Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Sistema para la gestión de la iluminación en entornos domésticos

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Millán Díaz, Noelia

Tutor: Posadas Yagüe, Juan Luis
Cotutor: Poza Luján, José Luis

Curso: 2018-2019

Resumen

El presente proyecto nace de la necesidad de crear un sistema autónomo para la iluminación en entornos domésticos, en el cual los componentes del mismo sean capaces de ejecutar su función sin necesidad de intervención humana, además será inteligente porque se comportará adaptándose a la situación con la que se encuentre en cada momento, también escalable porque debe permitir ampliar el sistema de manera sencilla, accesible económicamente para que su implementación en una casa inteligente no suponga un coste destacable y capaz de adaptarse a cualquier tipo de hogar. Mediante el protocolo de comunicación WiFi y el protocolo de control de transmisión (TCP), se comunicarán todos los dispositivos que forman el sistema, esta comunicación hará uso de una pasarela de conexión entre los elementos, con el objetivo de establecer diferentes escenarios de confort lumínico para el usuario, cada escenario se debe adaptar a la necesidad de cada momento y situación. Con todo lo expuesto anteriormente se pretende conseguir una reducción en el consumo eléctrico mediante la correcta iluminación del entorno del hogar. Cada elemento del sistema estará contenido en un cubo diseñado para ser construido utilizando PVC, de manera que se pueda colocar en cualquier parte del hogar, toda la electrónica del sistema estará gestionada por una controladora Arduino y toda la electrónica estará dentro de este cubo.

Palabras clave: Autónomo, accesible, WiFi, TCP, confort lumínico, Arduino.

Abstract

The current project arises from the need to create an autonomous system for lighting in domestic environments, in which the components of the same are able to execute their function without human intervention, it will also be intelligent because it will behave according to the situation in each moment, also scalable because it must be easily adapted, affordable system so it implementation in a smart home does not suppose a remarkable cost and able to adapt to any home. By means of the WiFi communication protocol and the transmission control protocol (TCP), all the devices that be part of the system will be communicated, this communication will make use of a connection gateway between the elements of the system, in order to establish different scenarios of Light comfort for the user, each scenario must be adapted to the needs of each moment and situation. With all of the above, it is intended to achieve a reduction in electricity consumption through the correct lighting of the home environment. Each element of the system will be contained in a cube designed to be made in PVC, so it can be placed anywhere in the home with PVC, all the electronics of the system will be managed by an Arduino controller and all the electronics will be inside this cube.

Keywords: Autonomous, accesible, WiFi, TCP, light comfort, Arduino.

Índice

Índice de tablas

Índice de ilustraciones

1. Introducción

El término domótica puede hacer referencia a cualquier sistema capaz de automatizar una vivienda. El ámbito de este proyecto se va a centrar en la ambientación lumínica, que permite iluminar edificaciones u hogares de manera automática. Actualmente existen diferentes sistemas para lograr este propósito ya que, entre otras cosas, aporta la posibilidad de hacer un consumo más racional de energía:

- Persianas que suben o bajan a la hora programada.
- Luces que se encienden y se apagan a la hora programada.
- Zonas con sensores que detectan proximidad y movimiento.
- Reóstato para crear ambientes adaptados.
- Bombillas *RGB* para crear confort lumínico.

Algunos de los motivos por los que estos dispositivos son cada vez más utilizados en la actualidad son las ventajas que aportan:

- Aumento del confort.
- Creación de situaciones adaptadas al uso.
- Mejora de la eficiencia energética.
- Ahorro.
- Sencillo control a través de *smartphone*, voz…
- Se evita contacto con la corriente eléctrica.

Todos estos sistemas se caracterizan por ser utilizados a través de una aplicación instalada en un smartphone, tablet, o por comandos por voz, además se puede controlar en el propio hogar y desde fuera de él, también se puede dejar programado.

Como toda tecnología emergente, el uso de estos elementos de confort presenta opciones de mejora y una de estas podría ser la autonomía de los sistemas, es decir, convertirlo en un sistema inteligente y liberarse del manejo de dispositivos para su control, establecer comunicaciones entre ellos y que ellos decidan en cada momento o situación cómo actuar para que la eficiencia energética sea óptima.

1.1. Objetivos

El principal objetivo de este proyecto es la construcción de un sistema autónomo con diferentes componentes para que, a través de sensores, puedan captarse datos del entorno, transmitirse entre dichos componentes mediante comunicación inalámbrica *WiFi* y como resultado de su procesamiento pueda decidirse el comportamiento de los dispositivos.

A partir de este primer objetivo, surgen una serie de metas secundarias, pero necesarias para el desarrollo de este proyecto.

El primero de ellos es la elección del *hardware* indicado para poder construir cada componente. Con este objetivo se pretende conseguir que se obtengan valores como la proximidad al componente, la luz que el sensor recibe, movimiento y sonido para decidir cómo ambientar la vivienda.

El segundo objetivo es la comunicación entre los diferentes componentes a través una red *WiFi* para que mediante los datos que se procesen se pueda decidir qué ambiente crear en cada momento, por ejemplo decidir la intensidad lumínica y si deben o no estar encendidas.

El tercer y último objetivo es que el sistema sea accesible económicamente para todo el mundo y que sea fácilmente manejable, además de hacer un uso eficiente de la energía.

1.2. Estructura del documento

El presente documento se divide en cuatro grandes bloques, el primero de ellos es el estudio de algunos de los sistemas inteligentes actuales que pueden tener objetivos similares a este proyecto, como brindar un confort, seguridad y ahorro energético. Después de estudiar sus principales características se han establecido otras diferentes que debe cumplir este proyecto y que lo diferencian del resto de sistemas.

El segundo bloque consiste en establecer los requisitos del proyecto a partir de las características extraídas, se analizarán diferentes soluciones que puedan cumplir los requisitos establecidos, explicando el motivo porque algunas han sido desechadas y por último se describirá la solución elegida, se justificará su elección y se enumerarán diversos casos de uso y se comentarán sus deferentes funcionalidades.

El tercer bloque es el diseño de la solución, en él se va a describir cuales son los elementos físicos de la solución y cual es la relación que hay entre ellos, después se explicará el diseño lógico de la solución mediante un diagrama UML que muestra como todos los objetos de la lógica del algoritmo actúan en conjunto y por último se van a describir las tecnologías utilizadas, tanto los componentes hardware que formarán el sistema como el software empleado para su funcionamiento.

El cuarto y último bloque es la implementación del proyecto, en el que se describirá el montaje del prototipo físico de manera detallada, como se conectará el hardware electrónicamente y el ensamblaje final, se describirá detalladamente también la implementación lógica del algoritmo y protocolos de comunicación que dotarán de funcionalidad a este proyecto.

Una vez se han descrito los cuatro bloques principales que forman parte del desarrollo del proyecto, en el punto 6 se describirán las pruebas que se realizarán para validar el sistema, en el punto 7 se presentarán una serie de conclusiones y en el punto 8 futuros trabajos a desarrollar que guardarán relación con la materia del proyecto.

2. Situación actual

La domótica se compone de sistemas que son capaces de automatizar cualquier tipo de vivienda, pueden estar integrados por medio de redes de comunicaciones interiores y exteriores y además ofrecen servicios para la seguridad, el confort y la comunicación, entre otros, cuyo control puede ser desde dentro o desde fuera del hogar [1].

Dentro de los servicios que ofrece la domótica está el confort lumínico que permite automatizar el sistema de iluminación. Este sistema presenta ventajas como la capacidad de controlar consumo lumínico para facilitar el ahorro, automatizar las luces de la vivienda de manera centralizada desde un dispositivo como podría ser un *smartphone* o una *tablet* y aumentar el confort creando diferentes escenas adaptadas al uso que se haga de la vivienda o a la cantidad de personas que se encuentren en ella.

El confort lumínico de los hogares inicialmente se realizaba mediante lámparas de aceite, se han usado hasta hace 120 años aproximadamente, tenían una mecha que se empapaba en aceite y ardía, presentaban problemas como el gas y el humo que desprendían. También se usaba el alumbrado de gas, una tecnología utilizada para generar luz a partir de combustibles gaseosos como el hidrógeno, metano, propano entre otros [2].

Actualmente se utilizan controladores y sensores (de movimiento, de proximidad, de luz, de voz...) para crear un confort lumínico, para ello se deberá añadir al circuito de iluminación del hogar una instalación domótica que incluya estos dispositivos y una vez instalados actúen según las órdenes que se le haya dado al sistema mediante conectividad *WiFi* o *Bluetooth* [3].

Se están presentando plataformas como *HomeKit*, que permitirán integrar en una única aplicación todos los dispositivos conectados, asistentes de voz como *Cortana* de *Microsoft* o *Alexa* de *Amazon*, o el robot *Jibo* con el que se podrá mantener una conversación fluida, además de reconocer rostros y voz [4] [5].

2.1. Sistemas similares

- Echo Plus

Es un altavoz inteligente que se controla con la voz y usa el *Alexa Voice Service*, permite configurar los dispositivos del Hogar compatibles, escuchar música en *streaming*, consultar la previsión del tiempo, controlar luces o interruptores. Dispone además de un sensor de temperatura, siete micrófonos y cancelación de ruido. Aprende y añade nuevas funciones. Dispone de conectividad *WiFi* de doble banda, no es compatible con redes *WiFi ad hoc*, es compatible con dispositivos que admiten la especificación *Zigbee* como bombillas e interruptores, también es compatible con la conectividad *Bluetooth* [6].

Ilustración 1 – Echo plus Amazon

- Philips Hue White and Color Ambiance

Ilustración 2 - Philips Hue

Se puede controlar con la voz a través de *Alexa*. Permiten elegir entre 16 millones de colores además de controlar con la aplicación *Philips Hue* compatible con *iOS* y *Android* la tonalidad de la luz desde cualquier parte del mundo o configurar alarmas para encender o apagar luces, además es posible sincronizar la iluminación con el TV, la música o videojuegos para conseguir efectos envolventes. Dispone de conectividad mediante cable *Ethernet* y *software* actualizable [7].

- Sistema inteligente con paneles luminosos de Nanoleaf, Edición Rhythm (Apple HomeKit)

Con la app *Casa* (iOS 10) para *iPhone* o *iPad* se puede automatizar las luces para que se adapten a cada situación, elegir entre 16.7 millones de colores, también se puede controlar mediante *Siri*, la asistente de *Apple*, asistente de *Google* y *Alexa Amazon*. Este sistema admite hasta 30 paneles triangulares personalizables por unidad de control y se pueden poner en el orden deseado. Es compatible con *HomeKit* y admite conexiones inalámbricas como *Zigbee* [8].

Ilustración 3 - Apple HomeKit

- Control remoto TRADFRI

Ilustración 4 - Tradfri

Con este sistema se puede modificar el ambiente de la habitación y regular la intensidad o color de la iluminación, el dispositivo de conexión y la app *TRADFRI* permiten crear varios grupos de fuentes de luz (hasta 20 colores) y controlarlos de distintas maneras hasta un máximo de 10 bombillas. Solo es compatible con dispositivos de control inalámbricos de *IKEA* y con un alcance de hasta 10 metros [9].

- iRobot Roomba

Roomba es un robot aspirador que se desplaza por las habitaciones de manera autónoma, se puede configurar para que se mantenga alejado de las áreas y objetos indicados. Contiene sensores *Dirt Detect* para reconocer donde se concentra la suciedad, se programa a través de la app *iRobot HOME* desde cualquier lugar, también es posible presionar el botón clean para empezar la limpieza, además es compatible con *Alexa*. Cuando su batería se acaba vuelve a la base y así después continuar donde lo dejó. Dispone de conectividad *WiFi* [10].

Ilustración 5 - iRobot Roomba

2.2. Análisis de sistemas

Una vez se han descrito algunos sistemas similares se van a analizar de manera cuantitativa, con el objetivo de recopilar sus características medibles para hacer una comparación, y de manera cualitativa con el objetivo de recopilar las características no medibles y establecer una opinión más personal.

	Sensores				Conectividad		
	Luz	Movimiento	Proximidad	Sonido	WiFi	Bluetooth	Zigbee
Echo Plus				X	X	X	X
Philips Hue					X		
Paneles luminosos							X
Tradfri							
Roomba					X		

Tabla 1 - Comparativa sistemas

En cuanto a su aspecto cabe destacar el diseño de los paneles luminosos, su fisionomía triangular y modular permite crear diferentes figuras con diferentes tonalidades de color. El diseño de Echo plus tiene forma cilíndrica con superficie de tela en diferentes colores: piedra arenisca, gris jaspeado y carbón vegetal, los cuales se pueden combinar con cualquier tipo de mobiliario.

Las bombillas *Philips Hue* se pueden controlar mediante una aplicación *Android* o iOS. Para personas jóvenes o mediana edad el aprendizaje y la usabilidad de la aplicación puede resultar sencilla, pero para personas de avanzada edad, estas aplicaciones pueden no ser accesibles. De igual manera puede ocurrir con la aplicación *iRobot HOME de Roomba*.

En cuanto a la accesibilidad económica, se puede decir que todos los sistemas a excepción de *TRADFRI* exceden la cantidad de cien euros, con lo que pueden ser inaccesibles para una gran parte de la población.

2.3. Síntesis

Una vez analizadas las características de estos sistemas, se pueden tomar decisiones acerca del sistema a diseñar, como el precio, las funcionalidades o el aspecto:

- CA01: Dispositivos pequeños que se puedan colocar en el hogar.
- CA02: Dispositivos económicos accesibles para toda la población.
- CA03: Capaces de detectar características del entorno.
- CA04: Capacidad de escalabilidad del sistema.
- CA05: Deben ser capaces de comunicarse entre dispositivos.

3. Análisis del problema y requisitos de la solución

Para la realización de este proyecto se pretende desarrollar una solución que incluya cubos inteligentes con capacidad de funcionamiento autónomo sobre la ambientación lumínica, es decir, sin control por parte del usuario, además debe cumplir los siguientes requisitos:

- REQ01 (CA01): El tamaño del cubo ha de ser 10x10x10 centímetros.
- REQ02 (CA02): El precio por cubo no será mayor a 60 euros.
- REQ03 (CA04): El sistema permitirá colocar tantos cubos como el usuario necesite.
- REQ04 (CA05): El sistema tendrá un algoritmo de comunicación seguro.
- REQ05 (CA03): Se detectará el sonido ambiente.
- REQ06 (CA03): Se detectará la presencia de personas.
- REQ07 (CA03): Se detectará la proximidad hasta las personas.
- REQ08 (CA03): Se detectará la luz ambiental.
- REQ09 (CA05): El sistema dispondrá de un modulo de comunicación *WiFi.*
- REQ10 (CA03): El sistema regulará la intensidad de las luces segun las necesidades del escenario.
- REQ11 (CA04): El sistema responderá con un mínimo de velocidad.

El cubo no deberá exceder las medidas mencionadas para que ocupe el menor espacio posible y quede decorativo y discreto para el usuario, es decir que quede integrado en la decoración de cualquier hogar, además de ser manejable y fácilmente redistribuible. Para ello se ha diseñado un cubo que facilita la integración de los sensores, cableado y placa en el menor espacio posible.

Para conseguir un precio que no exceda los 60 euros, se hará uso de hardware de bajo coste que se ajusta a las necesidades de este proyecto. De esta manera el sistema puede ser accesible para el usuario.

El sistema será fácilmente escalable para permitir la adaptación a cualquier vivienda, pudiendo elegir así la cantidad de cubos que se desea utilizar. Para ello cada cubo contendrá su propio hardware y su propio algoritmo.

Para obtener un algoritmo seguro se ha utilizado comunicación *WiFi* cifrado, además se han asignado IP's estáticas a los cubos, que se comprueban en el algoritmo, limitando así la intrusión de otros dispositivos a la comunicación del sistema.

3.1. Análisis de soluciones posibles

Previo a decidir cuál es la solución final para este proyecto, se han barajado diferentes posibilidades para desarrollar, las cuales por diversas razones que se comentarán a continuación se ha decidido descartar.

Una de estas posibles soluciones consiste en un sistema de cubos inteligentes, los cuales contienen también el hardware mencionado, sin embargo cada cubo será totalmente autónomo, tendrá la total capacidad de decisión sobre cómo va a proceder en cada momento según los datos que obtenga. Esta solución se ha descartado debido a que aunque siga siendo un sistema inteligente, no existe una comunicación o interacción entre los diferentes cubos que componen el sistema para poder obtener una ambientación lumínica adecuada, un caso práctico que aclare este descarte podría ser la entrada al hogar de una persona sin hacer demasiado ruido, el cubo más cercano que detecta a esta persona enciende la lámpara, pero de ninguna manera puede avisar al resto de cubos para que enciendan un mínimo de intensidad y provocar una sensación de seguridad y confort al usuario. Esta solución cumple los requisitos: REQ01, REQ02, REQ03, REQ05, REQ06, REQ07, REQ08, REQ10, y REQ11, pero no cumple los requisitos REQ04 y REQ09

Otra posible alternativa que solucionaría el problema anterior sobre crear una sensación reconfortante sería los mismos cubos con el mismo hardware y que todos estén en continuo contacto con todos los demás, de esta manera siempre habría luz en la lámpara más cercana al usuario y un mínimo de intensidad en el resto. Sin embargo esta solución también se ha descartado debido a que el continuo envío de mensajes a todos los cubos sobrecargaría el medio de comunicación, ralentizaría considerablemente el procesado de datos y en consecuencia el funcionamiento del sistema de modo que no se conseguiría el confort deseado. Esta solución cumple los requisitos: REQ01, REQ02, REQ04, REQ05, REQ06, REQ07, REQ08, REQ09 y REQ10, pero no cumple los requisitos REQ03 y REQ11, debido a que si el medio se sobrecarga, el sistema no responderá con un mínimo de velocidad, ni será escalable

Por todas estas razones la solución elegida es la que combina ambas alternativas, un sistema formado por cubos en el que existen cubos clientes y un solo cubo servidor, este último tiene capacidad de decisión sobre sí mismo, pero también sobre el resto de cubos clientes, debido a que cada uno le manda constantemente los datos recogidos y los procesa, los clientes se limitan a obedecer las instrucciones del servidor, de esta manera también se consigue crear un confort lumínico a la vez que no sobrecarga el medio. Además cumple con todos los requisitos del sistema.

3.2. Solución propuesta

3.2.1. Descripcion de la solucion

En primer lugar se va a definir el concepto de comunicación maestro/esclavo, los cuales van a estar muy presentes en este proyecto:

El sistema de comunicación maestro-esclavo consta esencialmente de un equipo que se denomina maestro y uno o más equipos denominados esclavos; el maestro es quien gobierna los ciclos de comunicación, toda iniciativa de comunicación es llevada a cabo por este equipo, los esclavos solo responden a la petición del maestro, si les corresponde.

Como se ha mencionado anteriormente, el proyecto consiste en un cubo maestro que se compone de sensores y una placa arduino, quien decide por sí mismo en todo momento si ha de encender o no su lámpara, procesando los datos que recogen sus sensores, además mediante comunicación *WiFi* procesa también los datos que los esclavos le mandan, decide si se han de encender las luces o no y con cuanta intensidad, les responde basandose en estos datos y si hay alguna otra lámpara encendida o no, debido a que si hay más luces encendidas, significa que otra lámpara si ha detectado movimiento y por lo tanto la lámpara que recibe la orden ha de encenderse a una intensidad media para crear un ambiente. Todo esto se realiza para crear un sistema que actúe como un solo organismo, adaptando la luminosidad en cada uno de los componentes. Los esclavos por su parte, no procesan los datos que reciben de sus sensores, simplemente se los mandan al maestro en un mensaje. Sin embargo serán capaces de entrar en modo autónomo si tras repetidos intentos de conexión con el maestro no reciben respuesta alguna, después de unos minutos el esclavo vuelve a reintentar la conexión con el maestro.

