UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Aurora: diseño y programación de un automatismo para cortinas comunicado con alexa

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Saneleuterio Temporal, Rafael Luis

Tutor/a: Balbastre Betoret, Patricia

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

AURORA: DISEÑO Y PROGRAMACIÓN DE UN AUTOMATISMO PARA CORTINAS COMUNICADO CON ALEXA

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Rafael Luis Saneleuterio Temporal

TUTORIZADO POR

Patricia Balbastre Betoret

CURSO ACADÉMICO: 2021/2022

Resumen

El objetivo de este proyecto es diseñar, programar y construir el prototipo Aurora, un dispositivo capaz de interactuar con una cortina de forma automática. Para ello, se busca establecer una conexión con la asistente virtual Amazon Alexa, así como conseguir que, a través de ella, cualquier usuario pueda controlar la luz natural que entra en la estancia y consultar la temperatura de la sala. Esto es posible gracias a la programación de un módulo Wi-Fi de Espressif en el entorno de Arduino y el llamado "internet de las cosas" o IoT (internet of things). El diseño del automatismo se basa en un mecanismo que se agarra, tras la primera anilla, a una barra común de cortina y que puede deslizarse por ella, desplazando así el telón. Además, este instrumento domótico también dispone de un sensor de luz para la posible detección del amanecer y uno de temperatura para su telemetría. En conclusión, mediante Aurora, se consigue crear un sistema multifuncional de rápida y flexible instalación en una cortina, compatible con la asistente de voz Alexa.

Palabras clave: Arduino, domótica, IoT, Alexa, Espressif.

Resum

L´objectiu d´aquest projecte és dissenyar, programar i construir el prototip Aurora, un dispositiu capaç d´interactuar amb una cortina de forma automàtica. Per a això, es busca establir una connexió amb l'assistent virtual Amazon Alexa, així com aconseguir que, a través d'ella, qualsevol usuari puga controlar la llum natural que entra a la sala i consultar la temperatura de l'estància. Això és possible gràcies a la programació d'un mòdul Wi-Fi d'Espressif a l'entorn d'Arduino i l'anomenat "internet de les coses" o IoT (internet of things). El disseny de l'automatisme es basa en un mecanisme que s'agafa, després de la primera anella, a una barra comuna de cortina i que es pot moure, desplaçant així el teló. A més, aquest instrument domòtic també disposa d'un sensor de llum per a la possible detecció de la matinada i un de temperatura per a la telemetria. En conclusió, mitjançant Aurora, s'aconsegueix crear un sistema multifuncional de instal·lació ràpida i flexible en una cortina, compatible amb l'assistent de veu Alexa.

Paraules clau: Arduino, domòtica, IoT, Alexa, Espressif.

Abstract

The objective of this project is to design, program and build the Aurora prototype, a device capable of automatically interacting with a curtain. To do this, it seeks to establish a connection with the Amazon Alexa virtual assistant, as well as to ensure that, through her, any user can control the natural light that enters the room and check its temperature. This is possible thanks to the programming of an Espressif Wi-Fi module in the Arduino environment and the so-called "internet of things" or IoT. The design of the automation is based on a mechanism which is attached, behind the first ring, to a common curtain rod and which can slide along it, moving the curtain by the way. In addition, this home automation instrument also has a light sensor for possible sunrise detection and a temperature sensor for telemetry. In conclusion, through Aurora, it is possible to create a multifunctional system that is quick and flexible to install in a curtain, compatible with the Alexa voice assistant.

Keywords: Arduino, home automation, IoT, Alexa, Espressif.

Índice general

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

AURORA: DISEÑO Y PROGRAMACIÓN DE UN AUTOMATISMO PARA CORTINAS COMUNICADO CON ALEXA

1. MEMORIA

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Rafael Luis Saneleuterio Temporal

TUTORIZADO POR

Patricia Balbastre Betoret

CURSO ACADÉMICO: 2021/2022

Índice memoria

Índice de figuras

Índice de tablas

1. Introducción

En este proyecto final de grado se lleva a cabo el diseño y programación de un dispositivo domótico llamado Aurora que dispone de varios sensores y actuadores sobre una cortina que, a su vez, se comunica con Alexa. El funcionamiento nominal de Aurora es su movimiento a lo largo de una barra de cortina situándose entre dos anillas (véase figura 1), permitiendo desplazar el telón hacia un sentido u otro.

Figura 1: Apariencia de Aurora en una cortina

Los procesos automáticos han ayudado al ser humano desde antes del uso de la electrónica; un ejemplo de ello son los molinos de viento de La Mancha. Este mecanismo del siglo XVI puede moler grandes cantidades de grano, una labor que, a mano, era lenta y escasa. Gracias a este sistema de engranajes se consiguió aumentar la producción, llegando a alimentar a más personas y situando a la región en un comercio a nivel internacional [1].

En las últimas décadas, el uso de sistemas automáticos se ha expandido a prácticamente cualquier ámbito. Desde el año 1990 se llevan construyendo fábricas totalmente automáticas [2]. Hoy en día ya se tienen este tipo de sistemas implementados en ciudades (urbótica), huertos, hospitales, edificios (inmótica) y hogares (domótica), facilitando el día a día de los usuarios. En España, aproximadamente un 48% de las viviendas tienen algún dispositivo domótico [3]. Desde aspiradoras inteligentes como la Conga [4], de la empresa valenciana Cecotec, hasta iluminaciones automáticas se encuentran ya en muchas viviendas u otro tipo de inmuebles. Pero también está en auge el despliegue de otros instrumentos inteligentes variados como sistemas de riego automáticos o medidores de consumo eléctrico.

Es ahora cuando los asistentes virtuales entran en juego, siendo los más conocidos Google Home, Siri de Apple y Amazon Alexa. Estos servicios pueden utilizarse directamente desde plataformas móviles como smartphones o adquiriendo un altavoz inteligente de la propia marca. En 2019 Amazon ya había sobrepasado la cifra de los 100 millones de ventas de productos de Alexa [5]. Aparte de la asistencia por voz que ofrecen este tipo de dispositivos,

también brindan la posibilidad de gestionar de forma inteligente muchos aparatos domóticos dotados de una conexión a internet y al llamado "internet de las cosas" o IoT (internet of things). De esta forma, se llega a unificar en una sola herramienta el control de toda una casa.

En este punto queda contextualizar el concepto de IoT, el proceso de conectar elementos físicos a internet. Estos objetos van desde los presentes en el ámbito doméstico hasta dispositivos médicos o de mantenimiento, por mencionar algunos ejemplos [6]. Sin embargo, no basta con establecer una conexión a la red, los elementos que pertenecen al IoT deben cumplir una o varias de las siguientes funciones: monitoreo, control, optimización y automatización [7]. Estas cuatro capacidades corresponden respectivamente a la recolección de datos a través de sensores, la ejecución de acciones, la mejora de prestaciones y la independencia rutinaria.

Por consiguiente, la realización del proyecto queda justificada por la creciente necesidad de automatizar actividades de la vida cotidiana, fruto del deseo de dotar de inteligencia a elementos domésticos. Alcanzar este objetivo mediante dispositivos de fácil instalación y evitando sustituir los elementos originales ha sido el principal impulsor de este proyecto. Se trata, por tanto, de seguir la idea de desarrollar nuevas tecnologías que se adapten a lo que ya se tiene establecido, con el objetivo de mejorar sus capacidades.

Antes de proseguir con la propia memoria, cabe comentar brevemente su estructura. Para tratar de alcanzar una redacción completa de todos los aspectos del prototipo Aurora, el documento se divide en los apartados de objetivos, antecedentes, estudio de necesidades, soluciones alternativas y justificación de la adoptada, la descripción y justificación detallada de la solución. Finalmente , se encuentran unas conclusiones y bibliografía. Dado que se va a llevar a cabo un montaje electrónico, una programación y un diseño físico del prototipo Aurora, el mencionado apartado de descripción y justificación detallada de la solución adoptada está dividido en tres grandes subsistemas: el electrónico, el de programación y el mecánico. Para acabar, conviene señalar que, como es habitual en proyectos de ingeniería, aparte de la memoria, se incluyen los documentos de planos, pliego de condiciones y presupuesto.

2. Objeto

El objetivo general (OG) del presente trabajo es desarrollar un prototipo domótico que actúe sobre una cortina y se controle a través de Alexa.

Como objetivos específicos (OE) del proyecto Aurora, se pretende:

- OE1. Montar un circuito electrónico capaz de conectarse a internet de forma independiente para trabajar en el entorno IoT.
- OE2. Capacitar al dispositivo Aurora de un comportamiento sensible a la luz exterior al amanecer.
- OE3. Dotar a Aurora de un sensor de temperatura para la posible telemetría de la estancia.
- OE4. Desarrollar la programación del prototipo en el entorno del software libre de Arduino.
- OE5. Diseñar una estructura mecánica que permita a Aurora abrir y cerrar diferentes tipos de cortinas instalándose de forma flexible, rápida e intuitiva para cualquier usuario.

3. Antecedentes

Durante los últimos años, se han lanzado al mercado algunos automatismos para cortina con amplias funcionalidades. Uno de los dispositivos más conseguidos es la cortina de SwitchBot, que funciona sobre barras cilíndricas de hasta 4 cm de diámetro y también puede adaptarse a una barra de cortina con raíles en forma de “U” [8]. Dispone de sensor de luz y es capaz de detectar si se le aplica fuerza manualmente a la cortina para ponerse en funcionamiento. A su vez, permite ser controlada por una aplicación y, mediante el uso adicional de un *hub* de la misma marca, permite comunicarse con asistentes inteligentes como Alexa o Google Home. Recientemente, la misma empresa ha lanzado una segunda versión del dispositivo llamada SwitchBot Curtain Rod 2, la cual incluye mejoras como un agarre más flexible, un movimiento más fluido y la posibilidad de instalar una placa fotovoltaica [9]. Otra alternativa es la cortina de Tuya, la cual tiene unas características similares a la anteriormente mencionada, si bien esta dispone de sensor adicional de temperatura [10].

El precio de estos productos para el hogar con tecnología inteligente varía dependiendo de la vía de compra, pero desde las páginas oficiales los números rondan los 100 € (sin incluir el *hub*). En las tiendas *online* más asequibles, como Amazon, el precio de las cortinas de SwitchBot se sitúa cerca de los 80 € y la cortina de Tuya se vende por alrededor de 60 €.

Los inconvenientes de los productos descritos son de diferente naturaleza: por un lado, económicamente su precio es desproporcionado teniendo en cuenta sus posibles aplicaciones. Por otro lado, se trata de dispositivos dependientes para conectarse a internet de un puerto de enlace físico o *hub*, que se vende aparte y que supone un consumo de tiempo, espacio y energía.

En el proceso de diseño de Aurora se han pretendido minimizar los contras y potenciar los pros de los antecedentes más evolucionados del mercado. Aun así, el objetivo de este proyecto no es crear una versión comercial ni competir con los productos anteriormente nombrados; se trata de diseñar un prototipo a través de materiales económicos y herramientas de libre uso que pueda conectarse de forma independiente a internet y, por consiguiente, a una nube IoT. De esta manera, se crea la primera versión de Aurora, cuyo coste de producción puede ser asequible para un primer desarrollo, siendo un dispositivo directamente controlado mediante Alexa y sin necesidad de aparatos adicionales como puertos de enlace (*hubs*). Evidentemente, este trabajo puede llegar a ser un punto de partida para futuras aplicaciones en el mundo de la domótica y también un provechoso arranque para la creación de una posible versión comercial.

4. Estudio de necesidades, limitaciones y condicionantes

Aurora permite el control de una cortina a distancia y esto puede ser útil tanto para personas con diversidad funcional como para usuarios que busquen complementar una casa inteligente. El hecho de poder regular cuándo entra el sol en las estancias y cuándo no la convierte en una herramienta que puede ayudar a despertarse de forma agradable, impedir que los muebles se dañen con la exposición continua al sol o ahorrar energía en climatización con la buena gestión de la luz natural.

Para ello, el diseño del dispositivo Aurora busca poder adaptarse a un amplio tipo de barras de cortina. No obstante, se consideran una serie de restricciones y circunstancias. Cabe comentar que no hay ninguna normativa que restrinja el desarrollo y uso de proyectos de esta índole; por ello, las diferentes limitaciones nacen de la naturaleza del propio prototipo.

Los requisitos físicos para instalar Aurora son los siguientes: su mecanismo de agarre tolera diámetros entre 5 y 45 mm; el desplazamiento puede efectuarse a lo largo de barras cilíndricas u otros prismas, sin límite de largaria (el recorrido máximo son 10 m para una sola orden, extensión infrecuente en cortinajes domésticos; igualmente, se puede repetir la orden las veces que hiciera falta). El peso máximo de la tela que puede arrastrar el proyecto es de 3 kg, aunque ello depende, en gran medida, del rozamiento de las anillas con la barra. Por esto último, se ha de garantizar que la cortina sobre la que se actúa se corra sin forcejeos.

Para el funcionamiento nominal de Aurora, la cortina en cuestión debe estar ligada a la barra mediante anillas y estas han de tener una separación igual o mayor a 15 cm entre ellas. Sobre la barra debe haber un espacio mínimo de 5 cm y una distancia a la pared o persiana mayor a 5 cm. Si se quiere dar uso al sensor de luz, la barra es necesario que la barra se sitúe a menos de 15 cm por encima de la entrada de luz. En caso contrario, aunque es cierto que el sensor de luz puede orientarse hacia abajo, su efectividad se verá reducida.

Las posibilidades de Aurora permiten trabajar también con cortinas sin anillas. En ese caso, si el agarre a la barra es a través de la propia tela o si directamente el prototipo no cabe en su posición nominal, cabe la posibilidad de instalarlo fuera del telón. Para ello, se ha de posicionar junto a este y enganchar la tela a alguno de los agarres de los que dispone el dispositivo. De todas formas, aunque puede haber más métodos para la implementación del automatismo, este proyecto está enfocado a su uso nominal.

5. Planteamiento de soluciones alternativas y justificación de la solución adoptada

Con el motivo de analizar las soluciones posibles para la aplicación del proyecto planteado, se parte de los aspectos generales de los diferentes subsistemas de Aurora y, en caso de ser conveniente, se plantean alternativas más específicas.

5.1. Alternativas en el entorno electrónico

En cuanto al primer subsistema, el montaje electrónico puede llevarse a cabo mediante un sistema empotrado, donde todos los componentes están soldados y se reduce al máximo el espacio ocupado. Una alternativa a esto es trabajar sobre una placa de prototipos, *protoboard* o *breadboard*. Esta última opción es idónea para el desarrollo del proyecto puesto que, siendo asequible, permite prototipar con más facilidad, a pesar de ser una opción menos económica que la anterior.

A la hora de elegir una placa de desarrollo para volcar el programa y conectar todos los componentes, las alternativas más extendidas son los módulos de Espressif cuando se habla de establecer una conexión a internet. Esta marca tiene disponible una gran variedad de controladores como el ESP-01, el ESP-12E o el NodeMCU. Todos estos modelos

contienen el sistema en chip Wi-Fi ESP8266. Las diferencias entre cada una de las placas de desarrollo, a grandes rasgos, son el número de pines a los que se tiene acceso. Para este proyecto se va a dar uso del integrado NodeMCU, ya que tiene un número suficiente de pines y, además, viene provisto de un puerto micro USB para una fácil habilitación del programa.

Para controlar el motor de Aurora, es conveniente poder trabajar con una tensión superior a la usada en la lógica de la placa de desarrollo de 3,3 V. Una solución simple es la implementación de un transistor BJT o uno de tipo MOSFET, lo cual permite el control de un motor DC, pero solo accionando el giro en un sentido. Es por ello por lo que se busca una solución alternativa que aporte la función de poder girar el motor en sentido horario y antihorario para que Aurora abra y cierre la cortina. La respuesta a esta problemática es el uso de la conocida configuración de transistores conocida como puente H, la cual permite el control completo del motor. El módulo L298N es un controlador de motores que incorpora esta tecnología, por tanto, se procede a la selección de este *driver*.

El siguiente componente electrónico sobre el que se plantean soluciones alternativas de peso es el sensor de temperatura. Para la selección de este, se busca una implementación sencilla, ya que no se busca conseguir una medición muy precisa de esta magnitud. Por un lado se tiene el sensor LM35, el cual es el más extendido en montajes de Arduino. Posee una precisión de un cuarto de grado Celsius y es capaz de medir temperaturas en el rango de −55 °C a 150 °C. Por otro lado se encuentra el DHT11, sensor que, utilizando el mismo número de pines que el anterior, permite medir además la humedad del ambiente. Sin embargo, los mencionados sensores no están fabricados para funcionar con una tensión de 3,3 V; por ello, se selecciona para el proyecto Aurora el TMP36, cuyo funcionamiento es análogo al LM35 y permite trabajar con voltajes menores.

Dado que la placa de desarrollo NodeMCU solo tiene una entrada analógica y aparte de la medición de temperatura también se van a sensar diferentes niveles de luz, hay que implementar un sistema compatible con esta problemática. Una alternativa es la de conectar ambos sensores a la misma entrada analógica y sus respectivos pines de alimentación a dos salidas digitales. De esta forma, a través de la programación, se puede ir turnando el uso del ADC. Para conseguir lo mencionado, es necesario utilizar dos diodos para que en la conexión común de los dos sensores no se falsee la medición. Por contraparte, se puede evitar este uso compartido de la entrada analógica y adquirir los datos de medición de luz a través de varias entradas digitales. Esto se consigue utilizando un sensor lumínico resistivo o LDR en conjunto con un divisor de voltaje múltiple. En esencia, se consiguen menos valores de medición que en la implementación anterior pero suficientes para detectar varios niveles de amaneceres. Teniendo en cuenta que el coste de los diodos, pese a ser componentes muy económicos, es bastante superior al de las resistencias, se procede a la elección de la alternativa del divisor resistivo.

Para acabar con las alternativas del subsistema electrónico, queda la cuestión de cómo se va a alimentar el circuito. En una primera instancia, se tiene el uso de una batería, la cual permite a Aurora ser independiente de cables haciéndola mucho más manejable. Si bien es cierto que el dispositivo consume poca electricidad, una batería se descarga por completo a los pocos días para esta aplicación. La placa de desarrollo tiene una opción para ponerla a “dormir” o en *stand by,* a través de los estados de *sleep* o *deep sleep;* de esta forma, se lleva a la NodeMCU a un consumo mínimo. No obstante, estas funciones desactivan la conexión a internet y, por tanto, queda descartado su uso. Si se implementara una batería, se debería cargar cada pocos días o bien dar uso de una placa solar. Esta última opción, el panel fotovoltaico, dependiendo de la orientación de la casa, estaría prácticamente todo el día en contacto con la luz solar debido a la posición estratégica de Aurora por el exterior de

la cortina; gracias a ello, se mantendría la batería cargada en todo momento. Una forma de materializar la última propuesta sería implementando un módulo cargador de baterías como el TP4056 y un convertidor elevador o *voltage booster* para regular la tensión de entrada al circuito. Sin embargo, debido a las limitaciones de tiempo, esta parte de diseño se va a reducir al uso de una batería de pequeño tamaño, con su debida implementación, para alimentar al prototipo.

5.2. Alternativas en el entorno de programación

Para justificar la elección de Alexa como la interfaz de control del dispositivo domótico Aurora frente a otros asistentes como Siri o Google Home, pueden mencionarse dos argumentos a favor. El primero es que, por el momento, es la asistente con mejores valoraciones y prestaciones en aspectos muy amplios de su funcionamiento, además de ser compatible con una gran parte de dispositivos del IoT. El segundo argumento es la alta capacidad de personalización de diferentes habilidades o *skills,* al igual que la posibilidad de crear diferentes rutinas de funcionamiento. Todo ello brinda al usuario la libertad de controlar los dispositivos con Alexa de formas muy variadas, frente a sus competidores.

A la hora de programar la implementación de Aurora con Alexa, Amazon cuenta con una herramienta llamada Alexa Skills Kit. Esta aplicación está orientada a desarrolladores de producto y da uso de un lenguaje propio de programación. No obstante, Arduino dispone de un servicio en línea para trabajar con su nube de IoT, en la cual puede introducirse cualquier usuario de forma libre. De todas formas, cabe comentar que esta herramienta es gratuita para los dos primeros objetos que se quiera conectar a la nube. Si se desea ampliar el plan, el precio máximo se sitúa, a fecha de la realización de la presente memoria, en 19,99 € al mes para el control de 100 dispositivos conectados al IoT de la nube de Arduino.

Para programar el módulo NodeMCU existen dos lenguajes muy extendidos: Lua y Arduino. La primera alternativa no es tan conocida como la segunda, pero tiene algunas ventajas como su rápida ejecución o su estructura compacta. Ambos lenguajes son de uso libre, pero para el código del proyecto se va dar uso del entorno de desarrollo Arduino IDE, ya que es de los más utilizados a lo largo del grado de Ingeniería Electrónica Industrial y Automática y, además, es completamente compatible con el servicio de Arduino IoT Cloud.

5.3. Alternativas en el entorno mecánico

Antes de proceder a la explicación detallada del prototipo Aurora, se van a exponer algunas alternativas generales en el desarrollo de la mecánica.

Aurora es un actuador, cuya función móvil es la de abrir y cerrar una cortina. La primera opción que se baraja es la de fijar el dispositivo en un lado de la zona de acción y, a través de poleas y cuerdas, poder controlar la cortina. Esta opción tiene la ventaja de ser, por lo general, compatible con todo tipo de cortinajes, ya que podría tirar de cualquier tela. Además, si se trata de una cortina motorizada, Aurora podría actuar sobre el propio sistema de motores ya implementado. La desventaja más evidente es la dificultad de la instalación del sistema de cuerdas y, si se trata de una cortina motorizada, es complicado diseñar un modelo flexible que se adapte a más de un tipo de sistema preestablecido de motores.

Por ello, surge la alternativa de dotar al prototipo de movilidad, que Aurora se desplace junto con la cortina. La intención es que el prototipo se agarre a la barra y, con ayuda de una rueda, consiga moverse. El propio dispositivo puede empujar las anillas y también disponer de unos agarres para enganchar la tela de la cortina. Esta solución es la elegida para la

implementación de Aurora porque, a pesar de depender mucho de la forma de la barra de la cortina, tiene un diseño compacto que facilita la instalación, y que permite que cualquier usuario pueda colocar a Aurora de forma intuitiva.

Para dar forma a la estructura, existen múltiples opciones. Lo idóneo para un prototipo es imprimir las piezas en 3D, ya que por moldes sería necesario una gran inversión inicial, algo desmesurado para un modelo no comercial. Para la impresión 3D, las alternativas más usadas son el uso de impresoras SLA y FDM. El primer método se basa en el uso de la estereolitografía, que utiliza tecnología láser para modelar las piezas en un depósito de resina. Esta opción consigue unos acabados muy precisos y es perfecta para figuras con pequeños detalles. La gran desventaja es su coste, que puede llegar a ser cuatro veces menos rentable que los otros tipos de impresoras. En cuanto al FDM, es la forma más típica de imprimir piezas en tres dimensiones con un error de alrededor de 0,25 mm, pero con precios generalmente más asequibles. Su funcionamiento se basa en ir apilando un pequeño hilo de material fundido mediante un robot cartesiano hasta formar la pieza. Para la fabricación de las diferentes piezas mecánicas de Aurora se decide elegir la solución de la impresión FDM, ya que tiene un mejor precio y no se busca un alto nivel de detalle en las figuras.

Estas últimas decisiones conducen a que el material de fabricación utilizado deberá ser compatible con las impresoras de hilo o FDM. Los más extendidos en este campo son los plásticos PLA, ABS, PETG, TPE, PP o fibra de carbono, por mencionar algunos. Las propiedades de cada material son muy variadas; para Aurora se elige el uso del PLA o poliácido láctico, ya que presenta unas características ideales para la construcción de un prototipo inicial, como la fácil y veloz impresión, además de ser muy estable, ecológico y resistente a la radiación ultravioleta. A pesar de ser el material más extendido en la impresión 3D, tiene algunas desventajas, como la sensibilidad a la humedad, al calor (a partir de 60 ºC) y resistencia mecánica media o baja dependiendo de su forma de impresión. En síntesis, para el ámbito de uso del dispositivo Aurora, el material PLA impreso a máxima densidad es el que más facilitará el proceso de producción cumpliendo también con sus funciones mecánicas.

Por último, cabe comentar la selección del color. Como Aurora es un dispositivo que puede estar expuesto al sol durante varias horas, interesa utilizar un color de carcasa que refleje la máxima radiación posible para no calentar excesivamente el prototipo. De esta manera, se elige el color blanco y se minimiza el efecto térmico del sol sobre los componentes mecánicos y electrónicos, así como la medición de la temperatura. Cabe comentar que el blanco también es un color bastante neutro, lo que se traduce en una mayor afinidad con otros colores y, en la mayoría de los casos, una mínima ruptura de la estética de la estancia.

6. Descripción y justificación detallada de la solución adoptada

En este apartado se explica de forma precisa cada uno de los componentes y apartados resultantes de las alternativas de los tres diferentes subsistemas. En la figura 2 se plantea el organigrama general en el que se estructura este apartado descriptivo de Aurora.

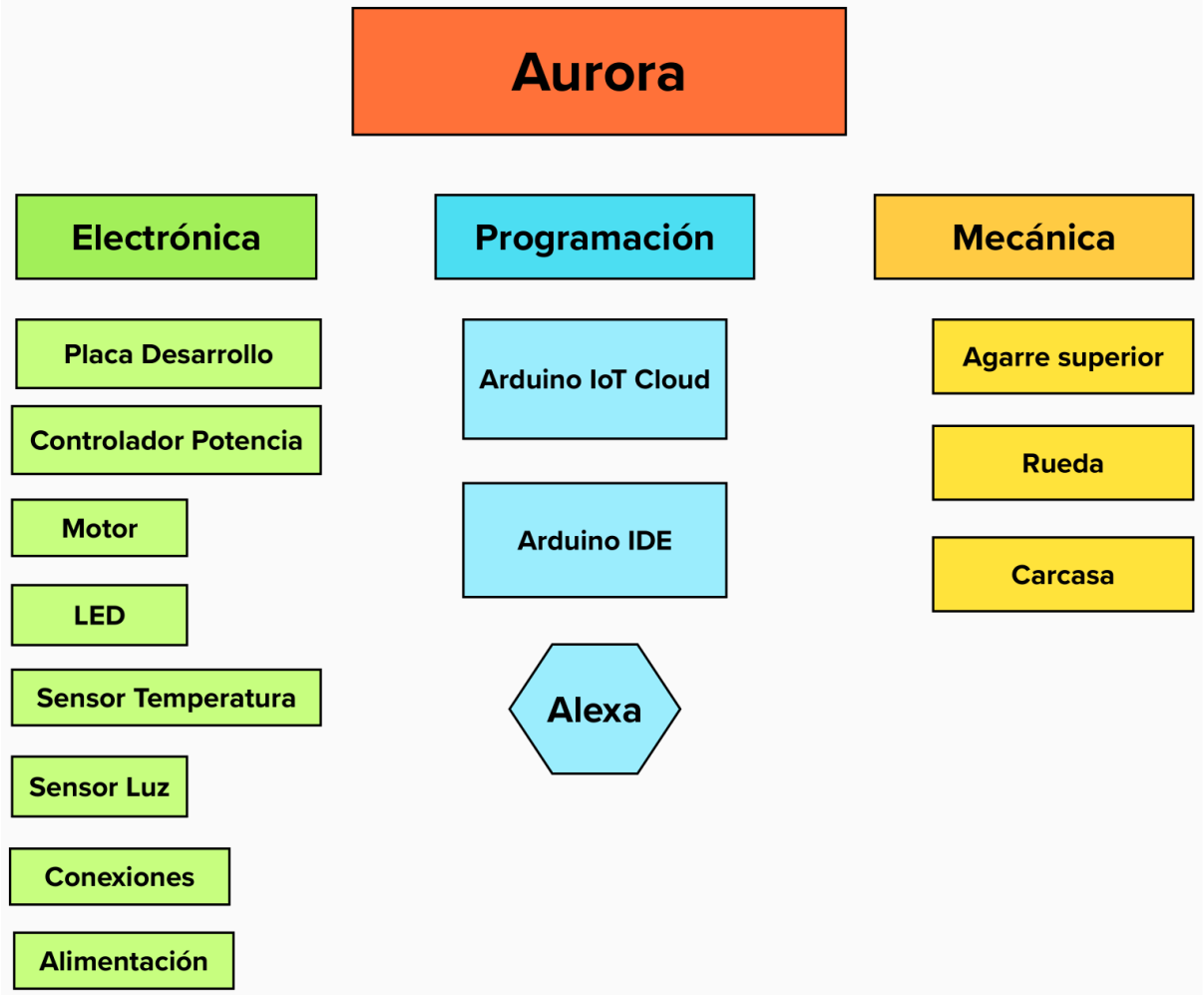


Figura 2: Organigrama general del proyecto Aurora

6.1. Subsistema electrónico

El subsistema de la electrónica del proyecto se conforma por una serie de elementos de fácil adquisición y bajo coste cuyos usos se describen y justifican de forma detallada en este apartado.

6.1.1. Placa de desarrollo

El elemento electrónico más importante del proyecto es el kit de desarrollo NodeMCU [11]. Esta placa proporciona numerosos pines con los que va a ser más ágil el prototipado para trabajar con el módulo ESP-12E y, en definitiva, para diseñar dispositivos conectados a internet. Todo ello con un bajo consumo, ya que su tensión de trabajo es 3,3 V (se le puede alimentar con 3,3 V o 5 V, pero sus componentes funcionan con el primer valor).

Se trata de una placa de *hardware* abierto, por lo que cualquier fabricante puede distribuir libremente su propia creación. En este caso se ha elegido la versión 1.0 correspondiente a la segunda generación de este tipo de placas. Más concretamente, la proporcionada por el fabricante Lolin / Wemos con el nombre común V3.

Este dispositivo (ver figura 3) lleva integrado el módulo Wi-Fi ESP-12E, y este contiene el sistema en chip o SoC (System on chip) ESP8266 de Espressif Systems. Este conjunto utiliza como microcontrolador el Tensilica L106 de 32 bits que será el encargado de ejecutar

cada una de las sentencias del código. Por tanto, este dispositivo alberga el programa, se conecta a la red Wi-Fi y gestiona todos los sensores y actuadores del proyecto.

