Este proyecto tiene como objetivo diseñar y desarrollar un sistema de autocultivo usando componentes de bajo consumo con el objetivo de ahorrar en agua y energía. El diseño de este sistema consta de una alta escalabilidad y puede ser usado tanto en pequeños hogares como en grandes granjas de cultivo.

# Moisterino

## Proyecto de Autocultivo para Sistemas Empotrados Distribuidos

Daniel Cabañas & Hanjie Zhu

# RESUMEN

En este proyecto se ha diseñado e implementado un sistema de autocultivo, utilizando una Raspberry Pi como controlador central y varios Arduino como controladores extremos. Los Arduino están equipados con sensores de humedad y luz para medir las condiciones del entorno de las plantas, enviando estas mediciones a la Raspberry Pi. La Raspberry Pi gestiona el riego automático de las plantas a través del control de bombas de agua basándose en las mediciones de humedad de los Arduino. Asimismo, el sistema proporciona luz artificial a las plantas mediante la activación de luces, controladas por la Raspberry Pi, cuando las mediciones de luz ambiente indican que las plantas necesitan más luz.

El sistema desarrollado es altamente escalable y puede ser utilizado tanto en pequeños hogares como en grandes granjas de cultivo. La implementación de un sistema de riego automático y un control eficiente de la iluminación permiten el ahorro de agua y energía, así como un menor esfuerzo humano para el mantenimiento de las plantas y cultivos. Los objetivos del proyecto incluyen el diseño y desarrollo de un sistema de autocultivo utilizando múltiples controladores que se comuniquen entre sí, la implementación de un consumo eficiente de agua para el riego de las plantas, y la optimización del sistema para su funcionamiento eficiente y con bajo consumo de energía. Este sistema de autocultivo puede mejorar la eficiencia y rentabilidad en la producción agrícola, contribuyendo a la sostenibilidad ambiental y económica en el sector agrícola.

# PALABRAS CLAVE

Sistemas de Autocultivo, Sistemas de Riego Automático, Sistemas Empotrados, Sistemas Distribuidos, Controlador, Arduino, Raspberry Pi, Agricultura, Ahorro Energético, Medio Ambiente

# 1 ÍNDICE

# 1. INTRODUCCIÓN

La agricultura es una actividad fundamental para la supervivencia humana y para la economía de muchos países en todo el mundo. Si nos centramos en sector agrícola, éste se enfrenta a numerosos desafíos, como pueden ser un uso ineficiente en los recursos necesarios para desarrollar las tareas de cuidado y cultivo de plantas.

El riego de los campos de cultivo es uno de los mayores consumidores de agua a nivel mundial, y gran parte de esa agua se desperdicia debido a que los sistemas de riego pueden ser muy ineficientes. Entre el despilfarro de agua y el consumo de energía que requieren estos sistemas de riego podemos destacar que no son únicamente costosos para los granjeros, sino que además tienen un impacto negativo en el medio ambiente.

El mantenimiento de las granjas agrícolas puede ser una tarea laboriosa y costosa, especialmente en áreas remotas o de difícil acceso. La labor y el esfuerzo humano necesarios para mantener todas estas infraestructuras funcionales son considerables. Por lo tanto, es necesario desarrollar soluciones innovadoras y eficientes para mejorar la sostenibilidad y la eficiencia de la agricultura, y reducir la carga sobre los trabajadores agrícolas y el medio ambiente.

En este contexto, surge la necesidad de nuevos y revolucionarios sistemas de cultivo, que puedan contribuir a mejorar la eficiencia y sostenibilidad de la agricultura. Estos sistemas pueden ayudar a optimizar el uso del agua y la energía, así como a reducir el esfuerzo y el costo del mantenimiento de las granjas.

## 1.1. MOTIVACIÓN

La motivación de nuestro proyecto surge de la necesidad de hacer frente a los desafíos actuales de la agricultura previamente mencionados. Nuestro objetivo es desarrollar un sistema que sea sostenible, eficiente y fácil de implementar, que pueda ayudar a los agricultores a optimizar el cuidado y mantenimiento de sus cultivos y reducir el esfuerzo y el costo del mantenimiento de sus granjas, reduciendo así la huella de carbono en la agricultura.

Queremos que nuestro sistema de autocultivo permita a las personas cultivar sus propios alimentos de manera más sostenible, ya que tenemos la intención de desarrollar un sistema altamente escalable, que permita adaptarse a distintas situaciones de manera práctica. Con ello alcanzaríamos a ayudar tanto a pequeños hogares como a grandes granjas de cultivo, y así proporcionar una solución accesible para todos. La escalabilidad es un factor clave para lograr la eficiencia y rentabilidad en la agricultura y garantizar que nuestra solución pueda ser adoptada por la mayor cantidad de agricultores posible.

## 1.2. OBJETIVOS

1. Diseñar y desarrollar un sistema de autocultivo que integre varios controladores interconectados, permitiendo una gestión centralizada de los cultivos y una monitorización remota de su estado.
2. Implementar una solución de riego inteligente y eficiente, que ajuste el suministro de agua a las necesidades específicas de cada planta, evitando el uso excesivo de agua y asegurando la máxima supervivencia y salud de los cultivos

3. Optimizar el consumo de energía y recursos del sistema de autocultivo usando componentes de bajo consumo.
4. Proporcionar una solución escalable y personalizable, que pueda adaptarse a diferentes tamaños de cultivo y tipos de plantas, así como a las necesidades específicas de cada usuario.

## 1.3. ORGANIZACIÓN DE LA MEMORIA

La memoria ha sido divida en los siguientes apartados.

El primero de ellos, Diseño, será utilizado para realizar una descripción del proceso de idealización del proyecto, donde se detallarán los conceptos iniciales del proyecto, las modificaciones que ha sufrido respecto al diseño inicial y la causa de estas modificaciones.

El segundo apartado, Desarrollo, realiza un resumen del funcionamiento del sistema, describe el proceso de montaje del prototipo presentado para el proyecto, y detalla cada uno de los sistemas que lo forman.

El tercer apartado es Integración, Pruebas y Resultados. Este apartado explica cómo ha sido montado el prototipo, cuáles han sido las pruebas a las que ha sido expuesto y cuáles han sido los resultados obtenidos por dichas pruebas.

La memoria finaliza con el apartado de Conclusiones y Trabajo Futuro, donde se comentarán los objetivos cumplidos en el proyecto y las posibilidades que este ofrece de cara al futuro.

# 2. DISEÑO

En esta sección del proyecto, se explicará el diseño del sistema de autocultivo inicialmente planteado.

Para ello, se presentarán los casos de uso y los diagramas correspondientes, que permiten comprender las diferentes funcionalidades del sistema, así como la interacción entre sus componentes. Con ello podremos establecer los requisitos funcionales y no funcionales que debe cumplir el sistema, identificar los componentes necesarios, su funcionalidad y su interconexión.

## 2.1. DESCRIPCIÓN DEL DISEÑO

En esta sección se hará un repaso de las ideas iniciales desde las que partió el proyecto pasando por aquellas modificaciones que ha ido sufriendo debido a inconvenientes o decisiones de diseño.

### 2.1.1. Planteamiento inicial del proyecto

Este proyecto surge de la necesidad de encontrar una alternativa a los sistemas de riego tradicionales, siendo esta alternativa una de bajo consumo energético, que fuese capaz de monitorizar y cuidar de varias plantas mediante subsistemas de riego y de iluminación.

También se buscó desde el principio poder ofrecer esta nueva solución tanto a pequeños hogares como a grandes empresas agrícolas.

Por ello, desde un primer momento se planteó un sistema con un potencial de escalabilidad muy alto, con un controlador central que gestionase toda la información al que se le pudiesen acoplar un número variables de brazos extremo, logrando que el sistema obtuviese una adaptabilidad para cada situación.

Los brazos extremos serían los encargados de controlar y gestionar los sensores que medirían la información sensible de cada planta, y estos estarían compuestos por un controlador por cada brazo más el conjunto de componentes necesarios para ello.

### 2.1.2. Centralización en el sistema de riego

Poco después de idear el proyecto, se llegó a la conclusión de que para el primer prototipo (el presentado en este proyecto), se buscaba comprobar la eficacia y funcionalidad del sistema automático de riego. Por ello se decidió implementar que las bombas de agua eléctricas estuviesen gestionadas por el controlador central, debido a que no se consideraba necesario que fueran los controladores extremo quienes gestionasen estos actuadores.

### 2.1.3. Conexiones entre los controladores

Inicialmente, y por motivos de escalabilidad, se planteó mantener una comunicación inalámbrica entre los controladores extremo y el controlador central. Entre varias posibilidades se acabó eligiendo una comunicación utilizando el protocolo de radiofrecuencia RF24. Para implementarlo, era necesario utilizar módulos de radiofrecuencia NRF24L01. Las pruebas iniciales que se hicieron entre los controladores extremo resultaron ser satisfactorias. Sin embargo, los resultados obtenidos con el

controlador central hicieron que se tuviese que revalorar otras posibilidades, debido a los inconvenientes y las incompatibilidades con ciertas librerías. Al final, por una cuestión de tiempo y de presupuesto, se eligió mantener una comunicación Serial mediante cable USB Tipo A – Tipo B.

#### 2.1.4. Diodo LED simbólico

Desde un principio, el proyecto se centró más en el riego automático que en un cultivo total de las plantas. Por ello se priorizó el Subsistema de Riego frente al Subsistema de Iluminación. Teniendo en cuenta los problemas ocasionados por los intentos de implementación del protocolo de radiofrecuencia, el presupuesto fue afectado y por lo tanto se tuvieron que sacrificar la implementación de otros componentes con fue el sistema de luces LED de espectro completo para dotar a las plantas de la iluminación necesaria en aquellos momentos de condiciones de iluminación desfavorables. Así mismo, se decidió implementar el sistema de Iluminación con un LED simbólico para poder simular este componente de luces LED de espectro completo, con la idea de que en futuras implementaciones o versiones se pudiese acoplar de manera sencilla.

## 2.2. DIAGRAMAS

A continuación, mostraremos los diagramas utilizados en el diseño y en el desarrollo del proyecto. Estos diagramas son los diagramas de clases, diagramas de casos de uso, diagrama de estados, diagramas de secuencia y diagramas de actividad o flujo.

### 2.2.1. Diagrama de Clases

El diagrama de clases que se presenta en este documento tiene como objetivo representar las clases del sistema, sus atributos, sus operaciones y métodos y la relaciones entre estos.

