Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE ILUMINACIÓN NATURAL BASADO EN ARDUINO

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Alberto Barres Martínez

Tutor: Jorge Más Estellés

Director Experimental: Antonio Pinci Ferrer

2017/2018

Agradecimientos

Me gustaría mostrar mi agradecimiento a Jorge Más Estellés, que ha sido la persona que se ha encargado de la supervisión del proyecto desde el primer momento, la cual me ha tratado de una forma estupenda en todo momento, resolviéndome las posibles dudas que me han ido surgiendo y atendiéndome siempre que se lo he pedido.

Hacer especial mención a mi familia y amigos, que han estado en todo momento apoyándome, haciendo que sacara fuerzas de donde fuera y nunca dejando que me rindiera ante los reveses con los que me he ido encontrado durante este tiempo.

Resumen

El objetivo de este proyecto consiste en el control del nivel de iluminación natural, bien sea de una casa o de un edificio. Para ello, nuestra principal tarea será la automatización del movimiento de subida y bajada de las persianas de la estancia deseada, basándonos, para ello, en la utilización de software y hardware libre, escogiendo Arduino como plataforma para llevar a cabo la implementación, así como una serie de dispositivos, los cuáles harán que el movimiento sea trivial.

Palabras clave: Arduino, iluminación, natural, LDR, DS3231.

Resum

L'objectiu d'aquets projecte consisteix en el control del nivell d'iluminatció natural, ja siga de una casa o un edifici. La nostra principal tarea serà l'automatització del moviment de pujada i baixada de les persianes de la estada que desitjem, basant-nos en la utilització de software i hardware lliure, triant Arduino com a plataforma per dur a terme la implementació, així com una sèrie de dispositius, els quàls faran que el moviment siga trivial.

Paraules clau: Arduino, iluminació, natural, LDR, DS3231.

Abstract

The aim of this project is the control of the natural lighting, either from a house or from a building. For this, our main objective will be the automation of the movement of raising and lowering the blinds of any residence, based, for this, on the use of free software and hardware, choosing Arduino as the platform to carry out the implementation, as well as a series of devices, which will make the movement trivial.

Keywords: Arduino, lighting, natural, LDR, DS3231.

Tabla de contenidos

Tabla de contenido

1. Introducción

A lo largo de este documento vamos a explicar el proceso que se ha de llevar a cabo para poder implementar un sistema el cuál controle la iluminación de una casa (de forma natural) en base a la cantidad de luz que tengamos en el exterior y, de esta forma, poder ahorrar en el consumo eléctrico.

1.1. Motivación

El hecho de que la tecnología ha venido sufriendo un continuo avance durante los últimos años es algo indiscutible. Allá donde miremos, podemos ver la presencia de tecnología: móviles de últimas generaciones capaces de reconocernos nuestra huella digital o facial, coches que conducen por nosotros, casas autónomas que hacen nuestras tareas del hogar y un largo etcétera.

Y es del concepto de la domótica de donde surge la idea de este proyecto. La capacidad de poder controlar los aspectos de la casa de manera automática es uno de los grandes retos para el futuro.

Gracias a esta idea, vamos a desarrollar un sistema autónomo capaz de satisfacer nuestras necesidades lumínicas explotando, para ello, la luz solar y abaratando, así, nuestra factura de la luz.

1.2. Objetivos

1. Diseñar e implementar un sistema capaz de automatizar el movimiento de una serie de componentes los cuáles hagan que las persianas de un edificio suban y bajen de manera autónoma en función de la luz proveniente del exterior.
2. Utilización de hardware y software accesible para toda persona que quiera llevar a cabo este proyecto.
3. Minimizar el consumo eléctrico.

1.3. Estructura

Tal y como se muestra en la tabla de contenido, el documento se va a dividir en 9 partes.

1. Introducción. En este apartado damos una breve explicación sobre cuál va a ser el proyecto y en qué va a consistir.
2. Estado del arte. A lo largo del punto dos, vemos las distintas aplicaciones que se pueden comparar con nuestro proyecto y damos una explicación de por qué elegimos adentrarnos en este proyecto.
3. Análisis del problema. Explicamos las distintas oportunidades que nos pueden surgir al llevar a cabo nuestro sistema.
4. Diseño de la solución. Durante el apartado de diseño de la solución vamos a comentar cuáles han sido los materiales utilizados, tanto de la parte de software como de la de hardware.

5. Desarrollo de la solución propuesta. Vemos cuál era la idea principal de la que partimos y cómo ha ido avanzando el proyecto hasta llegar a la solución final implementada.
6. Implantación. En este apartado vemos la puesta en marcha de nuestro proyecto.
7. Pruebas. Se muestran las distintas pruebas realizadas para comprobar el correcto funcionamiento del dispositivo.
8. Conclusiones. Se da una valoración del proyecto donde muestro que se han cumplido los objetivos.
9. Trabajos futuros. Haremos hincapié en las distintas ideas que se han ido valorando durante la realización del proyecto y que no ha sido posible llevar a cabo.
10. Referencias. En este último apartado se muestra una serie de información gracias a la cual se puede comprobar de donde hemos extraído las fuentes necesarias para la implementación del proyecto.

2. Estado del arte

Durante este apartado, vamos a realizar un estudio estratégico en relación al proyecto que queremos implementar y para ello necesitamos saber si alguien ha puesto en práctica nuestra idea o algo que se asemeje.

La primera aplicación con la que nos encontramos viene de la mano de la empresa Loxone, la cual ha desarrollado un sistema para que el movimiento de las lamas de las persianas se realice de forma autónoma, teniendo en cuenta una serie de parámetros tales como la temperatura, la luminosidad o el horario. En función de la posición del sol, la hora del día y la temperatura objetivo, el sombreado automático mueve las lamas de la persiana. Gracias a este sistema se asegura la privacidad al atardecer, bajando por completo las persianas. Además, en verano evita que la luz del sol sobrecaliente la estancia y en invierno lo utiliza para ayudar al sistema de calefacción. Cuenta, también, con un sistema para que las persianas de toda la casa suban a una hora estipulada. Este sistema es aplicable a cualquier tipo de persiana, toldo, cortina o veneciana. Si se quiere un control individualizado, cuenta con una app disponible para Smartphone, tabletas y ordenadores en la que podrás tener total control de la persiana, así como ver su estado y su posición desde cualquier lugar.

El siguiente ejemplo ha sido desarrollado por la empresa Distech Controls. Éstos han creado unos controladores configurables tanto para iluminación artificial como natural con los que prometen una reducción en el consumo energético de hasta un 70% a través de una multitud de técnicas de control tales como la programación horaria basada en eventos, medición de luz exterior o personalización de tareas, entre otras. Esta idea surgió al realizar un estudio sobre el uso de energía en un edificio, que demostró que suponía entre un 30% y un 40%. Además de los métodos citados anteriormente para el control de la iluminación, el sistema proporciona una serie de funciones para obtener más control y automatización, entre los que se incluye el control de personal, el control de mantenimiento y el forzado manual, avisos automáticos y reloj astronómico.

Ambos ejemplos llevan asociados el concepto de iluminación natural y eficiencia energética. Esto último es lo que más preocupa a la hora de implantar una solución de esta índole, puesto que va relacionado el poder explotar al máximo la luz natural con el gasto de luz artificial.

A continuación, hablamos sobre un par de ejemplos que siguen el camino del ahorro del consumo eléctrico, pero esta vez actuando sobre la luz artificial directamente. Fue, a partir de la idea del control de la iluminación artificial, de donde salió la idea de aprovechar al máximo la luz solar.

El primer ejemplo que quiero mostrar es un artículo muy interesante sobre la domótica en las casas, que ha sido llevado a cabo por Domoprac, en el cual nos explica, paso a paso cómo ha ido evolucionando esta idea. En él, nos enseña qué se puede hacer con la iluminación si disponemos de domótica y nos da a conocer una serie de tecnologías para el control domótico de la iluminación.
Por último, la empresa Schneider Electric nos muestra su visión sobre el control de la iluminación. Esta empresa nos enseña la tecnología en edificios inteligentes mediante el estándar KNX, el cuál es un estándar válido globalmente que garantiza ahorros en el consumo de energía y, por lo tanto, también en el costo de ésta. Entre las funciones manejadas por este estándar se encuentran la iluminación, la calefacción / aire acondicionado / ventilación, las alarmas y, por supuesto, las persianas eléctricas. Las características de estas últimas son el control de automatización automático, gracias al cual puede activar persianas motorizadas o luces y el control de iluminación automático, también válido para persianas y luces. El sistema puede ser usado para controlar luces y persianas de acuerdo a la luz natural o mediante la detección de movimiento.

2.1. Crítica al estado del arte

Si bien es cierto que hay un elevado número de Trabajos de Fin de Grado (TFG) relacionados con Arduino, ninguno de ellos consigue plasmar la idea que se plantea en este proyecto.

Sí que es cierto que uno de los TFG se acerca a la idea del control de la iluminación, pero en este caso no hace ningún hincapié en la luz natural. Simplemente propone la idea de reducir el impacto ambiental que implica el uso de iluminación en entornos exteriores. En él, podemos observar un sistema de gestión inteligente de la iluminación el cual ha sido desarrollado para un jardín que será iluminado de forma que el ahorro energético sea lo más elevado posible.

Así es que nos encontramos con que nuestra idea no ha sido desarrollada por ninguna otra persona que haya pasado por la Escuela Técnica Superior de Ingeniería Informática (ETSINF) y mucho menos que vaya a usar los recursos con los que pretendemos sacar adelante el trabajo, como es Arduino y los diferentes sensores y actuadores a partir de los cuales basaremos el proyecto.

2.2. Propuesta

En resumen, una vez hecho un estudio de las aplicaciones que se asemejan a la nuestra y de los objetivos que abarcan y viendo que nunca antes se ha realizado este desarrollo en la ETSINF, es hora de pasar a explicar lo que será nuestra propuesta de trabajo.

Nuestra propuesta se basa en el uso de hardware y software libre mediante la utilización de la plataforma Arduino. Gracias a esto, nos resultará menos costoso realizar el desarrollo de la aplicación puesto que tanto el software como el hardware es accesible para todo el mundo.

Hay que ver esta idea como una innovación, ya que son muy pocas las empresas que se hayan atrevido a poner en funcionamiento un sistema de este calibre y mucho menos basándose en los conceptos que vamos a llevar a cabo, que harán, como ya se ha comentado antes, las cosas mucho más simples.

En definitiva, nos encontramos con que abarcamos un campo que está poco explotado y que nos puede abrir las puertas hacia un futuro muy prometedor.

3. Análisis del problema

Nos adentramos en un nuevo apartado para hablar, ahora, sobre las oportunidades que nos proporciona nuestro proyecto y desarrollar un análisis exhaustivo de lo que va a ser el funcionamiento de éste.