Cabe comentar que su consumo de datos para la aplicación de Aurora es de unos 6,8 MB por hora (valor medido experimentalmente).

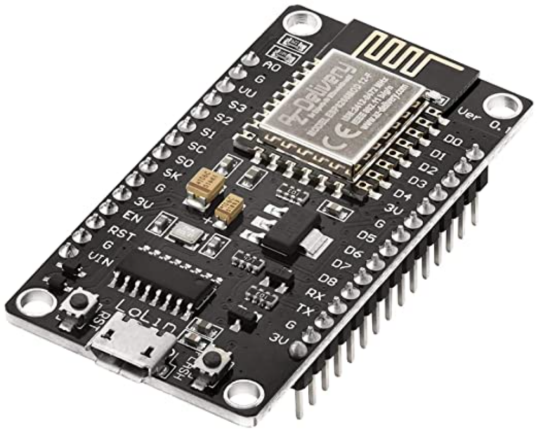


Figura 3: Placa de desarrollo NodeMCU V3
Fuente: Az-Delivery [11]

6.1.2. Controlador de potencia

Para manejar el motor del que dispone el automatismo se va a utilizar el *driver* o controlador L298N (figura 4). En esencia, este módulo [12] permitirá trabajar con una tensión mayor que la tensión de funcionamiento de la placa. En concreto, este componente está destinado al control de motores de tensiones entre 5 y 35 voltios gracias a sus dos puentes H. Este tipo de puente de transistores permite girar el motor en ambos sentidos.

En este proyecto, el controlador se alimenta con 7 V y, de esta manera, el motor dispone de entre 5 y 6 V para funcionar. Además, este módulo incluye una salida de 5 V que se utiliza para alimentar a la placa de desarrollo.

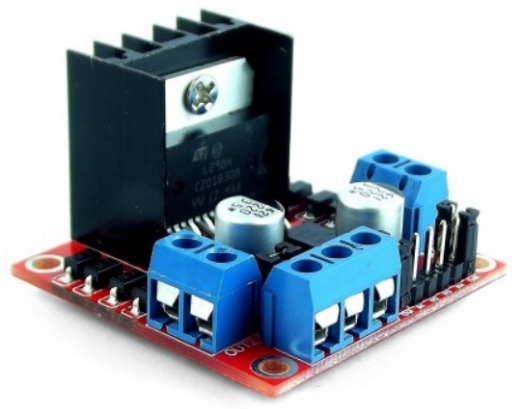


Figura 4: Controlador L298N
Fuente: Amazon [12]

6.1.3. Motor

Se va utilizar un motor de corriente continua (DC) de 3 a 6 V (véase figura 5). Concretamente se dará uso a la versión con engranajes reductores [13] (carcasa amarilla) para ofrecer fuerza a cambio de velocidad de giro. Para un mejor ahorro de espacio se escoge la versión de un solo eje.

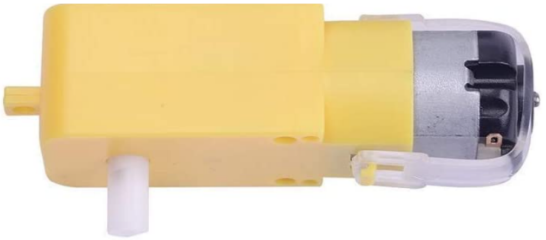


Figura 5: Motor DC
Fuente: Amazon [13]

6.1.4. LED

Se utiliza un LED rojo junto con una resistencia de 220 Ω (R1) para indicar que el sistema domótico está moviendo el motor para correr la cortina.

6.1.5. Sensor de temperatura

Para medir satisfactoriamente la temperatura de la estancia se utiliza el TMP36 [14] (véase figura 6). Sus características más relevantes son:

- Factor de escala: 10 mV/Cº el cual se tendrá en cuenta en la programación.
- Baja tensión de alimentación: 2,7 V a 5,5 V. Rango adecuado para la tensión de trabajo de la placa de desarrollo (3,3 V).
- Bajo autocalentamiento.
- Rango de −40 °C a +125 °C
- Error máximo de 2 ºC.

Para reducir la influencia térmica de los otros componentes electrónicos, el sensor se sitúa en la parte inferior de todo el montaje.

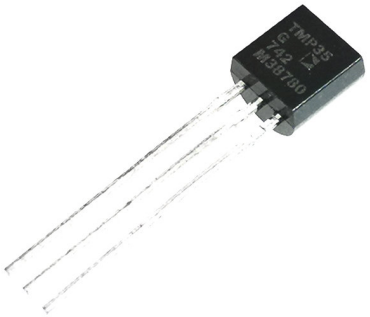


Figura 6: Sensor temperatura TMP36
Fuente: Amazon [14]

6.1.6. Sensor de luz

El sensor de luz estará conformado por un divisor resistivo conectado a una fotorresistencia o LDR (*light dependent resistor*) [15]. Concretamente se trata del modelo GL5516 (véase figura 7). Esta LDR tiene una gran resistencia eléctrica en la oscuridad (del orden de megaohmios) y, conforme incide más luz sobre ella, su valor disminuirá drásticamente (hasta un mínimo de unos 30 Ω, valor estimado experimentalmente) y lo hará de forma no lineal.

Figura 7: LDR GL5516
Fuente: Amazon [15]

El divisor resistivo tendrá por un lado la resistencia base R2 y por otro lado tres pequeñas resistencias R3, R4 y R5 de mismo valor junto con el LDR. El objetivo de estas últimas resistencias es ofrecer tres medidas distintas que se pondrán a nivel alto (a partir de los 1,8 V aproximadamente) a niveles de luz diferentes.

Para obtener los valores adecuados de las resistencias se parte de un circuito simplificado, como muestra la figura 8, donde Rx es el conjunto de resistencias divisoras, es decir:

$$Rx = R3 + R4 + R5 \qquad (1)$$

Las salidas del divisor V1 y V3 hacen referencia a los pines que se pondrán a nivel alto en los estados extremos, el primero con poca intensidad lumínica y el segundo con la máxima. La tercera salida restante, V2, será la que se pondrá a nivel alto con un nivel de luz intermedio, no obstante, esta se situará en la siguiente fase del diseño.

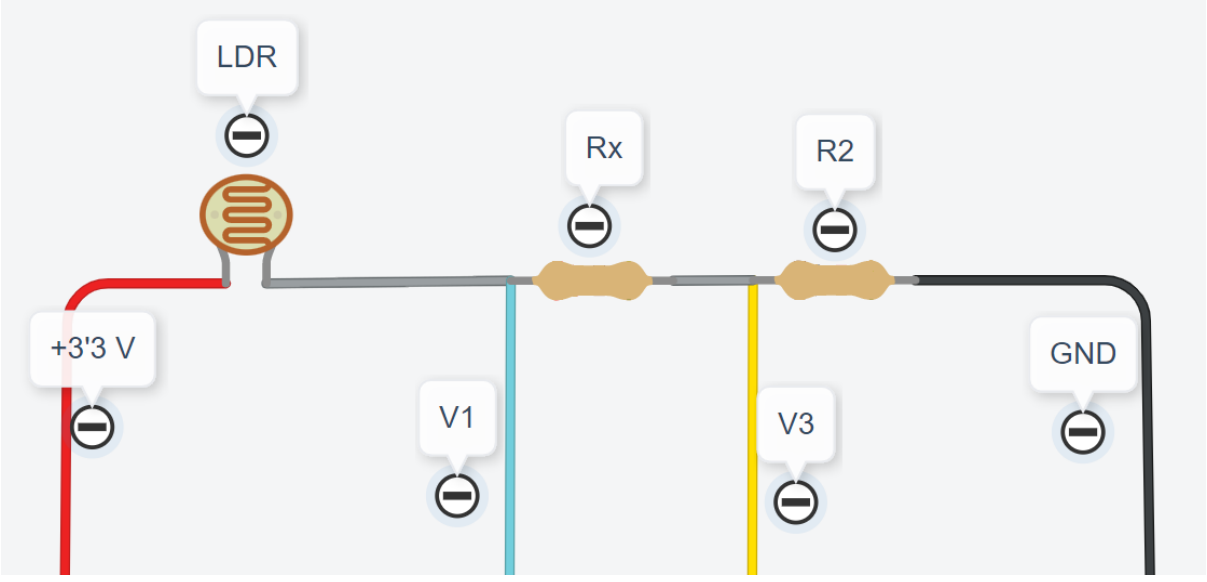


Figura 8: Divisor resistivo simplificado

A continuación, se plantean las diferentes ecuaciones 2 y 3 que definen las salidas del circuito:

$$V1 = 3,3 \frac{Rx+R2}{Rx+R2+LDR} \quad (2)$$

$$V3 = 3,3 \frac{R2}{Rx+R2+LDR} \quad (3)$$

De los valores expuestos se conocen los siguientes:

- La tensión de alimentación de 3,3 V que ofrece el módulo NodeMCU.
- Las salidas V1 y V3 cuyo valor ha de ser como mínimo 1,8 V para que la placa las detecte como entradas a nivel alto (IH). De todos modos, para no trabajar justo en los límites, se considera un valor de tensión de 1,9 V en los cálculos.
- Los valores que puede alcanzar el LDR son conocidos de forma experimental. Al igual que en el caso anterior, para no trabajar en los límites el valor a pleno sol se considera 50 Ω (LDR mínimo) y una cifra adecuada para un nivel de luz tenue es 1,3 kΩ (LDR máximo), valor medido experimentalmente.

Se construye, por tanto, un sistema de dos ecuaciones (4 y 5) y dos incógnitas (Rx y R2).

$$V(IH) = 3,3 \frac{Rx+R2}{Rx+R2+LDR(máx.)} \quad (4)$$

$$V(IH) = 3,3 \frac{R2}{Rx+R2+LDR(mín.)} \quad (5)$$

$$1,9 = 3,3 \frac{Rx+R2}{Rx+R2+1300} \quad (6)$$

$$1,9 = 3,3 \frac{R2}{Rx+R2+50} \quad (7)$$

Sustituyendo los parámetros conocidos se obtienen las ecuaciones 6 y 7. Se despejan las resistencias incógnitas y se resuelve el sistema para obtener los valores de R2 y Rx:

- R2 toma el valor 1044,56 Ω, cuya normalización en la serie E12 es 1000 Ω.
- Al despejar Rx se obtiene el valor 719,69 Ω y, como se ha visto en la fórmula 1, esta resistencia se descompone en varias para conseguir una medición intermedia de la luz.

Tras una serie de pruebas experimentales se ha concluido que esta medición, V2, ha de tener un valor resistivo menor entre V3 que entre V1 para que las tres detecciones de luz se diferencien correctamente. Esto es debido a que el comportamiento del LDR no es lineal frente a los cambios de luz. En este punto queda dividir Rx en otras resistencias diferentes que, para simplificar, serán tres resistencias de igual valor R3, R4 y R5. Por tanto, se obtiene la igualdad 8:

$$R3 = R4 = R5 = \frac{Rx}{3} \quad (8)$$

De la división de Rx se obtiene el valor de 239,9 Ω cuyo valor más cercano en la serie normalizada E12 es 220 Ω.

Antes de continuar, como se han normalizado las resistencias, los valores de las salidas del divisor resistivo ya no serán los 1,9 V establecidos en los valores límite. Por ello, conviene comprobar que tanto V1 como V3 siguen estando por encima de los 1,8 V mínimos para considerarse un nivel alto en las condiciones preestablecidas. En las igualdades 9 y 10 se verifica lo mencionado.

$$V1 = 3,3\frac{Rx+R2}{Rx+R2+1300} = 3,3\frac{220\cdot3+1000}{220\cdot3+1000+1300} = 1,85\ V \quad (9)$$

$$V2 = 3,3\frac{R2}{Rx+R2+50} = 3,3\frac{1000}{220\cdot3+1000+50} = 1,93\ V \quad (10)$$

Por tanto, y a modo resumen, el diseño del divisor resistivo final (figura 9) requiere los siguientes componentes:

- R2 con valor de 1 kΩ.
- R3, R4 y R5 con valor de 220 Ω.
- LDR modelo GL5516.

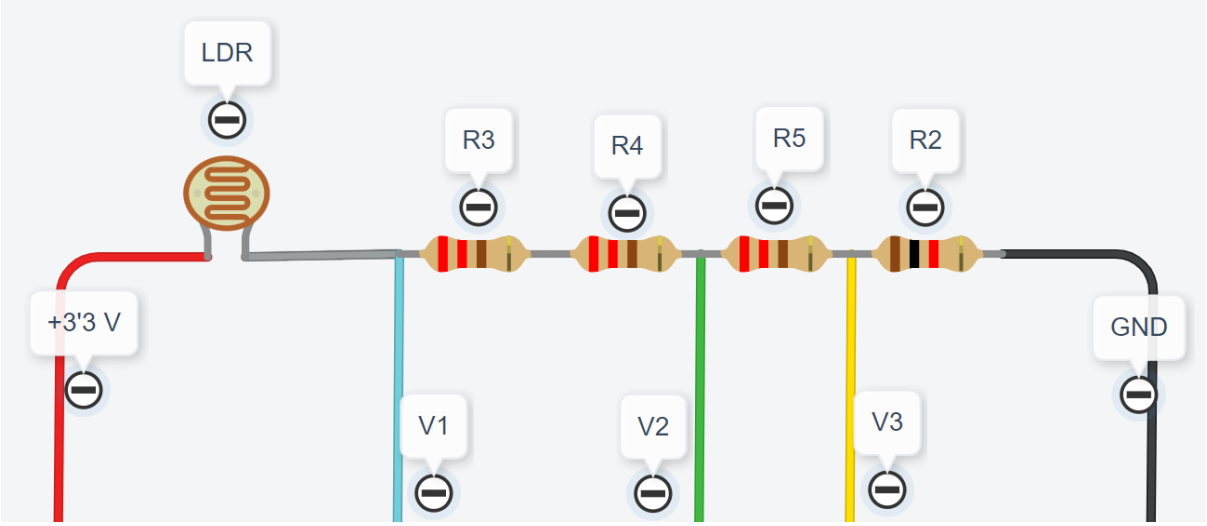


Figura 9: Divisor resistivo final

6.1.7. Conexiones

6.1.7.1. Placa de prototipos

Para conectar la placa a los demás componentes se utilizan dos placas de prototipos o *protoboards* de 170 puntos [16] como los de la figura 10. Estas pequeñas bases se pueden unir entre sí para formar una estructura más estable, donde el NodeMCU conecta cada fila de pines a un *breadboard* distinto.

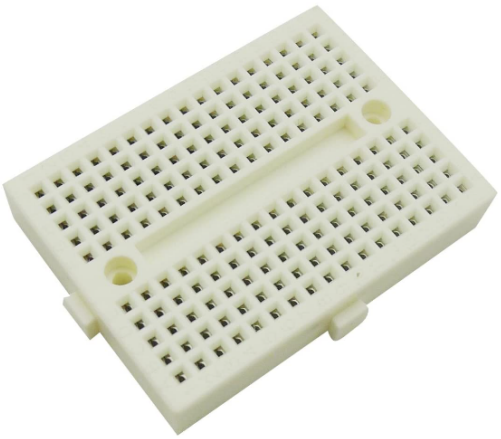


Figura 10: Protoboard 170 puntos
Fuente: Amazon [16]

6.1.7.2. Cables de interconexión

Cabe mencionar que se utilizan cables de macho a macho y de macho a hembra en el montaje electrónico. Todos se conectan directamente sin necesidad de herramientas, a excepción de los dirigidos al controlador de potencia que requieren destornillador de punta plana o en cruz de 2 mm y los que se unen a los pines de los motores, los componentes de la alimentación y el LDR, que se sueldan con estaño.

6.1.7.3. Cable USB

Para subir el código y comprobar el funcionamiento por el monitor serie es necesario el uso de un cable que conecte el ordenador con la placa. En este caso se ha de utilizar un cable USB de transmisión de datos micro B 2.0 (figura 11). El propio cable también sirve para cargar el dispositivo.

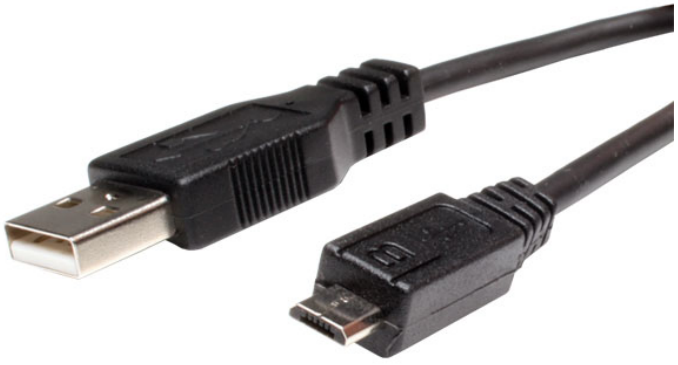


Figura 11: Cable USB a micro USB B 2.0

6.1.8. Alimentación

Como ya se ha estipulado, para suministrar energía al circuito, se va a dar uso de una batería junto con una serie de componentes que permitan tanto servirse de la batería como cargarla.

6.1.8.1. Batería

La batería que se elige es una célula recargable de polímero de iones de litio (ver figura 12) que cuenta con 2600 mAh y 3,7 V. Además, este componente incorpora un integrado de protección [17]. Teniendo en cuenta que el consumo medio de la electrónica es de 230 mA (valor medido experimentalmente), la independencia que ofrece es de 11,3 horas.

Figura 12: Batería de iones de litio 2600 mAh
Fuente: Amazon [17]

6.1.8.2. Convertidor

Dado que las baterías no proporcionan una diferencia de potencial constante, ya que se van descargando con el tiempo y uso, es necesario utilizar un convertidor DC-DC. En concreto se da uso al módulo MT3608 (figura 13) que es un convertidor *step-up* que permite, a coste de intensidad, entregar un voltaje constante y más alto a su salida [18]. Además, a través de un potenciómetro, es posible ajustar la tensión con gran exactitud. Para alimentar al proyecto, se gira el potenciómetro hasta alcanzar 7 V en vacío.

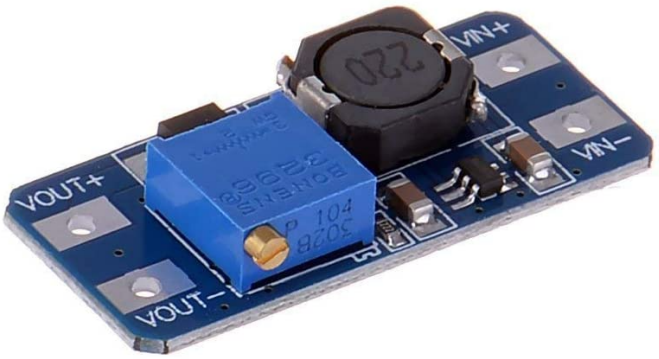


Figura 13: Convertidor DC-DC MT3608
Fuente: Amazon [18]

6.1.8.3. Cargador de batería

Para cargar la batería se utiliza el extendido módulo TP4056 cargador de baterías (figura 14) [19]. Dado que incorpora un puerto micro USB B 2.0, se puede cargar a Aurora con el mismo cable de programación.

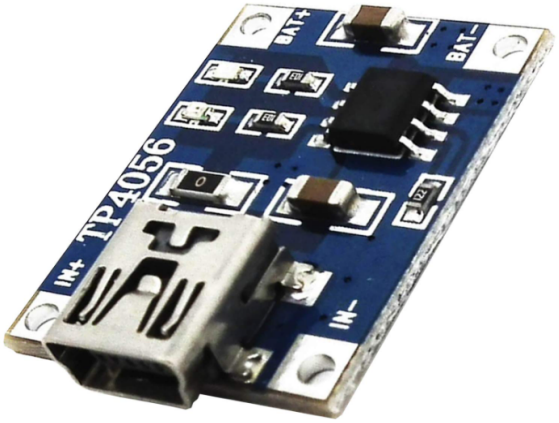


Figura 14: Cargador de batería TP4056
Fuente: Amazon [19]

6.1.8.4. Interruptor

Es importante poder cambiar entre tres estados la alimentación de Aurora: encendido, apagado y modo carga. Para conseguir esto, se incorpora al montaje un interruptor de 3 posiciones [20] como se puede ver en la imagen 15. La posición I activará a Aurora, la posición 0 la apagará y la II es el modo para cargar de batería a través del cable USB.

Figura 15: Interruptor triestado
Fuente: Amazon [20]

6.1.9. Diagrama de conexiones

En la tabla 1 se pueden consultar todas las conexiones entre los componentes de este subsistema. Finalmente, el montaje electrónico queda de forma análoga al representado en la figura 16.

LED: pin negativo	R1: borne 2
NodeMCU: pin D2	LED: pin positivo
NodeMCU: pin GND	R1: borne 1
NodeMCU: pin A0	TMP36: pin analógico
NodeMCU: pin 3V	TMP36: pin positivo
NodeMCU: pin GND	TMP36: pin negativo
NodeMCU: pin GND	Divisor resistivo: pin GND
NodeMCU: pin D5	Divisor resistivo: pin V1
NodeMCU: pin D6	Divisor resistivo: pin V2
NodeMCU: pin D7	Divisor resistivo: pin V3
NodeMCU: pin 3V	Divisor resistivo: pin 3,3 V
NodeMCU: pin D1	L298N: pin IN4
NodeMCU: pin D0	L298N: pin IN3
NodeMCU: pin VU	L298N: pin +5 V
NodeMCU: pin GND	L298N: pin GND
Motor DC: pin positivo	L298N: pin OUT3
Motor DC: pin negativo	L298N: pin OUT4
MT3608: pin VOUT -	L298N: pin GND
MT3608: pin VOUT +	L298N: pin +12 V
MT3608: pin VIN -	Interruptor: pin I
MT3608: pin VIN +	Batería: pin +
Interruptor: pin 0	Batería: pin -
TP4056: B+	Batería: pin +
TP4056: B-	Interruptor: pin II

Tabla 1: Interconexiones entre componentes electrónicos

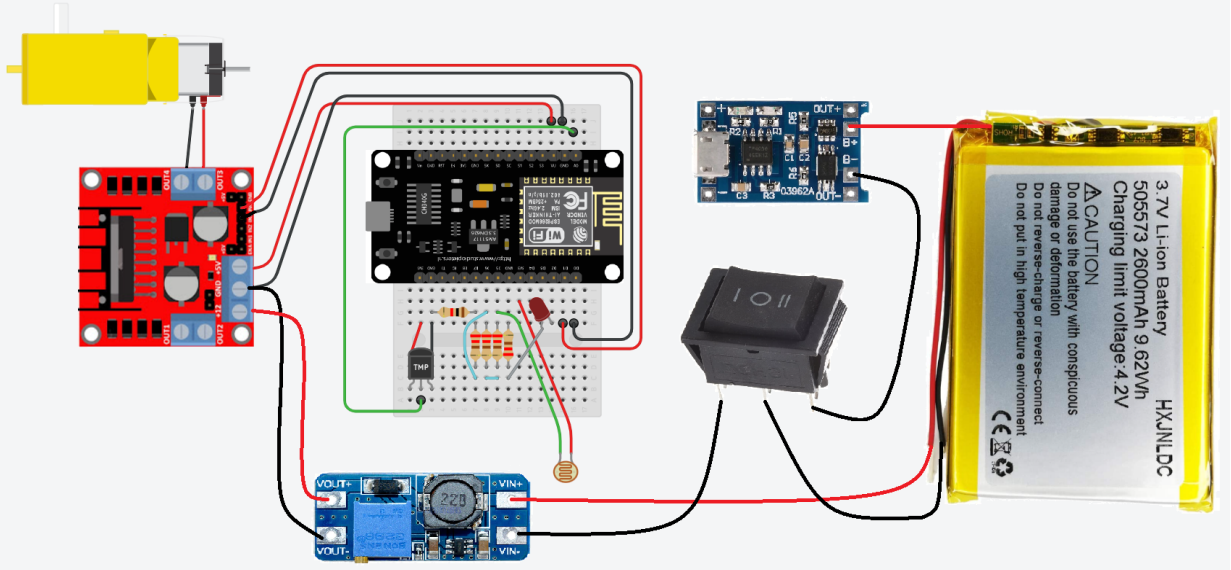


Figura 16: Diagrama de conexiones electrónico

6.2. Subsistema de programación

La configuración de todo el apartado de *software* requiere disponer de acceso a la web de la nube de Arduino, el programa Arduino IDE y la aplicación de Amazon Alexa. Se procede en este apartado a explicar cada punto del subsistema de programación.

6.2.1. Arduino IoT Cloud

Para poder trabajar con las herramientas en línea que ofrece Arduino Cloud es necesario registrarse. Una vez creada la cuenta, hay que acceder al apartado de la nube de Arduino IoT situado en la página principal (figura 17). A continuación, aparece una lista con todas las "cosas" creadas por la cuenta. En un principio no habrá ningún objeto asociado al perfil, hay que dirigirse a la opción *create thing,* como puede verse en la figura 18.

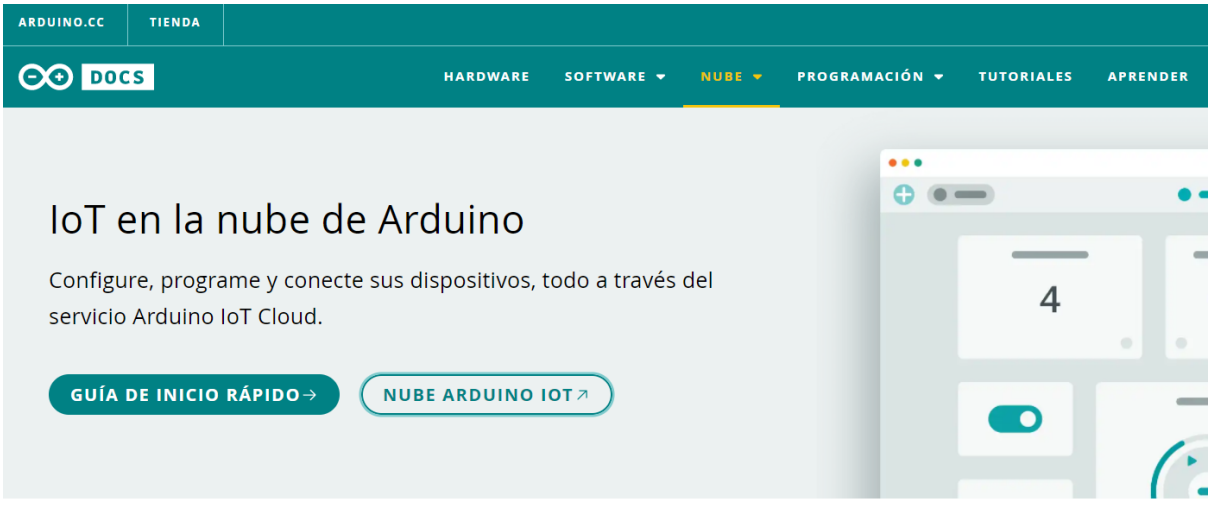


Figura 17: Página principal Arduino IoT Cloud

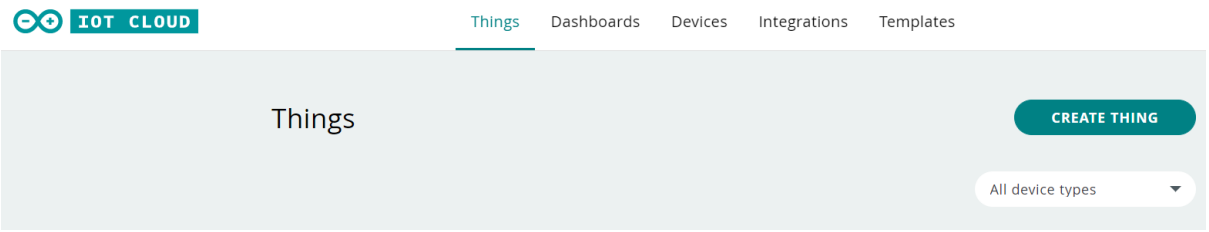


Figura 18: Menú de creación de objetos IoT

Entonces se genera una nueva “cosa” con un *thing ID*, que es un número de identificación único asignado de forma automática por la página. La siguiente tarea es determinar un nombre, un dispositivo, una zona horaria, las variables asociadas a la “cosa” y una red de internet. El aspecto inicial del objeto virtual debería ser como el de la figura 19.

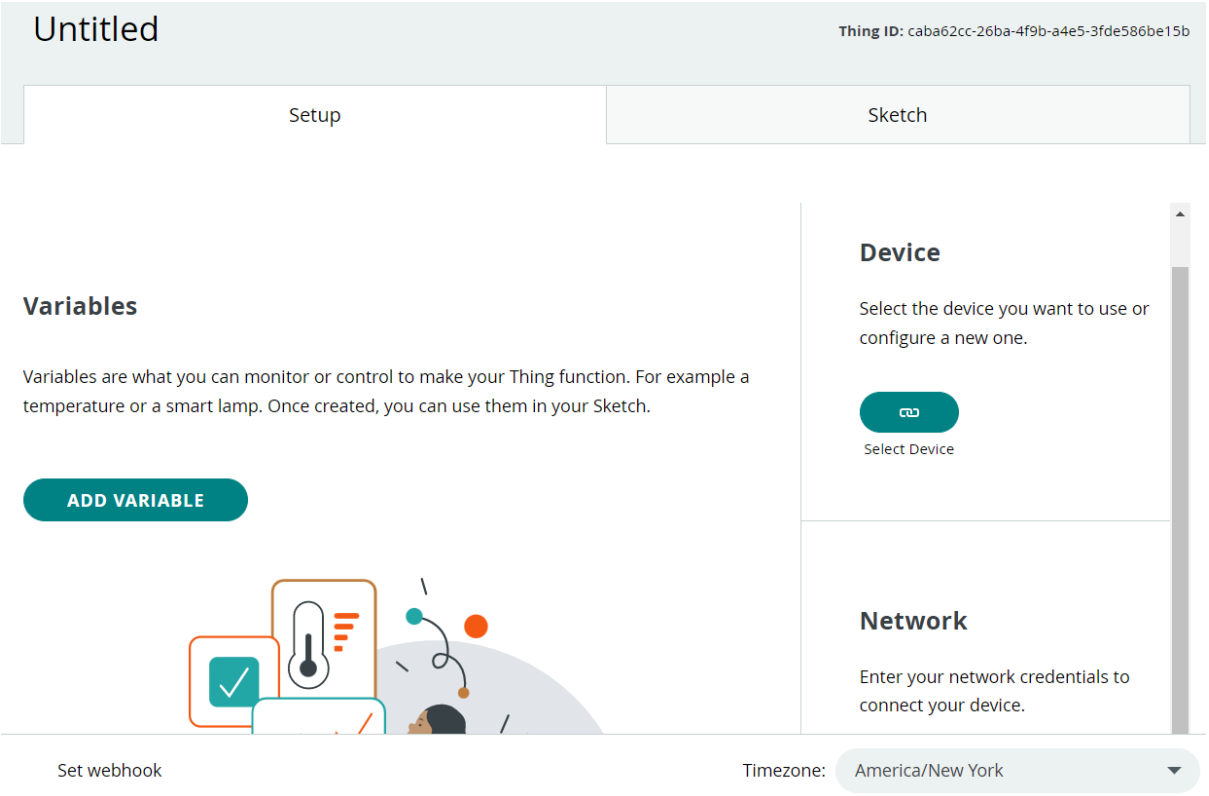


Figura 19: Aspecto inicial del objeto virtual

A este objeto se le ha puesto el nombre de “Aurora” y se le debe asignar un dispositivo con el que se va a programar el proyecto. Dado que inicialmente no hay dispositivos guardados, se selecciona uno nuevo. Como se ha mencionado con anterioridad, este proyecto utiliza la placa de desarrollo NodeMCU que lleva integrada la ESP8266, por tanto, se seleccionará la opción de dispositivos de terceros compatibles con Arduino. Asimismo, se han de seleccionar las opciones acordes a las especificaciones comentadas (figura 20). A continuación, se le asigna un nombre al dispositivo; en este caso, NodeMCU.