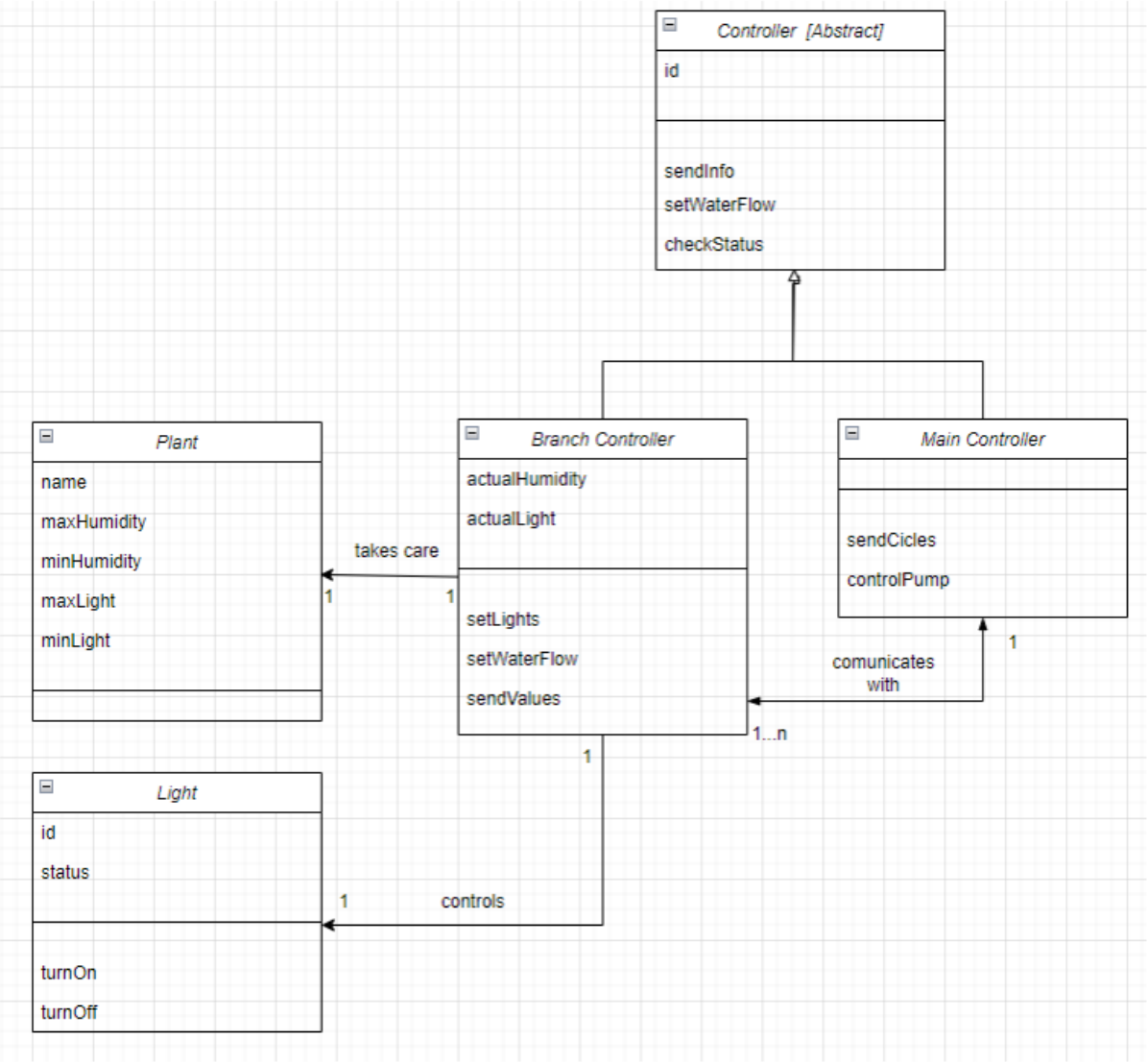


*Figura 2.1: Diagrama de Clases*

Como se puede apreciar en la figura anterior, nuestro sistema consta de cuatro clases principales: *Main Controller*, *Branch Controller*, *Plant* y *Light*.

*Main Controller* es la clase que representa al controlador central, el cual se comunica con *Branch Controller* que representa a los controladores extremo. Este último a su vez se relaciona con *Plant* y *Light* en el sentido de que realiza mediciones y captura información de estas últimas clases para luego compartirlas con *Main Controller*.

Ambos *Main Controller* y *Branch Controller* heredan de una clase abstracta *Controller* debido a las similitudes en cuanto a funcionalidad.

### 2.2.2. Diagrama de Casos de Uso

El diagrama de casos de uso se utiliza para representar las acciones que tienen lugar en el sistema representado y las interacciones entre el usuario o actor con el sistema.

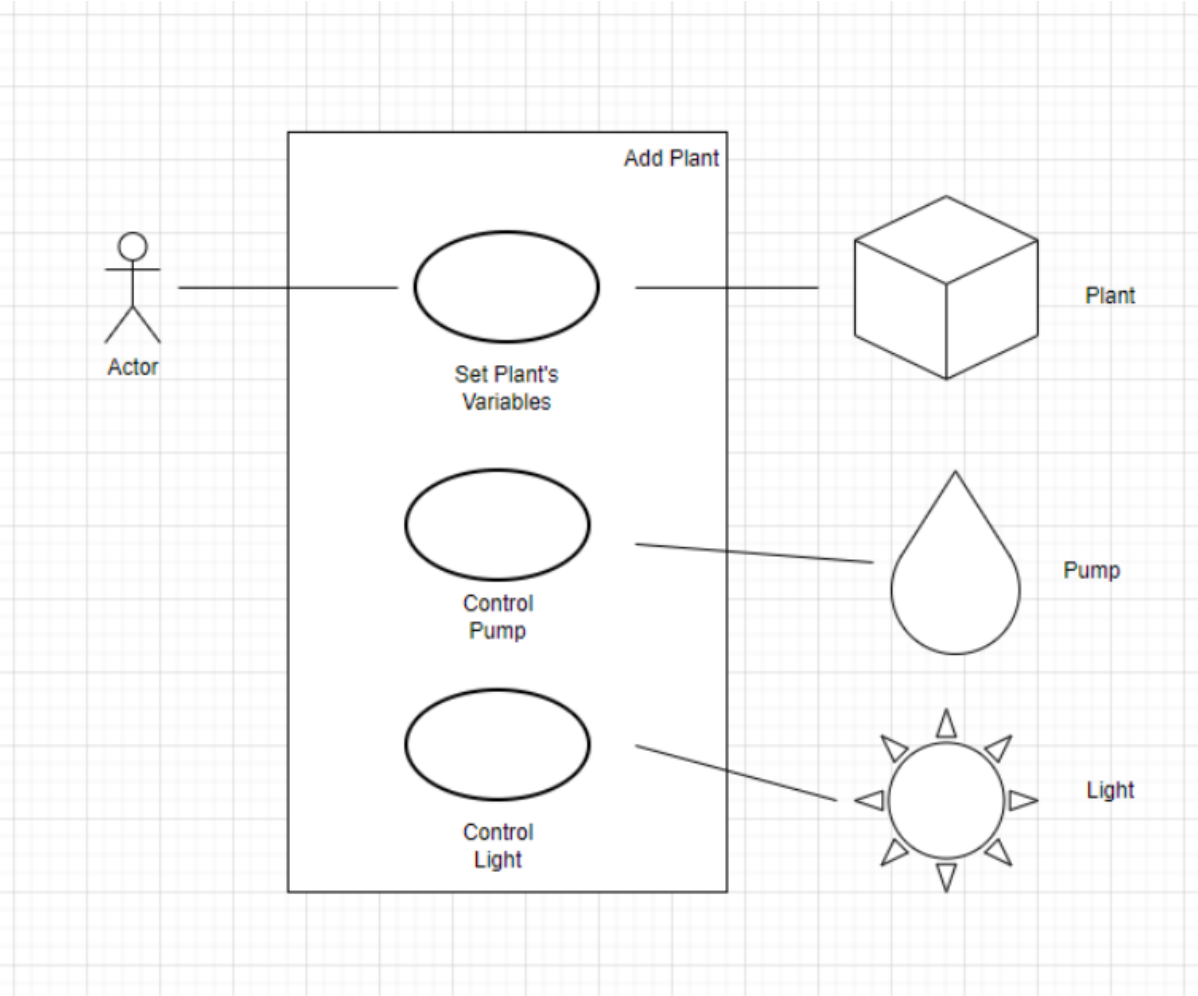


*Figura 2.2: Diagrama de Casos de Uso*

Debido a que este proyecto tiene como objetivo ser un sistema automático que busca la máxima independencia de la interacción humana, se muestra en la Figura 2.2 que las interacciones con el actor y el sistema se limitan al establecimiento inicial de valores y parámetros para cada tipo de planta, mientras que existen múltiples acciones que realiza el sistema con distintos elementos como podrían ser el uso de agua y luz o las mediciones de los valores de humedad de las plantas.

### 2.2.3. Diagrama de Estados

El diagrama de estados de este proyecto tiene como objetivo mostrar el conjunto de estados por los cuales pasa el sistema en una serie de eventos a la hora de realizar acciones.