3.1. Identificación y análisis de soluciones posibles

Como bien se ha comentado ya a lo largo de este trabajo, la plataforma sobre la que se va a desarrollar va a ser Arduino. Hay otra serie de opciones en el mercado perfectamente válidas para desarrollar el proyecto. Un duro competidor de Arduino es Raspberry Pi. Pese a parecer muy similares, tienen claras diferencias en cuanto a funcionalidad se refiere. La principal diferencia entra ambos es que, mientras que Arduino es un microcontrolador, Raspberry Pi es, básicamente, un mini ordenador. Hay que aclarar que un microcontrolador es una pequeña parte de lo que forma un ordenador, por lo que Arduino es sólo una pequeña parte de lo que puede ofrecer Raspberry Pi. Mientras que con Arduino se pueden programar pequeñas aplicaciones, con Raspberry Pi serás capaz de programar un sistema operativo completo.

Entonces, ¿por qué elegir Arduino y no Raspberry Pi? La respuesta es bien sencilla. Arduino es una plataforma de prototipos electrónica de código abierto, basada en software y hardware flexible y fácil de usar. Gracias a ello, ofrece una serie de ventajas, cuanto menos, interesantes. A continuación, se citarán una serie de ventajas que nos harán decidirnos por esta tecnología. Por ejemplo:

- Es barato, por lo que está al alcance de cualquier persona que quiera adentrarse en este mundo.
- Es multiplataforma, con lo que nuestra idea puede ser desarrollada en sistemas operativos tales como Windows, Macintosh o Linux.
- El entorno de programación es simple y claro, lo que lo hace válido tanto para principiantes como para programadores experimentados.

En definitiva, Arduino es la herramienta perfecta para llevar a cabo todo tipo de proyectos de electrónica, mediante el uso de componentes y sensores, que es lo que buscamos en nuestra idea.

Gracias a la aceptación tanto de Arduino como de Raspberry Pi, es fácil encontrar a una serie de aventureros que se quieran adentrar en este mundo. Y éste es el caso de Omega2, una pequeña propuesta que destaca por su reducido tamaño. Esta placa está destinada, primordialmente, al Internet de las Cosas (Internet Of Things, en inglés (IOT)), por lo que, que su tamaño sea reducido, no quiere decir que lo vaya a ser su conectividad.

Obviamente hay muchas más opciones aparte de las citada anteriormente que se pueden barajar para el desarrollo que se quiere llevar a cabo, pero las propuestas que tuve en mente desde el primer momento fueron éstas.

3.2. Solución propuesta

En los países mediterráneos, la iluminación natural en el interior de los edificios puede llegar a ser excesiva para determinadas tareas, por lo que es necesario amortizar la iluminación natural, lo que se puede hacer mediante diversos métodos, uno de los cuales es la utilización de persianas. Por otra parte, también en algunos momentos, tanto al amanecer como al anochecer, es conveniente aprovechar al máximo la luz natural, lo que puede hacerse de manera manual, pero también es posible aprovechar las nuevas tecnologías y diseñar sistemas que sean capaces de controlar el nivel de iluminación deseado de manera automática. Ello puede conseguirse, de manera sencilla y barata, utilizando el sistema Arduino, implementando un algoritmo capaz de medir el nivel de iluminación existente en una estancia, comparándolo con el nivel de iluminación previamente ajustado por el usuario, y, en función de esta comparación, actuar sobre los motores que mueven las persianas de la estancia. También deberá tenerse en cuenta el horario de utilización, puesto que, en ciertas horas, independientemente del nivel de iluminación, puede ser deseable que haya total oscuridad (por ejemplo, por la noche). El sistema podría ser útil no sólo para viviendas privadas, sino también para edificios públicos.

El sistema deberá constar de:

- Una placa Arduino que actuará como controlador del sistema.
- Un sensor de iluminación que proporcionará información sobre el nivel de luminosidad a la placa Arduino.
- Un Reloj de Tiempo Real (Real Time Clock (RTC)) para saber qué hora es en todo momento.
- Un interruptor para poder desconectar el sistema de forma manual siempre que el usuario quiera.
- Sendos leds que imitarán el movimiento de subida y bajada de la persiana.

3.3. Presupuesto

A continuación, vamos a desglosar los costes de los distintos componentes hardware, así como de las horas que han sido necesarias para desarrollar el proyecto, para hacer una estimación del presupuesto que podríamos ofrecerle a un usuario que se haya interesado por nuestro sistema. Primero que nada, vamos a hacer la estimación de costes de los distintos componentes que se usan.

Componente	Precio	Unidades	Total
Arduino Mega 2560	34,50 €	1	34,50 €
Fotoresistor (LDR)	1,75 €	1	1,75 €
Relé SRS	0,42 €	2	0,84 €
Led	0,20 €	2	0,40 €
Resistencia	0,80 €	2	1,60 €
Real Time Clock (RTC)	7,09 €	1	7,09 €
Interruptor	0,15 €	1	0,15 €
Protoboard	3,49 €	1	3,49 €
Set de cables	0,80 €	1	0,80 €

Base Imponible	Tipo IVA	IVA	Total
50,62 €	21%	10,63 €	61,25 €

Una vez realizado el presupuesto de los componentes que se van a utilizar a lo largo del proyecto, es hora de imputar las horas que han sido necesarias para el desarrollo del mismo. Habrá que hacer una distinción entre las distintas fases por la que ha ido pasando el proyecto, desde el análisis inicial hasta completar las pruebas necesarias.

Descripción	Horas	Precio	Total
Estado del arte	30	5 €	150 €
Análisis del problema	40	6 €	240 €
Diseño de la solución	60	8 €	480 €
Desarrollo de la solución propuesta	80	10 €	800 €
Implantación	40	7 €	280 €
Pruebas	50	6 €	300 €

Base Imponible	Tipo IVA	IVA	Total
2250 €	21%	472,50 €	2722,50 €

4. Diseño de la solución

A lo largo del punto cuatro se va a describir lo que va a ser el diseño de nuestro sistema. Para ello, vamos a dividir este cuarto punto en tres subapartados. En el primero de ellos, hablo de los componentes hardware que se utilizan para poner en marcha nuestra idea. En el segundo, comento el código implementado para hacer que los componentes descritos, funcionen. Finalmente, en el tercer punto, explico cuál ha sido la tecnología utilizada para implantar el proyecto.

4.1. Arquitectura del sistema

En este primer subapartado, voy a dar una explicación de los materiales hardware que han sido utilizados a lo largo del proyecto. Una vez decidido que nuestra idea se llevaría a cabo a partir de la plataforma Arduino, era importante decidir qué placa sería la elegida para conectar nuestros componentes hardware.

En este caso, la placa utilizada ha sido una Arduino Mega 2560. La elección de esta placa y no otra ha sido, simplemente, por el hecho de que yo ya había trabajado anteriormente con esta placa desarrollando proyectos de Arduino, así que partía con una clara ventaja al saber a qué me estaba enfrentando con este dispositivo hardware. Pese a ello, podría haber escogido algún otro tipo de placa, como puede ser la Arduino Uno. Vamos a entrar un poco más en materia y, a continuación, vamos a explicar las características de esta placa. La Arduino Mega 2560 es una placa de desarrollo open-source, la cual está construida sobre un microcontrolador modelo Atmega2560, que posee pines de entrada y salida, que pueden ser tanto analógicas como digitales. Esta placa está formada por 54 pines de entradas y salidas, 16 entradas analógicas, cristal oscilador de 16MHz, conexión USB, jack de alimentación, conector ISCP y botón de reset. Su voltaje de entrada oscila entre 7 y 12V, cuenta con una memoria flash de 256K y la velocidad del reloj es de 16MHz.

Imagen 1. Placa Arduino Mega 2560. http://arduino.cl/arduino-mega-2560/

El segundo componente de mayor importancia en nuestro proyecto es el fotoresistor, en inglés light-dependent resistor o LDR, que es como lo vamos a nombrar a partir de ahora a lo largo del proyecto. Este componente nos permitirá saber qué nivel de luz hay en el exterior. Su funcionamiento se basa en una resistencia, la cual va variando dependiendo de la luz recibida. Al incidir la luz sobre él, algunos fotones son absorbidos, provocando que electrones pasen a la banda de conducción, disminuyendo, así, la resistencia del componente. En resumen, se puede afirmar que a medida que aumenta la luz sobre el LDR, disminuye la resistencia de éste. Los valores oscilan entre 1Mohm (total oscuridad) y 50-100Ohm (luz brillante).

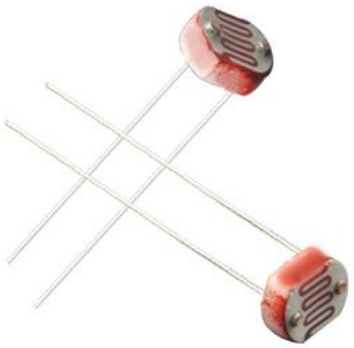

Imagen 2. Fotoresistor o LDR. https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/

La función de los LDR en nuestro proyecto es la de apertura o cierre de sendos relés, que son los que harán que la persiana suba o baje, dependiendo del estado en que se halle el relé. Cabe destacar que, la corriente eléctrica que suministra el relé es de 40 mA, que coincide con la intensidad máxima que nos pueden proporcionar tanto las entradas como las salidas del Arduino. Como no puedo traer una persiana completa, para emular la subida y bajada de las persianas, se han usado dos leds, los cuales se iluminan dependiendo de si la persiana está subiendo o bajando. A estos leds, se le han añadido resistencias de 1K ohmio para limitar la cantidad de corriente que le llegue a los leds y, de esta manera, evitar que se nos puedan quemar.