Cuando reciben la respuesta del maestro, automáticamente saben si encender la lámpara o no y con cuanta intensidad; el maestro en todo momento sabe a quién debe responder y el que gracias al identificador que manda el esclavo en el mensaje.

A continuación se muestra una plantilla donde se puede ver el esquema del sistema que se va a desarrollar:

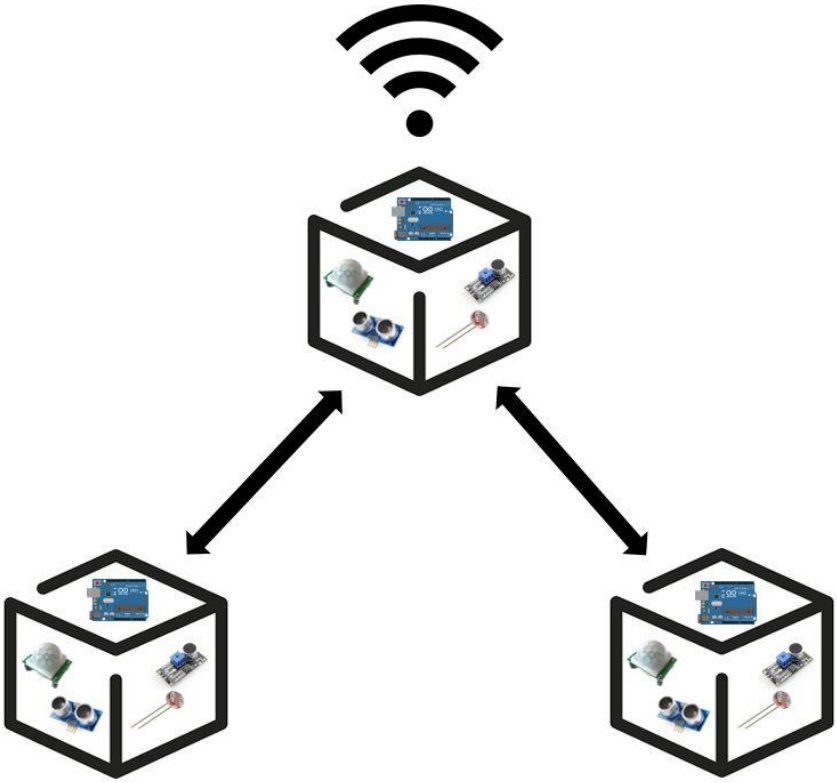

Ilustración 6 - Esquema sistema

Una vez desarrollado, se realizarán pruebas colocando tres cubos en un pasillo cercano a una puerta de entrada al hogar, el orden de colocación de los cubos será irrelevante porque están en continua comunicación. La primera prueba consistirá en repetidas entradas al hogar de una persona o varias, para comprobar que las lámparas, según como sea la entrada (con más o menos ruido) se enciendan con más o menos intensidad; en condiciones normales, se entiende que un grupo grande de personas entrando a una vivienda generará más ruido que una persona sola, por lo tanto deberían encenderse las lámparas con mayor intensidad. La segunda prueba consistirá en recorrer el pasillo varias veces para comprobar que las lámparas responden correctamente al movimiento de la persona y se encienden cuando está próxima y disminuye la intensidad conforme se va alejando. Cabe mencionar que si los sensores detectan suficiente luz en el hogar como para no encender las luces, aunque haya muchas personas estas no se encenderán para reducir el consumo energético.

3.2.2. Casos de uso

- Primer caso de uso:

  En este caso de uso, el actor es el autor del Sistema, cuando inicializa todas las variables y conecta los cubos inteligentes por primera vez, tanto el maestro como los esclavos comienzan con su ejecución, un bucle que se ejecuta constantemente hasta que los cubos vuelven a ser desconectados de la corriente eléctrica.

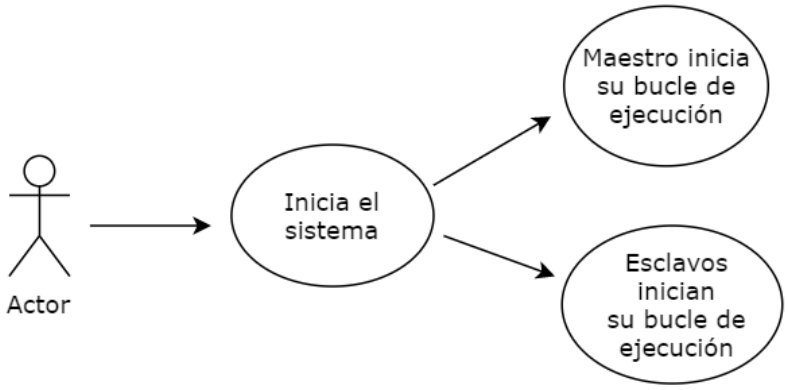

Ilustración 7 - Caso de uso 1

- Segundo caso de uso:

  En este caso de uso el actor principal es el usuario, antes de entrar al hogar, el sistema ya estaba en funcionamiento, los sensores captan información pero no se dan las condiciones necesarias para que las luces se enciendan, cuando entra al hogar, el sensor de presencia, de movimiento y de sonido, empiezan a captar datos del entorno proporcionados por la presencia del usuario, el sensor de luminosidad no se asocial al usuario porque depende de la luz que reciba del entorno, sin embargo, todos ellos dan lugar a que los cubos establezcan un primer escenario de confort.

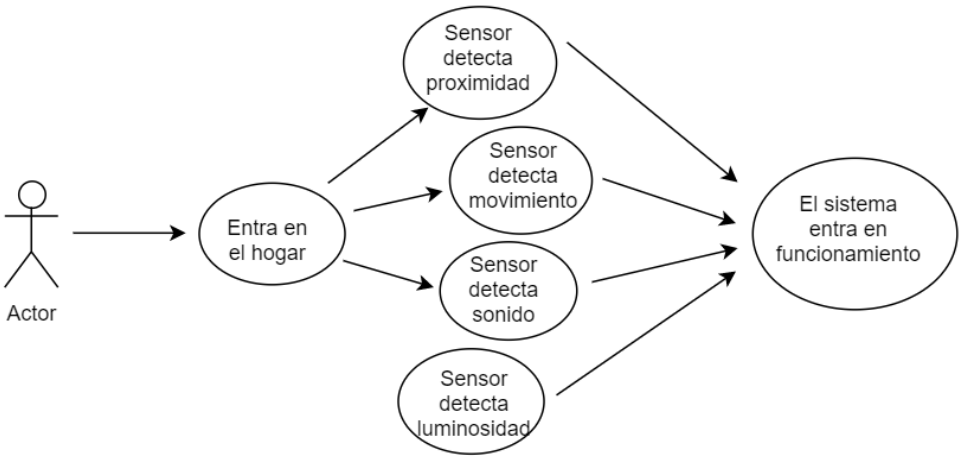

Ilustración 8 - Caso de uso 2

- Tercer caso de uso:

  Este caso de uso es similar al anterior, el actor también es el usuario, y en lugar de entrar al hogar se mueve de una habitación a otra, en este caso el sistema ya estaba en funcionamiento con lo que los valores captados varían a cada movimiento del usuario, lo que provoca que el maestro entre en constante comunicación con los esclavos para crear el escenario de confort óptimo.

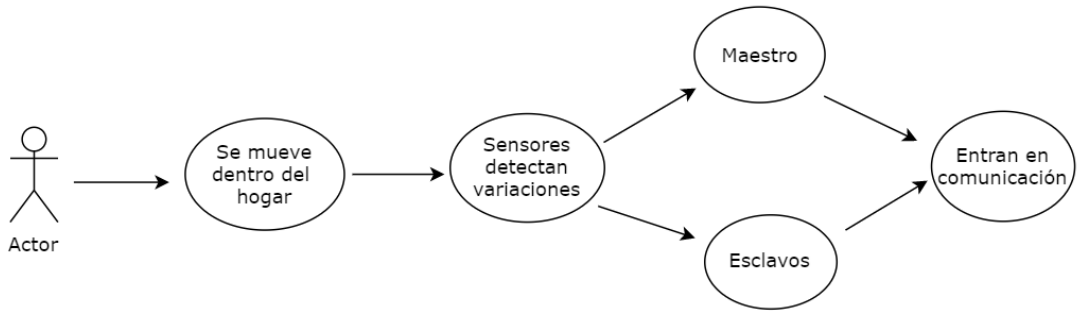

Ilustración 9 - Caso de uso 3

3.2.3. Funcionalidades

En éste apartado se van a describir las diferentes funcionalidades del sistema, extraídas a partir de los casos de uso, cuales son sus entradas y salidas y una breve descripción.

- Funcionalidades primer caso de uso:

Funcionalidad 1	Ejecutar el algoritmo del maestro ininterrumpidamente
Input	Datos percibidos por los sensores propios y de los esclavos
Output	Ambientación lumínica basada en los datos recogidos por los propios sensores

Tabla 2 - Funcionalidad 1 caso de uso 1

Funcionalidad 2	Ejecutar el algoritmo del esclavo ininterrumpidamente
Input	Datos percibidos por los propios sensores
Output	Ambientación lumínica basada en los datos recibidos por el maestro

Tabla 3 - Funcionalidad 2 caso de uso 1

- Funcionalidades segundo caso de uso:

Funcionalidad 1	Detectar la proximidad
Input	Sensor de proximidad
Output	Distancia en centímetros entre el sensor y un sujeto

Tabla 4 - Funcionalidad 1 caso de uso 2

Funcionalidad 2	Detectar movimiento
Input	Sensor de movimiento
Output	1 si detecta movimiento o 0 si no detecta

Tabla 5 - Funcionalidad 2 caso de uso 2

Funcionalidad 3	Detectar sonido
Input	Sensor de sonido
Output	Nivel de sonido expresado en db

Tabla 6 - Funcionalidad 3 caso de uso 2

Funcionalidad 4	Detectar luz
Input	Sensor de luz
Output	Umbral entre 0 y 100 de luz detectada

Tabla 7 - Funcionalidad 4 caso de uso 2

- Funcionalidades tercer caso de uso:

Funcionalidad 1	Recibir mensaje del esclavo
Input	Datos de sensores del esclavo
Output	Comunicar instrucción de encendido o apagado de luces al esclavo

Tabla 8 - Funcionalidad 1 caso de uso 3

Funcionalidad 2	Enviar mensaje al maestro
Input	Datos de los sensores del esclavo
Output	Mensaje con los valores de los sensores al maestro

Tabla 9 - Funcionalidad 2 caso de uso 3

4. Diseño de la solución

En este apartado se va a explicar los dos bloques en los que se ha dividido el diseño de la solución, un bloque es la arquitectura física, donde se describirá el diseño de los cubos inteligentes y de que manera contiene los sensores, el otro bloque es la arquitectura lógica que es la encargada del cumplimiento de todos los requisitos especificados. También se va a describir las tecnologías utilizadas, es decir, los componentes que formarán parte del sistema, y el software utilizado, todos ellos harán posible la consecución del objetivo.

4.1. Arquitectura física del Sistema

4.1.1. Elementos del sistema

Los elementos que forman el sistema son cualquier estancia del hogar, que contendrá tantas luminarias como el usuario necesite ya que el hecho de que el sistema sea escalable permite poner tantas como se desee. En el esquema se pueden observar tres luminarias, rodeadas de una línea discontinua que representa su alcance, todo lo que esté dentro será captado por los sensores de esa luminaria y será transmitido o procesado según el papel que desempeñe. Más en profundidad se puede ver que todas las luminarias contienen un cubo con un controlador al que están conectados diversos sensores, estos sensores captan lo que está dentro del rango de alcance de las luminarias. En consecuencia a esto el controlador muestra valores que producirán una acción sobre las luminarias.

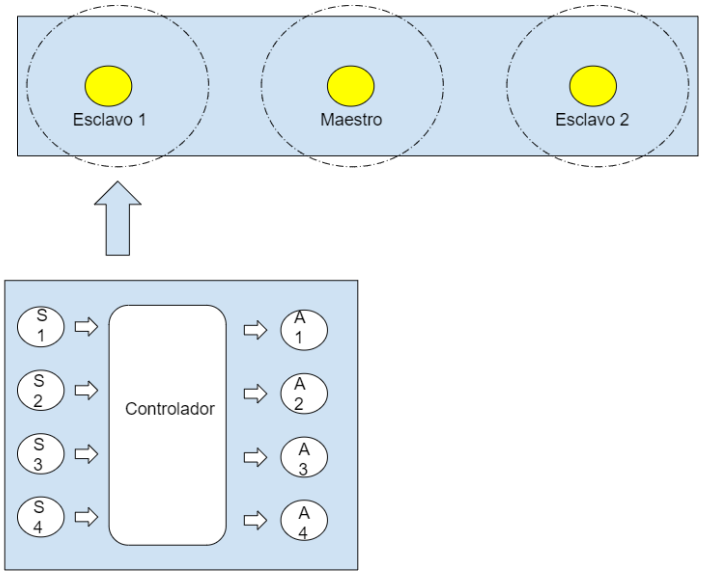

Ilustración 10 - Diseño físico

4.1.2. Relación entre elementos

La relación que existe entre los elementos es un canal de comunicación vía TCP porque ofrece mucha confiabilidad a la hora de la transmisión de datos y un servicio orientado a la conexión, en este proyecto es necesaria una alta confiabilidad debido a que los datos de los sensores de los esclavos deben llegar íntegros al maestro y viceversa para asegurar la sensación de confort adecuada para el usuario. Los esclavos no guardan relación entre ellos, únicamente con el maestro, y solamente el maestro con todos ellos porque como ya se comentó anteriormente, una comunicación entre todos los elementos del sistema sobrecargaría el medio y no actuaría con la suficiente velocidad para crear el ambiente de manera instantánea con el movimiento del usuario.

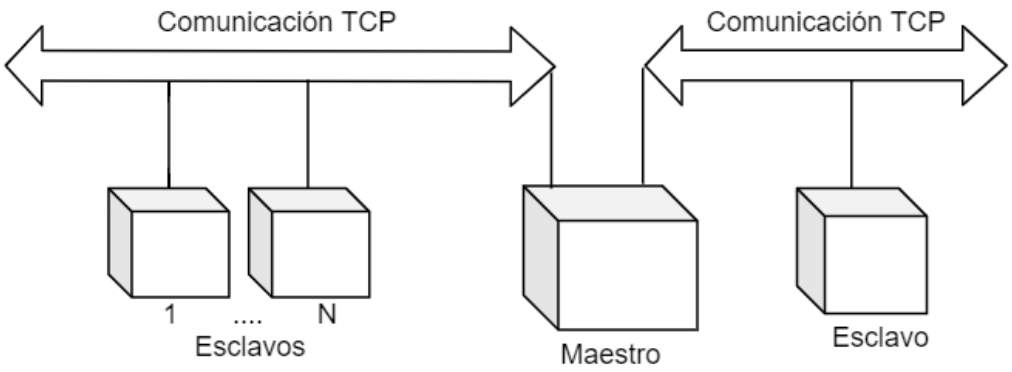

Ilustración 11 - Relación de elementos del diseño físico

4.2. Arquitectura lógica del Sistema

La arquitectura lógica se va explicar mediante un diagrama *UML* de secuencia. Un diagrama *UML* o *Lenguaje Unificado de Modelado* es un conjunto de herramientas de modelado que orienta la creación y notación de diversos diagramas.

Un diagrama de secuencia es un tipo de diagrama de interacción porque describe cómo un grupo de objetos funcionan en conjunto. Estos diagramas se suelen usar para comprender los requisitos de un nuevo sistema, también se les conoce como diagramas de eventos.

En el siguiente diagrama se muestra la ejecución del algoritmo, en primer lugar se inicia el sistema, y se piden los datos de los diferentes sensores, todos ellos devuelven los datos recogidos, los datos los reciben tanto el maestro como el esclavo a través de sus propios sensores, el maestro devuelve los datos procesados con la información sobre si se enciende la luminaria o no y el esclavo manda sus datos al maestro para que los procese y le devuelva la instrucción, después el esclavo manda la información sobre si se enciende la luminaria o no.

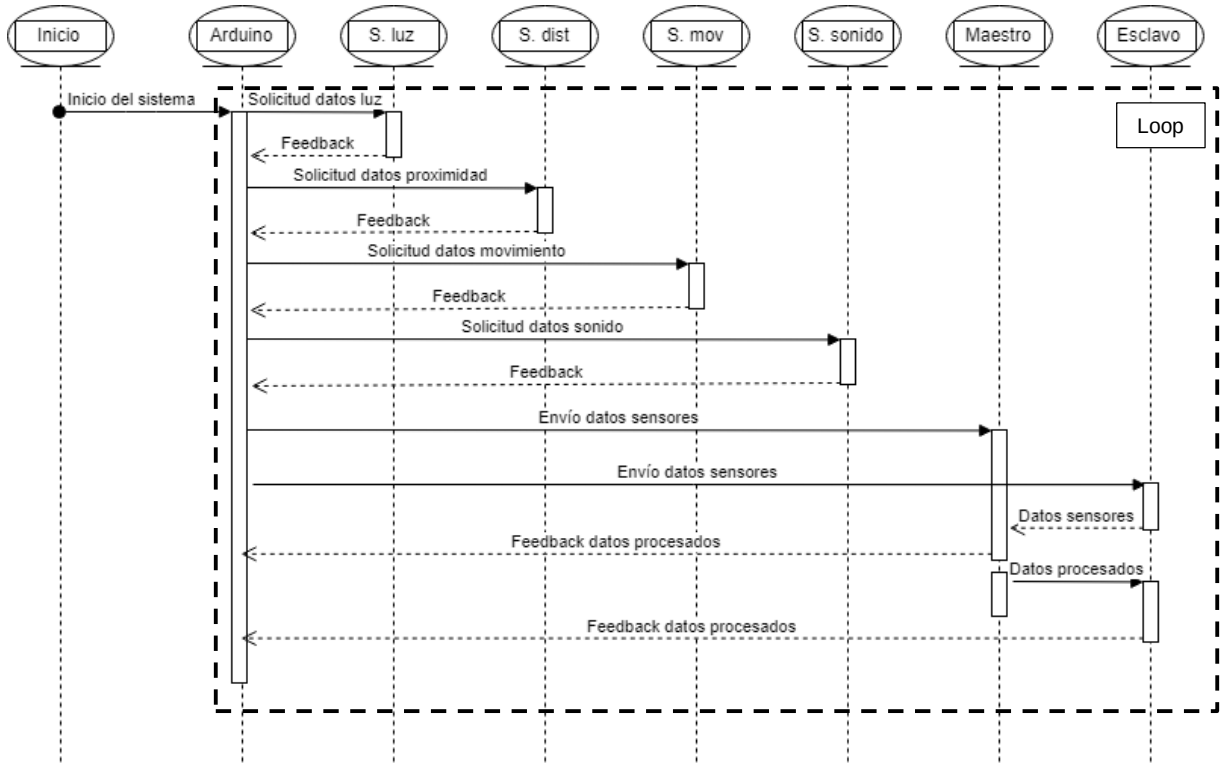

Ilustración 12 - Diseño lógico

4.2.1. Protocolo de comunicación

Durante la comunicación el maestro y el esclavo intercambian diversos mensajes para la creación del escenario que corresponda según sus valores, la comunicación la comienza el esclavo, que lee los valores de sus sensores y si obtiene conexión con el maestro formará un mensaje con los siguientes campos:

Campos	Significado
Id	Dirección IP que identifica al esclavo
Distancia	Distancia recogida desde la persona al sensor
Luz	Umbral de luz que recibe el sensor
Movimiento	Indica si hay movimiento o no
Sonido	Umbral de sonido que recibe el sensor

Tabla 10- Mensaje del esclavo

El maestro evalúa los datos que el esclavo le manda, decide si la lámpara se ha de encender o no y con cuanta intensidad, la respuesta del maestro puede contener cualquiera de estas opciones:

Respuesta	Significado
"0"	Luz apagada
"1"	Luz a media intensidad
"2"	Luz con intensidad alta

Tabla 11 - Mensaje del maestro

El formato del mensaje es una cadena de tipo String, tanto el envío del esclavo como el envío del maestro:

```
String Message = "192.168.4.2, ";
  Message += valorDistancia;
  Message += ", ";
  Message += valorLuz;
  Message += ", ";
  Message += valorMovimiento;
  Message += ", ";
  Message += valorSonido;
```

Este es el envío del esclavo al maestro.

