Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Informatización de un módulo para la construcción de jardines inteligentes

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Axel Guzman Godia

Tutores: Poza Luján, José Luis y Posadas Yagüe, Juan Luis

2015/2016

Resumen

Mantener un cultivo requiere de una serie de cuidados rutinarios. Sería ideal poder contar con algún tipo de automatismo que nos ayudara a realizar las acciones necesarias para mantener en buen estado nuestra planta. En este documento se va a realizar el diseño y la implementación de un sistema para automatizar el cuidado de una planta.

Este sistema será capaz de regarse, moverse para buscar la luz, y percibir las condiciones de temperatura y humedad del entorno. Para ello haremos uso de un microcontrolador conectado a una serie de sensores. Todo el conjunto se montará en un soporte con motores para poder moverse. El sistema desarrollado se integra en un proyecto que abarca aspectos de comunicación e interfaces con el usuario a través de la red y de los teléfonos inteligentes.

Palabras clave: Automatización, inteligencia artificial, ecología.

Abstract

Taking care of a plant requires some routine jobs. It would be ideal to have an automatism in order to help doing that kind of tasks. This document contains the design and the implementation of a system meant to take care of a plant.

This system will be able to water itself, to move to find sunlight, and to sense the ambient conditions, as temperature and humidity. To achieve this, we will use a microcontroller connected to several sensors. All the system will be placed on a mobile platform featuring its own engines to provide movement capabilities. The developed system it’s part of a bigger project that covers communication and interfaces through mobile phones with the final user.

Keywords : Automatism, artifical intelligence, ecology.

Tabla de contenidos

Ilustraciones

Tablas

1 Introducción

1.1 Motivación

Desde los principios de la agricultura el ser humano se ha dedicado a cultivar y hacer crecer plantas. Al principio se cultivaba para obtener alimento. Sin embargo más adelante se empezó a cultivar simplemente como afición o para fines estéticos.

Hoy en día son muchas las personas que mantienen plantas en sus casas, oficinas o jardines, tanto con fines decorativos como alimenticios.

Encargarse de cuidar un cultivo es una labor que depende en gran medida del tipo de cultivo pero que requiere dedicar tiempo. La falta de atención podría acabar con la muerte del cultivo ya sea por falta de agua, abono o por condiciones ambientales adversas (frío, calor excesivo, sobreexposición a los rayos solares…)

Tradicionalmente estos cuidados han sido llevados a cabo por personas encargadas de mantener en buen estado las plantaciones. Sin embargo, más adelante empezaron a aparecer diversos sistemas para ocuparse de las labores más sencillas.

En los cultivos industriales suelen utilizarse automatizaciones de control de riego o sistemas para controlar el entorno (invernaderos). Sin embargo en el sector particular lo más habitual es automatizar el riego, en caso de que se utilice algún automatismo.

Sería posible mejorar el control de las tareas introduciendo sistemas encargados de monitorizar las condiciones ambientales. Se podrían incluir sensores [1] para determinar factores ambientales como temperatura, radiación solar, humedad relativa e incluso para conocer la previsión meteorológica y actuar en consecuencia. Todo esto sería gobernado por un sistema encargado de tratar la información y tomar las decisiones más apropiadas.

En este contexto se aborda la ingeniería informática como una solución al problema de la automatización.

1.2 Objetivos

El objetivo de este trabajo es diseñar e implementar un sistema que sea capaz de satisfacer el problema planteado. La finalidad es conseguir un sistema que sea capaz de encargarse del cuidado de nuestros cultivos. Dicho sistema debe ser capaz de regarse a sí mismo y de detectar las condiciones del entorno. También se espera que sea capaz de moverse para buscar la luz solar. Por último también deberá ser capaz de comunicarse con el exterior y con otros sistemas similares.

Se espera obtener un sistema capaz de realizar todas estas acciones, además de ser capaz de comunicarse con el exterior.

1.3 Estructura del documento

Este documento se ha organizado en siete capítulos. La estructura que se ha seguido se explica a continuación.

El primer capítulo es la introducción, aportando información sobre la motivación, los objetivos y la estructura del trabajo. A continuación, en el siguiente capítulo, se realiza un estudio de mercado, buscando sistemas similares al que se pretende desarrollar y

analizando las características más destacadas de los sistemas. El tercer capítulo está dedicado a redactar la especificación de requisitos del proyecto. Seguido, en el cuarto capítulo, se presenta todo el diseño realizado para desarrollar el sistema.

El quinto capítulo está dedicado a los detalles de implementación. En este capítulo se muestra el código desarrollado y el montaje del sistema. Además también se realizan pruebas con el resultado final y se exponen los resultados. El penúltimo capítulo hace un repaso a todo el trabajo realizado y los resultados obtenidos, así como los problemas encontrados. Por último, en el capítulo siete, se muestran las referencias bibliográficas.

2 Entorno de realización

2.1 Sistemas similares

El objetivo de este apartado es mostrar una selección de sistemas cuya finalidad es solucionar el mismo problema que se planeta en este trabajo o parte del problema. La finalidad es realizar un estudio sobre los sistemas encontrados para determinar qué características son deseables y cuáles no.

2.1.1 Citysens

Citysens [2] es un sistema modular de jardines verticales. Está basado en el cultivo hidropónico; es decir elimina la necesidad de tener un sustrato para el cultivo. Actualmente disponen de diferentes modelos en el mercado.

Ilustración 1. Jardín de interior *Citysens*[1].

El principal enfoque de este producto se basa en el cultivo en interiores. En consecuencia el sistema sólo se encarga de controlar el riego, que está automatizado.

No se tienen en cuenta factores como la cantidad de luz recibida o la temperatura ambiental. Tampoco incluye ningún tipo de comunicación con el exterior o con otros módulos del mismo tipo.

El principal punto fuerte de este sistema es el diseño y la modularidad, que permite flexibilidad a la hora de diseñar jardines de interior.

2.1.2 Growiee

Growiee [3] es un sistema de jardines en miniatura para interior. Se trata de un sistema hidropónico diseñado para acelerar el crecimiento de los cultivos. En la Ilustración 2 se muestra el aspecto del sistema.

[1] fuente: http://www.citysens.com/es/15-jardines-verticales.

Ilustración 2. Módulo de cultivo *Growiee*[2] .

La característica principal de este sistema es que se encarga de modificar el entorno con el fin de producir condiciones ideales para las plantas que alberga. Esto se hace mediante iluminación LED, y como resultado se acelera el crecimiento del cultivo. Además también monitoriza el riego de las plantas y posee sensores de temperatura para controlar las condiciones ambientales. El sistema también incluye conectividad *bluetooth* con teléfono inteligente y una aplicación para monitorizar los datos del sistema (Ilustración 3).

Ilustración 3. Aplicación móvil[3] .

Este sistema se centra sobre todo en mejorar la producción de los cultivos a baja escala; para obtener alimentos aptos para consumo en el menor tiempo posible en ámbito doméstico.

2.1.3 Blue Marble

Blue Marble [4] está pensado para jardines grandes de exterior. El sistema se compone de diversos módulos que pueden adquirir-se y funcionar por separado. Los módulos no necesitan estar ubicados en el mismo punto.

Ilustración 4.Módulo controlador[4].

[2] fuente: http://www.growiee.com/
[3] fuente: http://www.growiee.com/

Estos módulos pueden ser aspersores, sensores o controladores. Este último se encarga de controlar todo el sistema y de gestionar las comunicaciones.

Además de las comunicaciones entre los elementos del sistema también existe una aplicación para monitorizar los parámetros del jardín. La comunicación con dicha aplicación la hace el módulo controlador a través de Wi-Fi. La aplicación también permite configurar algunos parámetros del sistema.

Ilustración 5. Módulo aspersor[5].

Cabe destacar que los módulos no necesitan alimentación eléctrica ya que disponen de pequeñas placas fotovoltaicas para obtener la energía necesaria. La excepción es el módulo controlador que sí necesita alimentación.

En conclusión se trata de un sistema que se centra únicamente en el control de la irrigación del jardín, tomando en cuenta parámetros cómo la humedad la temperatura y la luz. Está pensado para jardines exteriores de usuarios particulares.

2.1.4 *BYXAS Flower Pot* FP-188

BYXAS flower Pot [5] es un sistema de jardines en miniatura para interior. Sus funciones son la automatización del riego y el control del crecimiento mediante iluminación LED.

Ilustración 6. Maceta Byxas[6].

Este sistema cuenta con una interfaz de botones físicos. Dicha interfaz permite programar el riego y la iluminación con los LEDS. El sistema se alimenta de una toma de corriente estándar. No cuenta con ningún tipo de conectividad con el exterior. El fabricante no ofrece más datos sobre el producto.

---

[4] Fuente: http://bluemarbleirrigation.com/
[5] Fuente: http://bluemarbleirrigation.com/
[6] Fuente: http://www.ricodigital.com/byxas-multifunctional-intelligent-p-188-a

2.1.5 *Parrot flower sensor*.

Parrot flower sensor [6] es un sistema de monitorización de cultivos. Se encarga de detectar cuatro parámetros (humedad, temperatura, radiación solar y fertilizante en el suelo). A partir de estas medidas envía alertas al usuario.

Ilustración 7. Sensor *Parrot flower*[7].

La principal característica de este sistema es que actúa solo como informador. No realiza ningún tipo de acción física sobre el cultivo que monitoriza. Sólo se encarga de detectar los parámetros que pueden afectar al crecimiento de las plantas. A partir de los datos recopilados envía información a la aplicación móvil en forma de alertas. Este sistema puede utilizarse tanto en macetas como en suelo corriente. Es imprescindible tener la app móvil instalada ya que no cuenta con ninguna otra salida de información.

2.1.6 Nimbus Pot

Nimbus Pot [7] es un sistema de riego automático que basa su funcionamiento en principios físicos. Consta de una maceta diseñada específicamente para posibilitar su funcionamiento y un depósito vertical de agua.

Ilustración 8.Sección de una maceta NIMBUS[8] .

Este sistema no cuenta con ningún tipo de microcontrolador y tampoco dispone de comunicación con el exterior. En cambio presenta la ventaja de que no necesita alimentación, ya que no dispone de ningún tipo de elemento o actuador [8] eléctrico.

[7] Fuente: http://www.parrot.com/usa/products/flower-power/
[8] Fuente: http://www.mysmahome.com/REVIEWS/2638/product-introduction-more-than-your-regular-plant-pot-nimbus-offers-intelligent-watering-system.aspx

Otra característica interesante es que se encarga de alternar ciclos de riego con ciclos secos. Esto permite mantener en un buen estado las raíces de la planta, permitiendo que se oxigene la tierra.

Para utilizar este sistema sólo debemos ocuparnos de rellenar el depósito de agua cuando sea necesario. Este depósito tiene una duración aproximada de 4 meses. Este sistema es apto para su uso tanto en exteriores como interiores.

2.1.7 DIY *Smart Plant pot*

DIY Smart Plant pot [9] es un proyecto *open-source* que integra una placa *Arduino* [10] y diferentes sensores en una maceta. El objetivo de este sistema es monitorizar los parámetros de humedad temperatura y luz. A partir de los datos recopilados envía alertas al usuario por *Tweeter*.

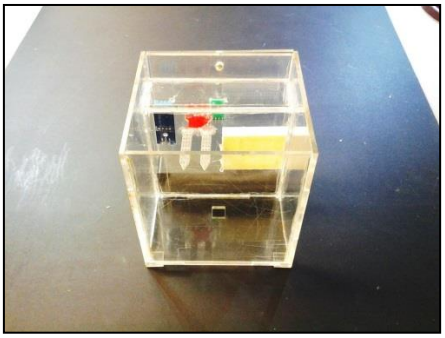

Ilustración 9. Maceta DIY Smart *Plant pot*[9].

Este sistema actúa sólo como informador y no dispone de ningún tipo de actuador físico. La comunicación con el exterior se realiza vía Wi-Fi; que viene integrado en el microcontrolador.

El sistema cuenta con su propia maceta, pero puede integrarse en otras macetas modificando sólo el cableado. Además también cuenta con un Led RGB para ofrecer información al usuario sobre que está haciendo el microcontrolador.

El envío de alertas por *Twitter* se hace utilizando una plataforma de Internet dedicada a monitorizar datos: *ThingSpeak*. Dicha plataforma se sincroniza por Internet con el sistema y se encarga de enviar las alertas por *Twitter*. Esta plataforma también permite mostrar gráficas de los datos y descargar la información en formato Excel.

[9] Fuente: http://www.instructables.com/id/DIY-Smart-Plant-pot/?ALLSTEPS

2.1.8 Open Garden.

Open Garden [11] es un proyecto *Open Source* para automatizar el control de jardines. Permite monitorizar factores cómo la humedad, la temperatura del aire y del suelo y la cantidad de luz recibida.

Ilustración 10.Kit de montaje Open Garden para exteriores[10].

Esta plataforma cuenta con tres tipos de kits: orientados al cultivo en exteriores, orientados a cultivo en interiores y orientados a cultivo hidropónico. Todos los kits incluyen cableado, cajas para montar los componentes, sensores de luz, agua y humedad y el *Shield* propio de esta plataforma. Además según el producto elegido se proporcionan distintos tipos de sensores; por ejemplo el kit para cultivo hidropónico cuenta con un sensor de PH. Cada pack también cuenta con actuadores específicos, por ejemplo el kit para interior cuenta con bomba de agua, mientras que el kit pata exteriores cuenta con aspersores.

Dentro del sistema, los componentes se agrupan en nodos. Los nodos son puntos de procesamiento encapsulados en las cajas que se proporcionan. Cada nodo cuenta con sus propios sensores y comunicaciones. También hay que destacar que las cajas de los nodos cuentan con células fotovoltaicas, permitiendo reducir la necesidad de alimentación eléctrica del conjunto del sistema.

Las comunicaciones se gestionan a través de un nodo encargado de conectarse con internet. Dicho nodo recibe las comunicaciones de los demás nodos a través de tecnología RFID (radiofrecuencia). El nodo encargado de gestionar las comunicaciones se conecta al servidor mediante Wi-fi, 3G o GPRS. Finalmente también dispone de aplicación móvil que permite visualizar el estado del sistema e interaccionar con los actuadores.

Este sistema se comercializa en paquetes de piezas, siendo el usuario responsable del montaje de todo el sistema, lo que requiere ciertos conocimientos por parte del usuario. El usuario también es responsable de configurar el servidor y de cargar todo el código en el hardware. Además el usuario debe adquirir por separado un (o diversos según montaje) microcontrolador Arduino.

[10] Fuente: https://www.cooking-hacks.com/open-garden-outdoor-1node-1gw

2.1.9 Garduino

Garduino [12] es un proyecto *Open Source* para el control y el automatismo de jardines. Este sistema se encarga de monitorizar los parámetros de humedad y temperatura. El sistema también es capaz de conectarse a internet para obtener información meteorológica. En función de la información recopilada, el sistema dispone de actuadores para controlar el riego.

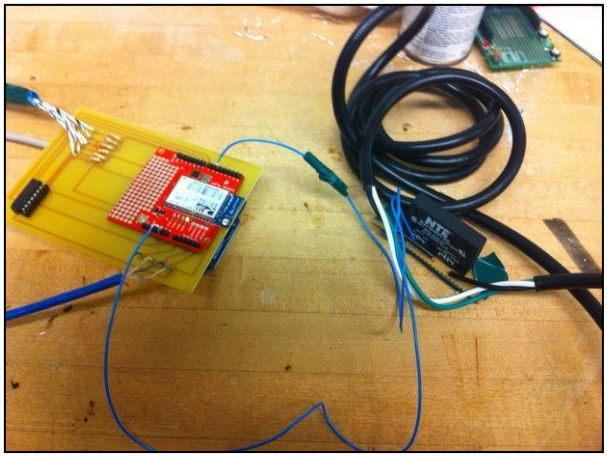

Ilustración 11.Proyecto *Garduino*[11].

El control del riego en este sistema se determina por la humedad detectada en el sustrato y la predicción meteorológica. Si la predicción meteorológica indica lluvias, el sistema suspenderá el riego para poder aprovechar el agua de las precipitaciones. Esto hace que el sistema esté indicado sólo para su uso en exteriores.

Este proyecto requiere un montaje a bajo nivel de una complejidad elevada. El usuario debe encargarse de ensamblar todas las piezas, desde los sensores a los actuadores, incluyendo el cableado y todos los detalles técnicos. El usuario también debe encargarse de cargar el código en el controlador.

El sistema no dispone de ningún tipo de aplicación para controlar el estado del mismo. Tampoco permite programar o forzar el riego del jardín. Para conectarse a Internet y obtener la información meteorológica se hace uso de un *shield* de Arduino que incorpora conectividad Wi-Fi.

El sistema no está comercializado, por lo que el usuario debe encargarse de adquirir todos los elementos por separado antes de poder proceder a su montaje.

[11] Fuente: http://www.instructables.com/id/Garduino-Automated-Gardening-System/?ALLSTEPS

2.1.10 PotPet: Pet-like Flowerpot Robot

PotPet [13] es una propuesta para crear macetas móviles. Este proyecto consiste en macetas inteligentes que pueden moverse para buscar una mayor exposición solar o para expresar que necesitan ser regadas.

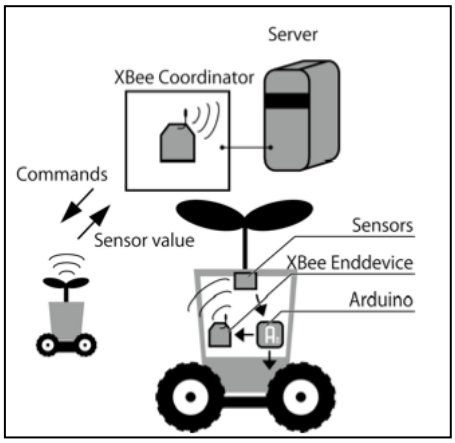

Ilustración 12. Arquitectura de la propuesta *PotPet*[12] .

El sistema detecta la humedad del sustrato y la cantidad de luz recibida. También dispone de sensores de ultrasonidos para detectar obstáculos y sensores de movimiento para detectar la presencia de personas.

PotPet no cuenta con actuadores para regar el cultivo. Sin embargo dispone de motores para moverse. El movimiento tiene dos funciones. La primera función es buscar una zona de mayor exposición a la luz solar. La segunda función es expresar las necesidades del cultivo realizando movimientos. Por ejemplo, cuando el sistema detecta que necesita ser regado y hay una persona cerca, se desplaza cerca de la persona para atraer su atención.

Para gestionar el movimiento se propone el uso de un controlador independiente que se encarga sólo de los motores. Dicho controlador encargado de los motores recibe las órdenes del microcontrolador principal a través de comunicaciones inalámbricas, empleando la tecnología Wi-Fi.

Para controlar el sistema, que puede tener múltiples unidades *PotPet* se propone centralizar las comunicaciones en un servidor. Las comunicaciones con este servidor se hacen vía tecnología Wi-Fi. Dicho servidor se encarga de gestionar la información global del sistema.

[12] Fuente: http://media.tumblr.com/b9d4f389dd668edac83d99b5797c893d/tumblr_inline_mhvbvjDqFX1qhx915.png.

2.2 Análisis de sistemas similares.

Ahora que se han estudiado los diferentes productos que existen en el mercado para solucionar el problema de la automatización de jardines, el siguiente paso es realizar un análisis de dichos sistemas.

En este apartado se presentan tablas comparativas para poder comparar las diferentes características de los sistemas estudiados.

2.2.1 Análisis cuantitativo

En este apartado se hará un análisis de las características cuantificables.

2.2.1.1 Características generales

Las características generales de los sistemas analizados se exponen en la Tabla 1, donde las columnas tienen el siguiente significado:

- Sistema: Nombre del sistema por el que se comercializa.
- Tamaño: Dimensiones del producto en cm.
- Consumo: Potencia eléctrica consumida para funcionamiento normal.
- Acel. Crec. (Aceleración crecimiento): El sistema permite reducir el tiempo normal que tarda un determinado cultivo en crecer.
- Hidropónico: El cultivo no tiene sustrato. Todos los nutrientes necesarios le llegan disueltos en agua.
- Precio: Precio de venta del sistema.
- Modularidad: El sistema se compone de partes independientes que pueden funcionar autónomamente o en conjunto con otros elementos del sistema.

A continuación se muestran los valores obtenidos en una selección de los sistemas más representativos de entre todos los sistemas localizados.

Sistema	Tamaño	Consumo	Acel. Crec.	Hidropónico	Precio	Modularidad
Citysens	115 x 30 cm	5 W	No	Si	89,00€	Si
Growiee	28 x 18 x 30 cm	10 W	Si	Si	100,00€	Si
Blue Marble	(1)	(1)	No	No	(2)	Si
BYXAS Flower Pot	30 x 35cm	(1)	Si	No	10 €	No
Parrot flower sensor	15 x 20 x 2 cm	(1)	No	(1)	40 € (3)	No
Nimbus Pot	(1)	(4)	No	No	35€	No
DIY Smart Plant pot	(1)	(1)	No	No	(1)	No
Open Garden	(1)	(1)	(5)	(5)	199,00€	Si
Garduino	(1)	(1)	No	No	(6)	No
PotPet	(7)	(7)	(7)	No	(7)	(7)

Tabla 1. Características generales de los sistemas analizados

(1) Dato no disponible por parte del fabricante.
(2) No comercializado.
(3) Precio mínimo.
(4) Sistema no alimentado.
(5) Característica disponible sólo en algunos modelos.
(6) El precio depende de los componentes adquiridos para montar el sistema.
(7) Se trata de un proyecto de investigación.