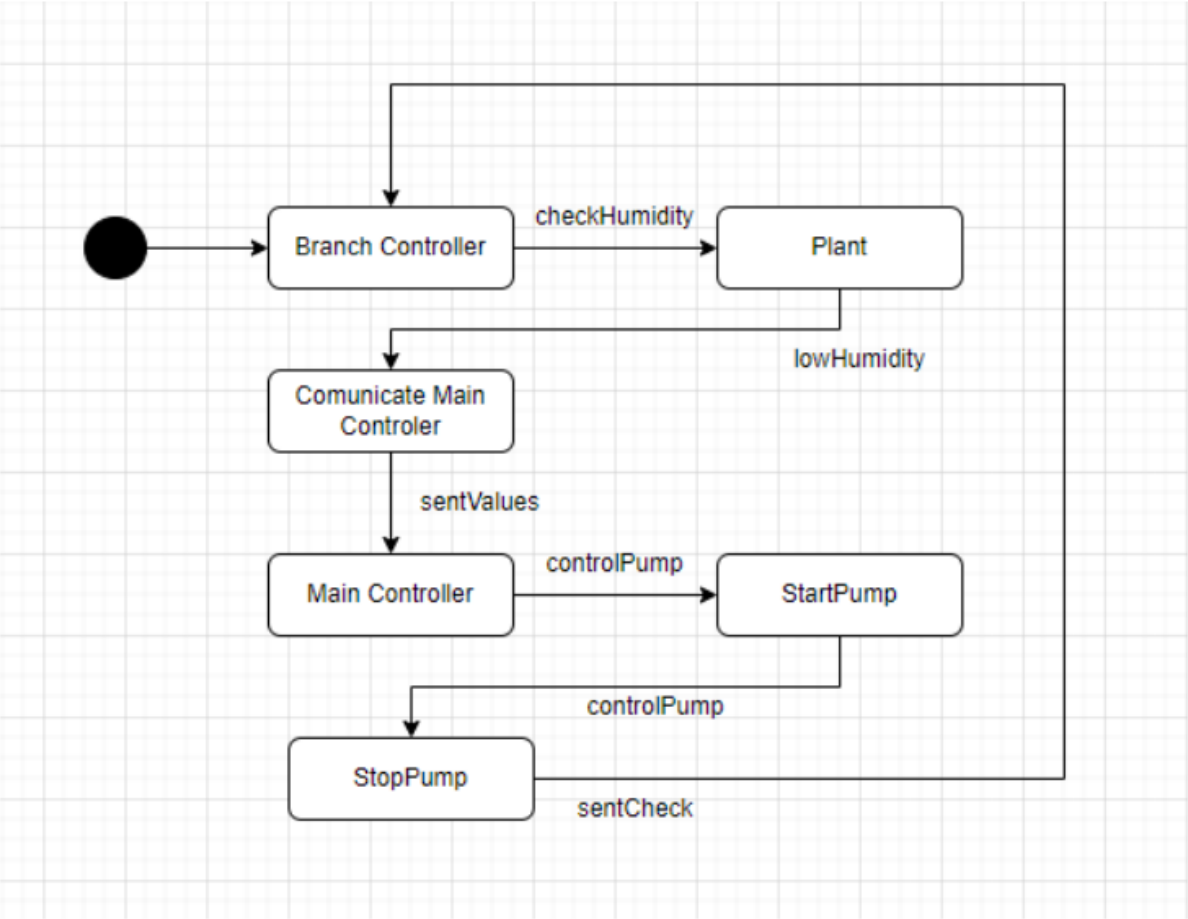


*Figura 2.3: Diagrama de Estados*

En este diagrama de estados podemos ver los estados por los que pasa nuestro sistema de riego automático desde que un controlador extremo realiza una medición de humedad en la tierra de una maceta hasta que la bomba de agua del sistema riega la maceta, representando un ciclo donde el sistema vuelve a recorrer los mismos estados debido a que es un sistema automático e independiente.

### 2.2.4. Diagrama de Secuencia

El diagrama de secuencia de este proyecto tiene como objetivo poder mostrar cómo y en qué orden colaboran entre sí las distintas partes del sistema de riego automático.

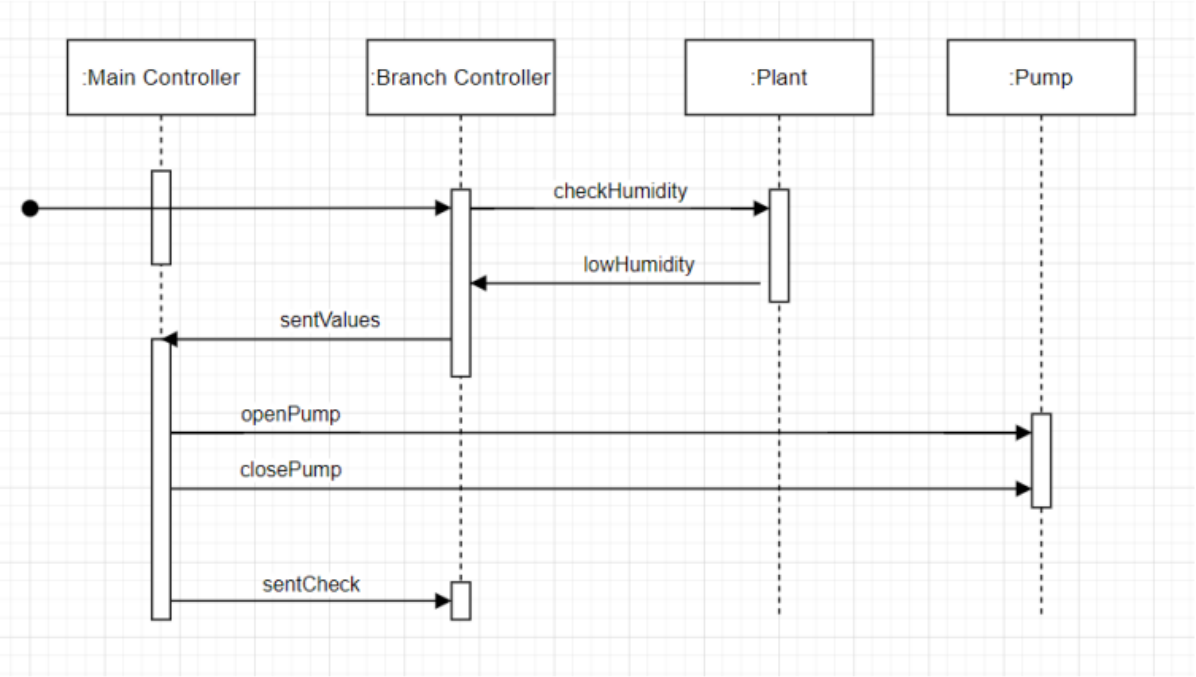


*Figura 2.4: Diagrama de Secuencia*

Como se puede observar en la Figura 2.4, este diagrama de secuencia representa el ciclo de comprobación de humedad de una planta y cómo se riega esta planta si necesita más agua. La secuencia de acciones que tienen lugar entre los distintos componentes acaba con otra llamada a la primera de las acciones representando así el ciclo automático que comenzaría otra vez tras acabar el anterior ciclo.

### 2.2.5. Diagrama de Actividad o Flujo

El diagrama de flujo de este proyecto tiene como objetivo describir el proceso que sigue el sistema para poder realizar todas las acciones y relaciones necesarias entre los componentes con el objetivo de controlar y gestionar la humedad de las plantas que el sistema estaría cuidado.

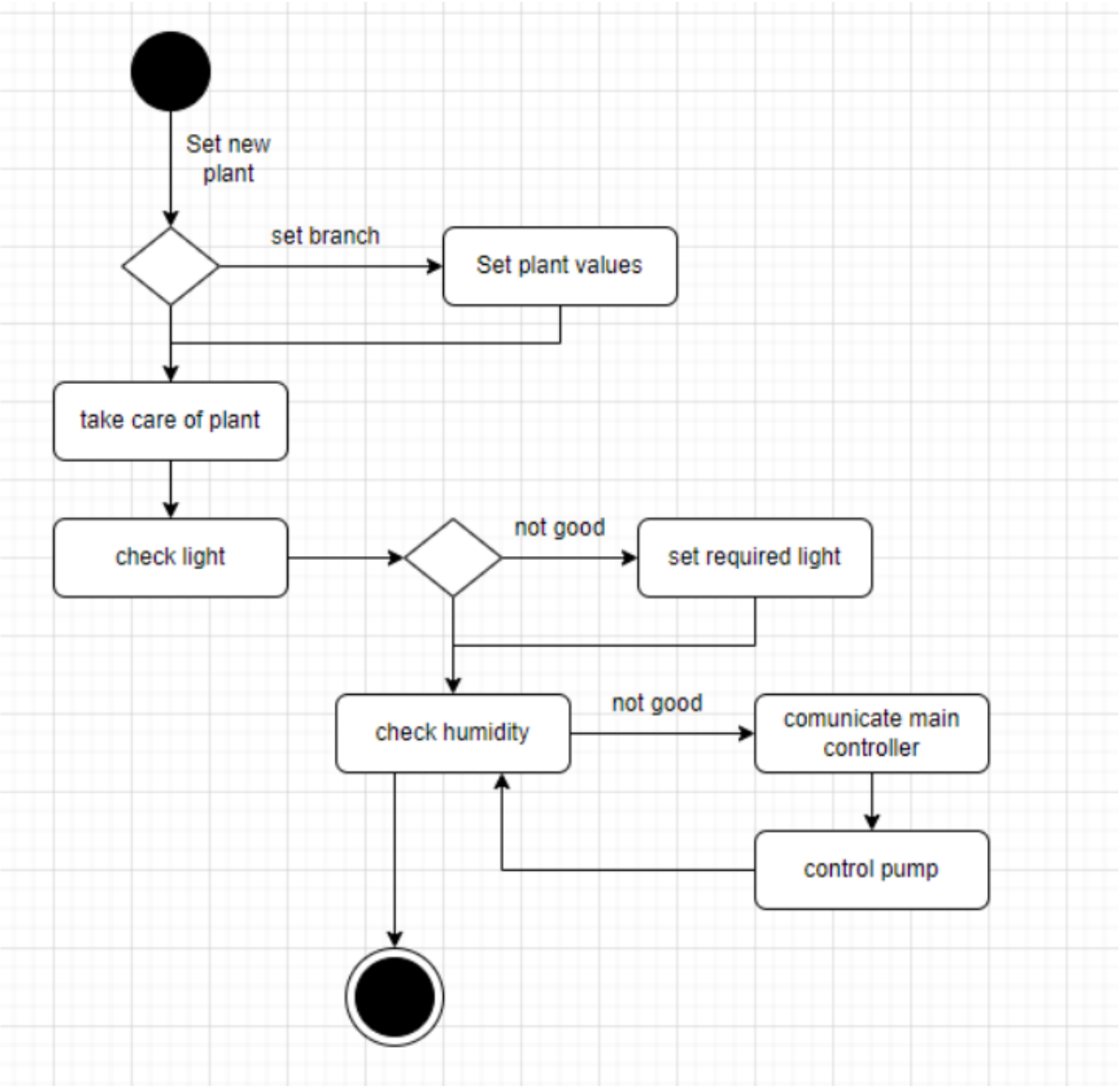


*Figura 2.5: Diagrama de Flujo*

Este diagrama describe cómo se implementan nuevas plantas en el sistema y cómo se gestionan los cuidados de estas mediante la comprobación cíclica de sus condiciones de luz y de humedad, formando bucles para reaccionar ante las condiciones indeseadas como la falta de luz o la falta de agua. Todo este diagrama tiene un carácter cíclico puesto que se implementa recurrentemente por cada planta implementada en el sistema.

# 3. DESARROLLO

En esta sección del proyecto, se presentará el proceso de desarrollo del sistema. Se describirán los pasos seguidos para llevar a cabo la implementación del sistema, desde la elección y adquisición de los componentes necesarios hasta el montaje y programación de los mismos. También se explicará cómo se han establecido las comunicaciones entre los distintos controladores, así como los protocolos de transmisión de datos utilizados. El objetivo de esta sección es proporcionar una visión detallada de todo el proceso de desarrollo del sistema, destacando los aspectos más importantes y los retos encontrados durante el mismo.

## 3.1. MONTAJE

A continuación, se comentará por orden cronológico el desarrollo y montaje de cada uno de los subsistemas que se han implementado en el proyecto. También se detallarán los inconvenientes encontrados, donde algunos de ellos han provocado cambios en el proyecto con respecto al planteamiento inicial

### 3.1.1. Subsistema de Iluminación

El Subsistema de Iluminación fue el primer sistema que se desarrolló debido a diversos motivos. El primero fue su sencillez, ya que iba a permitir una primera toma de contacto con el desarrollo del proyecto de manera simplista y cómoda, con poco margen de error. El segundo motivo fue su similitud en cuanto al Sistema de Riego, ya que ambos se basan en realizar mediciones desde los controladores extremo, enviar estas mediciones al controlador central y realizar acciones de respuesta en función de las decisiones que tome el controlador central.