El maestro envía el resultado de la evaluación a través de una variable de tipo String: `messageForSlave` que devuelve 0, 1 o 2.

La secuencia de envío de mensajes desde que se inica la conexión hasta el fin del ciclo es la siguiente:

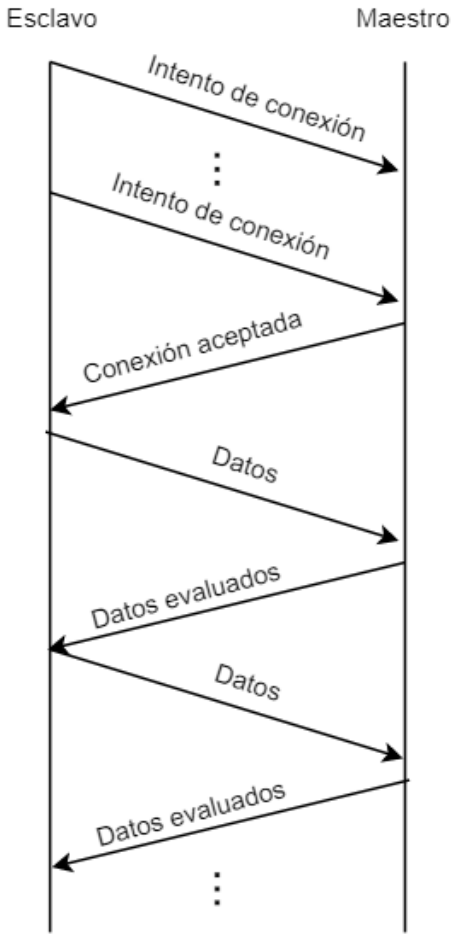

Ilustración 13 - Secuencia mensajes

El esclavo manda intentos de conexión hasta que el maestro está preparado y se conectan, y una vez el maestro ha aceptado la conexión el esclavo manda los datos, el maestro le devuelve otro mensaje con la resolución, y esto se repite varias veces.

4.3. Tecnología empleada

En este apartado se van a comentar los componentes hardware y el software que van a formar parte del sistema.

4.3.1. Sensor PIR

Los *sensores infrarrojos pasivos* o *PIR* son dispositivos que detectan movimiento, esto ocurre porque todos los cuerpos emiten energía infrarroja mayor cuanto mayor es su temperatura. Su coste es asequible, son pequeños, de baja potencia y son sencillos de usar. Por esta razón se usan normalmente en aplicaciones domóticas. Estos sensores disponen de un sensor capaz de captar la radiación infrarroja y convertirla en señal eléctrica [11].

Ilustración 14 - Sensor PIR

Cada sensor está dividido en dos campos, si ambos reciben la misma cantidad de infrarrojos, la señal eléctrica resultante es nula, si los dos campos realizan una medición diferente se genera una señal eléctrica.

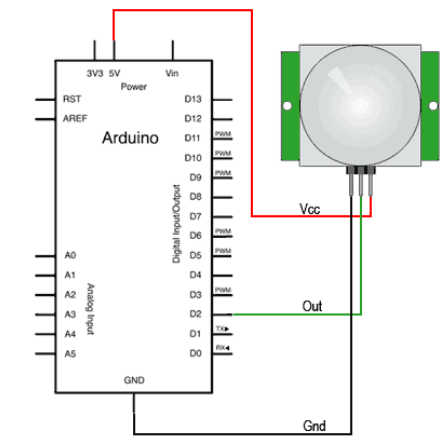

Ilustración 15 – Esquema eléctrico PIR

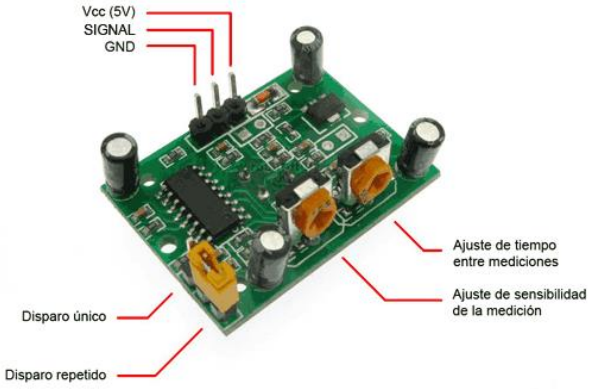

Ilustración 16 - Esquema patillaje PIR

4.3.2. Micro

El *micrófono Electret* es una variante del micrófono de condensador que usa un electrodo. Se polarizan las placas durante la fabricación, así que no necesita alimentación, (aunque el preamplificador interno si necesita) lo que permite prolongar su tiempo de vida (aproximadamente 50 años). Puede ser omnidireccional o unidireccional, son sencillos de encontrar y de utilizar, son robustos, económicos y pequeños con lo que son sencillos de esconder, por éste motivo se usan en teléfonos móviles, grabadoras de bolsillo, se pueden ocultar en la ropa o en objetos.

Ilustración 17 – Micrófono Electret

Responden bién en frecuencia en el rango audible de 50 a 15.000Hz, y tienen una sensibilidad entre -50 dB y -70 dB, se alimenta con una tension de entre 2 y 12V, además de manera interna está preamplificado para dar una mayor señal y calidad, por otra parte es resistente, poco ruidoso, muy sensible e insensible al calor; aunque la humedad puede ocasionar corto circuito, no es en sí un factor de riesgo para el funcionamiento.

La principal causa que puede ocasionar fallos en el micrófono es la acumulación de polvo porque deteriora el rendimiento y eficacia, produce zumbidos e interferencias.

Tiene una baja respuesta ante tonos altos, lo cual es una ventaja si no se va a trabajar con tonos altos como ultrasonidos, ya que si no capta estas interferencias que son inaudibles no es necesario filtrarlas, a su vez es apto para sonidos débiles lo que es un inconveniente si se trabaja con sonidos altos, pues el preamplificador se satura produciendo una señal distorsionada.

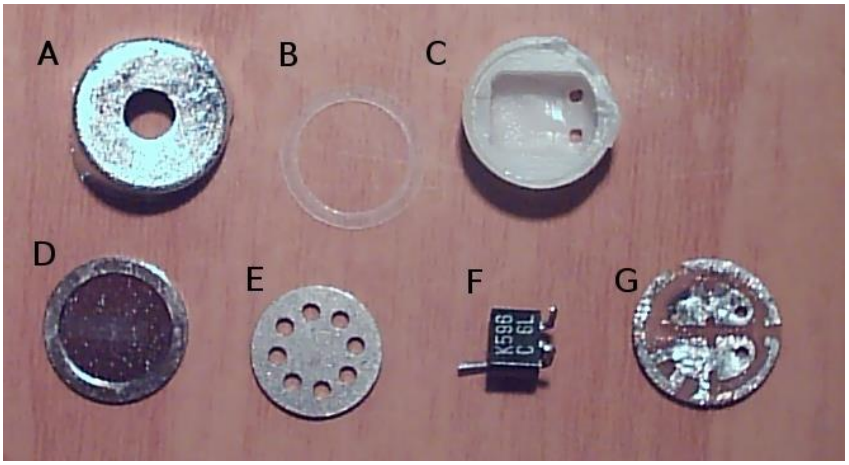

Ilustración 18 - Partes del micrófono Electret

4.3.3. Sensor de ultrasonidos

Es un dispositivo para medir distancias, funciona enviando pulsos de alta frecuencia no audible por el humano, este pulso rebota en el objeto y es reflejado hacia el sensor que dispone de un micrófono adecuado para esa frecuencia. Midiendo el tiempo entre pulsos, conociendo la velocidad del sonido, se puede estimar la distancia del objeto contra cuya superficie impactó el impulso de ultrasonidos. Son sensores económicos y sencillos de usar, el rango de medición teórico es de 2 a 400 centímetros. Son ampliamente utilizados, en robótica es habitual usarlo para la detección de obstáculos.

Ilustración 19 - Sensor de ultrasonidos

Su funcionamiento se basa en medir el tiempo entre envío y recepción de un pulso sonoro, la velocidad del sonido es 343 m/s. Transformando unidades resulta:

$$343\frac{m}{s} \cdot 100\frac{cm}{m} \cdot \frac{1}{1000000}\frac{s}{\mu s} = \frac{1}{29.2}\frac{cm}{\mu s}$$

Ilustración 20 - Ecuación transformación de unidades

El sonido tarda 29.2 microsegundos en recorrer un centímetro. Por tanto se puede obtener la distancia a partir del tiempo entre la emisión y recepción del pulso:

$$Distancia(cm) = \frac{Tiempo(\mu s)}{29.2 \cdot 2}$$

Ilustración 21 - Ecuación obtención de distancia

Se divide entre dos el tiempo porque se mide el tiempo que tarda el pulso en ir y volver, por lo tanto la distancia recorrida es el doble [12].

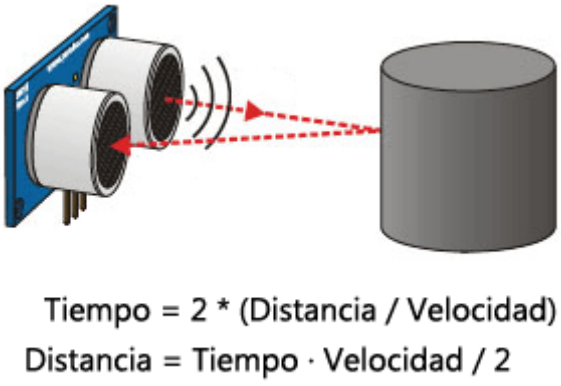

Ilustración 22 - Funcionamiento sensor ultrasonidos

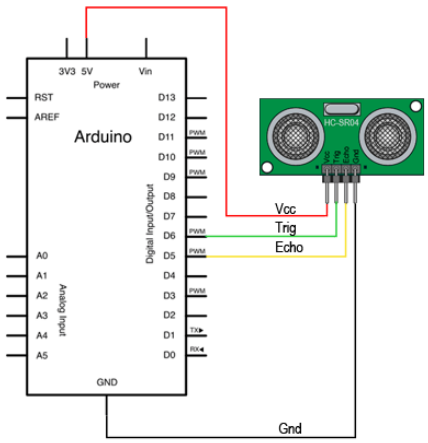

Ilustración 23 - Esquema eléctrico

4.3.4. Sensor fotoresistor

Un *fotoresistor* o *LDR (GL55)* es un dispositivo cuya resistencia varía en función de la luz recibida. Se puede usar esta variación para medir a través de las entradas analógicas una estimación del nivel de luz. Está formado por un semiconductor, y al incidir luz sobre él algunos de los fotones, son absorbidos provocando que electrones pasen a la banda de conducción y, por lo tanto disminuye la resistencia del componente a medida que incide la luz sobre él. Un *LDR* es un sensor que resulta adecuado para proporcionar medidas cuantitativas sobre el nivel de luz, en interiores como exteriores y reaccionar por ejemplo encendiendo una luz o subiendo una persiana.

Ilustración 24 - Sensor fotoresistor

La relación entre la iluminancia y la resistencia de una LDR sigue una función potencial, siento R0 la resistencia a una intensidad I0, ambas conocidas, la constante gamma tiene un valor de 0.5 a 0.8.

$$\frac{I}{I_0} = \left(\frac{R}{R_0}\right)^{-gamma}$$

Ilustración 25 - Relación iluminancia resistencia

A continuación se muestra su esquema eléctrico y una gráfica donde se puede apreciar el funcionamiento del fotoresistor, cuánto mayor resistencia opone el resistor, menos corriente pasa y menos luz emite el led [13].

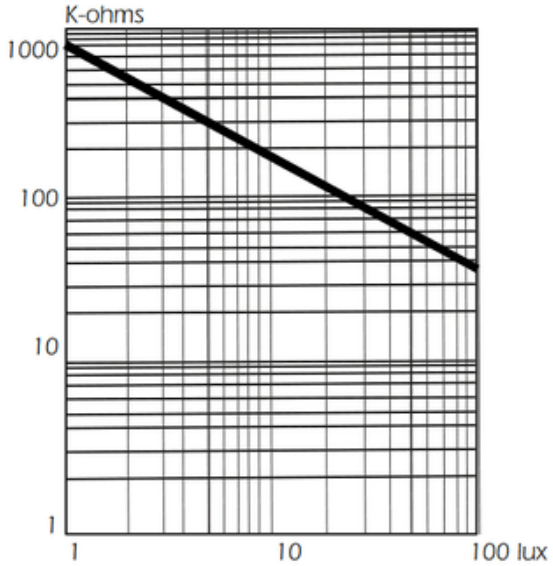

Ilustración 26 - Funcionamiento fotoresistor

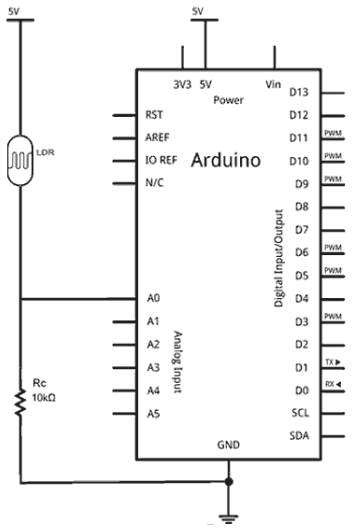

Ilustración 27 - Esquema eléctrico

4.3.5. Módulo WiFi ESP8266 ESP01

Es un microprocesador de bajo coste con *WiFi* incorporado, antes de este módulo las posibilidades disponibles para conectar un *Arduino* a *WiFi* tenían un coste elevado. Muchos modelos de placas ya integran el *ESP8266*. El *ESP01*, un módulo que fue de los primeros en aparecer con el chip *ESP8266*, tiene comunicación integrada, *802.11 b/g/n*, incluidos modos *WiFi Direct (P2P)* y *soft-Ap*, también incluye una pila *TCP/IP* completa. Mencionar que este módulo se puede usar para dotar de conectividad a los *Arduinos*, sin embargo la comunicación con internet supone una sobrecarga para la placa.

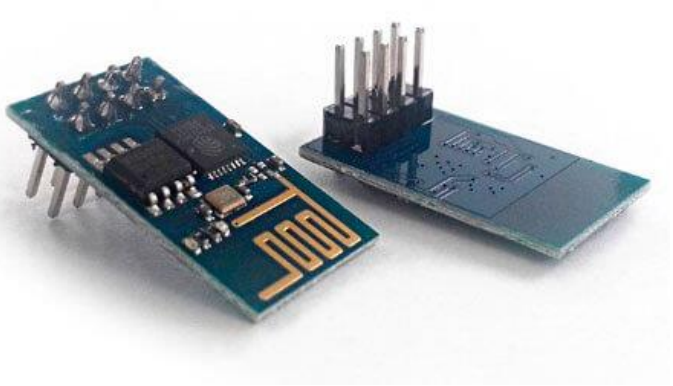

Ilustración 28 - Módulo WiFi

No es excesivamente complicado la conexión de *ESP01* con *Arduino*, la única dificultad que se presenta es la alimentación. El *ESP8266*, y en particular *ESP01*, tiene una tensión de alimentación de 3.3V. No se debe alimentar a una tensión mayor a 3.6V o se puede estropear el módulo.

En otro orden de cosas, el consumo del módulo puede superar los 200mA, sobre todo durante la conexión y arranques, sin embargo, el regulador de voltaje de 3.3V de *Arduino* sólo proporciona 50mA, lo que no es suficiente para alimentar el *ESP01* así que será necesario alimentarlo con una fuente externa de 3.3V. De lo contrario se producirán repetidos cortes durante el funcionamiento, que además reducirán su vida útil.

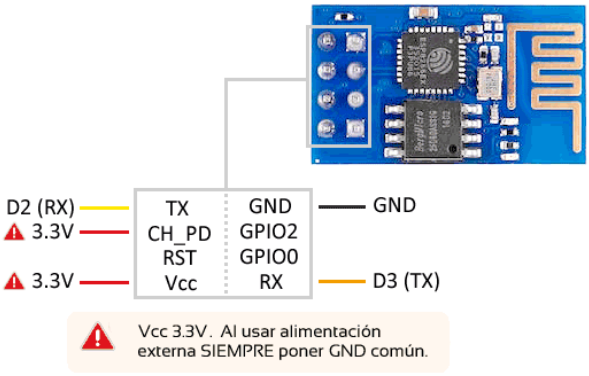

Ilustración 29 - Esquema módulo WiFi

El resto de la conexión no presenta dificultad alguna, el pin *CH_PD* que apaga o enciende el módulo conectándolo a Gnd o 3.3V. El pin *RST* reinicia el módulo si se conecta a Gnd.

Para finalizar, cabe mencionar que la comunicación con el módulo se realizará mediante puerto serie.

4.3.6. Placa Arduino

Es una placa para microcontroladores que contiene 14 pines de I/O digital, 6 entradas analógicas, un USB, un conector de alimentación, un encabezado *ICSP* y un botón de reiniciar. Para utilizarse es suficiente con conectarlo al ordenador con un cable USB. *"Uno"* significa uno en italiano, se decidió este nombre para marcar el lanzamiento de *Arduino Software (IDE)* 1.0, las cuales fueron la versión de referencia.

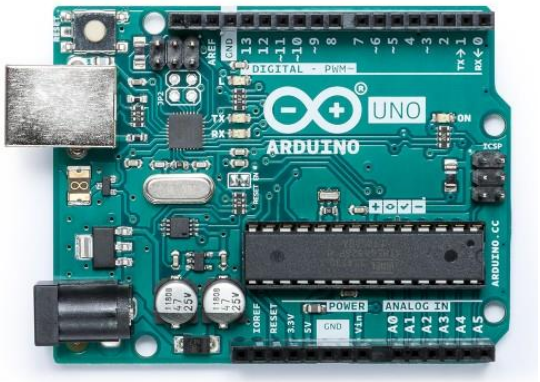

Ilustración 29 - Placa parte delantera

Ilustración 31 - Placa parte trasera

A continuación se muestra una tabla donde se pueden ver las especificaciones técnicas de la placa *"Uno":*

Microcontrolador	ATmega328P
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Pernos digitales de I/O	14 (de los cuales 6 proporcionan salida PWM)
PWM Digital I/O Pins	6
Clavijas de entrada analógica	6
Corriente DC por Pin de I/O	20 mA
Corriente DC para 3.3V Pin	50 mA
Memoria flash	32 KB (ATmega328) de los cuales 0.5 KB utilizados por el cargador de arranque
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)

Velocidad de reloj	16 MHz
LED_BUILTIN	13
Longitud	68.6 mm
Anchura	53.4 mm
Peso	25 g

Tabla 12 - Especificaciones placa

A continuación se van a comentar los pines de alimentación de la placa *Arduino "Uno",* son los siguientes:

- Vin: El voltaje de entrada a la placa cuando usa una fuente de alimentación externa.
- 5V: Este pin emite 5 V regulados desde el regulador en la placa.
- 3V3: Un suministro de 3.3 voltios generado por el regulador de a bordo. El consumo máximo de corriente es de 50 mA.
- GND: Pines de tierra.
- IOREF: Este pin proporciona la referencia de voltaje con la que opera el Microcontrolador.