2.2.1.2 Características técnicas

Las características técnicas de los sistemas analizados se exponen en la Tabla 2 cuyas columnas tienen el siguiente significado.

- Sistema: nombre del sistema por el que se comercializa.
- Microcontrolador: Nombre comercial de microcontrolador integrado.
- Con. Int. (Conectividad interna): Capacidad de comunicación entre elementos internos de un producto o entre unidades de un mismo producto. En caso de disponer de este tipo de conectividad se especifica la tecnología empleada.
- Con. Ext. (Conectividad externa): Tecnología empleada para las comunicaciones con el exterior.
- Inteligente: El sistema es capaz de tomar decisiones y encargar-se del mantenimiento de los cultivos sin necesidad de acción humana.

Sistema	Microcontrolador	Con. Int.	Con. Ext.	Inteligente
Citysens	(1)	No	No	No
Growiee	(1)	No	Bluetooth	Si
Blue Marble	(1)	Wi-Fi	Wi-Fi	Si
BYXAS Flower Pot	(1)	No	No	No
Parrot flower sensor	(1)	No	Bluetooth	Si
Nimbus Pot	(2)	No	No	No
DIY Smart Plant pot	Arduino Cactus Micro	No	Wi-Fi	Si
Open Garden	Arduino	Si	Wi-FI,3G,GPRS (3)	Si
Garduino	Arduino	No	Wi-Fi	Si
PotPet	Arduino	Wi-Fi	Wi-fi	Si

Tabla 2. Características técnicas.

(1) Dato no disponible por parte del fabricante.
(2) El sistema no cuenta con ningún tipo de microcontrolador.
(3) El sistema puede utilizar diversos tipos de tecnologías para las comunicaciones.

2.2.2 Análisis cualitativo

En este apartado se analizan aquellas características que no pueden medirse con un número. Se comparan aspectos como diseño, facilidad de uso, etc. Para ello, de nuevo se organiza la información en una tabla.

En la tabla adjunta se expone una comparativa de las características cualitativas de los sistemas. Dentro de la tabla las columnas tienen el siguiente significado:

- Sistema: se expone el nombre del sistema.
- Diseño: Mide el aspecto y la estética del producto
- Facilidad de uso: Mide el esfuerzo que tiene que realizar el usuario para utilizar el producto.
- Facilidad de instalación: Mide la cantidad de esfuerzo que se necesita para preparar el sistema antes de poder utilizarlo.
- Facilidad de reubicación: Mide la dificultad para cambiar el sistema de lugar.
- Capacidad de expansión: Mide la posibilidad de añadir más componentes al sistema.

Para la valoración de las características se ha empleado una escala de tres valores con los siguientes códigos:

*** Muy adecuado
** Adecuado
* Poco adecuado
- No valorable

Sistema	Diseño	Facilidad de uso	Facilidad de instalación	Facilidad de reubicación	Capacidad de expansión
Citysens	***	***	***	***	*
Growiee	**	***	***	***	*
Blue Marble	*	*	*	*	***
BYXAS Flower Pot FP-188	*	*	***	***	*
Parrot flower sensor	***	**	***	***	-
Nimbus Pot	**	***	**	*	-
DIY Smart Plant pot	-	*	*	***	**
Open Garden	**	*	*	**	***
Garduino	-	*	*	*	**
PotPet	-	-	-	-	-

Tabla 3. Características cualitativas.

2.3 Síntesis

Se han analizado los diferentes productos que existen en el mercado con un enfoque similar al de este proyecto. Tras el estudio de los sistemas presentados se obtienen una serie de características deseables para nuestro proyecto. Se empleará el código CAxx para diferenciar cada característica.

CA1: Monitorización del entorno: temperatura, humedad del aire, humedad del suelo, cantidad de luz recibida.

CA2: Actuación sobre el entorno: control de riego y de depósito de agua.

CA3: Búsqueda de luz: cambio de ubicación para buscar la luz del sol y evitación de obstáculos.

CA4: Conectividad: capacidad de conexión y comunicación con otros sistemas similares y con sistemas exteriores.

2.4 Tecnología a emplear

Para satisfacer estas necesidades, parece razonable utilizar algún tipo de microcontrolador que vaya montado en el producto, ensamblado en su interior. Entre la gran variedad de soluciones disponibles en el mercado, la plataforma Arduino ofrece una gama amplia de microcontroladores a precios competitivos.

Arduino es un proyecto Open-Source que cuenta con su propio microcontrolador y su propio lenguaje de programación. El microcontrolador puede ser programado desde un PC y puede ejecutar código de cualquier tipo. Sólo necesita alimentación y puede interactuar directamente con sensores y actuadores. Además también posee capacidad de conectividad ya sea mediante conexión inalámbrica (será necesario comprar piezas adicionales) o conexión cableada. Para poder interactuar con el entorno utilizaremos sensores con los que Arduino es compatible. Hay una amplia gama de sensores de todo tipo en el mercado, desde sensores de temperatura o humedad hasta sensores de huella dactilar. Gracias a todo esto, Arduino resulta ser la plataforma ideal para llevar a cabo el proyecto.

En el apartado de comunicación; sería interesante disponer de algún tipo de aplicación para poder controlar el sistema de manera remota. Inicialmente se plantea la necesidad de desarrollar una aplicación en Android. Android es un lenguaje de programación compatible en la gran mayoría de teléfonos inteligentes. También sería interesante contar con alguna interfaz web, de modo que el sistema sea accesible desde un amplio rango de dispositivos. Esta parte del desarrollo será llevada a cabo por otro miembro participante en el proyecto.

Otra característica interesante sería poder comunicar los módulos entre sí. Así, los distintos módulos podrían intercambiar información e incluso llegar a cooperar.

Para gestionar todas estas comunicaciones puede hacerse uso de la tecnología Wi-fi. Las ventajas de esta tecnología son que permite conexión cifrada, y que es compatible con la gran mayoría de teléfonos inteligentes y ordenadores en el mercado. No obstante, será necesario comprar hardware adicional para dotar al producto de conectividad Wi-fi. La implementación y el desarrollo de conectividad entre módulos será llevada a cabo por otro miembro del proyecto.

2.5 Conclusiones

En este capítulo se ha dado un repaso a las soluciones que existen en el mercado al problema de la automatización de cultivos. Tras analizar una muestra representativa de los productos disponibles se observa que la tendencia de estos sistemas es monitorizar los parámetros que influyen en el crecimiento de la planta, sin realizar acciones para modificar las condiciones observadas. No obstante, también existen sistemas que cuentan con actuadores. Estos sistemas se centran sobre todo en el control del riego.

Se comprueba también que algunos sistemas cuentan con su propia aplicación móvil para permitir una mayor accesibilidad al usuario final. Disponer de una aplicación móvil es de gran importancia, ya que mejora enormemente la interacción con el usuario y mejora la accesibilidad del producto a todos los públicos.

Una vez concluido el análisis, se han extraído las características deseadas en el producto final. El sistema deberá ser capaz de moverse, de regarse y de comunicarse con el exterior.

En el apartado técnico, se ha optado por utilizar la plataforma Arduino para llevar a cabo el desarrollo, por su bajo coste y su capacidad de funcionamiento autónomo.

Para realizar las comunicaciones se ha decidido utilizar la tecnología Wi-fi. Se hará uso de Wi-fi para gestionar las comunicaciones tanto entre el producto y la aplicación, como entre las demás unidades del producto.

En el siguiente capítulo se concretarán todas estas características deseables de manera formal. Para ello, se redactará la especificación de requisitos del proyecto, basándose en el estándar IEE 830.

3 Especificación de requisitos

Tras haber realizado el estudio de los sistemas similares y haber extraído la información necesaria, el siguiente paso es disponer la información de manera más formal. En este capítulo vamos a concretar todas las funcionalidades y los requisitos del proyecto. Para ello, utilizaremos como guía la última versión del estándar IEE 830.

3.1 Introducción

3.1.1 Propósito

El objetivo de esta especificación de requisitos (ERS) es determinar de manera clara e inequívoca las capacidades de este proyecto. Se determinará que puede o que no puede hacer el proyecto. También se determinará el personal involucrado en el desarrollo, así como las dependencias y suposiciones del proyecto.

3.1.2 Alcance

FlowerPot es un sistema de automatización de jardines. Es capaz de determinar las condiciones ambientales y las necesidades del cultivo que alberga. Puede monitorizar la temperatura, la humedad y la cantidad de luz recibida. También puede moverse para buscar zonas de mayor exposición solar o para protegerse de los rayos del sol en caso de necesidad. Además también cuenta con un depósito de agua que permite regar el cultivo que alberga.

El sistema también dispone de capacidad para comunicarse con otros módulos, de modo que es posible compartir información con otras unidades similares. También dispone de comunicación con el usuario a través de una aplicación móvil. Estas comunicaciones están centralizadas en un servidor que hace de intermediario entre la aplicación y el macetero.

3.1.3 Personal involucrado

Este apartado muestra la información de todas las personas involucradas en el proyecto. Esta información se presenta en forma de tablas con información de interés sobre los participantes.

Nombre	Axel Guzmán Godia
Rol	Desarrollador automatización.
Categoría profesional	Estudiante
Responsabilidades	Desarrollar el código Arduino y montar el Hardware
Información de contacto	axguzgo@inf.upv.es
Aprobación	-

Tabla 4. Miembro Axel Guzmán Godia

Nombre	Laia Ferrando Ferragut
Rol	Desarrollador Aplicación móvil.
Categoría profesional	Estudiante
Responsabilidades	Diseño aplicación móvil
Información de contacto	laferfe@inf.upv.es
Aprobación	-

Tabla 5. Miembro Laia Ferrando Ferragut

Nombre	Israel Beltrán Arias
Rol	Desarrollador comunicaciones
Categoría profesional	Estudiante
Responsabilidades	Implementación comunicaciones entre módulos.
Información de contacto	-
Aprobación	-

Tabla 6. Miembro Israel Beltrán Arias

Nombre	Alejandro Delgado
Rol	Desarrollador comunicaciones
Categoría profesional	Estudiante
Responsabilidades	Implementación comunicaciones módulo-servidor
Información de contacto	-
Aprobación	-

Tabla 7. Miembro Alejandro Delgado

Nombre	José Luis Poza Luján
Rol	Supervisor.
Categoría profesional	Profesor Contratado Doctor
Responsabilidades	Supervisar proyecto.
Información de contacto	joplu@disca.upv.es
Aprobación	-

Tabla 8. Miembro José Luis Poza Luján.

Nombre	Juan Luís Posadas Yagüe
Rol	Supervisor.
Categoría profesional	Profesor Contratado Doctor
Responsabilidades	Supervisar proyecto.
Información de contacto	jposadas@disca.upv.es
Aprobación	-

Tabla 9. Miembro Juan Luís Posadas Yagüe.

3.1.4 Definiciones, acrónimos y abreviaturas

Este apartado define la terminología a emplear en el documento. Se incluyen definiciones de algunos conceptos clave, acrónimos y también se especifica el significado de algunos términos dentro del documento.

Módulo: En este proyecto la palabra módulo hace referencia a una unidad del producto montada en una maceta. Así pues un módulo contiene un microcontrolador Arduino, sensores, un depósito de agua y actuadores.

Proyecto: El uso de la palabra proyecto hace referencia al sistema global que se está desarrollando, del cual este documento sólo implementa una parte limitada. El término proyecto abarca por desde la implementación de los módulos, hasta las comunicaciones y el desarrollo del servidor central.

Producto: Este término hace referencia a todo el sistema, desde la aplicación móvil hasta los maceteros. Este término es sinónimo del término proyecto definido anteriormente.

Sistema: El uso de la palabra sistema dentro de este documento hace referencia a la parte en concreto del proyecto que se está desarrollando. Esto quiere decir que sistema comprende la implementación del código Arduino encargado de gestionar los sensores y actuadores, y el hardware necesario para llevar a cabo las funcionalidades definidas

en este documento. Se excluyen por tanto del significado de sistema otros aspectos como por ejemplo las comunicaciones, la aplicación móvil o el control del servidor.

Maceta: Emplearemos el término maceta para como sinónimo del término módulo definido anteriormente.

Variable de comunicación: Uno de los objetivos del sistema es poder ejecutar órdenes que le lleguen través de la capa de comunicaciones. Emplearemos este nombre para designar únicamente aquellas variables que nos llegan desde la capa de comunicaciones.

3.2 Descripción general

3.2.1 Perspectiva del producto

El sistema desarrollado en este documento forma parte de un proyecto mayor. El proyecto está compuesto por una serie de capas que van desde el nivel más bajo, en contacto con el entorno, hasta el nivel superior, en contacto con el usuario final. Entre estas partes se encuentran el desarrollo de una aplicación móvil para comunicarse con el usuario, un servidor central que gobierne los módulos y las comunicaciones entre módulos. La organización del proyecto puede verse en la siguiente ilustración:

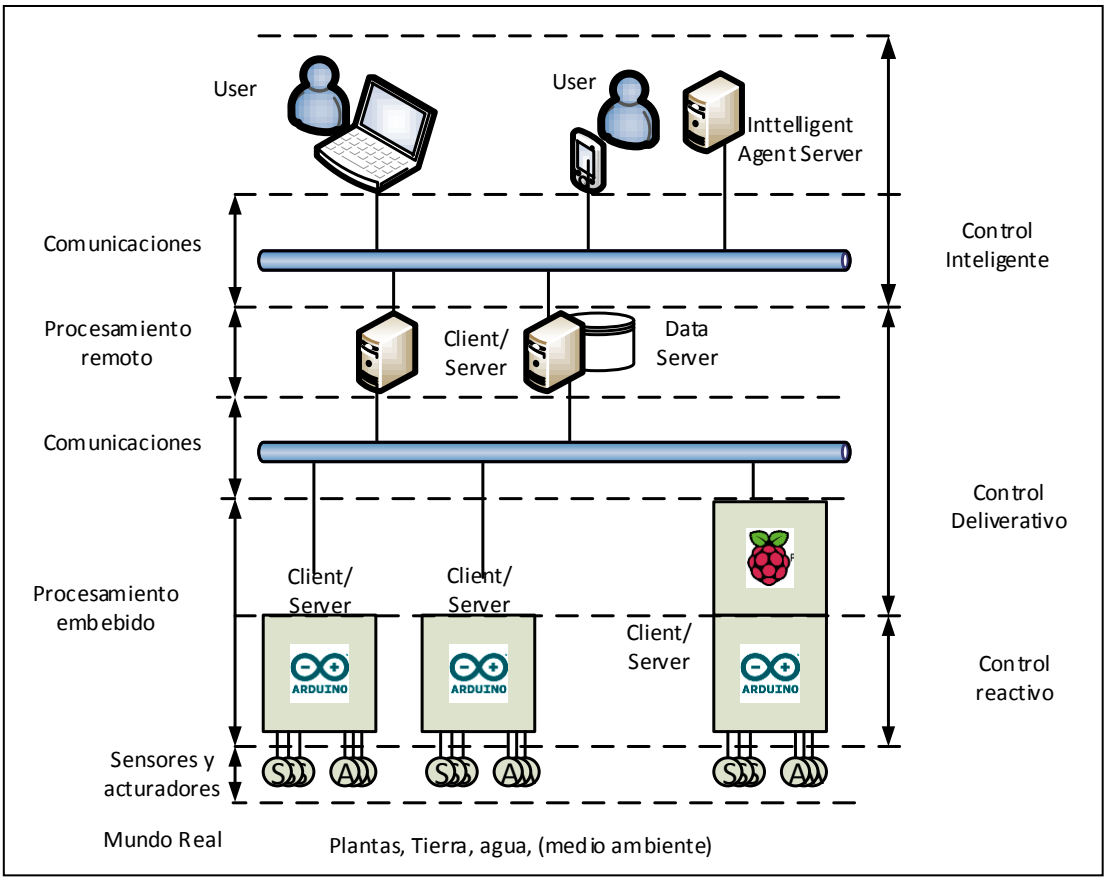

Ilustración 13.Diagrama de capas del proyecto.[13]

En la parte superior están los usuarios, que interactuaran mediante la web o la aplicación móvil. Si continuamos descendiendo encontramos los servidores,

[13] Fuente: Elaboración propia.

encargados de gestionar el conjunto. Más abajo llegamos al hardware que irá montado en las macetas.

Dentro de la Ilustración 13, el sistema desarrollado en este documento se sitúa en la parte inferior. Concretamente se cubre la capa de sensores y actuadores y la capa de control reactivo, excluyendo la parte de comunicaciones. A partir de este punto las demás partes del proyecto serán desarrolladas por los otros integrantes del equipo.

3.2.2 Funcionalidades del producto

El objetivo de esta sección es presentar las funcionalidades que debe incorporar el sistema sin entrar en detalle en cómo se implementaran dichas funcionalidades.

Control y monitorización de un cultivo: el sistema debe ser capaz de monitorizar los parámetros que afectan al crecimiento de un cultivo y tomar decisiones a partir de esta información. El sistema puede gestionar el riego del cultivo a partir de la humedad del sustrato. La gestión del riego requiere realizar un control sobre el depósito de agua. Además, el sistema debe ser capaz de desplazarse para aumentar o reducir la exposición solar.

Comunicación: Las macetas deben ser capaces de comunicarse con otros módulos y con el servidor. Esta comunicación incluye tanto el envío de la información local cómo la recepción de órdenes. La implementación del envío y recepción de dichas comunicaciones será desarrollada por otro miembro del proyecto. El sistema deberá determinar qué hacer con los mensajes que le lleguen, y decidir si se puede o no realizar ciertas acciones.

3.2.3 Requisitos funcionales

En este apartado vamos a definir los casos de uso del sistema. Para ello, haremos uso de diagramas UML.

Pueden distinguirse dos grupos de funcionalidades generales. Por un lado están las funcionalidades que el usuario final del proyecto solicita explícitamente. Estas funcionalidades son las órdenes que el usuario manda a través de la interfaces (aplicación, web) y que finalmente se ejecutan en las macetas. Los casos de uso asociados son tres: Controlar riego, Mover y Actualizar valores. La siguiente imagen ilustra los casos de uso del Usuario final:

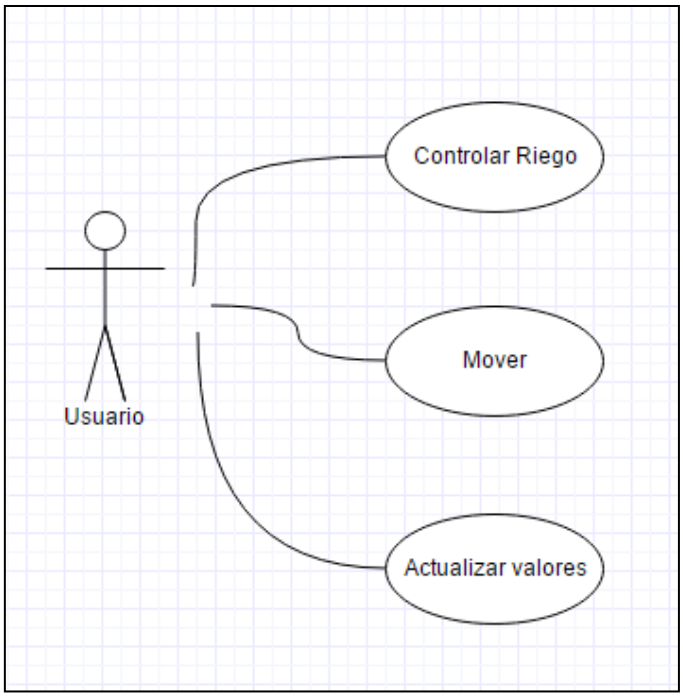

Ilustración 14.Casos de uso Usuario[14]

Estos casos de uso modelan las acciones que el usuario podrá solicitar. Estas órdenes pasaran por las otras capas del proyecto (comunicaciones) hasta llegar al módulo correspondiente.

