 8A	4,4	3	13,2
Transistor MOSFET	1,9	3	5,7
Puente H L298N	2,6	3	7,8
Resistencias (5 uds por valor)	9,35	3	28,05
Caja 2AA con conversos 5V	9,6	1	9,6
Interruptor	0,95	3	2,85
Pulsador 12mm	0,5	3	1,5
Sensor humedad HH10D	8,9	3	26,7
Sensor barometrico y de Tª BMP085	16,9	3	50,7
Micro camara TTL (640x480)	45	3	135
Sensor de temperatura DS18B20	4,9	3	14,7
Camara CMOS IR (500x582)	29,9	3	89,7
Sensor de humedad y Tª SHT15	32,9	3	98,7

Arduino 4Display Shiel	63,2	2	126,4
Soldador	8,9	1	8,9
Base soldador	10,5	1	10,5
Esponja limpiadora	9,45	1	9,45
Bobina de Estaño	7,4	1	7,4
Maya para desoldar	14,1	1	14,1
Pinza con ventosa	22,8	1	22,8
Multimetro	39,9	1	39,9
Set de cables M/M	27,9	2	55,8
Bobina de cable de prototipo (Color)	3,19	5	15,95
Caja Arduino	9,9	3	29,7
Placa de prototipo	9	2	18
Mini placa de prototipo	3,25	2	6,5
Placa de prototipo con alimentación	11,4	3	34,2
Separadores (10 uds)	1,9	5	9,5
Tornillos (10 uds)	1,2	5	6
Samsung Galaxy Nexus	519	1	519
Samsung Galaxy Tab 2 3G 10,1"	599	1	599
		TOTAL	2626,48

9. ASPECTOS TECNOLÓGICOS UTILIZADOS

En esta sección se describen otros aspectos tecnológicos utilizados en la tesina.

9.1. Rational Unified Process

Dentro de las metodologías de desarrollo de software encontramos los paradigmas de desarrollo iterativo. Éstos se basan en la idea de completar iterativamente el producto hasta que se desarrolle el sistema deseado. Permiten reducir la repetición del trabajo y dar oportunidad de retrasar las decisiones hasta tener experiencia en el sistema.

Con el desarrollo incremental se consigue:

- Ofrecer a los clientes un producto desde la primera iteración.
- Perfeccionar los requisitos a medida que se ven las entregas.
- Disminuir el riesgo de fracaso, ya que el producto se va conociendo poco a poco.
- Las funcionalidades más importantes se entregan al principio, de esta forma se realizan más pruebas.

Entre este tipo de metodologías destaca RUP (*Rational Unified Process*) cuyas características esenciales son:

- Dirigido por casos de uso: Los casos de uso representan los requisitos del sistema, permiten guiar su diseño, implementación y prueba. Son el elemento integrador.

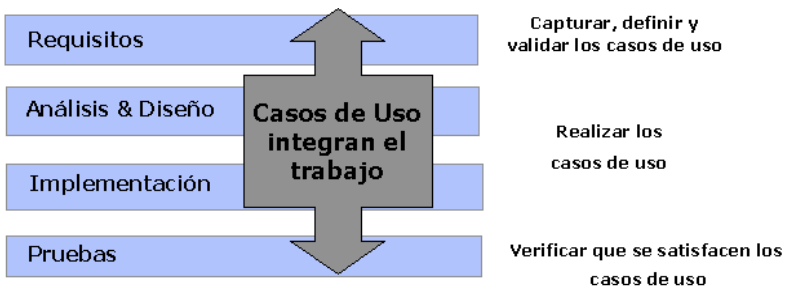


FIGURA 9.1: INTEGRACIÓN DE LOS CASOS DE USO EN RUP

- Centrado en la arquitectura: La estructura del sistema en RUP se consolida en las primeras fases del proyecto. Se busca una buena arquitectura que no se vea afectada ante cambios durante la construcción y el mantenimiento.

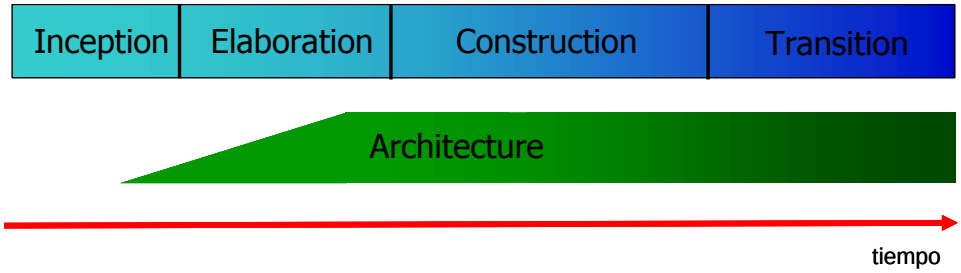


FIGURA 9.2: EVOLUCIÓN DE LA ARQUITECTURA EN RUP

- Proceso iterativo e incremental: Propone una estrategia en la que el trabajo se divida en partes más pequeñas. De esta forma se consigue integrar la arquitectura con los casos de uso poco a poco. Cada iteración estará formada por las fases fundamentales (requisitos, análisis, diseño, implementación y pruebas) y abordará una parte de la funcionalidad total.

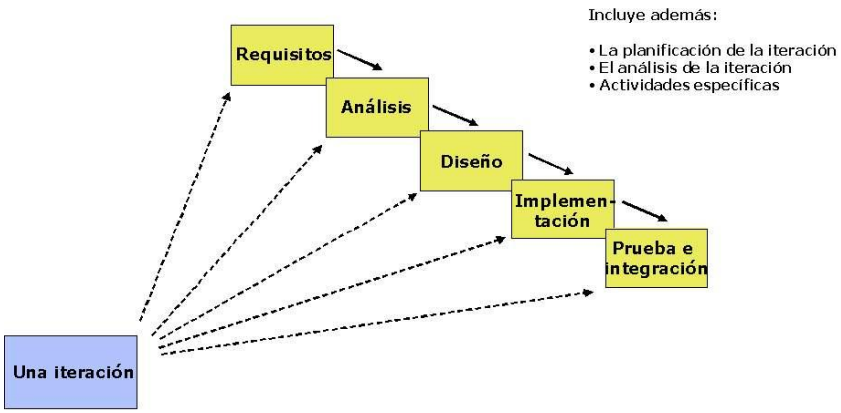


FIGURA 9.3: ITERACIÓN EN RUP

9.2. Subversión

Subversion (SVN) es un sistema de control de versiones libre y de código fuente abierto. Maneja ficheros y directorios a través del tiempo con un árbol de ficheros en un repositorio central (en un sistema de control de versiones centralizado) que recuerda todos los cambios hechos a sus ficheros y directorios. Esto permite recuperar versiones antiguas o examinar el historial de cambios. Además permite acceder al repositorio a través de redes, lo que facilita la colaboración entre personas.

Subversion no es un sistema diseñado únicamente para administrar código fuente y configuraciones software, sino que es un sistema general que puede ser usado para cualquier tipo de ficheros: código fuente, video, audio, texto plano, documentos ofimáticos, etcétera.

Las características que aporta SVN son:

- Versionado de directorios: Implementa un sistema de versionado virtual que sigue los cambios sobre el árbol de directorio, a diferencia de CVS que lleva el historial de ficheros individuales.
- Historial de versiones: Al poseer un versionado a nivel de directorio se puede añadir, borrar, copiar y renombrar ficheros y directorios sin problemas. Cada fichero nuevo añadido empieza con un historial nuevo, limpio y completamente suyo. En CVS al ser un versionado individual estas operaciones no pueden ocurrir sin heredar el historial de fichero anterior.
- Envíos atómicos: Al igual que pasa con las bases de datos relacionales a nivel de transacción, los envíos o se completan o no llegan al repositorio.
- Versionado de propiedades: Cada fichero y directorio tiene un conjunto de propiedades asociadas a él. Estas propiedades también son versionadas a través del tiempo.
- Elección de capas de red: Subversion ofrece una noción abstracta de acceso al repositorio, facilitando la implementación de nuevos mecanismos de red. Un usuario se puede conectar por HTTP, SSH o en un servidor local.
- Manipulación consistente de datos: Subversion expresa las diferencias del fichero usando un algoritmo de diferenciación binario, que funciona idénticamente con ficheros de textos (legibles para humanos) como con ficheros

binarios. Además con este algoritmo se permite que solo viajen por la red las diferencias entre las versiones, es decir, solo se mandan los cambios, no todo el fichero como en CVS.