Cada uno de los 14 pines digitales puede usarse como I/O utilizando las funciones `pinMode()`, `digitalWrite()` y `digitalRead()`, funcionan a 5 V, pueden dar o recibir 20 mA, tiene una resistencia pull-up interna de 20 a 50k ohm y no se debe sobrepasar los 40mA en ningún pin de I/O para evitar daños. Algunos pones tienen funciones especiales:

- Serie: 0 (RX) y 1 (TX). Se usa para recibir (RX) y enviar (TX) datos serie TTL.
- Interrupciones externas: 2 y 3. Se pueden configurar para activar una interrupción.
- PWM: 3, 5, 6, 9, 10 y 11. Proporcionan una salida *PWM* de 8 bits con la función `analogWrite()`.
- SPI: 10 (SS), 11(MOSI), 12(MISO), 13(SCK), soportan la comunicación *SPI* utilizando la biblioteca SPI.
- LED: 13, tiene un led incluido, cuando el pin tiene un valor ALTO se enciende y cuando está BAJO, apagado.
- TWI: pin A4 o *SDA* y pin A5 o *SCL*.

"Uno" tiene además 6 entradas desde A0 a A5 que son analógicas y todas ellas proporcionan 10 bits de resolución (1024 valores diferentes).

4.3.7. Fritzing

Iniciativa de hardware de código abierto que hace que la electrónica sea accesible como material creativo. Es una herramienta de software que permite a los usuarios documentar sus prototipos, compartirlos, enseñar electrónica y diseñar y fabricar pcbs profesionales.

Este esquema consta de un *Arduino* conectado a un led, un fotoresistor y dos resistencias. Para insertar todos estos elementos hay un panel a la derecha con todo tipo de componentes, bastaría con arrastrarlos sobre el esquema que se desee montar.

Ilustración 32 - Montaje diseño con protoboard

De la misma manera que se hace el prototipo sobre una protoboard se puede hacer de manera esquemática y PCB, incluye también una interfaz para desarrollar código.

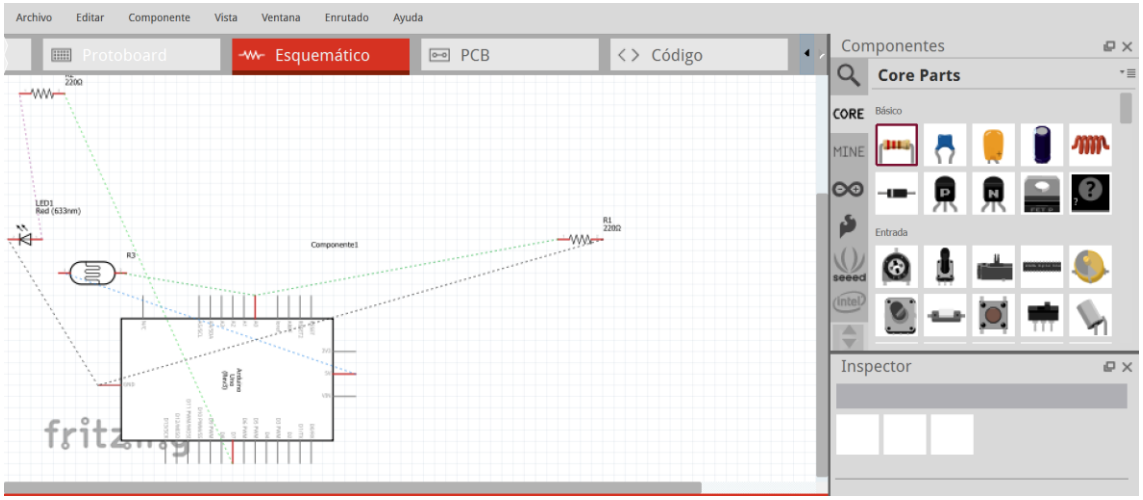

Ilustración 33 - Montaje diseño esquemático

4.3.8. Draw.io

Es una aplicación web gratuita que permite crear diagramas de flujo, diagramas *UML*, organigramas, etc.. desde el navegador, de manera sencilla y sin tener que comprar licencias de *Microsoft*. Dispone de diversas formas y diseños que se pueden moldear, permite agregar imágenes externas utilizando el buscador de *Google*.

Este diagrama consta de diversos elementos que se pueden encontrar en el panel de la izquierda, para colocarlos simplemente basta con arrastrarlos al centro o pulsar encima de él. El panel de la derecha sirve para la configuración del papel donde se añaden los elementos.

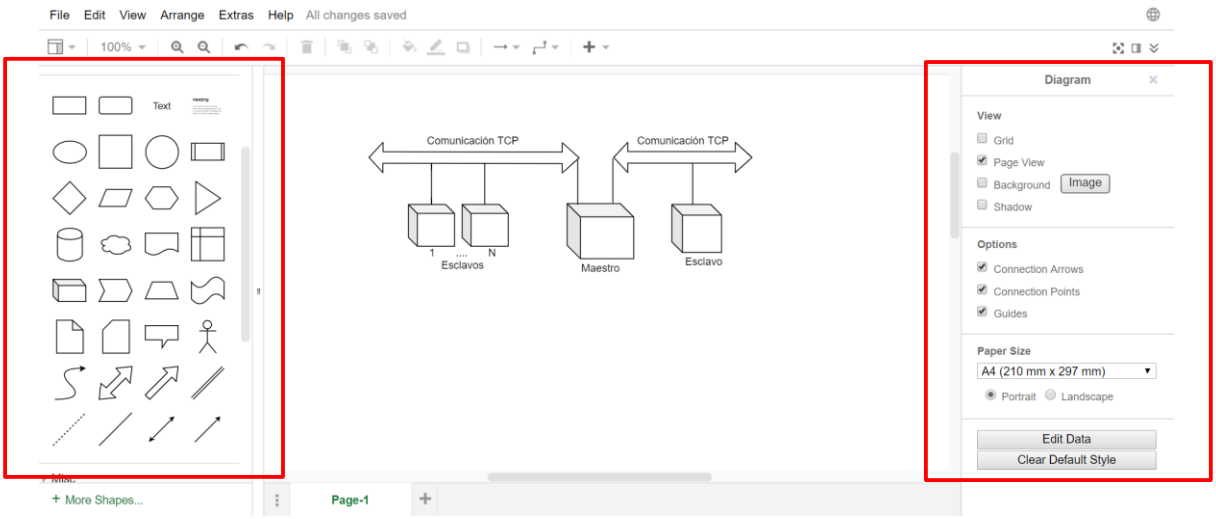

Ilustración 34 - Elementos para montaje

Cada elemento que se introduce en el papel se puede modificar a mano o bién se pueden modificar sus atributos con el panel de la derecha, como el tamaño, la posición, el color, la inclinación, etc..

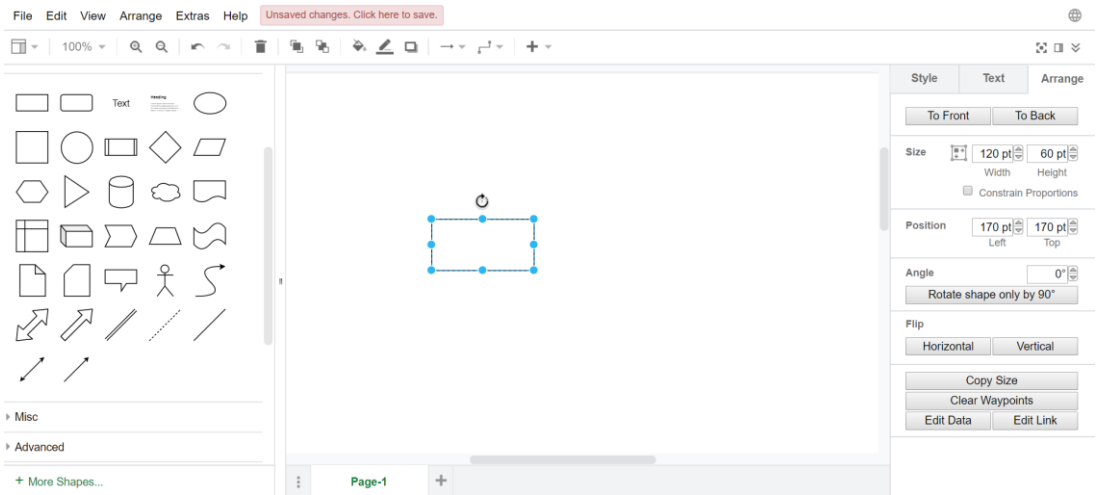

Ilustración 35 - Edición de elementos

4.3.9. Google drawings

Es un software gratuito para hacer diagramas desarrollado por *Google* que permite a los usuarios colaborar y trabajar en equipo en tiempo real. Permite crear diagramas de flujo, organigramas, wireframes de sitios web, mapas conceptuales entre otros. Está disponible como aplicación de *Chrome* que funciona sin conexión y los archivos se guardan de manera predeterminada en *Google Drive*.

Para insertar una forma o cualquier elemento se pulsa encima del elemento que se quiere dibujar y con el raton o touchpad se pulsa y se arrastra para pintar la forma.

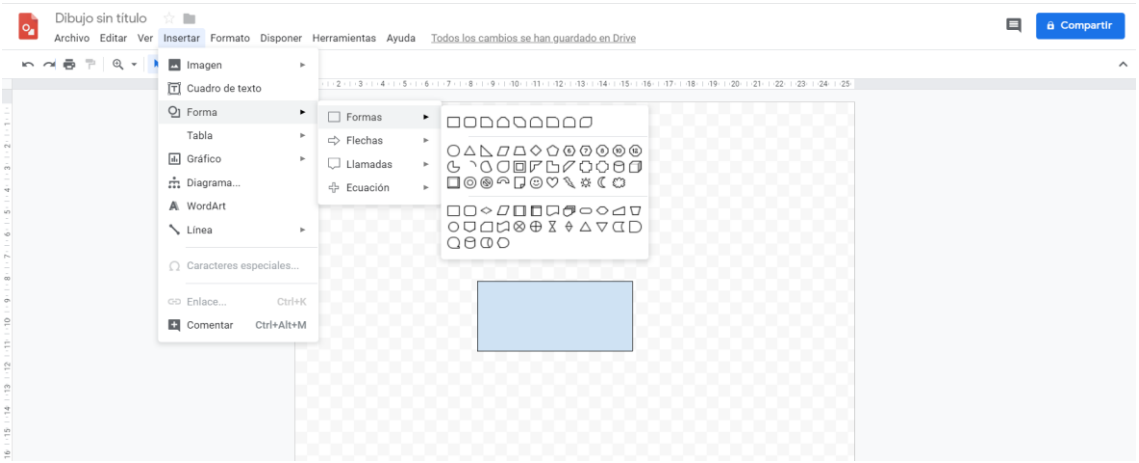

Ilustración 36 - Inserción de elementos

Para modificar los atributos de la forma insertada se pulsa encima de la misma y aparece un nuevo menu en el que se puede modificar la línea del borde, el grosor, el color de la forma, se puede insertar texto dentro, entre otras cosas.

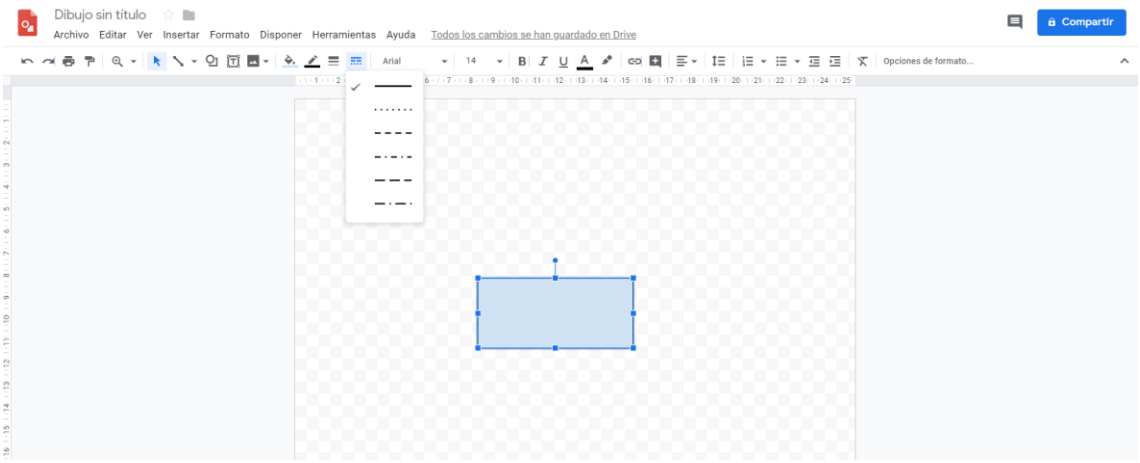

Ilustración 37 - Edición de elementos

4.3.10. Arduino (IDE)

El *IDE de Arduino* (integrated development environment) es un programa informático formado por herramientas de programación, se puede usar para uno o varios lenguajes de programación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). En caso de *Arduino* contiene herramientas para cargar los programas compilados en la memoria flash del hardware a través del puerto serie.

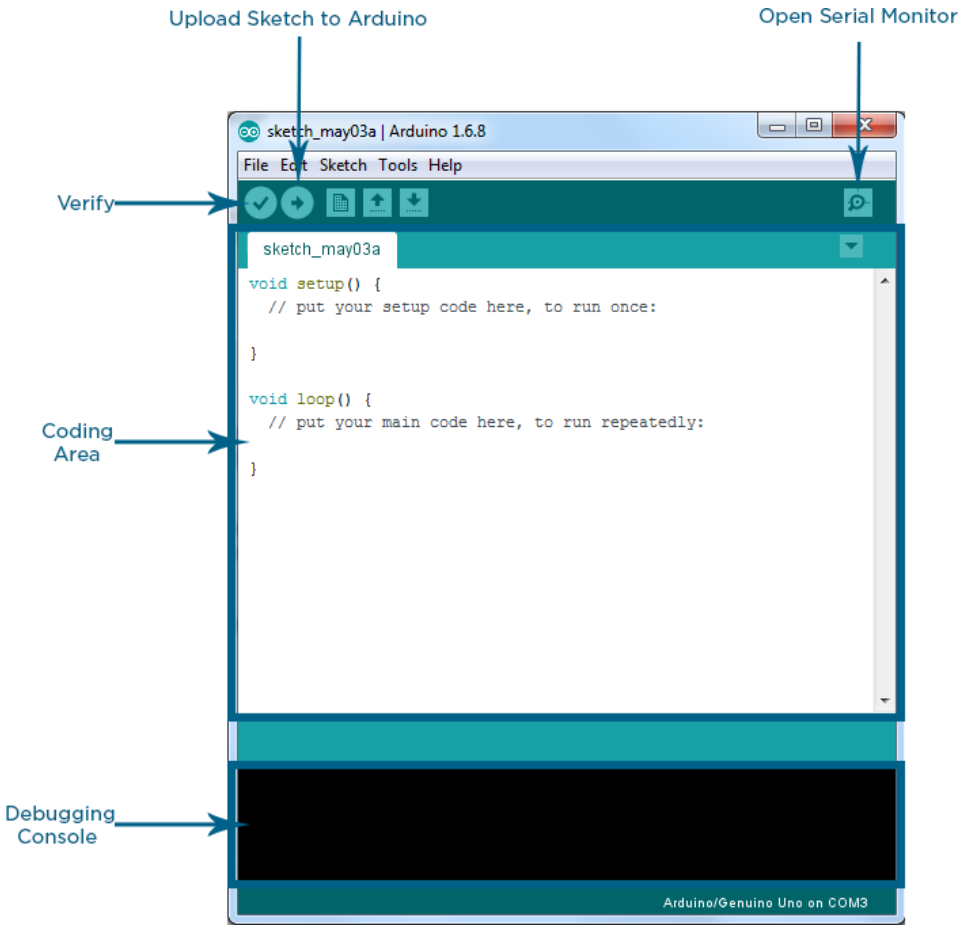

Ilustración 38 - IDE Arduino

Los programas están formados por un único fichero con extension “ino”, pero es posible organizarlo en diversos ficheros. Es importante que el fichero principal esté en una carpeta con el mismo nombre que el fichero. La última version del *IDE* es la 1.6.8, pero en la version 1.6.2 se introdujeron la gestión de librerías y gestión de placas.

5. Implementación

En este punto se van a describir las implementaciones de los bloques físicos y lógicos. El prototipo físico describirá como los sensores estarán interconexionados con la placa *Arduino*. El bloque lógico es el encargado de inicializar el protocolo de comunicación *WiFi 802.11 b/g/n*, procesar todos los datos que se recogen por los sensores incorporados y enviar y recibir instrucciones con estos datos.

5.1. Implementación del prototipo físico

Una vez se ha descrito el diseño del sistema, se va a explicar más en detalle el ensamblaje del prototipo físico, es decir, como se han conectado electrónicamente todos los elementos entre ellos para formar el cubo:

1. En primer lugar se han sacado unos cables de los pines 5V y GND a la protoboard para dotarla de toma a tierra y toma positiva.
2. Conexión de los sensores y módulo *WiFi*:
    - El sensor *PIR* utiliza los pines 11 para la patilla OUT, y las patillas VCC y GND se conectan a la protoboard a su correspondiente polaridad.
    - El *micro* utiliza la entrada analógica A1, las patillas VCC y GND se conectadan a la protoboard con su correspondiente polaridad.
    - El sensor de *ultrasonidos* utiliza los pines 8 para el Echo y el 9 para el Trig, las patillas VCC y GND se conectan a la protoboard con su correspondiente polaridad.
    - El sensor *fotoresistor* utiliza el pin A0.
    - Por último el módulo *WiFi*, la patilla de recepción (RX) utiliza el pin 2 y el de transmisión (TX) utiliza el pin 3.

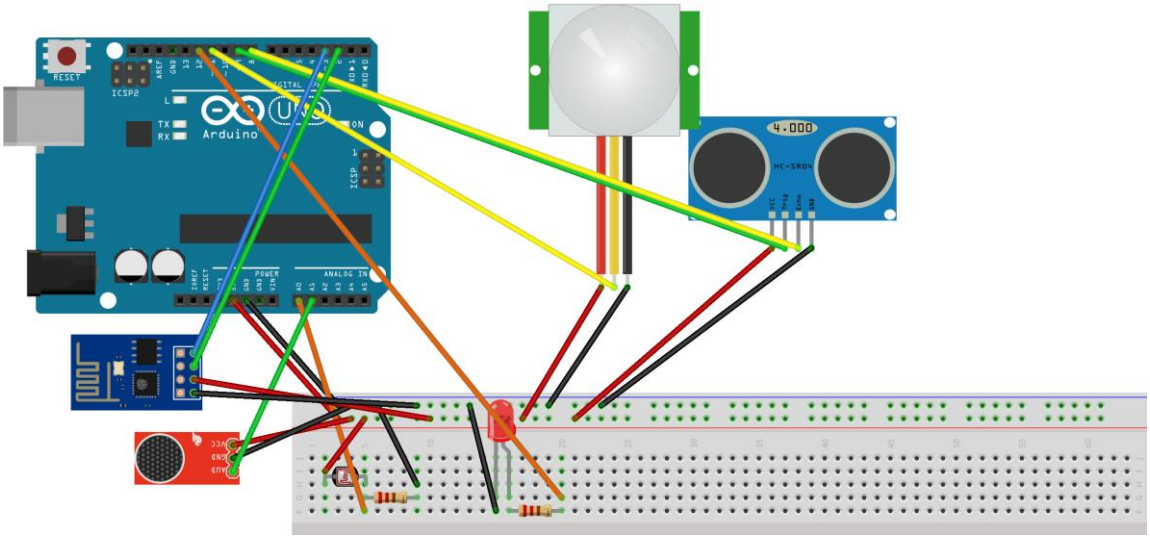

Ilustración 39 - Prototipo físico

3. Una vez se ha obtenido el diseño físico de la ilustración anterior se han soldado los circuitos de los sensores para evitar el uso de la protoboard y poder insertar la placa *Arduino* en los cubos.