Por otro lado, están las funcionalidades de control. Los casos de uso que se incluyen aquí no los activa el usuario, pero forman parte del control realizado por el sistema. A continuación se muestra un diagrama con los casos de uso asociados al control:

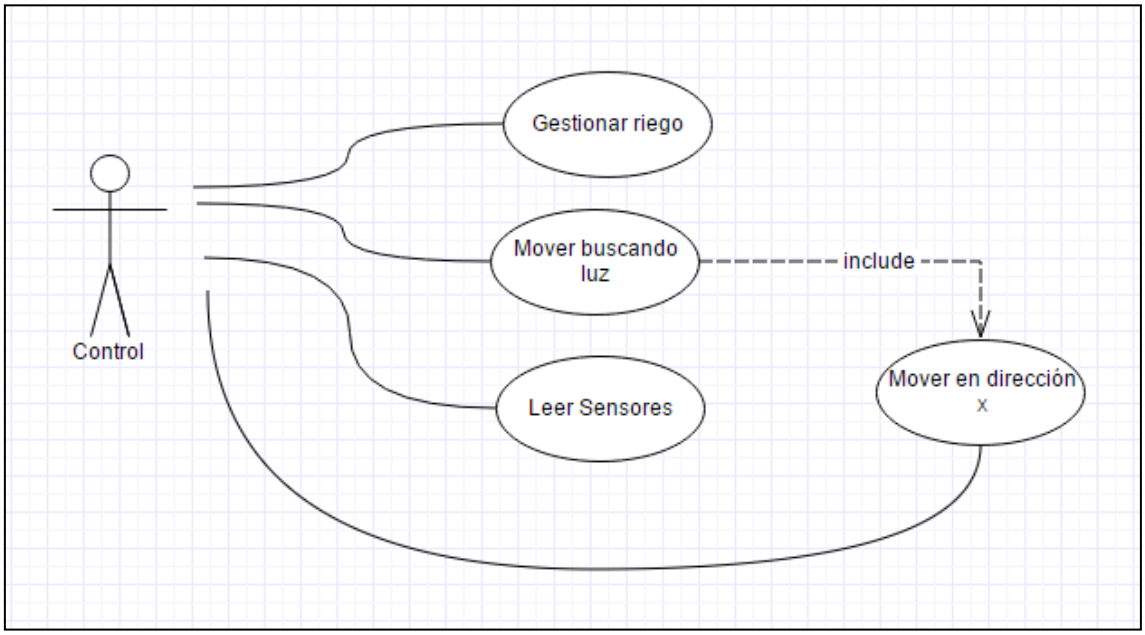

Ilustración 15.Casos de Uso Control.[15]

[14] Fuente: elaboración propia.

A continuación se explican más detalladamente los casos de uso.

3.2.3.1 CU01: Controlar Riego

Este caso de uso implementa la funcionalidad de activar y desactivar el riego. El usuario puede solicitar al sistema que se active o que se pare el riego. En ambos casos, el sistema comprueba si los niveles del depósito son correctos para permitir realizar dichas acciones. En caso de que el nivel del depósito sea inferior al mínimo necesario no se permitirá activar el riego.

3.2.3.2 CU02: Mover en dirección X

Este caso de uso implementa la funcionalidad de controlar el movimiento de la maceta. El usuario puede solicitar a través de las interfaces que la maceta se mueva en una dirección determinada. El sistema realizará el control necesario para comprobar si existen obstáculos en la dirección de movimiento deseada. En caso de encontrar algún obstáculo la maceta no se moverá.

3.2.3.3 CU03: Actualizar Valores

Cuando el usuario está utilizando la aplicación web o la aplicación móvil, tiene la posibilidad de refrescar los valores de los sensores. De este modo se solicita al sistema que se vuelvan a leer los sensores para obtener un contenido lo más actualizado posible.

3.2.3.4 CU04 Gestionar Riego.

Uno de los objetivos fundamentales del cuidado de una planta es controlar el riego. Para ello se comprueban los valores de humedad del sustrato. Dependiendo del valor de humedad se activa el riego. Para poder activar el riego se requiere realizar el control del depósito.

3.2.3.5 CU05 Mover buscando luz

A través del valor de los sensores de luz y del control interno se determina en qué dirección hay que moverse para conseguir mayor exposición a la luz. Según la dirección determinada se utiliza el caso de uso encargado de gestionar el movimiento.

3.2.3.6 CU06 Leer Sensores

El sistema realiza una lectura secuencial de todos los sensores de que dispone. Los valores leídos se utilizan para realizar el control y se escriben en la memoria EEPROM cada cierto tiempo.

3.2.4 Requisitos de los usuarios

En esta sección se determinan las características necesarias para poder interactuar con esta parte del sistema. Distinguiremos dos tipos de usuarios:

Básico: Este tipo de usuario sólo tiene acceso a las lecturas de los sensores. Para ello, podrá leer la información almacenada en la EPROM, que contiene los datos de los sensores. También podrá escribir en una zona reservada de la EPROM. Estas escrituras permitirán enviar órdenes desde las capas exteriores del proyecto.

Avanzado: Este usuario tiene acceso total al sistema. Es el usuario encargado de implementar las funcionalidades y del montaje. Por ello requiere conocimientos en circuitos electrónicos básicos, y conocimientos en programación.

---

[15] Fuente: Elaboración propia.

En la siguiente tabla se muestra una comparativa de los usuarios según las funcionalidades asociadas. Estas funcionalidades son las siguientes:

1. Leer EPROM (órdenes)
2. Leer sensores
3. Evaluar órdenes
4. Control motores: (mover /detener)
5. Control riego: (iniciar/detener)
6. Escribir EPROM (resultados)

Usuario	Func.1	Func.2	Func.3	Func.4	Func.5	Func.6
Básico		X				
Avanzado	X	X	X	X	X	X

Tabla 10.Funcionalidades según usuario.

3.2.5 Restricciones de diseño

Todo el sistema está desarrollado bajo la plataforma Arduino. Esto hace un requisito indispensable el utilizar los microcontroladores propios de dicha plataforma. En consecuencia, tanto el código desarrollado como las posibles ampliaciones posteriores deben desarrollarse en código Arduino. Además, teniendo en cuenta que se utiliza la EEPROM para implementar la persistencia y las comunicaciones, es necesario utilizar algún modelo de microcontrolador que disponga de dicha memoria. Sin la presencia de una memoria EEPROM, no funcionarán las funcionalidades de comunicación.

Esta consideración es de especial importancia si se desea utilizar alguna placa Arduino que no disponga de memoria EEPROM. No obstante, puede utilizarse un chip de memoria EEPROM externo para satisfacer este requisito.

Otra característica que debe tenerse en cuentas es el número de entradas de que dispone el controlador. Se va a hacer uso de un número elevado de conexiones por lo que será necesario utilizar un microcontrolador con suficientes entradas/salidas para poder interactuar con todos los sensores.

3.2.6 Suposiciones y dependencias

En este apartado se declaran las suposiciones que se han asumido a la hora de diseñar el producto y las dependencias que, si se modifican, pueden alterar el resultado final.

3.2.6.1 Suposiciones

A continuación se describen las suposiciones que se han tenido en cuenta para el desarrollo del sistema:

- Se asume que el sistema trabajará en entornos con terreno constante, sin grandes desniveles. Un terreno demasiado irregular podría dificultar e incluso imposibilitar el correcto funcionamiento de las funcionalidades de movimiento.
- Se supone un ambiente libre de interferencias que dificulten o impidan el correcto funcionamiento de las comunicaciones Wi-fi. En caso de que haya

dificultades o impedimentos para la conectividad, el sistema seguirá funcionando pero no será capaz de recibir o enviar información.

3.2.6.2 Dependencias

Esta sección describe las dependencias necesarias para el correcto funcionamiento del sistema.

El sistema forma parte de un proyecto mayor, por lo que para cubrir todas las funcionalidades descritas en este documento es necesaria la presencia de los demás elementos del proyecto completo. (Véase Ilustración 13). No obstante, en caso de no disponer de los demás elementos del proyecto, se mantienen algunas funcionalidades, cómo el control de irrigación, de entorno (temperatura y humedad) y el seguimiento de luz.

3.2.7 Evolución previsible del sistema

En este apartado se exponen algunas características que no están presentes en el producto, pero que es probable que se incorporen en un futuro.

Una funcionalidad que podría añadirse más adelante es la capacidad de obtener información meteorológica. A partir de esta información se pueden tomar decisiones como no regar el cultivo si se han previsto precipitaciones o emitir una alerta al usuario si se avecinan condiciones extremas que puedan dañar el producto. Esta funcionalidad podría implementarse a nivel del servidor, de modo que el servidor se encargara de obtener y procesar los datos meteorológicos. Posteriormente, los módulos solicitarían al servidor dicha información, obteniendo información ya procesada.

Por otro lado, el sistema desarrollado se basa en el control reactivo. Esto significa que el sistema percibe cierta información y en base a estos estímulos realiza acciones. Sin embargo, la tendencia actual es sustituir este tipo de control por agentes inteligentes. Esto permitirá lograr un control más dinámico y mejorar las capacidades de adaptación del sistema.

La inclusión de un agente inteligente en el sistema puede aportar ventajas en el comportamiento del producto, así como capacidades de aprendizaje. Sin embargo, esta modificación también requeriría mayor capacidad de cómputo, por lo que es también previsible que se incorpore en un futuro algún controlador con mayores capacidades técnicas. Una posibilidad bastante factible sería incluir un módulo *Raspberry*.

Finalmente, también existe la posibilidad de implementar nuevas funcionalidades de manera externa. Esto es posible gracias a la capacidad para recibir órdenes. Por ejemplo podría crearse un software que envíe a cada hora del día una serie de indicaciones para mover los maceteros, o que active el riego a una determinada hora. Las posibilidades son muy grandes en este aspecto.

3.3 Requisitos Específicos

3.3.1 Requisitos de interfaces externas

3.3.1.1 Interfaces Hardware:

Para poder interactuar con el producto es necesario disponer de un dispositivo móvil con conexión Wi-Fi. Si bien ésta es la configuración más habitual, tiene cabida cualquier dispositivo capaz de ejecutar la aplicación móvil del proyecto (programada en Android) y con capacidad de conexión Wi-FI.

Otra opción es acceder a la información mediante la página web. Para poder acceder por este método sería suficiente con cualquier dispositivo capaz de conectarse a Internet y visualizar páginas web. Caben muchas posibilidades en esta opción, desde ordenadores y teléfonos inteligentes hasta tabletas digitales o incluso televisores inteligentes con capacidades para navegar por la web.

3.3.1.2 Interfaces software:

La interacción con las demás partes del sistema se hará mediante el uso de variables compartidas. Para ello se utilizará la memoria EPROM disponible en Arduino. La memoria se dividirá en dos zonas. Una destinada al control interno y otra a las comunicaciones.

3.3.2 Requisitos funcionales

En este apartado se presenta una descripción detallada de las funcionalidades del sistema. Se describen las entradas necesarias, el proceso realizado y las salidas.

Número requisito	01
Nombre	Leer sensores
Inputs	Sensores
Outputs	Valores de los sensores escritos en la EPROM
Proceso	Leer sensores y escribir en la posición correspondiente de la EPROM
Restricciones	-

Tabla 11. Caso de uso Leer sensores.

Número de requisito	02
Nombre	Evitar obstáculo
Inputs	Sensores ultrasonidos
Outputs	Evento obstáculo
Proceso	Lectura secuencial de todos los sensores de ultrasonidos disponibles
Restricciones	

Tabla 12.Caso de uso Evitar obstáculo.

Número de requisito	03
Nombre	Activar Riego
Inputs	Petición usuario (Variable EPROM?)
Outputs	Activar bomba riego
Proceso	Comprobar nivel depósito Activar bomba
Restricciones	Si el nivel del depósito es bajo no se

	activará el riego.

Tabla 13.Caso de uso Activar riego.

Número de requisito	04
Nombre	Parar riego
Inputs	Petición usuario (Variable EPROM?)
Outputs	Detener bomba riego
Proceso	Detener bomba riego
Restricciones	-

Tabla 14.Caso de uso parar riego.

Número de requisito	05
Nombre	Mover en la dirección X
Inputs	Posición deseada
Outputs	Movimiento motores
Proceso	Comprobar obstáculos Activar motores en función de la posición deseada
Restricciones	Si hay un obstáculo no se permite el movimiento en la dirección donde está el obstáculo

Tabla 15.Caso de uso Mover en la dirección X.

Número de requisito	06
Nombre	Detener movimiento
Inputs	Control usuario (Variable en EPROM?)
Outputs	Parar motores
Proceso	Desactivar motores
Restricciones	-

Tabla 16.Caso de uso Detener movimiento.

Número de requisito	07
Nombre	Moverse buscando/huyendo de la luz
Inputs	Valor de exposición /Sensores luz
Outputs	Posición destino/Accionar motores
Proceso	Según si estamos buscando o huyendo de la luz cambiar la dirección del movimiento a la ubicación del sensor que recibe más/menos luz.
Restricciones	Si se ha detectado un obstáculo el movimiento parará

Tabla 17.Caso de uso Moverse buscando luz.

Número de requisito	08
Nombre	Control depósito
Inputs	Sensores de nivel de agua
Outputs	Activación bomba llenado/vaciado
Proceso	Determinar si el nivel leído en los sensores está por encima o por debajo de unos umbrales
Restricciones	-

Tabla 18.Caso de uso Control depósito

3.3.3 Requisitos no funcionales

En esta sección se describen todos aquellos requisitos que no están relacionados con las funcionalidades del sistema.

3.3.3.1 Requisitos Tecnológicos

En esta sección se definen las tecnologías necesarias para el desarrollo del producto sobre las cuales se va a hacer la implementación.

Dado que el sistema debe formar unidades independientes, se ha optado por utilizar la plataforma Arduino. El sistema contará con un controlador Arduino y el código será programado en el lenguaje propio de este entorno. Como el sistema tendrá que realizar control de motores, también se utilizará el circuito integrado L293D, diseñado para el control de motores.

Por otro lado, para obtener la información del exterior se utilizarán sensores de cinco tipos: humedad, temperatura, humedad del suelo, luz y ultrasonidos. La tecnología empleada en dichos sensores puede variar sin afectar al producto. Sin embargo es necesario que dichos sensores proporcionen la información por el mismo método, es decir, puedan leerse del mismo modo. Esto permite intercambiar sensores sin modificar el código.

3.3.3.2 Portabilidad

Migrar el sistema a otro controlador requiere volver a desarrollar todo el código. No afecta sin embargo que cambie la tecnología empleada en capas superiores, mientras se entiendan con la interfaz Wi-Fi

3.4 Conclusiones

En este capítulo se han convertido las características deseables obtenidas en el apartado anterior en requisitos formales. Se han definido todos los aspectos claves del proyecto así como las funcionalidades que se van a implementar. También se han definido las características de los usuarios del sistema y las capacidades de cada uno.

Ahora que se ha especificado toda la información del proyecto, el siguiente paso es abordar el diseño del sistema. En el siguiente capítulo se expondrán los diagramas de casos de uso y el hardware necesario para poder implementar el proyecto.

4 Diseño del sistema

4.1 Introducción

En este punto del proyecto, ya se ha dejado claro cuáles son los objetivos que debe cumplir el sistema desarrollado. El siguiente paso es determinar cómo se van a realizar estos objetivos.

Para ello, el siguiente paso es definirlas acciones que se llevarán a cabo en el sistema a bajo nivel.

Por otro lado también se concretará el hardware necesario para poder realizar todas las acciones necesarias. Para ello, se presentarán diversas alternativas que pueden funcionar con el sistema desarrollado.

4.2 Especificación

Para abordar el diseño del sistema es imprescindible adaptarse al modo de funcionamiento de Arduino. El código Arduino se basa en una fase de inicialización y un bucle que se ejecuta indefinidamente.

Cada vez que se enciende el microcontrolador se ejecuta primero la fase de inicialización, y a continuación se ejecuta constantemente la fase de bucle. Por ello es importante organizar las tareas que tiene que ejecutar el controlador de manera cíclica.

Nada más iniciar el sistema, se ejecutará la inicialización. En esta fase se realiza la lectura de la información que hay almacenada en la EPROM. Esto permitirá recuperar el estado de la última vez antes de apagar el controlador. También se realizará la configuración de los pines de Arduino, y la sincronización con los sensores que lo requieran.

A continuación, se realiza la fase de bucle, que es la más importante, pues es la que se ejecutará la mayor parte del tiempo. La siguiente imagen muestra el funcionamiento del sistema en la fase de bucle:

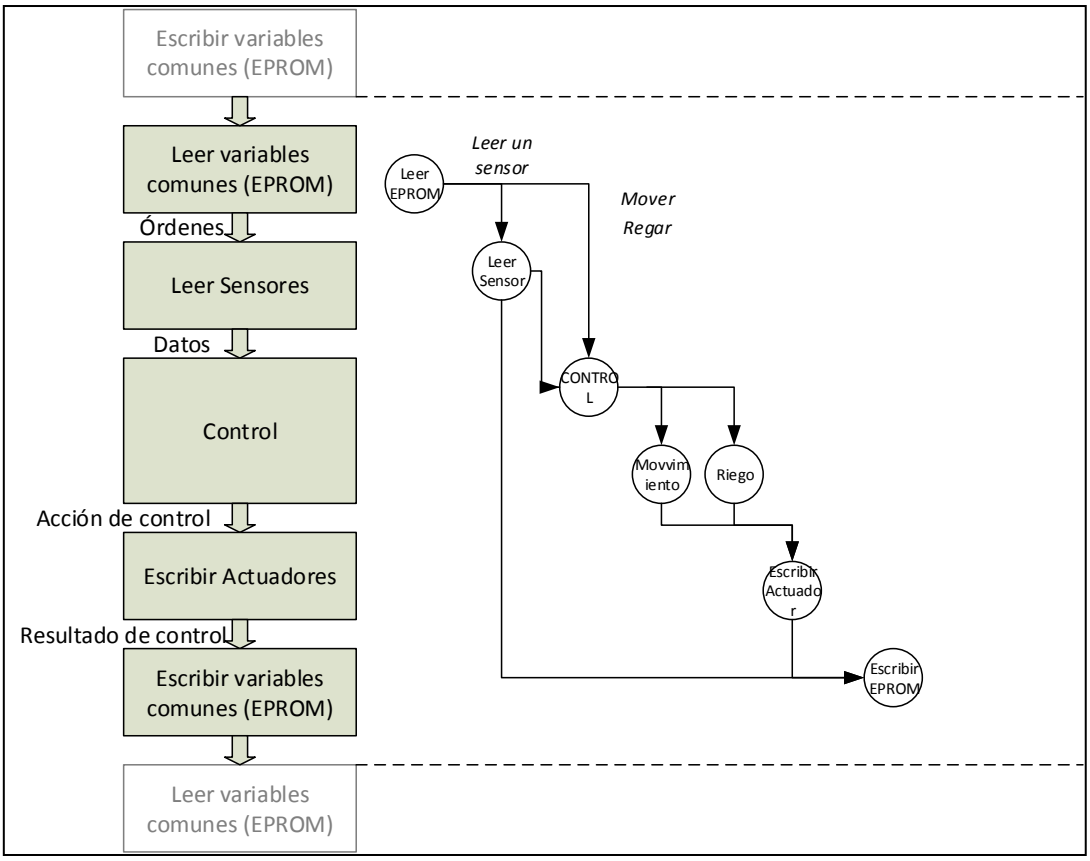

Ilustración 16. Funcionamiento del sistema[16].

En esta ilustración se muestran las actividades que debe llevar a cabo el controlador dentro de la fase de bucle. Como ya hemos comentado anteriormente, el flujo de acciones que se muestra en la figura se repetirá constantemente. A continuación se explica más detalladamente el funcionamiento.

El primer paso es leer las variables de comunicación en la EPROM. Estas variables contienen la información que se envía desde las otras capas del sistema (por ejemplo, órdenes del usuario). A continuación se leen los sensores y se anotan sus valores en las posiciones correspondientes en la EPROM.

Una vez se dispone de todos los datos recopilados, se realiza el control. En este punto se decide si las acciones solicitadas a través de las variables de comunicación pueden realizarse. También se determinan aspectos cómo el riego o el movimiento en busca de luz en función del valor de los sensores.

El resultado del paso anterior es escribir los actuadores, es decir, una vez se determina que acciones se deben tomar, se accionan los actuadores necesarios para realizar dichas acciones.

Para finalizar el ciclo, se escribe de nuevo en la EPROM las variables de comunicación, para poder enviar la información sobre las acciones que se han tomado y el estado del sistema.

[16] Fuente: elaboración propia.

Cabe destacar que tanto al inicio cómo al final del ciclo descrito se realizan las comunicaciones. La realización de dichas comunicaciones, como ya se ha comentado anteriormente, pertenece a otra parte del proyecto.

4.3 Hardware

El hardware es una parte imprescindible de este sistema. Para poder cubrir las funcionalidades descritas y obtener la información necesaria se necesitan una serie de sensores, actuadores y controladores que se describen a continuación.

4.3.1 Sensores

Los parámetros a monitorizar por este sistema son cuatro: humedad, temperatura humedad del suelo y luz. Adicionalmente también necesitamos sensores para detectar obstáculos y para poder controlar el nivel del depósito de agua.

4.3.1.1 Temperatura

Los sensores de temperatura se encargaran de monitorizar tanto la temperatura del aire como la temperatura del suelo. La precisión en la medida de la temperatura no es un elemento crítico en este sistema, por lo que pueden tolerarse márgenes de error de +1/-1 ºC. Pueden usarse sensores digitales o analógicos para satisfacer las necesidades de detección de temperatura.