- Ramificación y etiquetado eficiente: El coste de la ramificación y del etiquetado no es proporcional al tamaño. En SVN el coste es O (1) y en CVS O (n).
- *Hackability*: Está implementado como una colección de bibliotecas compartidas en C, con una API bien definida. Esto hace que sea extremadamente fácil de mantener y de reutilizar.
- Arquitectura cliente-servidor.

Podemos trabajar en SVN mediante línea de comandos o aplicaciones gráficas:

Para importar un repositorio utilizamos comando *svn checkout.*

```
$ svn checkout http://svn.collab.net/repos/svn/trunk subv
A subv/subversion.dsw
A subv/svn_check.dsp
…
Checked out revision 2499.
```

Una vez importado podemos realizar cambios en la copia local y registrarlos en el servidor mediante la orden *svn commit PATH –m "mensaje de registro"*:

```
$ svn commit --message "Corrected number of cheese slices."
Sending sandwich.txt
Transmitting file data .
Committed revision 3.
```

Para regresar a la versión anterior solo tenemos que invocar *svn rever path*, con la opción *–recursive* si queremos deshacer todos los cambios del directorio.

Cuando retomemos el trabajo deberemos actualizar la copia local, para ello *svn update* nos actualizará e informará de los cambios:

```
$ svn update
U foo.c
U bar.c
Updated to revision 2.
```

Pero pueden aparecer conflictos, por ejemplo debido a que se modifican y se suben versiones diferentes. De este problema nos informa Subversion colocando marcas

en los ficheros para mostrar las áreas solapadas. Además para cada conflicto coloca tres ficheros temporales extra en la copia de trabajo local: *filename.mine*, *filename.ROLDREV*, *filename.RNEWREV*. Para solucionarlo podemos fusionar el conflicto a mano, copiar uno de los ficheros temporales sobre su fichero de trabajo o ejecutar *svn revert* para eliminar los cambios locales. Una vez solucionado el conflicto es necesario comunicárselo a Subversión con *svn resolved PATH*, lo que borrará los ficheros temporales y se olvidará de que hubo un conflicto.

```
$ svn update
C sandwich.txt
Updated to revision 2.
$ ls sandwich.*
sandwich.txt sandwich.txt.mine sandwich.txt.r2
sandwich.txt.r1
$ cp sandwich.txt.r2 sandwich.txt
$ svn resolved sandwich.txt
```

.

Todas estas órdenes están también disponibles en las aplicaciones gráficas, como TortoiseSVN:

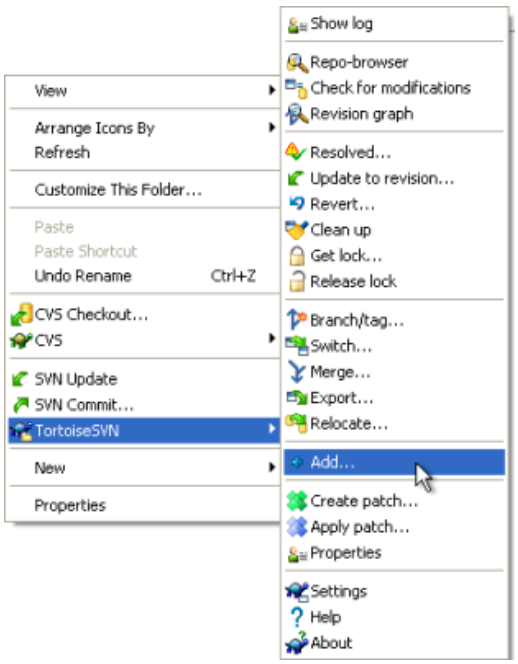


FOTO 9.1: OPERACIONES DEL CLIENTE SUBVERSION TORTOISESVN

9.3. Google Code

Google Code es un sitio para desarrolladores donde Google ofrece herramientas de desarrollo, *APIs* y recursos técnicos como documentación y foros especializados de discusión.

El conjunto de *APIs* que ofrece abarca toda la gama de productos Google: Maps, Youtube, Gmail, Calendar… Estas librerías están disponibles para la mayoría de lenguajes de programación, tanto para software convencional (Java, C++, C#) como para desarrollos web.

La plataforma Google Code ofrece un servicio de alojamiento de proyectos que incluye las siguientes funcionalidades:

- Control de versiones: Una cuota de 4GB en un sistema de control de versiones centralizado (Subversion) o descentralizado (Git o Mercurial). Además ofrece un visor web del repositorio que permite consultar los cambios realizados, descargar revisiones, subir ficheros, etcétera.
- Portal web: Con la plataforma disponemos de un portal web para describir el proyecto.
- Gestión de proyectos: Para facilitar la gestión de proyectos la plataforma Code pone a disposición del usuario una sección donde éste puede publicar nuevos desarrollos, notificar errores o solicitar revisiones.
- Wiki
- Área de descargas: Permite subir ficheros de hasta 200 MB dentro de la cuota máxima del proyecto (4GB).
- Información del equipo de desarrollo: Enlaces a los perfiles de cada desarrollador, con datos de contacto como el correo electrónico.

FOTO 9.2: GOOGLE CODE - PROYECTO SYSREG

Otras herramientas que pone a disposición del desarrollador son *plugins* para Eclipse, como el WindowBuilder Pro, el Google App Engine (una plataforma de programación en la nube que permite subir aplicaciones web a los *data centers* de Google) o el Google Web Toolkit (una herramienta de código abierto para crear aplicaciones Ajax en Java).

Google Code ofrece también soporte para desarrolladores a través de documentación *online*. Esta documentación incluye tutoriales, exploradores de *APIs*, búsquedas de proyectos relacionados y comunidades de desarrolladores especializados.

Google Code forma parte de Google Developers, una infraestructura para desarrolladores de Google que engloba todos los servicios de la compañía (Android, Google Apps y Chrome)

9.4. Fritzing

Fritzing es una iniciativa *open-source* para apoyar a diseñadores, artistas, investigadores y aficionados a trabajar de una manera creativa e interactiva con la electrónica.

Su objetivo es permitir a los usuarios documentar sus prototipos electrónicos y, a partir del diseño, crear un circuito impreso (PCB).

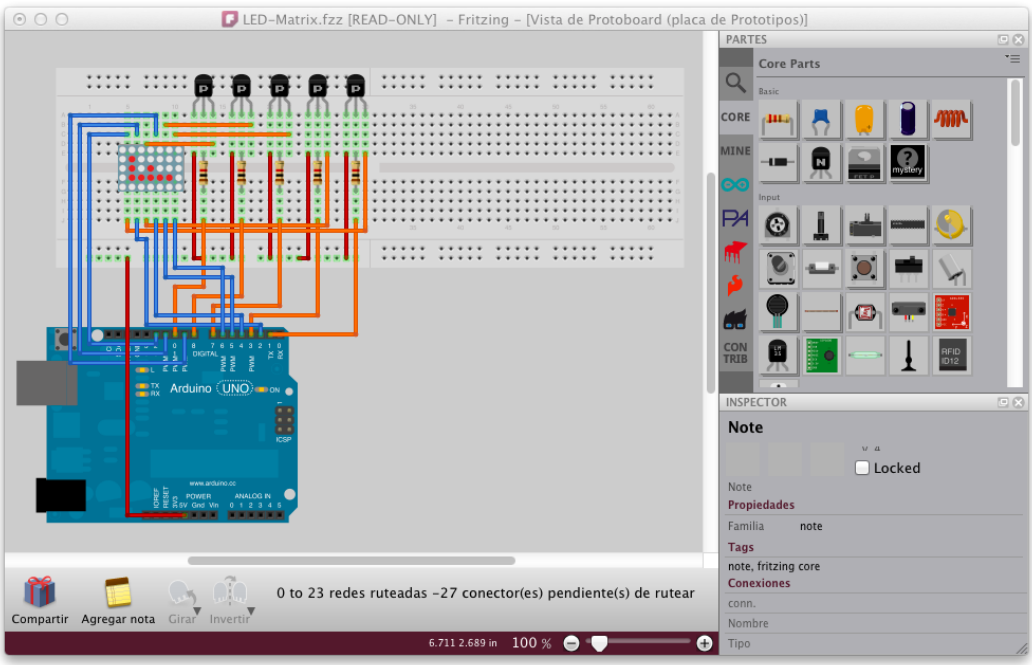


FOTO 9.3: FRITZING VISTA PLACA DE PROTOTIPOS

Fritzing forma parte del grupo de software dirigido a automatizar el diseño de componentes electrónicos. El programa básicamente consiste en la interacción de tres vistas: placa de prototipado, esquema de conexiones y circuito impreso.