4. Para la realización de los cubos de PVC se han cortado placas de 10x10 y se han perforado con las medidas de los sensores para que tengan salida al exterior y puedan así captar los valores del ambiente. A continuación se muestra el diseño de los cubos:

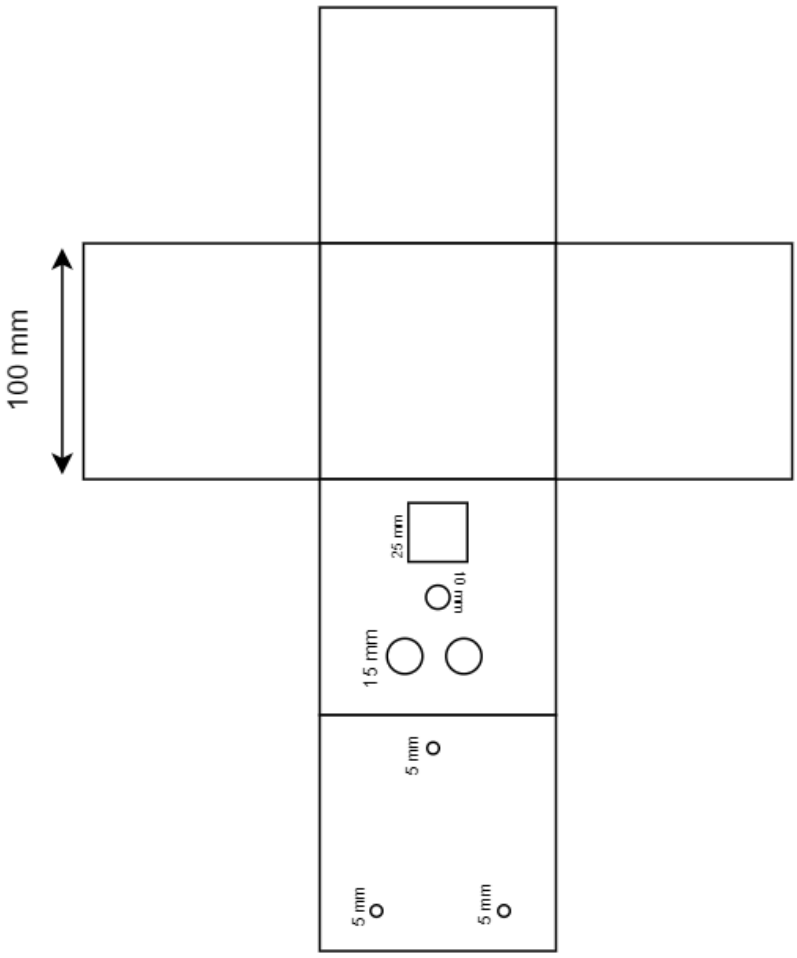

Ilustración 40 - Diseño de los cubos

A continuación se muestra una ilustración sobre el resultado final del diseño de los cubos con los sensores ya integrados.

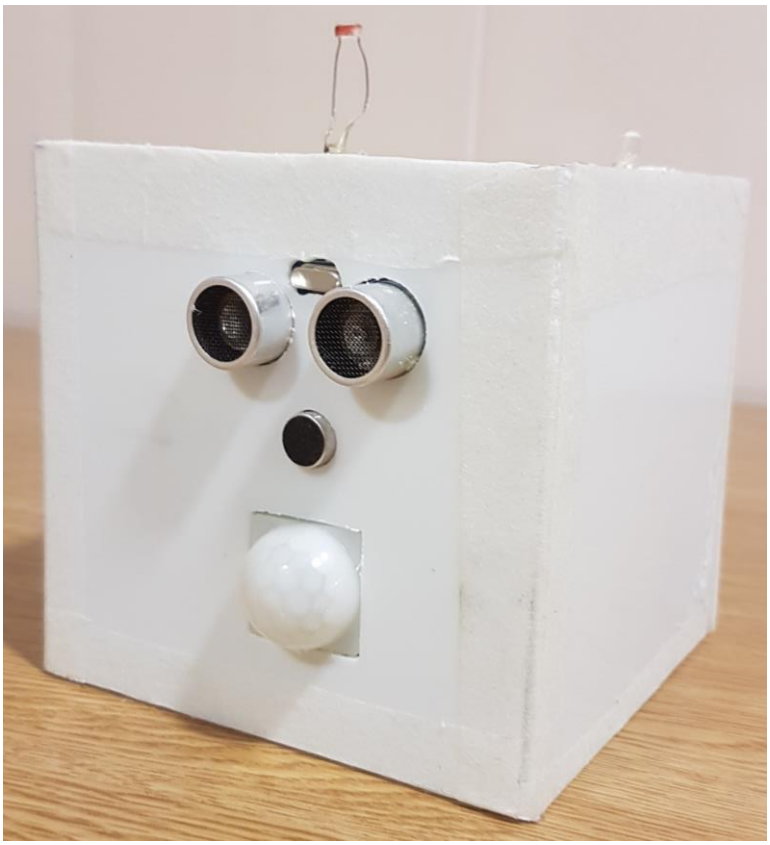

Ilustración 41 - Resultado diseño físico

5.2. Implementación lógica

A continuación se va a explicar de manera detallada la implementación lógica tanto del maestro como de los esclavos, protocolos de comunicación, librerías utilizadas y métodos aplicados para la consecución del objetivo.

Antes de comenzar con los detalles de la implementación se van a explicar que son los comandos *AT*, ya que son utilizados constantemente tanto por el maestro como por el esclavo para la configuración e inicialización del módulo *ESP8266*, el protocolo de comunicación *WiFi 802.11 b/g/n* y *TCP*.

El conjunto de comandos Hayes o *AT* son instrucciones codificadas que forman un lenguaje de comunicación entre personas y terminales modem, esta serie de comandos *AT* fueron desarrollados en 1977 por Dennys Hayes (fundador de Hayes Communications) como interfaz de comunicación para poder configurarlos y mandarles instrucciones, estos comandos se denominan así por la abreviatura de *attention*.

El estándar *IEEE 802.11* define el uso de los dos niveles inferiors de la arquitectura OSI, especificando reglas de funcionamiento de una red de área local inalámbrica. Lo que significa b/g/n es que es compatible con el *WiFi* estándar B, G y N [14].

- 802.11b: Funciona con conexiones de hasta 11mbps, opera en la banda de 2.4GHz.
- 802.11g: Funciona con conexiones de hasta 54mbps, opera en la banda de 2.4GHz.
- 802.11n: Funciona con conexiones de hasta 600mbps, opera en la banda de 2.4GHz y 5GHz.

Transporte orientado a la conexión (*TCP*) es un protocolo de la capa de transporte de Internet, fiable y orientado a la conexión. Se dice que *TCP* está orientado a la conexión porque antes de que un proceso de la capa de aplicación pueda comenzar a enviar datos a otro, ambos procesos primero tienen que establecer una comunicación. Una comunicación TCP casi siempre es conexión punto a punto, entre un único transmisor y receptor y una vez que se ha establecido la conexión, los dos procesos de aplicación pueden enviarse datos el uno al otro [15].

5.2.1. Implementación del maestro

El maestro es el encargado de organizar y dirigir el comportamiento de todo el sistema, a grandes rasgos se puede decir que su implementación se divide en tres grandes bloques, tiene dos modos de funcionamiento, activo y autónomo, en el modo activo el maestro recibe los datos de los sensores del esclavo y los procesa, en el modo autónomo, el esclavo no recibe datos por parte del esclavo, así que pasa a funcionar en modo autónomo, donde solo evalua sus sensores y actúa según estos.

MODO ACTIVO:

El primer bloque consiste en inicializar y configurar el módulo *ESP8266*, para ello se ha implementado un método llamado `inicializarEsp8266()` que hace uso de los comandos *AT* anteriormente definidos y que se van a mencionar a continuación con mas detalle, también se configura el servidor *TCP*.

1. En primer lugar se reinicia el modulo con el comando `AT+RST`, con este comando lo que se consigue es que si el modulo ya tenía una red guardada se vuelve a conectar, y se espera la respuesta, si `SerialEsp8266.find("ready")`, es decir, la respuesta es ready, el modulo estará listo.
2. Después el maestro se configurará como router asignandole al siguiente comando AT el número 2: `AT+CWMODE_DEF=2`, este comando configura el modo de operación y guarda la configuración en la

memoria Flash, es decir, la siguiente vez que se inicie el sistema, de manera automática iniciará este modo.

3. El siguiente paso es configurar el router con el comando `AT+CWSAP_DEF=\"luminaria\",\"Arduino\",5,2`, que establece la configuración de *ESP8266 soft-AP* y la guarda en la Flash, su lista de parámetros obligatorios es la siguiente:

    - <ssid>: Es el nombre del AP, en este caso es luminaria.
    - WPA-PSK <pwd>: Es la contraseña del punto de acceso, puede contener una longitude de 8 a 64 bytes ASCII, en este caso es Arduino.
    - <chl>: Id del canal, en este caso 5.
    - <ecn>: Método de encriptación, en este caso 2, no soporta WEP:

        - 0: OPEN
        - 2: WPA_PSK
        - 3: WPA2_PSK
        - 4: WPA_WPA2_PSK

En este caso el método de encriptación es una red *WiFi WPA-PSK* ("Wi-Fi Protected Access")-("Pre-shared key") dispone de una contraseña conocida por todos y cada uno de los clientes que se conectan a la red *WiFi*, es decir, la seguridad de la red se basa en un secreto compartido (la contraseña de la red), que conocen sus usuarios y el punto de acceso.

Para configurar el servidor TCP se ha implementado el método `configurarServidorTCP(int port)`

1. Se configura el router de manera que acepte multiples conexiones mediante el comando `AT+CIPMUX=1`, este comando acepta varios valores.

    - 0: Una única conexión.
    - 1: Múltiples conexiones.

2. Después se configura como Servidor *TCP* en el puerto que recibe el método por parámetros con el comando *AT* `AT+CIPSERVER=1,80`, el primer parámetro configura el router como servidor y el segundo es el puerto.
3. Para finalizar se configura el timeout del servidor con el comando `AT+CIPSTO=15`, en 15 segundos. Timeout significa que un servidor está tardando demasiado en responder a una solicitud de datos de otro dispositivo.

Después de todo esto, el router quedará a la espera de una conexión por parte de cualquier esclavo. Cabe añadir que no se configurará como servidor *DHCP* porque como ya se ha comentado, las IP's se asignarán de manera estática.

El segundo bloque de la implementación del maestro consiste en la lectura y procesado de los datos que reciben sus propios sensores.

1. Cálculo del sensor de distancia:

```
void iniciarTrigger() {
  // Ponemos el Triiger en estado bajo y esperamos 2 ms
  digitalWrite(PinTrig, LOW);
  delayMicroseconds(2);

  // Ponemos el pin Trigger a estado alto y esperamos 10 ms
  digitalWrite(PinTrig, HIGH);
  delayMicroseconds(10);

  // Comenzamos poniendo el pin Trigger en estado bajo
  digitalWrite(PinTrig, LOW);
}

double calculoDistancia() {

  // La función pulseIn obtiene el tiempo que tarda en cambiar entre
estados, en este caso a HIGH
  unsigned long tiempo = pulseIn(PinEcho, HIGH);

  // Obtenemos la distancia en cm, hay que convertir el tiempo en
segudos ya que está en microsegundos
  // por eso se multiplica por 0.000001
  distancia = tiempo * 0.000001 * VelSon / 2.0;

  return distancia;
}
```

2. Cálculo del sensor de sonido:

```
double calculoSonido()  {

  double valorTotal = 0;
  for (int i = 0; i < 6; i++) {
    double valorSound = medidorSonido();
    valorTotal = valorTotal + valorSound;
  }
  valorTotal = valorTotal / 15;

  return valorTotal;
}

double medidorSonido() {

  unsigned long startMillis = millis();
  unsigned int peakToPeak = 0;
  unsigned int signalMax = 0;
  unsigned int signalMin = 1024;
  // collect data for 50 mS
  while (millis() - startMillis < sampleWindow) {
    sample = analogRead(soundPin);
    if (sample < 1024)  {
```

```
      if (sample > signalMax)  {
        signalMax = sample;
      }
      else if (sample < signalMin) {
        signalMin = sample;
      }
    }
  }
  peakToPeak = signalMax - signalMin;
  double volts = (peakToPeak * 5.0) / 1024;

  return volts;

}
```

3. Cálculo de los sensores de sonido y movimiento:

```
double filterData(double sensor, String sensorName) {
  if (sensorName == "sensorMovimiento") {
    //Media para la medicion de detecciones
    for (int i = 0; i < deteccionesParaMedia; ++i)
    {
      mediaMedicion += digitalRead(sensor);
      delay(7);
    }
    mediaMedicion /= deteccionesParaMedia;

    if (mediaMedicion > 0.3) {
      valorSensor = HIGH;
    } else {
      valorSensor = LOW;
    }
  } else if (sensorName == "sensorLuz") {
    //Media para la medicion de detecciones
    for (int i = 0; i < deteccionesParaMedia; ++i)
    {
      mediaMedicion += analogRead(sensor);
      delay(7);
    }
    mediaMedicion /= deteccionesParaMedia;
    if (mediaMedicion > umbralLuz) {
      valorSensor = LOW;
    } else {
      valorSensor = HIGH;
    }

  }

  return valorSensor;

}
```

4. Una vez se han hecho los cálculos de los valores obtenidos por los sensores se crean las variables que se pasarán después al método `intensidadLuminaria(valorLuz, valorMovimiento, valorDistancia, valorSonido)` que evaluará si se cumplen las condiciones para encender la lámpara

```
valorDistancia = calculoDistancia();
valorLuz = filterData(LDRPin, "sensorLuz");
valorMovimiento = filterData(sensorMov, "sensorMovimiento");
valorSonido = calculoSonido();
```

Éste método evalua si se dan las condiciones necesarias para que se encienda la lámpara mediante el algoritmo uno y por simplicidad los valores devuelven 0 si debe estar apagada o 2 si se enciende a la intensidad más alta. Las comprobaciones tienen la misma estructura que se muestra a continuación:

```
int intensidadLuminaria(int lightIntens, int movement, double
proximity, double soundIntens) {
  int intens = 2;
  if (movement == HIGH && proximity < umbralProx && lightIntens ==
HIGH && soundIntens > umbralSon) {
    intens = 2;
  }
```

A continuación se muestra la tabla de verdad del algoritmo uno, donde se pueden apreciar todas las condiciones de encendido o apagado.

Luz	Proximidad	Luz	Movimiento	Sonido
0	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1

Tabla 13 - Tabla de verdad algoritmo 1

5. Por último el valor devuelto por el método intensidad se asigna a una variable `intensidadLuz` que se evaluará para decidir si la lámpara se enciende o no y con que intensidad.

```
intensidadLuz = intensidadLuminaria(valorLuz, valorMovimiento,
valorDistancia, valorSonido);
  if (intensidadLuz == 0 && stateSlave == 2) {
        lightStateMaster = 1;
        digitalWrite(pinLed, lightStateMaster);
        digitalWrite(pinLedDos, LOW);
  } else if (intensidadLuz == 2) {
        lightStateMaster = 1;
        digitalWrite(pinLed, lightStateMaster);
        digitalWrite(pinLedDos, lightStateMaster);
  } else if (intensidadLuz == 0) {
        lightStateMaster = 0;
        digitalWrite(pinLed, lightStateMaster);
        digitalWrite(pinLedDos, lightStateMaster);
}
```

El tercer y último bloque de la implementación del maestro consiste en la recepción y procesado de los datos que el esclavo manda, y el envío de datos evaluados desde el maestro al esclavo. Previo al envío se evalua si se dan las condiciones necesarias para que se encienda la lámpara según lo que reciba la variable `intensidadLuz`, del método `intensidadLuminaria(valorLuz, valorMovimiento, valorDistancia, valorSonido);` por simplicidad los valores son: 0 si debe estar apagada o 1 si debe encenderse la lámpara, sin embargo solo se guarda el estado de la variable `intensidadLuz` no se ejecuta la acción de encendido o apagado.

```
intensidadLuz = intensidadLuminaria(valorLuz, valorMovimiento,
valorDistancia, valorSonido);
  if (intensidadLuz == 0) {
    //LUZ APAGADA
    lightStateMaster = 0;
  } else if (intensidadLuz == 2) {
    //LUZ ENCENDIDA
    lightStateMaster = 1;
}
```

Para el envío y la recepción se hace uso del método `void recibirMensaje()` y procede de la siguiente manera:

1. En primer lugar se comprueba que hayan datos del ESP8266, mediante la orden `while (!SerialEsp8266.find("+IPD,"))`, en caso de que hayan datos se leen línea a línea y se guardan en la variable message.
2. En la variable message se guarda un string recibido con los valores de los sensores, para el correcto procesado es necesario descomponer ese string valor a valor, se ha hecho uso del método `indexOf(",")` para en este caso, separarlo por comas y después se ha usado el

método `toInt()` para pasarlo de string a int. Por simplicidad, se han transformado los valores a 1 o 0 según si superan el umbral necesario apagado. Con las variables `booleanDist`, `booleanLum`, `booleanMov`, `booleanSon`, `stateSlave` y `lightStateMaster` se ha programado el algoritmo uno de estados que se mostrará a continuación mediante una tabla de verdad.

INTENSIDAD LUZ		SENSORES				
Intensidad	Luz	booleanProx	booleanLuz	booleanMov	booleanSon	stateSlave || lightStateMaster
0%	0	0	0	0	0	0
50%	1	0	0	0	0	1
100%	1	0	0	0	1	0
100%	1	0	0	0	1	1
100%	1	0	0	1	0	0
100%	1	0	0	1	0	1
100%	1	0	0	1	1	0
100%	1	0	0	1	1	1
0%	0	0	1	0	0	0
0%	0	0	1	0	0	1
0%	0	0	1	0	1	0
0%	0	0	1	0	1	1
0%	0	0	1	1	0	0
0%	0	0	1	1	0	1
0%	0	0	1	1	1	0
0%	0	0	1	1	1	1
100%	1	1	0	0	0	0
100%	1	1	0	0	0	1
100%	1	1	0	0	1	0
100%	1	1	0	0	1	1
100%	1	1	0	1	0	0
100%	1	1	0	1	0	1
100%	1	1	0	1	1	0
100%	1	1	0	1	1	1
0%	0	1	1	0	0	0
0%	0	1	1	0	0	1
0%	0	1	1	0	1	0
0%	0	1	1	0	1	1
0%	0	1	1	1	0	0
0%	0	1	1	1	0	1
0%	0	1	1	1	1	0
0%	0	1	1	1	1	1

Tabla 14 - Tabla de verdad algoritmo 2

Se debe tener en cuenta que el maestro debe evaluar también si el esclavo 2 o él mismo está encendido antes de contestar al esclavo 1 porque aunque el esclavo 1 no detecte nada, el esclavo 2 o el maestro si, lo que es condición necesaria para encender la lámpara del esclavo 1 a media intensidad y crear un ambiente. Para ello se evalua que esclavo ha mandado el mensaje y se utiliza una variable `stateSlave` a la que se le asigna el estado del esclavo contrario, y la variable `lightStateMaster` con el estado del maestro. Con todo esto, la forma de evaluar si una lámpara esclavo debe encenderse o no y con cuanta intensidad es la siguiente:

```
if (booleanDist == 0 && booleanLum == 0 && booleanMov == 0 &&
booleanSon == 0 && (stateSlave == 2 || lightStateMaster == 1)) {
    messageForSlave = "1";
 }
```

Por último se debe guarder el estado del esclavo en cuestión (encendido o apagado), para evaluar si el siguiente esclavo debe encenderse o no.

3. Una vez se han procesado los datos y se ha creado la variable respuesta para el esclavo, `recibirMensaje()` llama al método `conectarConPasarelaYEnviarMensaje(id, PORT_PASARELA, messageForSlave, 4)` el primer argumento es la id del esclavo, dirección IP del esclavo para saber a quien se debe contester, el segundo es el Puerto, el tercero es el resultado del procesamiento de datos y el cuarto es el identificador de la conexión.
4. Para empezar, en este método se prepara el comando con la petición para conectar con la pasarela vía *TCP* mediante el comando `AT+CIPSTART` cuyos parámetros son:

    - Id de la conexión, un entero del 0 al 7.
    - Tipo de conexión *TCP* o *UDP*: En este caso la conexión es *TCP*.
    - IP del servidor en la pasarela.
    - Puerto del servidor en la pasarela.