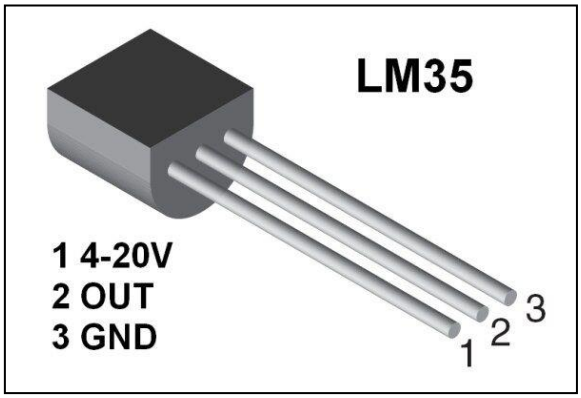

Ilustración 17.Sensor de Temperatura LM35[17].

Existe una gama variada de este tipo de sensores en el mercado a precios muy económicos. Es habitual que los sensores de temperatura de midan también la humedad relativa del aire. Una opción muy adecuada es utilizar sensores que miden temperatura y humedad del aire a la vez, lo cual permite simplificar un poco las conexiones.

4.3.1.2 Humedad del aire

Los sensores de humedad de aire permitirán conocer la humedad relativa del aire. Existen diversos tipos de sensores para realizar dichas mediciones. Las mediciones de la humedad del aire no son un aspecto crítico del sistema. Por tanto, pueden tolerarse márgenes de error en la lectura de hasta el 5% sin afectar significativamente. La mayoría de los sensores disponibles en el mercado tienen una precisión mayor que la indicada, por lo que existe un gran número de opciones adecuadas al proyecto.

---

[17] Fuente: http://cdn.instructables.com/FE0/DHQ4/HV2AIB01/FE0DHQ4HV2AIB01.MEDIUM.jpg

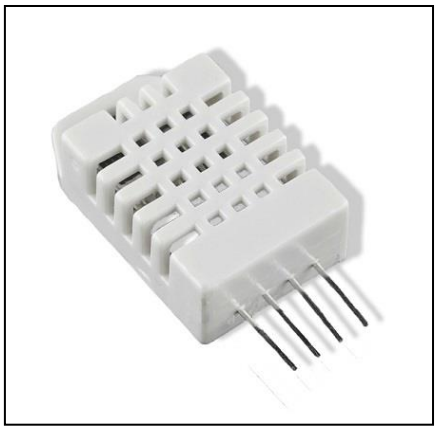

Ilustración 18. Sensor de humedad[18]

Cómo se ha comentado en el apartado anterior, es habitual encontrar este tipo de sensores combinados con sensores de temperatura. Ésta resulta ser la elección que mejor se adapta al proyecto, ya que ofrece la lectura de dos parámetros utilizando una sola conexión.

4.3.1.3 Humedad del suelo

Los sensores de humedad se usarán para medir la cantidad de agua que hay en el sustrato donde se alberga el cultivo. Este tipo de sensores son de tipo resistivo y tienen una parte que se introduce directamente en el sustrato.

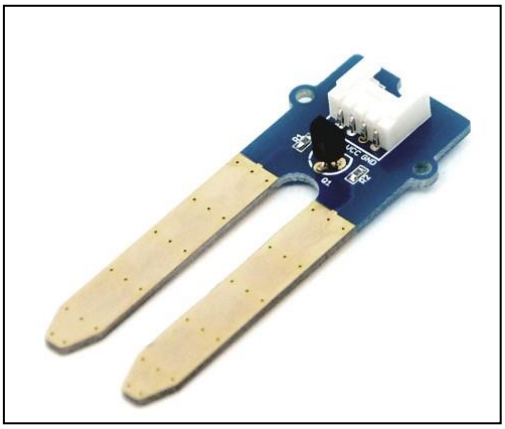

Ilustración 19.Sensor de humedad de suelo[19].

Estos sensores suelen sufrir desgaste debido a la propia humedad, que produce que se oxiden sus patas. Existen distintas gamas y calidades en el mercado. Los sensores más económicos cumplen con suficiencia los requisitos del proyecto.

4.3.1.4 Luz

Existen diversos modos de detectar la cantidad de luz recibida. Podemos hacer uso de sensores resistivos LDR. Este tipo de sensores modifican el valor de su resistencia en función de la cantidad de luz recibida.

---

[18] Fuente: http://www.electronicaestudio.com/i/f/SHT-169.jpg

[19] Fuente: http://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=888304048&cPath=1343&gclid=CLWw8b7MmsoCFUQcGwod9YsIVQ

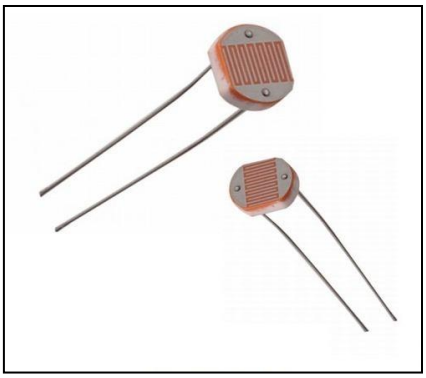

Ilustración 20.Sensores LDR[20].

El tiempo de respuesta de estos sensores es reducido y su precio muy económico, por lo que son una alternativa adecuada al proyecto.

4.3.1.5 Detección de obstáculos

Para detectar la presencia de obstáculos existen una gran variedad de sensores. Los más habituales son los que se basan en el envío y recepción de pulsos de ultrasonidos o infrarrojos. Sin embargo, dado que la velocidad de movimiento será reducida, podemos utilizar cualquier sensor de gama media o baja.

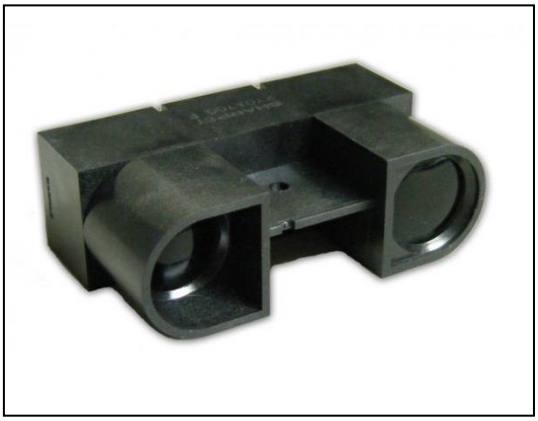

Ilustración 21.Sensor de distancias por Infrarrojos.[21]

Una opción bastante adecuada es el uso de sensores de ultrasonidos. El funcionamiento de estos sensores se basa en el envío de un pulso ultrasónico y su posterior recepción; calculando el tiempo que ha tardado en rebotar y obteniendo información sobre la distancia al obstáculo más próximo.

[20] Fuente: http://electronicastore.net/wp-content/uploads/2015/12/Photoresistor-LDR.jpg
[21] Fuente: https://www.openhacks.com/uploadsproductos/1-600x600.jpg

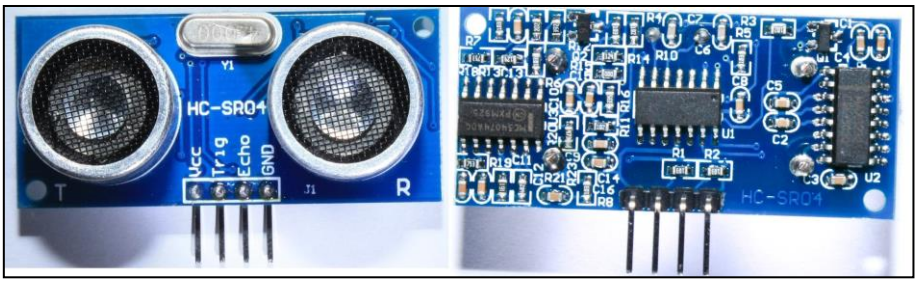

Ilustración 22.Sensores de Ultrasonidos[22].

4.3.1.6 Nivel depósito.

Para poder hacer un uso adecuado de la funcionalidad de riego es necesario disponer de un depósito de agua. Es importante controlar el nivel del depósito para evitar producir daños en el sistema de bombeo de agua

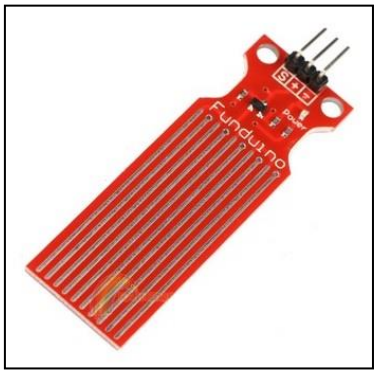

Ilustración 23.Sensor de nivel[23].

Para detectar el nivel del líquido existen diversas alternativas. Pueden usarse sensores de ultrasonidos, sensores de flotación o incluso sensores ópticos. También hay sensores que se introducen en el líquido y permiten detectar el nivel exacto. Sin embargo, dado que en este entorno no es crítico conocer el nivel del líquido, pueden utilizarse sensores de flotación interruptores.

Ilustración 24. Sensor de flotación interruptor.[24]

Este tipo de sensores se montan dentro del depósito. Incluyen una boya que en función del nivel del líquido flota más o menos. Cuando la boya llega a un cierto nivel, cierra un interruptor, por lo que solo se detecta si el interruptor está cerrado o abierto. Este tipo

[22] Fuente: https://i.imgur.com/kdWiw.jpg
[23] Fuente: http://www.prometec.net/wp-content/uploads/2014/10/WaterSensor.jpg
[24] Fuente: https://images-na.ssl-images-amazon.com/images/I/41YZJuidvJL.jpg

de sensores no permite conocer el nivel exacto del líquido, sólo permite saber si el nivel ha llegado a ciertos umbrales, que dependerán de la colocación física del sensor. Por ello utilizaremos un sensor para detectar la capacidad máxima del depósito (En la parte superior del depósito) y otro para detectar el nivel mínimo necesario (En la parte inferior del depósito). Una ventaja que ofrece el uso de estos sensores es que no dependen de las dimensiones del depósito.

4.3.2 Actuadores

Para poder cumplir con las funcionalidades de riego y movimiento son necesarios dos tipos de actuadores, motores y bombas de agua.

4.3.2.1 Motores

Existe una gran variedad de motores. Para aplicaciones de movimiento pueden emplearse motores de corriente continua normales o servomotores de rotación continua. Los dos tipos de motores tienen unos requisitos similares, aunque difieren en su funcionamiento.

Ilustración 25.motores de corriente continua.[25]

En caso de utilizar motores de corriente continua será necesario emplear circuitos adicionales para lograr el control. La potencia de estos motores dependerá del voltaje suministrado. Si se hace uso de servomotores de rotación continua, no es necesario conectar hardware adicional.

4.3.2.2 Bombas de agua

La bomba de agua permite mover agua desde un depósito a otra ubicación a través de un conducto. La capacidad de la bomba y la presión del líquido varían en función del modelo.

[25] Fuente: http://www.electrokit.com/public/upload/productimage/45955-5210-4.jpg

Ilustración 26.Bomba de agua.[26]

Estas bombas se introducen dentro del depósito, sumergidas en el líquido. Es importante controlar el nivel del depósito porque las bombas tienen un nivel de funcionamiento. Si se activa la bomba cuando el nivel del líquido es inferior al nivel recomendado por el fabricante, ésta podría romperse.

La distancia entre el depósito y el sustrato será muy reducida, por lo tanto, una bomba sencilla con capacidad de bombear agua a poca distancia es suficiente.

4.3.3 Microcontrolador

Para poder gestionar todos los elementos descritos es necesario contar con un microcontrolador. El microcontrolador elegido debe tener capacidad para conectarse con los sensores y actuadores. Esto implica que necesariamente deberá disponer de entradas y salidas tanto analógicas como digitales.

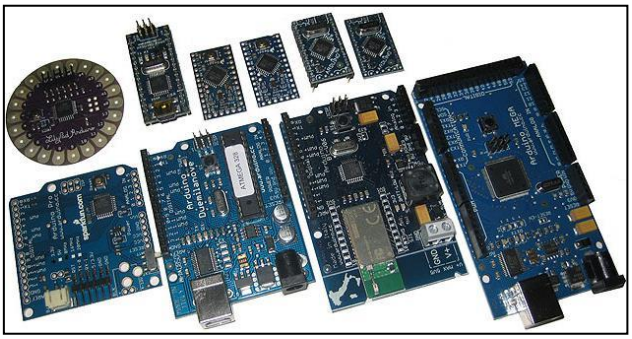

Ilustración 27.Controladores Arduino.[27]

También debe tener capacidad para cargar y ejecutar código. Otra característica necesaria es la existencia de una memoria persistente (EEPROM).

Cómo ya se ha especificado anteriormente todo el código del proyecto se va a realizar en Arduino, por lo que es necesario utilizar un microcontrolador perteneciente a esta plataforma.

[26] Fuente: http://www.tiendafotovoltaica.es/WebRoot/StoreES/Shops/61359426/MediaGallery/TP18.jpg
[27] Fuente: https://carlossolon.files.wordpress.com/2011/03/imagem1.jpg

4.3.4 Elementos estructurales.

Para poder montar el conjunto se necesitan algunos elementos que no tienen cabida en las categorías anteriores.

El primer elemento es el depósito, necesario para almacenar el agua destinada al riego. Otro elemento necesario son las ruedas, indispensables para el movimiento. Finalmente necesitamos una estructura donde montar todo el hardware descrito, además de la maceta que albergará el sustrato y el cultivo.

4.4 Control depósito

Uno de los elementos más importantes del sistema es el depósito. El depósito permitirá regar la planta cuando las condiciones lo requieran. Sin embargo, requiere realizar un control para asegurar que los niveles están dentro de unos umbrales apropiados. Lo más crítico es controlar que el nivel inferior sea superior a un mínimo para evitar daños en la bomba de agua que se encarga del riego.

4.4.1 Esquema

Tomando en cuenta estas características, para realizar el control del depósito se utilizarán dos sensores de nivel, uno de nivel superior y otro de nivel inferior. En la Ilustración 28 se muestra el esquema de los sensores y actuadores correspondientes al depósito.

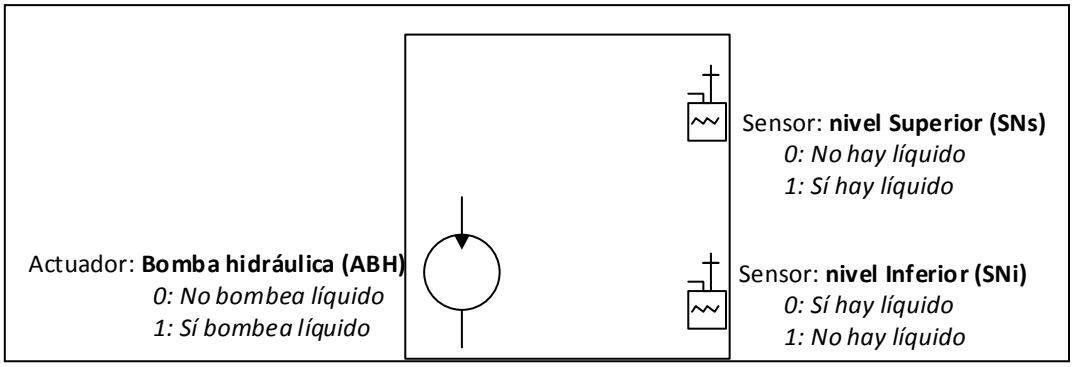

Ilustración 28. Depósito, componentes y nomenclatura.

4.4.2 Tabla de verdad

El funcionamiento consiste en no permitir que el depósito sobrepase el nivel superior (para evitar que desborde) y no permitir que la bomba hidráulica actúe en vacío (sin agua). Esto se puede plasmar en la siguiente tabla de verdad.

SNs	*SNi*	*ABH*	*Explicación*
0	0	1	Hay líquido entre el Sin y el SNs, por lo que se permite accionar la bomba hidráulica.
0	1	0	No hay líquido (por debajo del mínimo), por lo que no se permite accionar la bomba hidráulica.
1	0	1	Máximo líquido admisible, se permite accionar la bomba.
1	1	Error	

Tabla 19. Tabla de verdad depósito.

4.4.3 Máquina de estados

4.4.3.1 Diagrama

El control del depósito se puede modelar como una máquina de estados UML. A continuación se muestra el diseño:

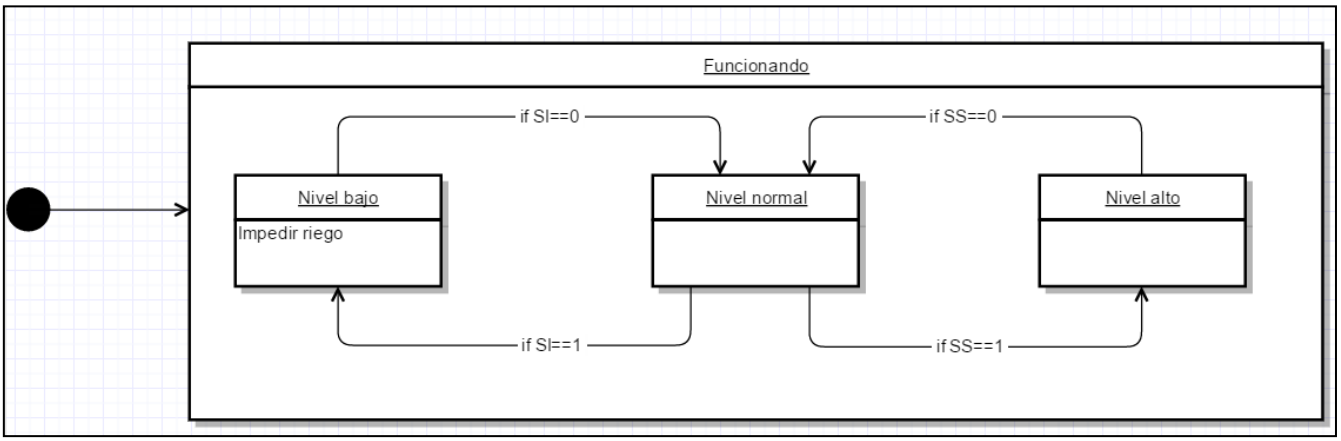

Ilustración 29.Máquina de estados UML depósito.[28]

4.4.3.2 Estados

La siguiente tabla muestra la codificación de los eventos, así como una descripción de los mismos:

Nombre estado	Código	Descripción
EST_INICIAL	0	Estado inicial. Este estado solo sirve para inicializar el control del depósito.
EST_BAJO	1	En este estado el nivel del depósito está por debajo del mínimo. No se debe permitir activar el riego en este estado
EST_ALTO	2	El nivel del depósito está por encima del nivel máximo. No se debe añadir más agua al depósito. Se permite el riego
EST_NORMAL	3	El nivel del depósito se encuentra por debajo del máximo y por encima del mínimo. Se permite el riego.

Tabla 20.Estados del control del depósito.

4.4.3.3 Eventos depósito

Para determinar los eventos que producen los cambios de estado se tomará como entrada el valor de los sensores de nivel. A partir de estos valores se emite un evento que modificará el estado según las transiciones definidas en el diagrama. Los eventos se codifican según la siguiente tabla:

Nombre evento	Código	Descripción
EV_NINGUNO	0	Este evento se emite cuando el nivel del depósito es inferior al límites superior y mayor que el límite inferior.
EV_LIMITE_SUPERIOR	1	Este evento se emite cuando se supera el límite superior del depósito
EV_LIMITE_INFERIOR	2	Este evento se emite cuando el nivel está por debajo del límite inferior del depósito.

Tabla 21. Eventos del depósito de agua

[28] Fuente: Elaboración propia.

4.5 Control de movimiento

El control de movimiento permitirá gestionar las funcionalidades de reubicación del sistema. Es importante llevar a cabo en este apartado un control de posibles obstáculos, que se detectarán mediante el sensor de ultrasonidos incorporado.