En la primera de ellas el usuario introduce el diseño de su circuito de una manera amigable, con figuras de los componentes, cables coloreados, etc. En la segunda vista se puede observar el diseño de una manera más formal, representando con símbolos y conexiones los componentes.

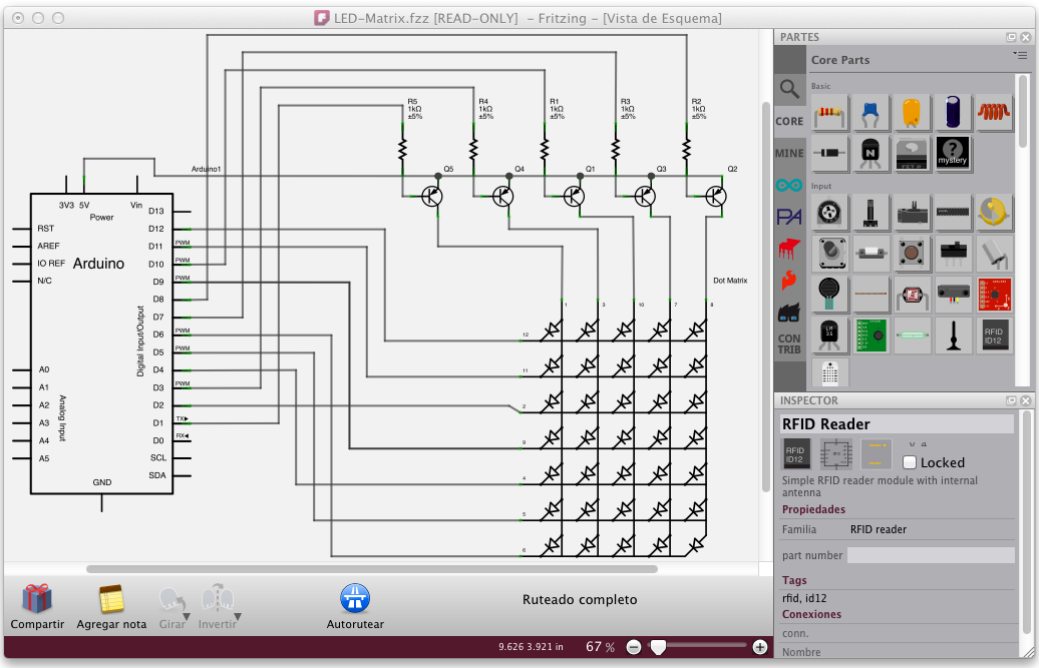


FOTO 9.4: FRITZING VISTA DE ESQUEMA

Y por último en la vista PCB se ofrece una visión de circuito impreso preparado para llevar a producción, con unas dimensiones compatibles con placas Arduino, pistas escritas y *slots* para ensamblar los componentes.

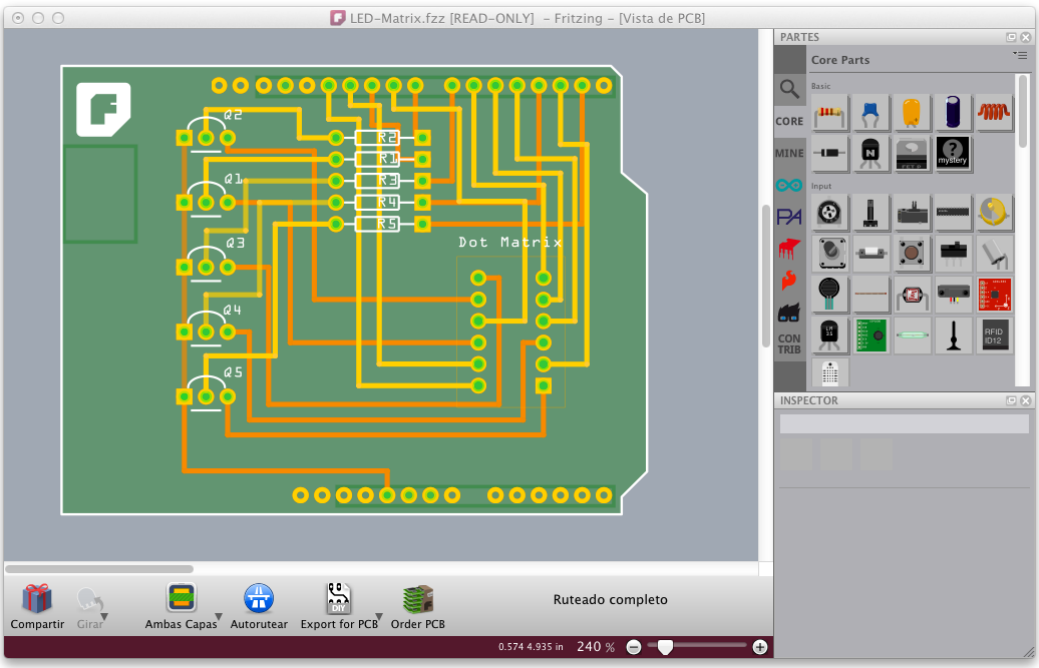


FOTO 9.5: FRITZING VISTA DE PCB

Además Fritzing incluye un editor de partes que permite crear diagramas y esquemas de dispositivos propios. Con esta herramienta, tanto empresas como usuarios, pueden modelar sus componentes hardware para documentar sus productos y para incluirlos en otros diagramas.