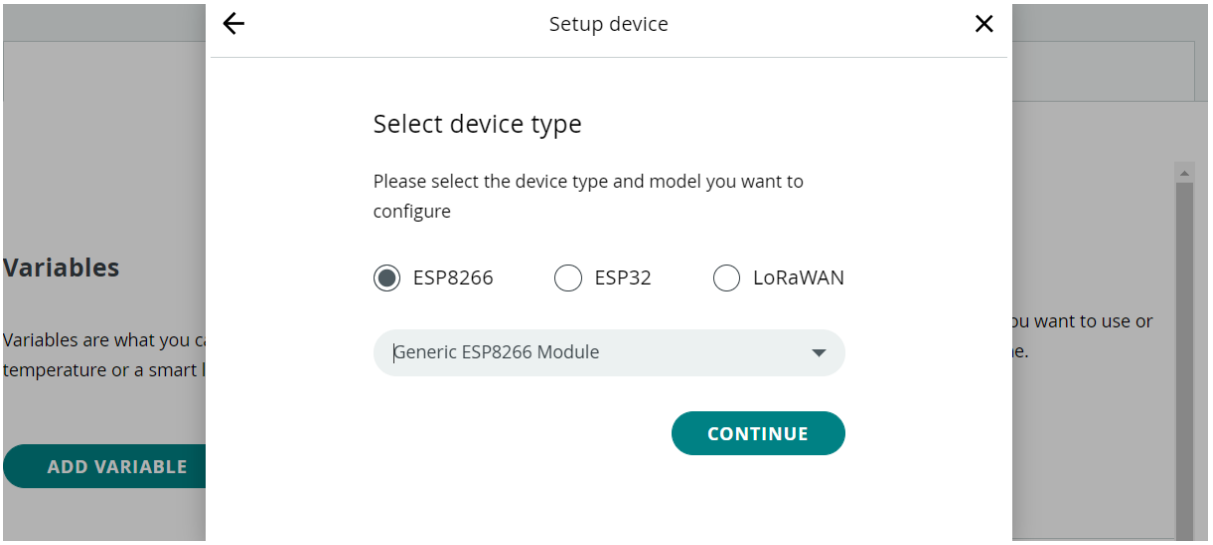


Figura 20: Selección de la placa de desarrollo

La página informará de la asignación de un número de serie y una clave para el dispositivo nuevo, datos que es importante guardar para su posterior uso. En la figura 21 pueden verse los datos comentados, así como la posibilidad de descargar un PDF con esa misma información.

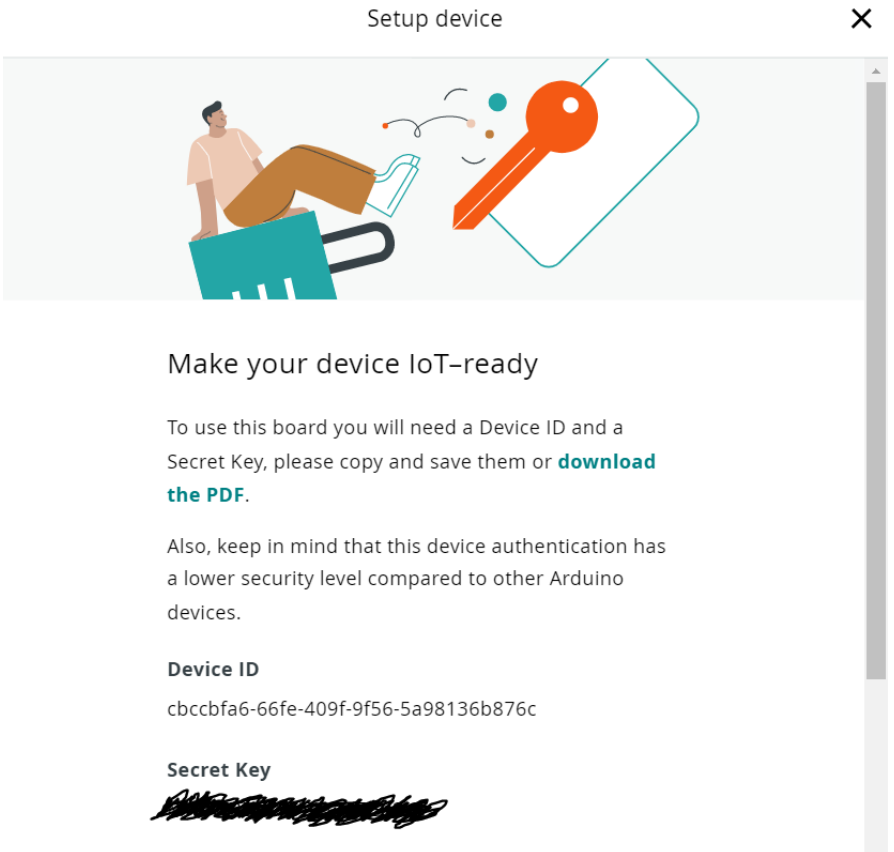


Figura 21: Cartel informativo del número y clave del dispositivo

En la esquina inferior derecha del menú se encuentra la opción de cambiar la zona horaria. Para este proyecto se selecciona la de Europa/Madrid.

El paso más relevante en este apartado es la creación de las diferentes variables que van a definir qué propiedades va a tener el objeto. Como se ha comentado previamente, Aurora va a tener una función de abrir y cerrar una cortina, ajustarse a la longitud de la barra, telemetría de temperatura y un comportamiento variable según nivel de luz. En la nube de Internet of Things de Arduino todas estas funciones se pueden compactar en tres variables.

La primera, en este caso, indica si la cortina está abierta o cerrada, a la vez que contiene un parámetro numérico auxiliar que representa la longitud en decímetros de la barra. Arduino no tiene la posibilidad de crear este tipo de variables desde cero en su nube, pero ofrece una serie de tipos compatibles con Alexa, de los cuales Dimmed Light cumple con lo que el proyecto necesita. El nombre que se le asigne a la variable será el mismo con el que el usuario se referirá oralmente a ella al usar la función. Por tanto, se elige “luz_natural” como nombre de la primera variable, permitiendo su lectura y escritura, como se puede ver en la figura 22. La última opción que es necesario configurar es el modo en el que la función de la variable se va actualizar; en este caso, se selecciona *on change,* para que esto ocurra cuando la propiedad se modifique.

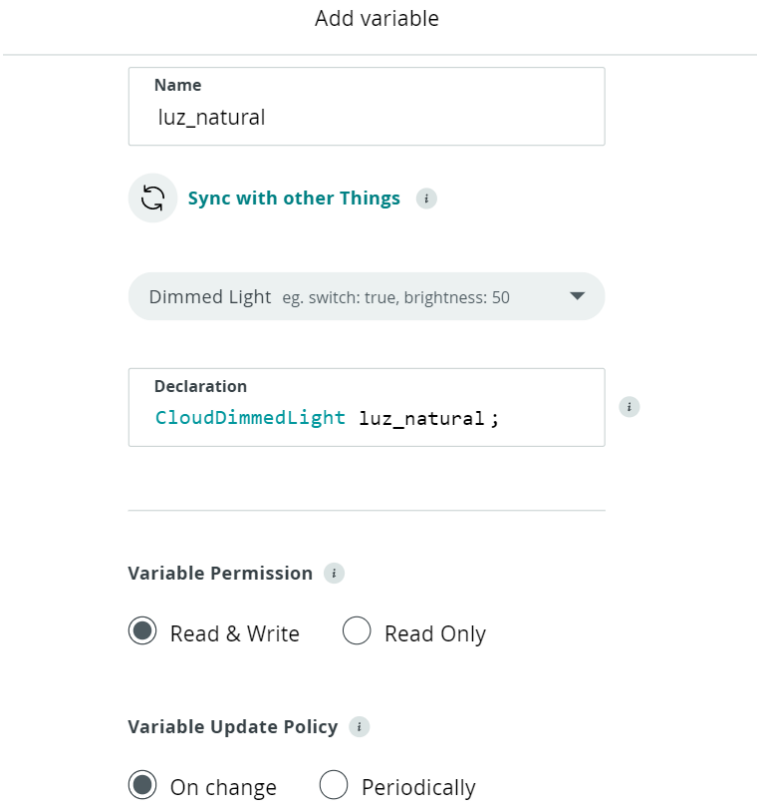


Figura 22: Creación de la variable “luz_natural”

A continuación, como puede verse en la figura 23, se crea la segunda variable con el nombre de “temperatura”. El tipo seleccionado será *Temperature Sensor* en grados Celsius y, como se trata de una propiedad de telemetría, solo se le podrá consultar su valor. La actualización del valor de la temperatura en la nube será cuando este varíe 1 ºC o más.

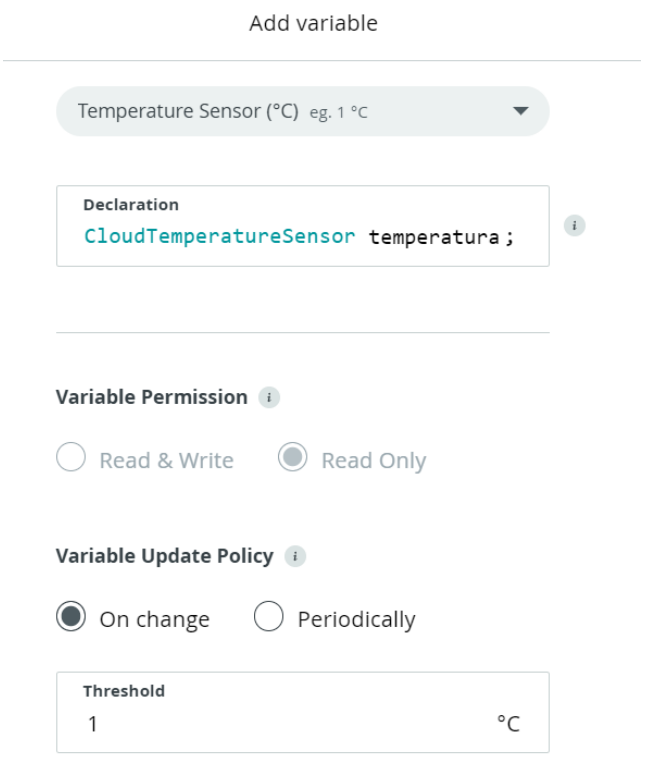


Figura 23: Creación de la variable “temperatura”

La última propiedad que hay que configurar es la que se encarga de activar o desactivar el sensor de luz para que la cortina se comporte de forma automática en el amanecer. Como el nivel de luminosidad al que reacciona Aurora tiene que poder ser regulado por el usuario, el tipo ha de ser *Dimmed Light* (véase figura 24).

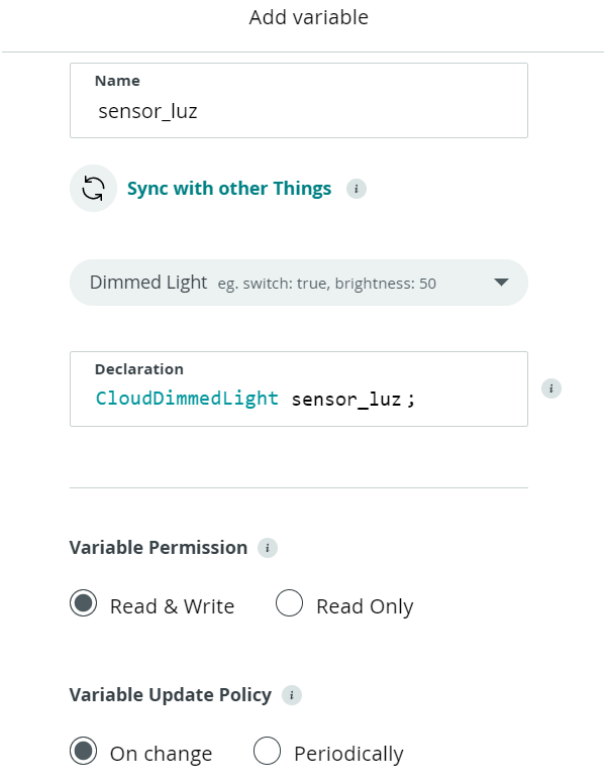


Figura 24: Creación de la variable “sensor_luz”

Para que el objeto virtual esté correctamente creado se le asigna una red de internet Wi-Fi en el apartado *network.* Es necesario introducir el nombre, la contraseña y la clave secreta del dispositivo, la cual ha sido generada previamente. Al finalizar los ajustes, el objeto debe quedar con una apariencia similar a la que se ve en la figura 25.

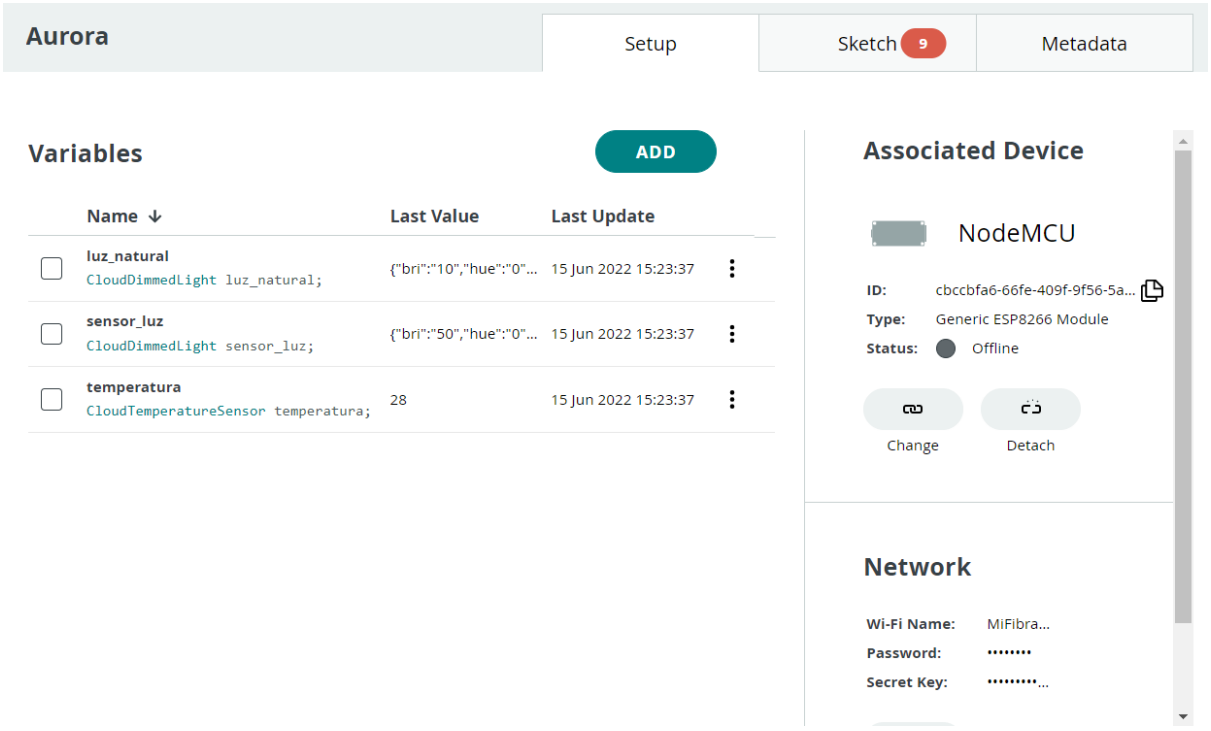


Figura 25: Aspecto final del objeto virtual Aurora

Se habrá generado un código plantilla en la pestaña de *sketch*. Al entrar en ella y posteriormente abrir el editor virtual completo, Arduino permite descargar todos los archivos y bibliotecas con los que se va a seguir trabajando. A pesar de que se hayan declarado la primera y la segunda variable del objeto de forma similar, en el siguiente apartado se podrá ver cómo sus comportamientos y programación en Arduino son totalmente diferentes.

6.2.2. Arduino IDE

En este apartado se configura y programa toda la parte del código. En primer lugar, se accede al archivo tipo INO y se comprueba la existencia del programa principal y de dos librerías. En segundo lugar, antes de empezar a editar, se debe incluir la placa del proyecto, instalar el controlador de la ESP8266, descargar su respectiva biblioteca y configurar las herramientas de trabajo.

Para ello, como se ve en la figura 26, en el menú “Archivo”, dentro del submenú “Preferencias”, en el apartado “Gestor de URLs adicionales de tarjetas” se debe introducir la ubicación URL: <https://arduino.esp8266.com/stable/package_esp8266com_index.json>. Esto se obtiene de la página web de GitHub, buscando en ella el apartado de Arduino en ESP8266, en la opción de instalación de administrador de tableros, disponible en el siguiente enlace: <https://github.com/esp8266/Arduino#installing-with-boards-manager>.

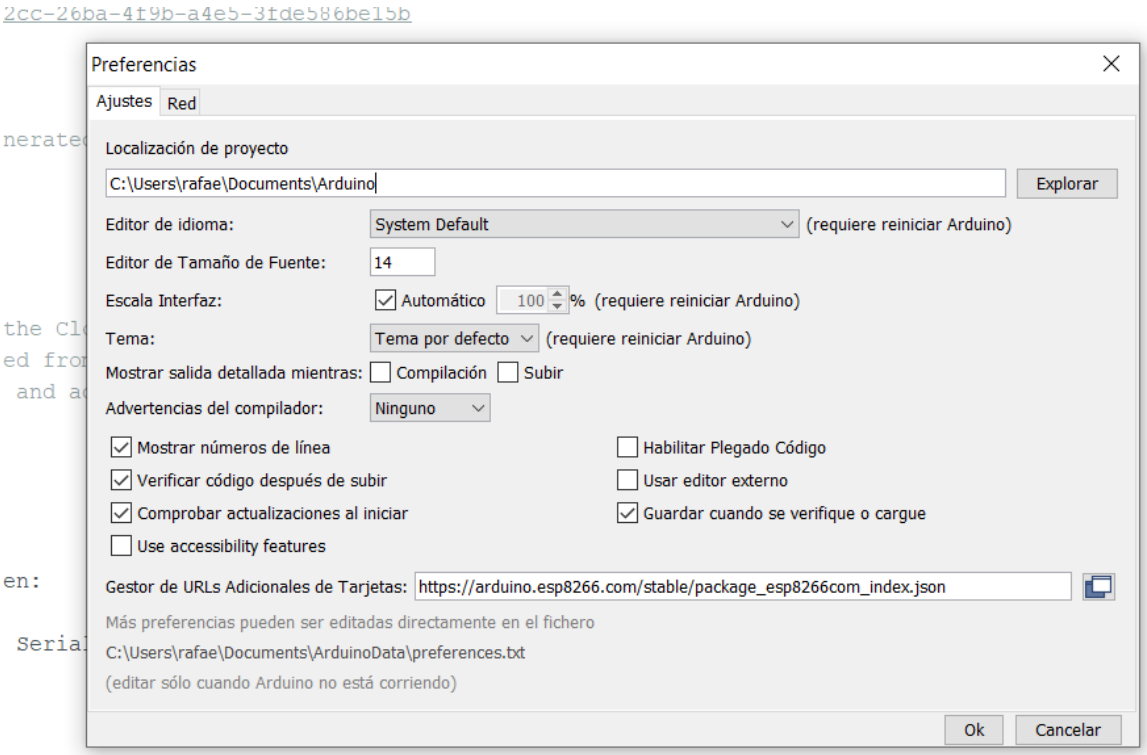


Figura 26: Configuración de la librería ESP8266

El controlador de la placa tiene como nombre CH340G *driver*. Este se puede descargar de múltiples fuentes; para este proyecto se ha obtenido de la web de Skulltrap. Una vez dentro de la página se busca el archivo y, teniendo la placa conectada, se descarga e instala desde este *link*: <http://www.skulltrap.co/2019/09/configurar-driver-nodemcu-v30-en-ide.html>.

El siguiente paso es buscar y descargar la biblioteca de la ESP8266. En el programa de Arduino IDE, en el menú “Programa”, dentro de “Incluir librería”, se accede al apartado de administración de bibliotecas. En este punto se busca el módulo ESP8266 y se procede a su instalación.

Para acabar con la configuración inicial, se han de seleccionar algunas opciones del menú “Herramientas” (véase figura 27). La placa que conviene elegir es la “Generic ESP8266 Module”, la velocidad de comunicación serie sería 115200 baudios y el puerto, el correspondiente al conectar la placa.

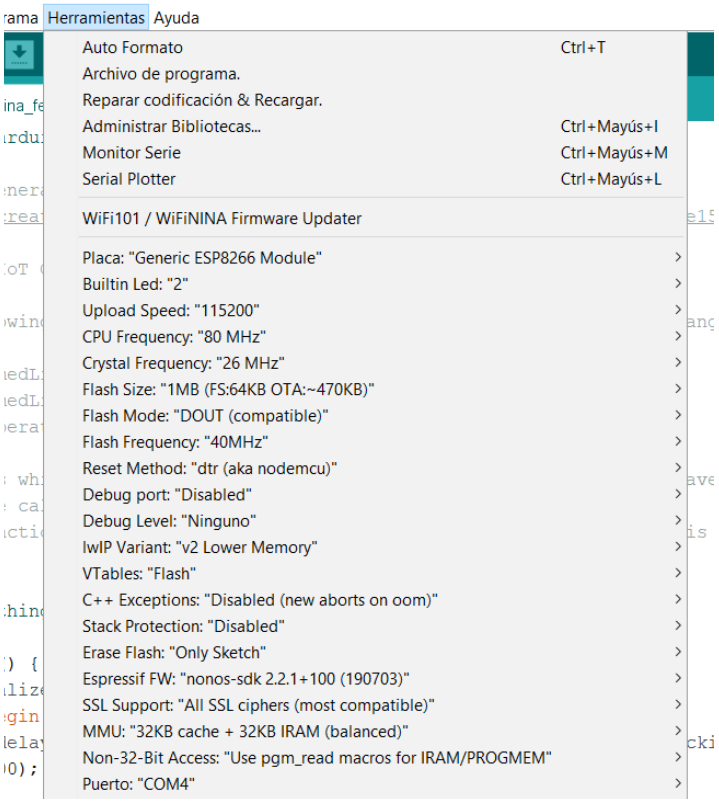


Figura 27: Configuración de las herramientas de Arduino IDE

Aparte del programa principal se tienen dos librerías. Una de ellas es “arduino_secrets.h”, que contiene la breve definición de las variables privadas de la clave del dispositivo, el nombre de la red y su contraseña, como se puede apreciar en la figura 28. La otra biblioteca es “thingProperties.h” (véase figura 29), que contiene el código generado por la nube de Arduino IoT. Esta última librería no se va a modificar y su función es definir la programación de todas las propiedades y características creadas en el anterior apartado, así como su conexión a la nube de Arduino.

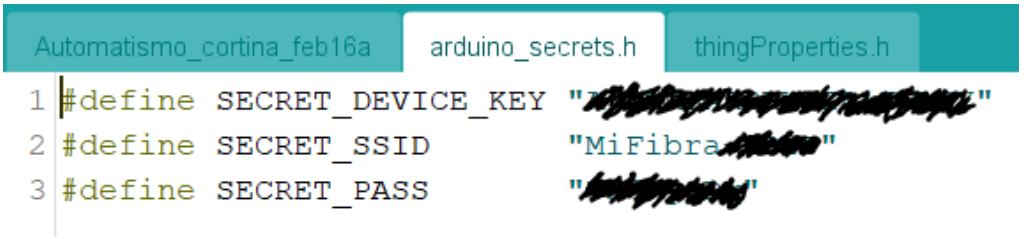


Figura 28: Biblioteca “arduino_secrets.h”