Para modelar la funcionalidad del movimiento se ha diseñado la siguiente máquina de estados UML:

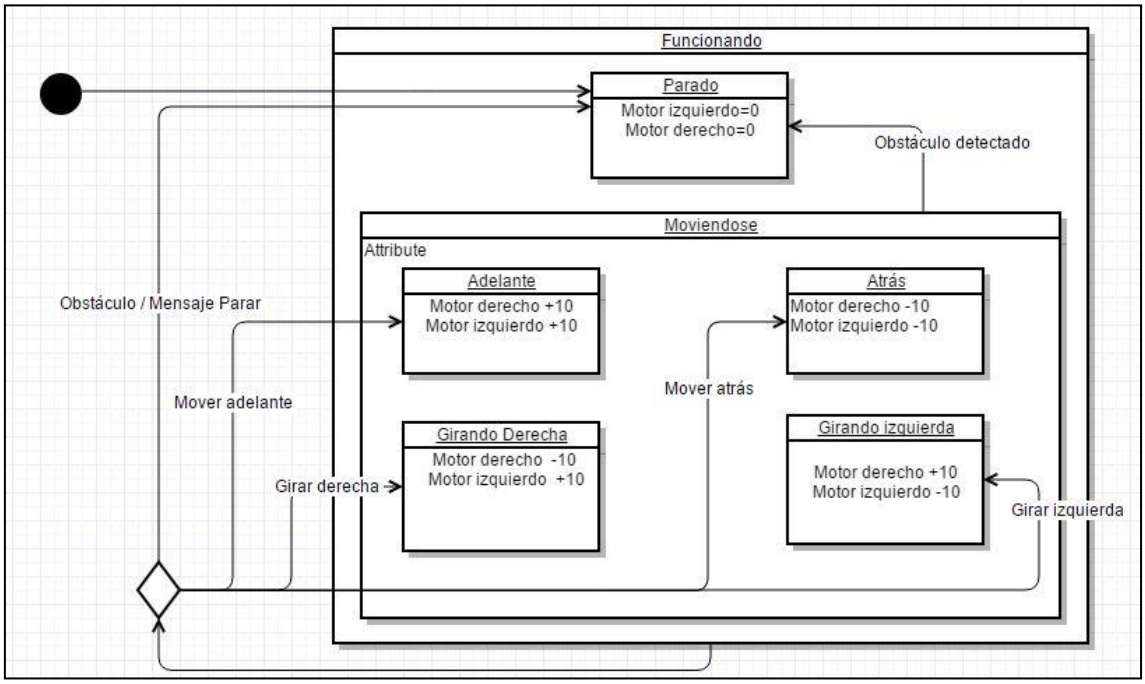

Ilustración 30.Máquina de estados UML movimiento.[29]

El funcionamiento se divide en dos bloques, parado o moviéndose. Dentro del bloque moviéndose encontramos cuatro posibilidades: moverse adelante o atrás y girar a izquierda o derecha. La siguiente tabla muestra la codificación y la descripción de los estados:

Nombre del estado	Código	Descripción
EST_INICIAL	0	Estado inicial. Solo sirve para inicializar el control.
EST_PARADO	1	En este estado los motores están apagados.
EST_MOVIENDOSE_ADELANTE	2	Los dos motores están activados y girando hacia adelante.
EST_MOVIENDOSE_ATRAS	3	Los dos motores están activados y girando hacia detrás
EST_MOVIENDOSE_IZQUIERDA	4	Los dos motores están activados, el de la derecha gira hacia adelante y el de la izquierda hacia atrás.
EST_MOVIENDOSE_DERECHA 6	5	Los dos motores están activados, el de la izquierda gira hacia adelante y el de la derecha hacia atrás.

Tabla 22.Estados del control de movimiento.

[29] Fuente: Elaboración propia

Para decidir los eventos que afectan a esta máquina de estados se tomarán en cuenta diversos factores. Primero, se comprobará si se han detectado obstáculos. Después también se comprobará si se han recibido órdenes para moverse en una determinada dirección. Finalmente, si se encuentra activa la búsqueda de luz se tendrán en cuenta los valores de las LDR para determinar la dirección de movimiento. Según la información disponible se emitirá el evento correspondiente. La siguiente tabla muestra la información sobre los eventos que se pueden emitir:

Nombre del estado	Código	Descripción
EV_NINGUNO	0	Este evento se emite cuando no hay ninguna orden de movimiento.
EV_PARAR	1	Cuando se emite este evento los motores deben apagarse.
EV_ADELANTE	2	Los dos motores deben activarse girando hacia adelante.
EV_ATRAS	3	Los dos motores deben activarse girando hacia atrás
EV_IZQUIERDA	4	Debe activarse el giro hacia la izquierda
EV_DERECHA	5	Debe activarse el giro hacia la derecha
EV_OBSTACULO	6	Se ha detectado un obstáculo. Los motores deben apagarse.

Tabla 23. Eventos de movimiento.

4.6 Interfaz de datos

Para poder comunicar los datos con los que trabaja el sistema se hace uso del paso de variables. Este método consiste en escribir la información en variables que puedan ser leídas por otras partes del código. Para escribir estas variables utilizaremos la memoria EEPROM.

4.7 Conclusiones

En este apartado se ha concretado todo el diseño del sistema. Se ha especificado todo el hardware necesario para proceder a la implementación. También se ha abordado el diseño de los casos de uso para poder organizar el código a desarrollar. Ahora que ya están claros los elementos necesarios para el montaje del proyecto y el diseño de su funcionamiento, el siguiente paso es realizar la implementación.

En el siguiente capítulo se procederá al montaje del sistema. También se desarrollará el código que se cargará finalmente en el controlador. Por último, se realizarán una serie de pruebas para evaluar el resultado obtenido.

5 Implementación, implantación y evaluación

5.1 Introducción

Tras haber realizado el diseño del sistema, sólo queda proceder a su montaje y desarrollar el código necesario. En este capítulo se van a especificar los elementos hardware finales utilizados en el proyecto. Después se mostrarán las partes más destacadas del código. Finalmente, se realizarán pruebas sobre el sistema desarrollado y se evaluará el resultado obtenido.

5.2 Implementación

5.2.1 Hardware utilizado

5.2.1.1 Arduino Mega

Arduino Mega es un microcontrolador perteneciente a la familia de placas de Arduino. Dispone de 54 entradas/salidas digitales (14 de las cuales pueden utilizar PWM) y 16 entradas analógicas. También dispone de conectividad USB y entrada para alimentación eléctrica de la placa (Batería o fuente de alimentación). También dispone de una memoria EEPROM con una capacidad de 4KB.

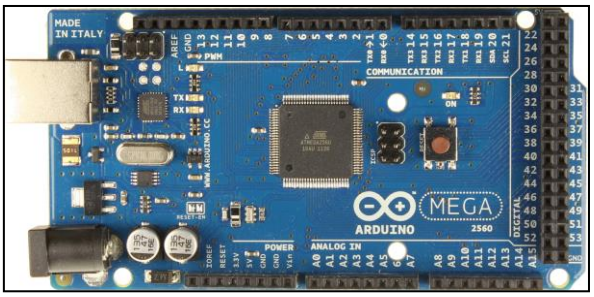

Ilustración 31.Vista superior de la placa Arduino Mega.[30]

Este será el cerebro del sistema, pues aquí se cargará y se ejecutará todo el código desarrollado. La conexión con los sensores y actuadores se hará de manera directa con las entradas y salidas de la placa. También pasan por aquí todas las comunicaciones desde el módulo al servidor.

5.2.1.2 Sensor LDR

Cómo ya se ha comentado anteriormente, para poder detectar la cantidad de luz solar, se hará uso de dos sensores LDR. Estos sensores permiten detectar la cantidad de luz recibida mediante la variación de la resistencia interna. Por ello, será necesario conectarlos a entradas analógicas de Arduino. A continuación podemos ver una foto del circuito para poder leer de este sensor:

[30] Fuente: https://paruro.pe/sites/default/files/ArduinoMega2560_R3_Front.jpg

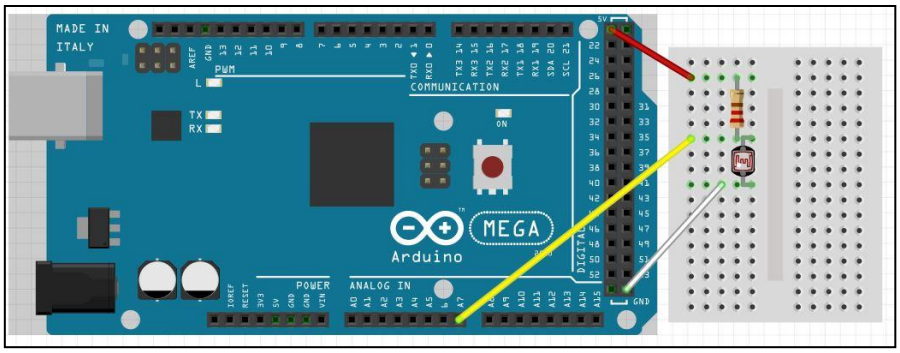

Ilustración 32.Conexión sensor LDR[31]

Estos sensores proporcionan un tiempo de respuesta muy reducido y tienen un precio muy económico por lo que son ideales para el proyecto.

Cuando se realiza la lectura de estos sensores, en realidad se está leyendo el voltaje que hay en el pin correspondiente. Por tanto dependiendo de la conexión el voltaje variará inversamente con la cantidad de luz o al contrario. La conexión utilizada para detectar el nivel de luz se conoce como divisor de tensión.

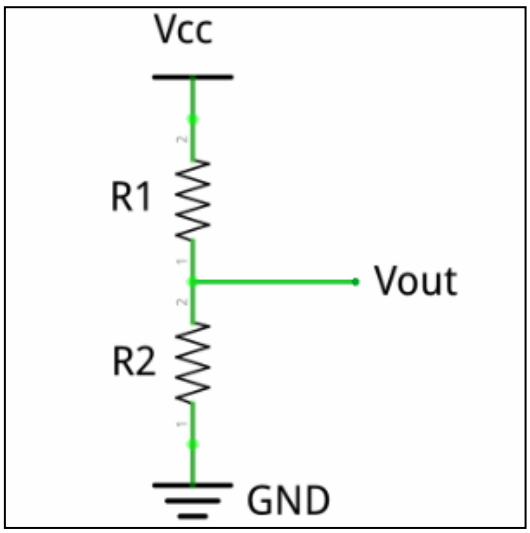

Ilustración 33. Divisor de tensión[32].

Este tipo de circuito eléctrico permite regular el voltaje que se recibe en el punto Vout. La lectura analógica de una entrada en Arduino funciona precisamente leyendo el voltaje de una entrada, por lo que el valor que percibe el sensor dependerá directamente del voltaje.

En el esquema realzado la LDR se situará en la posición de R2 en Ilustración 33. El voltaje variará de modo inverso a la cantidad de luz. Esto se debe a que la resistencia de la LDR se reduce cuando incide la luz sobre ella. Esto implica una mayor caída de voltaje en la primera resistencia del circuito y por consiguiente la lectura de un valor más pequeño. En cambio, cuando no incide luz sobre la LDR su resistencia aumenta, lo

[31] Fuente: Elaboración propia.
[32] Fuente: http://www.educachip.com/utilizar-ldr-arduino/

que provoca que la caída de tensión de la primera resistencia sea menor, y se lea un valor mayor.

Los valores que pueden leer oscilan entre 0 y 1023. Esto se debe a que Arduino dispone de un conversor de ADC (analógico a digital) de 10 bits. Por ello el mayor valor que se puede leer es 1023.

La precisión de las lecturas analógicas depende directamente del voltaje de referencia. Si no se suministra un voltaje de referencia en el pin disponible para tal efecto, se toma como valor de referencia la tensión de alimentación. Arduino utiliza el voltaje de referencia para asociarlo al valor 1023, que es el máximo valor para 10 bits. En nuestro caso, se alimenta Arduino con 5V, por lo que leer el valor 1023 en una entrada significa que están llegando 5V a esa entrada. Reducir el voltaje de referencia puede aumentar la precisión de los sensores, pero con 5V se consigue la precisión suficiente para el correcto funcionamiento del sistema.

En base al rango de valores que leemos en los sensores LDR puede realizarse la conversión a lúmenes. Sin embargo las resistencias LDR tienen una capacidad limitada y no son adecuadas cómo medidor de luz, debido entre otros factores, a que se ven muy afectadas por los cambios de temperatura y son sensibles solo a determinadas longitudes de onda. En cambio, proporcionan suficiente información como para saber dónde hay mayor cantidad de luz.

5.2.1.3 Sensor de humedad de suelo DFRobot

Para medir la cantidad de agua en el suelo se empleará un sensor de humedad de suelo *DFRobot*. Este sensor se introduce en el suelo y mide la humedad dependiendo de la resistencia que detecta entre sus dos pines. La siguiente imagen muestra cómo se realiza la conexión.

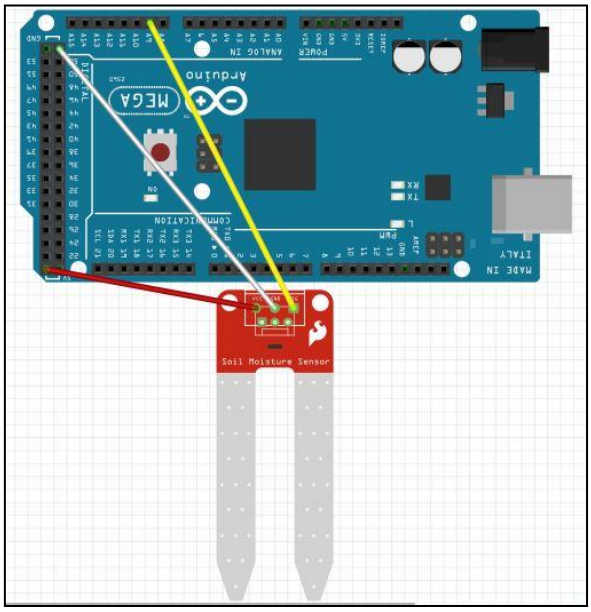

Ilustración 34.Montaje Sensor de humedad suelo[33]

[33] Fuente: elaboración propia.

Cabe destacar que la salida del sensor proporciona el valor de una resistencia, y por ello debe conectarse a una entrada analógica para su correcta lectura. A modo orientativo, el fabricante proporciona información sobre la cantidad de agua asociada a las lecturas del sensor. Esta información puede consultarse en la siguiente tabla:

Valor	Descripción
0-300	Suelo seco.
300-700	Suelo húmedo.
700-900	Inmerso en agua.

Tabla 24.Valores del sensor de humedad[34].

5.2.1.4 Sensor de temperatura y humedad DHT11

Este sensor nos permite detectar la temperatura del aire y a la vez la humedad relativa. Se trata de un sensor económico y con una precisión adecuada al entorno en el que se va a utilizar. La conexión del sensor se muestra en la siguiente imagen.

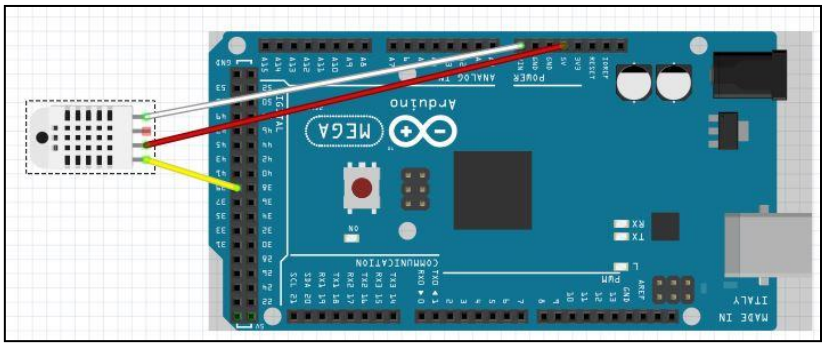

Ilustración 35.Montaje Sensor DHT11[35].

Para poder leer los valores de este sensor haremos uso de la librería que proporciona el fabricante. Esta librería se sincroniza con el sensor. Una vez sincronizado, el sensor envía de manera digital la información sobre la humedad y la temperatura del aire, por lo que sólo es necesario un pin digital para leer estas dos magnitudes.

Los valores que nos proporciona este sensor son los siguientes:

Magnitud	Rango
Temperatura	0ºC - 50ºC
Humedad	20% - 90%

Tabla 25.Valores proporcionados por el sensor DHT 11[36]

5.2.1.5 Sensores de nivel de depósito

Para realizar las mediciones del depósito utilizaremos dos sensores de flotación de nivel. Estos sensores actúan como interruptores y se accionan por flotación. Cuando el

[34] Fuente: http://www.dfrobot.com/index.php?route=product/product&product_id=599#.V2RAObsgXIU

[35] Fuente: Elaboración propia.

[36] Fuente: http://www.miniinthebox.com/es/arduino-compatible-temperatura-dht11-digital-del-modulo-del-sensor-de-humedad_p903332.html?currency=EUR&litb_from=paid_adwords_shopping&utm_source=google_shopping&utm_medium=cpc&adword_mt=&adword_ct=100812690328&adword_kw=&adword_pos=101&adword_pl=&adword_net=g&adword_tar=&adw_src_id=7107338275_427144048_30699529408_pla-89156505194&gclid=CN2Y7OTXr80CFaoV0wod5wMHxQ

nivel del agua sube, el sensor flota y cierra un interruptor integrado. La conexión para detectar el estado de dos interruptores se muestra en la siguiente imagen:

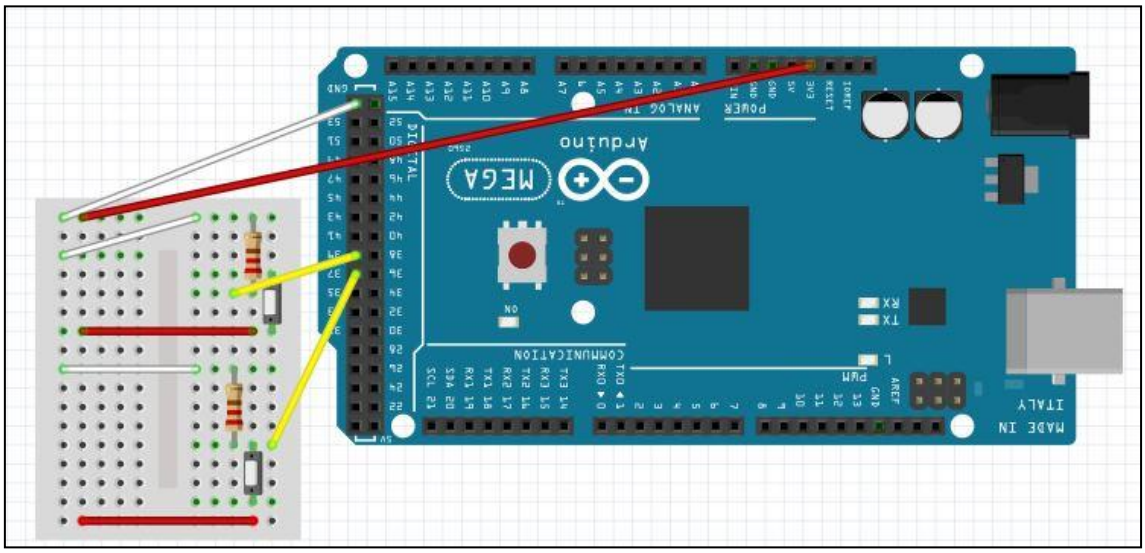

Ilustración 36. Esquema interruptores depósito[37].

Los valores que proporcionan estos sensores obviamente sólo tienen dos posibilidades. La lectura de estos sensores se hace mediante un pin digital, de modo que sólo se pueden leer valores *HIGH* o *LOW*, que se corresponden a interruptor cerrado o abierto respectivamente.

Estado interruptor	Valor
Abierto	0
Cerrado	1

Tabla 26.Valores posibles de los sensores de nivel.

5.2.1.6 Bomba de agua

Para poder implementar la funcionalidad de riego se ha elegido una bomba de agua de 12V. Esta bomba tiene potencia suficiente para poder mover el líquido del depósito al sustrato. Para poder accionar la bomba es necesario un relé, dado que funciona con 12V y Arduino no es capaz de sacar ese voltaje por sus pines. El relé funciona como un interruptor controlado por corriente, de modo que puede ser accionado por cualquier pin digital de Arduino. La conexión que se ha realizado es la siguiente:

[37] Fuente: elaboración propia.

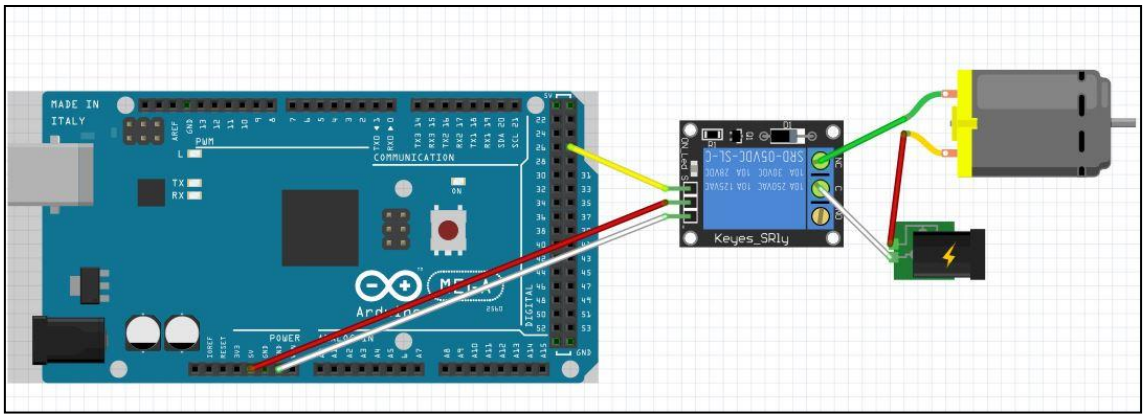

Ilustración 37.Conexión bomba de agua[38]

La bomba de agua se comporta exactamente igual que un motor de corriente continua, girando en un sentido u otro dependiendo de la polaridad con que se alimenta. También es importante remarcar que el relé se puede conectar en dos modos: normalmente abierto o normalmente cerrado. El objetivo es accionar la bomba cuando sea necesario y tenerla apagada el resto del tiempo, por lo que se ha optado por la conexión en modo normalmente cerrado.

5.2.1.7 Motores Pololu

Los motores elegidos para el proyecto son los motores *Pololu* 37Dx54L. Se trata de motores de corriente continua que pueden alimentarse a doce voltios. Se han elegido estos motores porque es necesaria suficiente potencia para mover el conjunto de la maceta, los sensores, la tierra, el agua y el microcontrolador.