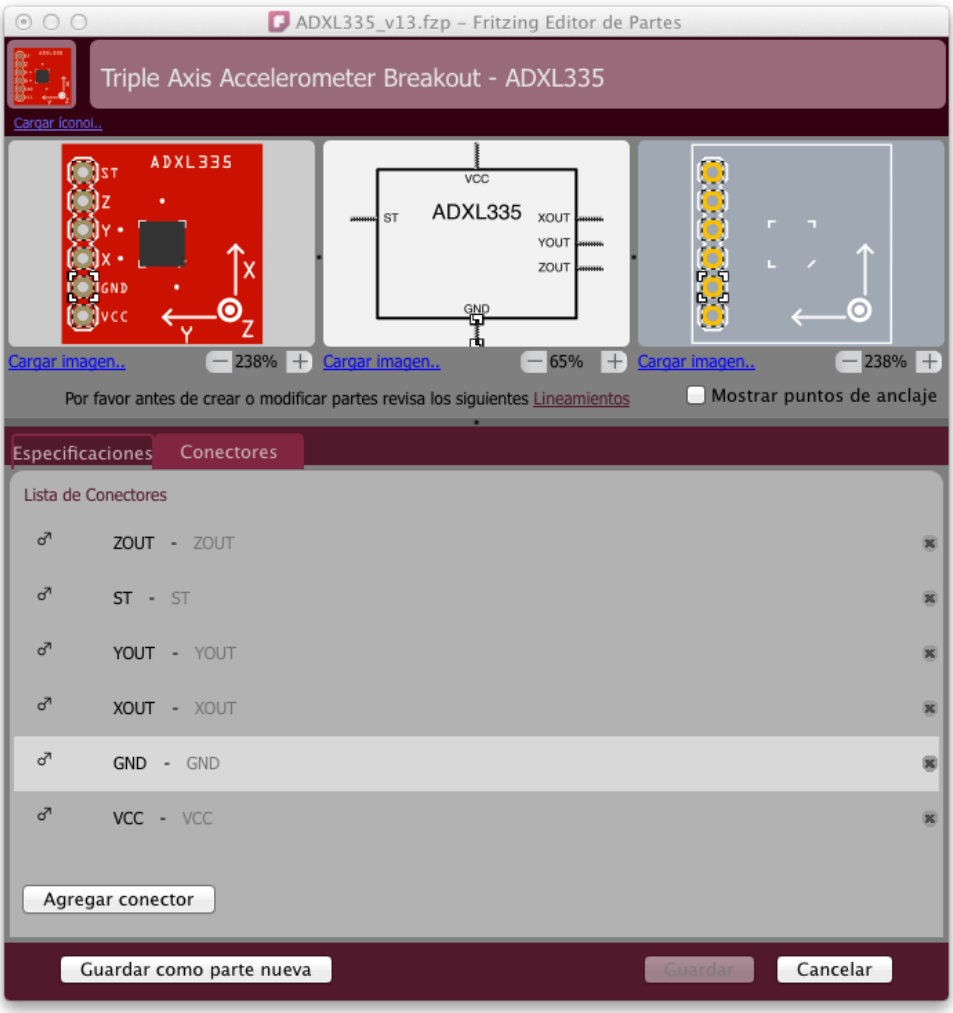


FOTO 9.6: FRITZING EDITOR DE COMPONENTES

9.5. Entorno de desarrollo

Para implementar los casos de usos enumerados en la fase de diseño, hemos utilizado principalmente dos entornos de desarrollo: el Arduino IDE y Eclipse.

El entorno de desarrollo Arduino está formado por un editor de texto, un área de mensajes, una consola de texto, una barra de accesos directos y un conjunto de herramientas:

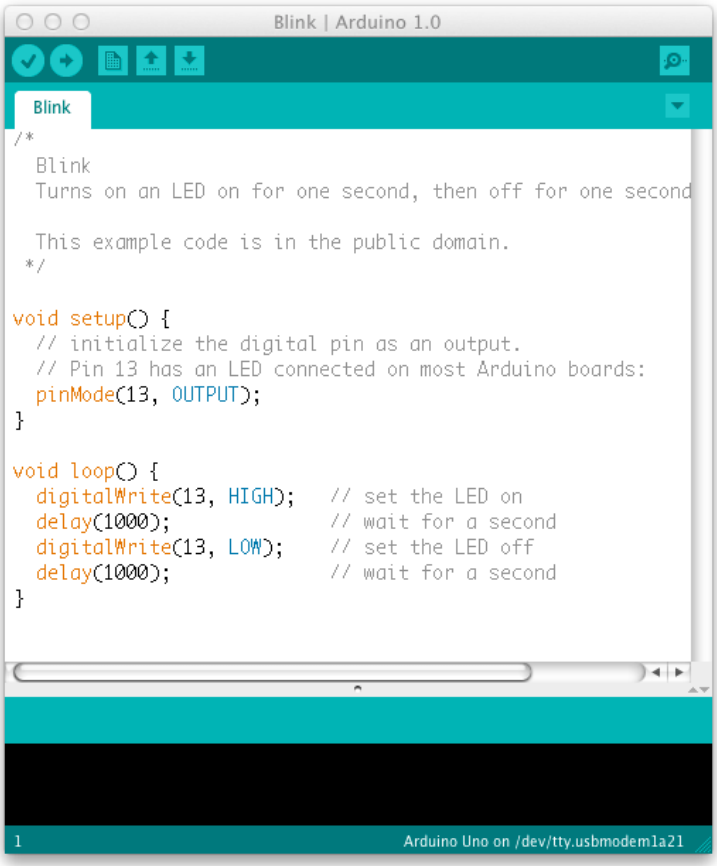


FOTO 9.7: ARDUINO IDE

Las principales funciones del IDE son:

- Verificar y compilar el *sketch*: En primer lugar realiza un análisis léxico, sintáctico y semántico. A continuación el IDE traduce el programa escrito a lenguaje C, y una vez en C es compilado por el compilador *avr-gcc* al lenguaje máquina del microcontrolador.
- Volcado a la placa: Una vez compilado el código es volcado a la placa. Esto se realiza utilizando el *bootolader* de Arduino.

- Monitorización serie: Abre un terminal serie entre la placa y el computador de manera que se pueden enviar y recibir datos de la placa. Permite ajustar variables como la velocidad de conexión (*baud rate*) y el tipo de codificación.
- Herramientas: Incluye además herramientas para auto formatear el texto, seleccionar la placa utilizada, grabar el gestor de arranque (*bootloader*), incorporar librerías y otras herramientas de edición.

La otra plataforma de desarrollo utilizada ha sido Eclipse. Eclipse es un entorno de desarrollo multiplataforma abierto disponible para multitud de lenguajes. La versión utilizada ha sido Eclipse IDE *for* Java *Developers*, una versión enfocada a desarrolladores Java.

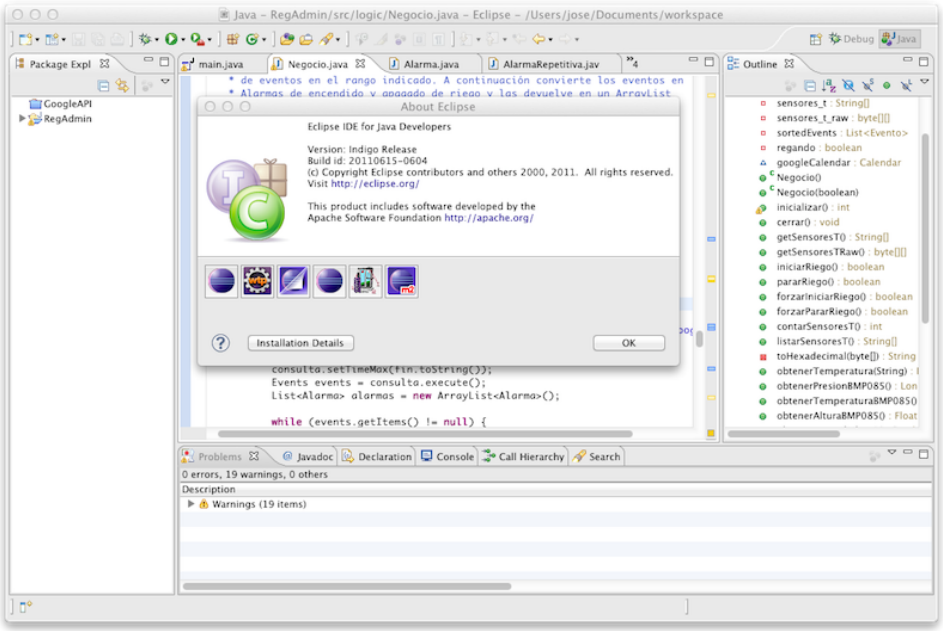


FOTO 9.8: ECLIPSE INDIGO

La plataforma Eclipse es extensible a través de *plugins*. En nuestro caso hemos utilizado WindowBuilder, un diseñador de interfaces gráficas para Java WYSIWYG (*What You See Is What You Get).*

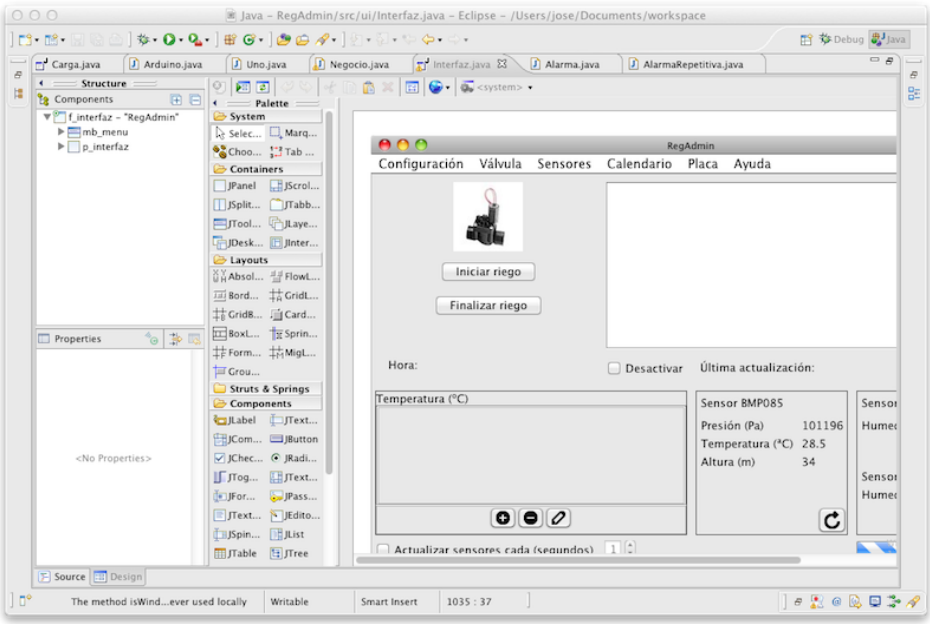


FOTO 9.9: WINDOWBUILDER

9.6. Microcontroladores: AVR y PIC

Un microcontrolador es un circuito integrado programable capaz de ejecutar órdenes guardadas en su memorias. El microcontrolador incorpora los tres componentes principales de una computadora: unidad central de proceso, memoria y periféricos de entrada y salida.