    Se envía el comando al *ESP8266* y se espera la respuesta, que se comprueba con `SerialEsp8266.find("OK")`.
5. Si la respuesta es la esperada se establece la conexión y el método `conectarConPasarelaYEnviarMensaje(id, PORT_PASARELA, messageForSlave, 4)` llama a `enviarMensaje(mensaje, id_conexión)`.

6. Este método prepara la petición del mensaje a enviar con el comando `AT+CIPSEND` con los siguientes parámetros:

- Id de la conexión.
- Longitud de los datos a enviar, con un máximo de 2048 bytes.

Se realiza la petición de envío y se comprueba la respuesta del ESP8266, si se obtiene la respuesta esperada, se envía el mensaje que corresponde al esclavo.

MODO PASIVO:

De la misma manera que en el modo pasivo se inicializa y configura el módulo *ESP8266* con el método `inicializarEsp8266()`, se configura el servidor *TCP* con el método `configurarServidorTCP(int port)`.

La diferencia está en el envío y la recepción, cuando se llama al método `void recibirMensaje()`, el maestro en el modo pasivo no recibirá datos algunos, por lo que su funcionamiento se basará en sus propios sensores. Para ello hará una llamada al método `loop()`:

1. Consiste en un bucle infinito que evalua constantemente las siguientes variables:

```
valorDistancia = calculoDistancia();
valorLuz = filterData(LDRPin, "sensorLuz");
valorMovimiento = filterData(sensorMov, "sensorMovimiento");
valorSonido = calculoSonido();
```

2. Se hace uso de estas variables para extraer el valor de la variable `intensidadLuz`, variable que decide si la luz se ha de encender o no

```
void calculoSensores(){
  do {
   intensidadLuz = intensidadLuminaria(valorLuz,
valorMovimiento, valorDistancia, valorSonido);
  if (intensidadLuz == 0) {
    //LUZ APAGADA
    lightState = 0;
    digitalWrite(pinLed, lightState);
    digitalWrite(pinLedDos, lightState);
  } else if (intensidadLuz == 2) {
    //LUZ ENCENDIDA
    lightState = 1;
    digitalWrite(pinLed, lightState);
    digitalWrite(pinLedDos, lightState);
  }

  } while(true);
}
```

El método `intensidadLuminaria(valorLuz, valorMovimiento, valorDistancia, valorSonido);` da valor a `intensidadLuz` para ello se ha hecho

uso del algoritmo uno el cual se ha visto su tabla de verdad anteriormente, con lo que las comprobaciones, de manera general, quedarían de la siguiente manera:

```
  if (movement == HIGH && proximity < umbralProx && lightIntens ==
HIGH && soundIntens > umbralSon){
     intens = 2;
```

5.2.2. Implementación del esclavo

El esclavo igual que el maestro tiene también dos modos de funcionamiento, pasivo y autónomo, en el modo pasivo se encarga de leer los datos que recibe por sus sensores, con la diferencia con el maestro de que no los procesa, se los manda para que él decida que debe ocurrir. De la misma manera que el maestro, la implementación del esclavo también se divide en tres grandes bloques, con una funcionalidad similar, por ello se van a comentar las diferencias más significativas. El esclavo entrará en modo autónomo cuando no detecte conexión con el maestro, es decir, abortará todo tipo de conexión y se encenderá en función de lo que sus sensores le indiquen.

MODO PASIVO:

El modulo ESP8266 también tiene que ser inicializado en el esclavo, para ello se hará uso del método `inicializarEsp8266()`, las diferencias con el maestro son las siguientes:

1. El esclavo se configurará como host asignandole al comando *AT* `AT+CWMODE_DEF` el número 2, de la misma manera que en el maestro, este comando configura el modo de operación y guarda la configuración en la memoria Flash, es decir, la siguiente vez que se inicie el sistema, de manera automática iniciará este modo.

También hace uso del método `configurarServidorTCP(int port)`, sin embargo no se va a comentar nada ya que la configuración se realiza exáctamente igual que en el maestro.

La diferencia mas significativa que hay entre el maestro y el esclavo es el método `conectarConRouterWifi(String ID_Router, String password)` y se configura de la siguiente manera:

1. En primer lugar se le indica a que AP se debe conectar con el comando `AT+CWJAP_CUR=\""+ ID_Router + "\",\"" + password +"\"` siendo el ID y la pass los parámetros que recibe el método, el resultado de este comando se guarda en una variable llamada cmd.
2. Después se envía el comando al ESP8266 mediante la orden `SerialEsp8266.println(cmd)`.

3. Por último se espera a que la respuesta llegue al buffer, se comprueba si ha llegado con la instrucción `if (SerialEsp8266.find("OK"))`, si se ha recibido el OK el *ESP8266* se ha conectado correctamente al AP.

El segundo bloque de la implementación del esclavo también consiste en la lectura de los datos que reciben sus propios sensores, sin embargo no los procesa, forma un mensje con esos datos como se va a ver a continuación:

1. La lectura de los valores de los sensores se hace de la misma manera que en el maestro.
2. Una vez hechos los cálculos, se crean también las variables mencionadas anteriormente

```
valorDistancia = calculoDistancia();
valorLuz = filterData(LDRPin, "sensorLuz");
valorMovimiento = filterData(sensorMov, "sensorMovimiento");
valorSonido = calculoSonido();
```

3. Con estas variables, en lugar de evaluar si las luces se deben encender o no, el esclavo creará un mensaje para enviar al maestro con sus valores y lo guardará en la variable Message.

```
  String Message = "192.168.4.2, ";
  Message += valorDistancia;
  Message += ", ";
  Message += valorLuz;
  Message += ", ";
  Message += valorMovimiento;
  Message += ", ";
  Message += valorSonido;
```

4. Ésta variable Message será la que pase al método `conectarConPasarelaYEnviarMensaje(IP_PASARELA, PORT_PASARELA, Message, 4)`

El tercer y último bloque tendra la misma funcionalidad que el bloque del maestro, envío y recepción de mensajes, pero es el esclavo quién comienza la comunicación y después queda a la espera de la respuesta. Al contrario de lo que sucedia con el maestro que para empezar la comunicación hace uso del método `recibirMensaje()`. Este último bloque de la implementación del esclavo, hace uso en primer lugar del método `conectarConPasarelaYEnviarMensaje(IP_PASARELA, PORT_PASARELA, Message, 4)`, el cual procede de manera muy similar al mismo método usado por el maestro por lo que solo se van a comentar las diferencias más significativas.

1. El comando AT con la petición para conectar con la pasarela via TCP es el mismo y se configura de la misma manera.
2. A continuación se llama al método `if(enviarMensaje(mensaje, id_conexión))`, se hace de esta manera porque si el mensaje se

envía, se llama al método *recibirMensaje()*. Los métodos `enviarMensaje()` y `recibirMensaje()` proceden de manera similar en el maestro y en el esclavo, la diferencia entre el maestro y el esclavo respect al método `recibirMensaje()` es que el maestro como ya se ha mencionado procesa los datos recibidos y el esclavo recibe ordenes directas sobre si se debe encender la luz o no.

```
if (S == "2") {
        lightState = 1;
        digitalWrite(pinLed, lightState);
        digitalWrite(pinLedDos, lightState);
      } else if (S == "1") {
        lightState = 1;
        digitalWrite(pinLed, lightState);
        digitalWrite(pinLed, LOW);
      } else if (S == "0") {
        lightState = 0;
        digitalWrite(pinLed, lightState);
        digitalWrite(pinLed, lightState);
      }
```

Donde S representa el mensaje de tipo String que el maestro le manda al esclavo, el esclavo lo único que tiene que hacer es encender la lámpara o no según la orden que ha recibido

El orden de las llamadas a los métodos varían debido a que el maestro debe recibir para enviar y esclavo debe enviar para recibir, cabe mencionar que no habrá método de cierre de conexión porque no tendría sentido para el funcionamiento de este sistema.

MODO ACTIVO:

1. De manera normal, se ejecutará el método `inicializarEsp8266()`.
2. De manera normal se ejecuta el método `conectarConRouterWifi(SSID, PASS);` en este punto, cuando haga varios intentos de conexión con el maestro y todos sean fallidos, entrará en modo activo, y hará una llamada al método `loop()`, el cual consiste en un bucle infinito que calcula las variables:

```
valorDistancia = calculoDistancia();
valorLuz = filterData(LDRPin, "sensorLuz");
valorMovimiento = filterData(sensorMov, "sensorMovimiento");
valorSonido = calculoSonido();
```

Evalua si se ha de encender la lámpara o no:

```
void calculoSensores(){
  do {

    intensidadLuz = intensidadLuminaria(valorLuz, valorMovimiento,
valorDistancia, valorSonido);
  if (intensidadLuz == 0) {
    //LUZ APAGADA
    lightState = 0;
    digitalWrite(pinLed, lightState);
    digitalWrite(pinLedDos, lightState);
  } else if (intensidadLuz == 2) {
    //LUZ ENCENDIDA
    lightState = 1;
    digitalWrite(pinLed, lightState);
    digitalWrite(pinLedDos, lightState);
  }

  } while(true);
}
```

El valor de `intensidadLuz` se extrae haciendo una llamada al metodo `intensidadLuminaria(valorLuz, valorMovimiento, valorDistancia, valorSonido);` y para ello se ha hecho uso del algoritmo uno el cual se ha visto su tabla de verdad anteriormente, con lo que las comprobaciones, de manera general, quedarían de la siguiente manera:

```
  if (movement == HIGH && proximity < umbralProx && lightIntens ==
HIGH && soundIntens > umbralSon){
     intens = 2;
}
```

6. Pruebas

Para validar el presente proyecto se posicionarán los diferentes cubos distribuidos en el pasillo de un hogar. Una vez colocados los cubos se realizarán dos tipos de pruebas:

- Modo autónomo.
- Modo activo/pasivo.

El modo autónomo es el escenario en el cual, cada cubo funcionará de manera independiente a los demás. Para ello no se conectará al maestro, así los esclavos no encontrarán conexión y entrarán en modo autónomo. Una vez los esclavos están en modo autónomo, se conectará también al maestro para ver el funcionamiento de los tres dispositivos. Para hacer éste tipo de prueba se pueden dar los siguientes casos:

- Se conectarán los esclavos más tarde que el maestro, para que éste no reciba datos y entre en modo autónomo, así funcionarán los 3 de manera autónoma.
- Después se actuará de manera contraria, primero se conectarán los esclavos para comprobar que no detectan al maestro y entrarán así en modo autónomo, más tarde se conectará también al maestro.

El modo activo pertenece al maestro, se refiere a la forma de proceder cuando los esclavos detectan su conexión y le mandan mensajes, él los procesa y responde. El modo pasivo pertenece a los esclavos, se refiere a la forma de proceder cuando establecen conexión con el maestro y le mandan sus datos para recibir la orden que proceda. Y si todo funciona correctamente los cubos deben proceder como en la imagen que se muestra a continuación.

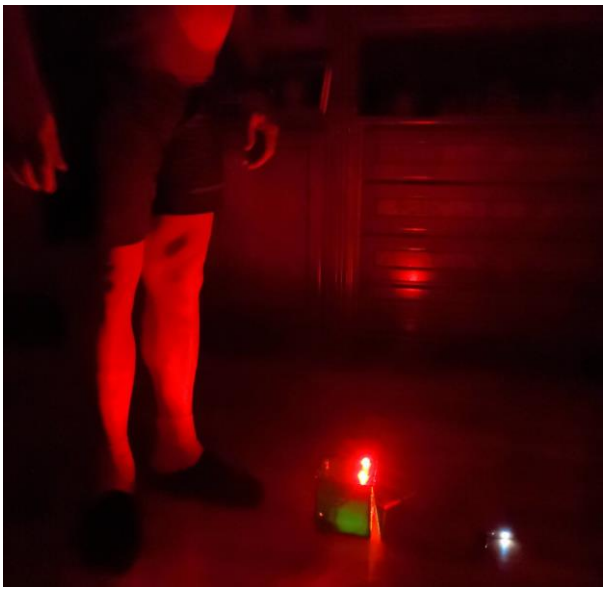

Ilustración 42 - Funcionamiento cubo

Para la realización de las pruebas se conectarán los 3 cubos a la luz con el software que corresponda cargado, una vez los cubos entren en funcionamiento, entrarán al hogar una o diversas personas actuando de manera habitual para probar los diferentes modos. El escenario de prueba será el siguiente.

Ilustración 43 - Escenario de pruebas

7. Conclusiones

Todo proyecto consiste en la consecución de ciertos objetivos. Se puede considerar que este trabajo los ha alcanzado de manera satisfactoria y a partir de ellos se han extraído ciertas conclusiones.

El objetivo principal de este proyecto consiste en crear un ambientación lumínica autónoma e inteligente que sea capaz de provocar en el usuario, una sensación de confort y seguridad.

El primer objetivo consistía en la lectura por parte de los sensores de los datos del entorno y seguidamente el procesado para evaluar si se ha de encender la luz o no de manera automática. Este objetivo se ha cumplido satisfactoriamente, lo que convierte a este sistema en inteligente.

El segundo objetivo consistía en establecer un algoritmo de comunicación mediante el protocolo *WiFi 802.11 b/g/n*, para que los *Arduinos* pudieran estar en continua comunicación. Este objetivo ha presentado ciertas dificultades a la hora de la sincronización de los mensajes entre maestro y esclavos, sin embargo se ha cumplido consiguiendo así crear un escenario de confort.

Una de las desventajas que presenta el hecho de hacerlo con placas Arduino es que la capacidad de cómputo es pequeña y además no puede realizar tareas multihilo, aunque el algoritmo responde y cumple con los objetivos establecidos, el tiempo de respuesta podría disminuir. Una posible alternativa sería hacerlo con placas *Raspberry Pi* porque tienen una capacidad de cómputo mayor, son multiprocesador, llevan *WiFi* y *Bluetooth* integrados. Sin embargo su coste es más elevado que el de las placas *Arduino*, por lo que habría que estudiar una relación de mejora del tiempo de respuesta del algoritmo con el precio que supondría cambiarlas.

Por último mencionar que tanto el algoritmo de comunicación como el procesado de datos se ha realizado en el lenguaje de programación *C++,* lo cual aporta diversas ventajas como la portabilidad ya que es posible compilar el código en diferentes plataformas, es un lenguaje formativo ya que se pueden aprender muchos otros a partir de él con facilidad como *C#, Java, Javascript*, entre otros, es un lenguaje robusto y versátil, su código es eficiente y existe gran cantidad de documentación.

8. Trabajos futuros

Un proyecto de investigación como el que aquí se presenta, ofrece una solucion sobre una temática, sin embargo, podría tener diversas ampliaciones ya que cualquier aspecto sobre esta materia está abierto a progresos.

En este trabajo se han resuelto ciertas cuestiones, pero se han dejado otras preparadas para futuros proyectos que a continuación se van a comentar:

- Añadir un motor y un módulo *WiFi* a las persianas del hogar para que, cuando los sensores capten los valores y se den las condiciones necesarias, suban o bajen automáticamente.

- Integración del código del proyecto con la *API* de *Echo Amazon Alexa* para que además de ser un sistema autónomo, se pueda gestionar la iluminación del hogar por control de voz.

- Inserción de un sensor de temperatura y un led infrarrojo para que cuando el sistema detecte una temperatura por debajo o por encima de un umbral establecido, encienda el aire acondicionado automáticamente.

- Ampliación del sistema con una cámara de vídeo vigilancia que actúe de manera contraria, cuando no detecte presencia en el hogar, comience a grabar las estancias y después guarde las grabaciones en un repositorio al que solo tenga acceso el usuario.

- Inserción de un segundo sensor de ultrasonido que permita detectar el nivel de agua, cuando detecte un umbral se cerrará el grifo, por ejemplo en bañeras, fregaderos, etc.. de esta manera el usuario podrá realizar otras tareas sin miedo a que el recipiente en cuestión desborde.

- Ampliación del sistema con un seguidor solar y un nuevo fotorresistor que detecte donde hay más luz solar durante el día para aumentar la eficiencia en la captación de energía solar y poder realizar un autoconsumo.

- Ampliación del sistema con un zumbador y un nuevo sensor PIR que detecte movimiento cuando el usuario lo establezca, de esta manera cuando el PIR detecte un movimiento el zumbador comenzará a sonar avisando al usuario de la situación.

9. Referencias

[1] J. HILL, 12 9 2015. [Online]. Available: https://www.t3.com/features/the-smart-home-guide.

[2] M. Rinaldi, 26 6 2013. [Online]. Available: https://www.instalia.eu/historia_de_la_iluminacion_escenica_4270/.

[3] C. domótica, 9 4 2016. [Online]. Available: https://www.casasdigitales.com/iluminacion-domotica-controla-regula-la-iluminacion-hogar/.

[4] R. Garrido, 17 7 2014. [Online]. Available: https://www.xatakahome.com/domotica/jibo-el-robot-que-pretende-ser-el-acompanante-perfecto-para-nuestro-hogar-inteligente.

[5] P. Santamaria, 24 7 2014. [Online]. Available: https://www.xataka.com/domotica-1/domotica-y-el-futuro-que-siempre-esta-llegando.

[6] E. R. d. Luis, 13 5 2019. [Online]. Available: https://www.xataka.com/seleccion/que-me-gusta-mi-amazon-echo-google-home-homepod.

[7] Amazon, 1 9 2017. [Online]. Available: https://www.amazon.es/Philips-Hue-White-Color-Ambiance.

[8] K. Jenkins, 27 3 2019. [Online]. Available: https://prezi.com/p/nmfmdyujsirz/nanoleaf-presentation/.

[9] M. López, 27 8 2018. [Online]. Available: https://www.applesfera.com/accesorios/ikea-ampliara-su-gama-dispositivos-inteligentes-tradfri-enchufe-inteligente-10-euros.

[10] S. Navas, 19 4 2018. [Online]. Available: https://isenacode.com/irobot-roomba-896-analisis-del-robot-aspirador-mas-geek/.

[11] L. Llamas, 24 7 2015. [Online]. Available: https://www.luisllamas.es/detector-de-movimiento-con-arduino-y-sensor-pir/.

[12] L. Llamas, 16 6 2015. [Online]. Available: https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/.

[13] L. Llamas, 16 3 2015. [Online]. Available: https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/.

[14] E. Álvarez, 27 3 2018. [Online]. Available: https://computerhoy.com/noticias/internet/diccionario-del-wifi-80211-bgn-banda-dual-5-ghz-78159.

[15] R. Kurose, "Redes de computadores," in *Un enfoque diferente*, Pearson, 2010, pp. 228-249.

Anexo

Comandos AT:

[1] S. S. W. S. Ltd., 2012. [Online]. Available: http://www.mt-system.ru/sites/default/files/simcom_sim5320_atc_en_v1.23.pdf.

Documentación Arduino:

[2] R. E. Herrador, 13 11 2009. [Online]. Available: http://www.uco.es/aulasoftwarelibre/wp-content/uploads/2010/05/Arduino_user_manual_es.pdf.