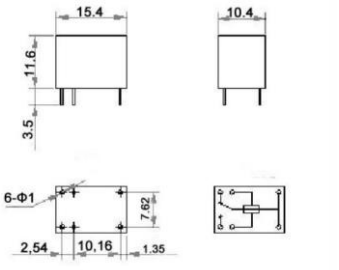

Imagen 3. Esquema eléctrico relé SRS. https://articulo.mercadolibre.com.co/MCO-462246709-rele-relay-o-relevo-srs-24vdc-sl-24v-3a-6p-4100-24v-_JM

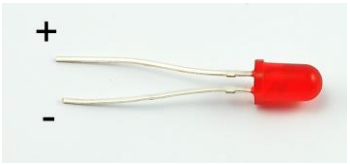

Imagen 4. Led. https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/

Imagen 5. Resistencia 1K ohmio, http://www.ka-shay.net/index.php?id_product=101&controller=product

Como ya se ha comentado en algún punto de este documento, habrá una serie de horas en las que no queramos que haga nada. Para saber en todo momento qué hora es, hemos incorporado a nuestro circuito un reloj de tiempo real, Real Time Clock en inglés o RTC, que es como lo vamos a nombrar a partir de ahora. El RTC en cuestión es el DS3231. Antes de obtener este modelo de RTC, estuve barajando distintas opciones. La primera de ellas fue un DS1302, pero, informándome bien, me di cuenta de que ese modelo estaba obsoleto y que había opciones más interesantes. Descartada esta primera opción, me quedaban dos posibilidades: un DS1307 o un DS3132, ambos similares en cuanto a prestaciones. La decisión de escoger el DS3231 y no el DS1307 fue que, en el primero, las patillas de conexión ya venían soldadas al propio componente, mientras que, en el segundo, había que soldarlas manualmente, por lo que decidí que no valía la pena ponerse a soldar patillas cuando el precio de ambos difería en unos pocos euros. Ahora bien, ¿cómo funciona un RTC? Un RTC permite obtener mediciones de tiempo en las unidades temporales que empleamos habitualmente. Están formados por un resonador de cristal integrado con la electrónica necesaria para contabilizar el paso del tiempo. En mi caso, el DS3231 incorpora una pila para mantener el valor del tiempo en caso de que se pierda la alimentación. Una ventaja más a favor del DS3231 frente al DS1307 es que la precisión del primero es muy superior a la del segundo, lo que equivale a un desfase máximo de un segundo cada seis días del DS3231, frente a uno o dos minutos al día de desfase del DS1307.

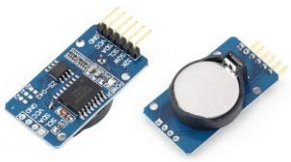

Imagen 6. RTC DS3231. https://www.luisllamas.es/reloj-y-calendario-en-arduino-con-los-rtc-ds1307-y-ds3231/

Siguiendo con la arquitectura hardware, el siguiente componente de que está provisto nuestro sistema es de un interruptor gracias al cual es posible la desconexión de nuestro sistema durante un determinado tiempo simplemente manteniéndolo apretado durante cinco segundos. Al igual que con los leds, también está provisto de una resistencia para evitar que nuestro componente sufra algún tipo de daños.

Imagen 7. Interruptor. https://www.luisllamas.es/reloj-y-calendario-en-arduino-con-los-rtc-ds1307-y-ds3231/

Para la incorporación de nuestros distintos componentes, así como de los circuitos necesarios para hacerlos funcionar, se ha utilizado una protoboard, en la cual se han ido realizando los distintos conexionados para ir uniendo, uno a uno, todos y cada uno de los componentes que hemos comentado a lo largo de este subapartado. Una protoboard (también llamada placa de pruebas) es un tablero con orificios, los cuales se encuentran conectados eléctricamente entre sí. Gracias a ello, es posible conectar los componentes electrónicos y los cables.

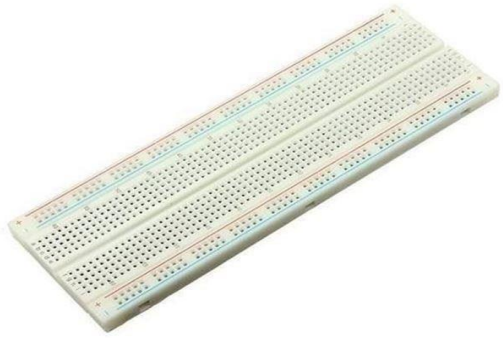

Imagen 8. Protoboard. https://www.e-ika.com/protoboard-de-830-pines

Para realizar la conexión entre la protoboard y la placa Arduino ha sido necesaria la utilización de cables, que han permitido la conectividad entre las distintas entradas/salidas de la placa y los distintos componentes. Aunque realmente, en un sistema comercial, en lugar de una protoboard, debería utilizarse una placa, especialmente construida, que incorporará todos los dispositivos.

Imagen 9. Cableado. https://www.dynamoelectronics.com/accesorios-para-arduino/113-cable-montaje-protoboard-o-arduino-10cm.html

4.2. Diseño detallado

En este nuevo apartado, se va a describir el código que se ha implementado para poder llevar a cabo la solución descrita. Partimos de la base de que el lenguaje de programación de Arduino está basado en C++. La estructura de un sketch de Arduino (sketch es como se define a un programa de Arduino) tiene, al menos, dos partes diferenciadas, las cuales son obligatorias: setup() y loop(). La primera, será la parte que se encarga de recoger la configuración y la segunda será la parte que contenga el programa que queremos que se ejecute cíclicamente. Anterior a setup(), podemos añadir comentarios de cómo será el funcionamiento de nuestro programa y, también, la declaración de variables.

El código desarrollado en este sketch consta de las tres partes diferenciadas en el párrafo anterior. La primera parte, como bien hemos dicho, contendrá la declaración de las variables. En mi caso, contiene la declaración del reloj de tiempo real (RTC), dos arrays de String con los días de la semana y los meses del año y el resto son declaraciones de las entradas y salidas que se usarán en la placa Arduino. Finalmente, las variables arriba y abajo, son dos contadores para saber cuándo la persiana está arriba o abajo del todo. Mediante estas variables, lo que conseguimos es que el programa no se quede en un bucle, y que, una vez pasados un máximo de 25 segundos, la placa vuelva a su estado normal. A continuación, podemos ver una imagen en la que se aprecia lo descrito anteriormente y en la que se comentan una serie de líneas para clarificar el código.

```
RTC_DS3231 rtc;
String daysOfTheWeek[7] = { "Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo" };
String monthsNames[12] = { "Enero", "Febrero", "Marzo", "Abril", "Mayo",  "Junio", "Julio","Agosto","Septiembre","Octubre","Noviembre","Diciembre" };

const int ledPin1 = 13;   //El LED rojo actúa como el motor que baja. Conectado a la salida 13
const int ledPin2 = 12;   //El LED verde actúa como el motor que sube. Conectado a la salida 12
const int ldrPin = A0;    //El LDR está conectado a la entrada A0
const int buttonPin = 11; //El pulsador está conectado a la salida 11
int arriba = 0;           //Contador para saber si la persiana está arriba del todo
int abajo = 0;            //Contador para saber si la persiana está abajo del todo
```

Imagen 10. Zona declarativa.

Tal y como se puede observar en la imagen anterior, la salida 13 está conectada al led rojo, la salida 12, al led verde y la salida 11 al interruptor. Por otro lado, tenemos la entrada cero en la cual conectamos el LDR.

Continuemos ahora con la zona del setup(). Esta función se ejecuta en Arduino cada vez que se enciende la placa de Arduino o cuando pulsamos la tecla de Reset. Se encarga de la inicialización de periféricos, comunicaciones, variables, etc. En esta función nos encontramos con que es necesario establecer la velocidad de datos en bits por segundo (baudios) para la transmisión de datos, cosa que le indicamos con la línea Serial.begin(9600). Seguidamente, le asignaremos un output o un input a las variables indicadas en la zona declarativa, según si queremos que actúe como entrada o salida en nuestro programa. El siguiente fragmento de código se realiza por si el RTC no se ha puesto en marcha, que nos muestre un aviso indicando que el componente no ha sido encontrado. Finalmente, si tenemos una pérdida de alimentación, fijaremos el RTC a la fecha y hora en la cual se hizo la compilación. Toda esta explicación la vamos a poder visualizar en la siguiente imagen. Como en la anterior, disponemos de una serie de comentarios a modo de clarificación.

```
void setup() {
  Serial.begin(9600);       //Abre el puerto serie y fija la velocidad en baudios para la transmisión de datos en serie. El valor típico de velocidad para comunicarse con el ordenador es 9600,
                            //aunque otras velocidades pueden ser soportadas.
  pinMode(ledPin1, OUTPUT);  //Se inicializa el LED 13 como salida
  pinMode(ledPin2, OUTPUT);  //Se inicializa el LED 12 como salida
  pinMode(ldrPin, INPUT);    //Se inicializa el LDR A0 como entrada
  pinMode(buttonPin, INPUT); //Se inicializa el pulsador 11 como entrada

  if (!rtc.begin()) {
    Serial.println(F("No se ha encontrado el RTC"));
    while (1);
  }

  // Si se ha perdido la corriente, fijar fecha y hora
  if (rtc.lostPower()) {
    // Fijar a fecha y hora de compilacion
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
}
```

Imagen 11. Función setup().

Por último, vamos a ver la zona del loop(), la cual se ejecuta de forma continuada y es donde se realiza lo más pesado del código. En mi caso, lo primero que hago es una lectura del estado del LDR y del interruptor, además de fijar el RTC a la fecha y hora actual. Para poder entender qué es lo que hace el código, voy a adjuntar un par de imágenes y, a partir de éstas, iré explicando cada una de las líneas del loop. La primera de ellas es la siguiente:

```
void loop() {
  int ldrStatus = analogRead(ldrPin);   //Lee el valor del LDR
  int buttonStatus = digitalRead(buttonPin);
  DateTime now = rtc.now();

  //Contra más alto sea el valor, más brillo (es decir, más luz hay). Si pongo >= 800 necesita menos oscuridad para encender el LED verde que si pongo >= 300
  if(ldrStatus < 600){
    digitalWrite(ledPin2, HIGH);              //Enciende LED verde
    digitalWrite(ledPin1, LOW);              //Apaga LED rojo
    arriba++;
  }
  else if(ldrStatus > 750){
    digitalWrite(ledPin1, HIGH);              //Enciende LED rojo
    digitalWrite(ledPin2, LOW);              //Apaga LED verde
    abajo++;
  }
  else if(ldrStatus >= 600 && ldrStatus <= 750){
    digitalWrite(ledPin1, LOW);              //Apaga LED rojo
    abajo = 0;
    digitalWrite(ledPin2, LOW);              //Apaga LED verde
    arriba = 0;
  }
  if(arriba >= 5){
    digitalWrite(ledPin2, LOW);              //Apaga LED verde
  }
  if(abajo >= 5){
    digitalWrite(ledPin1, LOW);              //Apaga LED rojo
  }
```

Imagen 12. Función loop() 1.

En ella vemos que, además de lo citado anteriormente, se hacen una serie de comparaciones mediante la estructura condicional if. En la primera estructura if, lo que se va a realizar es una comparación entre el valor que nos está leyendo nuestro LDR y los valores de luminosidad que nosotros deseamos tener en nuestra estancia. En consecuencia, si nuestra lectura realizada es menor que el valor mínimo de luminosidad que deseamos, debemos poner en marcha el led verde (el cual hace la simulación del motor de la persiana subiendo hacia arriba) y detener el led rojo (el cual hace el movimiento inverso al led verde). Una vez hecho esto, sumaremos uno al contador que nos habíamos inicializado anteriormente para saber si la persiana está arriba del todo. De forma contraria, si tenemos demasiada luz, lo que deberemos hacer es encender el led rojo y apagar el led verde, así como aumentar en uno el contador de descenso para saber cuándo ha llegado abajo del todo la persiana. De esta forma, haríamos la simulación de la persiana descendiendo. Cuando no se cumplan ninguna de estas dos condiciones querrá decir que estamos en el rango óptimo de luminosidad, por lo que deberemos detener ambos motores y pondremos los contadores de subida y bajada a cero. Hemos determinado que el máximo tiempo que la persiana está subiendo o bajando es de 25 segundos. Como veremos posteriormente, la lectura del LDR se realiza cada cinco segundos. El añadir los contadores es, simplemente, para hacernos a la idea de cuándo habrá llegado seguro arriba o abajo la persiana (aunque se puede dar el caso de que llegue antes) y detener los motores para que no estén continuamente haciendo el movimiento de subida o bajada y puedan permanecer en reposo.