La explosión de los microcontroladores programables que ha tenido lugar durante los últimos años ha originado una intensa competencia entre dos compañías que ha dado lugar a la utilización de los términos AVR y PIC.

Los términos AVR y PIC no son más que dos familias de microcontroladores de las dos principales casa de micros: Atmel y Microchip Technology Inc.

Ambas familias incluyen micros de 8 bits, de muy bajo coste (unos pocos euros) y programables. Tienen unos pocos KB de RAM y EEPROM y utilizan periféricos como relojes y conversores analógico/digitales.

La familia Atmel AVR (Advanced Virtual RISC), como su nombre indica, es una arquitectura Harvard RISC de 8 bits surgida en 1996. En la misma pastilla incluye memoria flash y una pequeña ROM programable. Esta familia incluye los grupos tinyAVR, megaAVR, XMega, FPSLIC y AVR32, siendo el grupo megaAVR el

utilizado por la plataforma Arduino. Su desarrollo se enfocó para ser eficiente en la ejecución de código C.

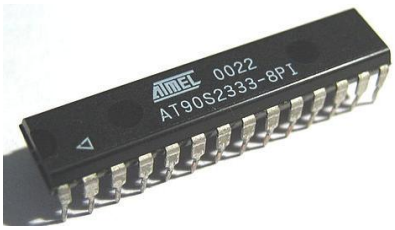


FOTO 9.10: MICROCONTROLADOR ATMEL

Los PIC (Peripheral Interface Controller) responden también a una arquitectura Harvard modificada de 8 bits. Su diseño data del 1975, motivo por está más orientado a instrucciones ensamblador.

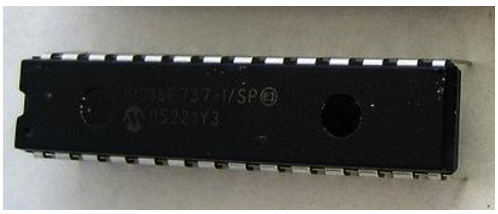


FOTO 9.11: MICROCONTROLADOR PIC

En ambas plataformas encontramos amplias comunidades de desarrolladores que ponen a disposición del público entornos de desarrollo avanzados, compiladores y otro tipo de herramientas.

Sin embargo las dos familias son incompatibles debido a las diferencias internas de los microchips. Encontramos diferencias a nivel de conjuntos de instrucciones (las instrucciones AVR son más regulares), la ubicación de la pila de llamadas, el acceso a la memoria RAM (los PIC tienen un acceso por bancos o segmentado mientras que los AVR son más uniformes).

En cuanto a rendimiento ambas familias cuentan con unas características similares.

En cambio, en la mayoría de comunidades de desarrollo se recomienda empezar por la familia AVR, ya que su enfoque es más estructurado (por el uso del lenguaje C) y posee un mayor número de herramientas para facilitar el aprendizaje.

9.7. USB *host* y USB *device*

Para entender la diferenciación entre dos interfaces USB tenemos que entender el funcionamiento del protocolo USB.

La arquitectura USB está definida de modo asimétrico, es decir, formada un *host* (dispositivo anfitrión con multitud de puertos USB) y múltiples periféricos conectados en una topología de estrella por niveles en la que se pueden incluir *hubs* o concentradores. Los dispositivos USB anfitriones implementan el controlador *host* (normalmente 1 por cada puerto USB) que es el encargado de iniciar la conexión, definir velocidad y asignar un identificador único de 7-bits. El host controla el flujo de información entre dispositivos y reparte el bus (con el algoritmo *round-robin* o cíclico).

En cambio el cliente USB solo incorpora los drivers necesarios para llevar a cabo la comunicación.

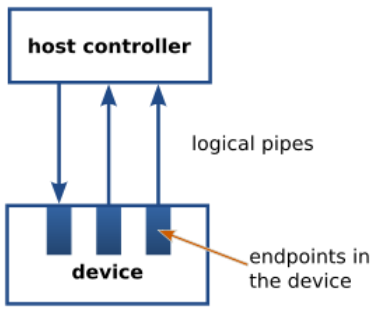


FIGURA 9.4: CONEXIÓN USB *HOST – DEVICE*

9.8. Librerías

9.8.1. I^2C

El protocolo I^2C, también llamado interfaz 2-wire, es un bus maestro-esclavo inventado por Philips en 1992 y revisado en el año 2000. En su modo estándar tiene una velocidad de conexión de 100 Kbits.

El bus está formado por 3 líneas: SDA, SDL y GND. La línea de SDA es utilizada para transmitir los datos con un ancho de dirección de 7 bits. La línea SCL es el reloj del bus y sirve para sincronizar los dispositivos. Estas dos líneas están conectadas en modo drenador abierto (*pull-up*). También tenemos la línea de tierra compartida por todos los dispositivos.

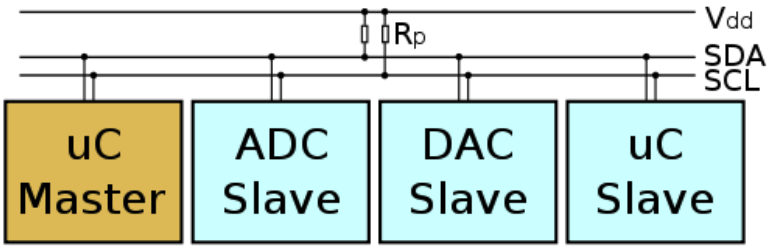


FIGURA 9.5: CIRCUITO I²C EN EL QUE TENEMOS UN MICORCONTROLADOR QUE ACTÚA COMO MAESTRO, UN CONVERSOR ANALÓGICO A DIGITAL COMO ESCLAVO, UN CONVERSOR DIGITAL A ANALÓGICO COMO ESCLAVO Y OTRO MICROCONTROLADOR FUNCIONANDO TAMBIÉN COMO ESCLAVO

Es un bus de tipo multi-maestro, en el que los dispositivos pueden actuar tanto como maestro como esclavo, intercambiándose la capacidad de controlar la conexión entre dispositivos (modo maestro).

Los mensajes en el bus tienen un formato definido:

| start | A7 A6 A5 A4 A3 A2 A1 | R/W | ACK | ... DATA ... | ACK | stop | idle |

En él los bits A7-A1 definen la dirección del dispositivo objetivo, siendo el A0 el bit que indica si la operación de lectura o escritura, y a continuación, tras un carácter de reconocimiento, enviamos la información (datos u órdenes).

Su controlador se encuentra compilado para la mayoría de sistemas operativos, como en Linux cuyo código está incluido en el *kernel*.

En Arduino disponemos de la librería *Wire.h* para interaccionar con los dispositivos. La librería *Wire.h* está soportada oficialmente por la comunidad Arduino, incluyéndola en el IDE oficial. Por defecto la placa UNO tiene configurado los pines analógicos 4 como línea de datos (SDA) y el 5 como línea de reloj (SDL).