#### *3.1.1.1. Medición:*

La primera parte del sistema de Iluminación consta de medir la luz ambiental a través de una resistencia fotosensible o resistencia LDR. Para ello se monta el siguiente circuito.

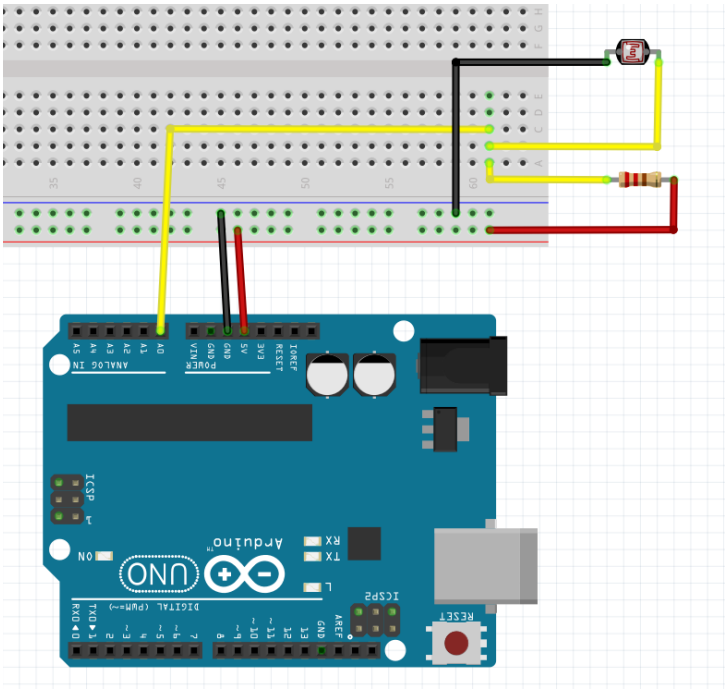

*Figura 3.1.1.1: Circuito de la primera parte del Sistema de Iluminación*

Como se puede observar en la figura anterior, se conecta una resistencia LDR en serie con otra resistencia de 10.000 ohmios (para no fundir los componentes) y se alimentan desde los pines 5V, A0 y GND del controlador externo Arduino. El controlador recibe la señal desde el pin A0 y automáticamente la transforma en un número dentro del intervalo 0-1023. Con ello se puede establecer referencias de iluminación.

### *3.1.1.2. Respuesta del Sistema:*

La segunda parte del sistema de iluminación consta de encender una luz para iluminar las plantas en caso de que la medición de la primera parte indicase una falta de iluminación en el ambiente. Inicialmente se planteó usar luces LED de espectro completo para poder suministrar el tipo de luz adecuado para cada tipo de planta, pero esta idea se tuvo que descartar por motivos de presupuesto (se valora para futuras implementaciones o versiones del proyecto). Por lo tanto, con el objetivo de aprovechar la implementación del Sistema de Iluminación, se ha decidido utilizar un diodo LED para que represente de manera simbólica la luz LED de espectro completo a implementar en futuras implementaciones. Para ello se monta el siguiente circuito.

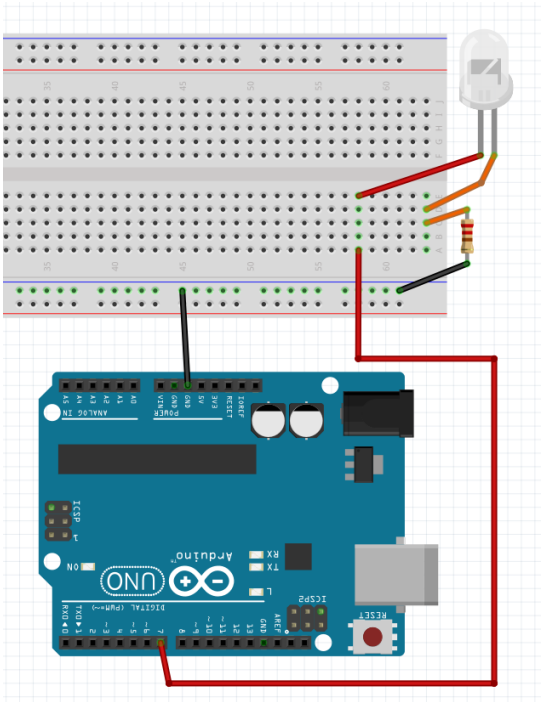

*Figura 3.1.1.2: Circuito de la segunda parte del Sistema de Iluminación*

Como se puede observar en la figura anterior, se conecta un diodo LED en serie con otra resistencia de 220 ohmios y se alimenta desde los 5V, D7 y GND del controlador externo Arduino. Como se puede observar, en este sistema se realiza tanto la medición como la respuesta acorde a las mediciones a través del controlador extremo.

### 3.1.2. Subsistema de Riego

El Subsistema de Riego es similar al Subsistema de Iluminación a nivel conceptual. También utiliza unos sensores en los controladores extremo para realizar mediciones que luego estos controladores enviaran al controlador central que, en función de estas mediciones, tomará decisiones de actuación. La principal diferencia entre este subsistema y el de Iluminación es que los componentes actuadores y su interacción no se encuentran gestionados por los controladores extremo sino por el controlador central.

#### *3.1.2.1. Medición:*

La primera parte del sistema de Riego consta de medir la humedad de la tierra de macetas a través de sensores de humedad capacitivos. Para ello se monta el siguiente circuito.

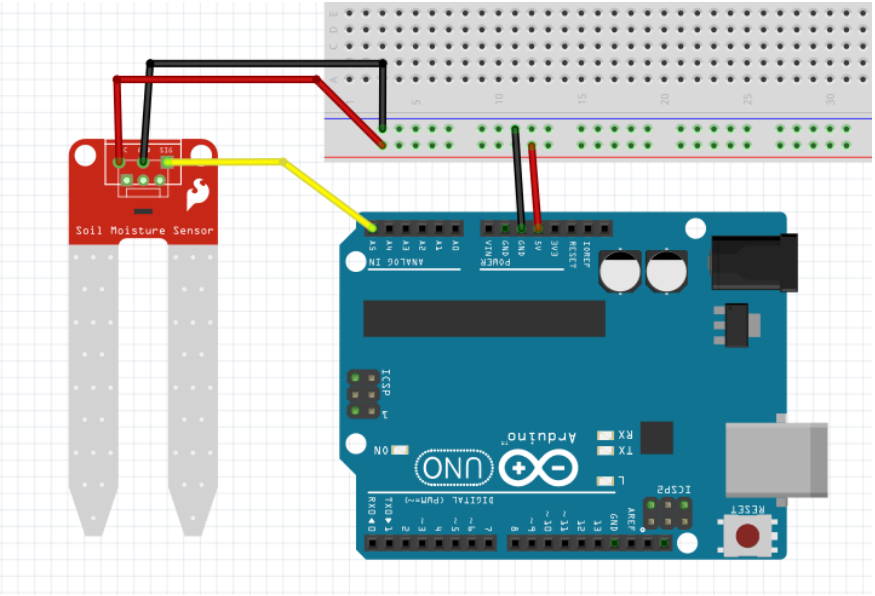


*Figura 3.1.2.1: Circuito de la primera parte del Sistema de Riego*

Como se puede observar en la figura anterior, el circuito es más simple que el circuito de la resistencia LDR, puesto que todo el microcircuito ya viene integrado en los sensores de humedad capacitivos. Cada sensor consta de 3 cables (alimentación, datos y tierra) que se conectaran a los pines 5V, A5 y GND del controlador externo Arduino. El controlador recibe la señal desde el pin A5 y automáticamente la transforma en un número dentro del intervalo 0-1023. Con ello se puede establecer referencias de humedad.

### *3.1.2.2. Respuesta del Sistema:*

La segunda parte del sistema de Riego consta de encender una bomba de agua de achique que se encuentra sumergida en un tanque para regar las plantas en caso de que la medición de la primera parte indicase una falta de humedad en la tierra de la maceta. Es en este punto donde el sistema de Riego se diferencia del sistema de Iluminación, puesto que esta parte del sistema no se encuentra en los brazos externos del prototipo, sino gestionada por el controlador central.

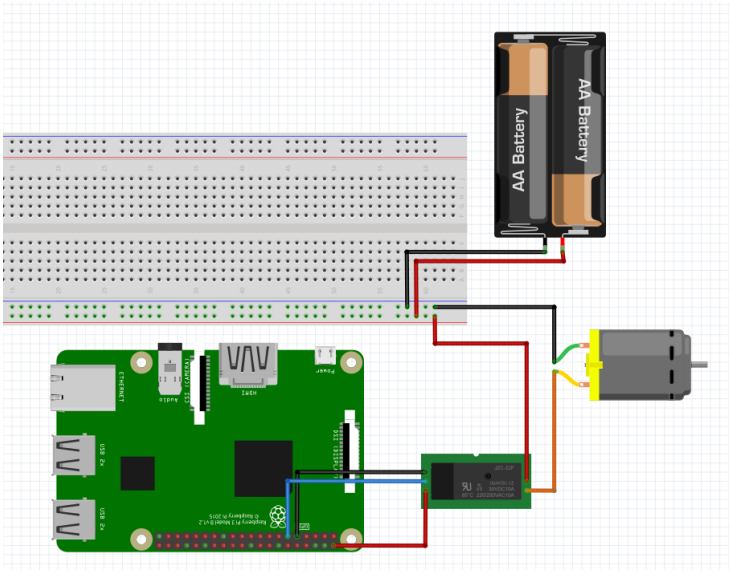


*Figura 3.1.2.2: Circuito de la segunda parte del Sistema de Riego*

Como se puede observar en la figura anterior, se conecta un relé por bomba de agua al controlador central (en nuestro proyecto, los relés están conectados a los pines 11 (GPIO 17), 9 (GND) y 2 (5V) de la Raspberry Pi). Estos relés forman un circuito con una fuente de alimentación externa que es la que proporcionará el voltaje para alimentar las bombas de agua.

### 3.1.3. Subsistema de Comunicaciones

El Subsistema de Comunicaciones tiene como objetivo gestionar la comunicación entre los controladores del proyecto. Inicialmente, esta comunicación iba a ser una comunicación inalámbrica, ya que se había planteado inicialmente implementar un protocolo de comunicación por radiofrecuencia. Este protocolo era el RF24 y para poder implementarlo se adquirieron los módulos inalámbricos NRF24L01. Se intentó implementar el protocolo y se consiguieron resultados positivos en las comunicaciones bidireccionales entre los controladores extremo, pero ocurrieron dificultades causadas por conflictos entre librerías que imposibilitaron la comunicación con el controlador central. Por motivos de tiempos y presupuesto, se decidió abandonar la implementación de este protocolo y buscar alternativas mediante el uso de cables, a expensas de la practicidad del proyecto. Se plantea de cara al futuro la posibilidad de implementar comunicaciones inalámbricas con tecnología Wifi.