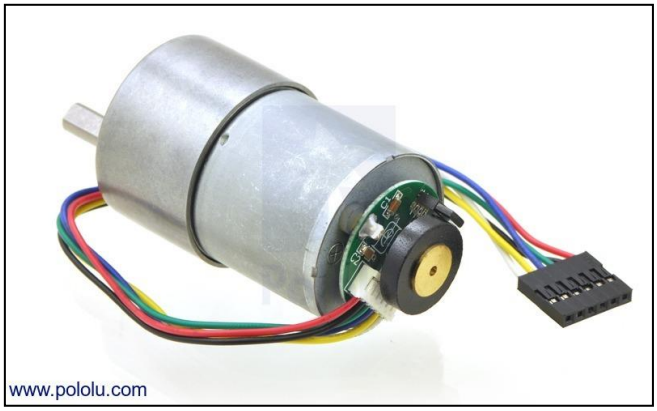

Ilustración 38.Motor *Pololu*[39].

Estos motores incorporan además un *encoder* que permite conocer la velocidad de rotación, aunque no vamos a utilizar esta característica en el primer prototipo.

Para poder controlar los motores se hará uso de un módulo L298N. Este módulo implementa un puente en H, lo cual permite controlar el sentido de giro de hasta dos motores de corriente. El siguiente esquema muestra las conexiones necesarias para poder utilizar los motores con el puente en H.

[38] Fuente: Elaboración propia.
[39] Fuente: https://www.pololu.com/picture/view/0J4045

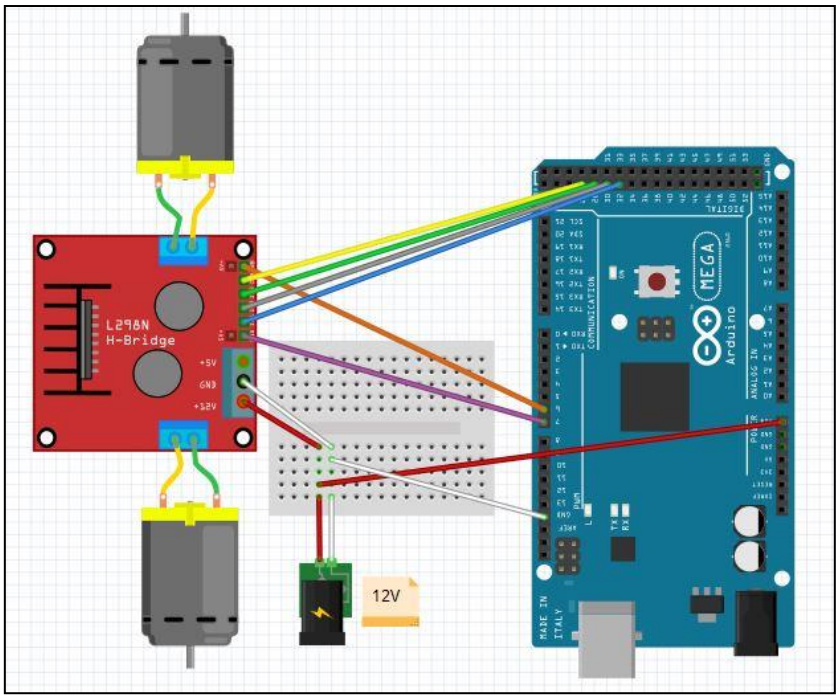

Ilustración 39.Conexión motores[40].

Es importante tener en cuenta que las entradas de habilitación del circuito L298N (ENA y ENB) tienen que estar conectadas necesariamente con salidas PWM de Arduino. Mediante estas entradas, se cambiará la potencia asignada a cada motor. El resto de entradas se conectan a pines digitales. Además, se ha utilizado la batería para alimentar tanto motores como la placa Arduino.

5.2.1.8 Sensores de ultrasonidos HC-SR04

Utilizaremos el sensor HC-SR04 para detectar posibles obstáculos. Este sensor envía un pulso de ultrasonidos, y cuando este pulso rebota en algún objeto, calcula el tiempo que ha tardado en volver a llegar el rebote. A partir de este tiempo y conociendo la velocidad del sonido en el aire se puede calcular la distancia del obstáculo detectado. Se ha elegido este sensor por su reducido coste y porque satisface las necesidades de detección de objetos. La siguiente imagen muestra la conexión con este sensor:

[40] Fuente: elaboración propia.

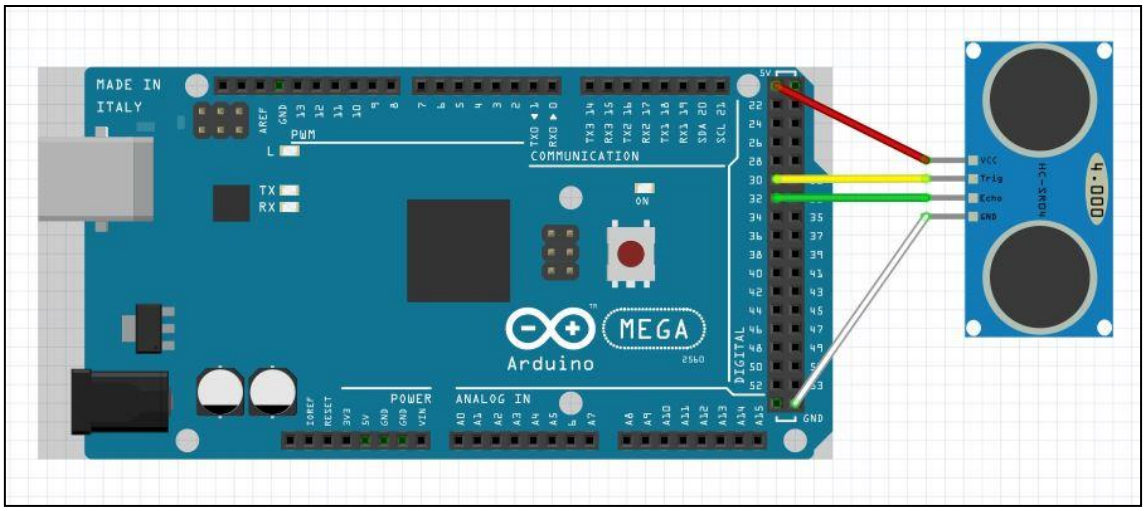

Ilustración 40.Montaje Sensor HCSR04[41].

Este sensor proporciona un valor digital, que es el tiempo que tarda en rebotar el pulso ultrasónico contra un objeto. A partir de este valor es necesario aplicar un proceso para calcular la distancia. Conociendo la velocidad del sonido, es suficiente con aplicar la siguiente fórmula:

$$Distancia = \frac{tiempo \ * 340 \ m/s}{2}$$

El rango de funcionamiento de este tipo de sensores depende de la proximidad de un obstáculo, y es el siguiente:

Tipo	Valor
Distancia mínima	2 cm
Distancia máxima	4m

Tabla 27.Rango funcionamiento sensor HCSR04[42]

5.2.2 Capa de persistencia /interfaz

Para poder guardar el estado del sistema de manera no volátil se hace uso de la memoria EEPROM disponible en la placa Arduino Mega. Además, esta memoria también servirá para implementar las comunicaciones con los demás programadores.

La capacidad de esta memoria varía según el modelo de la placa que se está utilizando. Cabe destacar que esta memoria tiene un número de escrituras limitados sobre cada posición; por lo que es importante implementar alguna política para poder reducir el número de escrituras.

5.2.2.1 Organización y estructura

Para poder trabajar con la EEPROM de Arduino primero es necesario que entender su funcionamiento. La memoria EEPROM de Arduino proporciona persistencia de datos y puede ser utilizada mediante la librería "*EEPROM.h* ". Dicha librería permite realizar operaciones de lectura y escritura en la memoria. Se trabaja con las direcciones byte a byte. La cantidad de memoria EEPROM depende de la placa Arduino que se esté usando, en nuestro caso, se dispone de hasta 4KB de almacenamiento.

[41] Fuente: elaboración propia.
[42] Fuente: http://descargas.cetronic.es/HCSR04.pdf

Placa Arduino	Memoria EEPROM (KB)
UNO	1
DUE	-
Leonardo	1
Mega	4
Micro	1
Mini	1
Nano	1
Ethernet	1
Esplora	1
Bluetooth	1
Fio	1
Pro Mini	1
Lilypad	1

Ilustración 41. Capacidad EEPROM según placa Arduino.[43]

Para organizar la distribución de la memoria, ésta se ha dividido en dos partes, una de uso interno del sistema y otra para recibir las variables de comunicación del exterior. La siguiente tabla muestra la organización de la memoria:

Variable	Desplazamiento (en bytes)	Tamaño	Tipo
temperatura	0	2B	Int
humedad	2	2B	Int
humedad_suelo	4	2B	Int
LDR1_val	6	2B	Int
LDR2_val	8	2B	Int
deteccion1	10	2B	Int
deteccion2	12	2B	Int
sensor_superior	14	1B	Boolean
sensor_inferior	15	1B	Boolean
estado_movimiento	16	1B	Byte
estado_deposito	17	1B	Byte
regando	18	1B	byte
(...)			
Última variable sistema	1999	1B	-
Variable comunicaciones 1	2000	xB	-
Variable comunicaciones 2	2000+xB	yB	-
(...)			
Última Variable comunicaciones	3999	1B	-

[43] Fuente: http://www.educachip.com/wp-content/uploads/Capacidad-EEPROM-Arduino.jpg

Las variables asociadas al control interno sólo deben ser escritas por el propio sistema. En cambio las variables de comunicación pueden ser leídas y escritas tanto por el sistema como por el encargado de gestionar las comunicaciones.

La finalidad de esta organización en memoria es disponer de una zona donde están disponibles los valore leídos por los sensores, y otra zona donde se alojan variables que implementan órdenes o alertas. Esto permite desacoplar el desarrollo del sistema con el desarrollo de las comunicaciones, del cual se encarga otro miembro del proyecto.

5.2.2.2 Implementación

Una vez se ha decidido cómo organizar la memoria, sólo queda desarrollar el código para escribir o leer la información según el caso.

La librería EEPROM.h ofrece varios métodos para leer o escribir un dato. Sin embargo, la memoria trabaja byte a byte y se están utilizando valores de más de un byte (int). Afortunadamente hay disponibles las funciones *put()* y *get()*. Estas funciones nos permiten escribir o leer un dato en una posición de memoria indicando el tipo del dato. Es importante destacar que la función *put()* solo actualiza la memoria si el valor que se quiere guardar es diferente al valor que hay guardado. De este modo se reduce el tiempo de ejecución y se ahorran algunas escrituras, lo cual aumenta la vida útil de la memoria.

Para hacer uso de las funcionalidades de memoria se han implementado dos métodos, que permiten encapsular el proceso de escritura.

```
176  //Método para escribir en EEPROM un dato de un solo byte (byte, boolean)
177  void escribir_byte_EEPROM(int direccion,byte dato ){
178    EEPROM.update(direccion,dato);}
179
180  //Método para escribir en EEPROM un dato de dos  bytes (int,long , float)
181  void escribir_int_EEPROM(int direccion,int dato){
182    EEPROM.put(direccion,dato);
```

Ilustración 42. Métodos para escritura EEPROM.[44]

Para poder racionalizar el uso de la memoria se va a limitar la frecuencia de actualización de los datos. Esto significa que mientras en cada iteración leemos los sensores y actuamos en consecuencia, pero sólo se actualizan los datos de la EEPROM cada cierto tiempo. Esto permitirá alargar su vida útil. La única desventaja será que los datos leídos por la capa de comunicaciones podrán estar desactualizados respecto al valor más actual. No obstante, se trata de un problema menor, dado que lo más crítico es tener los valores actualizados para el control.

Para modelar el comportamiento descrito, se ha implementado el método actualizar_EEPROM. Éste método se encarga de escribir los valores de los sensores en la memoria EEPROM cada cinco minutos o cuando el usuario lo solicite mediante el envío de una orden.

La implementación del método puede verse en la siguiente imagen:

[44] Fuente: Elaboración propia

```
223 void actualizar_EEPROM(){
224   //Solo escribimos en la EEPROM cada cierto periodo de tiempo o si lo solicita el usuario
225     if(millis()- contador_actualizacion > umbral_actualizacion || mensaje_actualizacion==1) {
226       escribir_int_EEPROM(POS_TEMP_AIRE,temperatura);
227       escribir_int_EEPROM(POS_HUM_AIRE,humedad);
228       escribir_int_EEPROM(POS_HUM_SUELO,humedad_suelo);
229       escribir_byte_EEPROM(POS_LDR1,LDR1_val);
230       escribir_byte_EEPROM(POS_LDR2,LDR2_val);
231       escribir_byte_EEPROM(POS_ULTRASONIDO1,deteccion1);
232       escribir_byte_EEPROM(POS_ULTRASONIDO2,deteccion2);
233       escribir_byte_EEPROM(POS_DEPOSITO_SUP,sensor_superior);
234       escribir_byte_EEPROM(POS_DEPOSITO_INF,sensor_inferior);
235       contador_actualizacion = millis();
236     }
```

Tabla 28. Implementación actualizar_EEPROM.[45]

Para realizar el control del tiempo transcurrido se hace hecho uso de la función "*millis()*". Esta función devuelve el número de milisegundos transcurridos desde el encendido de la placa. Realizando unos cálculos sencillos se puede determinar cuánto tiempo ha pasado.

5.2.3 Lectura de sensores

En este apartado se aborda el código desarrollado a partir de los diseños y diagramas anteriores. Se va a empezar por la lectura de sensores, dado que cada uno de ellos tiene requerimientos distintos, y en algunos casos ha sido necesario incluir un procesado de la información.

5.2.3.1 Método deteccion()

Este método implementa la lectura de los dos sensores de ultrasonidos disponibles. Estos sensores se sitúan en la parte frontal de la maceta. Como ya se ha comentado en apartados anteriores, estos sensores proporcionan como salida el tiempo que tarda en rebotar un pulso ultrasónico. Es necesario aplicar una serie de operaciones para transformar esta información en una distancia. Dependiendo de la distancia obtenida, se compara con un umbral y, en caso de ser menor, se considera que hay un obstáculo.

Uno de los problemas encontrados en la lectura de este tipo de sensores es que ocasionalmente se leen valores anómalos. Concretamente, cada cierto período de tiempo suelen aparecer lecturas que se corresponden a una distancia muy grande. Esto se interpreta por el sistema cómo ausencia de obstáculos y genera inestabilidad.

Para subsanar este problema se hace uso de una variable que guarda información sobre el último valor detectado. En caso de que llegue un valor demasiado diferente, se descarta la lectura. Este procesado ha resultado altamente eficaz, eliminando la inestabilidad producida por las lecturas anómalas. El código que implementa ésta funcionalidad es el siguiente:

[45] Fuente: Elaboración propia.

```
243 void detectar(){
244   //Sensor1
245   digitalWrite(pulso,LOW);                    /* Por cuestión de estabilización del sensor*/
246   delayMicroseconds(5);
247   digitalWrite(pulso, HIGH);                  /* Envío del pulso ultrasónico*/
248   tiempo=pulseIn(echo, HIGH);                 /* Función para medir la longitud del pulso entrante. Mide el tiempo que transcurrido entre el envío
249                                               del pulso ultrasónico y cuando el sensor recibe el rebote, es decir: desde que el pin 12 empieza a
250                                               recibir el rebote, HIGH, hasta que deja de hacerlo, LOW, la longitud del pulso entrante*/
251   distancia= int(0.017*tiempo);               /*fórmula para calcular la distancia obteniendo un valor entero*/
252   /*Procesado información*/
253   if(abs(anterior1-distancia)<200){           //Si leemos un valor anomalo descartamos la lectura
254   if(distancia<umbral_obstaculo){deteccion1=true;}//Si la distancia es menor que el umbral definido, detectado obstaculo
255   else{deteccion1=false;}
256   anterior1=distancia;}                       //Actualización última lectura para control de lecturas incorrectas
```

Ilustración 43.Función para detectar obstáculos.

Este proceso se aplica a los dos sensores, y cómo resultado se obtiene un valor en las variables *deteccion1* y *deteccion2* que indica si se ha detectado o no un obstáculo en el sensor correspondiente.

5.2.3.2 Métodos leer_luz() y leer_humedad_suelo()

Estos dos métodos implementas las lecturas de los valores de luz recibida y humedad del suelo respectivamente. Los sensores que proporcionan la lectura de dichas magnitudes son de tipo analógico, por lo que la implementación es similar.

```
300 /*Método para leer la humedad del suelo*/
301 void leer_humedad_suelo(){
302   humedad_suelo=analogRead(pin_humedad_suelo);
303 }//Fin leer_humedad_suelo
```

Ilustración 44. Método leer_humedad()[46]

Para leer el valor asociado a la lectura de cada sensor, se utilizará la función *analogRead()*. Esta función de Arduino lee el voltaje que entra por el pin especificado.

```
193 /*Método para leer la cantidad de luz recibida en las LDR*/
194 void leer_luz() {
195   LDR1_val=analogRead(LDR1_pin);
196   LDR2_val=analogRead(LDR2_pin);
197 }
```

Ilustración 45. Método leer_luz()[47]

Por los motivos explicados anteriormente en el apartado de sensores, no se realizará la conversión del valor de las LDR a lúmenes. En el caso del sensor de humedad tampoco se realiza ninguna conversión, ya que el fabricante proporciona una descripción para cada rango de valores leídos (Véase Tabla 24).

5.2.3.3 Método leer_temp_hum()

Este método implementa la lectura del sensor DHT11. Para ello, se hace uso de la librería "*DHT.h*", proporcionada por el fabricante. La implementación puede verse en la siguiente imagen:

[46] Fuente: Elaboración propia.
[47] Fuente: Elaboración propia.

```
222 /*Método para leer la humedad del suelo*/
223 void leer_humedad_suelo(){
224   humedad_suelo=analogRead(pin_humedad_suelo);
225   //Escribir en EEPROM
226   escribir_int_EEPROM(POS_HUM_SUELO,humedad_suelo);
227 }//Fin leer_humedad_suelo
```

Ilustración 46.Método leer_temp_hum()[48]

Es importante destacar que para que éste método funcione debe inicializarse la sincronización con el sensor, mediante el método *dht.begin(),* que debe llamarse en la fase de inicialización (método setup()). También cabe destacar que este sensor proporciona una precisión de hasta dos decimales. Sin embargo, para gestionar mejor la memoria y el tiempo de ejecución se convertirán estos valores a enteros. Esto se hace porque Arduino no trabaja demasiado bien con valores de tipo *float*. La pérdida de precisión provocada por ésta conversión no supone un gran problema en el contexto de nuestro sistema.

5.2.3.4 Método leer_depósito()

Para leer el valor de los sensores de depósito se utiliza el método digitalRead(). Este método lee el valor de tensión del pin correspondiente y devuelve un valor *HIGH* o *LOW,* que se corresponden respectivamente con 1 y 0. Es importante tener en cuenta que el voltaje de alimentación de los interruptores no debe superar los 5V, máximo valor de entrada en los pines digitales de Arduino, ya que podría dañarse la placa.

```
210 //Método para leer el estado de los sensores del depósito
211 void leer_deposito(){
212   sensor_superior_val = digitalRead(sensor_superior);
213   sensor_inferior_val = digitalRead(sensor_inferior);
214  }
```

Ilustración 47. Método leer_deposito()[49]

5.2.4 Generación de eventos

Para poder hacer uso de las máquinas de estados definidas en el capítulo de diseño es necesario implementar la generación de eventos. A continuación se explican las funciones implementadas para generar eventos.

La implementación de los eventos se ha realizado mediante dos variables globales que se encargan de almacenar el evento emitido. Posteriormente en la fase de control se utilizará el valor de estas variables para determinar las transiciones correspondientes en las máquinas de estados.

5.2.4.1 Método determinar_evento_movimiento()

La generación de eventos de movimiento viene determinada por tres factores. El primer factor es la detección de un obstáculo. El segundo factor es que esté activo el modo de

[48] Fuente: Elaboración propia
[49] Fuente: Elaboración propia.

búsqueda de luz. El tercer factor es que el usuario haya enviado una orden de movimiento.

Para determinar el evento de movimiento que se debe generar se atienden estas entradas de información en el orden especificado. Esto significa que la máxima prioridad es la detección de obstáculos, y aunque el usuario pida que la maceta se mueva, el evento generado será el evento obstáculo si se ha detectado un objeto.

La segunda prioridad serán las órdenes enviadas por el usuario. Para mejorar la estabilidad del sistema, si se recibe una orden de movimiento del usuario se desactivará temporalmente la búsqueda de luz, aunque se reúnan las condiciones para iniciar el proceso.

Finalmente, si no se produce ninguna de las situaciones anteriores, se realizará la función de búsqueda de luz. Dicha función determinará un evento de movimiento en función de la luz recibida.