Código del maestro:

```
#include <SoftwareSerial.h>
#include <Servo.h>
SoftwareSerial SerialEsp8266(3, 2); // RX, TX

#define IP_PASARELA "192.168.4.2" //Ordenador pasarela
#define PORT_PASARELA "80"

int lightStateMaster;
int contadorAutonomo = 0;

int idnodo = 1;
String mi_IP = "";
String idGlobal = "";
boolean conecionEstablecida = false;
boolean controlDeFlujo = false;


//INICIALIZACIÓN VARIABLES
const int LDRPin = A0;
int valorLuz = 0;

int pinLed = 12;
int pinLedDos = 7;
int intensidadLuz = 1;

const int PinTrig = 9;
const int PinEcho = 8;
const float VelSon = 34000.0;
float distancia;

int soundPin = A1;
const int sampleWindow = 50; // Sample window width in mS (50 mS = 20Hz)
unsigned int sample;
double valorSonido = 0.0;

int sensorMov = 11;
int valorSensor = 0;
int valorMovimiento = 0;

float mediaMedicion = 0;
int deteccionesParaMedia = 50;
double valorDistancia = 0.0;
```

```
const int umbralLuz = 35;
const double umbralSon = 3.0;
const double umbralProx = 50.0;

int stateSlave = 0;
int stateSlaveOne = 0;
int stateSlaveTwo = 0;
String messageForSlave = "";

//CIERRE INICIALIZACIÓN VARIABLES


void setup() {
  //Para conectar con el modulo esp8266
  SerialEsp8266.begin(9600);
  //Para conectar con el modulo serie
  Serial.begin(9600);

  inicializarEsp8266();
  configurarServidorTCP(80);

  //VARIABLES SENSORES
  // Ponemos el pin LDR en modo entrada
  pinMode(LDRPin, INPUT);
  // Ponemos el pin Trig en modo salida
  pinMode(PinTrig, OUTPUT);
  // Ponemos el pin Echo en modo entrada
  pinMode(PinEcho, INPUT);
  // Ponemos el pin sensorMov en modo entrada
  pinMode(sensorMov, INPUT);
  // Ponemos el pin Led en modo salida
  pinMode(pinLed, OUTPUT);
  // Ponemos el pin Led en modo salida
  pinMode(pinLedDos, OUTPUT);
  //FIN VARIABLES SENSORES
  // La función pulseIn obtiene el tiempo que tarda en cambiar entre estados, en este
caso a HIGH

}



//BUCLE DE EJECUCION
void loop() {

  //Calculos


  iniciarTrigger();

  valorDistancia = calculoDistancia();
  Serial.print("PROXIMITI ");
  Serial.println(valorDistancia);

  valorLuz = filterData(LDRPin, "sensorLuz");
  Serial.print("LUZ ");
  Serial.println(valorLuz);

  valorMovimiento = filterData(sensorMov, "sensorMovimiento");
  Serial.print("MOVIMIENTO ");
  Serial.println(valorMovimiento);

  valorSonido = calculoSonido();
  Serial.print("SONIDO: ");
  Serial.println(valorSonido);

  //Cierre calculos

  if(contadorAutonomo >= 35 ){
    Serial.println("Modo autonomo");
    do{
  Serial.println("-----------------------------------------------");
      iniciarTrigger();

  valorDistancia = calculoDistancia();
```

```
  Serial.print("PROXIMITI ");
  Serial.println(valorDistancia);

  valorLuz = filterData(LDRPin, "sensorLuz");
  Serial.print("LUZ ");
  Serial.println(valorLuz);

  valorMovimiento = filterData(sensorMov, "sensorMovimiento");
  Serial.print("MOVIMIENTO ");
  Serial.println(valorMovimiento);

  valorSonido = calculoSonido();
  Serial.print("SONIDO: ");
  Serial.println(valorSonido);

   intensidadLuz = intensidadLuminaria(valorLuz, valorMovimiento, valorDistancia,
valorSonido);
if (intensidadLuz == 2) {
    lightStateMaster = 1;
    digitalWrite(pinLed, lightStateMaster);
    digitalWrite(pinLedDos, lightStateMaster);
  } else if (intensidadLuz == 0) {
    lightStateMaster = 0;
    digitalWrite(pinLed, lightStateMaster);
    digitalWrite(pinLedDos, lightStateMaster);
  }

    } while(true);

  }

  //ACCION
  intensidadLuz = intensidadLuminaria(valorLuz, valorMovimiento, valorDistancia,
valorSonido);
  if (intensidadLuz == 0) {
    //LUZ APAGADA
    lightStateMaster = 0;
    // digitalWrite(pinLed, lightStateMaster);
  } else if (intensidadLuz == 2) {
    //LUZ ENCENDIDA
    lightStateMaster = 1;
    // digitalWrite(pinLed, lightStateMaster);
  }

  controlDeFlujo = true;

  recibirMensaje();
  //Serial.println("El metodo recibir mensaje esta atascado");

  intensidadLuz = intensidadLuminaria(valorLuz, valorMovimiento, valorDistancia,
valorSonido);
  if (intensidadLuz == 0 && stateSlave == 2) {
    lightStateMaster = 1;
    digitalWrite(pinLed, lightStateMaster);
    digitalWrite(pinLedDos, LOW);
  } else if (intensidadLuz == 2) {
    lightStateMaster = 1;
    digitalWrite(pinLed, lightStateMaster);
    digitalWrite(pinLedDos, lightStateMaster);
  } else if (intensidadLuz == 0) {
    lightStateMaster = 0;
    digitalWrite(pinLed, lightStateMaster);
    digitalWrite(pinLedDos, lightStateMaster);
  }


}

// despues de recibir llamada a mis sensores e

boolean inicializarEsp8266() {

  do { //Se reinicia el modulo esp8266
    Serial.print("Reseteando modulo esp8266");
    SerialEsp8266.println("AT+RST"); //Se envía el comando correspondiente al ESP8266
    do {
      delay(100);//Se espera contestacion del ESP8266
```

```
    } while (!SerialEsp8266.available());
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("ready")) {
      Serial.println("... Modulo preparado");
      break;
    } else {
      Serial.println("... ERROR");
    };
  } while (true);

  /**/
  do { //Se configura como modo 2: SOFTAP (router)
    Serial.print("Estableciendo modo 2");
    SerialEsp8266.println("AT+CWMODE_DEF=2"); //Se envía el comando correspondiente al
ESP8266
    do {
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK")) {
      Serial.println("... Modo 2 soft-AP (router) establecido");
      break;
    } else {
      Serial.println("... ERROR");
    };
  } while (true);

  do { //Se configura el router
    Serial.print("Configurando el router luminaria cerrado");
    SerialEsp8266.println("AT+CWSAP_DEF=\"luminaria\",\"Arduino\",5,0"); //Se envía el
comando correspondiente al ESP8266
    do {
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK")) {
      Serial.println("... router configurado");
      break;
    } else {
      Serial.println("... ERROR");
    };
  } while (true);
  /**/

}

boolean configurarServidorTCP(int port) {
  do { //Aceptar multiples conexiones
    Serial.print("Estableciendo aceptar multiples conexiones");
    SerialEsp8266.println("AT+CIPMUX=1");//Se envía el comando correspondiente al
ESP8266
    do {
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK")) { //Si ya está configurado devuelve "Link is builded"
      Serial.println("... establecido");
      break;
    } else {
      Serial.println("... ERROR");
    };
  } while (true);

  do { //Configurar como Server TCP en el puerto "port"
    Serial.print("Configurando servidor TCP en puerto: " + String(port));
    String cmd = "AT+CIPSERVER=1,";
    cmd += String(port);
    SerialEsp8266.println(cmd);//Se envía el comando correspondiente al ESP8266
    //Con SerialEsp8266.println("AT+CIPSERVER=1,"+port) NO FUNCIONA
    do {
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK")) {
      Serial.println("... configurado.");
      break;
    } else {
```

```
      Serial.println("... ERROR");
    };
  } while (true);

  do { //Configurar el timeout
    Serial.print("Configurando el timeout del server...");
    String cmd = "AT+CIPSTO=15";
    SerialEsp8266.println(cmd);//Se envía el comando correspondiente al ESP8266
    do {
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK")) {
      Serial.println("... configurado.");
      break;
    } else {
      Serial.println("... ERROR");
    };
  } while (true);

  Serial.println("Esperando conexion...");
}

void recibirMensaje() {
  if (SerialEsp8266.available() > 0 || conecionEstablecida) { //Ver si hay mensaje del
ESP8266
    while (!SerialEsp8266.find("+IPD,")) {
      contadorAutonomo++;
      Serial.println(contadorAutonomo);
      Serial.println("==========> ESPERANDO MSJ ==========>");
      if (contadorAutonomo >= 35) {
        loop();
      }

      delay(100);//Se espera contestacion del ESP8266

    }
    contadorAutonomo = 0;
    //if (SerialEsp8266.find("+IPD,")) { //Ver si el servidor ha recibido datos*/
    Serial.println("################## VEO QUE HAY IPD");
    String Message = SerialEsp8266.readStringUntil('\r'); //Lee los datos hasta que se
encuentra el carriage return character (ASCII 13, or '\r'). El caracter buscado no se
devuelve en la cadena resultado.
    Serial.println("==========> RECIBIDO: " + Message);


    messageForSlave = procesarDatos(Message);

    while (SerialEsp8266.available() && controlDeFlujo) { //Si hubiera mas lineas, no
deberia si el cliente cumple el formato que se ha decidido, se leen para vaciar el
buffer.
      SerialEsp8266.read(); //c = SerialEsp8266.read();
      Serial.println("====== DATOS CONECTAR PASARELA ANTES DE ENVIAR ========");
      Serial.println(IP_PASARELA);
      Serial.println(PORT_PASARELA);
      conectarConPasarelaYEnviarMensaje(idGlobal, PORT_PASARELA, messageForSlave, 4);
    }
  }
}

void iniciarTrigger()
{
  // Ponemos el Triiger en estado bajo y esperamos 2 ms
  digitalWrite(PinTrig, LOW);
  delayMicroseconds(2);

  // Ponemos el pin Trigger a estado alto y esperamos 10 ms
  digitalWrite(PinTrig, HIGH);
  delayMicroseconds(10);

  // Comenzamos poniendo el pin Trigger en estado bajo
  digitalWrite(PinTrig, LOW);
}


double filterData(double sensor, String sensorName) {
  if (sensorName == "sensorMovimiento") {
```

```
valorSensor = digitalRead(sensorMov);


    Serial.println(valorSensor);

    if(valorSensor == 1){
      Serial.println("Se mueve");
      valorSensor == HIGH;
    }
    else if (valorSensor == 0){
      Serial.println("No se mueve");
      valorSensor == LOW;
    }
    delay(1000);
  } else if (sensorName == "sensorLuz") {
    //Media para la medicion de detecciones
    for (int i = 0; i < deteccionesParaMedia; ++i)
    {
      mediaMedicion += analogRead(sensor);
      delay(7);
    }
    mediaMedicion /= deteccionesParaMedia;
    if (mediaMedicion > umbralLuz) {
      valorSensor = LOW;
    } else {
      valorSensor = HIGH;
    }

  }

  return valorSensor;

}

double calculoDistancia() {

  // La función pulseIn obtiene el tiempo que tarda en cambiar entre estados, en este
caso a HIGH
  unsigned long tiempo = pulseIn(PinEcho, HIGH);

  // Obtenemos la distancia en cm, hay que convertir el tiempo en segudos ya que está en
microsegundos
  // por eso se multiplica por 0.000001
  distancia = tiempo * 0.000001 * VelSon / 2.0;

  return distancia;

}

double calculoSonido() {

  double valorTotal = 0;
  for (int i = 0; i < 6; i++) {
    double valorSound = medidorSonido();
    valorTotal = valorTotal + valorSound;
  }
  valorTotal = valorTotal / 6;

  return valorTotal;
}

double medidorSonido() {

  unsigned long startMillis = millis(); // Start of sample window
  unsigned int peakToPeak = 0;   // peak-to-peak level
  unsigned int signalMax = 0;
  unsigned int signalMin = 1024;
  // collect data for 50 mS
  while (millis() - startMillis < sampleWindow) {
    sample = analogRead(soundPin);
    if (sample < 1024) { // toss out spurious readings
      if (sample > signalMax) {
        signalMax = sample;  // save just the max levels
      }
      else if (sample < signalMin) {
        signalMin = sample;  // save just the min levels
```

```
      }
    }
  }
  peakToPeak = signalMax - signalMin;  // max - min = peak-peak amplitude
  double volts = (peakToPeak * 5.0) / 1024;  // convert to volts

  return volts;

}

// Devuelve la potencia que debe tener el reostator
int intensidadLuminaria(int lightIntens, int movement, double proximity, double
soundIntens) {
  int intens = 2;
  if (movement == HIGH && proximity < umbralProx && lightIntens == HIGH && soundIntens >
umbralSon) {
    intens = 2;
  } else if (movement == HIGH && proximity < umbralProx && lightIntens == HIGH &&
soundIntens < umbralSon) {
    intens = 2;
  } else if (movement == HIGH && proximity > umbralProx && lightIntens == HIGH &&
soundIntens > umbralSon) {
    intens = 2;
  } else if (movement == HIGH && proximity > umbralProx && lightIntens == HIGH &&
soundIntens < umbralSon) {
    intens = 2;
  } else if (movement == LOW && proximity < umbralProx && lightIntens == HIGH &&
soundIntens > umbralSon) {
    intens = 2;
  } else if (movement == LOW && proximity < umbralProx && lightIntens == HIGH &&
soundIntens < umbralSon) {
    intens = 2;
  } else if (movement == LOW && proximity > umbralProx && lightIntens == HIGH &&
soundIntens > umbralSon) {
    intens = 2;
  }
  else {
    intens = 0;
  }
  return intens;
  delay (1000);
}


//Envia mensaje por la conexión establecida
boolean enviarMensaje(String mensaje, int id_conexion) {
  //Al aceptar multiples conexiones (+CIPMUX=1) hay que indicar el identificador de la
conexion
  boolean resultado = false;
  //Se prepara el mensaje
  String men = mensaje;

  //men += "\r\n"; //No hace falta porque enviamos estos dos caracteres con el println
(no con el print) pero sí hay que tenerlos en cuenta en la longitud del mensaje a enviar
(men.length()+2)
  //pinntln: ASCII text followed by a carriage return character (ASCII 13, or '\r') and
a newline character (ASCII 10, or '\n')
  //Se prepara el comando con el tamaño de los datos a enviar
  String cmd = "AT+CIPSEND=";
  cmd += String(id_conexion);
  cmd += ",";
  cmd += String(men.length() + 2); //Muy importante sumar +2 correspondientes a los
caracteres "\r\n"
  //que se envían con el println (no con el print, en ese caso habría que añadir los
caracteres al mensaje (men += "\r\n";)
  //Se realiza la peticion de envío
  Serial.print("Enviando mensaje ++++++++++++++++++");
  SerialEsp8266.println(cmd);//Se envía el comando correspondiente al ESP8266
  delay(500);
  do {
    delay(100);//Se espera contestacion del ESP8266
  } while (!SerialEsp8266.available());
  //Se comprueba si se ha obtenido la respuesta esperada:
  if (SerialEsp8266.find(">")) {
    // Se envía el mensaje
    SerialEsp8266.println(men);
    do {
```

```
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
    if (SerialEsp8266.find("OK")) {
      Serial.println("... enviado: " + men);
      resultado = true;
    } else {
      Serial.println("... ERROR en el envio de: " + men);
    };
  } else {
    Serial.println("... ERROR. No recibido >");
    //Si no se ha recibido el caracter > es que ya no existe la conexion
  };
  return resultado;
}

void conectarConPasarelaYEnviarMensaje(String IP_PAS, String PORT_PAS, String mensaje,
                                       int id_conexion) {
  int contadorMensajesEnviados = 0;
  //Si se aceptan multiples conexiones (+CIPMUX=1) hay que indicar el identificador de
  //la conexion.
  //Se prepara el comando con la petición para conectar con la pasarela via TCP:
  String cmd = "AT+CIPSTART=";
  cmd += String(id_conexion);
  cmd += ",";
  cmd += "\"TCP\",\"";
  cmd += IP_PAS; // ip del servidor en la pasarela
  cmd += "\",";
  cmd += PORT_PAS;

  do { //Se realiza la peticion de conexión
    if (true) {

      Serial.println("Enviando peticion de conexion a la pasarela con IP: " + IP_PAS + "
y puerto: " + PORT_PAS);
      SerialEsp8266.println(cmd); //Se envía el comando correspondiente al ESP8266
      delay(100); //Tiempo para que responda el módulo ESP8266.
    }
    while (!SerialEsp8266.available()) { //Esperamos a que la respuesta llegue al buffer
      delay(100);
    };
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK") || SerialEsp8266.find("ALREADY CONNECT") ||
conecionEstablecida) { //La respuesta es "ALREADY CONNECT" si ya existe la
      conecionEstablecida = false;
      if (conecionEstablecida) {
        Serial.println("¿Establecida conexion? --> " + conecionEstablecida);
      }
      //conexión
      Serial.println("Conectado y enviando");
      //Se ha conectado con la pasarela y ahora se envía el mensaje:
      if (enviarMensaje(mensaje, id_conexion)) {
        Serial.println("OK ENVIADO");
        break;
      };
    } else {
      Serial.println("Esperado para enviar ...");
    };
    contadorMensajesEnviados ++;
  } while (contadorMensajesEnviados <= 15); //Si algo ha fallado vuelve a intentarse
conectar con la pasarela y
  //enviarle el mensaje
  controlDeFlujo = false;
}

String procesarDatos(String Message) {

  //PROCESADO DE DATOS

  int booleanDist = 0;
  int booleanLum = 0;
  int booleanMov = 0;
  int booleanSon = 0;

  String str = Message.substring(5);
  Serial.println("El string es: " + str);

  int first = str.indexOf(",");
```

```
  idGlobal = str.substring(0, first);
  idGlobal.trim();
  //id = id.toDouble();
  str = str.substring(first + 1);
  Serial.println("El string es: " + str);

  int second = str.indexOf(",");
  String distancia = str.substring(0, second);
  distancia.trim();
  booleanDist = distancia.toInt();
  if (booleanDist < umbralProx) {
    booleanDist = 1;
  } else {
    booleanDist = 0;
  }
  str = str.substring(second + 1);
  Serial.println("El string es: " + str);

  int third = str.indexOf(",");
  String luminosidad = str.substring(0, third);
  luminosidad.trim();
  booleanLum = luminosidad.toInt();
  if (booleanLum > umbralLuz) {
    booleanLum = 1;
  } else {
    booleanLum = 0;
  }
  str = str.substring(third + 1);

  int fourth = str.indexOf(",");
  String movimiento = str.substring(0, fourth);
  movimiento.trim();
  booleanMov = movimiento.toInt();
  str = str.substring(fourth + 1);

  int fiveth = str.indexOf(",");
  String sonido = str.substring(0, fiveth);
  sonido.trim();
  double sonidoN = sonido.toDouble();
  if (sonidoN > umbralSon) {
    booleanSon = 1;
  }

  Serial.print("El id es: ");
  Serial.println(idGlobal);
  Serial.print("La distancia es: ");
  Serial.println(booleanDist);
  Serial.print("La luminosidad es: ");
  Serial.println(booleanLum);
  Serial.print("La movimiento es: ");
  Serial.println(booleanMov);
  Serial.print("El sonido es: ");
  Serial.println(booleanSon);

  if (idGlobal == "192.168.1.2") {
    stateSlave = stateSlaveTwo;
  } else if (idGlobal == "192.168.1.3") {
    stateSlave = stateSlaveOne;
  }


  if (booleanDist == 0 && booleanLum == 0 && booleanMov == 0 && booleanSon == 0 &&
(stateSlave == 2 || lightStateMaster == 1)) {
    messageForSlave = "1";
  } else if (booleanDist == 0 && booleanLum == 0 && booleanMov == 0 && booleanSon == 1
&& stateSlave == 0) {
    messageForSlave = "2";
  } else if (booleanDist == 0 && booleanLum == 0 && booleanMov == 0 && booleanSon == 1
&& (stateSlave == 2 || lightStateMaster == 1)) {
    messageForSlave = "2";
  } else if (booleanDist == 0 && booleanLum == 0 && booleanMov == 1 && booleanSon == 0
&& stateSlave == 0) {
    messageForSlave = "2";
  } else if (booleanDist == 0 && booleanLum == 0 && booleanMov == 1 && booleanSon == 0
&& (stateSlave == 2 || lightStateMaster == 1)) {
    messageForSlave = "2";
```

```
  } else if (booleanDist == 0 && booleanLum == 0 && booleanMov == 1 && booleanSon == 1
&& stateSlave == 0) {
    messageForSlave = "2";
  } else if (booleanDist == 0 && booleanLum == 0 && booleanMov == 1 && booleanSon == 1
&& (stateSlave == 2 || lightStateMaster == 1)) {
    messageForSlave = "2";
  } else if (booleanDist == 1 && booleanLum == 0 && booleanMov == 0 && booleanSon == 0
&& stateSlave == 0) {
    messageForSlave = "2";
  } else if (booleanDist == 1 && booleanLum == 0 && booleanMov == 0 && booleanSon == 0
&& (stateSlave == 2 || lightStateMaster == 1)) {
    messageForSlave = "2";
  } else if (booleanDist == 1 && booleanLum == 0 && booleanMov == 0 && booleanSon == 1
&& stateSlave == 0) {
    messageForSlave = "2";
  } else if (booleanDist == 1 && booleanLum == 0 && booleanMov == 0 && booleanSon == 1
&& (stateSlave == 2 || lightStateMaster == 1)) {
    messageForSlave = "2";
  } else if (booleanDist == 1 && booleanLum == 0 && booleanMov == 1 && booleanSon == 0
&& stateSlave == 0) {
    messageForSlave = "2";
  } else if (booleanDist == 1 && booleanLum == 0 && booleanMov == 1 && booleanSon == 0
&& (stateSlave == 2 || lightStateMaster == 1)) {
    messageForSlave = "2";
  } else if (booleanDist == 1 && booleanLum == 0 && booleanMov == 1 && booleanSon == 1
&& stateSlave == 0) {
    messageForSlave = "2";
  } else if (booleanDist == 1 && booleanLum == 0 && booleanMov == 1 && booleanSon == 1
&& (stateSlave == 2 || lightStateMaster == 1)) {
    messageForSlave = "2";
  } else {
    messageForSlave = "0";
  }

  if (idGlobal == "192.168.1.2" && messageForSlave == "2") {
    stateSlaveOne = 2;
  } else if (idGlobal == "192.168.1.2" && messageForSlave == "1") {
    stateSlaveOne = 1;
  } else if (idGlobal == "192.168.1.2" && messageForSlave == "0") {
    stateSlaveOne = 0;
  }

  if (idGlobal == "192.168.1.3" && messageForSlave == "2") {
    stateSlaveTwo = 2;
  } else if (idGlobal == "192.168.1.3" && messageForSlave == "1") {
    stateSlaveTwo = 0;
  } else if (idGlobal == "192.168.1.2" && messageForSlave == "0") {
    stateSlaveOne = 0;
  }

  return messageForSlave;
}
```