Actualmente el proyecto cuenta con una comunicación Serial mediante el uso de USB para comunicar y alimentar a los controladores extremo a través del controlador central.

#### *3.1.3.1. Mensajes:*

A continuación, listaremos los mensajes que tienen lugar en las comunicaciones entre los controladores del proyecto.

#### 3.1.3.1.1. Mensajes de Extremo a Central:

1. **Medición sobre Luz Ambiental** – Incluye un valor numérico entre 0 y 1023 que indica la cantidad de luz que recibe la resistencia LDR. También incluye una letra que sirve como identificador del sensor que está midiendo la luz.
2. **Medición sobre Humedad** – Incluye un valor numérico entre 0 y 1023 que indica la cantidad de humedad que detecta el sensor de humedad capacitivo. También incluye una letra que sirve como identificador del sensor de humedad que está realizando la medición.

#### 3.1.3.1.2. Mensajes de Central a Extremo:

1. **Activar diodo LED** – Contiene un código de palabras interpretable para los controladores extremo (“DARK”) y un identificador para poder distinguir el controlador y el LED a encender.
2. **Desactivar diodo LED** – Contiene un código de palabras interpretable para los controladores extremo (“LIGHT”) y un identificador para poder distinguir el controlador y el LED a apagar.

### 3.1.4. Subsistema de Señalización

El proyecto cuenta con un Subsistema de Señalización basado en la proyección de información sobre las acciones que se llevan a cabo en el proyecto a través de una pantalla LCD. Esta pantalla muestra mensajes cortos y concretos sobre las acciones que se llevan a cabo en tiempo real con el objetivo de informar a los usuarios de lo que está ocurriendo en cada instante. Para ello se monta el siguiente circuito.

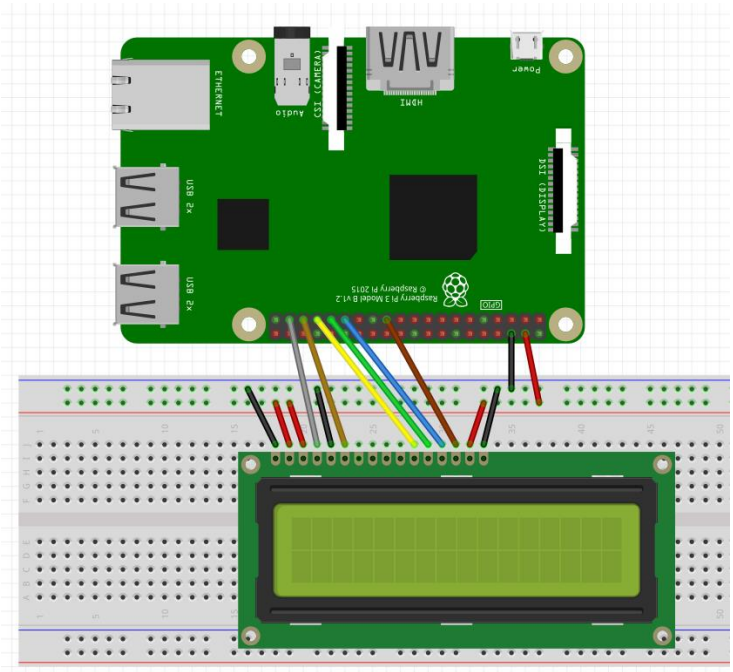

*Figura 3.1.4.1: Circuito de la primera parte del Sistema de Riego*

Como se puede observar en la figura anterior, el circuito se basa en una conexión donde se conectan los pines del LCD a la Raspberry Pi mediante la siguiente configuración:

1. Los pines 1, 5 y 16 de la LCD a GND.
2. Los pines 2, 3 y 15 de la LCD a Vcc (5V).
3. El pin 4 de la LCD al pin 37 (GPIO 26) de la Raspberry Pi.
4. El pin 6 de la LCD al pin 35 (GPIO 19) de la Raspberry Pi.
5. El pin 11 de la LCD al pin 33 (GPIO 13) de la Raspberry Pi.
6. El pin 12 de la LCD al pin 31 (GPIO 6) de la Raspberry Pi.
7. El pin 13 de la LCD al pin 29 (GPIO 5) de la Raspberry Pi.
8. El pin 14 de la LCD al pin 23 (GPIO 11) de la Raspberry Pi.

#### *3.1.4.1. Mensajes:*

A continuación, listaremos los mensajes que se proyectan en la pantalla LCD.

1. **Turning LED X on** – Se muestra cuando el controlador central ordena a un controlador extremo que encienda el diodo LED X (donde X representa el identificador del diodo LED en cuestión).
2. **Turning LED X off** – Se muestra cuando el controlador central ordena a un controlador extremo que apague el diodo LED X (donde X representa el identificador del diodo LED en cuestión).
3. **Turning PUMP X on** – Se muestra cuando el controlador central activa la Bomba de Agua X (donde X representa el identificador de la bomba de agua en cuestión).
4. **Turning PUMP X off** – Se muestra cuando el controlador central desactiva la Bomba de Agua X (donde X representa el identificador de la bomba de agua en cuestión).

## 3.2. DESCRIPCIÓN DEL SISTEMA

En esta sección se realizará una descripción del Sistema de Riego automático, que incluirá diagramas conceptuales sobre cómo ha sido ideado el sistema.

### 3.2.1. Descripción General

El proyecto se divide principalmente en dos partes, que serían los brazos extremos y el núcleo.

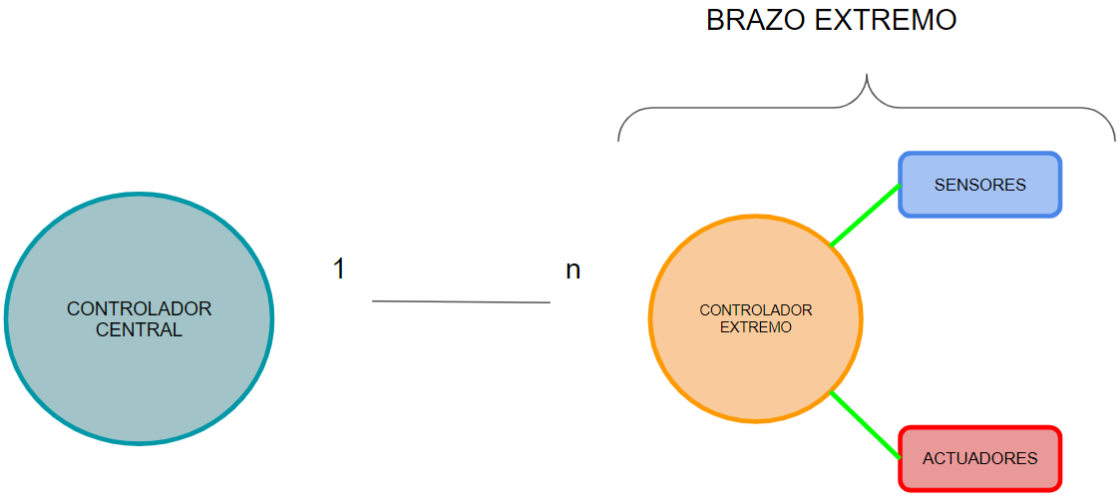


*Figura 3.2.1.1: Diagrama conceptual del sistema*

En la figura anterior se puede observar cómo interactúan las partes diferenciadas. También se puede apreciar la facilidad que tiene este proyecto en cuanto a escalabilidad puesto que se permite añadir múltiples brazos a cada núcleo, y múltiples sensores por cada brazo como se verá posteriormente.

### 3.2.2. Brazo extremo

El proyecto ha sido diseñado para poder implementar múltiples brazos. Cada brazo consta de un controlador extremo, un Arduino UNO R3, y otra serie de componentes. Estos componentes serían los que formarían algunos de los circuitos descritos en la Sección 3.2.

A continuación, listaremos los componentes utilizados en cada uno de los sistemas.

#### *3.2.2.1. Subsistema de Iluminación*

1. Resistencia fotosensibles o LDR
2. Resistencia de 10.000 ohmios
3. Resistencia de 220 ohmios
4. Diodo LED
5. Jumpers o cables de pines

#### *3.2.2.2. Subsistema de Riego*

1. Sensor de humedad capacitivo
2. Jumpers o cables de pines

Cada brazo extremo puede incluir hasta un máximo de cinco sensores (incluyendo ambos sensores de humedad y resistencias LDR) por cómo están distribuidos los pines del controlador extremo.

### 3.2.3. Núcleo

El proyecto consta de un único núcleo formado por un controlador denominado controlador central, que en nuestra implementación ha resultado ser una Raspberry Pi 3 B, y una serie de componentes. Estos componentes serían los que formarían el resto de los circuitos descritos en la Sección 3.2.

A continuación, listaremos los componentes utilizados en cada uno de los sistemas.

#### *3.2.3.1. Subsistema de Riego*

1. Relé
2. Bomba de Agua Eléctrica
3. Jumpers o cables de pines

#### *3.2.3.2. Subsistema de Señalización*

1. Pantalla LCD
2. Jumpers o cables de pines

La comunicación entre los controladores extremo y el controlador central se hace usando cables USB Tipo A – Tipo B. Las conexiones entre los componentes y los controladores se ha mediante cables jumpers o cables de pines y breadboards o protoboards.

## 3.3. SELECCIÓN DE COMPONENTES

En esta sección se listarán los componentes utilizados para la realización del prototipo. Se adjuntan los enlaces de compra de cada componente en el Apéndice A. También se adjunta al proyecto el coste y consumo de los componentes en los consiguientes apéndices: Apéndice B y Apéndice C.

1. Raspberry Pi 3 B
2. Arduino UNO R3
3. Elegoo UNO R3
4. Resistencia LDR
5. Resistencia de 10.000 ohmios
6. Resistencia de 220 ohmios
7. Diodo LED
8. Sensor de humedad capacitivo
9. Relé
10. Bomba de Agua Eléctrica 3-5 V
11. Pantalla LCD
12. Cables USB Tipo A – Tipo B
13. Jumpers o cables de pines
14. Fuente de Alimentación de 3 V
15. Breadboards

## 3.4. CODIFICACIÓN DEL PROYECTO

Se adjunta el código del proyecto en los apéndices del documento. Más en concreto, en el Apéndice D pueden encontrar el código del controlador central, mientras que en el Apéndice E pueden encontrar el código del controlador extremo.

También pueden encontrar el código del proyecto completo en el apartado número 14 de las referencias del proyecto.