Las siguientes instrucciones condicionales hacen referencia a los contadores de subida y de bajada. En ellas vemos que, si el contador de subida es mayor o igual que cinco, querrá decir que es seguro que la persiana está arriba del todo (lectura de LDR cada cinco segundos

multiplicado por el valor del contador, que es cinco o superior, tenemos los 25 segundos, que es el tiempo máximo que tarda la persiana en llegar arriba del todo, suponiendo el peor caso, que es cuando la persiana está bajada completamente), por lo que deberemos detener el motor de subida, apagando, de esta forma, el led verde. El caso contrario equivale a cuando el contador de bajada sea mayor o igual que cinco, con lo que deberemos apagar el led rojo, simulando la desconexión del motor de bajada.

Una vez claro el funcionamiento de subida y bajada de las persianas, es hora de adjuntar la siguiente imagen.

```
if(now.hour() == 19 && now.minute() == 01){
  digitalWrite(ledPin2, LOW);              //Apaga LED verde
  digitalWrite(ledPin1, HIGH);              //Enciende LED rojo
  delay(2500);
  digitalWrite(ledPin1, LOW);              //Apaga LED rojo
  for (int i = 0 ;  i  <  16 ; i++){
    LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF);                    //Si son las 19:01, el programa se detiene durante un minuto
  }
}
if(buttonStatus == 1){
  digitalWrite(ledPin1, LOW);              //Apaga LED rojo
  digitalWrite(ledPin2, LOW);              //Apaga LED verde
  LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);                  //Cuando pulse el pulsador, el programa se detiene
}
delay(5000);                               //Hace una lectura del LDR cada 5 segundos
}
```

Imagen 13. Función loop() 2.

Ahora es cuando entra en escena la bajada y subida automática de las persianas. Vemos de nuevo, un par de instrucciones condicional if. En la primera de ellas, se está realizando una lectura de la hora y los minutos, de tal forma que si se hace la hora determinada (en este caso las 19:01), la persiana descenderá hasta que esté cerrada completamente, y permanecerá en este estado durante un minuto. Vemos que, en el peor de los casos, la persiana estaría subida del todo, por lo que la hacemos descender durante 25 segundos, que es el tiempo máximo que tarda de la posición fin a la posición inicio y, una vez llegada a la posición deseada, dejamos nuestra placa Arduino en modo reposo.

El poder dejar nuestra placa en este modo es gracias al modo sleep. Como podemos visualizar en la imagen, se pasa una función powerDown() con tres parámetros, los cuales son: el tiempo que queremos que esté durmiendo nuestra placa (el máximo son ocho segundos), el ADC_OFF, que apaga los convertidores analógico a digital y el BOD_OFF, el cual apaga el circuito de Brown Out Detection, que es el que se encarga de detectar niveles de tensión peligrosamente bajos. Entonces, ¿cómo es posible que dejes la placa en modo reposo durante un minuto si el tiempo máximo que la podemos dejar durmiendo son ocho segundos? Bien, aquí es donde cobra sentido el bucle for realizado justo antes. Lo que conseguimos con esto es hacer un bucle de 15 periodos de cuatro segundos cada periodo. En conclusión, 15 periodos multiplicados por cuatro segundos, que es lo que dura cada periodo, nos da un total de 60 segundos, por lo que ya tenemos nuestro Arduino durmiendo durante un

minuto. Si quisiéramos dejar nuestro Arduino con la persiana cerrada durante ocho horas (por ejemplo, de 23:00 a 07:00), deberíamos hacer un bucle de 7200 periodos de cuatro segundos cada periodo.

Para finalizar la explicación de nuestro código, paso a comentar el último if con el que nos encontramos. Este bloque de código realiza la función de detener el programa (en este caso durante ocho segundos) de forma manual, simplemente mediante la pulsación de un interruptor durante un periodo máximo de cinco segundos. Lo que se consigue es dejar la persiana al nivel actual durante el tiempo deseado.
Como hemos visto en estos dos últimos bloques de código, es muy trivial el uso del tiempo en las placas de Arduino, aunque bien es cierto que debemos de tener mucho cuidado. Hay una opción en Arduino que es el SLEEP_FOREVER, la cual se pasa como primer argumento de la función power_Down(). Esta función únicamente se sale con una interrupción hardware o COMM, por lo que, si se usa sin conocimiento, tu Arduino puede quedarse en estado de reposo sin poder hacer que éste despierte.

Tal y como se ha comentado con anterioridad, la última línea del bloque representa el tiempo que se ha de esperar entre ejecución y ejecución, que en este caso es de cinco segundos.

4.3. Tecnología utilizada

Nos adentramos en un nuevo apartado para comentar, esta vez, las herramientas que hemos utilizado para llevar a cabo nuestro proyecto. Primero que nada, el sistema en el cual se ha desarrollado la solución ha sido un Windows 7 de 64 bits. En él, nos instalamos el ambiente de desarrollo Arduino, que es donde hemos ido implementando nuestro código. En la siguiente imagen vemos el ambiente del que hemos hablado:

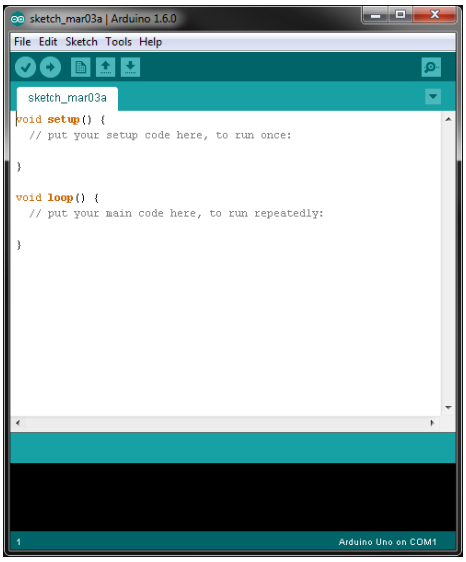

Imagen 14. Ambiente Arduino.
https://aprendiendoarduino.wordpress.com/2016/03/29/entorno-de-programacion-de-arduino-ide/

El lenguaje de programación que utiliza Arduino es C++, el cual es una extensión del exitoso lenguaje de programación C. Algunas de las características de este lenguaje son:

- Está asociado al sistema operativo UNIX.
- Trata con objetos básicos tales como caracteres o números, aunque también trabaja con bits y direcciones de memoria.
- Es portable.
- Es utilizado para la programación de sistemas.

La intención de C++ fue crear mecanismos que permitieran la manipulación de objetos. Más tarde, se añadieron facilidades de programación genérica que, sumados a los otros dos paradigmas que estaban admitidos (programación estructurada y programación orientada a objetos) hicieron que este lenguaje fuera denominado lenguaje multiparadigma.

Aunque el lenguaje de programación de Arduino esté basado en C++, utiliza una serie de referencias, las cuales se pueden encontrar en el siguiente enlace: https://www.arduino.cc/reference/en/

Hablando de forma más específica para mi implementación, he tenido que hacer uso de una serie de librerías que van a ser comentadas seguidamente. La primera librería necesaria para implementar el proyecto ha sido la librería LowPower, gracias a la cual ha sido posible hacer que nuestra placa Arduino entre en modo de reposo, mediante el uso de la función LowPower.powerDown(). De esta manera, mediante la instrucción include, podemos agregar la librería deseada a nuestro proyecto. Esta librería está disponible en: https://www.arduinolibraries.info/libraries/low-power

La siguiente librería utilizada ha sido RTClib, la cual me ha permitido acceder a los métodos y funciones necesarias para obtener la fecha y hora actual, así como para ajustar estos parámetros. Nuevamente, mediante un include, podemos hacer uso de ella.

```
#include <LowPower.h>
#include <RTClib.h>
```

Imagen 15. Incluir librerías.

Ambas librerías no vienen instaladas por defecto por lo que, para poder hacer uso de ella, es necesario incluirlas en nuestro programa. Una vez descargadas las librerías a través de los enlaces proporcionados, nos iremos a la ventana donde estemos desarrollando nuestro código. De esta manera, nos dirigiremos a *Programa -> Incluir Librería -> Añadir librería .ZIP…* De esta manera, al seleccionar nuestros archivos .ZIP, habremos instalado las librerías y ya podremos hacer uso de ellas mediante las

instrucciones include, ya explicadas anteriormente. A continuación, se muestra una imagen de lo explicado en este último párrafo.

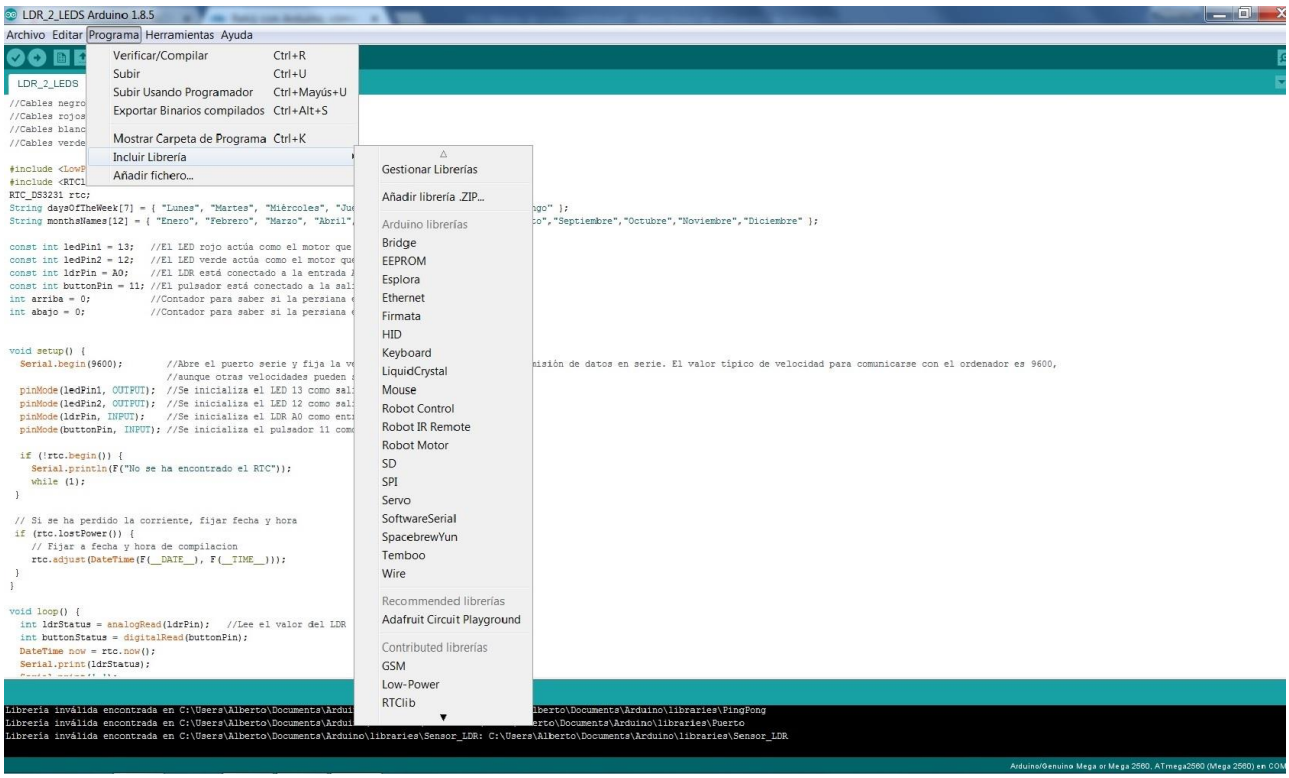

Imagen 16. Librerías Arduino.