Tomando en cuenta estas consideraciones se ha realizado la siguiente codificación. (Se ha ocultado el contenido del switch para una lectura más sencilla. El switch se encarga de determinar el evento según la orden de movimiento)

```
361 //Metodo para determinar el evento de movimiento
362 void determinar_evento_movimiento() {
363   evento_movimiento=EVM_NINGUNO;
364   //Si detectamos un obstaculo emitimos evento obstaculo
365   if(deteccion1==true || deteccion2==true){
366           evento_movimiento=EVM_OBSTACULO;}
367   //Si no hay obstaculo comprobamos las ordenes recibidas
368   else{
369     if(mensaje_movimiento!=NINGUN_MENSAJE){
370         contador_mensaje=millis(); //Se desactiva la búsqueda de luz temporalmente
371         buscando_luz=false;
372         switch(mensaje_movimiento){
390       }//Fin comprobacion mensajes
391     //Si no hay obstaculo ni mensaje realizamos la busqueda de luz
392     else{
393       if(millis()-contador_mensaje>umbral_desactivar_luz)
394          {buscar_luz();}
395       }
396   }//Fin else
397 }//Fin determinar_evento_movimiento
```

Ilustración 48. Método determinar_evento_movimiento[50].

5.2.4.2 Método determinar_evento_deposito()

La generación de eventos del depósito representa un caso más sencillo que el que se acaba de describir. Sólo hay una fuente de información para determinar el evento, y son los sensores de nivel. Dependiendo del valor de las lecturas se emitirá el evento correspondiente (Véase Control depósito). A continuación se muestra la implementación:

[50] Fuente: elaboración propia.

```
622  //Metodo para determinar los eventos del deposito
623  void determinar_evento_deposito() {
624    if (sensor_superior_val == HIGH) {
625      evento_deposito = EVD_LIMITE_SUPERIOR;
626    }
627    else {
628      if (sensor_inferior_val == HIGH) {
629        evento_deposito = EVD_LIMITE_INFERIOR;
630      }
631      else {
632        evento_deposito = EVD_NINGUNO;
633      }
634    }
635  }//Fin determinar_evento_deposito
```

Ilustración 49. Método determinar_evento_deposito[51].

5.2.5 Capa de control

En este apartado se explica el funcionamiento general del código. Para ello, se irán explicando las distintas fases que se ejecutan en el bucle principal (Véase Ilustración 16).

```
775  void loop () {
776    //0-Comunicaciones
777    //1-Leer variables comunicación
778      leer_variables_comunicacion();
779      actualizar_EEPROM();
780    //2-Leer sensores
781       leer_sensores();
782    //3-Control
783       deposito();
784       movimiento();
785       riego();
786    //4-Escribir variables comunicación
787    escribir_variables_comunicacion();
788    //5-Comunicaciones
789  } /* end loop */
```

Ilustración 50. Bucle principal (loop).[52]

5.2.5.1 Leer variables comunes de la EEPROM

En este primer método, se realiza un recorrido de las posiciones de memoria donde se albergan las variables de comunicaciones. Esto permite leer todas las órdenes que hayan llegado de la capa de comunicaciones.

En el momento de implementar este método todavía no se han definido las variables de la capa de comunicaciones. Sin embargo se ha realizado una implementación que realiza la lectura de unos pocos valores para simular el escenario final.

[51] Fuente: Elaboración propia.
[52] Fuente: Elaboración propia.

```
247 void leer_variables_comunicacion(){
248   leer_byte_EEPROM(POS_MENSAJE_RIEGO,mensaje_riego);
249   leer_byte_EEPROM(POS_MENSAJE_MOVIMIENTO,mensaje_movimiento);
250   leer_byte_EEPROM(POS_MENSAJE_ACTUALIZACION,mensaje_actualizacion);
251 }//Fin leer_variables_comunicacion
```

Ilustración 51.Lectura de variables de comunicación.

Para leer las variables se hace uso a la vez de un método sencillo que hace la lectura de una dirección de memoria dada y deja su contenido en la variable especificada.

```
219 //Método para leer de la EEPROM
220 void leer_byte_EEPROM(int direccion, byte dato){
221   EEPROM.get(direccion,dato);
222 }//Fin leer byte EEPROM
```

Ilustración 52. Método leer_byte_EEPROM

Este método hace uso de la función "*get()*" de la librería EEPROM, que deja el contenido de la dirección especificada en la variable que se le pasa cómo parámetro. La ventaja de éste método es que según el tipo de variable ajusta la cantidad de bytes a leer de la EEPROM.

5.2.5.2 Leer sensores

El siguiente paso es realizar la lectura de los sensores. Mediante un método que engloba la lectura de todos los sensores se actualizan los valores de todas las variables. (Para ver cómo se realiza dicha lectura véase Lectura de sensores.)

5.2.5.3 Control

En esta fase se toman las decisiones a partir de la información recibida en las variables de comunicación y las lecturas de los sensores. El control se ha dividido en tres métodos: movimiento, depósito y riego. A continuación se va a explicar la función de cada uno de ellos.

5.2.5.3.1 Método movimiento()

El método movimiento() se encarga de gestionar la máquina de estados que controla el movimiento de la maceta. Para implementar los diferentes estados y las transiciones posibles se hace uso de las instrucciones *switch* y *case*. Cada *case* del *switch* general representa un estado. Dentro de cada estado hay otro switch que permite discriminar a partir del evento actual. De este modo en cada llamada al método primero se selecciona el estado actual y después según el evento generado se activa el *case* correspondiente a la combinación de estado y evento del switch anidado dentro del case del estado actual .La implementación de un estado y sus posibles transiciones la siguiente:

```
case ESTM_PARADO:
      switch (evento_movimiento) {
        case EVM_ADELANTE:
       /*Serial.println("Estabamos en estado parado y nos ha llegado el evennto
       izquierda. Cambiamos estado a moviendo_izquierda");*/
          mover_adelante();
          estado_movimiento = ESTM_MOVIENDOSE_ADELANTE;
          break;

        case EVM_ATRAS:
```

```
        /*Serial.println("Estabamos en estado parado y nos ha llegado el evento
        izquierda. Cambiamos estado a moviendo_izquierda");*/
            mover_atras();
            estado_movimiento = ESTM_MOVIENDOSE_ATRAS;
            break;

          case EVM_DERECHA:
        /*Serial.println("Estabamos parado y nos ha llegado el evento derecha.
        Cambiamos estado a moviendo_derecha");*/
            girar_derecha();
            estado_movimiento = ESTM_MOVIENDOSE_DERECHA;
            break;

          case EVM_IZQUIERDA:
        /*Serial.println("Estabamos parado y nos ha llegado el evennto
        izquierda. Cambiamos estado a moviendo_izquierda");*/
            girar_izquierda();
            estado_movimiento = ESTM_MOVIENDOSE_IZQUIERDA;
            break;



          default: break;

        } break;//Fin Estado Parado
```

El código adjunto representa el estado parado y sus posibles transiciones. Por ejemplo, si el estado actual es parado, el switch general entrará por las líneas de código mostradas. Una vez dentro, ya se ha determinado en qué estado estamos. El siguiente paso es determinar la transición. Si por ejemplo llega el evento obstáculo, no se realizará ninguna acción ya que no hay ninguna transición que vaya desde el estado parado a otro estado con el evento obstáculo. Es por esto que en el *switch* no hay ningún *case* para dicho evento. En cambio, si llegara el evento EVM_IZQUIERDA, se realizarían las acciones definidas en el último *case*. Esto implica cambiar de estado y realizar las acciones asociadas a dicho estado.

En cada iteración del bucle, dependiendo del estado actual y del evento que se haya generado se realiza la transición al siguiente estado. Así, esta codificación se extiende a todos los estados y eventos posibles, siguiendo el diseño realizado en el anterior capítulo.

5.2.5.3.2 Método deposito()

La implementación de este método es muy similar a la del método movimiento descrito recientemente. En este método se definen los estados posibles del depósito y sus transiciones. Los estados del depósito no realizan ninguna acción.

Para determinar el evento del depósito se hace uso del método determinar_evento_deposito(). Este método toma como entrada las lecturas de los sensores de nivel del depósito. En función de los valores leídos se emite el evento correspondiente (Véase Eventos depósito).

5.2.5.3.3 Método riego()

Este método se encarga de gestionar la activación de la bomba de agua. El funcionamiento de éste método es muy sencillo, ya que sólo existen dos casos en los que se activa la bomba. El primer caso es que la humedad detectada por el sensor de humedad de tierra sea inferior al umbral. El segundo caso es que llegue la orden de

activar el riego. En ambos casos se comprueba el estado del depósito. Si el estado del depósito lo permite, se acciona el riego.

De manera similar, en caso de que la humedad del suelo supere le umbral o que llegue un mensaje para detener el riego, se apagará la bomba de agua.

5.2.5.3.4 Método buscar_luz()

El objetivo de éste método es determinar a qué dirección debe moverse el macetero para encontrar una zona con mayor exposición solar. Para ello se utiliza un umbral a partir del cual se considera que se ha encontrado una zona soleada.

Mediante la información proporcionada por las dos resistencias LDR, se emiten eventos de movimiento que indican al sistema en qué dirección debe moverse. En caso de que se detecte una cantidad suficiente de luz en ambas resistencias el macetero se para y permanece quieto.

Si durante un tiempo fijado en la configuración no se consigue alcanzar las condiciones de luz buscadas, se cambia a una búsqueda más pesimista. Para ello se toma como valor objetivo el máximo valor de luz que se haya detectado durante la búsqueda anterior.

```
341 //método para determinar el evento de movimiento cunado se está buscando luz.
342 void buscar_luz(){
343   //Colocacion: LDR1->izquierda  LDR2->derecha
344   //Actualización máximos encontrados
345   if(LDR1_val<max_luz_actual){max_luz_actual=LDR1_val;}
346   if(LDR2_val<max_luz_actual){max_luz_actual=LDR2_val;}
347   //Si llevamos un rato buscando luz y no encontramos, cmabiamos a una búsqueda más pesimista.Utilizamos como meta el mejor valor que hayamos detectado.
348   if(millis()-tiempo_busqueda>umbral_busqueda_optimista && buscando_luz){
349     umbral_luz=max_luz_actual;
350     tiempo_busqueda=millis();}       //Actualizamos contador para el tiempo de busqueda
351   if(abs(LDR1_val-umbral_luz)<40 && abs(LDR2_val-umbral_luz)<40){evento_movimiento=EVM_PARAR; buscando_luz=false;}
352   else{
353           buscando_luz=true;                                          //Variable para saber si estamos o no buscando luz
354           if(LDR1_val-LDR2_val>10){evento_movimiento=EVM_DERECHA;}     //Hay más luz en la derecha
355           else if(LDR2_val-LDR1_val>10){evento_movimiento=EVM_IZQUIERDA;} //Hay más luz en la izquierda
356                else{evento_movimiento=EVM_ADELANTE;}                   //No se detecta luz en ninguna dirección concreta
357   }
358 }//Fin buscar_luz
```

Ilustración 53. Método buscar_luz[53].

La implementación de este método se ha decidido tras realizar varias pruebas. Esta funcionalidad es una de las que mayor dificultad entraña debido a que los factores ambientales modifican la luz constantemente. Además el valor de la luz va variando durante el día por lo que no se puede trabajar con valores fijos.

5.2.5.4 Escribir variables de comunicación.

La última fase del bucle consiste en actualizar las variables de comunicación. De este modo se deja información sobre las acciones que ha realizado el control, para que puedan ser utilizadas en el apartado de comunicaciones. Para ello, se hace uso de los métodos de escritura en EEPROM que se han presentado anteriormente.

5.3 Implantación

5.3.1 Instalación

[53] Fuente: Elaboración propia.

La primera parte de este proyecto consiste en el montaje de todo el hardware. Para ello, empezaremos montando los elementos estructurales, la plataforma móvil que albergara todo el conjunto y el depósito de agua.

Para montar la plataforma se ha empleado un soporte para macetas con ruedas.

Ilustración 54.Soporte para macetas.

Para poder adaptarlo al proyecto, se han instalado dos soportes para motores. Sólo hemos dejado una de las tres ruedas existentes, a la cuál ha sido necesario añadirle un suplemento para equilibrar la altura con las ruedas del motor. El resultado con los motores ya montados visto desde la parte de abajo es el siguiente:

Ilustración 55. Montaje del soporte[54].

Una vez montado el soporte que albergará todos los demás componentes hay que montar el depósito. Para ello, hay que colocar la bomba de agua y los sensores en un depósito.

Se ha elegido un depósito transparente para poder ver mejor los componentes. Tras realizar agujeros para poder instalar los componentes, el resultado es el siguiente:

[54] Fuente: Elaboración propia

Ilustración 56.Montaje del depósito[55].

Como puede verse en la imagen también se ha instalado un tubo conectado a la bomba que permite redirigir la salida del agua y se han anclado los sensores en las posiciones correspondientes para detectar el nivel del líquido. La colocación de los sensores de nivel es importante para cumplir con el diseño del apartado anterior. Nótese que el sensor inferior se ha colocado para que se abra cuando el nivel del líquido es superior a él. El sensor superior se comporta de modo inverso.

A continuación, se deben instalar los sensores y el microcontrolador.

Para realizar el montaje de estas piezas se han utilizado tres *protoboards*, para organizar mejor las conexiones. El montaje de los sensores se realiza cómo ya se ha explicado en el apartado anterior. Tras conectar todos los sensores al microcontrolador, el resultado es el siguiente:

[55] Fuente: Elaboración propia

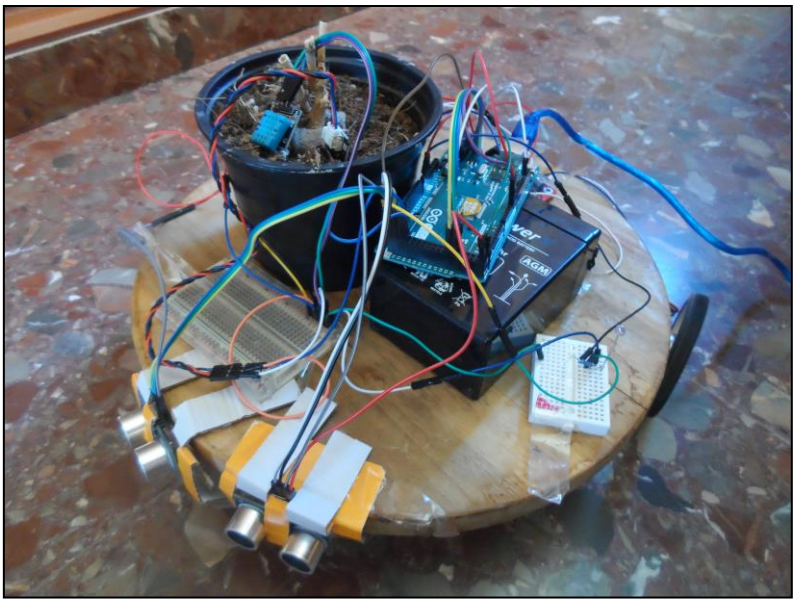

Ilustración 57. Montaje del prototipo.[56]

La colocación de los sensores es importante para determinar la efectividad de los sensores de ultrasonidos. También es determinante la colocación de las LDR, que afecta directamente a la funcionalidad de búsqueda de luz. En este prototipo se han colocado dos LDR, una a la izquierda y otra a la derecha. En el caso de los ultrasonidos, se han colocado en la parte delantera, de manera que cubran la mayor parte posible.

En este punto aparece con un problema estructural. El peso de la batería combinado con el peso del depósito de agua lleno de líquido es demasiado grande. Este peso pone en riesgo la integridad de las ruedas.

A partir de aquí sería necesario cambiar a unas ruedas de mayor tamaño. Sin embargo ha resultado casi imposible encontrar unas ruedas de mayor tamaño debido a que los ejes del motor no son estándar. Por este motivo el montaje del sistema se va a dejar así, dejando para la próxima versión el montaje en una estructura más fuerte que pueda soportar el peso de todos los componentes.

5.3.2 Configuración

En este apartado vamos a repasar los distintos valores de configuración de sistema, tales como valores por defecto de variables o umbrales.

5.3.2.1 Velocidad máxima

La variable potencia configura la velocidad a la que se mueve el sistema. Como la configuración de la velocidad se hace mediante el envío de un pulso PWM al puente en H, los valores de esta variable deben oscilar entre 0 y 255. Dado el rango de valores que se pueden tomar esta variable se declara de tipo *byte* ya que este tipo de variables tiene el mismo rango de valores mencionado.

El valor por defecto de la variable potencia es 250. Este valor, que en principio puede parecer elevado, se debe a que se ha utilizado una fuente de alimentación de 6V, que es la mitad de la potencia recomendada para los motores. Por ello, para lograr que los

[56] Fuente: Elaboración propia.

motores se muevan, es necesario dar la máxima potencia. Es importante modificar el valor de la variable potencia si se cambia la fuente de alimentación.

5.3.2.2 Umbral obstáculo

La variable umbral_obstaculo determina la distancia en centímetros a partir de la cual se considera que se ha detectado un obstáculo. Esto quiere decir que si la distancia a la que se detecta un objeto es menor al umbral, se considera que se ha detectado una colisión y se lanza el evento obstáculo, que activa la rutina de evitación de colisiones.

El valor de esta variable afecta directamente al comportamiento del macetero, ya que si fijamos un valor muy pequeño puede que cuando se detecte la colisión sea demasiado tarde. Si por el contrario fijamos un valor demasiado grande, se considerará cualquier objeto lejano cómo un obstáculo. El valor por defecto de esta variable es 15 centímetros. Este valor se ha obtenido de manera experimental a partir de las pruebas de movimiento realizadas.

5.3.2.3 Umbral luz

La variable umbral_luz configura el valor para el cual se considera que la maceta está en un sitio soleado. El valor de esta variable determina el comportamiento de la funcionalidad de buscar luz. El valor de esta variable por defecto es 890, y se ha obtenido de manera experimental. En versiones futuras de éste proyecto lo ideal sería que el valor de esta variable se determinará después de realizar una exploración del entorno y en función del tipo de cultivo que alberga la maceta.

5.3.2.4 Tiempo actualización memoria.

Uno de los factores determinantes en la utilización de la memoria EEPROM es su vida útil. Según el fabricante la vida media de una celda de la EEPROM es de unas 100.000 escrituras. Por ello se hace necesario reducir en la medida de lo posible las escrituras en memoria. La variable umbral_actualizacion configura el tiempo en milisegundos que tarde en escribirse la información de los sensores en la EEPROM. Por defecto la variable umbral_actualizacion toma el valor 300000, que equivale a cinco minutos.

5.4 Resultados de las pruebas

5.4.1 Prueba lectura

Las primeras pruebas que se han realizado han sido la lectura de todos los sensores de que disponen los maceteros. El objetivo de estas pruebas sencillas ha sido comprobar el funcionamiento de los sensores y aprender a interpretar los valores que proporcionan. A partir de éstas pruebas se han ido desarrollando todos los métodos que implementan las lecturas de cada sensor.

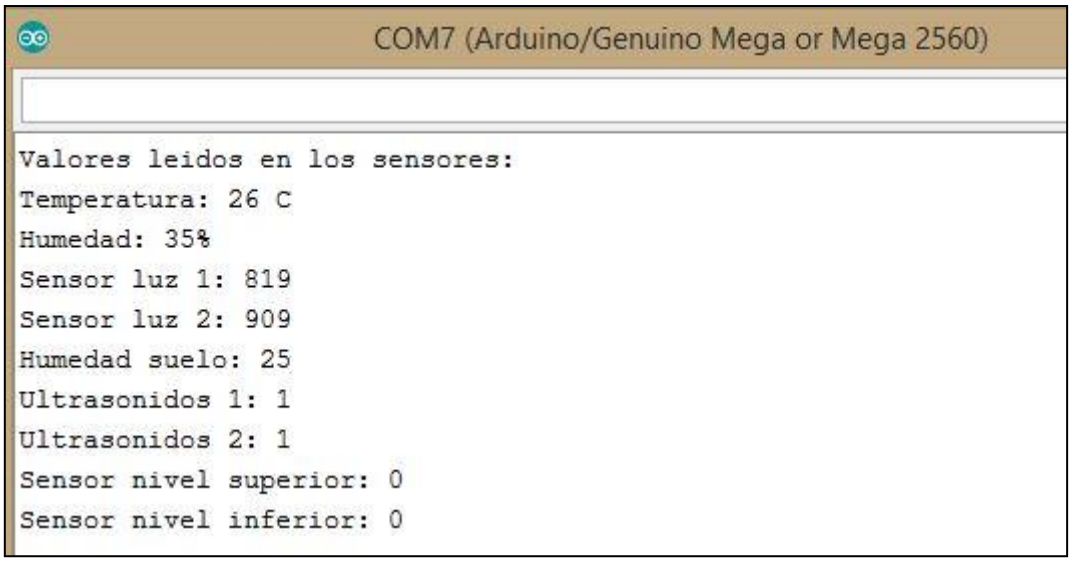

Ilustración 58. Lectura de sensores.[57]

5.4.2 Prueba riego

En esta prueba vamos se comprueba el funcionamiento del depósito de agua y la funcionalidad de riego. Como resultado se obtiene la confirmación del funcionamiento de la máquina de estados del depósito. También sirve para verificar que se activa el riego cuando las condiciones de humedad del suelo lo requieren.