# 4. INTEGRACIÓN, PRUEBAS Y RESULTADOS

En esta sección se resumirá tanto cómo se ha implementado nuestro prototipo a presentar, como las pruebas que se han realizado para comprobar el correcto funcionamiento del prototipo y los resultados obtenidos de estas pruebas.

## 4.1. INTEGRACIÓN

En el apartado de Integración se realizará una descripción del prototipo y de cómo ha sido implementado. Esta descripción es la del prototipo que se ha presentado junto con esta memoria.

El prototipo se ha desarrollado con un total de dos brazos extremo. Esto quiere decir que el prototipo consta de dos controladores extremo y uno central.

Cada brazo extremo implementa un total de dos sensores de humedad y una resistencia LDR en sus correspondientes circuitos, y un circuito con el diodo LED, todo ello montado en un breadboard por brazo extremo.

El núcleo implementa el subsistema de señalización con su correspondiente pantalla en un propio breadboard, y los actuadores del sistema de riego, utilizando un total de cuatro relés y cuatro bombas de agua eléctricas, que son alimentadas por una fuente de alimentación externa de tres voltios.

Como se ha comentado anteriormente, la comunicación entre los controladores se realiza mediante conexión Serial con un cable USB Tipo A – Tipo B por cada controlador extremo con el controlador central, y la conexión entre los distintos componentes se realiza mediante jumpers o cables de pines.

## 4.2. PRUEBAS

En esta sección se repasarán las pruebas a las que se le ha ido sometiendo al prototipo cuya finalidad era la de comprobar el correcto funcionamiento del mismo e intentar encontrar errores y fallos.

### 4.2.1. Pruebas de medición de luz

Estas pruebas han consistido en realizar mediciones utilizando la resistencia fotosensible. El objetivo de estas pruebas era el de comprobar que la primera parte del Subsistema de Iluminación funcionase correctamente y también para comprobar los niveles de luz y poder referenciar cuándo se debería encender y apagar la luz (el diodo LED en nuestro prototipo).

Para ello, se han realizado mediciones mientras se iba variando la iluminación ambiental de manera artificial, bloqueando y desbloqueando la luz recibida en el sensor LDR, todo ello mientras se monitorizaban los valores medidos.

### 4.2.2. Pruebas de medición de humedad

Estas pruebas han consistido en realizar mediciones utilizando los sensores de humedad capacitivos. El objetivo de estas pruebas era el de comprobar que la primera parte del Subsistema de Riego

funcionase correctamente y también para comprobar los niveles de humedad en la tierra de las macetas y poder referenciar cuándo se debería encender y apagar las bombas de agua eléctricas.

Para ello, se han realizado mediciones mientras se iba variando la humedad recibida de manera artificial, añadiendo o retirando agua de los sensores de humedad capacitivos, todo ello mientras se monitorizaban los valores medidos.

### 4.2.3. Pruebas de encendido y apagado de luz

Estas pruebas han consistido en comprobar que la segunda parte del Subsistema de Iluminación funcionase correctamente encendiendo y apagando la luz (el diodo LED en nuestro prototipo).

Este tipo de pruebas se han realizado primero ordenando de manera directa apagar y encender las luces para luego probarlas con instrucciones en función de los valores medidos por el Subsistema de Iluminación.

### 4.2.4. Pruebas de encendido y apagado de riego

Estas pruebas han consistido en comprobar que la segunda parte del Subsistema de Riego funcionase correctamente encendiendo y apagando las bombas de agua eléctricas.

Este tipo de pruebas se han realizado primero ordenando de manera directa apagar y encender las bombas de agua eléctricas para luego probarlas con instrucciones en función de los valores medidos por el Subsistema de Riego.

### 4.2.5. Pruebas de señalización de información

Estas pruebas han consistido en comprobar que el Subsistema de Señalización funcionase correctamente enviando información que mostrar a la pantalla LCD, y también probar que la pantalla muestre cada acción que se realiza en el sistema.

Este tipo de pruebas se han realizado primero enviando directamente caracteres a mostrar por la pantalla LCD para luego probarlas con acciones que debiera mostrar la pantalla LCD en función de los demás Subsistemas.

### 4.2.6. Pruebas de comunicación

Estas pruebas han consistido en comprobar que el Subsistema de Comunicaciones funcionase correctamente enviando información de manera bidireccional entre los controladores extremo y el controlador central. También se ha probado que se envíen correctamente los mensajes en el momento indicado.

## 4.3. RESULTADOS

En esta sección se comenta de manera general los resultados obtenidos en las diversas pruebas comentadas en la Sección 4.2.

### 4.3.1. Resultados obtenidos tras las pruebas de medición de luz

Los resultados obtenidos en estas pruebas han sido favorables. Hemos comprobado que las resistencias LDR y los circuitos que componen la primera parte del Subsistema de Iluminación funcionan de manera correcta.

También hemos podido concluir un valor medio para poder determinar si hay suficiente luz ambiental o no en función del valor numérico obtenido durante las pruebas de mediciones. Ese valor está definido entre 500 y 800, y para el proyecto se ha designado 700 como umbral para decidir si se decide encender o no la luz. Si el valor de las mediciones se encuentra por debajo de 700 entonces se decidirá encender la luz (se considera que no hay suficiente iluminación ambiental)., mientras que en el caso de que supere o iguale el valor 700 entonces se decidirá apagar la luz (se considera que hay suficiente iluminación ambiental).

### 4.3.2. Resultados obtenidos tras las pruebas de medición de humedad

Los resultados obtenidos en estas pruebas han sido favorables. Hemos comprobado que los sensores de humedad capacitivos y los circuitos que componen la primera parte del Subsistema de Riego funcionan de manera correcta.

También hemos podido concluir un valor medio para poder determinar si hay suficiente humedad en la tierra de las macetas o no en función del valor numérico obtenido durante las pruebas de mediciones. Ese valor está definido entre 200 y 500, y para el proyecto se ha designado 400 como umbral para decidir si se decide encender o no las bombas de agua eléctricas. Si el valor de las mediciones se encuentra por debajo de 400 entonces se decidirá encender la bomba de agua (se considera que no hay suficiente humedad)., mientras que en el caso de que supere o iguale el valor 400 entonces se decidirá apagar la bomba de agua (se considera que hay suficiente humedad).

### 4.3.3. Resultados obtenidos tras las pruebas de encendido y apagado de luz

Los resultados obtenidos en estas pruebas han sido favorables. Hemos comprobado que las luces (en nuestro caso un diodo LED) y los circuitos que componen la segunda parte del Subsistema de Iluminación funcionan de manera correcta.

### 4.3.4. Resultados obtenidos tras las pruebas de encendido y apagado de riego

Los resultados obtenidos en estas pruebas han sido favorables. Hemos comprobado que las bombas de agua eléctricas y los circuitos que componen la segunda parte del Subsistema de Riego funcionan de manera correcta.

Además, hemos comprobado que el funcionamiento y achique de las bombas funciona de manera correcta.

### 4.3.5. Resultados obtenidos tras las pruebas de señalización de información

Los resultados obtenidos en estas pruebas han sido favorables. Hemos comprobado que la pantalla LCD y los circuitos que componen el Subsistema de Señalización funcionan de manera correcta.

### 4.3.6. Resultados obtenidos tras las pruebas de comunicación

Los resultados obtenidos en estas pruebas han sido favorables. Hemos comprobado que las comunicaciones y el envío y recepción de mensajes entre los controladores extremo y el controlador central funcionan de manera correcta.

# 5. CONCLUSIONES Y TRABAJO FUTURO

En esta sección se hará un comentario sobre las conclusiones obtenidas a partir del desarrollo de este proyecto y también se puntualizarán posibles aplicaciones futuras y evoluciones del prototipo desarrollado en este proyecto.

## 5.1. CONCLUSIONES

En primer lugar, podemos concluir que nuestro proyecto ha sido un éxito en cuanto al diseño y desarrollo de un sistema de autocultivo que riega las plantas de manera autónoma y eficiente utilizando un diseño que permite la escalabilidad de manera fácil y sencilla. Hemos logrado implementar un sistema distribuido compuesto por un controlador central y varios controladores extremos, cada uno equipado con sensores de humedad y luz ambiental. Creemos que nuestro prototipo es el primer paso para crear un producto que podría ser adecuado para su implementación tanto en pequeños hogares como huertos urbanos e incluso en grandes granjas de cultivo.

Sin embargo, también podemos concluir que existen limitaciones en nuestro prototipo actual que podrían afectar su escalabilidad. Una de las limitaciones más importantes es la conexión de los sensores (tanto de humedad como de luz) con el controlador extremo, que se realiza mediante pines y cuyo número es limitado. Además, el consumo de estos sensores también puede afectar directamente al controlador extremo. Además, la conexión mediante cables también limita el área de actuación del sistema y conlleva ciertos riesgos al manipular cables con agua cerca. Estas limitaciones podrían abordarse si logramos implementar sensores que se comuniquen de manera inalámbrica utilizando, por ejemplo, tecnología Bluetooth.

Otra limitación importante es la comunicación mediante cable entre los controladores extremos y el controlador central, que podría limitar la cantidad de controladores extremos que podrían conectarse al sistema, y también afectaría al área de actuación y consumo. Esto podría solucionarse implementando una comunicación inalámbrica entre los controladores extremo y el controlador central, e implementar una fuente de alimentación alternativa para los controladores extremo.

En conclusión, nuestro proyecto es un excelente punto de partida para el desarrollo de sistemas de autocultivo escalables y eficientes. Si bien existen limitaciones en el prototipo actual, creemos que estas pueden ser abordadas mediante el uso de tecnologías inalámbricas y sensores más profesionales, lo que permitiría aumentar la superficie de cultivo cubierta por el sistema y reducir los riesgos y costes existentes. En resumen, estamos muy satisfechos con los resultados obtenidos en este proyecto y creemos que nuestro trabajo puede tener aplicaciones muy interesantes en el campo de la agricultura y la alimentación sostenible.

## 5.2. TRABAJO FUTURO

En la sección de trabajo futuro, se valorarán las posibles evoluciones del prototipo actual para poder mejorarlo y adaptarlo a las necesidades del usuario. Una de las posibles evoluciones es la inclusión de luces led de espectro completo para proporcionar una iluminación más adecuada a las plantas para poder cubrir completamente el proceso de cultivo. Otra posible mejora sería el uso de conexiones Wifi en lugar de las conexiones por cable actuales para facilitar la comunicación entre los controladores

extremo y el controlador central, lo que podría aumentar la escalabilidad del sistema y el área de actuación. Además, se considerará la posibilidad de conectar los sensores de manera inalámbrica al controlador extremo, por ejemplo, mediante Bluetooth, lo que aumentaría la flexibilidad, la movilidad y el área de alcance del sistema.