Como se puede comprobar en la imagen, las dos últimas librerías que aparecen en la imagen son las que hemos estado citando a lo largo de este apartado. En ella también se pueden ver todas las librerías que viene por defecto instaladas.

5. Desarrollo de la solución propuesta

En este nuevo capítulo, vamos a ir comentando las distintas fases por las que ha ido pasando nuestro proyecto, las ideas que hemos ido añadiendo y los problemas con los que nos hemos ido encontrando durante el recorrido hacia la solución final.

Como punto de partida, deseábamos desarrollar un sistema que nos permitiera controlar la iluminación de una estancia. No sabíamos qué componentes eran necesarios, pero sí sabíamos que nuestra plataforma de desarrollo utilizada iba a ser Arduino, por el simple hecho de abaratar costes, ya que queríamos que esta idea estuviera al alcance de todos. Después de indagar sobre qué era necesario para llevar a cabo el trabajo, comprendí que la mejor forma de controlar la iluminación era mediante la lectura de un LDR. Una vez conseguido el LDR, hice una primera toma de contacto, en la cual conecté el LDR y un led a la placa, que lo único que hacía era que el led se encendiera o apagara en función de si tapaba el LDR o no, únicamente para testear cómo sería el funcionamiento de este circuito. El esquema del circuito lo saqué del siguiente vídeo: https://www.youtube.com/watch?v=4fN1aJMH9mM&index=4&list=LL9C0COIUhTbE2ldv1LfcA_w&t=0s. El esquema de este circuito sería el mostrado en la siguiente imagen:

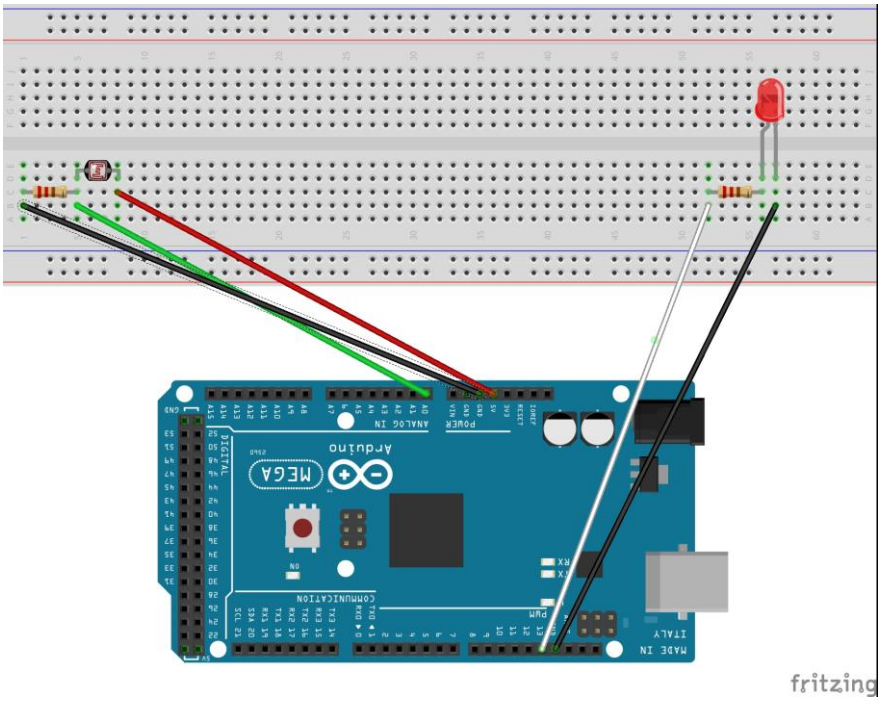

Imagen 17. Esquema primer circuito Arduino.

Una vez corroborado que el circuito desarrollado funcionaba a la perfección, tuve la idea de añadir un segundo led, así ya podría emular el movimiento de subida y bajada de la persiana, encendiendo un led u otro. Tras añadir el segundo led, ajusté mi código y, efectivamente, dependiendo

de si tapaba el LDR o no, brillaba un led o brillaba otro. Esto pintaba bien ya que había conseguido emular la subida y bajada de la persiana conectando dos leds a mi placa. Tras un tiempo de pensar en lo que había hecho, me di cuenta de que, en mi circuito, la persiana siempre estaba, o subiendo o bajando, y eso no podía ser. Tras una serie de rectificaciones en el código, conseguí el efecto deseado: que la persiana suba si no hay suficiente luz, que baje si ocurre el efecto contrario o que se mantenga en ese estado si se encuentra en un rango de luz óptimo.

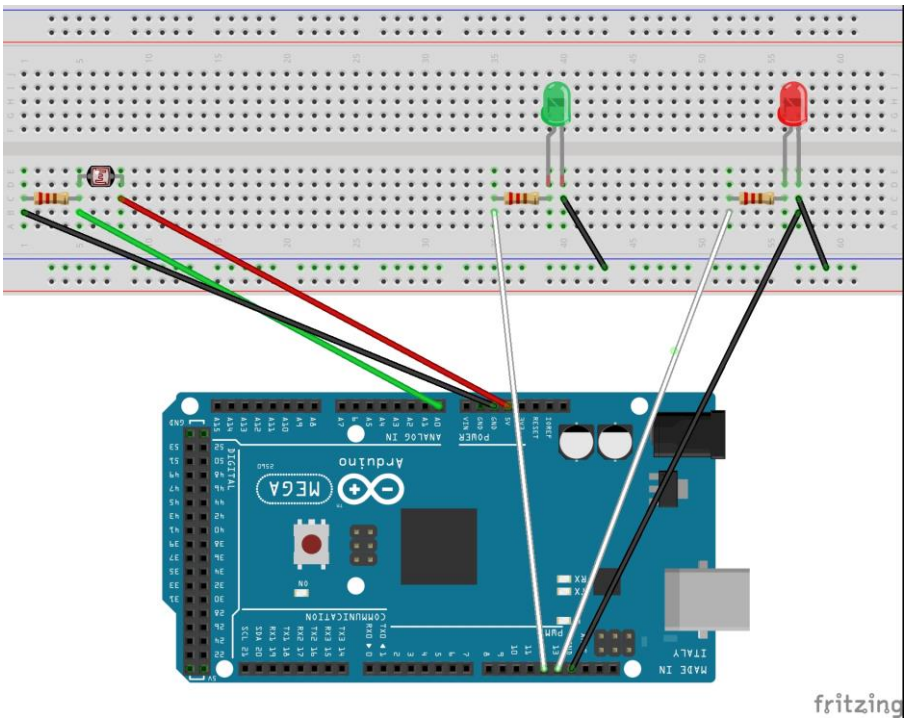

Imagen 18. Esquema segundo circuito Arduino.

Tuve verdaderas dudas en cómo debía subir o bajar la persiana. Estuve haciendo muchas pruebas viendo qué me devolvía la lectura del LDR dependiendo del día que había salido. Sabía que, si la lectura me devolvía cero o uno, era porque el LDR estaba tapado completamente, por lo que sería de noche y la persiana estaría bajada por completo. Si me devolvía un valor aproximado de 500, el día había salido nuboso. Mientras que, si el valor era de 800, era porque era un día soleado. Tras hablar con mi tutor, caí en la cuenta de que estaba cometiendo un error. Era necesario trabajar con rangos y no con valores absolutos. Gracias a eso, tras volver a tomar una serie de lecturas, comprendí qué rangos eran los óptimos.

Una vez corregido el código y viendo que el sistema trabajaba a la perfección, y tras comentar una serie de propuestas con mi tutor (se llegó a comentar la implantación de un sensor de humedad gracias al cual la persiana se cerrara completamente si el tiempo era lluviosos, aunque no

prosperó porque nos dimos cuenta de que no era realmente necesario para lo que yo quería implementar) ambos llegamos a la conclusión de que sería buena idea implementar una funcionalidad que permitiera la subida y bajada automática de la persiana en función de la hora. Me puse manos a la obra. Lo primero que necesitaba era medir el tiempo de la placa. Vi que había una librería que contenía el propio Arduino: la librería Time. Tras instalarla en mi programa y después de hacer una serie de pruebas, observé que no me funcionaba. Tras indagar en el por qué, observé que había gente a la que le pasaba lo mismo que a mí y no habían podido solucionar el problema, por lo que, la solución que se proponía era la de añadir un RTC al proyecto. Hice una comparación de qué RTC me sería válido para mi proyecto y por qué (se habló de eso en el capítulo cuatro) y, al final, obtuve un DS3231. Tras implantarlo en mi proyecto, instalé la librería específica para ese modelo de RTC y ya pude obtener la hora fácilmente. Gracias al siguiente vídeo me fue muy sencillo realizar el conexionado del RTC: https://www.youtube.com/watch?v=RoSVrVVMy0c&list=LL9C0COIUhTbE2ldv1LfcA_w&t=4s&index=2.

De forma paralela a la introducción del reloj de tiempo real, surge la idea de instalar un interruptor mediante el cual podamos desconectar el sistema de forma voluntaria. El circuito lo obtuve del siguiente vídeo: https://www.youtube.com/watch?v=JmUyNgGHdVE&list=LL9C0COIUhTbE2ldv1LfcA_w&t=205s&index=3. En la imagen podemos ver el esquema de conexionado de nuestra placa hasta el momento (en él aún no está implantado el RTC).