Automatismo_cortina_feb16a | arduino_secrets.h | thingProperties.h

```
#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>

const char THING_ID[]           = "caba62cc-26ba-4f9b-a4e5-3fde586be15b";
const char DEVICE_LOGIN_NAME[]  = "cbccbfa6-66fe-409f-9f56-5a98136b876c";

const char SSID[]               = SECRET_SSID;    // Network SSID (name)
const char PASS[]               = SECRET_PASS;    // Network password (use for WPA, or use as key for WEP)
const char DEVICE_KEY[]  = SECRET_DEVICE_KEY;    // Secret device password

void onLuzNaturalChange();
void onSensorLuzChange();

CloudDimmedLight luz_natural;
CloudDimmedLight sensor_luz;
CloudTemperatureSensor temperatura;

void initProperties(){

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.setThingId(THING_ID);
  ArduinoCloud.addProperty(luz_natural, READWRITE, ON_CHANGE, onLuzNaturalChange);
  ArduinoCloud.addProperty(sensor_luz, READWRITE, ON_CHANGE, onSensorLuzChange);
  ArduinoCloud.addProperty(temperatura, READ, ON_CHANGE, NULL, 1);

}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

Figura 29: Biblioteca “thingProperties.h”

Cabe comentar que se ha escrito el código del programa principal de forma que no existan esperas en el bucle infinito. Esto supone el uso de variables como marcas temporales, periodos de funcionamiento y la función “millis” de Arduino, la cual devuelve el valor del tiempo en milisegundos desde que empezó el programa. Esta función desborda a los cincuenta días aproximadamente, ya que el valor máximo del *unsigned long* que devuelve es de 4 294 967 295 milisegundos. En una versión comercial del producto sería necesario tener esto en cuenta en el bucle infinito para evitar un posible fallo puntual. Una forma de hacerlo es revisar el valor de la función “millis” y actualizar las variables temporales según corresponda antes del desbordamiento. En este proyecto se elude esta implementación, ya que el prototipo Aurora no va ser usado durante tanto tiempo continuado.

Para visualizar el funcionamiento del programa, se puede ver en la figura 30 un diagrama de flujo del comportamiento, a grandes rasgos, del programa de arduino IDE. Además, en la figura 31 se representa la interacción con internet y Alexa, es decir, la actualización de los datos de la nube de Arduino tanto de entrada como de salida a través del IoT.

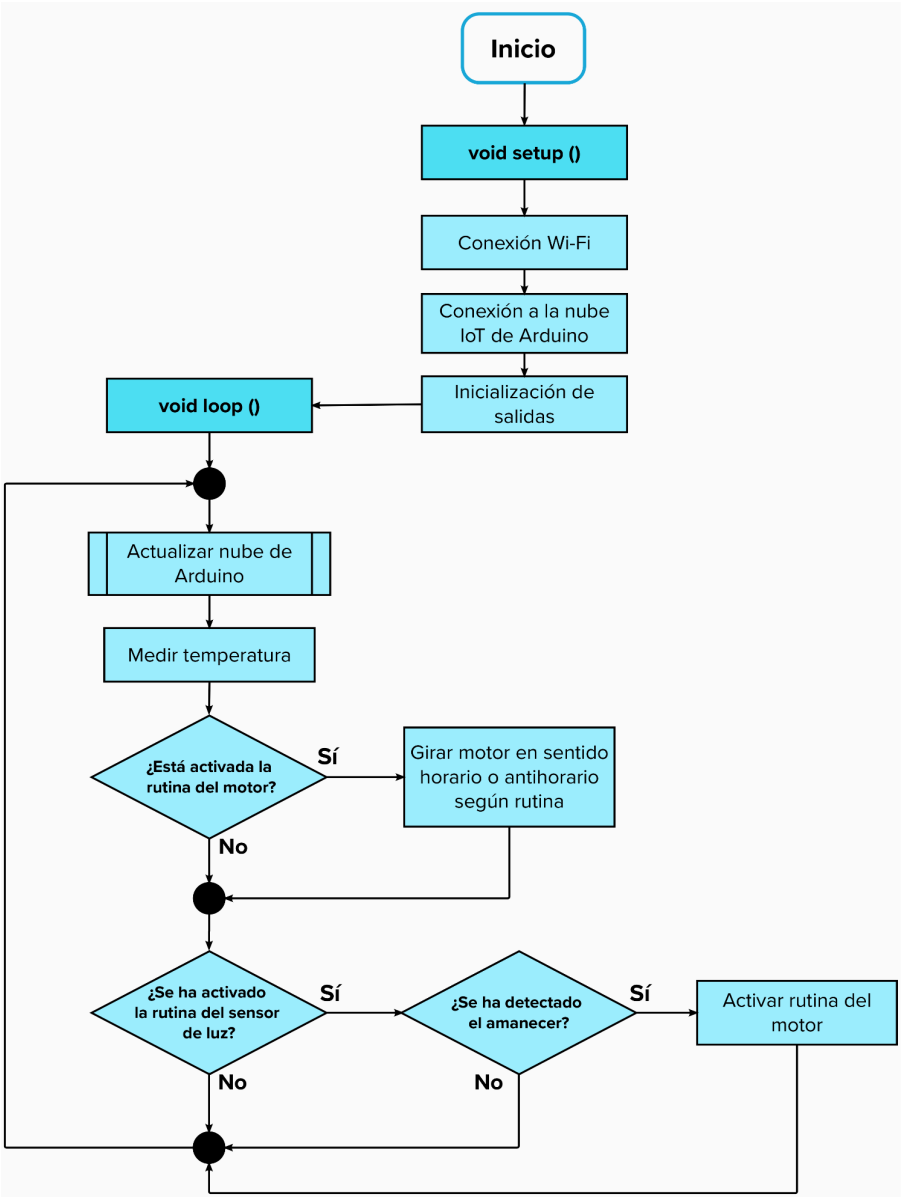


Figura 30: Diagrama de flujo simplificado del programa de Arduino IDE

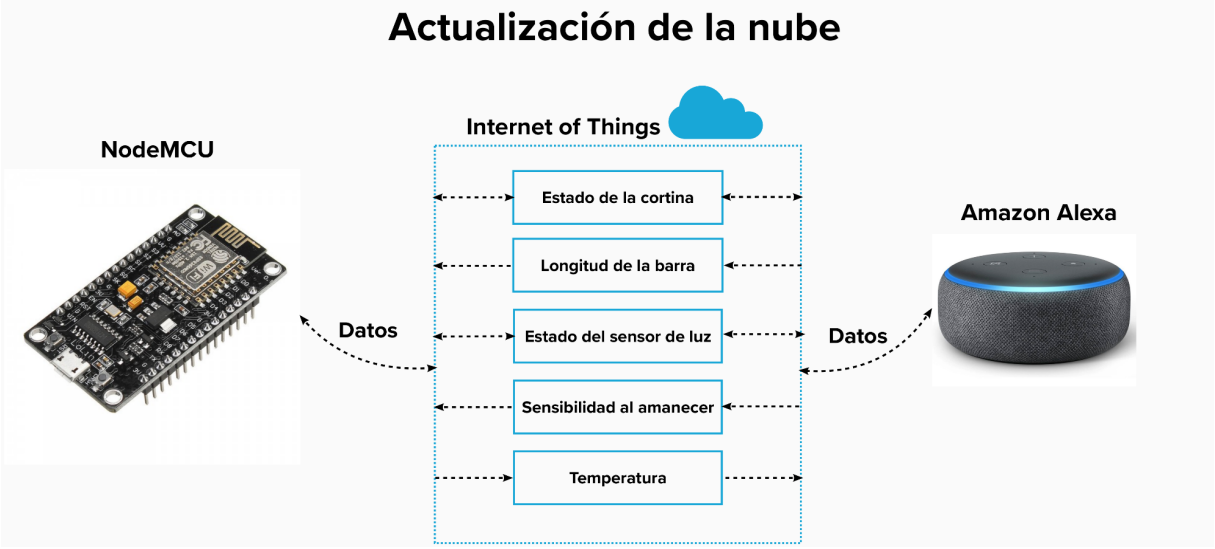


Figura 31: Comunicación entre Alexa y la placa de desarrollo

Partiendo de lo anterior, para comprender mejor la estructura del programa principal del Arduino IDE se ha construido un organigrama de contenido en la figura 32, en el que se puede visualizar de forma rápida la composición del cuerpo de todo el código.

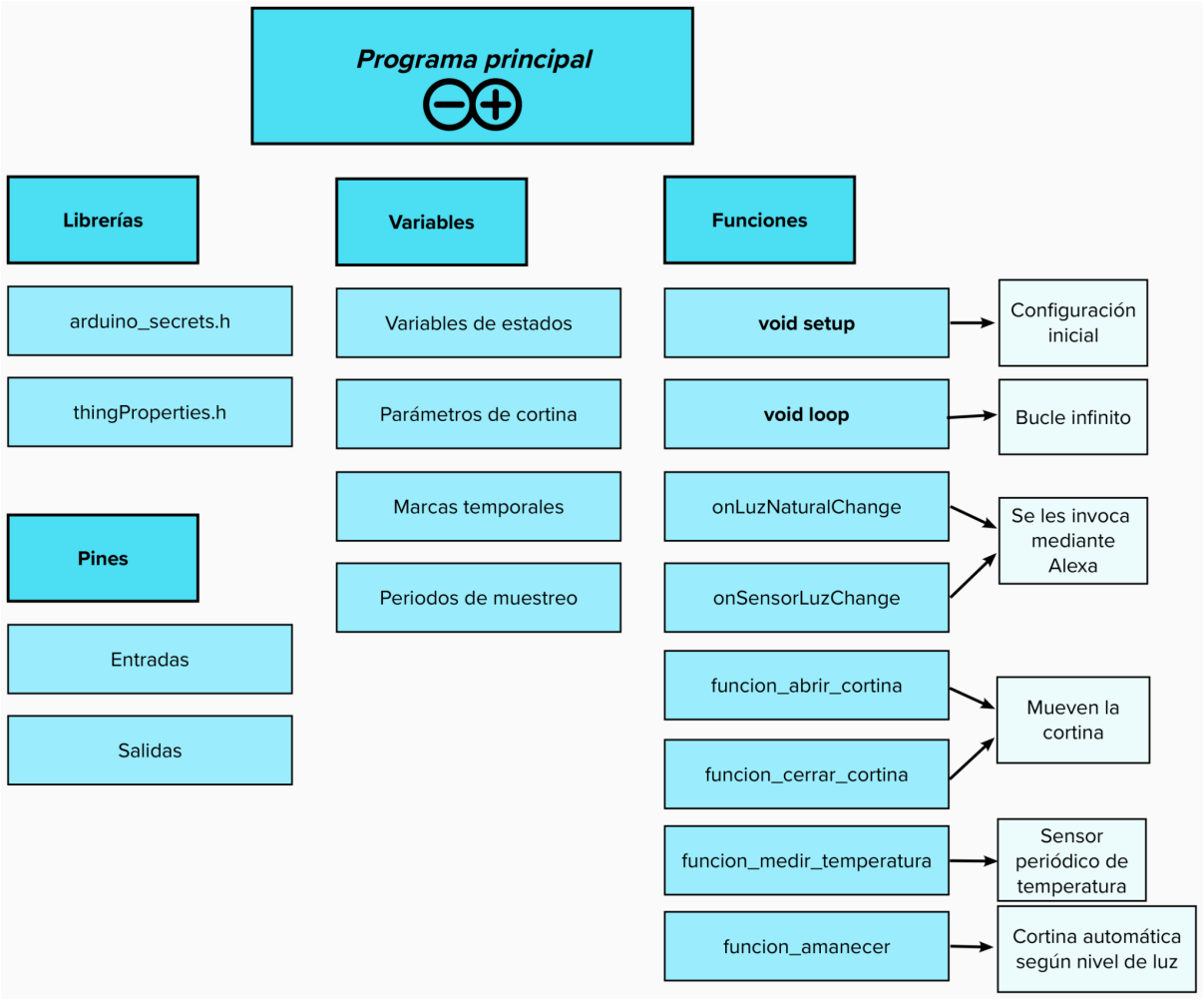


Figura 32: Organigrama de contenido del programa principal

6.2.2.1. Definiciones

En primer lugar se incluyen las librerías ya mencionadas de “arduino_secrets.h” y “thingProperties.h”, para que el programa disponga de sus códigos (véase figura 33). En segundo lugar, se definen las constantes que relacionan los números de pines de entradas y salidas de propósito general o GPIO (General Purpose Input/Output) del chip con los pines que se van a usar en la placa de desarrollo. La correspondencia de pines viene dada en la hoja de especificaciones.

A continuación, se les asigna a todos los pines el nombre de la entrada o salida que les corresponde. En total, se tienen tres salidas: giro horario del motor “MotorH”, giro antihorario “MotorA” y un LED indicador de funcionamiento “Pin_Led”. Por otra parte, hay un total de cuatro entradas: los tres pines de lectura digital del nivel de luz “Luz1”, “Luz2” y “Luz3” y el convertidor de analógico a digital o ADC (Analogic to Digital Converter) de 10 bits “Sensor_Temperatura”. Cabe comentar que el nombre del pin “A0” del chip, el único convertidor ADC del módulo, coincide con el impreso en la placa de desarrollo; es el motivo de que se haya prescindido de su definición previa. Toda esta declaración de los pines podría ser omitida si se usaran directamente los números del GPIO sobre las funciones de

lectura y escritura; pero, para un mejor entendimiento del código y una mayor coherencia en el montaje electrónico, se ha decidido proceder con lo explicado en un principio.

```
1  //Librerias
2  #include "arduino_secrets.h"
3  #include "thingProperties.h"
4
5
6  //Definicion de pines
7  #define D0   16   //GPIO16
8  #define D1   5    //GPIO5
9  #define D2   4    //GPIO4
10 #define D5   14   //GPIO14
11 #define D6   12   //GPIO12
12 #define D7   13   //GPIO13
13
14
15 //Salidas
16 const int MotorH = D0;
17 const int MotorA = D1;
18 const int Pin_Led = D2;
19
20 //Entradas
21 const int Sensor_Temperatura = A0;//ADC 10 bits
22 const int Luz1 = D5;
23 const int Luz2 = D6;
24 const int Luz3 = D7;
25
```

Figura 33: Declaraciones iniciales en Arduino IDE

A continuación, como se puede ver en la figura 34, se definen las diferentes variables auxiliares que van a aparecer a lo largo de todo el código. Las variables de estado son las que se encargaran de informar de la etapa de funcionamiento según el caso:

- “rutina_motor” tendrá tres estados posibles:
  - 0: motor parado.
  - 1: motor abriendo la cortina.
  - 2: motor cerrando la cortina.
- “rutina_sensor_luz” tendrá 4 estados posibles:
  - 0: sensor inactivo.
  - 1: sensor activo a baja intensidad.
  - 2: sensor activo a media intensidad.
  - 3: sensor activo a alta intensidad.
- “rutina_amanecer” tendrá dos estados:
  - 0: no se detecta suficiente luz.
  - 1: se ha detectado luz suficiente.

Seguidamente, se define la variable “dm_long_barra”, la cual representa la longitud en decímetros de la cortina. Este parámetro tendrá el valor inicial de 10 decímetros.

Se añaden finalmente las definiciones de los periodos de funcionamiento identificables con el prefijo “periodo” y las marcas temporales con el prefijo “prev”. Los sensores miden

variables de carácter ambiental; esto significa que tienen una dinámica lenta y pueden permitirse periodos de muestreo largos sin que afecte a la calidad del sistema. Se han elegido, por tanto, tres segundos para la temperatura (3 000 milisegundos) y diez segundos para la luz (10 000 milisegundos). El tiempo que el motor ha de estar en funcionamiento "periodo_corriendo" para abrir o cerrar completamente la cortina será proporcional a la longitud de la barra dividida entre una constante de valor "0,001", que representa la velocidad con la que se mueve el dispositivo por la barra en dm/ms (1 dm/s). Evidentemente, la velocidad de movimiento depende de múltiples variables mecánicas que no pueden predecirse para este dispositivo adaptable. Es por ello que se selecciona un valor representativo de velocidad y, si en la aplicación real el recorrido de Aurora no se ajusta a la longitud establecida, se le asigna más o menos decímetros de movimiento en función de la necesidad.

```
27 //Variables auxiliares
28 int rutina_motor = 0;
29 int rutina_sensor_luz=0;
30 bool rutina_amanecer = false;
31
32 int dm_long_barra = 10;
33
34 unsigned long periodo_corriendo = dm_long_barra /0.001;
35 unsigned long periodo_muestreo_temperatura = 3000;
36 unsigned long periodo_muestreo_luz = 10000;
37
38 unsigned long prev_cortina = 0;
39 unsigned long prev_temperatura = 0;
40 unsigned long prev_luz = 0;
41
```

Figura 34: Variables auxiliares en Arduino IDE

6.2.2.2. La configuración inicial

Como es común en los programas de Arduino IDE, se tienen por defecto las funciones "void setup" y "void loop". La primera función solo se ejecuta una vez al enchufar la placa y se encarga de hacer las configuraciones iniciales. En la figura 35 se puede visualizar como la nube de arduino ha rellenado parte de esta primera función de forma automática. En general, estas líneas de código se encargan de preparar el puerto serie, cargar las propiedades de la biblioteca "thingProperties.h", iniciar la comunicación con la nube de Arduino IoT a través de la red Wi-Fi y mostrar por el monitor serie información referente a estos procesos.

```
44 void setup() {
45   // Initialize serial and wait for port to open:
46   Serial.begin(115200);
47   // This delay gives the chance to wait for a Serial Monitor without
48   //blocking if none is found
49   delay(1500);
50
51   // Defined in thingProperties.h
52   initProperties();
53
54   //Arduino Internet of Things
55   ArduinoCloud.begin(ArduinoIoTPreferredConnection);
56     /*
57      The following function allows you to obtain more information
58      related to the state of network and IoT Cloud connection and errors
59      the higher number the more granular information you'll get.
60      The default is 0 (only errors).
61      Maximum is 4
62     */
63   setDebugMessageLevel(4);
64   ArduinoCloud.printDebugInfo();
65
```

Figura 35: Primera parte de la función de la configuración inicial

Antes de acabar la función "void setup", se escriben una serie de líneas que inicializan todas las salidas de la placa. Estos pines son los del motor y el LED, como puede verse en la figura 36.

```
66   //Inicializar pines
67   pinMode(Pin_Led, OUTPUT);
68   digitalWrite(Pin_Led, LOW);
69
70   pinMode(MotorA, OUTPUT);
71   pinMode(MotorH, OUTPUT);
72   digitalWrite(MotorA, LOW);
73   digitalWrite(MotorH, LOW);
74
75 }
76
```

Figura 36: Segunda parte de la función de la configuración inicial

6.2.2.3. El bucle infinito

En el bucle infinito "void loop" se van a estar ejecutando las líneas de código visibles en la figura 37 de forma periódica. Se trata, por tanto, del corazón del programa principal, lo que a grandes rasgos va a estar realizando continuamente la placa.

La herramienta "ArduinoCloud.update" con la que empieza el bucle se encarga de llamar a las funciones generadas por Arduino Cloud si se ha dado algún cambio a través de Alexa. Estas funciones son las llamadas "onLuzNaturalChange" y "onSensorLuzChange" y su funcionamiento se explicará más adelante. Cabe comentar que la herramienta en cuestión

también se encarga de actualizar el valor en la telemetría de la temperatura en el caso de que varíe por encima del grado centígrado.

A continuación el programa ejecuta la función de medir la temperatura y luego las funciones de abrir o cerrar la cortina según la rutina establecida. Por último, también dependiendo de la rutina, se ejecutará la función del amanecer que controla el comportamiento automático según el nivel de luz. La estructura y comportamiento de todas estas funciones complementarias serán explicados en el apartado siguiente.

```
78 void loop() {
79
80   ArduinoCloud.update();
81
82   funcion_medir_temperatura();
83
84   if(rutina_motor == 1)
85   funcion_abrir_cortina();
86
87   if(rutina_motor == 2)
88   funcion_cerrar_cortina();
89
90   if(rutina_sensor_luz)
91   funcion_amanecer();
92
93 }
```

Figura 37: Bucle infinito

6.2.2.4. Las funciones complementarias

La función “onLuzNaturalChange” se demandará cuando un usuario esté utilizando Alexa y provoque un cambio en la propiedad “luz_natural”: activando la cortina o ajustando el tamaño de la misma. Esta función empieza mostrando un mensaje propio por el monitor serie y luego actualiza los valores del periodo de ejecución del motor y del tamaño de la barra. Es importante recalcar que el tamaño de la barra se obtiene del parámetro auxiliar del que dispone el tipo de objeto que se eligió en el apartado de Arduino IoT Cloud, que tiene por defecto el nombre de *brightness*. Esto se debe a que Arduino interpreta que el objeto es un sistema de iluminación ajustable a pesar de que la realidad sea bien diferente.

La función continúa con la asignación de la rutina de la cortina la cual dependerá de si está activado o desactivado el *switch* de la luz natural. Y, por último, se establece la marca temporal de la ejecución de la cortina, ya que a partir de ese momento se llamará a alguna de las funciones encargadas de poner en funcionamiento el motor. En la figura 38 puede verse la estructura de la función mencionada.

```
95 void onLuzNaturalChange()  {
96
97   Serial.println("onLuzNaturalChange ...");
98
99   //Longitud de la cortina y calculo del tiempo de ejecucion
100  dm_long_barra = luz_natural.getBrightness();
101  periodo_corriendo = dm_long_barra * 1000;
102
103  if(luz_natural.getSwitch())
104  rutina_motor = 1;
105  else
106  rutina_motor = 2;
107
108  prev_cortina = millis();
109
110 }
```

Figura 38: Función “onLuzNaturalChange”

Las funciones que se encargan directamente de abrir y cerrar la cortina tienen una sintaxis análoga, con la pequeña diferencia de que el sentido de giro del motor se invierte. En las figuras 39 y 40 se muestran los códigos de abrir y cerrar respectivamente, aunque cabe recordar que son acciones relativas, es decir, que dependen del entorno. En otras palabras, el usuario puede que quiera abrir su cortina de derecha a izquierda o a la inversa; por tanto, el giro horario o antihorario del motor no tendrá una relación fija con la apertura o cierre de la cortina. En el programa se elige una correspondencia concreta, pero con el uso físico del dispositivo esta relación depende de la orientación de instalación, pues el aparato se puede colgar de la barra al derecho o al revés.

Tras las aclaraciones expuestas, se procede a la explicación de la estructura de las funciones de abrir y cerrar la cortina. En primer lugar, se puede encontrar una condición en la que se compara si el tiempo desde que se inició la rutina (realizando la diferencia entre el tiempo actual “millis” y la marca temporal “prev_cortina”) es menor que el periodo que debe durar el movimiento del motor “periodo_corriendo”. Si dicha condición se cumple, se activa a nivel alto un pin del motor, dejando el otro a nivel bajo, para generar el giro, y se encenderá el LED mientras dure el proceso. En caso contrario, todos los anteriores pines se apagan y se restablece la rutina del motor. En ambos casos se mostrará un mensaje informativo por el monitor serie según la fase del proceso.

```
113 void funcion_abrir_cortina() {
114
115   if ((millis() - prev_cortina) <= periodo_corriendo) {
116
117     digitalWrite(MotorH, LOW);
118     digitalWrite(MotorA,HIGH);
119
120     digitalWrite(Pin_Led, HIGH);
121     Serial.println("Abriendo cortina...");
122
123   }
124   else {
125     digitalWrite(MotorH, LOW);
126     digitalWrite(MotorA, LOW);
127
128     digitalWrite(Pin_Led, LOW);
129     Serial.println("Cortina abierta.");
130     rutina_motor = 0;
131   }
132 }
```

Figura 39: Función “funcion_abrir_cortina”

```
134 void funcion_cerrar_cortina() {
135
136   if ((millis() - prev_cortina) <= periodo_corriendo) {
137
138     digitalWrite(MotorH,HIGH);
139     digitalWrite(MotorA, LOW);
140
141     digitalWrite(Pin_Led, HIGH);
142     Serial.println("Cerrando cortina...");
143
144   }
145   else {
146     digitalWrite(MotorH, LOW);
147     digitalWrite(MotorA, LOW);
148
149     digitalWrite(Pin_Led, LOW);
150     Serial.println("Cortina cerrada.");
151     rutina_motor = 0;
152   }
153 }
154
```

Figura 40: Función “funcion_cerrar_cortina”

La función que se encarga de medir la temperatura (figura 41) tiene también una condición encargada de ejecutar o no su contenido. En este caso la marca temporal del sensor está dentro de la propia condición, actualizándose en cada medición. Se consigue, por tanto, una medición cada 3 segundos (periodo de muestreo de la temperatura). Para proceder a la medición se realiza una lectura analógica del pin del ADC llamado “Sensor_temperatura” y se guarda en una variable auxiliar “temp” de tipo entero. Como el convertidor es de 10 bits, se van a poder recibir valores de entre 0 y 1023. El valor más bajo será cuando el sensor entregue 0 V y el más alto cuando entregue la tensión de alimentación Vcc; en este caso se alimentará a 3,3 V. Por tanto, teniendo en cuenta la sensibilidad del TMP36 (10 mV/ºC), para leer el valor en grados Celsius se aplicará a la lectura un factor de conversión compuesto por la división de los 1023, la multiplicación de los 3,3 V (3300 mV) y la división por la sensibilidad.

A continuación, se procede a actualizar la variable “temperarura” del tipo “CloudTemperatureSensor” de la biblioteca “thingProperties.h”. Por último, a través del monitor serie se imprime por pantalla la temperatura medida. Cabe mencionar que el valor no tiene decimales, puesto que no se está trabajando con un sensor de mucha precisión y tampoco es imprescindible disponer de mucha exactitud para la aplicación desarrollada (medir la temperatura de la casa).

```
154 void funcion_medir_temperatura() {
155
156   if ((millis() - prev_temperatura) >= periodo_muestreo_temperatura) {
157
158     int temp = (analogRead(Sensor_Temperatura)/1023.0)*3300/10;
159     temperatura = temp;
160     Serial.print(temp);
161     Serial.println(" C");
162
163     prev_temperatura = millis();
164
165   }
166 }
```

Figura 41: Función “funcion_medir_temperatura”

La última funcionalidad de Aurora es el sensor de luz, que se encarga de abrir la cortina permitiendo que entre la luz natural al amanecer. Para conseguir esto intervienen dos funciones. La primera de ellas, “onSensorLuzChange”, es la que se invoca cuando hay un cambio en la propiedad del sensor a través de Alexa. Como se puede ver en la figura 42, el deber de esta función es averiguar en qué rutina del sensor de luz se encuentra el dispositivo. En el caso de que esté encendido, se realiza un mapeo de los 100 valores que puede tener el parámetro de luminosidad y se traducen a los 3 valores posibles de rutinas activas del sensor. El resto del código muestra a través del monitor serie el proceso y la rutina en cuestión.

```
168 void onSensorLuzChange()  {
169
170   Serial.println("onSensorluzChange ...");
171
172   if(sensor_luz.getSwitch())
173   rutina_sensor_luz = map(sensor_luz.getBrightness(), 1, 100, 1, 3);
174   else
175   rutina_sensor_luz = 0;
176
177   Serial.print("Rutina sensor de luz: ");
178   Serial.println(rutina_sensor_luz);
179
180 }
181
```

Figura 42: Función “onSensorLuzChange”

La segunda función (véase figura 43) se encarga de controlar la cortina en el momento del amanecer. La primera mitad de la función equivale a la medición de la luz. Su funcionamiento en cuanto al periodo de muestreo es análogo al del sensor de temperatura, pero utilizando, evidentemente, el periodo y marca temporal del sensor de luz. Como se ha comentado en el apartado de electrónica, el montaje del sensor de luz ofrece tres pines digitales de información: en el programa “Luz1”, “Luz2” y “Luz3”. Estos pines se pondrán a nivel alto cuando detecten luz, cada uno con una sensibilidad distinta.

La mitad de la función restante se encarga de activar la luz natural, es decir, de llamar a la función que activa la rutina del motor cuando la rutina del amanecer esté a nivel alto. Para ello se configura el estado a de la luz natural a *true*, para activarla, y se utiliza la función “onLuzNaturalChange”, cuyo código se ha comentado anteriormente. Estas dos líneas de programa generan el mismo efecto que tendría pedir directamente a Alexa que active la luz natural. Por último, se desactiva el sensor de luz ya que ha terminado su función. Esto se programa de forma análoga a las dos líneas previas, pero configurando el estado del sensor a *false.*