Para realizar la comunicación en primer lugar tenemos que ejecutar el método *begin()* en la función *setup* del *sketch*.

```
void setup(void){
       Wire.begin();
       [[...]]
}
```

A continuación para establecer la conexión con un esclavo tenemos que iniciar la comunicación (indicando la dirección del dispositivo), escribir el comando y cerrar la transmisión.

```
Wire.beginTransmission(ADDRESS);
Wire.write(0xF4);
Wire.write(0x2E);
Wire.endTransmission();
```

Y para leer hacemos una petición al dispositivo

```
Wire.requestFrom(ADDRESS, N);
while(Wire.available()<2)
      ;
msb = Wire.read();
lsb = Wire.read();
```

Estos son básicamente las funciones necesarias para establecer la comunicación con un dispositivo esclavo I^2C, aunque también disponemos de las funciones *onRecieve()* y *onRequest()* que permite definir manejadores para los eventos de envío y petición.

9.8.2. OneWire

1-Wire es un protocolo de comunicaciones de bajo nivel. Está formado por un bus, un maestro y varios esclavos. Tiene la peculiaridad de que la línea de datos puede servir de alimentación a los esclavos, que se conectan al bus de datos con una resistencia *pull-up*.

Fue diseñado por la empresa Dallas Semiconductor, hoy en día parte de Maxim Integrated Products.

El bus define una serie de señales que dirigen el funcionamiento de los dispositivos. Manteniendo una señal de 0 voltios durante 480 microsegundos se produce un reinicio de todos los dispositivos conectados. El envío y recepción de datos los inicia el maestro registrando durante unos 15 microsegundos un valor lógico bajo (0 V). Los esclavos, que tienen un identificador único para cada dispositivo, envían o reciben datos en grupos de 8 bits, con códigos de detección de errores CRC. Las comunicaciones se

establecen reiniciando el bus, siendo implementación propia de cada dispositivo la interpretación de los comandos enviados.

En comparación con el protocolo I^2C, 1-Wire tiene un consumo más bajo, menor velocidad y un alcance más amplio.

Encontramos diversos dispositivos que implementan este protocolo: sensores meteorológicos (e. g. termómetros), contadores, relojes, memorias…

Para interaccionar con dispositivos 1-Wire disponemos de una librería escrita en C y portada a la plataforma Arduino: la librería *OneWire.h*.

Esta librería permite definir el pin que será el bus 1-Wire y trabajar con él a modo de clase, con funciones de envió y recepción de datos.

Lo primero es definir un objeto global que identifique el pin del bus 1-Wire:

```
OneWire ds(12);
```

Podemos realizar un reseteo del bus y almacenar los identificadores de los dispositivos en vectores de bytes con las siguientes órdenes:

```
byte addr[8];
ds.reset_search();
ds.search(addr);
```

Una vez identificado el esclavo solamente queda seleccionarlo y enviar la orden correspondiente, esperando respuesta de byte en byte con el método *read():*

```
ds.select(sensor_id);
ds.write(0xBE);
for (int i = 0; i < N; i++)
       data[i] = ds.read();
```

9.8.3. FreqCounter

Frequency Counter es una librería para Arduino desarrollada por Lab3, el laboratorio de Ciencias de la Computación en la Academia de Artes Audiovisuales de Colonia.

Esta librería lee la señal en forma de frecuencia por el pin digital 5 y la transforma a un entero largo. Para realizar correctamente la conversión es necesario

definir la resolución del muestreo en ms y definir una compensación, generalmente con un valor de 8.

Una vez configurada simplemente tendremos que esperar a que esté preparada y leer el valor.

```
#include <FreqCounter.h>
void setup() {
        Serial.begin(57600);
        Serial.println("Frequency Counter");
}
long int frq;
Void loop() {
        FreqCounter::f_comp= 8;
        FreqCounter::start(100);
        while (FreqCounter::f_ready == 0)
                frq=FreqCounter::f_freq;
        Serial.println(frq);
        delay(20);
}
```

9.8.4. Time

Para la gestión del tiempo disponemos en Arduino de una librería soportada oficialmente que, mediante componentes únicamente software, permite manejar el tiempo y las fechas.

En primer lugar establecemos la hora, ya sea en formato UNIX (segundos desde 1 de Enero de 1970) o en formato estándar, ajustando en caso necesariola zona horaria:

```
setTime(t);
setTime(hr,min,sec,day,month,yr);
adjustTime(adjustment);
```

Una vez establecida, la librería se encargará de actualizar el reloj internamente para poder realizar consultas con los métodos proporcionados:

```
hour();            // the hour now  (0-23)
minute();          // the minute now (0-59)
second();          // the second now (0-59)
day();             // the day now (1-31)
weekday();         // day of the week, Sunday is day 1
month();           // the month now (1-12)
year();            // the full four digit year: (2009, 2010 etc)a
hourFormat12();    // the hour now in 12 hour format
isAM();            // returns true if time now is AM
isPM();            // returns true if time now is PM
now();             // returns the current time as seconds since Jan 1 1970
```

También podemos sincronizar el reloj desde otras fuentes, como GPS, el protocolo de tiempo NTP (a través de un *shield* Ethernet) o de un dispositivo externo físico como el módulo DS1307:

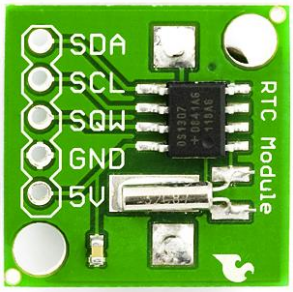


FIGURA 9.6: MÓDULO RTC (REAL TIME CLOCK) DE SPARKFUN

9.8.5. TimeAlarms

Complementando a la librería *Time.h* de Arduino tenemos la librería *TimeAlarms.h*, que permite definir alarmas y disparadores en función del tiempo.

Tanto las alarmas como los disparados se definen internamente como *triggers* que se activan por interrupciones. Podremos definir hasta un máximo de 255 alarmas (modificando la variable *dtNBR_ALARMS* del fuente *TimeAlarms*.h).

Las alarmas son disparadores definidos en un momento exacto. Tenemos la posibilidad de definirlas para que se ejecuten una sola vez o de manera repetitiva. Las funciones para establecer una alarma son:

```
       AlarmID_t alarmRepeat(time_t value, OnTick_t onTickHandler);
       AlarmID_t alarmRepeat(const int H, const int M, const int S, OnTick_t
onTickHandler);
       AlarmID_t alarmRepeat(const timeDayOfWeek_t DOW, const int H,  const int M,
const int S, OnTick_t onTickHandler);
       AlarmID_t alarmOnce(time_t value, OnTick_t onTickHandler);
       AlarmID_t alarmOnce( const int H, const int M, const int S, OnTick_t
onTickHandler);
       AlarmID_t alarmOnce(const timeDayOfWeek_t DOW, const int H,  const int M,
const int S, OnTick t onTickHandler);
```

Por otra parte podemos definir temporizadores que se activarán pasado un cierto periodo de tiempo, ya sea una vez o de manera indefinida:

```
AlarmID_t timerOnce(time_t value, OnTick_t onTickHandler);
AlarmID_t timerRepeat(time_t value, OnTick_t onTickHandler);
```

Existen también funciones privadas para eliminar las alarmas o modificar su valor:

```
void free(AlarmID_t ID);
void enable(AlarmID_t ID);
void disable(AlarmID_t ID);
void write(AlarmID_t ID, time_t value);
```

Esta librería no viene incluida por defecto en el IDE de Arduino, por lo que tendremos que recurrir a la descarga e importación en la carpeta *~/Arduino/libraries* para poder trabajar con ellas.

9.8.6. RXTX

RXTX es una librería Java que, usando una implementación nativa para cada sistema operativo (vía la interfaz nativa de Java o JNI), provee de una comunicación serie y paralela al JDK (kit de desarrollo Java).

Está diseñada basándose en las especificaciones de la API de comunicaciones de Java (el método proporcionado por Oracle para facilitar la implementación de interfaces de comunicación, como por ejemplo la comunicación serie RS-232), aunque utiliza paquetes de las librerías *gnu.io*, lo que la hace incompatible.

Todo el proyecto está liberado bajo la licencia GNU LGPL (*Lesser GPL*).