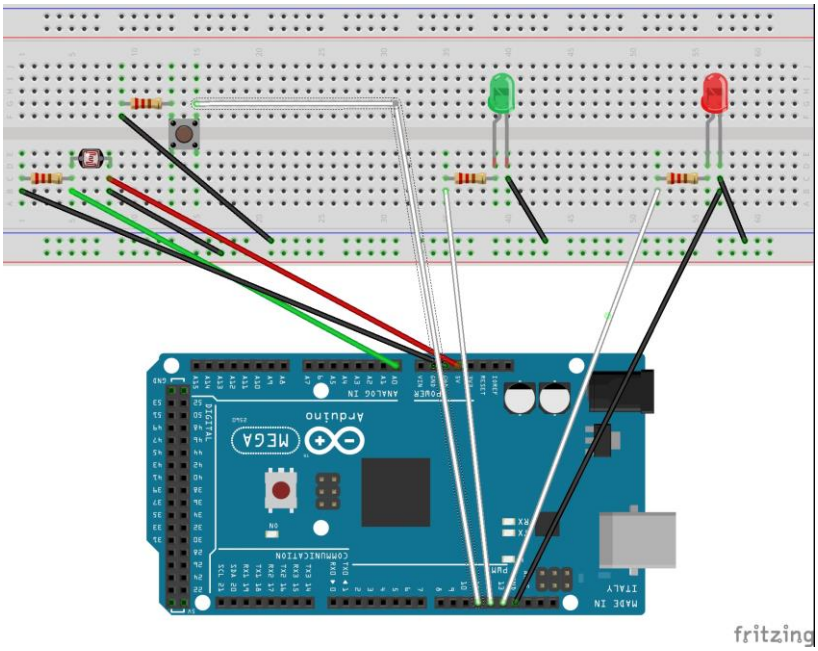

Imagen 19. Esquema tercer circuito Arduino.

Tras incluir los componentes citados anteriormente en el código del proyecto e implementando la solución para que cada uno de ellos realizara la funcionalidad deseada, llegamos a nuestro circuito final, que es el que se muestra en las siguientes imágenes.

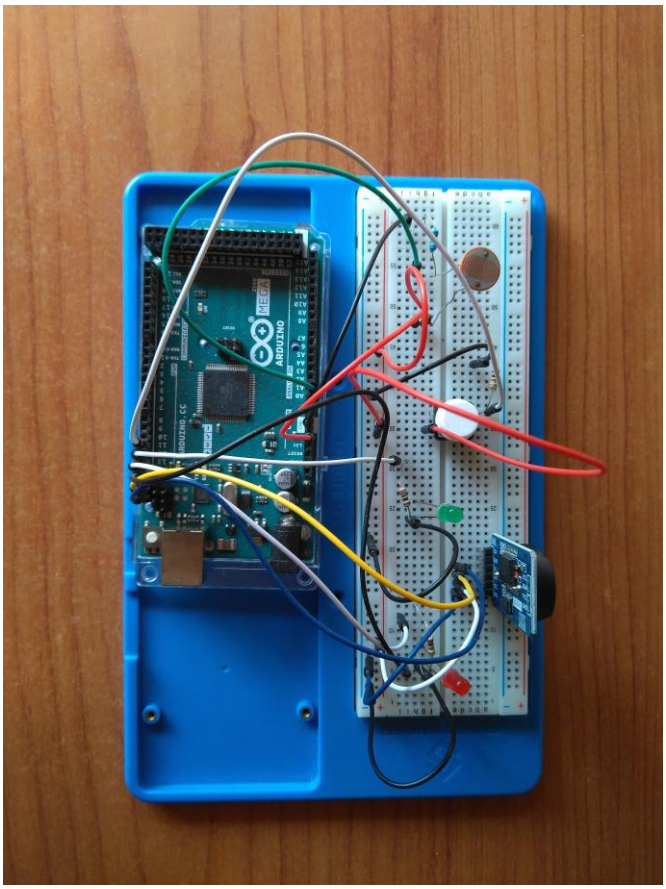

Imagen 20. Circuito final visto desde arriba.

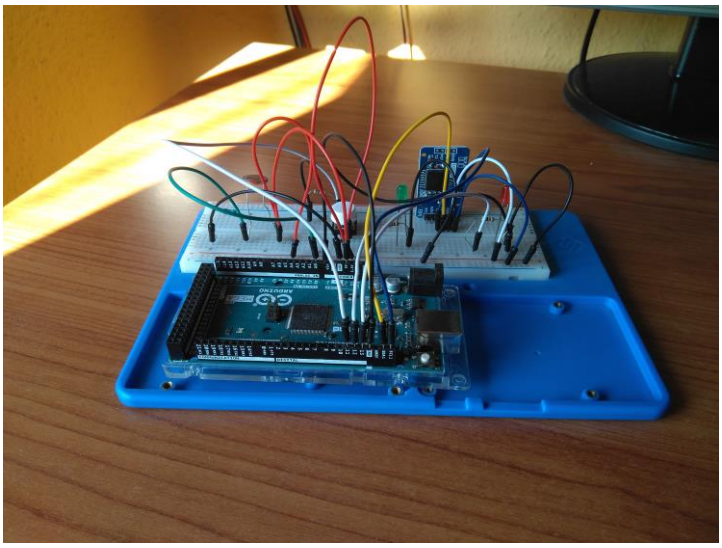

Imagen 21. Vista del circuito final desde otra perspectiva.

6. Implantación

Llegamos a un nuevo capítulo para hablar, esta vez, de los pasos necesarios a seguir para poder realizar la implantación de nuestro sistema.

Primero que nada, habría que ver cuál es la estancia escogida para hacer funcionar nuestro sistema. Es importante elegir la ubicación de la placa, puesto que la entrada de luz va a depender de dónde se sitúe el LDR. Tiene que ser un lugar despejado para que no haya nada que pueda interferir en las lecturas realizadas por el fotoresistor. Una vez elegido el lugar, sería necesario hacer una serie de modificaciones en nuestro circuito, ya que, deberíamos cambiar los leds con los que estamos trabajando ahora, por los motores de las persianas. Esto no conlleva ningún problema, gracias a la utilización de nuestro relé, que permite conectar cualquier tipo de dispositivo, ya sea un led o motores de distintos tipos.

Una vez hecho esto, sería necesario testear el funcionamiento del sistema durante un período de tiempo, puesto que no en todas las estancias entra la luz de igual forma. Si fuera necesario, se realizaría un ajuste de los valores que vimos en la imagen 12 para adecuar el nivel de luminosidad que entra en cada momento a las exigencias del cliente.

Otra modificación que tendríamos que adaptar al cliente sería la hora a la que desea que la persiana baje por completo y el tiempo que se desea que ésta permanezca en ese estado (es posible que haya variación entre un consumidor y otro).

Finalmente, otro punto a tener en cuenta durante la implantación sería el tiempo que se desea que nuestro sistema esté en estado de reposo una vez hemos pulsador el interruptor.

Hay que tener en cuenta que, nuestro sistema es, meramente, un simple prototipo que podría salir al mercado por el impacto que puede tener hacia el consumidor debido a su bajo coste. Es necesario aclarar, que, para que este proyecto viera la luz, sería necesario hacer el dispositivo cerrado, de manera que no haya ningún cable a la vista y que, lo único a lo que se tenga acceso, sea a los pines de conexión de los motores de las persianas con éste. También sería necesario dotar de cableado al LDR para que no sea necesario que el fotoresistor y el sistema tengan que estar en el mismo lugar, sino que habría que dar la facilidad de poder instalar el LDR en el lugar que se desee.

7. Pruebas

A lo largo del capítulo siete se va a comentar lo que ha sido la etapa de testeo. Para ello, vamos a ir explicando qué pruebas han sido llevadas a cabo para corroborar el correcto funcionamiento de nuestro dispositivo.

Si bien es cierto que el proyecto no lleva una gran cantidad de software tras de sí, ya que, la mayor parte del proyecto ha consistido en hacer que los distintos componentes hardware se conecten de forma que permita la interacción de todos ellos para la unificación del proyecto, han sido necesarias una serie de pruebas, gracias a las cuales hemos podido verificar que todo funciona según nuestra idea prevista. El entorno de programación de Arduino permite imprimir una serie de mensajes en lo que para ellos se denomina el monitor serie. La función de éste es la de mostrar los datos que nuestra placa Arduino nos envía, utilizando, para el envío, el puerto serie. Pero para imprimir los datos no basta, únicamente, con conectar nuestra placa al ordenador. Es necesario hacer uso de una serie de instrucciones que nos imprimirán el resultado por pantalla. Esta instrucción es Serial.print(data), donde data puede ser el dato de alguno de nuestros dispositivos que almacenamos en una variable, como, por ejemplo, la lectura que está haciendo el LDR, aunque también puede ser un string que muestre el texto que nosotros le hayamos insertado.

Bueno, una vez introducido qué es necesario para efectuar las pruebas, vamos a lo realmente interesante, que es el exponer una serie de ejemplos e imágenes de las pruebas que se han llevado a cabo. La primera prueba que realicé fue la de saber qué valor me devolvía la lectura del LDR. Y, para ello, lo que tuve que hacer, es acercar mi placa a una ventana e ir midiendo qué valores me iba devolviendo. Claro está que esta prueba no puedes realizarla en un único día o en la misma franja horaria, sino que necesitas recabar datos de cuantos más días mejor. Ésta ha sido, bajo mi punto de vista, la más importante de todas las pruebas, puesto que ha sido la que me ha permitido obtener los rangos de luz óptimos, ya que se han medido niveles de luminosidad cuando el día ha sido soleado y nuboso y viendo la diferencia que había entre una franja horaria y otra.

La siguiente prueba que se realizó fue la de saber la fecha y hora de la placa. Fue importante saber cómo se mostraban los datos respectivos a la hora y los minutos, ya que, como se vio anteriormente, se necesitaban ambos datos para que, a la hora y los minutos fijados, nuestro sistema bajara la persiana y la mantuviese durante el tiempo estimado. Así pues, caí en la cuenta de que, tanto las horas como los minutos, eran un mero entero. Me sirvió, también, para verificar que, realmente, la placa pasaba a un estado de reposo una vez llegábamos a la hora establecida.

Y, por último, otra prueba necesaria fue la de saber la lectura que se hacía del interruptor. En un primer momento, mi intención era la de realizar una pulsación

al interruptor y, de esta forma, que su estado pasara de ser cero a ser uno (recordemos que la finalidad del pulsador era la de detener el dispositivo por voluntad propia). Tras ver que el dispositivo no respondía a la funcionalidad programada, opté por imprimir por pantalla el estado del interruptor. Tras comprobar que, cuando pulsaba, no cambiaba de estado, me di cuenta de que la lectura, tanto del LDR como del estado del botón, se realiza cada cinco segundos, por lo que, si dejaba el interruptor pulsado durante un máximo de cinco segundos, su estado pasaba de cero a uno y era entonces cuando se detenía la ejecución y la placa se quedaba en reposo durante el tiempo que se indica.