Otra posible evolución del prototipo sería la integración de un sistema de control de temperatura y humedad en el entorno de cultivo, lo que permitiría controlar de manera más precisa las condiciones de cultivo y obtener mejores resultados en la producción de alimentos. También se valorará la posibilidad de incluir nuevos sensores, como sensores de pH o de nutrientes, para monitorizar el estado de las plantas y ajustar el riego y la fertilización de manera más precisa. En resumen, existen muchas posibilidades de mejora y evolución del prototipo actual, lo que permitiría adaptar el sistema de autocultivo a las necesidades específicas de cada usuario y mejorar la producción de alimentos tanto en entornos domésticos como en entornos profesionales.

# BIBLIOGRAFÍA


Blum, J. (2020). *Exploring Arduino – Tools and Techniques for Engineering Wizardry Second Edition.* John Wiley & Sons Inc.

Kölling, M. (2020). *Raspberry PI: A complete guide to start learning RaspberryPi on your own.* Independent.

Lutz, L., & Ray, R. (2018). *Aprender a Programar : Raspberry PI 3 y Python.* CreateSpace.

Moreno Muñoz, A., & Córcoles Córcoles, S. (2017). *Aprende Arduino en un fin de semana.* Software Developer Coach.

Torrente Artero, Ó. (2016). *El mundo GENUINO-ARDUINO. Curso práctico de formación.* RC Libros.

# REFERENCIAS


[1] Raspberry Pi Tutorial 34 - Wireless Pi to Arduino Communication with NRF24L01+ Part 1

https://www.youtube.com/watch?v=_68f-yp63ds

[2] How nRF24L01+ Wireless Module Works & Interface with Arduino

https://lastminuteengineers.com/nrf24l01-arduino-wireless-communication/

[3] How to make Automatic Plant Watering System! | Arduino Project

https://www.youtube.com/watch?v=-jI3SebMXog

[4] Conectar Raspberry Pi 3 a nRF24L01

https://www.laboratoriogluon.com/conectar-raspberry-pi-3-a-nrf24l01/

[5] COMMUNICATION BETWEEN ARDUINO AND RASPBERRY PI USING NRF24L01

https://medium.com/@anujdev11/communication-between-arduino-and-raspberry-pi-using-nrf24l01-818687f7f363

[6] Serial communication between Raspberry Pi and Arduino

https://www.aranacorp.com/en/serial-communication-between-raspberry-pi-and-arduino/

[7] BOMBA de AGUA con ARDUINO. Muy ÚTIL y SENCILLO!!!

https://www.youtube.com/watch?v=4KrkhyyxKEY

[8] How to Add an LCD Touchscreen to Your Raspberry Pi

https://maker.pro/raspberry-pi/tutorial/how-to-add-an-lcd-touchscreen-to-raspberry-pi

[9] How to connect LCD display to your Raspberry pi 3 (IoT)

https://medium.com/coinmonks/how-to-connect-lcd-display-to-your-raspberry-pi-3-iot-68d990843e0d

[10] Arduino Plant Watering System

https://www.instructables.com/Arduino-Plant-Watering-System/

[11] Automatic plant watering project using Arduino |Arduino smart Irrigation

https://techatronic.com/automatic-plant-watering-project-using-arduino-arduino-smart-irrigation/

[12] Soil Moisture Sensor with Arduino Uno

https://www.youtube.com/watch?v=EFvbS6XzTVo

[13] Using a 16x2 LCD Display with a Raspberry Pi

https://www.youtube.com/watch?v=cVdSc8VYVBM

[14] Github del proyecto completo

https://github.com/cabannas/UCM-SED

# APÉNDICES

## A. LISTA DE COMPONENTES Y ENLACES DE COMPRA

1. Raspberry Pi 3 B
   https://www.amazon.es/Raspberry-Pi-Modelo-Placa-Color/dp/B07BFH96M3/ref=sr_1_5?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2XAVYQ0B5X535&keywords=raspberry+pi+3b&qid=1682073867&sprefix=raspberry+pi+3b%2Caps%2C150&sr=8-5

2. Arduino UNO R3
   https://store.arduino.cc/collections/boards/products/arduino-uno-rev3

3. Elegoo UNO R3
   https://www.amazon.es/Tarjeta-Microcontrolador-ATmega328P-ATMEGA16U2-Compatible/dp/B01M7ZB2B4/ref=sr_1_1_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=202J44XF5KURP&keywords=Arduino+UNO+R3&qid=1682073909&sprefix=arduino+uno+r3%2Caps%2C140&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1

4. Resistencia LDR
   https://www.amazon.es/Photoresistor-fotoresistor-Sensor-Resistor-Light-Dependent/dp/B07CZJKBPK/ref=sr_1_6?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2B8YZSL02U3SW&keywords=ldr&qid=1682074020&sprefix=ld%2Caps%2C238&sr=8-6

5. Resistencia de 10.000 ohmios
   https://www.amazon.es/BOJACK-terminales-Potenci%C3%B3metros-Resistencia-variables/dp/B07ZJL91YP/ref=sr_1_3_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2SQXGOP9ARJWI&keywords=resistencia+10k&qid=1682074071&sprefix=resistencia+10k%2Caps%2C113&sr=8-3-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1

6. Resistencia de 220 ohmios
   https://www.amazon.es/Resistencias-tolerancia-Limitaci%C3%B3n-corriente-certificado/dp/B08QRXLKZQ/ref=sr_1_1_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=YOC9XMKFUCKN&keywords=resistencia%2B220&qid=1682074098&sprefix=resistencia%2B220%2Caps%2C139&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&th=1

7. Diodo LED
   https://www.amazon.es/Ociodual-Diodos-Amarillo-Arduino-Electronica/dp/B077SDNZHT/ref=sr_1_5?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=1V6C6IPKSVRGQ&keywords=diodo+led&qid=1682074117&sprefix=diodo+le%2Caps%2C122&sr=8-5

8. Sensor de humedad capacitivo
   https://www.amazon.es/AZDelivery-Humedad-higrometro-Capacitivo-Arduino/dp/B07HJ6N1S4/ref=sr_1_1_sspa?keywords=sensor%2Bhumedad%2Barduino&qid=1682074141&sprefix=sensor%2Bhumedad%2B%2Caps%2C120&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&smid=A1X7QLRQH87QA3&th=1

9. Relé
   https://www.amazon.es/Yizhet-acoplador-adecuado-Arduino-canales/dp/B07GXC4FGP/ref=sr_1_1_sspa?keywords=rel%C3%A9%2Barduino&qid=1682074181&sprefix=rel%C3%A9%2Bard%2Caps%2C227&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&th=1

10. Bomba de Agua Eléctrica 3-5 V
    https://www.amazon.es/RUNCCI-YUN-Sumergible-Cepillo-Tuber%C3%ADa-paraTanque/dp/B082PM8L6X/ref=sr_1_5?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=3SUC8GUTS6CSL&keywords=bomba%2Bagua%2B3v&qid=1682074205&sprefix=bomba%2Bagua%2B3v%2Caps%2C191&sr=8-5&th=1

11. Pantalla LCD
    https://www.amazon.es/AZDelivery-HD44780-visualizaci%C3%B3n-16-caracteres-Arduino/dp/B079T264ZZ/ref=sr_1_1_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2K0D37PKGVT7M&keywords=lcd%2Barduino&qid=1682074224&sprefix=lcd%2Barduino%2Caps%2C112&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&smid=A1X7QLRQH87QA3&th=1

12. Cables USB Tipo A – Tipo B
    https://www.amazon.es/CABLEPELADO-Cable-para-Impresora-Negro/dp/B08HBW8DGF/ref=sr_1_2_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=ST94LUTEQEWS&keywords=cable%2Busb%2Btipo%2Ba%2Btipo%2Bb&qid=1682074253&sprefix=cable%2Busb%2Btipo%2Ba%2Btipo%2Bb%2Caps%2C113&sr=8-2-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&smid=A9DEKVHK1XMRT&th=1

13. Jumpers o cables de pines
    https://www.amazon.es/YXPCARS-cables-hembra-femenino-Arduino/dp/B08HQ7K6M7/ref=sr_1_2_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=1LDRMZZA7Y29B&keywords=jumper+arduino&qid=1682074272&sprefix=jumper+arduino%2Caps%2C108&sr=8-2-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1

14. Fuente de Alimentación de 3 V
    https://www.amazon.es/SoulBay-alimentaci%C3%B3n-universal-cargador-adaptadores/dp/B07DG16TKS/ref=sr_1_1_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=15R078VZWSIYR&keywords=fuente+alimentaci%C3%B3n+3+-+12v&qid=1682074327&sprefix=fuente+alimentaci%C3%B3n+3+-+12v%2Caps%2C142&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1

15. Breadboards

https://www.amazon.es/ELEGOO-Breadboard-Prototipo-Soldaduras-Distribuci%C3%B3n/dp/B071ZGC75Y/ref=sr_1_2_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=1KN8K0S4JT2J1&keywords=breadboard&qid=1682074343&sprefix=breadboard%2Caps%2C120&sr=8-2-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&th=1

## B. COSTES DE LOS COMPONENTES DEL PROTOTIPO

1. Raspberry Pi 3 B – 120 €
2. Arduino UNO R3 – 30 €
3. Elegoo UNO R3 – 20 €
4. Resistencia LDR – 3 €
5. Resistencia de 10.000 ohmios – 3 €
6. Resistencia de 220 ohmios – 3 €
7. Diodo LED – 3 €
8. Sensor de humedad capacitivo – 8 €
9. Relé – 6 €
10. Bomba de Agua Eléctrica 3-5 V – 5 €
11. Pantalla LCD – 8 €
12. Cables USB Tipo A – Tipo B – 1 €
13. Jumpers o cables de pines – 10 €
14. Fuente de Alimentación de 3 V – 20 €
15. Breadboards – 10 €

Coste estimado del proyecto: 250 €

## C. CONSUMO DE LOS COMPONENTES DEL PROTOTIPO

1. Raspberry Pi 3 B – 230 / 250 mA; 1.2 / 1.8 W
2. Arduino UNO R3 – 46 / 200 mA; 0.23 / 1.02 W
3. Elegoo UNO R3 – 46 / 200 mA; 0.23 / 1.02 W
4. Resistencia LDR – 2 mA; 0.01 W
5. Diodo LED – 20 mA; 0.1 W
6. Sensor de humedad capacitivo – 5 mA; 0.025 W
7. Bomba de Agua Eléctrica 3-5 V – 50 mA; 0.25 W