Para usar la librería en nuestro código debemos establecer la conexión con el puerto deseado, abrir un flujo de entrada y uno de salida (*InputStream* y *Output Stream)* y utilizar las funciones *write* y *read* de la clase para enviar y recibir *arrays* de bytes:

```
void connect ( String portName ) throws Exception {
        CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier(portName);
        if ( portIdentifier.isCurrentlyOwned() )
                System.out.println("Error: Port is currently in use");
        else{
                CommPort commPort = portIdentifier.open(this.getClass().getName(),
        2000);
                if (commPort instanceof SerialPort ) {
                        SerialPort serialPort = (SerialPort) commPort;
                        serialPort.setSerialPortParams(57600, SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
                        InputStream in = serialPort.getInputStream();
                        OutputStream out = serialPort.getOutputStream();
                        read();
                        write();
                } else
                        System.out.println("Error: Invalid Serial Port.");
        }
}
```

```
void read () {
        byte[] buffer = new byte[1024];
        int len = -1;
        try     {
                while ( ( len = this.in.read(buffer)) > -1 )
                        System.out.print(new String(buffer,0,len));
        } catch ( IOException e ) {
                e.printStackTrace();
        }
}
void write (){
        try     {
                int c = 0;
                while ( ( c = System.in.read()) > -1 )
                        this.out.write(c);
        } catch ( IOException e ) {
                e.printStackTrace();
        }
}
```

9.8.7. Google API y google-api-java-client

Google dispone de una API para permitir desarrollar aplicaciones que accedan a sus servicios. En su última versión (3.0) la API está definida sobre una arquitectura REST (*Representational State Transfer*) que utiliza mensajes JSON a través del protocolo HTTP. Hablamos pues de una arquitectura orientada a servicios (SOA) que permite definir una interfaz de comunicación independiente del lenguaje.

Esta API de primer nivel requiere de la construcción de una petición POST que envíe los datos del mensaje en formato JSON (*JavaScript Object Notation*), como muestra el siguiente ejemplo que crea un evento con una frecuencia de repetición:

```
POST /calendar/v3/calendars/primary/events
...

{
  "summary": "Appointment",
  "location": "Somewhere",
  "start": {
    "dateTime": "2011-06-03T10:00:00.000-07:00",
    "timeZone": "America/Los_Angeles"
  },
  "end": {
    "dateTime": "2011-06-03T10:25:00.000-07:00",
    "timeZone": "America/Los_Angeles"
  }
}
"recurrence": [
    "RRULE:FREQ=WEEKLY;UNTIL=20110701T100000-07:00",
  ],
  "attendees": [
    {
      "email": "attendeeEmail",
      # Other attendee's data...
    },
    # ...
  ],
}
```

Para cada servicio de Google (Maps, Calendar, Gmail…) disponemos de una API dedicada.

Sin embargo el método de autenticación es compartido por todos los servicios. Este método en la API v3, es el llamado OAuth 2.0, un protocolo cuyo RFC todavía se encuentra en fase de borrador y que permite autentificar y autorizar peticiones a través de una URL y una clave. Para su funcionamiento la aplicación se debe registrar en el Google API Console con un identificador único, una cuenta Google y una descripción de los servicios que va a utilizar. Una vez registrada la aplicación cliente solicita el *token* de conexión al servidor de autorizaciones de Google y éste le devuelve una URL de autorización. Esta URL deberá ser introducida en el navegador y requerirá al usuario de su contraseña. Si el inicio de sesión es correcto, devolverá un *token* de acceso que, al enviarlo la aplicación cliente al servidor, permitirá establecer la comunicación. De esta manera la aplicación cliente no maneja ningún ni usuario ni contraseñas.

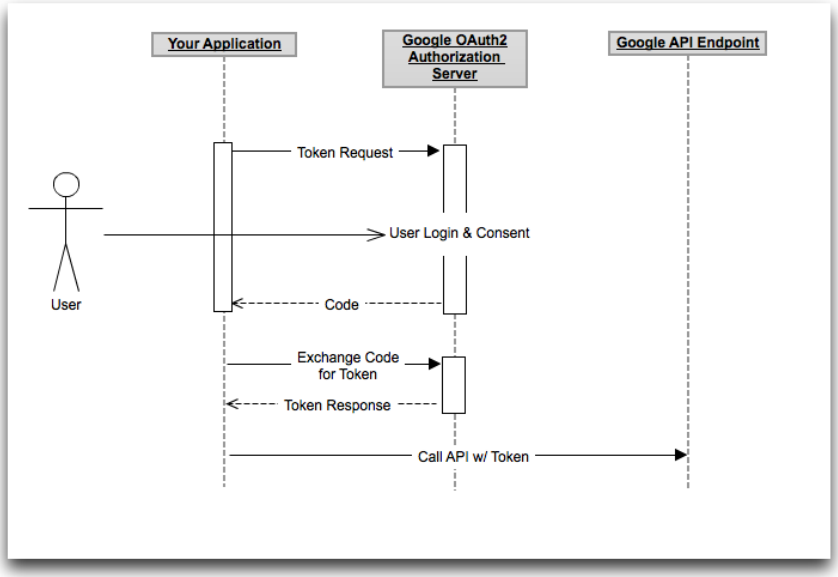


FIGURA 9.7: DIAGRAMA DE SECUENCIA DE LA CONEXIÓN MEDIANTE OAUTH 2.0

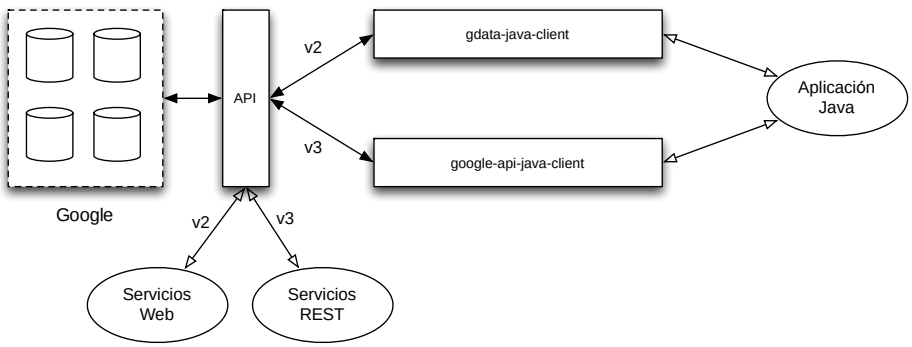
Toda la información de la API está disponible, estructurada por servicios, en el portal para desarrolladores de Google (https://developers.google.com/), con herramientas para generar peticiones, enlaces a RFC, códigos ejemplo…

Sin embargo esta arquitectura SOA tan especializada resulta tediosa para trabajar con lenguajes de alto nivel como Java. Por ello Google ha desarrollado librerías cliente para cada lenguaje que atacan la API. Tenemos clientes para Java, Python, PHP,

.NET y Ruby. Estas librerías actúan como intermediarios, modelando los datos (con clases y métodos para cada objeto) y empaquetando las peticiones.

Pero igual que ha evolucionado la API de Google hasta la versión 3, también lo han hecho las librerías clientes. En el caso de la librería cliente para Java tenemos actualmente dos versiones:

- *gdata-java-client*: Era el cliente de la API v2. Tiene un desarrollo más clásico, con una frecuencia menor de actualizaciones. Su modelo de datos utiliza clases más pesadas y transmite la información en formato XML (incrementando el tamaño de los paquetes). Google sigue ofreciendo soporte para esta librería pero recomienda a los desarrolladores migrar a la nueva librería.
- *google-api-java-client:* Librería cliente para la API v3 desarrollada con un modelo vista-controlador. Utiliza JSON, un formato más ligero que el XML, para intercambiar datos. Esta implementación es compatible con el sistema operativo Android y su ciclo de vida es más ágil (mayor frecuencia de actualizaciones). Para preservar la compatibilidad con la anterior librería se ha definido con un nuevo espacio de nombre. En la actualidad ya modela los datos de la mayor parte de servicios de Google



- **FIGURA 9.8: GOOGLE API**

10. CONCLUSIONES

Nos encontramos en un periodo de madurez de las tecnologías de la información donde cualquier ámbito de la vida real sufre un proceso de informatización. Mientras, nuevos paradigmas de la computación, como los sistemas ubicuos, avanzan hacia una fase de implantación, apoyados en parte por los progresos en redes y arquitecturas de bajo consumo.