```
182 void funcion_amanecer() {
183
184     if ((millis() - prev_luz) >= periodo_muestreo_luz) {
185
186     if(rutina_sensor_luz == 1)
187     rutina_amanecer = digitalRead(Luz1);
188
189     if(rutina_sensor_luz == 2)
190     rutina_amanecer = digitalRead(Luz2);
191
192     if(rutina_sensor_luz == 3)
193     rutina_amanecer = digitalRead(Luz3);
194
195     prev_luz = millis();
196
197    }
198
199    if(rutina_amanecer){
200    luz_natural.setSwitch(true);
201    onLuzNaturalChange();
202
203    sensor_luz.setSwitch(false);
204    onSensorLuzChange();
205    }
206
207 }
```

Figura 43: Función “funcion_amanecer”

Para subir el código a la placa hay que asegurarse de que el cable USB esté correctamente enchufado y en el puerto correspondiente. Seguidamente, se transfiere el programa pulsando el botón habilitado para ello en la zona superior del Arduino IDE. Para abrir el monitor serie y visualizar toda la información que se imprime, basta con acceder a la sección de herramientas y pulsar la opción homónima disponible en el menú.

5.2.3. Amazon Alexa

En este apartado se van a mostrar todos los pasos necesarios para configurar correctamente la adición de Aurora al control de Alexa.

En el desarrollo de este proyecto, se ha usado el Amazon Echo Dot de tercera generación para comunicarse con el asistente de voz de Alexa, pero se puede usar directamente desde la aplicación Amazon Alexa. Esta aplicación permite la descarga gratuita y es imprescindible para asociar cualquier dispositivo de domótica a Alexa, incluido el propio Echo Dot.

Una vez instalada la aplicación y creada una cuenta de usuario se procede con los pasos para asociar el automatismo para cortina. El primer paso es acceder al apartado de dispositivos del menú principal; seguidamente; se pulsa el símbolo de suma para añadir un dispositivo (paso 2). Una vez dentro, el tercer paso consiste en elegir el tipo de dispositivo y, como el del automatismo de cortina no está predefinido, se selecciona la última opción llamada “Otro”. Estos tres primeros pasos pueden visualizarse en la figura 44.

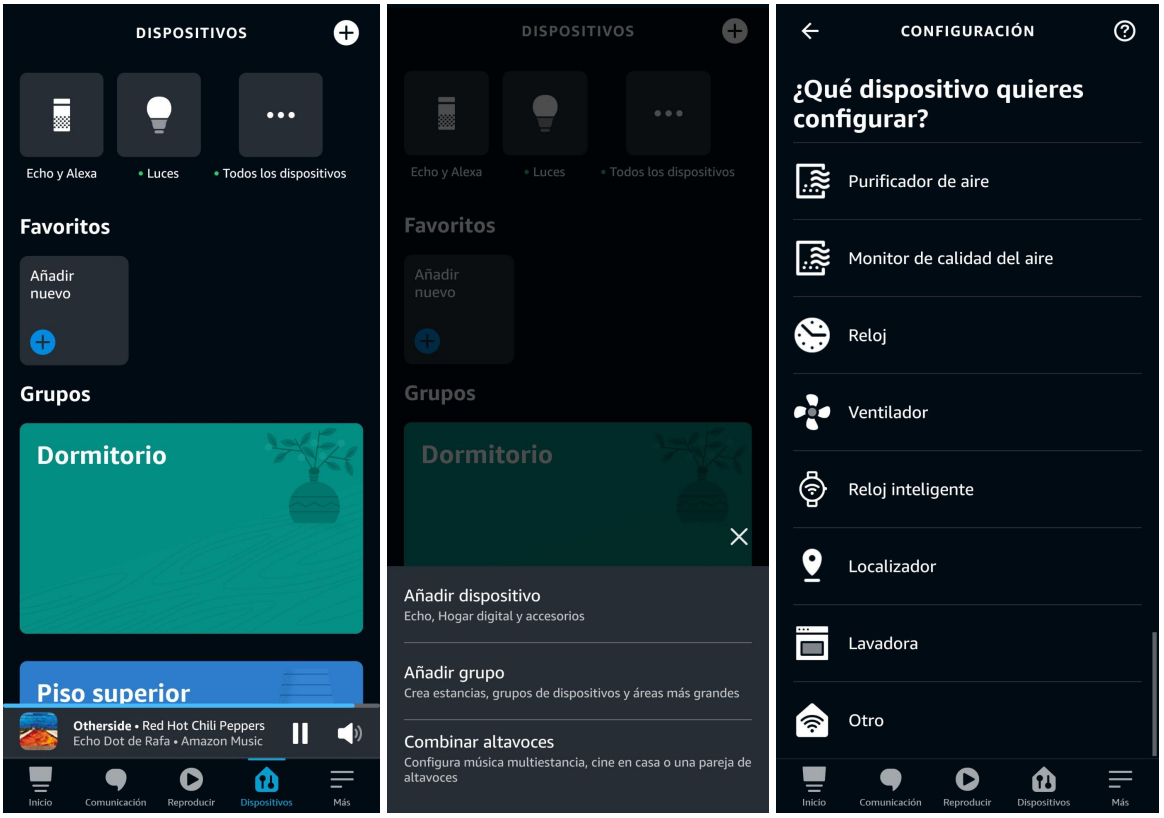


Figura 44: Pasos 1, 2 y 3 de la configuración en Amazon Alexa

Los siguientes tres pasos son: continuar con la detección, esperar uno o dos minutos y proceder con la configuración de los dispositivos. A continuación, en la figura 45 puede verse el transcurso de los pasos 4, 5 y 6.

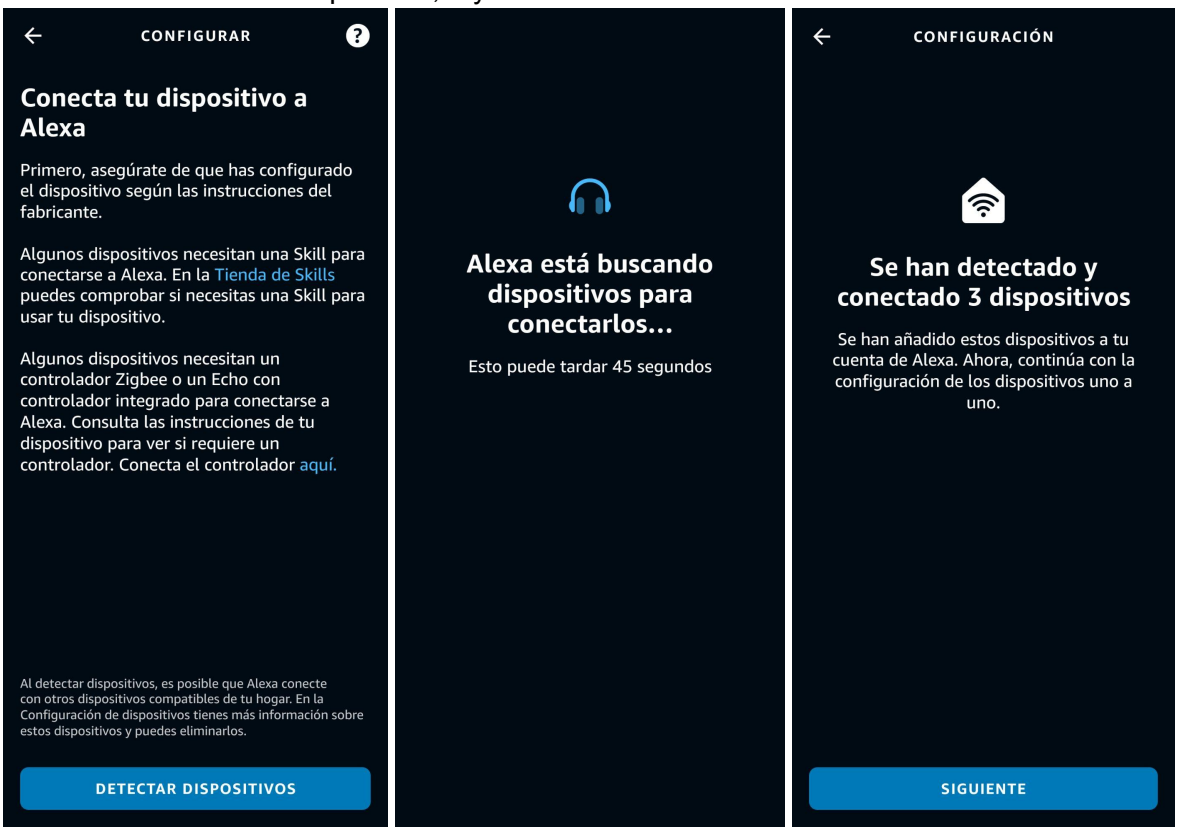


Figura 45: Pasos 4, 5 y 6 de la configuración en Amazon Alexa

Alexa interpreta que hay tres dispositivos en vez de uno solo. Estos dispositivos hacen referencia a las tres propiedades que tiene el automatismo para cortinas: la luz natural (cortina), el sensor de temperatura y el sensor de luz. Ello se debe a que, como se ha mencionado anteriormente, Alexa no conoce la estructura propia de Aurora, pero sí sabe cómo funcionan los tipos de variable que ha generado la nube de Arduino.

Por tanto, el siguiente paso es visualizar que, en efecto, se han detectado las variables del dispositivo: “luz_natural”, “temperatura” y “sensor_luz”. Los pasos 8 y 9 consisten en seleccionar cada parámetro y asociarlo, si se desea, a algún grupo de dispositivos o estancia (en este caso se omite esta opción). Es importante realizar los pasos 8 y 9 para cada una de las tres propiedades, como se ve en la figura 46.

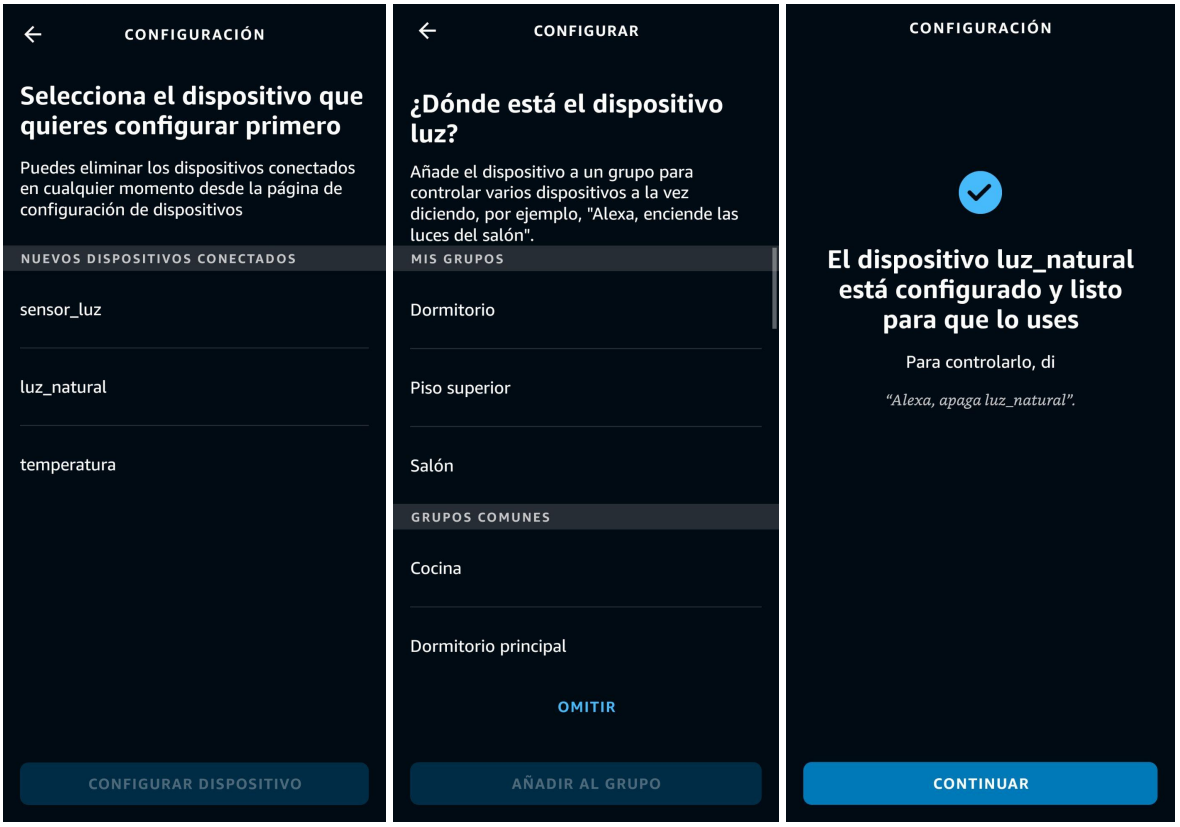


Figura 46: Pasos 7, 8 y 9 de la configuración en Amazon Alexa

En este punto ya se tienen asociadas todas las variables del automatismo para la cortina. En la figura 47 puede verse que Alexa interpreta que se han conectado tres nuevos dispositivos (paso 10). Por último, pueden personalizarse algunas opciones en función de las preferencias del usuario; en este caso, se han añadido las tres funcionalidades descritas al apartado de favoritos. Además, el usuario puede gestionar de diversas formas cualquiera de los parámetros: cambiar el nombre, asignar a una sala concreta, crear rutinas, activar las funciones de Aurora con frases personalizadas, etc. Las rutinas son comportamientos predefinidos que pueden asignarse a los dispositivos y rigen una conducta específica. Un ejemplo de rutina sería configurar que la cortina se cierre todos los días a las 10 de la noche y que se active el sensor de luz de manera automática a las 6 de la mañana.

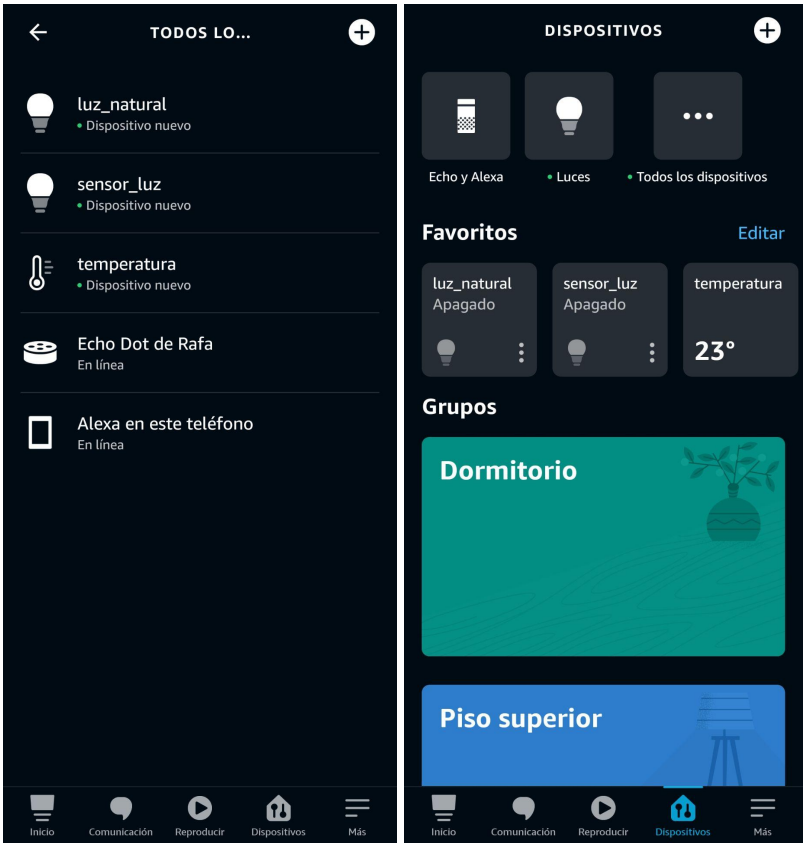


Figura 47: Paso 10 y preferencias de la configuración en Amazon Alexa

6.3. Subsistema mecánico

En este apartado se describen todas las partes que conforman el montaje de la estructura que alberga el circuito y que dota a Aurora de un sistema mecánico que interactúa con el entorno.

Al tratarse de un primer prototipo, las características físicas del dispositivo son limitadas, ya que el tiempo para la realización del proyecto se ha destinado principalmente a los anteriores subsistemas. Ello explica que el resultado final de este trabajo esté alejado de ser una versión para el uso en el día a día: parámetros físicos como el volumen o la fluidez de funcionamiento de Aurora deberían ser objeto de mejora en cualquier proceso de adaptación al mercado. Sin embargo, a modo de prueba de viabilidad, el prototipo alcanza los objetivos preestablecidos.

Por tanto, el método de diseño del subsistema mecánico que se ha seguido aquí es, en primer lugar, el modelado 3D de las diferentes piezas, para el que se ha empleado el *software* SolidWorks para el diseño asistido por ordenador o CAD. En segundo lugar, la impresión 3D de las piezas diseñadas, con el doble objetivo de utilizar a Aurora en el ámbito práctico y, tras numerosas pruebas, detectar fallos de diseño. En tercer lugar, la corrección de los errores mecánicos del prototipo hasta alcanzar el funcionamiento adecuado para Aurora, que es el que se ha descrito en este trabajo. Por último, la debida retroalimentación de los diseños 3D de las piezas enmendadas.

Los planos de despiece y de ensamblaje, donde se acotan de forma detallada las piezas diseñadas y se representa el montaje físico entre los diferentes componentes, van en el segundo documento del proyecto tras la memoria. En los planos se excluyen las medidas

del logo de Aurora, ya que su objetivo es meramente estético y no es vinculante para la realización del prototipo.

6.3.1. Agarre superior

Aurora se engancha a la barra de la cortina, de forma que cuelgue el dispositivo, de un modo análogo a como se colocaría una percha. Para conseguir esto será necesario una estructura que mantenga el equilibrio, que se acomode a la vara y que albergue un par de rodamientos para facilitar el desplazamiento. Esta estructura está compuesta de dos piezas que toman el nombre de pinzas A y B, las cuales se ensamblan a través de tres tornillos. Al unirse forman un par de enganches para la tela de la cortina; de esta forma Aurora es capaz de abrir y cerrar la cortina aun en caso de no poder ser colocada entre dos anillas. Estas piezas también incorporan una varilla cada una para encajarse en el resto del cuerpo e impedir su libre rotación.

A través del conjunto de piezas descrito anteriormente circula un perno M10 de 120 mm de longitud y cabeza hexagonal. Este componente normalizado tiene el papel de “columna vertebral” de Aurora. En otras palabras, se encarga de unir el agarre superior con la carcasa y permite, a través de su rotación, ajustarse a diferentes diámetros de barra. Es importante remarcar que el orificio del agarre por donde circula el perno no es roscado, ya que se busca facilitar su subida y bajada. Con el objetivo de ayudar a girar el perno, se diseña una pequeña llave que se acopla al mismo.

Por último, cabe comentar la colocación, alrededor del perno, de un muelle a modo de suspensión con el objetivo de adaptarse a barras que presentan tramos de diferentes diámetros (barras extensibles, por ejemplo). No obstante, tras una serie de pruebas, el funcionamiento de la suspensión no consigue adaptarse a este tipo de variaciones. Esto es debido a que, cuando el agarre se ajusta a la vara de la cortina, se inclina de forma asimétrica y, por tanto, no se aprovecha del todo la función del resorte. De todas formas, se ha decidido mantener el muelle, añadiendo una arandela para evitar dañar la unión del agarre superior. Por las razones expuestas, la suspensión de Aurora queda como elemento susceptible de mejoras. El resultado del diseño de este apartado puede verse en el renderizado generado por SolidWorks de la figura 48.

Figura 48: Agarre superior

6.3.2. Rueda

El componente de la rueda está formado por dos piezas: el disco y el neumático (figura 49). Los criterios para diseñar el disco se basan en el equilibrio de robustez y ahorro de material. Por tanto, el resultado es una pieza con tres nervios radiales que son suficientes para resistir el movimiento de Aurora.