Consumo estimado del proyecto: ~ 3.5 / 4 W

## D. CÓDIGO DEL CONTROLADOR CENTRAL RASPBERRY PI

```
import serial
import RPi.GPIO as GPIO
import time



# Define GPIO to LCD mapping
LCD_RS = 26
LCD_E  = 19
LCD_D4 = 13
LCD_D5 = 6
LCD_D6 = 5
LCD_D7 = 11
LED_ON = 15

RELAY_IN1 = 17
RELAY_IN2 = 27
RELAY_IN3 = 22
RELAY_IN4 = 23

# Define some device constants
LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line

# Timing constants
E_PULSE = 0.00005
E_DELAY = 0.00005




def main():

    # variables to control state changes
    lcdLdrA = False
    lcdLdrB = False
    lcdHumA = False
    lcdHumB = False
    lcdHumC = False
    lcdHumD = False
    ldrA = 0
    ldrB = 0
```

```
    lineA = ""
    lineB = ""
    clean_lcd_count = 0


    try:

        # Initialise display
        lcd_init()
        relay_init()

        serA = serial.Serial('/dev/ttyACM1', 9600, timeout=1)
        serB = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
        serA.flush()
        serB.flush()



        while True:

            # ARDUINO A ================================================
            if serA.in_waiting > 0:

                lineA = serA.readline().decode('utf-8').rstrip()
                print("line A = " + lineA)

                if lineA[:3] == "ldr":
                    ldrA, lcdLdrA, clean_lcd_count = ldrAbehaviour(lineA,
lcdLdrA, clean_lcd_count)

                    # Communications
                    if ldrA == 0:
                        serA.write(b"dark\n")
                    elif ldrA == 1:
                        serA.write(b"light\n")

                elif lineA[:4] == "huma":
                    lcdHumA, clean_lcd_count = humAbehaviour(lineA, lcdHumA,
clean_lcd_count)
                elif lineA[:4] == "humb":
                    lcdHumB, clean_lcd_count = humBbehaviour(lineA, lcdHumB,
clean_lcd_count)

            # ARDUINO B ================================================
            if serB.in_waiting > 0:

                lineB = serB.readline().decode('utf-8').rstrip()
                print("line B = " + lineB)
```

```
                if lineB[:3] == "ldr":
                    ldrB, lcdLdrB, clean_lcd_count = ldrBbehaviour(lineB,
lcdLdrB, clean_lcd_count)

                    # Communications
                    if ldrB == 0:
                        serB.write(b"dark\n")
                    elif ldrB == 1:
                        serB.write(b"light\n")

                elif lineB[:4] == "huma":
                    lcdHumC,clean_lcd_count = humCbehaviour(lineB, lcdHumC,
clean_lcd_count)
                elif lineB[:4] == "humb":
                    lcdHumD,clean_lcd_count = humDbehaviour(lineB, lcdHumD,
clean_lcd_count)


            # Controls duration of LCD ====================================
            clean_lcd_count += 1

            if clean_lcd_count >= 200000:

                lcd_byte(LCD_LINE_1, LCD_CMD)
                lcd_string("",2)
                lcd_byte(LCD_LINE_2, LCD_CMD)
                lcd_string("",2)

                clean_lcd_count = 0

    finally: # If interrupted, close GPIO and turn off LCD

        lcd_byte(LCD_LINE_1, LCD_CMD)
        lcd_string("",2)
        lcd_byte(LCD_LINE_2, LCD_CMD)
        lcd_string("",2)
        GPIO.cleanup()


def ldrAbehaviour(lineA, lcdLdrA, clean_lcd_count):

    stringA = lineA[3:]

    if float(stringA) >= 500:

        if lcdLdrA == False:

            lcdLdrA = True
```

```
            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning on",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("LED A",2)

        return 0, lcdLdrA, clean_lcd_count;

    else:

        if lcdLdrA:

            lcdLdrA = False

            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning off",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("LED A",2)

        return 1, lcdLdrA, clean_lcd_count;


def ldrBbehaviour(lineB, lcdLdrB, clean_lcd_count):

    stringB = lineB[3:]

    if float(stringB) >= 500:

        if lcdLdrB == False:

            lcdLdrB = True
            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning on",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("LED B",2)

        return 0, lcdLdrB, clean_lcd_count;

    else:

        if lcdLdrB:

            lcdLdrB = False
            clean_lcd_count = 0 # reset clean
```

```
            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning off",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("LED B",2)

        return 1, lcdLdrB, clean_lcd_count;

def humAbehaviour(lineA, lcdHumA, clean_lcd_count):

    stringA = lineA[4:]

    if float(stringA) <= 500:

        GPIO.output(RELAY_IN1, True)

        if lcdHumA == False:

            lcdHumA = True
            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning off",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("PUMP A",2)

    else:

        GPIO.output(RELAY_IN1, False)

        if lcdHumA:

            lcdHumA = False
            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning on",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("PUMP A",2)

    return lcdHumA, clean_lcd_count

def humBbehaviour(lineA, lcdHumB, clean_lcd_count):

    stringA = lineA[4:]

    if float(stringA) <= 500:
```

```
        GPIO.output(RELAY_IN2, True)

        if lcdHumB == False:

            lcdHumB = True
            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning off",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("PUMP B",2)

    else:

        GPIO.output(RELAY_IN2, False)

        if lcdHumB:

            lcdHumB = False
            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning on",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("PUMP B",2)

    return lcdHumB, clean_lcd_count

def humCbehaviour(lineB, lcdHumC, clean_lcd_count):

    stringB = lineB[4:]

    if float(stringB) <= 500:

        GPIO.output(RELAY_IN3, True)

        if lcdHumC == False:

            lcdHumC = True
            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning off",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("PUMP C",2)

    else:
```

```
        GPIO.output(RELAY_IN3, False)

        if lcdHumC:

            lcdHumC = False
            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning on",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("PUMP C",2)

    return lcdHumC, clean_lcd_count

def humDbehaviour(lineB, lcdHumD, clean_lcd_count):

    stringB = lineB[4:]

    if float(stringB) <= 500:

        GPIO.output(RELAY_IN4, True)

        if lcdHumD == False:

            lcdHumD = True
            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning off",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("PUMP D",2)

    else:

        GPIO.output(RELAY_IN4, False)

        if lcdHumD:

            lcdHumD = False
            clean_lcd_count = 0 # reset clean

            lcd_byte(LCD_LINE_1, LCD_CMD)
            lcd_string("Turning on",2)
            lcd_byte(LCD_LINE_2, LCD_CMD)
            lcd_string("PUMP D",2)

    return lcdHumD, clean_lcd_count
```

```
def relay_init():
    GPIO.setmode(GPIO.BCM)       # Use BCM GPIO numbers
    GPIO.setup(RELAY_IN1, GPIO.OUT)
    GPIO.setup(RELAY_IN2, GPIO.OUT)
    GPIO.setup(RELAY_IN3, GPIO.OUT)
    GPIO.setup(RELAY_IN4, GPIO.OUT)

def lcd_init():
    GPIO.setmode(GPIO.BCM)       # Use BCM GPIO numbers
    GPIO.setup(LCD_E, GPIO.OUT)  # E
    GPIO.setup(LCD_RS, GPIO.OUT) # RS
    GPIO.setup(LCD_D4, GPIO.OUT) # DB4
    GPIO.setup(LCD_D5, GPIO.OUT) # DB5
    GPIO.setup(LCD_D6, GPIO.OUT) # DB6
    GPIO.setup(LCD_D7, GPIO.OUT) # DB7
    GPIO.setup(LED_ON, GPIO.OUT) # Backlight enable
    # Initialise display
    lcd_byte(0x33,LCD_CMD)
    lcd_byte(0x32,LCD_CMD)
    lcd_byte(0x28,LCD_CMD)
    lcd_byte(0x0C,LCD_CMD)
    lcd_byte(0x06,LCD_CMD)
    lcd_byte(0x01,LCD_CMD)

def lcd_string(message,style):
    # Send string to display
    # style=1 Left justified
    # style=2 Centred
    # style=3 Right justified

    if style==1:
        message = message.ljust(LCD_WIDTH," ")
    elif style==2:
        message = message.center(LCD_WIDTH," ")
    elif style==3:
        message = message.rjust(LCD_WIDTH," ")

    for i in range(LCD_WIDTH):
        lcd_byte(ord(message[i]),LCD_CHR)

def lcd_byte(bits, mode):
    # Send byte to data pins
    # bits = data
    # mode = True  for character
    #        False for command

    GPIO.output(LCD_RS, mode) # RS
```

```
    # High bits
    GPIO.output(LCD_D4, False)
    GPIO.output(LCD_D5, False)
    GPIO.output(LCD_D6, False)
    GPIO.output(LCD_D7, False)
    if bits&0x10==0x10:
        GPIO.output(LCD_D4, True)
    if bits&0x20==0x20:
        GPIO.output(LCD_D5, True)
    if bits&0x40==0x40:
        GPIO.output(LCD_D6, True)
    if bits&0x80==0x80:
        GPIO.output(LCD_D7, True)

    # Toggle 'Enable' pin
    time.sleep(E_DELAY)
    GPIO.output(LCD_E, True)
    time.sleep(E_PULSE)
    GPIO.output(LCD_E, False)
    time.sleep(E_DELAY)

    # Low bits
    GPIO.output(LCD_D4, False)
    GPIO.output(LCD_D5, False)
    GPIO.output(LCD_D6, False)
    GPIO.output(LCD_D7, False)
    if bits&0x01==0x01:
        GPIO.output(LCD_D4, True)
    if bits&0x02==0x02:
        GPIO.output(LCD_D5, True)
    if bits&0x04==0x04:
        GPIO.output(LCD_D6, True)
    if bits&0x08==0x08:
        GPIO.output(LCD_D7, True)

    # Toggle 'Enable' pin
    time.sleep(E_DELAY)
    GPIO.output(LCD_E, True)
    time.sleep(E_PULSE)
    GPIO.output(LCD_E, False)
    time.sleep(E_DELAY)

if __name__ == '__main__':
    main()
```

## E. CÓDIGO DEL CONTROLADOR EXTREMO ARDUINO

```
// Define communication variable
String command;

// Define pins
const int humAPin = A4;
const int humBPin = A5;
const int ledPin = 7;
const int ldrPin = A0;

int humAVal = 0;
int humBVal = 0;
int ldrVal = 0;

void setup() {

  Serial.begin(9600);

  // Initialize pins
  pinMode(humAPin, INPUT);
  pinMode(humBPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(ldrPin, INPUT);

  delay(1000);
}

void loop() {

  //read LDR
  ldrVal = analogRead(ldrPin);

  //read Humidity Sensors
  humAVal = analogRead(humAPin);
  humBVal = analogRead(humBPin);

  // Send info to Raspi
  Serial.print("ldr");  // distinguish
  Serial.println(ldrVal);
  Serial.print("huma");
  Serial.println(humAVal);
  Serial.print("humb");
  Serial.println(humBVal);

  if(Serial.available()){

    // Read from Raspi
```

```
    command = Serial.readStringUntil('\n');
    command.trim(); // eliminates white spaces

    // Actions
    if (command.equals("dark")){

      digitalWrite(ledPin, HIGH);

    }else if (command.equals("light")){

      digitalWrite(ledPin, LOW);

    }else{

      // Default nothing
    }
  }

  delay(1000);
}
```