Además, en una coyuntura económica difícil, los proyectos de software libre empiezan a ser implantados en masa y plataformas como Arduino y Android son ya referentes en sus campos.

En este contexto las ciencias de la computación tienen una oportunidad única para dar un salto cualitativo a la sociedad. Sin embargo, estas soluciones llegarán únicamente a las industrias con un mayor poder económico, dejando de lado sectores tradicionales como la agricultura.

Es aquí donde surge el proyecto Sysreg: unir las investigaciones y desarrollos que se están llevando a cabo para evolucionar la agricultura.

Esta tesis de carácter profesional ha pretendido ser la punta de lanza de un sistema de información agrícola abierto y extensible que, apoyándose en métodos formales y nuevas tecnologías, construya un sistema inteligente, autónomo y de bajo coste.

El resultado de esta primera fase ha sido todo un éxito, ya que ha cumplido con los objetivos básicos del proyecto: mejorar la productividad, reducir costes y ser un proyecto innovador.

Asimismo en el área profesional la tesina ha cuajado en un proyecto de I+D cuya aprobación está pendiente.

Y ya por último a nivel personal este proyecto ha permitido indagar de una manera más rigurosa en el campo de la agronomía, estudiando los fundamentos de los cítricos y las técnicas formales de análisis. Además ha resultado gratificante observar como se pueden desarrollar proyectos innovadores utilizando componentes hechos por uno mismo y herramientas abiertas y gratuitas.

11. REFERENCIAS

[1] Agustí, M. *Citricultura.* Ediciones Mundi-Prensa 2000

[2] Conselleria de Agricultura, Pesca, Alimentación y Agua. http://www.agricultura.gva.es/

[3] Instituto Valenciano de Invetigaciones Agrarias. http://www.ivia.es

[4] Wikipedia, *Clima Mediterraneo.* http://es.wikipedia.org/wiki/Clima_Mediterr%C3%A1neo

[5] Wikipedia, *Clima de la Comunidad Valenciana.* http://es.wikipedia.org/wiki/Clima_de_la_Comunidad_Valenciana

[6] Wikipedia, *Geografia de la Comunidad Valenciana.* http://es.wikipedia.org/wiki/Geograf%C3%ADa_de_la_Comunidad_Valenciana

[7] Ministerio de Agricultura, Alimentación y Medio Ambiente. http://www.magrama.gob.es/es/

[8] Ministerio de Agricultura, Alimentación y Medio Ambiente, *Sistema de Información Agroclimática para el Regadio.* http://www.magrama.gob.es/es/agua/temas/observatorio-del-regadio-espanol/sistema-de-informacion-agroclimatica-para-el-regadio/default.aspx (Requiere Internet Explorer 6,7 u 8)

[9] FAO. *Evapotranspiración del cultivo: Guías para la determinación de los requerimientos de agua de los cultivos.* Estudio FAO Riego y Drenaje 56. ISBN 0254-5293

[10] Wikipedia, *Cooperativa.* http://es.wikipedia.org/wiki/Cooperativa

[11] Wikipedia, *Movimiento cooperativo.* http://es.wikipedia.org/wiki/Movimiento_cooperativo

[12] Wikipedia, *Cooperativa agraria.* http://es.wikipedia.org/wiki/Cooperativa_agraria

[13] Conselleria de Agricultura, Pesca, Alimentación y Agua. *Cooperativismo.* http://www.agricultura.gva.es/web/web/guest/cooperativismo/presentacion

[14] Anecoop. www.anecoop.com

[15] Institudo Valenciano de Investigaciones Agrarias. *Nuevas variedades de cítricos.* www.ivia.es/documentos/nuevasvariedades/nuevas.htm

[16] UBC Botanical Garden and Centre for Plant Research www.botanicalgarden.ubc.ca/

[17] Wikipedia, *Kiwi.* en.wikipedia.org/wiki/Actinidia_deliciosa

[18] Wikipedia, *Sexualidad vegetal.* http://es.wikipedia.org/wiki/Sexualidad_vegetal

[19] Comité de Agricultura Ecológica de la Comunidad Valenciana. www.caecv.com/

[20] Wikipedia, *Agricultura ecológica.* es.wikipedia.org/wiki/Agricultura_ecológica

[21] Wikipedia, *Compost.* es.wikipedia.org/wiki/Compost

[22] Accenture, *Simac Guía desarrolladores.*

[23] http://esircaint.gva.es/es/web/tramitacion/descripcion-tramita E-sirca

[24] Jose Antonio Hernández Martínez, *Proyecto Taronline S. A.* Estrategias y Nuevas Tecnologías, 3r curso Ingeniería en Informática.
[25] Javier Jaén. *Introducción SIU*. Sistemas de Información Ubicuos, Máster en Ingeniería del Software, Sistemas de Información y Métodos Formales.
[26] Xiao Hang Wang, Da Qing Zhang, Tao Gu1, Hung Keng Pung, *Ontology Based Context Modeling and Reasoning using OWL*. Institute for Infocomm Research, Singapore, School of Computing, National University of Singapore
[27] Mark Weiser, *The computer for the 21st Century*
[28] M. Weiser, R. Gold, JS Brown, *The origins of ubiquitous computing research at PARC in the late 1980s*
[29] Matthias Baldauf, *A survey on context-aware systems*. Int. J. Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, 2007
[30] Joan Fons, *Introducción AmI*. Diseño e implementación de Sistemas de Inteligencia Ambiental, Máster en Ingeniería del Software, Sistemas de Información y Métodos Formales.
[31] Wikipedia, *Processing.* en.wikipedia.org/wiki/Processing_(programming_language)
[32] Processing. processing.org
[33] Massimo Banzi, *Getting started with Arduino*. O'Reilly Make: Projects. ISBN 978-0-596-15551-3. 2009
[34] Tom Igoe, *Making Things Talk*. O'Reilly Make: Projects. ISB-978-0-596-51051-0
[35] Arduino. arduino.cc
[36] Wikipedia, *Power Over Ethernet.* es.wikipedia.org/wiki/PoE
[37] Cooking-hacks. Grupo Libelium. www.cooking-hacks.com
[38] Netduino. netduino.com
[39] Chris Walker, *Getting Started with Neduino*, O'Reilly Make: Projects.
[40] HackingLab, *Pingüino.* www.hackinglab.org/pinguino/index_pinguino.html
[41] Wikipedia, *Atmel AVR.* en.wikipedia.org/wiki/Atmel_AVR
[42] Wikipedia, *PIC.* en.wikipedia.org/wiki/PIC_microcontroller
[43] Wikipedia, *Solenoid vale.* en.wikipedia.org/wiki/Solenoid_valve
[44] Wikipedia, *H-bridge.* http://en.wikipedia.org/wiki/H_bridge
[45] Maxim, *Digital Thermometer DS18B20.* www.maxim-ic.com/datasheet/index.mvp/id/2812
[46] Wikipedia, *Humedad.* http://es.wikipedia.org/wiki/Humedad
[47] Bosh, *Digital pressure sensor BMP085.* dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Pressure/BST-BMP085-DS000-06.pdf
[48] Wikipedia, *I^2C.* es.wikipedia.org/wiki/I²C
[49] GardenBot, *Soil Moisture.* gardenbot.org/howTo/soilMoisture/
[50] Excelsion, *Convert Java to Exe.* www.excelsior-usa.com/articles/java-to-exe.html
[51] Lauch4j. http://launch4j.sourceforge.net
[52] Wikipedia, *Microcontrolador.* es.wikipedia.org/wiki/Microcontrolador

[53] RXTX. rxtx.qbang.org/wiki/index.php/Main_Page
[54] LabIII, *Frequency Counter Library.* interface.khm.de/index.php/lab/experiments/arduino-frequency-counter-library/
[55] Arduino, *Time library.* arduino.cc/playground/Code/Time
[56] Google Inc., *Google Developers.* developers.google.com
[57] Google Inc. *Google Calendar API.* developers.google.com/google-apps/calendar
[58] Google Inc. *OAuth2.* developers.google.com/accounts/docs/OAuth2
[59] Google Inc., *Google Code.* code.google.com/intl/es