Respecto al neumático, se trata de una pieza singular ya que, a diferencia de las demás, su fabricación es por molde. El molde se imprime en 3D con el material PLA y dentro de este se introduce silicona líquida para que tome la forma indicada. Este material garantiza que la rueda no se deslice y que Aurora se acomode mejor a la barra. El uso del molde queda justificado por el hecho de que las impresoras suelen dar problemas con la impresión directa de piezas elásticas. Además, tras una serie de pruebas, los hilos que suelen tener las piezas impresas en 3D en la textura de sus superficies reducen drásticamente el rozamiento y esto hace que la rueda resbale.

El criterio para diseñar el dibujo del neumático está en función de la adherencia: se trata de conseguir un patrón que funcione correctamente, es decir, que tenga un buen agarre con el medio por el que se desplaza. Por tanto, para encontrar un relieve adecuado se ha realizado una búsqueda de las mejores ruedas de Lego y, tras la comparación de los resultados [21], se ha llegado a la conclusión de que los modelos 32020c01, 6595c01 y 32077c01 presentan mejor rendimiento, con un coeficiente de fricción de 1,38. Por ser el más plano, el primer modelo resulta idóneo para Aurora; esto puede facilitar el agarre a una mayor diversidad de barras, ya que la mayoría suelen ser cilíndricas. Así pues, el diseño del neumático de este proyecto está basado en el patrón de hexágonos propio de la rueda 32020c01 de Lego (figura 49).

Figura 49: Rueda

6.3.3. Carcasa

Este apartado es el más extenso del subsistema mecánico ya que hay numerosos detalles que cubrir. La carcasa está formada por el armazón principal y la tapa que cierra el encapsulado de los componentes electrónicos. Además, en la unión de estas dos piezas se acopla un conducto, en la parte inferior de Aurora, que alberga el LDR. Esta pieza está habilitada para poder rotar y medir el nivel de luz tanto al derecho como al revés,

dependiendo de cómo se instale Aurora: una vez colocada, el usuario debe orientar el conducto hacia el exterior.

En cuanto al armazón principal, este dispone de una serie de detalles que se comentarán a continuación. Empezando de arriba a abajo, en primer lugar, el armazón tiene un par de orificios en la parte superior para que circulen las varillas del agarre.

En segundo lugar, se encuentra el agujero por donde se enrosca el perno visto en el agarre superior. Como las roscas tienen demasiado nivel de detalle, una impresora FDM no es capaz de imprimirlas correctamente. Para solucionar esto, se imprime el hueco cilíndrico liso y luego se utiliza un macho de roscar para generar la espiral.

En tercer lugar, dentro del dispositivo se disponen tres agujeros para atornillar el motor. En cambio, el controlador y la placa nodeMCU no van atornilladas, ya que no precisan de un agarre robusto, además de que su disposición dificulta el acceso de un destornillador. Para fijarlas, se han diseñado unos salientes en forma de cilindros, cuyas terminaciones tienen un diámetro ligeramente menor que sus bases (conos cortados). De esta manera, como los componentes electrónicos mencionados disponen de un orificio en cada esquina, se pueden acoplar rápidamente y sin herramientas.

En cuarto lugar, el armazón cuenta con una abertura comunicada con el interior, justo en el lugar del puerto USB de la nodeMCU, para su fácil acceso.

Por último, cabe comentar que la carcasa se encaja fácilmente gracias a un canal situado en los bordes de contacto entre el armazón y la tapa. En la figura 50, se puede ver cómo queda el armazón principal y, en la figura 51, se observan todos los componentes electrónicos dentro de este.

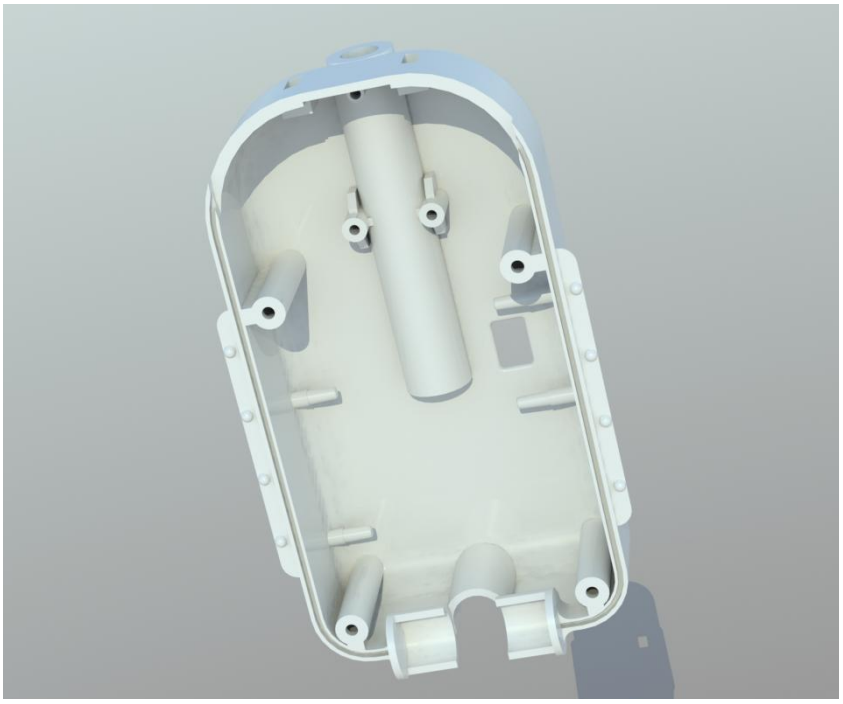


Figura 50: Armazón principal

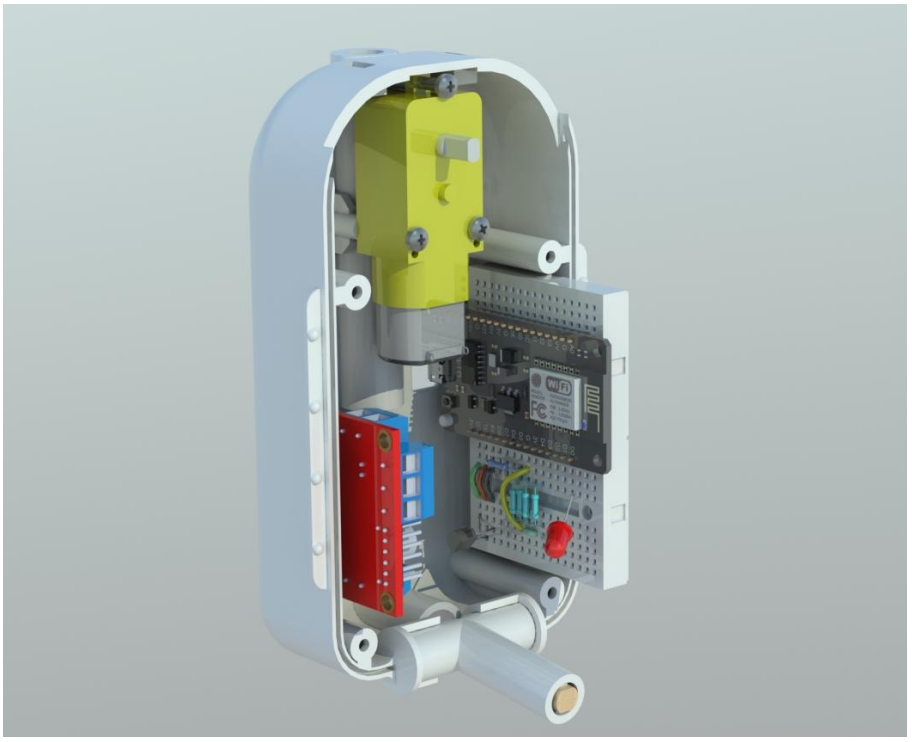


Figura 51: Armazón principal junto con el conducto y la electrónica

Uno de los últimos detalles del subsistema mecánico es la tapa. Esta tiene una forma análoga a la del armazón (véase figura 52). Al lado izquierdo hay una inscripción, “Luz Natural”, que señaliza por qué lado se va a abrir la cortina, ya que el motor, *a priori*, rota en sentido antihorario cuando se le pide a Alexa que active la luz natural. Esta pieza dispone de un orificio para introducir el interruptor y otro para el fácil acceso al cargador de la batería (figura 53).

Figura 52: Tapa

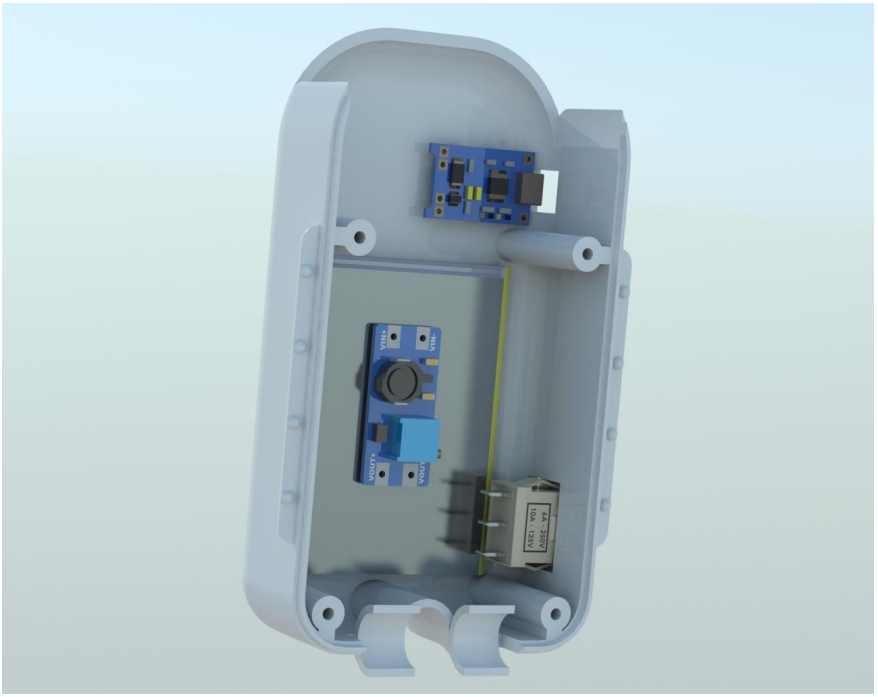


Figura 53: Tapa junto con la electrónica

Finalmente, la carcasa se sella a través de cuatro tornillos y, como es evidente, en la parte superior existe un hueco hacia el exterior, que es por donde gira la rueda. Una vez descritos todos los componentes del subsistema mecánico, el resultado del montaje puede visualizarse en la figura 54, que representa el aspecto final de Aurora.

Figura 54: Dispositivo Aurora ensamblado

7. Conclusiones

En el presente trabajo se ha diseñado la primera versión del dispositivo domótico Aurora. A modo de cierre, se exponen una serie de conclusiones sobre la evolución de su diseño, de manera que puede afirmarse, en resumen, que se ha alcanzado el objetivo general (OG), ya que se ha desarrollado un prototipo comunicado con Alexa para el control de cortinas.

En cuanto a los objetivos específicos (OE), en primer lugar, se ha montado un circuito capaz de conectarse a internet de forma independiente para poder trabajar directamente con el entorno IoT como se ha visto en el apartado del subsistema electrónico y en el de programación. Es decir, se ha logrado el OE1.

En segundo lugar, respondiendo al OE2, se ha conseguido que Aurora tenga la opción de reaccionar de forma sensible a la luz del amanecer.

En tercer lugar, se ha implementado un sensor de temperatura para su telemetría y, de esta forma, se cumple con el OE3.

En cuanto al OE4, Aurora se ha programado en el entorno de Arduino, además del uso de librerías propias de este *software* libre, como se ha descrito en el capítulo del subsistema de programación.

Por último, también el OE5 ha quedado cumplido, ya que se ha diseñado una estructura mecánica que permite a Aurora abrir y cerrar diferentes tipos de cortinas, instalándose de forma flexible, rápida e intuitiva para cualquier usuario, sin tener que sustituir a la cortina original, como se ha descrito con detalle en el apartado del subsistema mecánico.

Además de los objetivos intrínsecos del proyecto, su desarrollo ha servido también para alcanzar diversos propósitos de aprendizaje, todos valiosos como culmen del grado de Ingeniería Electrónica Industrial y Automática: desde la experiencia de trabajar con diferentes componentes electrónicos y el refuerzo en habilidades de programación hasta el cultivo de conocimientos relacionados con el diseño asistido por ordenador. Todas las facetas de esta toma de contacto real con el desarrollo de un producto ha conseguido aportar, sin duda, una visión cercana y práctica de cómo proyectar un invento en ingeniería.

Dada la actual extensión del proyecto y el tiempo con el que se ha contado para su desarrollo, no se han podido explotar algunas funciones adicionales. Por un lado, para alimentar la electrónica se ha dispuesto un sistema basado en una pequeña batería recargable que tiene una independencia muy reducida. Esta implementación ha sido suficiente para trabajar con Aurora y ponerla a prueba, pero no es el sistema ideal si se busca que el dispositivo domótico haga su función durante meses o años sin la necesidad de recargar continuamente.

Por otro lado, existe la limitación de tener que acceder al código para introducir el nombre y contraseña de la red Wi-Fi. A modo de comprobar la viabilidad del prototipo, ha sido la manera más directa, pero no resulta la más adecuada para un nivel de usuario.

Además, en cuanto al subsistema mecánico, ha surgido la problemática de la suspensión. Esto se traduce en que Aurora, si bien puede instalarse sobre un amplio rango de diámetros de barras, si se trata de una cortina que presenta tramos de diferentes tamaños, el dispositivo no se podrá mover correctamente por todos ellos.

Por consiguiente, una vez expuestas las limitaciones más relevantes del proyecto, queda abierta la posibilidad de futuras líneas de trabajo. La primera es que una próxima versión de

Aurora podría estar alimentada por una batería de mayor capacidad, junto con una placa solar que dote al dispositivo de una mayor independencia.

La segunda propuesta conveniente para seguir mejorando el prototipo sería apostar por una versión de Aurora que implemente un módulo bluetooth y que, a través de una *app,* pueda ser conectada a la red Wi-Fi por cualquier usuario, de una manera más accesible.