Código del esclavo:

```
#include <SoftwareSerial.h>
#include <Servo.h>
SoftwareSerial SerialEsp8266(3, 2); // RX, TX

int lightState;
int contadorAutonomo = 0;
```

```
#define SSID "luminaria" //El ssid del router
#define PASS "Arduino" //La clave Wifi
#define IP_PASARELA "192.168.4.1" //Ordenador pasarela
#define PORT_PASARELA "80"

int idnodo = 1;
String mi_IP = "";
String mensaje = "";


//INICIALIZACIÓN VARIABLES
const int LDRPin = A0;
int valorLuz = 0;

int pinLed = 12;
int pinLedDos = 7;
int intensidadLuz = 1;

const int PinTrig = 9;
const int PinEcho = 8;
const float VelSon = 34000.0;
float distancia;

int soundPin = A1;
const int sampleWindow = 50; // Sample window width in mS (50 mS = 20Hz)
unsigned int sample;
double valorSonido = 0.0;

int sensorMov = 11;
int valorSensor = 0;
int valorMovimiento = 0;

float mediaMedicion = 0;
int deteccionesParaMedia = 50;
double valorDistancia = 0.0;

const int umbralLuz = 35;
const double umbralSon = 3.0;
const double umbralProx = 50.0;

//CIERRE INICIALIZACIÓN VARIABLES


void setup() {
  //Para conectar con el modulo esp8266
  SerialEsp8266.begin(9600);
  //Para conectar con el modulo serie
  Serial.begin(9600);

  inicializarEsp8266();
  conectarConRouterWifi(SSID, PASS);
  configurarServidorTCP(80);

  //VARIABLES SENSORES
  // Ponemos el pin LDR en modo entrada
  pinMode(LDRPin, INPUT);
  // Ponemos el pin Trig en modo salida
  pinMode(PinTrig, OUTPUT);
  // Ponemos el pin Echo en modo entrada
  pinMode(PinEcho, INPUT);
  // Ponemos el pin sensorMov en modo entrada
  pinMode(sensorMov, INPUT);
  // Ponemos el pin Led en modo salida
  pinMode(pinLed, OUTPUT);
  // Ponemos el pin Led en modo salida
  pinMode(pinLedDos, OUTPUT);
  //FIN VARIABLES SENSORES
  // La función pulseIn obtiene el tiempo que tarda en cambiar entre estados, en este
caso a HIGH

}



//BUCLE DE EJECUCION
void loop() {
```

```
  //Calculos

  iniciarTrigger();
  valorDistancia = calculoDistancia();
  Serial.print("PROXIMITI 1");
  Serial.println(valorDistancia);
  //String(valorDistancia);

  valorLuz = filterData(LDRPin, "sensorLuz");
  Serial.print("LUZ ");
  Serial.println(valorLuz);
  //String(valorLuz);

  valorMovimiento = filterData(sensorMov, "sensorMovimiento");
  Serial.print("MOVIMIENTO ");
  Serial.println(valorMovimiento);
  //String(valorMovimiento);

  valorSonido = calculoSonido();
  Serial.print("SONIDO: ");
  Serial.println(valorSonido);

  //Cierre calculos


  if (contadorAutonomo >= 25 ) {
    Serial.println("Modo autonomo");
    while (true) {
      iniciarTrigger();

      valorDistancia = calculoDistancia();
      Serial.print("PROXIMITI 1");
      Serial.println(valorDistancia);
      //String(valorDistancia);

      valorLuz = filterData(LDRPin, "sensorLuz");
      Serial.print("LUZ ");
      Serial.println(valorLuz);

      valorMovimiento = filterData(sensorMov, "sensorMovimiento");
      Serial.print("MOVIMIENTO ");
      Serial.println(valorMovimiento);

      valorSonido = calculoSonido();
      Serial.print("SONIDO: ");
      Serial.println(valorSonido);

      intensidadLuz = intensidadLuminaria(valorLuz, valorMovimiento, valorDistancia,
valorSonido);
      if (intensidadLuz == 2) {
        lightState = 1;
        digitalWrite(pinLed, lightState);
        digitalWrite(pinLedDos, lightState);
      } else if (intensidadLuz == 0) {
        lightState = 0;
        digitalWrite(pinLed, lightState);
        digitalWrite(pinLedDos, lightState);
      }

    }
  }


  String Message = "192.168.4.2, ";
  Message += valorDistancia;
  Message += ", ";
  Message += valorLuz;
  Message += ", ";
  Message += valorMovimiento;
  Message += ", ";
  Message += valorSonido;


  conectarConPasarelaYEnviarMensaje(IP_PASARELA, PORT_PASARELA, Message, 4); //se pone 4
como identificador de conexion

}
```

```
boolean inicializarEsp8266() {

  do { //Se reinicia el modulo esp8266
    Serial.print("Reseteando modulo esp8266");
    SerialEsp8266.println("AT+RST"); //Se envía el comando correspondiente al ESP8266
    do {
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("ready")) {
      Serial.println("... Modulo preparado");
      break;
    } else {
      Serial.println("... ERROR");
    };
  } while (true);

  /**/
  do { //Se configura como modo 1: SOFTAP (no router)
    Serial.print("Estableciendo modo 1");
    SerialEsp8266.println("AT+CWMODE DEF=1"); //Se envía el comando correspondiente al
ESP8266
    delay(100);//Se espera contestacion del ESP8266
    while (!SerialEsp8266.available());
    Serial.println((char)SerialEsp8266.read());
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK")) {
      Serial.println("... Modo 1 (Station) establecido");
      break;
    } else {
      Serial.println("... ERROR");
    };
  } while (true);

}



boolean conectarConRouterWifi(String ID_Router, String password) {
  String cmd = "AT+CWJAP_CUR=\"" + ID_Router + "\",\"" + password + "\"";
  do { //Se conecta al router wifi
    Serial.print("Conectando al router wifi: " + ID_Router + " con password: " +
password);
    SerialEsp8266.println(cmd); //Se envía el comando correspondiente al ESP8266
    delay(100); //Tiempo para que responda el módulo ESP8266
    while (!SerialEsp8266.available()) { //Esperamos a que la respuesta llegue al buffer
      delay(100);
    };
    if (SerialEsp8266.find("OK")) { //Se comprueba si se ha obtenido la respuesta
esperada
      Serial.println("... Esp8266 conectado al router wifi");
      break;
    } else {
      Serial.println("... ERROR");
      contadorAutonomo++;
      Serial.println(contadorAutonomo);
      if (contadorAutonomo >= 25) {
        loop();
      }
    };
  } while (true);

  return true;
}

boolean configurarServidorTCP(int port) {
  do { //Aceptar multiples conexiones
    Serial.print("Estableciendo aceptar multiples conexiones");
    SerialEsp8266.println("AT+CIPMUX=1");//Se envía el comando correspondiente al
ESP8266
    do {
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
```

```
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK")) { //Si ya está configurado devuelve "Link is builded"
      Serial.println("... establecido");
      break;
    } else {
      Serial.println("... ERROR");
    };
  } while (true);

  do { //Configurar como Server TCP en el puerto "port"
    Serial.print("Configurando servidor TCP en puerto: " + String(port));
    String cmd = "AT+CIPSERVER=1,";
    cmd += String(port);
    SerialEsp8266.println(cmd);//Se envía el comando correspondiente al ESP8266
    //Con SerialEsp8266.println("AT+CIPSERVER=1,"+port) NO FUNCIONA
    do {
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK")) {
      Serial.println("... configurado.");
      break;
    } else {
      Serial.println("... ERROR");
    };
  } while (true);

  do { //Configurar el timeout
    Serial.print("Configurando el timeout del server...");
    String cmd = "AT+CIPSTO=30";
    SerialEsp8266.println(cmd);//Se envía el comando correspondiente al ESP8266
    do {
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK")) {
      Serial.println("... configurado.");
      break;
    } else {
      Serial.println("... ERROR");
    };
  } while (true);

  Serial.println("Esperando conexion...");
}


void recibirMensaje() {
  int contadorIntentos = 0;
  if (SerialEsp8266.available() > 0) { //Ver si hay mensaje del ESP8266
    while (!SerialEsp8266.find("+IPD,") && contadorIntentos <= 15) {
      Serial.println("==========> ESPERANDO AQUI ==========>");
      contadorIntentos = contadorIntentos + 1;

      delay(100);//Se espera contestacion del ESP8266

    }
    if (contadorIntentos >= 15) {
      Serial.println("==========> MAXIMO TIEMPO ESPERANDO ==========>");
    } else {
      // if (SerialEsp8266.find("+IPD,")) { //Ver si el servidor ha recibido datos*/

      String S = SerialEsp8266.readStringUntil('\r'); //Lee los datos hasta que se
encuentra el carriage return character (ASCII 13, or '\r'). El caracter buscado no se
devuelve en la cadena resultado.
      Serial.println("==========> RECIBIDO: "  + S);

      if (S == "2") {
        lightState = 1;
        digitalWrite(pinLed, lightState);
        digitalWrite(pinLedDos, lightState);
      } else if (S == "1") {
        lightState = 1;
        digitalWrite(pinLed, lightState);
        digitalWrite(pinLed, LOW);
      } else if (S == "0") {
        lightState = 0;
```

```
        digitalWrite(pinLed, lightState);
        digitalWrite(pinLed, lightState);
      }

      while (SerialEsp8266.available()) { //Si hubiera mas lineas, no deberia si el
cliente cumple el formato que se ha decidido, se leen para vaciar el buffer.
        SerialEsp8266.read(); //c = SerialEsp8266.read();


      }
    }

  }
}

void iniciarTrigger()
{
  // Ponemos el Triiger en estado bajo y esperamos 2 ms
  digitalWrite(PinTrig, LOW);
  delayMicroseconds(2);

  // Ponemos el pin Trigger a estado alto y esperamos 10 ms
  digitalWrite(PinTrig, HIGH);
  delayMicroseconds(10);

  // Comenzamos poniendo el pin Trigger en estado bajo
  digitalWrite(PinTrig, LOW);
}


double filterData(double sensor, String sensorName) {
  if (sensorName == "sensorMovimiento") {

    valorSensor = digitalRead(sensorMov);


    Serial.println(valorSensor);

    if (valorSensor == 1) {
      Serial.println("Se mueve");
      valorSensor == HIGH;
    }
    else if (valorSensor == 0) {
      Serial.println("No se mueve");
      valorSensor == LOW;
    }
    delay(1000);
  } else if (sensorName == "sensorLuz") {
    //Media para la medicion de detecciones
    for (int i = 0; i < deteccionesParaMedia; ++i)
    {
      mediaMedicion += analogRead(sensor);
      delay(7);
    }
    mediaMedicion /= deteccionesParaMedia;
    if (mediaMedicion > umbralLuz) {
      valorSensor = LOW;
    } else {
      valorSensor = HIGH;
    }

  }

  return valorSensor;

}


double calculoDistancia() {

  // La función pulseIn obtiene el tiempo que tarda en cambiar entre estados, en este
caso a HIGH
  unsigned long tiempo = pulseIn(PinEcho, HIGH);

  // Obtenemos la distancia en cm, hay que convertir el tiempo en segudos ya que está en
microsegundos
  // por eso se multiplica por 0.000001
```

```
  distancia = tiempo * 0.000001 * VelSon / 2.0;

  return distancia;
}

double calculoSonido() {

  double valorTotal = 0;
  for (int i = 0; i < 6; i++) {
    double valorSound = medidorSonido();
    valorTotal = valorTotal + valorSound;
  }
  valorTotal = valorTotal / 6;

  return valorTotal;
}

double medidorSonido() {

  unsigned long startMillis = millis(); // Start of sample window
  unsigned int peakToPeak = 0;   // peak-to-peak level
  unsigned int signalMax = 0;
  unsigned int signalMin = 1024;
  // collect data for 50 mS
  while (millis() - startMillis < sampleWindow) {
    sample = analogRead(soundPin);
    if (sample < 1024) { // toss out spurious readings
      if (sample > signalMax) {
        signalMax = sample;  // save just the max levels
      }
      else if (sample < signalMin) {
        signalMin = sample;  // save just the min levels
      }
    }
  }
  peakToPeak = signalMax - signalMin;  // max - min = peak-peak amplitude
  double volts = (peakToPeak * 5.0) / 1024;  // convert to volts

  return volts;

}



//Envia mensaje por la conexión establecida
boolean enviarMensaje(String mensaje, int id_conexion) {
  //Al aceptar multiples conexiones (+CIPMUX=1) hay que indicar el identificador de la
conexion
  boolean resultado = false;
  //Se prepara el mensaje
  String men = mensaje;
  //men += "\r\n"; //No hace falta porque enviamos estos dos caracteres con el println
(no con el print) pero sí hay que tenerlos en cuenta en la longitud del mensaje a enviar
(men.length()+2)
  //pinntln: ASCII text followed by a carriage return character (ASCII 13, or '\r') and
a newline character (ASCII 10, or '\n')
  //Se prepara el comando con el tamaño de los datos a enviar
  String cmdu = "AT+CIPSEND=";
  cmdu += String(id conexion);
  cmdu += ",";
  cmdu += String(men.length() + 2); //Muy importante sumar +2 correspondientes a los
caracteres "\r\n"
  //que se envían con el println (no con el print, en ese caso habría que añadir los
caracteres al mensaje (men += "\r\n";)
  //Se realiza la peticion de envío
  Serial.println("Enviando mensaje");
  SerialEsp8266.println(cmdu);//Se envía el comando correspondiente al ESP8266
  do {
    delay(100);//Se espera contestacion del ESP8266
  } while (!SerialEsp8266.available());
  //Se comprueba si se ha obtenido la respuesta esperada:
  if (SerialEsp8266.find(">")) {
    // Se envía el mensaje
    SerialEsp8266.println(men);
    do {
      delay(100);//Se espera contestacion del ESP8266
    } while (!SerialEsp8266.available());
```

```
    if (SerialEsp8266.find("OK")) {
      Serial.println("Correctamente enviado: " + men);
      resultado = true;
    } else {
      Serial.println("... ERROR en el envio de: " + men);
    };
  } else {
    //Serial.println("... ERROR. No recibido >");
    //Si no se ha recibido el caracter > es que ya no existe la conexion
  };
  return resultado;
}

void conectarConPasarelaYEnviarMensaje(String IP_PAS, String PORT_PAS, String mensaje,
                                       int id_conexion) {
  int contadorMensajesEnviados = 0;
  //Si se aceptan multiples conexiones (+CIPMUX=1) hay que indicar el identificador de
  //la conexion.
  //Se prepara el comando con la petición para conectar con la pasarela via TCP:
  String cmdx = "AT+CIPSTART=";
  cmdx += id_conexion;
  cmdx += ",";
  cmdx += "\"TCP\",\"";
  cmdx += IP_PAS; // ip del servidor en la pasarela
  cmdx += "\",";
  cmdx += PORT_PAS;

  Serial.println(cmdx);

  do { //Se realiza la peticion de conexión
    Serial.println("Enviando peticion de conexion a la pasarela con IP: " + IP_PAS + " y
puerto: " + PORT_PAS);
    SerialEsp8266.println(cmdx); //Se envía el comando correspondiente al ESP8266
    delay(100); //Tiempo para que responda el módulo ESP8266.
    while (!SerialEsp8266.available()) { //Esperamos a que la respuesta llegue al buffer
      delay(100);
    };
    //Se comprueba si se ha obtenido la respuesta esperada:
    if (SerialEsp8266.find("OK") || SerialEsp8266.find("ALREADY CONNECT")) { //La
respuesta es "ALREADY CONNECT" si ya existe la
      //conexión
      Serial.println("Conectado y enviando");
      //Se ha conectado con la pasarela y ahora se envía el mensaje:

      if (enviarMensaje(mensaje, id_conexion)) {
        recibirMensaje();
        //break;
      };
    } else {
      Serial.println("Esperando para enviar ...");
      //cerrarConexion(id_conexion); //Por si la respuesta fuera "ALREADY CONNECT"
      //entonces se cierra la conexión
    };
    contadorMensajesEnviados ++;
  } while (contadorMensajesEnviados <= 5); //Si algo ha fallado vuelve a intentarse
conectar con la pasarela y
  //enviarle el mensaje
}

int intensidadLuminaria(int lightIntens, int movement, double proximity, double
soundIntens) {
  int intens = 2;
  if (movement == HIGH && proximity < umbralProx && lightIntens == HIGH && soundIntens >
umbralSon) {
    intens = 2;
  } else if (movement == HIGH && proximity < umbralProx && lightIntens == HIGH &&
soundIntens < umbralSon) {
    intens = 2;
  } else if (movement == HIGH && proximity > umbralProx && lightIntens == HIGH &&
soundIntens > umbralSon) {
    intens = 2;
  } else if (movement == HIGH && proximity > umbralProx && lightIntens == HIGH &&
soundIntens < umbralSon) {
    intens = 2;
  } else if (movement == LOW && proximity < umbralProx && lightIntens == HIGH &&
soundIntens > umbralSon) {
    intens = 2;
```

```
  } else if (movement == LOW && proximity < umbralProx && lightIntens == HIGH &&
soundIntens < umbralSon) {
    intens = 2;
  } else if ((movement == LOW) && (proximity > umbralProx) && (lightIntens == HIGH) &&
(soundIntens > umbralSon)) {
    intens = 2;
  }
  else {
    intens = 0;
  }
  return intens;
  delay (1000);
}
```