Ilustración 59. Nivel mínimo depósito[58]

Tras realizar las pruebas comprobamos que el sistema tiene el comportamiento deseado; se activa el riego si las condiciones de humedad lo exigen y se desactiva en cuando se alcanza el nivel mínimo del depósito.

5.4.3 Prueba de movimiento

Una vez se han probado todos los sensores, la siguiente prueba ha sido la prueba de movimiento. El primer objetivo de esta prueba era verificar que el sistema se desplazaba en las direcciones solicitadas. Esta primera prueba se realiza sólo con los motores, el puente en H, y el controlador Arduino. Los resultados de la prueba son

[57] Fuente: Elaboración propia.
[58] Fuente: Elaboración propia.

satisfactorios, y además sirven para determinar qué velocidad se debe configurar. Es importante tener en cuenta que depende de la alimentación de los motores la velocidad puede variar. En la prueba se ha realizado la alimentación de los motores con 6V.

Una vez superado el primer objetivo, el segundo objetivo es abordar la detección y evitación de obstáculos. Para poder obtener una buena detección de obstáculos es crucial la colocación de los sensores de ultrasonidos.

Estas pruebas han servido para determinar la ubicación de los sensores de ultrasonidos. También han ayudado a decidir la estrategia a emplear una vez se localiza un obstáculo.

La primera estrategia fue activar el giro en función de la posición de obstáculo detectado. Si se detecta un obstáculo en el sensor izquierdo, se activa el giro hacia la derecha y viceversa. Sin embargo esta estrategia resulta ineficaz cuando la maceta se encuentra en una esquina, ya que gira alternativamente a izquierda y derecha y no es capaz de salir de esa posición.

La segunda estrategia implementada, y la definitiva, consiste en dos acciones. Cuando se detecta un obstáculo, en cualquier posición, se activa la marcha atrás durante un período de tiempo y a continuación se activa el giro a la derecha durante otro período de tiempo. Para ofrecer más posibilidades, por cada dos veces que se gira a la derecha se hace un giro a la izquierda. Esto significa que si se detectan tres obstáculos, en las dos primeras detecciones el robot hace marcha atrás y gira a la derecha, pero en la tercera detección hace marcha atrás y gira a la izquierda. En la siguiente detección se vuelve a aplicar el giro a la derecha y así sucesivamente. Esta estrategia supone una gran mejora respecto a la anterior. Ahora el sistema es capaz de moverse correctamente en la gran mayoría de los casos.

Como resultado de estas pruebas se comprueba que el sistema se comporta de manera correcta en entornos con obstáculos grandes y estáticos, como pueden ser paredes. En caso de otros obstáculos que pueden estar a diferentes alturas, en movimiento o que son más pequeños, la efectividad varía, dándose casos en que no se llegan a detectar algunas colisiones.

5.4.4 Prueba búsqueda de luz

El objetivo de esta prueba es comprobar el funcionamiento de la búsqueda de luz. Para ello, se ha situado el macetero en un pasillo oscuro, con la intención de que sea capaz de encontrar una zona con más luz. En el extremo del pasillo hay una habitación que reúne las condiciones necesarias de luz.

Ilustración 60. Situación inicial prueba de luz.[59]

Sin duda la funcionalidad de buscar luz es la parte más difícil del comportamiento del sistema. Las condiciones de luz pueden variar mucho a lo largo del día y no es fácil detectar si hay luz más allá de una esquina o no.

Tras ejecutar la prueba, obtenemos en la mayoría de los casos un resultado satisfactorio. El macetero es capaz de llegar a la habitación con luz.

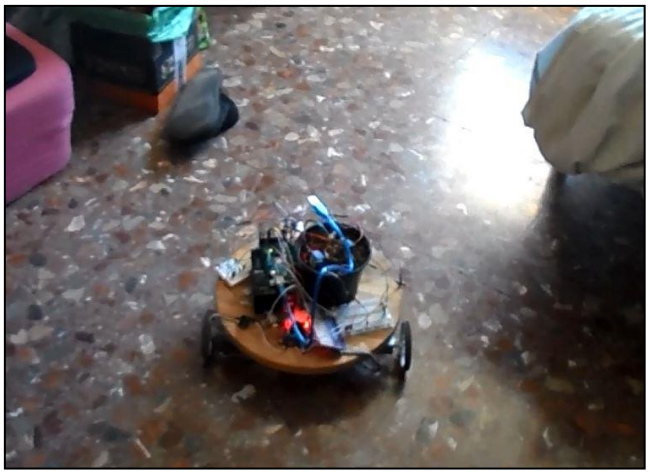

Ilustración 61. Posición final prueba de luz.[60]

Sin embargo, algunas veces la política de evitación de obstáculos interfiere mucho con la búsqueda de luz y se producen acciones contradictorias. En algunos casos el macetero no es capaz de encontrar la mejor zona y acaba conformándose con la iluminación que tiene disponible.

5.4.5 Prueba integración.

El objetivo de esta prueba es comprobar el funcionamiento del sistema una vez se han integrado todos los elementos desarrollados por separado.

Los módulos que se han desarrollado de manera independiente han sido las máquinas de estados (depósito y movimiento), el control de motores, la gestión de la memoria EEPROM y las lecturas de sensores.

[59] Fuente: Elaboración propia.
[60] Fuente: Elaboración propia.

Para realizar la integración se han migrado los métodos desarrollados a un único código, incluyendo todas las variables y definiciones necesarias.

Uno de los aspectos más importantes de esta prueba es verificar que el tiempo de reacción del sistema es suficientemente recudido. El principal motivo por el que esto podría provocar problemas es que el código se ha programado de manera secuencial, ya que Arduino no permite paralelismo real. No obstante, dado que las lecturas de los sensores se hacen en un tiempo muy reducido, no debería haber ningún problema.

Una vez realizada la integración el resultado es un código de unas 850 líneas aproximadamente. Tras repetir las pruebas de lectura de sensores y movimiento, comprobamos que efectivamente el sistema se comporta en tiempo real, y que pese a estar programado secuencialmente, no se aprecian problemas en la ejecución.

5.4.6 Prueba comunicaciones

En esta prueba vamos a comprobar que realmente se están leyendo y ejecutando las órdenes recibidas. Para simular el entorno final, esta prueba se realiza después de realizar la integración del código, de modo que se ejecuta con el código final del macetero. En el momento de realizar esta prueba todavía no estaban implementadas las capas de comunicación por lo que se ha realizado una simulación. Para realizar dicha simulación hemos conectado un *joystick* a nuestro controlado Arduino. Con este joystick mandaremos órdenes de movimiento que se escribirán en la EEPROM (Tal y como se hará en el proyecto).

Para poder realizar la prueba se utilizan las siguientes variables:

Variable	Posición en memoria (en bytes)	Tipo
mensaje_riego	2000	byte
mensaje_movimiento	2001	byte
mensaje_actualizacion	2002	byte

Tabla 29. Variables prueba comunicación.

Estas variables tienen el siguiente significado:

mensaje_riego	
Valor	Descripción
0	Ninguna orden
1	Parar riego
2	Activar riego

Tabla 30. Interpretación valores mensaje_riego.

mensaje_movimiento	
Valor	Descripción
0	Ninguna orden
1	Parar motores
2	Avanzar hacia adelante
3	Girar derecha
4	Girar izquierda

Tabla 31.Interpretación mensaje_movimiento.

mensaje_actualizacion	
Valor	Descripción
0	Ninguna orden
1	Escribir valor sensores en EEPROM

Tabla 32.Interpretación mensaje_actualizacion.

Para simular correctamente un escenario de comunicaciones tal y como se ha diseñado el proyecto hemos añadido el siguiente código:

```
831 void loop () {
832   //0-Comunicaciones
833   /*Simulación comunicaciones*/
834   int x=analogRead(7);
835   int y=analogRead(6);
836   int boton =digitalRead(38);
837   if(boton==0){escribir_byte_EEPROM(POS_MENSAJE_MOVIMIENTO,NINGUN_MENSAJE);}
838     else if(x>900){escribir_byte_EEPROM(POS_MENSAJE_MOVIMIENTO,GIRAR_DERECHA);}
839          else if(x<200){escribir_byte_EEPROM(POS_MENSAJE_MOVIMIENTO,GIRAR_IZQUIERDA);}
840               else if(y>800 ){escribir_byte_EEPROM(POS_MENSAJE_MOVIMIENTO,MOVER_ADELANTE);}
841                    else escribir_byte_EEPROM(POS_MENSAJE_MOVIMIENTO,NINGUN_MENSAJE);
842
```

Ilustración 62.Simulación comunicaciones[61]

Mediante las líneas expuestas, se leen los valores del joystick y se escriben en la EEPROM en la dirección donde se alojaría un mensaje real.

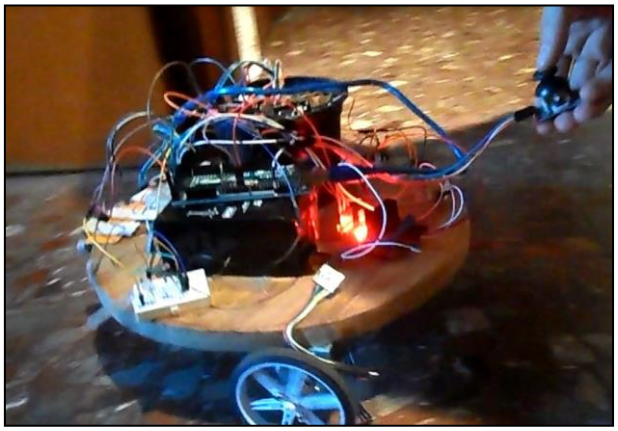

Ilustración 63. Maceta controlada mediante Joystick[62].

El sistema responde moviéndose en la dirección deseada, siempre que esté dentro de los movimientos posibles. Además también impide el movimiento contra obstáculos, tal y como se quería hacer. Adicionalmente también se desactiva la búsqueda de luz cuando se envía una orden de movimiento.

En conclusión, los resultados de esta prueba han sido exactamente los que queríamos conseguir.

5.5 Conclusiones

En este capítulo se ha puesto en práctica todo el diseño elaborado en el capítulo anterior.

Se ha comenzado especificando el hardware empleado y mostrando los esquemas de conexión.

[61] Fuente: Elaboración propia.
[62] Fuente: Elaboración propia.

A continuación, el foco de atención se ha centrado en la parte software. Primero se ha visto cómo gestionar la memoria EEPROM. A continuación se han desarrollado los métodos encargados de leer los sensores. Después se ha implementado la lógica de control.

A continuación se da paso al montaje del sistema, teniendo que realizar labores de bricolaje y tomar decisiones importantes como la colocación de los sensores. Por motivos estructurales se ha tenido que dejar fuera del prototipo el depósito de agua.

Finalmente se realizan las pruebas necesarias para verificar que el sistema cumple con las funcionalidades deseadas. Se han cubierto completamente todas las funcionalidades necesarias. No obstante, cabe destacar que las funcionalidades de búsqueda de luz y movimiento admiten todavía mejoras en su implementación.

6 Conclusiones

En este capítulo se realiza un repaso a todo el documento. Se resumen las tareas que se han realizado, los problemas encontrados y las posibles ampliaciones.

6.1 Trabajo realizado

La primera labor a la hora de realizar este proyecto ha sido realizar un estudio de mercado. A partir de este estudio se han encontrado sistemas similares al que se pretendía desarrollar. Dichos sistemas han servido de punto de partida para conocer qué características implementar.

Una vez decidido lo que se quería hacer, el siguiente paso ha sido realizar una especificación formal. Para ello se ha redactado una especificación de requisitos basada en el estándar IEE 830.

A continuación, se empieza mostrando el funcionamiento que se iba a implementar de modo general. Después se han descrito los elementos hardware necesarios. Finalmente, se ha modelado el control del depósito y el control de movimiento como máquinas de estados UML.

Una vez realizado todo el diseño, el último paso ha sido la implementación y las pruebas.

Se han implementado todos los métodos necesarios para realizar la lectura de sensores. También se han implementado políticas para reducir el uso de la memoria y aumentar su vida útil. Por otro lado, la implementación de las máquinas de estados ha resultado bastante mecánica. La mayor dificultad ha sido determinar el evento de movimiento que debía emitirse en cada momento ya que esto involucra tanto a la funcionalidad de evitación de obstáculos como a las de recepción de órdenes y búsqueda de luz.

Una vez realizada toda la implementación, se han preparado pruebas para verificar el funcionamiento del código. A partir de estas pruebas se han obtenido valores de referencia y se han mejorado algunos aspectos del código.

6.2 Problemas encontrados

Durante el desarrollo de este sistema han aparecido múltiples problemas. A continuación algunos de los más determinantes.

6.2.1 Ruedas

Uno de los problemas que encontrados ha sido la dificultad para adaptar los ejes del motor a las ruedas. Debido a que los ejes no tenían una medida estándar ha resultado casi imposible encontrar unas ruedas de calidad. Pese a que el fabricante del motor vende sus propias ruedas adaptables a los motores, éstas han resultado ser demasiado poco robustas. A consecuencia de esto no se ha podido montar el depósito en el macetero ya que se veía comprometida la integridad estructural de las ruedas.

6.2.2 Problemas de memoria dinámica

Cuando empezó a realizarse la implementación, la idea inicial era dejar un modo *verbose* preparado para facilitar la lectura y la depuración de errores. Sin embargo, el texto de los mensajes que se pasa como argumento a la función "*Serial.print*" se guarda en memoria dinámica. Esto no tendría mayores efectos si se estuviera utilizando con un

ordenador o un equipo potente. En cambio, la placa Arduino cuenta con un espacio de memoria dinámica muy limitado y dicha memoria se llenaba con los mensajes. Al final se optó por descartar el modo *verbose*.

6.2.3 Búsqueda de luz

La funcionalidad de búsqueda de luz ha resultado ser una gran fuente de problemas. Las condiciones de luz varían muchísimo dependiendo del sitio, la hora y otros factores. Además las resistencias LDR sólo detectan luz en un campo bastante pequeño. No es posible detectar luz al otro lado de un tramo de sombra. Por lo que si se deja la maceta en un espacio rodeado de sombra, no será capaz de salir.

Estos problemas podrían solucionarse utilizando otras estrategias cómo la elaboración de mapas de valores o el posicionamiento GPS. Para poder implementar estas estrategias sería necesario hardware adicional y probablemente mayor potencia de cómputo.

Finalmente se ha optado por dejar una estrategia simplista que realiza una búsqueda durante un cierto tiempo y se detiene si no es capaz de encontrar los valores deseados.

6.3 Aportaciones

6.3.1 De ámbito tecnológico

La principal aportación de este trabajo en el ámbito tecnológico puede dividirse en dos partes. Primero, el código desarrollado, que permite hacer uso de todos los recursos con los que cuenta el macetero y tomar las decisiones necesarias. En segundo lugar, el prototipo creado, que es el resultado del montaje de todos los componentes. Dicho prototipo ha requerido tomar decisiones de diseño como la colocación de los sensores o el tipo de motor a emplear.

Por otro lado también se ha realizado un estudio exhaustivo de las alternativas existentes en el mercado para resolver el problema de la automatización de cultivos. A partir de este estudio se pueden comprobar las tendencias en el mercado y las características más destacadas de los diferentes sistemas.

Finalmente, también se ha realizado una labor importante de diseño. Se han modelado dos máquinas de estados que permiten implementar el movimiento y el control del depósito. También se han descrito los elementos necesarios para poder montar el sistema y se han especificado los requisitos necesarios.

6.3.2 Sociales (ambientales)

La contribución del sistema desarrollado en el ámbito social se centra especialmente en la racionalización de los recursos hídricos. A través de los sensores y los actuadores de riego se permite regar sólo cuando es necesario , gestionando el agua de forma eficiente y evitando el malgasto de ésta.

6.3.3 Personales

A nivel personal, este trabajo ha servido para desarrollar y ampliar muchos conocimientos. Primero que nada, el desarrollo del proyecto ha estado muy marcado por el trabajo en equipo. Se han celebrado numerosas reuniones para discutir los aspectos importantes del proyecto. También se ha trabajado el uso de repositorios online para poder mantener el equipo en contacto de manera ininterrumpida. Esto ha

permitido profundizar las competencias de trabajo en equipo y conocer de primera mano los numerosos problemas que surgen cuando se trabaja con más gente, pero también las ventajas que esto conlleva.

Otros de los aspectos que he aprendido en el desarrollo de este trabajo han sido los fundamentos de los sistemas basados en eventos. El sistema que se ha desarrollado se basa en este paradigma, por lo que ha sido necesario aprender cómo funcionan este tipo de sistemas y cuál es la metodología para implementarlos.

Por otra parte, también he aprendido a interactuar con elementos hardware como sensores y actuadores. En este proceso he aprendido a utilizar correctamente los sensores y aplicar procesamientos a los datos para mejorar la utilidad de los sensores. Ha sido necesario además conocer el funcionamiento de algunos circuitos eléctricos básicos.

A nivel de microcontrolador, este trabajo ha servido para profundizar mucho los conocimientos sobre microcontroladores en general, y sobre Arduino en concreto. Ha sido vital conocer aspectos cómo los distintos tipos de entradas y salidas, la memoria EEPROM, el conversor analógico a digital, las salidas PWM y todos los detalles que implica utilizar un microcontrolador de estas características.

Finalmente, y a modo más general, el hecho de enfrentarme a un proyecto de esta envergadura me ha servido para aprender muchos otros aspectos. Primero de todo, la importancia que tiene una buena planificación, organización del trabajo, y diseño. Segundo, a enfrentar los numerosos problemas que surgen a medida que se va desarrollando el problema y los cambios que obligan a hacer. Por último, he aprendido a valorar la utilidad de las herramientas que proporcionan los editores de texto para tratar grandes cantidades de información. Estas herramientas tales como referencias, marcadores, índices automáticos, etc., resultan indispensables cuando se trata con documentos extensos.

6.4 Ampliaciones futuras

Pese a que en este documento se ha realizado una implementación funcional del sistema, también han aparecido una serie de ideas para futuras ampliaciones.

La primera ampliación debería ser la inclusión de más sensores de ultrasonidos, ya que en este prototipo no se tiene en cuenta la altura del cultivo que se alberga en la maceta, y algunas colisiones no consiguen evitarse eficazmente.

Otra ampliación debería ser incorporar un sistema capaz de discriminar longitudes de onda de luz. Esto permitiría evitar que las macetas sigan luces no deseadas como por ejemplo la luz de una linterna o de un coche que pasa. Además también debería de poderse detectar una fuente de luz. Esto podría hacerse con una cámara.

Otra mejora fundamental es realizar un estudio de la estructura para poder organizar todos los componentes. Esto incluye distribuir el peso de los componentes, aislar la electrónica de posibles pérdidas del depósito y fijar todos los elementos. Por supuesto también es necesario montar unas ruedas acordes al peso de toda la estructura.

7 Referencias

7.1 Bibliográficas

[1] Serna, A., Ros, F., & Rico, J. C. (2010). Guía práctica de sensores. Creaciones Copyright SL.

[8] Martín, J. C. (2010). Instalaciones domóticas. Editex.

[10] Artero, Ó. T. (2013). Arduino: curso práctico de formación. RC Libros.

7.2 Apuntes

Sistemas reactivos. José Luís Poza Luján , Sergio Sáez Barona. Noviembre 2015

Apuntes de la asignatura DIP (2016)

7.3 Internet

[2] Jardines verticales Citysens - Citysens. (s. f.). Recuperado 24 de febrero de 2016, a partir de http://www.citysens.com/es/

[3] Growiee - The world first modular smart home garden. (s. f.). Recuperado 30 de juniode2016, a partir de http://growiee.com/

[4] Home Page - Blue Marble Inc. (s. f.). Recuperado 25 de febrero de 2016, a partir de http://bluemarbleirrigation.com/

[5] Flower pots for sale BYXAS Multifunctional Intelligent Flower Pot FP-188 - Products - Ricokohki Limited. (s. f.). Recuperado 26 de febrero de 2016, a partir de http://www.ricodigital.com/byxas-multifunctional-intelligent-p-188-a

[6] Parrot - Flower Power - intelligent wireless sensor for your plants. (s. f.). Recuperado 25 de febrero de 2016, a partir de http://www.parrot.com/usa/products/flower-power/

[7] NIMBUS. (s. f.).Recuperado 30 de febrero de 2016, a partir de http://www.nimbuspot.com/#beginning-1

[9] DIY Smart Plant pot. (s. f.). Recuperado 24 de febrero de 2016, a partir de http://www.instructables.com/id/DIY-Smart-Plant-pot/

[11] Open Source Wireless Garden Kits: Cooking Hacks Open Garden- Postscapes. (s. f.). Recuperado 30 de febrero de 2016, a partir de http://postscapes.com/open-source-wireless-garden-systems-cooking-hacks-open-garden

[12] Garduino: Gardening + Arduino. (s. f.). Recuperado 1 de marzo de 2016, a partir de http://www.instructables.com/id/Garduino-Gardening-Arduino/

[13] Kawakami, A., Tsukada, K., Kambara, K., & Siio, I. (2011). PotPet: pet-like flowerpot robot (p. 263). ACM Press. http://doi.org/10.1145/1935701.1935755