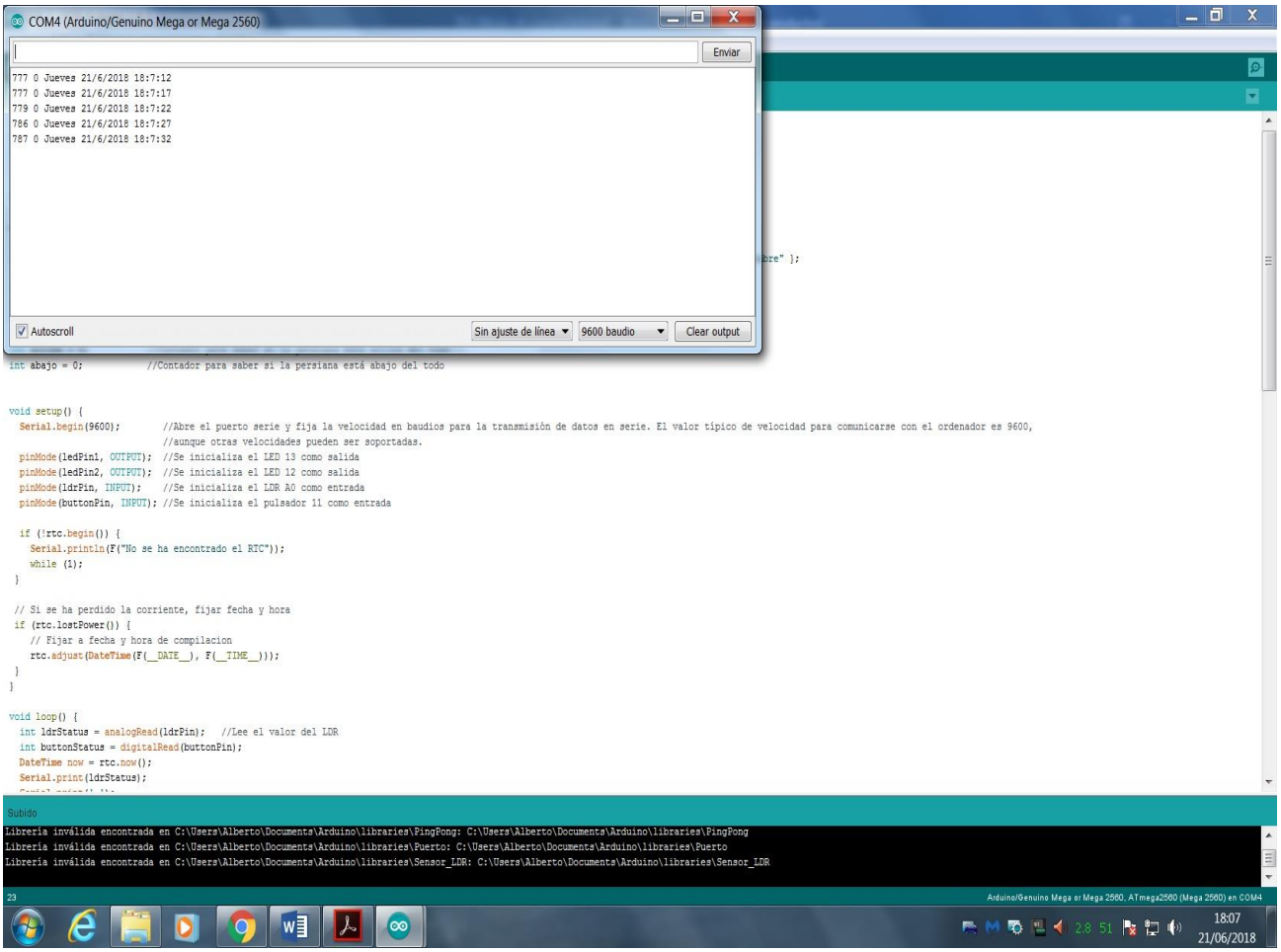

Imagen 22. Monitor serie Arduino.

Como se puede ver en la imagen, la impresión que nos está realizando es la del estado del LDR (en este caso, los valores oscilan entre 777 y 787), seguido por el estado del interruptor (en este caso es un cero ya que no se ha mantenido pulsado). Finalmente, podemos ver el día de la semana en el que estamos, la fecha y la hora. Podemos comprobar que el día y la hora de la placa concuerdan con el día y la hora del sistema. También podemos

observar en la parte superior izquierda de la imagen el puerto por el que se realiza el envío de los datos y la placa utilizada.

En la siguiente imagen se muestra el código necesario para obtener esta secuencia de texto de forma que sea legible.

```
Serial.print(ldrStatus);
Serial.print(' ');
Serial.print(buttonStatus);
Serial.print(' ');
Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
Serial.print(' ');
Serial.print(now.day(), DEC);
Serial.print('/');
Serial.print(now.month(), DEC);
Serial.print('/');
Serial.print(now.year(), DEC);
Serial.print(' ');
Serial.print(now.hour(), DEC);
Serial.print(':');
Serial.print(now.minute(), DEC);
Serial.print(':');
Serial.print(now.second(), DEC);
Serial.println();
```

Imagen 23. Instrucción Serial.print().

En ella podemos ver lo que se ha explicado al principio de este apartado, y es que, a la función, se le puede pasar tanto una variable como un texto fijo.

8. Conclusiones

Llegamos a uno de los capítulos más importantes de todo el documento para hablar, esta vez, de las conclusiones que se han obtenido una vez nuestro sistema ha sido desarrollado por completo.

Para ello, debemos basarnos en los objetivos que enumeramos al principio del trabajo. De esta forma, sabremos si realmente hemos conseguido que nuestro sistema realice las funciones pertinentes.

Bajo mi punto de vista, ha sido una experiencia muy enriquecedora puesto que, aunque ya había desarrollado algún que otro proyecto mediante esta plataforma, me ha permitido descubrir una serie de dispositivos que desconocía totalmente, así como de una serie de utilidades que jamás habría podido imaginar que la plataforma de Arduino me podría brindar.

A lo largo del documento hemos podido comprobar que nuestro sistema funciona de manera autónoma, sin necesidad de hacer nada más que elegir la ubicación adecuada para que el haz de luz que penetre sobre nuestro sistema haga el resto, por lo que nuestro primer objetivo ha quedado completamente desarrollado.

También hemos podido comprobar que, tanto los dispositivos utilizados para la realización del sistema como la metodología de programación en que se ha basado este proyecto, pueden encontrarse fácilmente y, en el caso del hardware utilizado, cualquier tienda de electrónica que se precie puede conseguir dichas piezas. Por lo tanto, nuestro objetivo de hacer que cualquier persona pueda desarrollar este sistema queda cubierto.

Y finalmente, uno de los temas que más nos preocupaba era el de la eficiencia energética, cosa que hemos visto que se puede solucionar mediante la puesta en estado de reposo de nuestra placa durante un tiempo que sabemos que no va a ser necesario que actúe.

En definitiva, no puedo estar más contento de haberme embarcado en este proyecto, ya que, desde el primer momento, me he sentido muy a gusto desarrollando el sistema. He visto como, con esfuerzo y perseverancia, se han ido incluyendo al circuito todas las piezas necesarias para su correcto funcionamiento. He comprobado, de buena mano, que he sido capaz de anteponerme a los problemas que me han ido surgiendo a lo largo del desarrollo, tanto en el tema componentes como en el tema programación (el camino no es fácil, pero cuando se tienen ganas, se puede con todo). Los objetivos que teníamos en mente desde el primer momento, han sido abarcados en todos sus aspectos, cosa que resulta muy gratificante.

Resumiendo, mis sensaciones sobre el estado final del sistema desarrollado no pueden ser más satisfactorias y siento que se ha hecho un gran trabajo de principio a fin, tanto por parte mía como por parte de mi tutor.

8.1. Relación del trabajo desarrollado con los estudios cursados

Es hora de explicar la concordancia entre los estudios cursados y la temática de este proyecto.

Este TFG, dentro de lo que es el grado en informática, pertenece a la rama de ingeniería de computadores, que es la que se encarga del desarrollo de la parte de hardware, que es la rama de especialización que escogí.

Pese a que dentro de lo que es la rama en sí, no hay asignaturas ofertadas que abarquen el tema del desarrollo en Arduino, es cierto que se hace hincapié en este tipo de tecnología. Y es, gracias a ello, que mi interés por la plataforma Arduino aumentó, y, de ahí, la finalidad de escoger este proyecto.

Este proyecto combina los dos aspectos primordiales de la informática: la integración de los dispositivos hardware y el desarrollo software.

Como bien se ha hablado a lo largo de este documento, el lenguaje de programación utilizado ha sido C++. Ya dijimos en su momento que este lenguaje era una ampliación del lenguaje C, con el cual nos empapan a lo largo de la carrera y que, por suerte, acabé desarrollando en ciertas asignaturas de la rama.

Es necesario remarcar que la realización de este proyecto no habría sido posible sin los conocimientos proporcionados en la asignatura de Fundamentos Físicos de la Informática, la cual me permitió saber interpretar los circuitos eléctricos, así como seguir una serie de pautas para el correcto funcionamiento de los sistemas eléctricos y, así, evitar males mayores como puede ser el sobrecalentamiento de algún componente debido a aplicarle una tensión que no corresponde.

9. Trabajos futuros

Y llegamos al final del último punto de este trabajo para hablar, esta vez, de las ideas que han ido surgiendo a lo largo del desarrollo y que no ha sido posible implementar debido al tiempo del que disponíamos para la realización del proyecto.

La primera idea que surgió fue la de poder hacer que el propio usuario sea capaz de modificar el nivel de luminosidad que desea que entre en la estancia. Para ello, necesitaríamos implementar una funcionalidad que haga que los rangos de luminosidad varíen de acuerdo a las necesidades del consumidor. La forma correcta de hacerlo sería mediante el uso de un potenciómetro, también conocidos como resistencias variables. Según varíe la posición de éste, variará el voltaje obtenido. De esta forma, si el potenciómetro está totalmente cerrado, obtendremos el máximo voltaje posible. Por el contrario, obtendremos cero voltios si lo tenemos totalmente abierto. Dentro de los distintos tipos de potenciómetros que existen, el que nosotros usaríamos sería un potenciómetro de variación lineal, donde el ángulo de giro es directamente proporcional a la resistencia. En la siguiente imagen podemos ver un ejemplo de dicho potenciómetro.

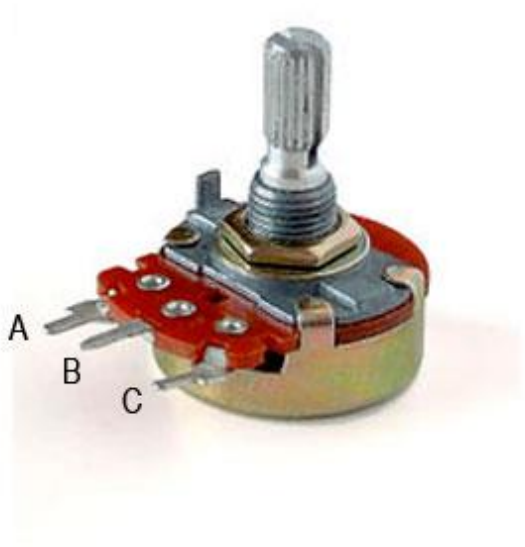

Imagen 24. Potenciómetro lineal. https://programarfacil.com/blog/el-potenciometro-y-arduino/

La modificación que deberíamos realizar en nuestro circuito sería la de sustituir la resistencia que tenemos actualmente conectada al LDR por el potenciómetro. De esta manera, el LDR tendría uno de los pines conectado a cinco voltios, el otro lo tendría conectado con el pin B del potenciómetro y el pin C de éste iría conectado a GND. Un ejemplo de conexionado sería el siguiente.

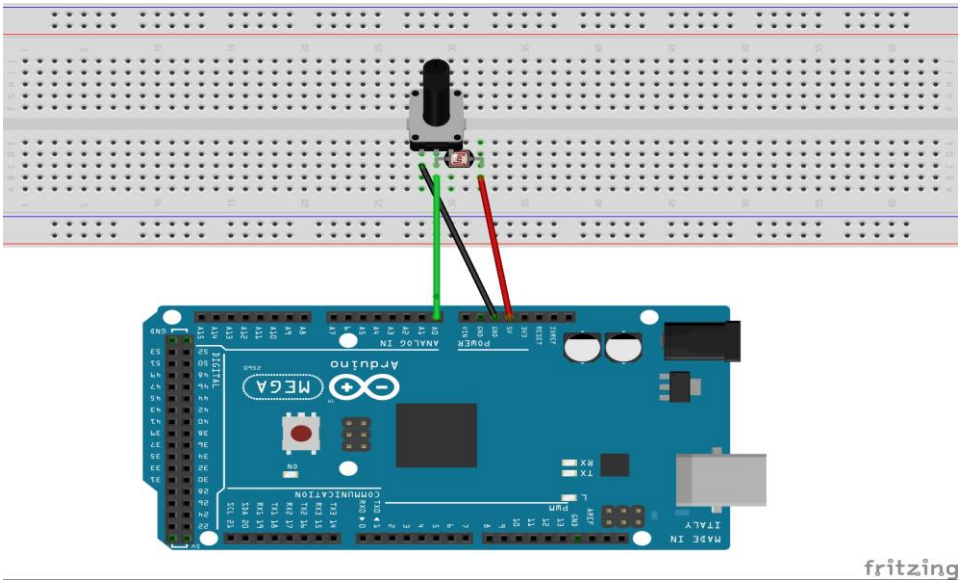

Imagen 25. Esquema conexionado potenciómetro.

En ella podemos ver que la entrada elegida sería la entrada cero, que hace referencia al cable verde. El cable rojo haría referencia a la conexión a cinco voltios, mientras que el cable negro haría referencia a la conexión a GND.

La inserción de este componente en nuestro circuito haría que los rangos de luminosidad definidos variaran en función de la posición del potenciómetro. Así, si en un momento determinado, deseamos más o menos luz, basta con mover el ángulo del potenciómetro hacia un lado u otro hasta hacer que la posición de la persiana se quede a nuestro gusto.

La siguiente idea que surgió tiene que ver con las horas a las que se desea que se suban y se bajen las persianas. Una solución que se me ocurrió, fue la de desarrollar una aplicación para móvil (válida para las principales plataformas de aplicaciones móviles) en la cual puedas insertar la hora a la que deseas que la persiana baje por completo y el tiempo que deseas que ésta se mantenga. Pero no sólo eso, gracias a esta aplicación, podrías tener un control total de tu dispositivo sin necesidad de estar en casa o en el lugar donde esté implantado el sistema, pudiendo mover la persiana a tu antojo. Esta funcionalidad puede ser muy práctica, sobre todo en los periodos vacacionales, ya que podrías programar la subida y bajada, haciendo ver que estás en casa, cuando, en realidad, estás disfrutando de tus vacaciones estivales.

También sería buena idea implantar un sensor de temperatura a nuestro circuito. De esta forma, sabrías, en todo momento (gracias a la aplicación móvil), a qué temperatura se encuentra la estancia, y poder definir una temperatura, que es la que consideras que es la adecuada para esa estancia.

De esta forma, dependiendo de la temperatura de nuestro sistema, éste dejaría entrar más o menos luz del exterior. Esto sería posible gracias al uso de un sensor de temperatura, un LM35, el cuál mide un rango de temperaturas entre -55ºC y 150ºC. El esquema de conexionado es muy similar al del potenciómetro. Tendríamos tres pines, dos de los cuales irían destinados a la alimentación y el tercero sería el que proporciona la medición en una referencia de tensión. A continuación, se muestran una serie de imágenes relacionadas con este componente.

Imagen 26. Sensor LM35. https://www.luisllamas.es/medir-temperatura-con-arduino-y-sensor-lm35/

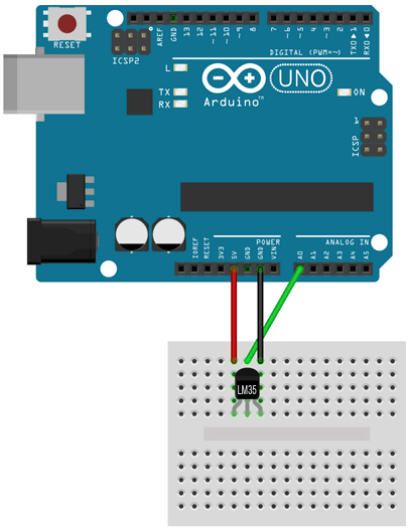

Imagen 27. Esquema conexionado LM35. https://www.luisllamas.es/medir-temperatura-con-arduino-y-sensor-lm35/

10. Referencias

I. Loxone. Persianas totalmente autónomas. Consultado el 03/06/2018 https://www.loxone.com/eses/smart-home/persianas-sombreado/

II. Distech Controls. Control de iluminación y persianas. Consultado el 03/06/2018 http://www.distech-controls.com/es/eu/products/lighting-and-shades-control/

III. Domoprac. Control del sistema de iluminación con domótica. Consultado el 03/06/2018 http://www.domoprac.com/domoteca/domoteca/integracion-domotica/control-del-sistema-de-iluminacion-con-domotica.html

IV. Schneider Electric. Soluciones en control de iluminación. Consultado el 03/06/2018 https://www.schneider-electric.com.co/documents/KNX/brochure-KNX-version-web.pdf

V. ¿Qué es Arduino? Consultado el 09/06/2018 http://arduino.cl/que-es-arduino/

VI. Arduino vs Raspberry Pi. Consultado el 09/06/2018 https://www.adslzone.net/2017/03/24/arduino-vs-raspberry-pi-cual-es-la-mejor-placa-para-iniciarse/

VII. Arduino Mega 2560. Consultado el 10/06/2018 http://arduino.cl/arduino-mega-2560/

VIII. Fotoresistor (LDR). Consultado el 10/06/2018 https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/

IX. Reloj de tiempo real (RTC). Consultado el 10/06/2018 https://www.luisllamas.es/reloj-y-calendario-en-arduino-con-los-rtc-ds1307-y-ds3231/

X. Programación Arduino. Consultado el 14/06/2018. https://aprendiendoarduino.wordpress.com/2017/01/23/programacion-arduino-5/

XI. Función setup() y loop(). Consultado el 14/06/2018. https://www.luisllamas.es/nuestro-primer-programa-en-arduino/

XII. Serial. Consultado el 14/06/2018. https://www.arduino.cc/en/Serial/Begin

XIII. Modo sleep Arduino. Consultado el 17/06/2018. https://www.prometec.net/el-modo-sleep-en-arduino/

XIV. Monitor serie. Consultado el 21/06/2018. https://playground.arduino.cc/ArduinoNotebookTraduccion/Serial

XV. Potenciómetro lineal. Consultado el 24/06/2018. https://programarfacil.com/blog/el-potenciometro-y-arduino/

XVI. Sensor LM35. Consultado el 24/06/2018. https://www.luisllamas.es/medir-temperatura-con-arduino-y-sensor-lm35/

Anexo

Código fuente

```
1  #include <LowPower.h>
2  #include <RTClib.h>
3  RTC_DS3231 rtc;
4  String daysOfTheWeek[7] = { "Domingo", "Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado" };
5
6  const int ledPin1 = 13;
7  const int ledPin2 = 12;
8  const int ldrPin = A0;
9  const int buttonPin = 11;
10 int arriba = 0;
11 int abajo = 0;
12
13 void setup() {
14    Serial.begin(9600);
15
16    pinMode(ledPin1, OUTPUT);
17    pinMode(ledPin2, OUTPUT);
18    pinMode(ldrPin, INPUT);
19    pinMode(buttonPin, INPUT);
20
21    if (!rtc.begin()) {
22        Serial.println(F("No se ha encontrado el RTC"));
23        while (1);
24    }
25
26    if (rtc.lostPower()) {
27        rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
28    }
29 }
30
31 void loop() {
32     int ldrStatus = analogRead(ldrPin);
33     int buttonStatus = digitalRead(buttonPin);
34     DateTime now = rtc.now();
35     if(ldrStatus < 600){
36         digitalWrite(ledPin2, HIGH);
37         digitalWrite(ledPin1, LOW);
38         arriba++;
39         abajo = 0;
40    }
41     else if(ldrStatus > 750){
42         digitalWrite(ledPin1, HIGH);
43         digitalWrite(ledPin2, LOW);
44         abajo++;
45         arriba = 0;
```

```
46      }
47      else if(ldrStatus >= 600 && ldrStatus <= 750){
48          digitalWrite(ledPin1, LOW);
49           abajo = 0;
50           digitalWrite(ledPin2, LOW);
51            arriba = 0;
52      }
53       if(arriba >= 5){
54           digitalWrite(ledPin2, LOW);
55       }
56       If(abajo >= 5){
57          digitalWrite(ledPin1, LOW);
58       }
59       if(now.hour() == 19 && now.minute() == 01){
60           digitalWrite(ledPin2, LOW);
61           digitalWrite(ledPin1, HIGH);
62           delay(2500);
63           digitalWrite(ledPin1, LOW);
64           for (int i = 0 ;  i  <  16 ; i++){
65               LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF);
66             }
67       }
68       if(buttonStatus == 1){
69          digitalWrite(ledPin1, LOW);
70          digitalWrite(ledPin2, LOW);
71          LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
72       }
73       delay(5000);
74  }
```