Para finalizar, como se ha sugerido en algunos puntos de la memoria, este trabajo puede llegar a ser un provechoso arranque para la creación de una futura versión comercial y, además, un punto de partida para otras aplicaciones en el amplio mundo de la domótica.

8. Bibliografía

[1] Avilés Pozo, A. (mayo de 2022). *Molinos de viento: así nacieron y así sobreviven los gigantes del Quijote*. elDiario, [en línea]. Disponible en: <https://www.eldiario.es/castilla-la-mancha/cultura/molinos-viento-nacieron-sobreviven-gigantes-quijote_1_9022169.html>

[2] Brunete, A., San Segundo, P., y Herrero R. (julio de 2020). *Breve historia de la automática*.Introducción a la automatización industrial, [en línea]. Disponible en: <https://bookdown.org/alberto_brunete/intro_automatica/breve-historia-de-la-automatica.html#edadcontemporanea>

[3] Torres, L. (junio de 2022). *La tecnología en los hogares: casi la mitad de los españoles cuenta con domótica en su vivienda*. elEconomista, [en línea]. Disponible en: <https://www.eleconomista.es/vivienda/noticias/11504194/12/21/La-tecnologia-en-los-hogares-casi-la-mitad-de-los-espanoles-cuenta-con-domotica-en-su-vivienda.html>

[4] Robot Conga. Sitio web de Cecotec (mayo de 2022), [en línea]. Disponible en: <https://www.storececotec.com/es/29-robots-aspiradores?gclid=CjwKCAjw7vuUBhBUEiwAEdu2pDA-gnWJCSCEOEUlbygNVnn0nhvycxchw91Z-L_pgZDO23ao6rpsChoCMekQAvD_BwE>

[5] Perez Fernandez, D. (enero de 2019). *Amazon vendió más de 100 millones de dispositivos Alexa*. Tecnonucleous, [en línea]. Disponible en: <https://tecnonucleous.com/2019/01/06/amazon-vendio-mas-de-100-millones-de/#:~:text=En%20la%20actulidad%20existen%20m%C3%A1s,usuarios%20con%20sus%20respec tivas%20IA.>

[6] Sin especificar. (enero de 2019). *¿Qué es el Internet de las cosas (IoT)?*. Red Hat, [en línea]. Disponible en: <https://www.redhat.com/es/topics/internet-of-things/what-is-iot>

[7] Sin especificar. (2017). *Internet de las cosas (IoT) ¿En qué consiste y cómo funciona?*. Canal de Youtube de InnovaSecure, [en línea]. Disponible en: <https://www.youtube.com/watch?v=gV7l2YOSOQ4&t=251s>

[8] Producto SwitchBot curtain. Sitio web de SwitchBot (febrero de 2022), [en línea]. Disponible en: <https://www.switch-bot.com/pages/switchbot-curtain>

[9] Producto SwitchBot curtain rod 2. Sitio web de SwitchBot (junio de 2022), [en línea]. Disponible en: <https://www.switch-bot.com/products/switchbot-curtain-rod2>

[10] Cortina de Tuya. Amazon (mayo de 2022), [en línea]. Disponible en: <https://www.amazon.es/Montloxs-Interruptor-Temporizador-Configuraci%C3%B3n-Compatible/dp/B09FLTN6P2/ref=sr_1_5?crid=37ICKKLAF3KW0&keywords=tuya%2Bcurtain&qid=1654536683&sprefix=tuya%2Bcu%2Caps%2C85&sr=8-5&th=1>

[11] Nodemcu Lua Lolin V3 Module ESP8266 ESP-12F WIFI. Az-Delivery (diciembre de 2021), [en línea]. Disponible en: <https://www.az-delivery.de/es/products/nodemcu-lolin-v3-modul-mit-esp8266?pr_prod_strat=copurchase&pr_rec_id=8d595004a&pr_rec_pid=9755979986&pr_ref_pid=771844898912&pr_seq=uniform>

[12] Controlador de motores L298N. Amazon, [en línea]. Disponible en: <https://www.amazon.es/DollaTek-Controlador-m%C3%B3dulo-Puente-Arduino/dp/B07DK6Q8F9/ref=sr_1_4?keywords=l298n&qid=1655570200&s=electronics&sprefix=l2%2Celectronics%2C87&sr=1-4>

[13] Motor DC 3-6 V. Amazon, [en línea]. Disponible en: <https://www.amazon.com/-/es/CHANCS-DC-Motor-engranajes-el%C3%A9ctrico/dp/B088YWSHZ9>

[14] Sensor de temperatura TMP36. Amazon, [en línea]. Disponible en: <https://www.amazon.com/-/es/Bridgold-TMP36GT9Z-TMP36-anal%C3%B3gicos-temperatura/dp/B07LG758H3/ref=sr_1_1?__mk_es_US=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2IV6W2NLM1IZJ&keywords=tmp35&qid=1655571217&sprefix=tmp35%2Caps%2C170&sr=8-1>

[15] Sensor de luz LDR GL5516. Amazon (mayo de 2022), [en línea]. Disponible en: <https://www.amazon.com/Gikfun-Photoresistor-GL5516-LDR-Resistencias/dp/B00RLGFIEY/ref=sr_1_1_sspa?__mk_es_US=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=1CQLDS5SVWGK&keywords=ldr+gl5516&qid=1655571666&sprefix=ldr+gl5516%2Caps%2C176&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEyQ05ST1JPNVkxU1AmZW5jcnlwdGVkSWQ9QTA0MzUyNDgyU0FaTlg5OENPVlQ1JmVuY3J5cHRlZEFkSWQ9QTA4Nzk4MDEyTDFZNFZHT0tDMk45JndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ>

[16] Protoboard 170 puntos. Amazon (mayo de 2022), [en línea]. Disponible en: <https://www.amazon.com/CenryKay-tablero-peque%C3%B1o-soldadura-unidades/dp/B07KG8VW21/ref=sr_1_1_sspa?__mk_es_US=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=X57K3X2CD2D7&keywords=protoboard+170&qid=165557191

0&sprefix=protoboard+170%2Caps%2C153&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEyVktOTzRNWUo5M1hUJmVuY3J5cHRlZElkPUEwOTIxNzg3MjdKVU5BMzIzOVZSMSZlbmNyeXB0ZWRBZElkPUEwNTk0NTU3NVo3TEJER1FDUlJGJndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ⇒

[17] Batería 2600 mAh. Amazon (junio de 2022), [en línea]. Disponible en: <https://www.amazon.es/2600mah-505573-sustituci%C3%B3n-bater%C3%ADas-pol%C3%ADmero/dp/B091YMWC72/ref=sr_1_34?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2WK2WZI5BGJA4&keywords=bateria+litio+2500&qid=1655635980&sprefix=bateria+litio+2500%2Caps%2C82&sr=8-34>

[18] Convertidor DC-DC MT3608. Amazon (junio de 2022), [en línea]. Disponible en: <https://www.amazon.es/ARCELI-Ajustable-Switching-Convertidor-alimentaci%C3%B3n/dp/B07MY3NZ18/ref=sr_1_1?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=VF3YGTYOGEN6&keywords=mt3608&qid=1655627827&sprefix=mt3608%2Caps%2C182&sr=8-1>

[19] Cargador de baterías de iones de litio TP4056. Amazon (junio de 2022), [en línea]. Disponible en: <https://www.amazon.es/AZDelivery-%E2%AD%90%E2%AD%90%E2%AD%90%E2%AD%90%E2%AD%90-TP4056-Regulador-bater%C3%ADa/dp/B07D2G345P/ref=sr_1_1_sspa?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=3AW95NQ7E9IMN&keywords=TP4056&qid=1655628301&sprefix=tp4056%2Caps%2C80&sr=8-1-spons&smid=A1X7QLRQH87QA3&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUExTTFQNVU1ME1YTzRBJmVuY3J5cHRlZElkPUEwNDYzNjA4MzREN1JYQk5KNTFMSSZlbmNyeXB0ZWRBZElkPUEwNTM5MjI2MTE3QllaVlc3SlAyRiZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU&th=1>

[20] Interruptor 3 posiciones. Amazon (junio de 2022), [en línea]. Disponible en: <https://www.amazon.es/SODIAL-Interruptor-Basculante-6-Terminales-Posicion/dp/B00HPKBZPM/ref=sr_1_29?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=2ZMMET83H8E6F&keywords=interruptor+triestado&qid=1655628457&sprefix=interruptor+triestado%2Caps%2C79&sr=8-29>

[21] Sin especificar. (abril 2021). *33 Lego Wheels/Tracks on Wood (worst to best)*. Canal de Youtube de Brick Experiment, [en línea]. Disponible en: <https://www.youtube.com/watch?v=cDpcdVf39kA>

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

AURORA: DISEÑO Y PROGRAMACIÓN DE UN AUTOMATISMO PARA CORTINAS COMUNICADO CON ALEXA

2. PLANOS

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

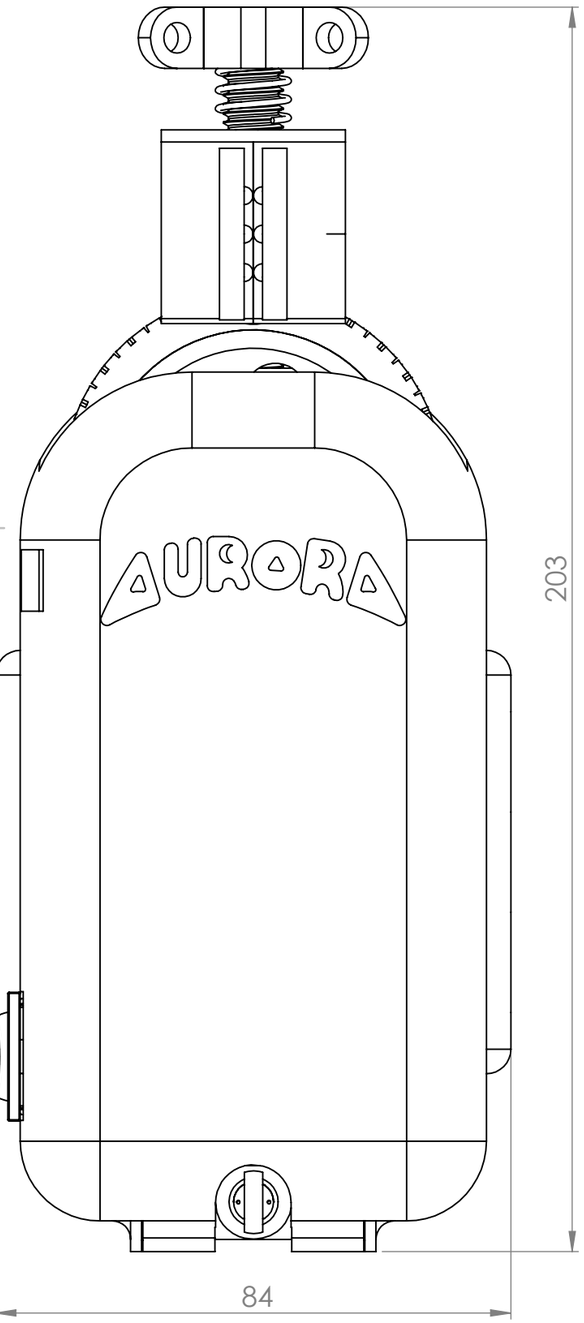
Rafael Luis Saneleuterio Temporal

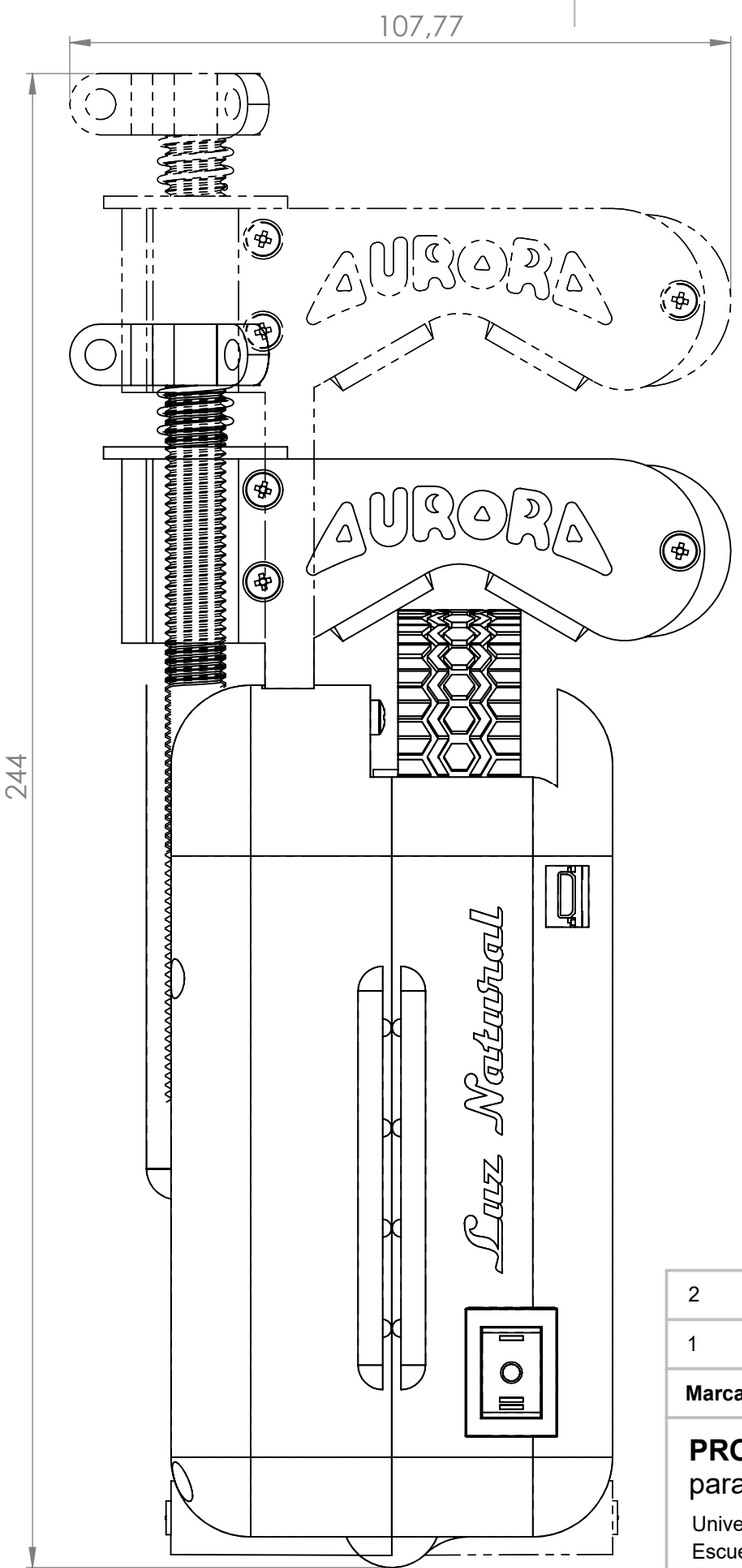
TUTORIZADO POR

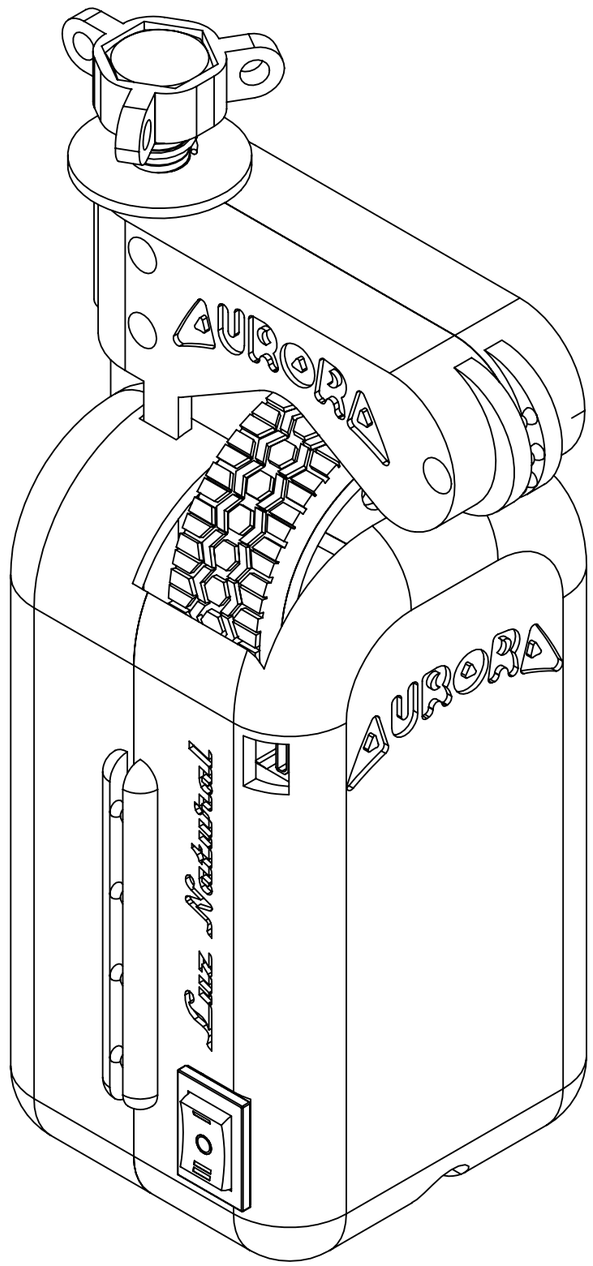
Patricia Balbastre Betoret

CURSO ACADÉMICO: 2021/2022

Índice planos







2	Conjunto carcasa: Plano Nº 8	1	Según esp. 84x156x76 mm
1	Conjunto agarre superior: Plano Nº 2	1	Según esp. 30x127x107,77 mm
Marca	**Descripción**	**Cantidad**	**Material y dimensiones**

PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

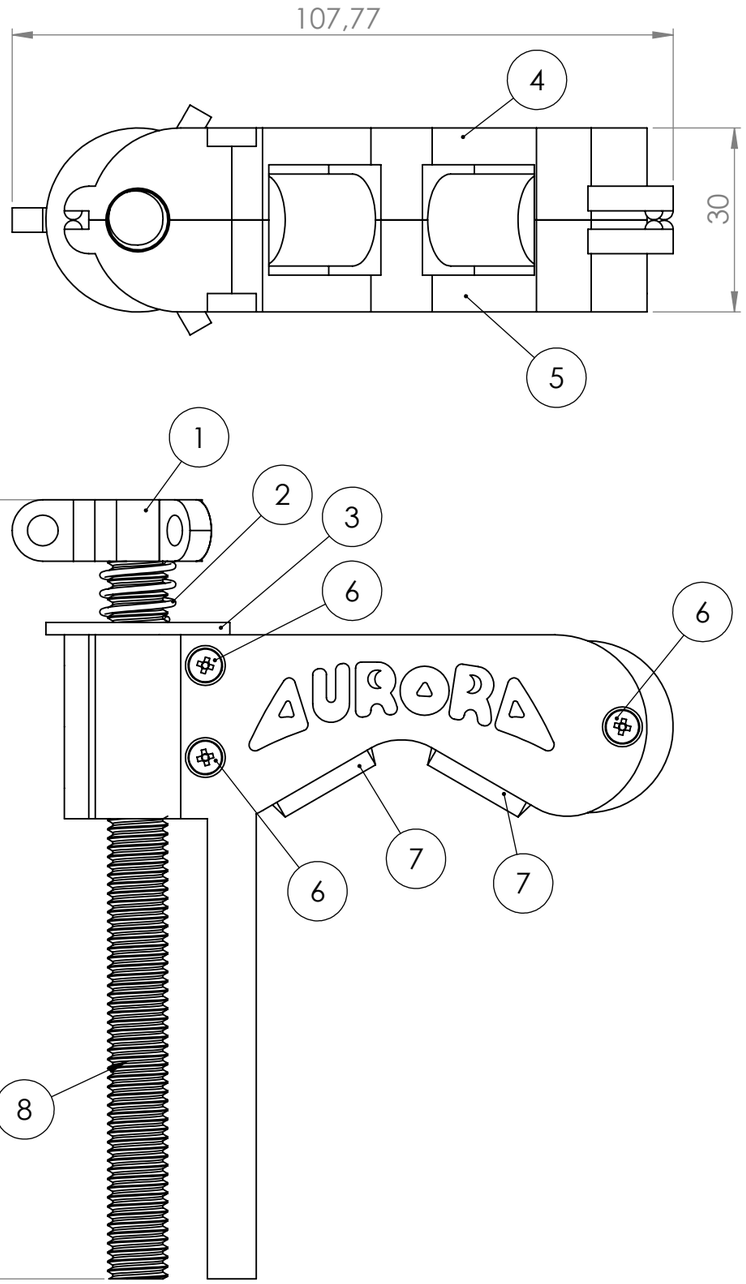
Fecha: 27/5/2022

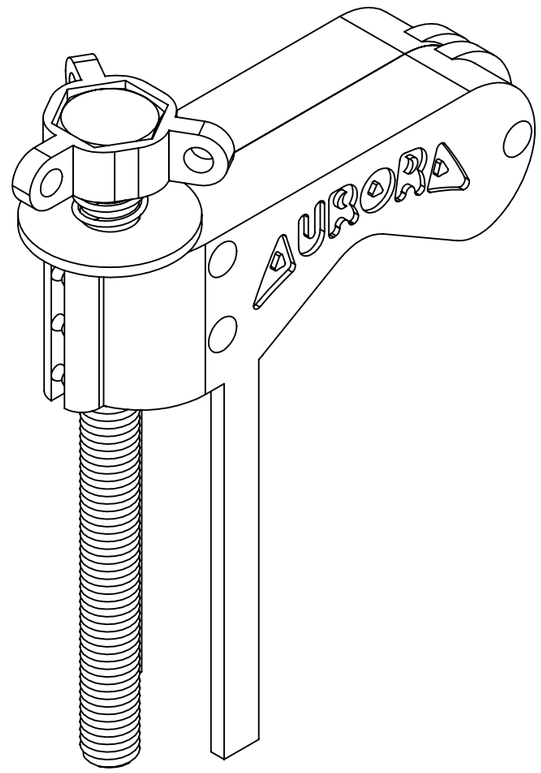
Escala **1:1**

Autor:
Rafael Luis
Saneleuterio Temporal

Plano:
Conjunto Aurora

Plano Nº: **01**





Marca	Descripción	Cantidad	Material y dimensiones
8	Perno de cabeza hexagonal	1	Acero ISO 4017 M10x120 DIN 933
7	Rodamiento: Plano Nº 7	2	PLA ∅15x28 mm
6	Tornillo de cabeza ranurada	3	Acero ISO 4017 M3x10 DIN 933
5	Pinza A: Plano Nº 5	1	PLA 19x105x99,27 mm
4	Pinza B: Plano Nº 6	1	PLA 15x105x99,27 mm
3	Arandela	1	Acero ISO 4017 M10 DIN 9021
2	Muelle	1	Acero 302 Di:10 d:1 L0:10 mm
1	Llave: Plano Nº 4	1	PLA 20,5x10x20,5 mm

PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

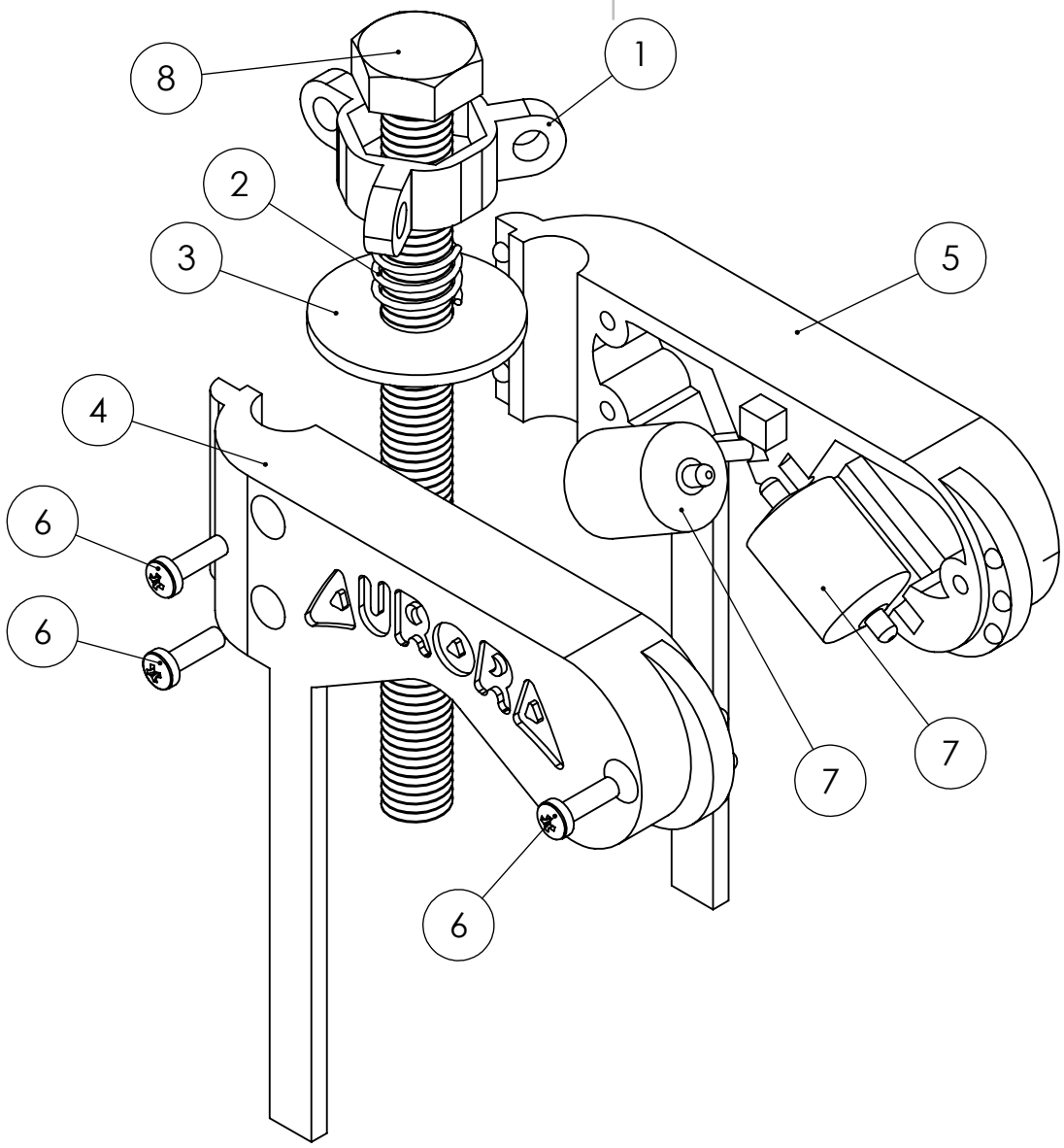
Fecha: 27/5/2022

Escala **1:1**

Autor:
Rafael Luis
Saneleuterio Temporal

Plano:
Conjunto agarre superior

Plano Nº: **02**



Marca	Descripción	Cantidad	Material y dimensiones
8	Perno de cabeza hexagonal	1	Acero ISO 4017 M10x120 DIN 933
7	Rodamiento: Plano Nº 7	2	PLA ∅15x28 mm
6	Tornillo de cabeza ranurada	3	Acero ISO 4017 M3x10 DIN 933
5	Pinza A: Plano Nº 5	1	PLA 19x105x99,27 mm
4	Pinza B: Plano Nº 6	1	PLA 15x105x99,27 mm
3	Arandela	1	Acero ISO 4017 M10 DIN 9021
2	Muelle	1	Acero 302 Di:10 d:1 L0:10 mm
1	Llave: Plano Nº 4	1	PLA 20,5x10x20,5 mm

PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

Fecha: 27/5/2022

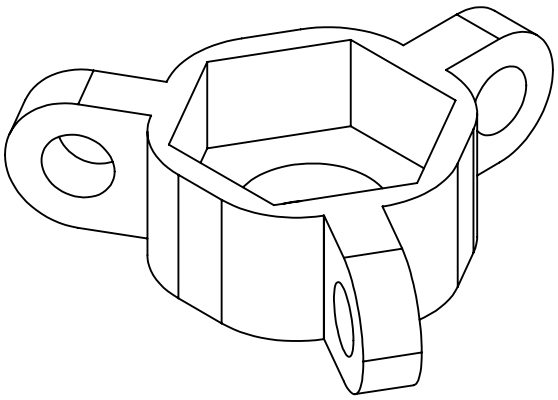
Escala **1:1**

Autor:
Rafael Luis
Saneleuterio Temporal

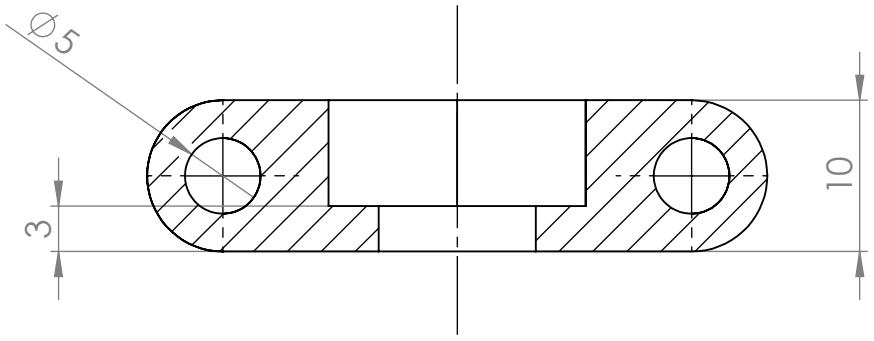
Plano:

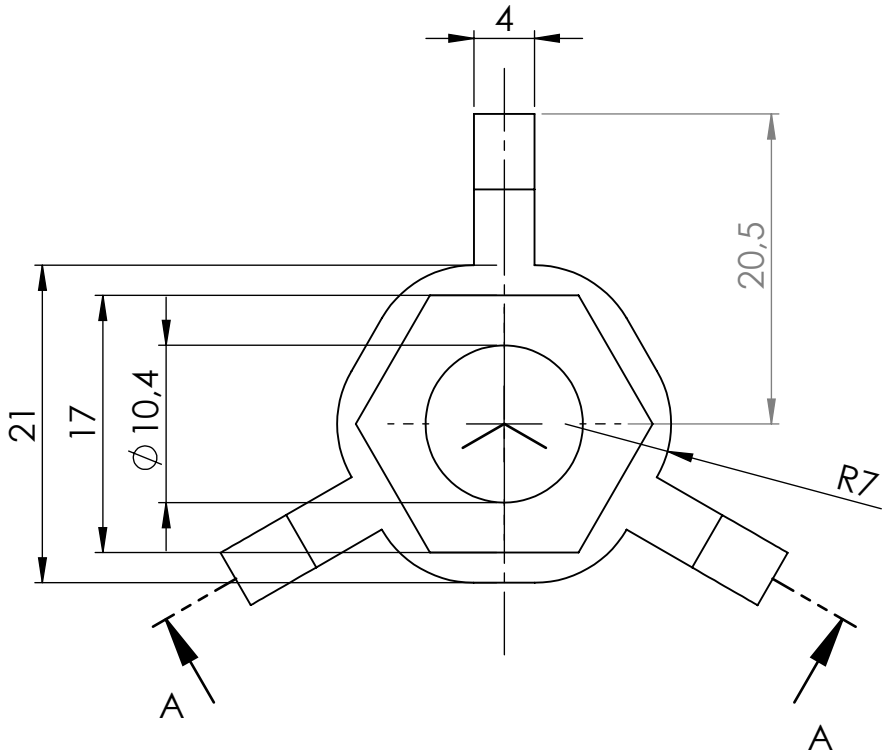
Montaje agarre superior

Plano Nº: **03**



SECCIÓN A-A





PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

Fecha: 27/5/2022

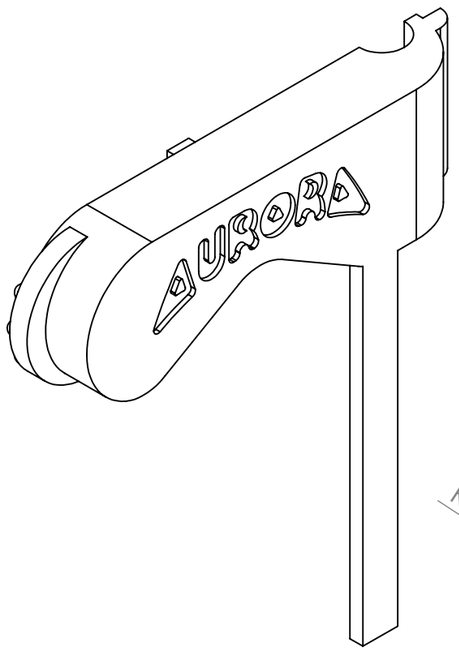
Escala **2:1**

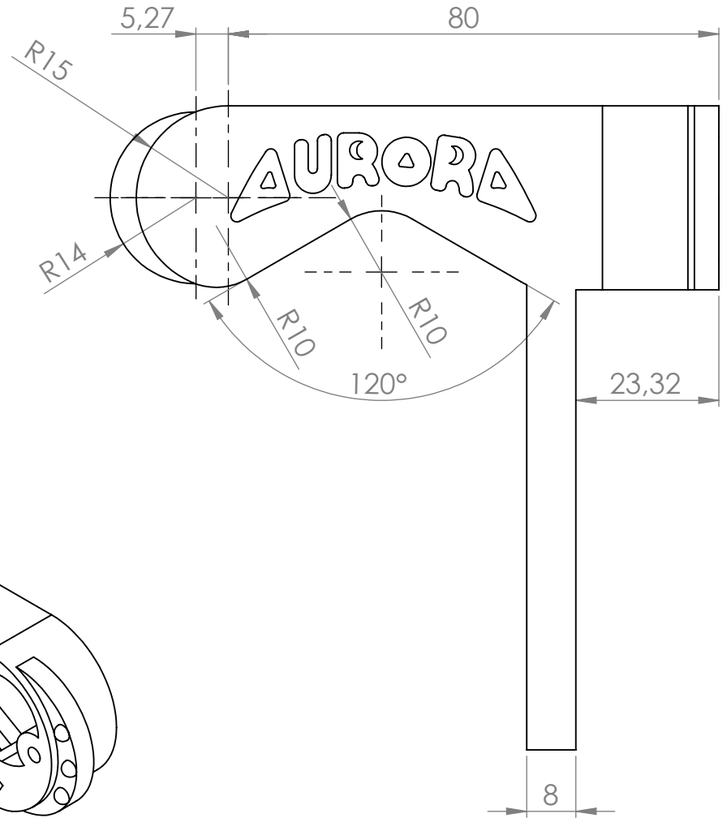
Autor:
Rafael Luis
Saneleuterio Temporal

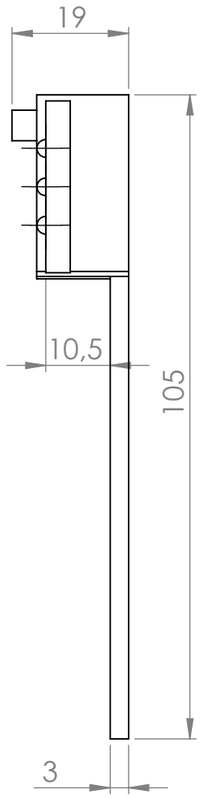
Plano:

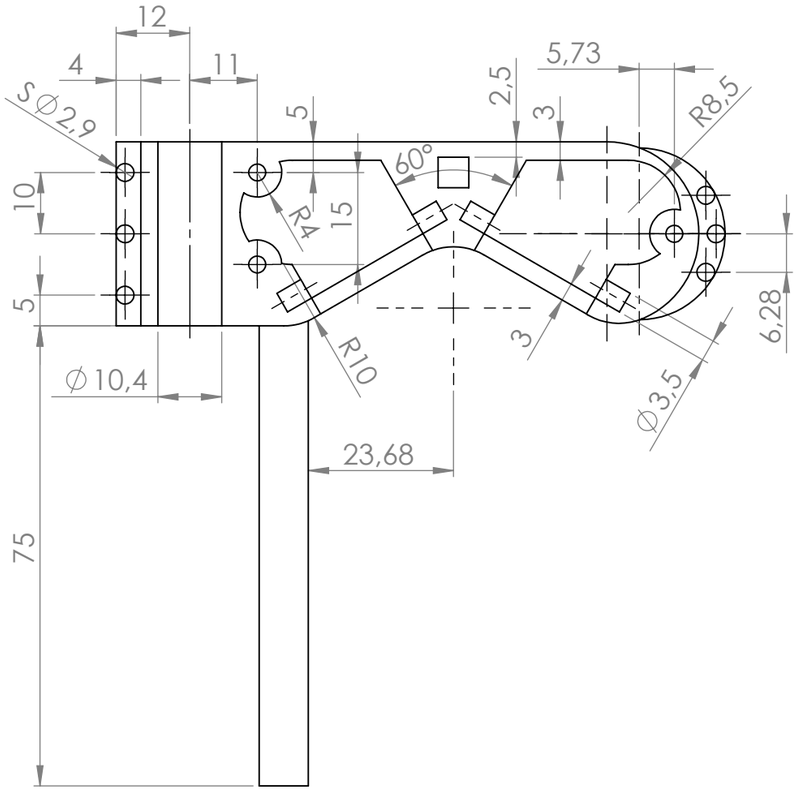
Llave

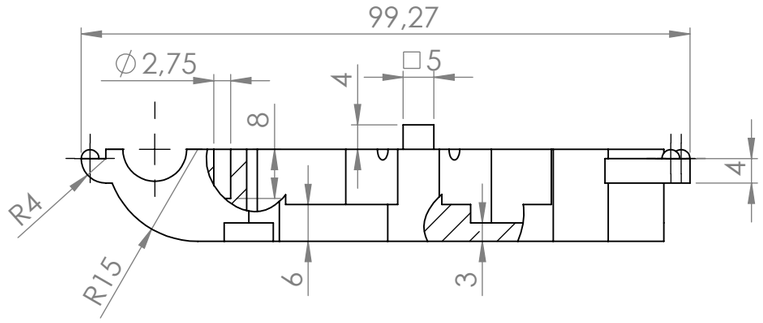
Plano Nº: **04**

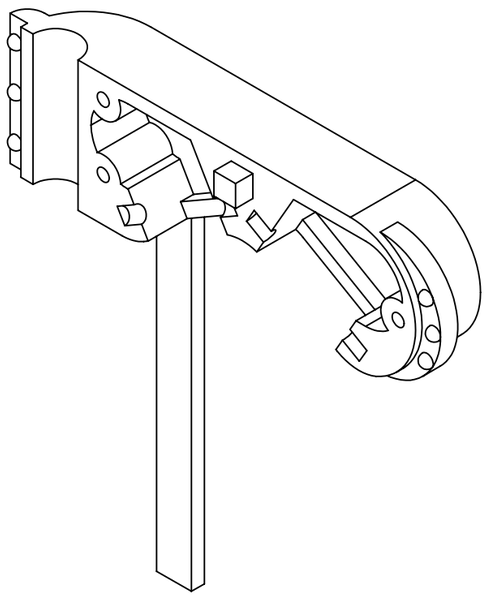




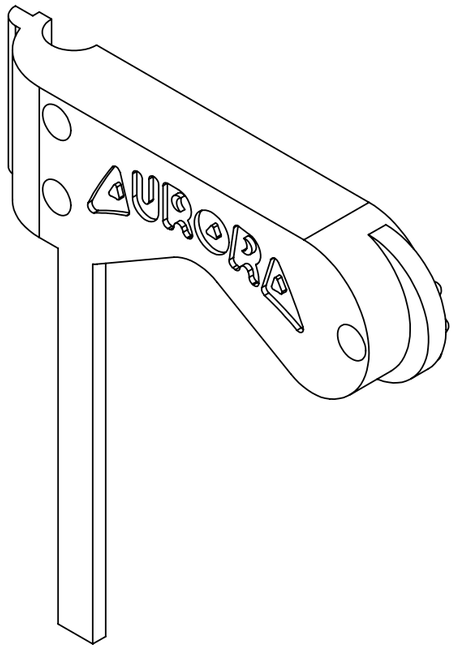


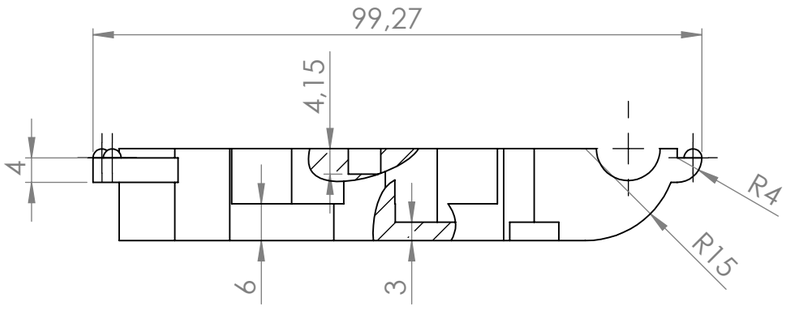


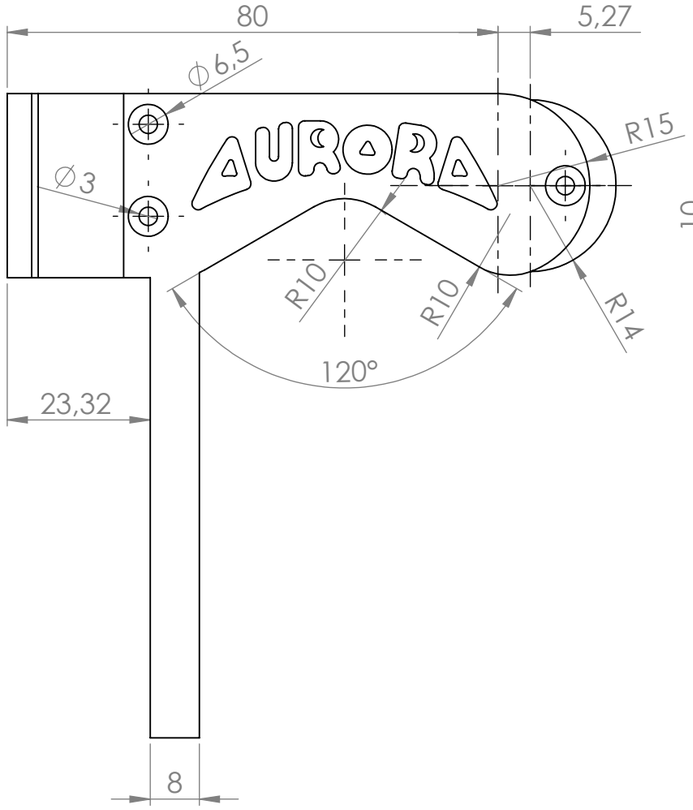


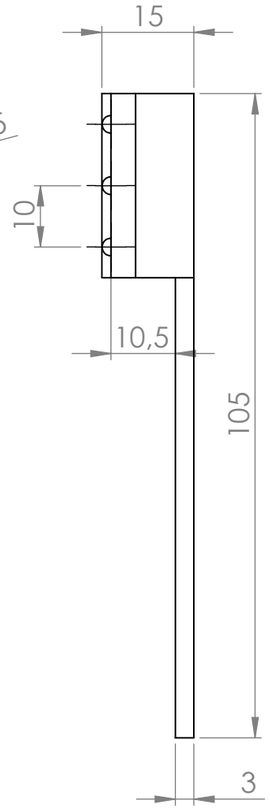


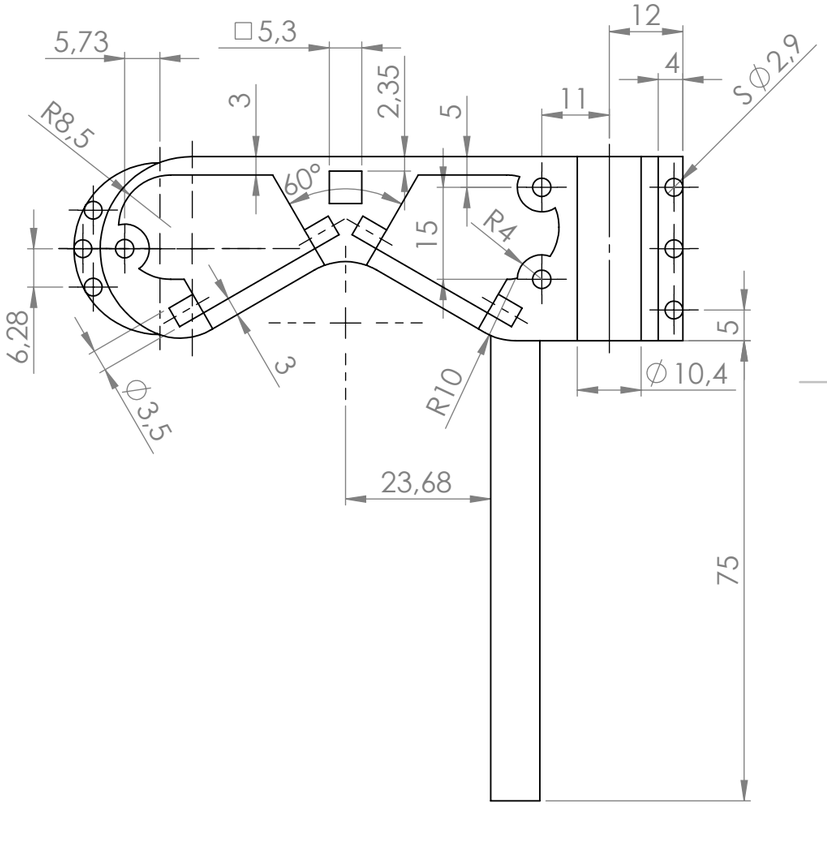
PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa Universitat Politècnica de València (UPV) Escuela Técnica Superior de Ingeniería del Diseño (ETSID)		Fecha: 27/5/2022 Escala **1:1**
Autor: Rafael Luis Saneleuterio Temporal	Plano: **Pinza A**	Plano Nº: **05**

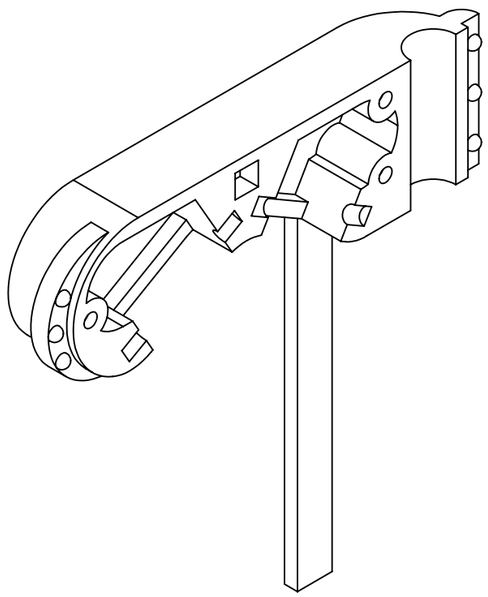




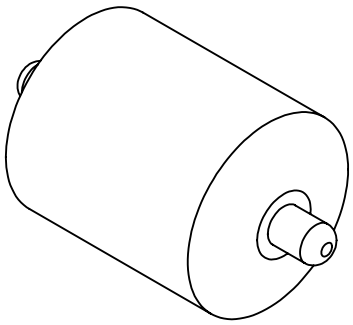


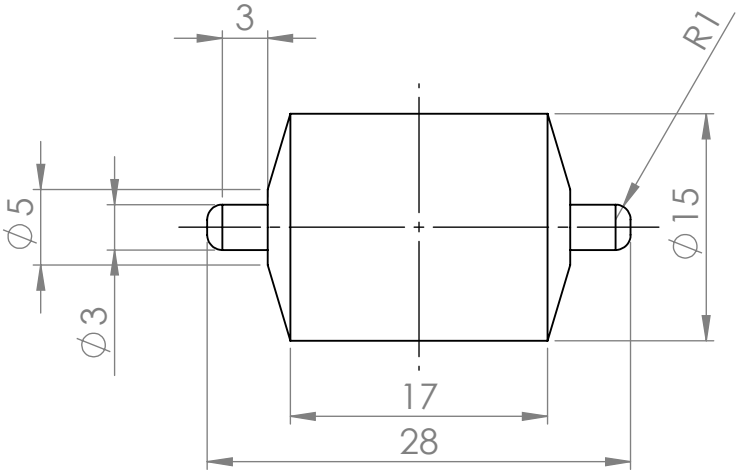






PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa Universitat Politècnica de València (UPV) Escuela Técnica Superior de Ingeniería del Diseño (ETSID)		Fecha: 27/5/2022 Escala **1:1**
Autor: Rafael Luis Saneleuterio Temporal	Plano: **Pinza B**	Plano Nº: **06**





PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

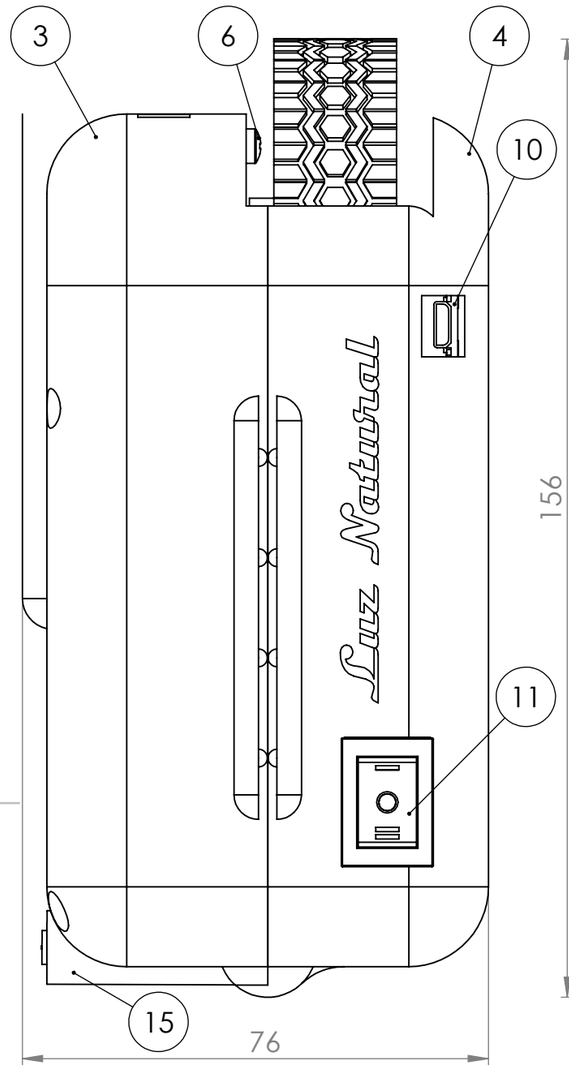
Fecha: 27/5/2022

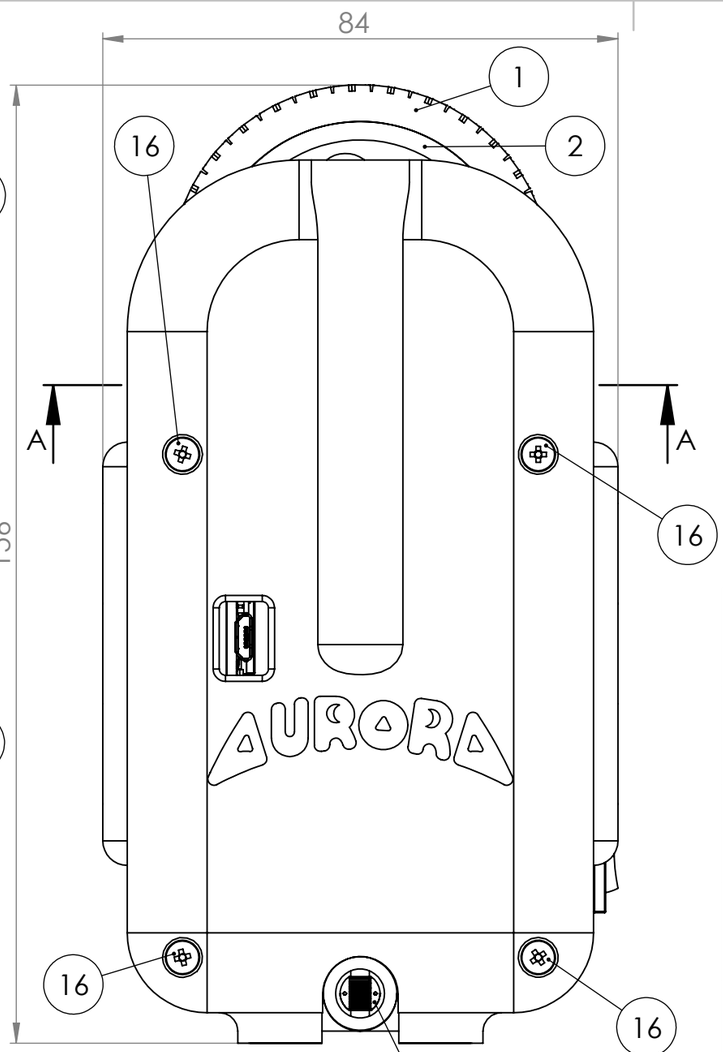
Escala
2:1

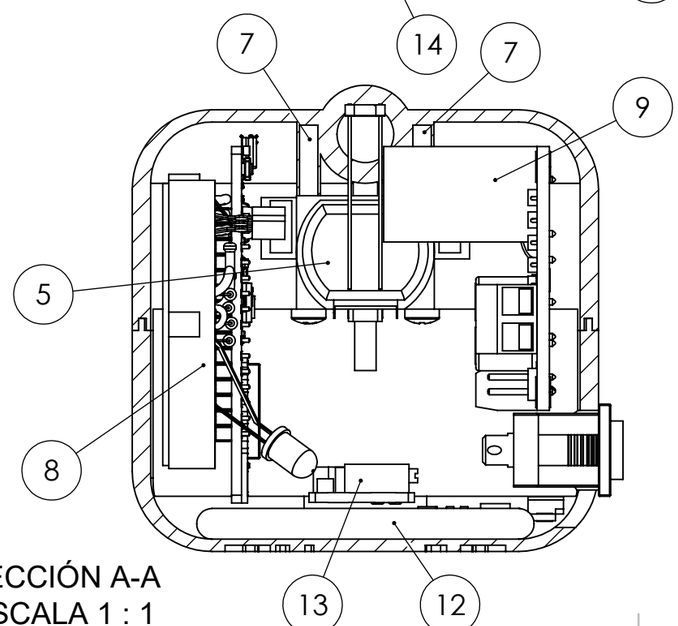
Autor:
Rafael Luis
Saneleuterio Temporal

Plano:
Rodamiento

Plano Nº:
07







SECCIÓN A-A
ESCALA 1 : 1

*Las interconexiones eléctricas quedan especificadas en el pliego de condiciones

16	Tornillo de cabeza ranurada	4	Acero ISO 4017 M3x10 DIN 933
15	Conducto: Plano Nº 14	1	PLA 36x12x42 mm
14	LDR GL5516	1	Según ficha técnica
13	Convertidor MT3608	1	Según ficha técnica
12	Batería iones de litio	1	Según ficha técnica
11	Interruptor triestado	1	Según ficha técnica
10	Cargador TP4056	1	Según ficha técnica
9	Controlador L298N	1	Según ficha técnica
8	Placa de prototipos 170 puntos	2	Según fichas técnicas
7	Tornillo de cabeza ranurada	2	Acero ISO 4017 M3x30 DIN 933
6	Tornillo de cabeza ranurada	1	Acero ISO 4017 M3x20 DIN 933
5	Motor DC 3-6 V	1	PLA 19x105x99,27 mm
4	Tapa: Plano Nº 13	1	PLA 76x143,37x38 mm
3	Armazón principal: Plano Nº 12	1	PLA 84x144x44 mm
2	Disco: Plano Nº 11	1	PLA ∅50x24 mm
1	Neumático: Plano Nº 10	1	Silicona A40 ∅62x20 mm
Marca	**Descripción**	**Cantidad**	**Material y dimensiones**

PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

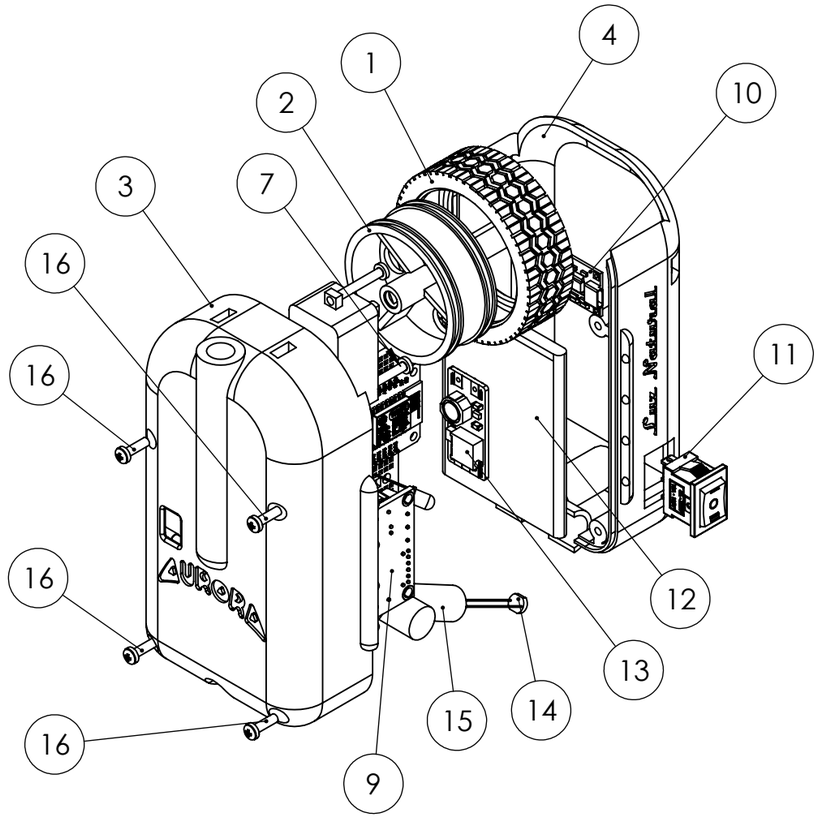
Fecha: 27/5/2022

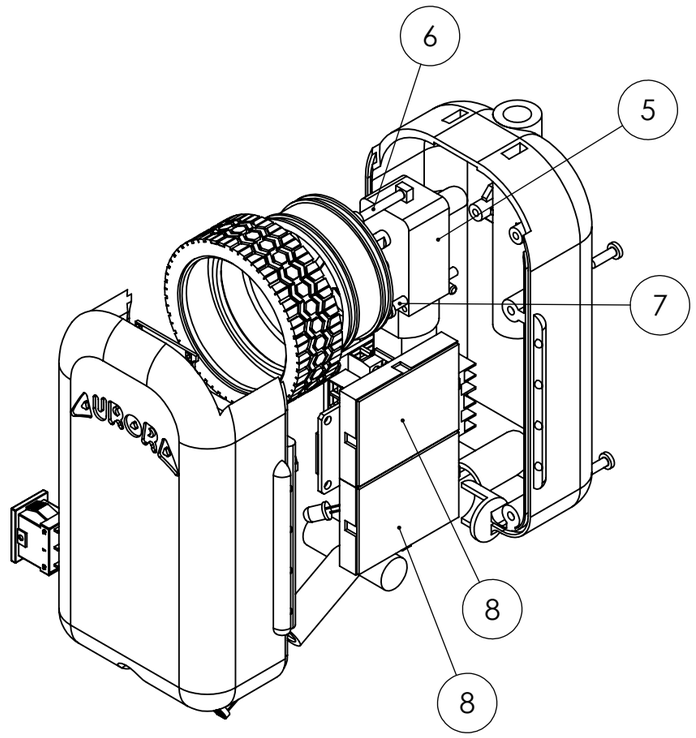
Escala **1:1**

Autor:
Rafael Luis
Saneleuterio Temporal

Plano:
Conjunto carcasa

Plano Nº:
08





*Las interconexiones eléctricas quedan especificadas en el pliego de condiciones

16	Tornillo de cabeza ranurada	4	Acero ISO 4017 M3x10 DIN 933
15	Conducto: Plano Nº 14	1	PLA 36x12x42 mm
14	LDR GL5516	1	Según ficha técnica
13	Convertidor MT3608	1	Según ficha técnica
12	Batería iones de litio	1	Según ficha técnica
11	Interruptor triestado	1	Según ficha técnica
10	Cargador TP4056	1	Según ficha técnica
9	Controlador L298N	1	Según ficha técnica
8	Placa de prototipos 170 puntos	2	Según fichas técnicas
7	Tornillo de cabeza ranurada	2	Acero ISO 4017 M3x30 DIN 933
6	Tornillo de cabeza ranurada	1	Acero ISO 4017 M3x20 DIN 933
5	Motor DC 3-6 V	1	PLA 19x105x99,27 mm
4	Tapa: Plano Nº 13	1	PLA 76x143,37x38 mm
3	Armazón principal: Plano Nº 12	1	PLA 84x144x44 mm
2	Disco: Plano Nº 11	1	PLA ∅50x24 mm
1	Neumático: Plano Nº 10	1	Silicona A40 ∅62x20 mm
Marca	**Descripción**	**Cantidad**	**Material y dimensiones**

PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

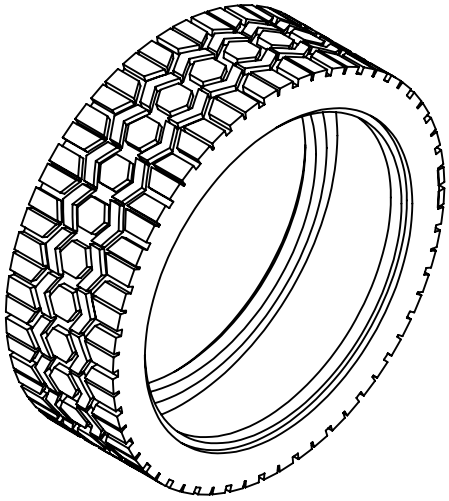
Fecha: 27/5/2022

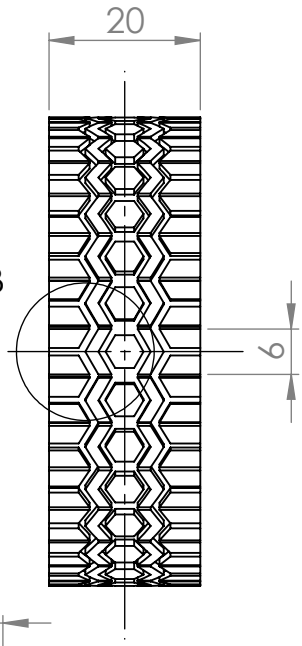
Escala **1:2**

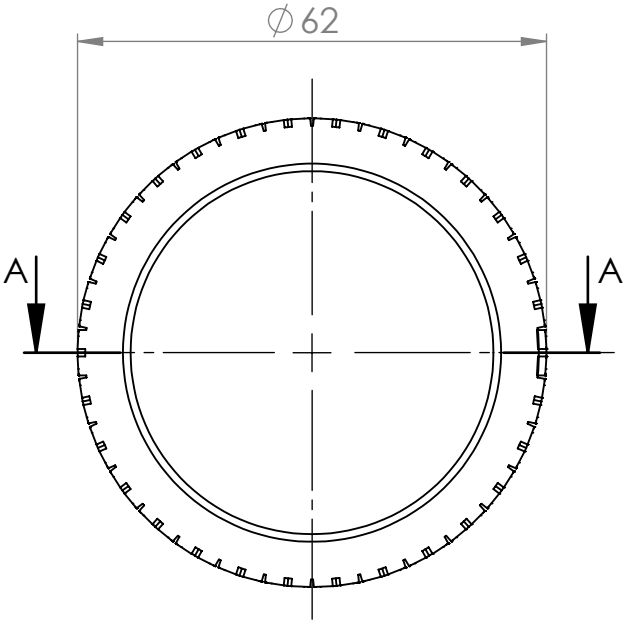
Autor:
Rafael Luis
Saneleuterio Temporal

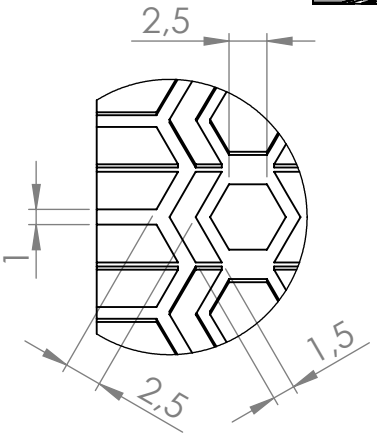
Plano:
Montaje carcasa

Plano Nº: **09**

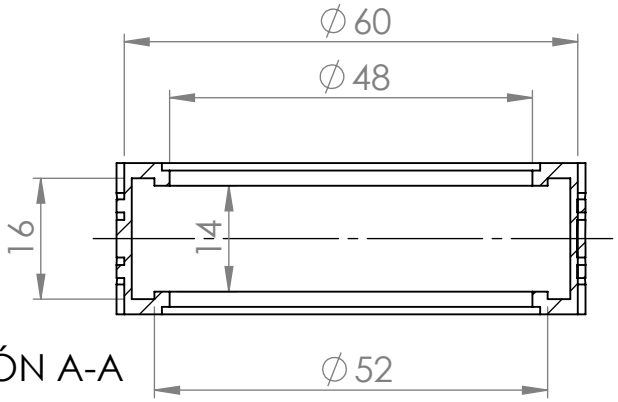








DETALLE B
ESCALA 2 : 1



SECCIÓN A-A

PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

Fecha: 27/5/2022

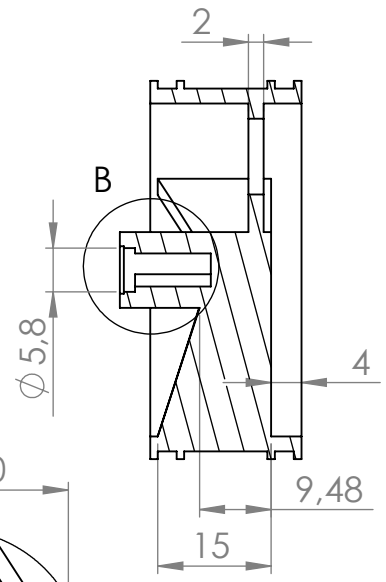
Escala
1:1

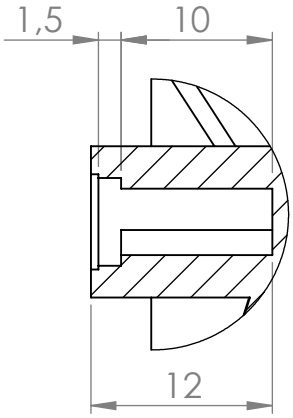
Autor:
Rafael Luis
Saneleuterio Temporal

Plano:
Neumático

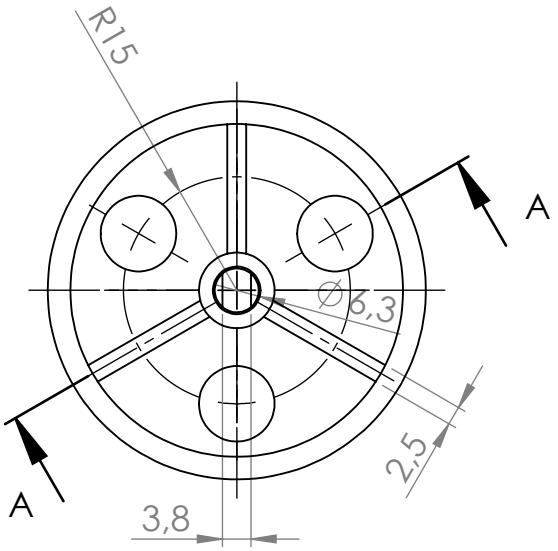
Plano Nº:
10

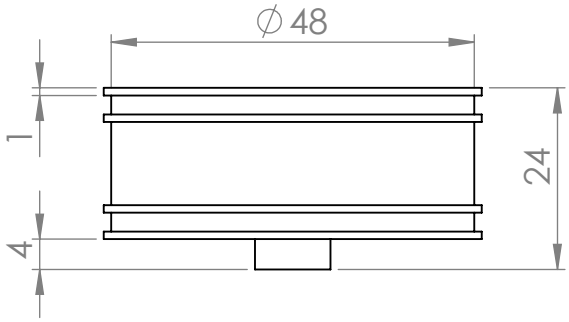
SECCIÓN A-A

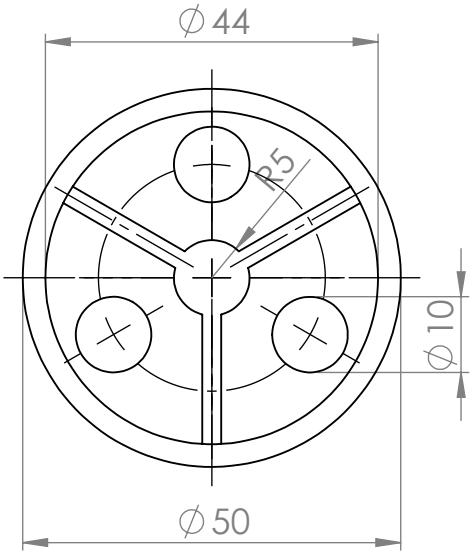


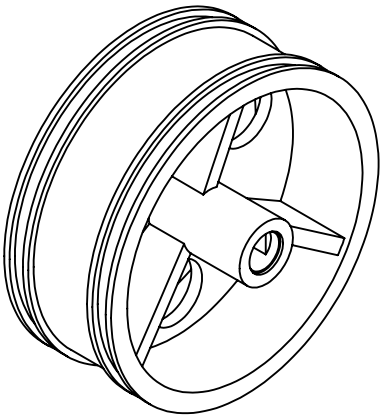


DETALLE B
ESCALA 2 : 1









PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

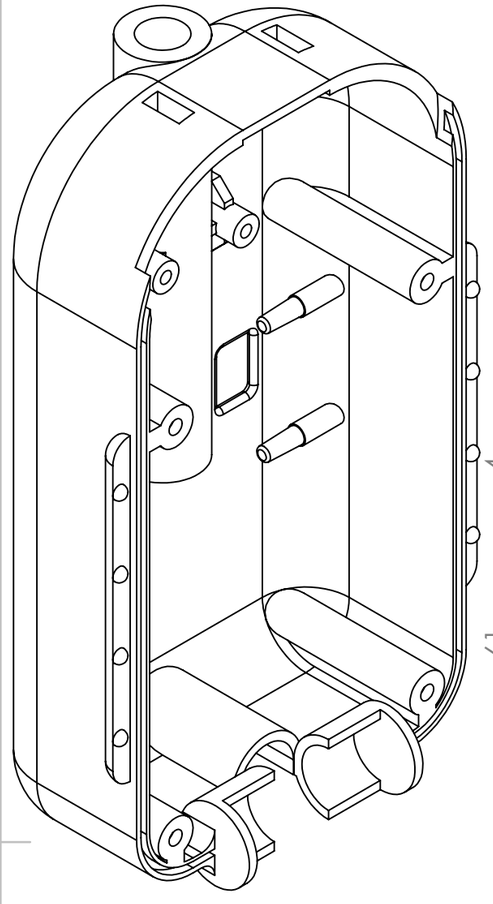
Fecha: 27/5/2022

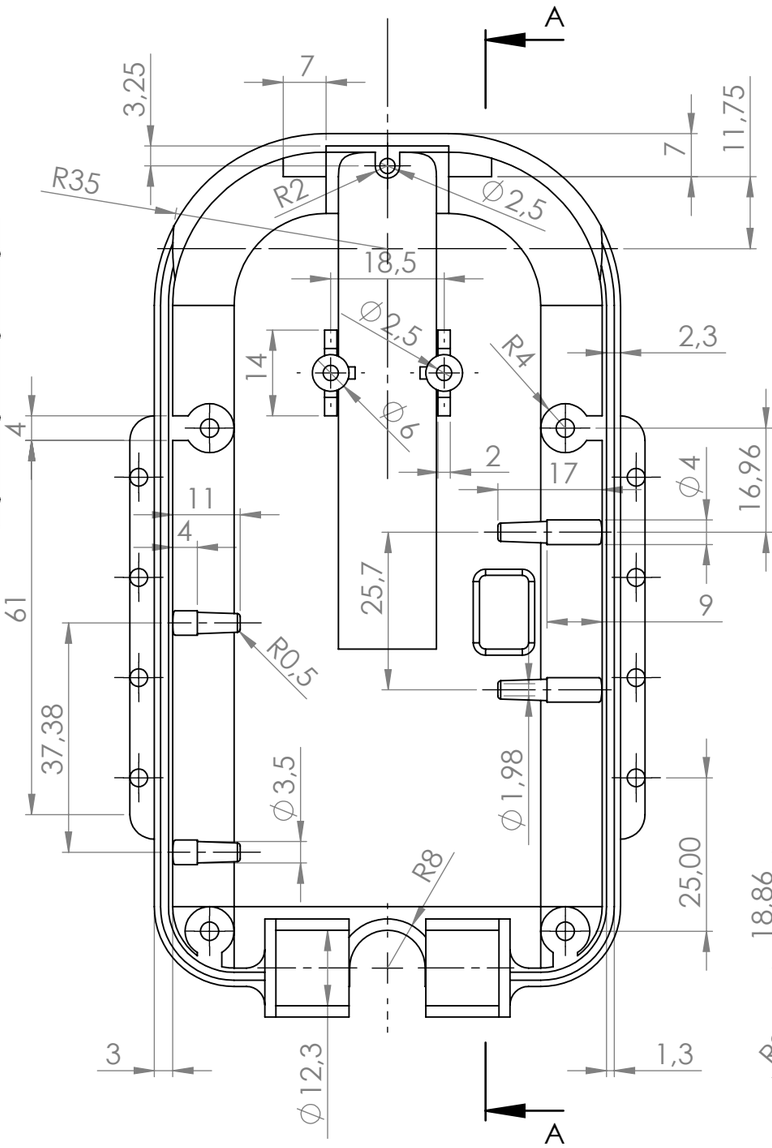
Escala
1:1

Autor:
Rafael Luis
Saneleuterio Temporal

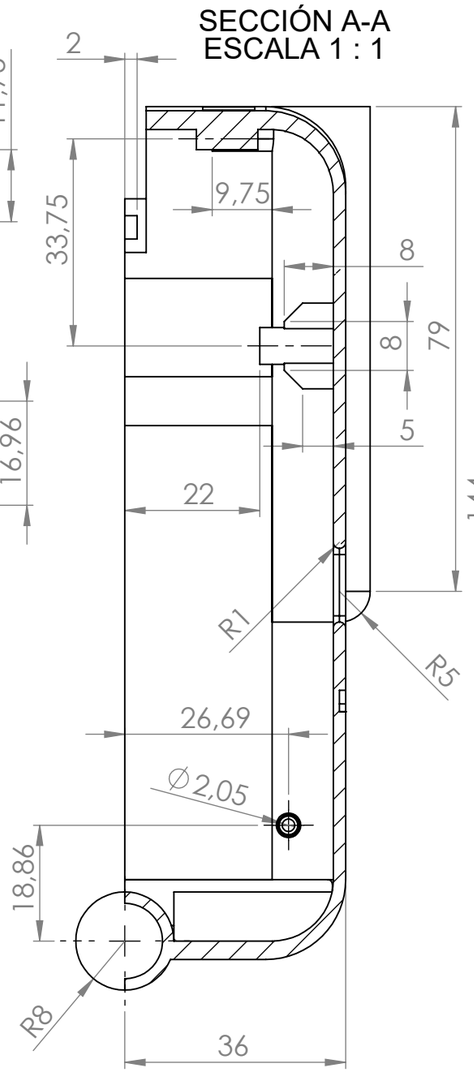
Plano:
Disco

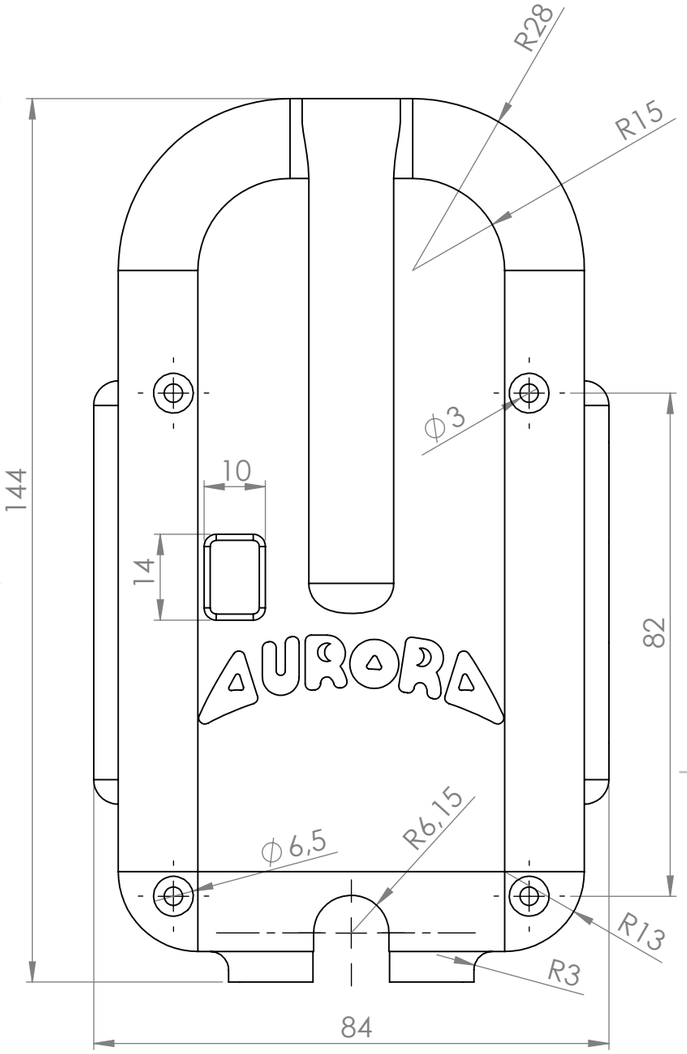
Plano Nº:
11

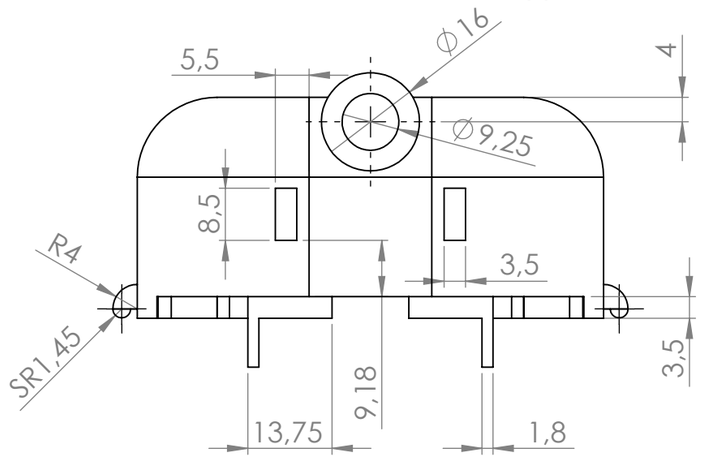




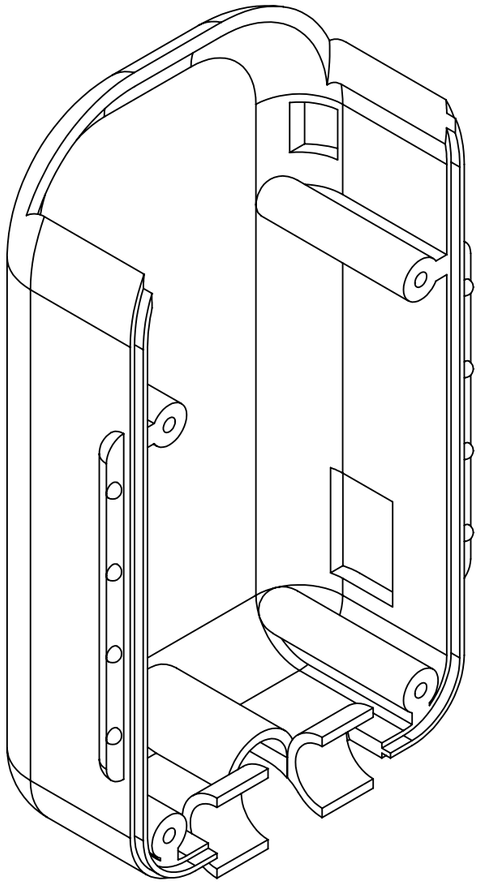
SECCIÓN A-A
ESCALA 1 : 1

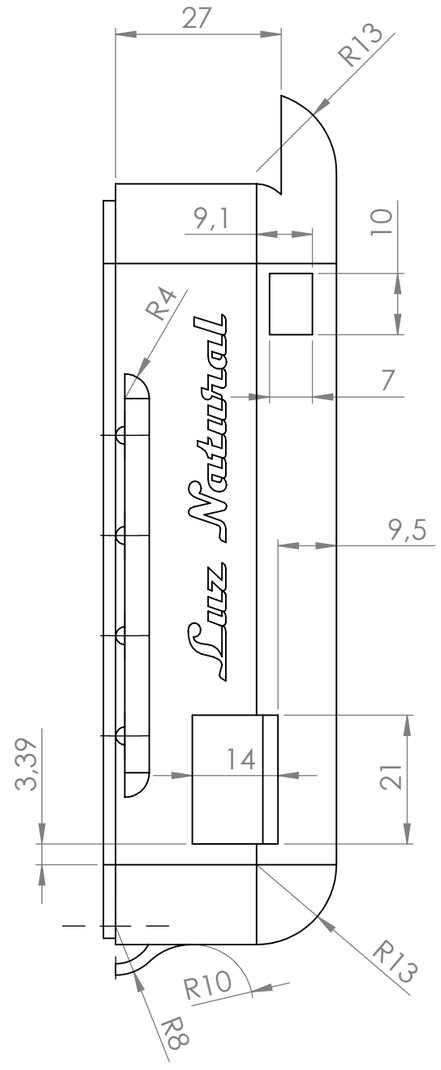


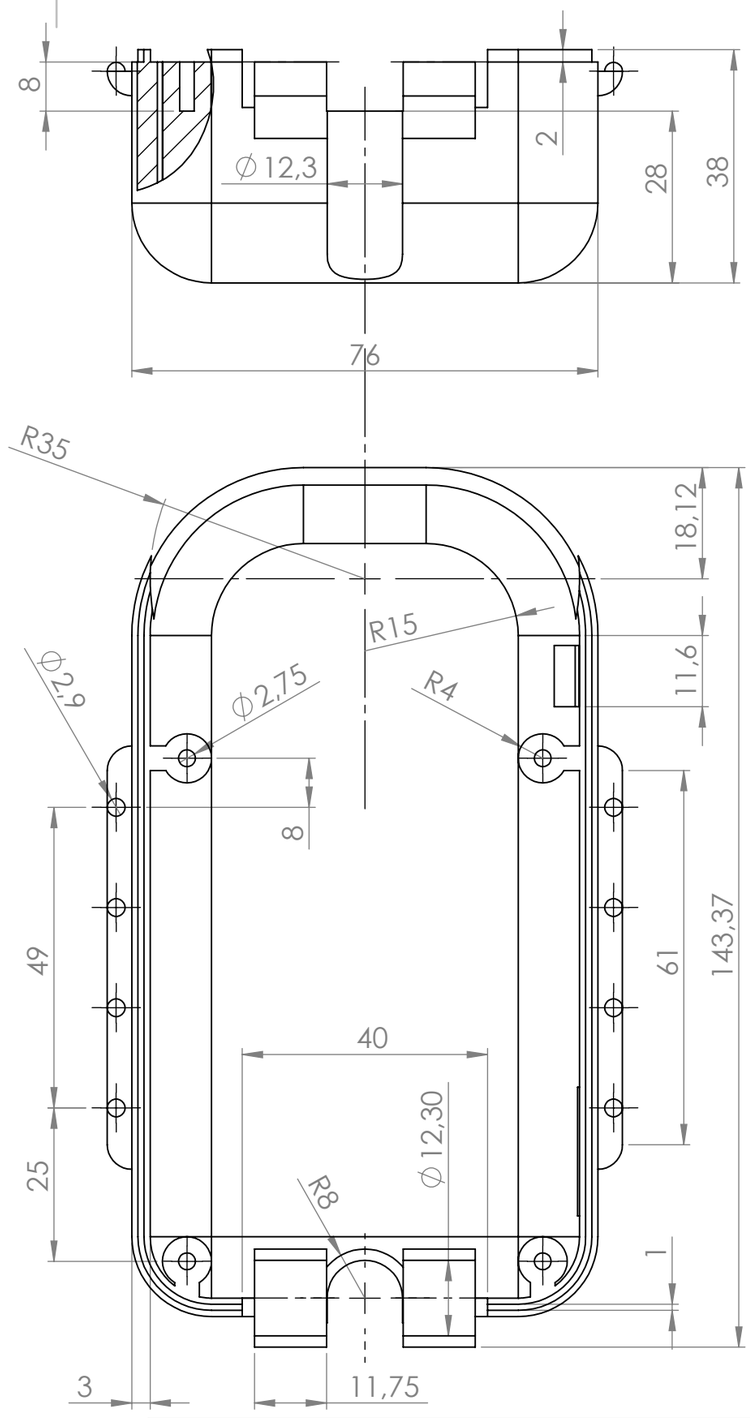




PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa Universitat Politècnica de València (UPV) Escuela Técnica Superior de Ingeniería del Diseño (ETSID)		Fecha: 27/5/2022 Escala **1:1**
Autor: Rafael Luis Saneleuterio Temporal	Plano: **Armazón principal**	Plano Nº: **12**







PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

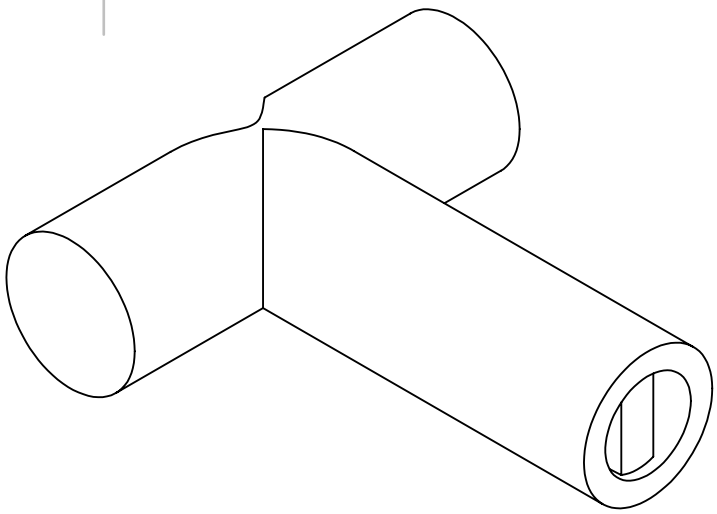
Fecha: 27/5/2022

Escala **1:1**

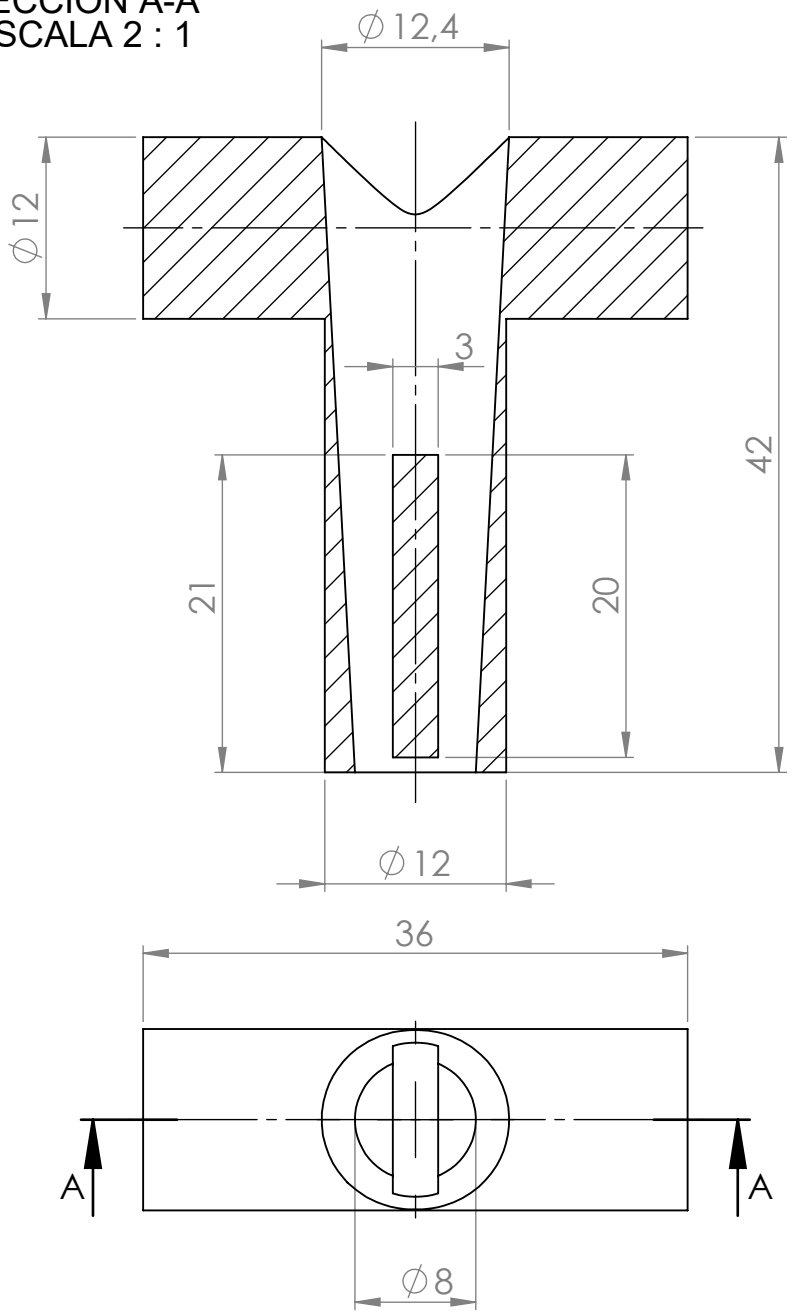
Autor:
Rafael Luis
Saneleuterio Temporal

Plano:
Tapa

Plano Nº:
13



SECCIÓN A-A
ESCALA 2 : 1



PROYECTO: Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa

Universitat Politècnica de València (UPV)
Escuela Técnica Superior de Ingeniería del Diseño (ETSID)

Fecha: 27/5/2022

Escala
2:1

Autor:
Rafael Luis
Saneleuterio Temporal

Plano:
Conducto

Plano Nº:
14

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

AURORA: DISEÑO Y PROGRAMACIÓN DE UN AUTOMATISMO PARA CORTINAS COMUNICADO CON ALEXA

3. Pliego de condiciones

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Rafael Luis Saneleuterio Temporal

TUTORIZADO POR

Patricia Balbastre Betoret

CURSO ACADÉMICO: 2021/2022

Índice pliego de condiciones

Índice de tablas del pliego

1. Definición y alcance del pliego

En el presente documento se especifican los materiales, servicios, ejecuciones, pruebas y ajustes para la correcta implementación del prototipo Aurora. El ámbito de aplicación de este apartado se extiende a todos los subsistemas electrónicos, de programación y mecánicos que forman parte del dispositivo.

Quedan excluidos todos los trabajos y condiciones en cuanto al modo de adquisición o compra de los elementos de Aurora así cómo la configuración detallada del *software* la cual se manifiesta en la memoria del proyecto. También quedan fuera de la especificación todas aquellas implementaciones que no se ajusten al funcionamiento nominal del prototipo.

2. Especificaciones técnicas

2.1. Condiciones de los materiales

2.1.1. Subsistema electrónico

En la siguiente tabla (tabla 1) se listan los componentes electrónicos utilizados junto con el modelo concreto y condiciones de utilización.

Componente	**Cantidad**	**Modelo**	**Observaciones**
Placa de desarrollo	1	NodeMCU Lua Lolin V3	Lógica a 3,3 V
Controlador de potencia	1	L298N	Máximo: 35 V, 2 A
Motor DC	1	Motor con reductor 1:120 de un solo eje	Rango: 3 V - 6 V Máximo: 130 mA
LED	1	Diodo LED rojo 5 mm	Rango: 1,6 V - 2 V
Sensor de temperatura	1	TMP36	Tª: -40 ºC - 125 ºC
Sensor de luz	1	LDR GL5516	Consumo máximo: 100 mW
Resistencia 1 KΩ	1	1000 Ohmios E12	Error máximo: 5 %
Resistencia 220Ω	4	220 Ohmios E12	Error máximo: 5 %
Placa de prototipos	2	*Protoboard* 170 puntos	Paso de *sockets*: 2,54 mm
Cable macho a hembra	2	Cables *jumpers* hembra/macho	Máximo: 3 A De 10 cm a 20 cm
Cable macho a macho	16	Cables *jumpers* macho/macho	Máximo: 3 A De 10 cm a 20 cm
Cable USB	1	Cable de USB a micro USB B 2.0	Transmisión de carga y datos

Batería	1	Batería de polímero de iones de litio 2600 mAh	Límite carga: 4,2 V Límite descarga: 3 A
Convertidor DC-DC	1	MT3608	Rango entrada: 2 V - 24 V
Integrado cargador	1	TP4056	Tensión entrada: 5 V
Interruptor	1	Interruptor basculante 3 posiciones	Máximo: 20 A
Alambre de soldar (5 g)	1	Alambre de estaño de 0,8 mm de grosor	Sn mínimo: 63 %

Tabla 1: Lista de material electrónico

2.1.2. Subsistema de programación

Se requiere de una computadora con acceso a la red. En concreto, se ha desarrollado el prototipo en un portátil con un Intel core i7 (séptima generación) con 16 GB de memoria RAM, y Windows 10 Pro. La tarjeta gráfica que se ha explotado es la propia integrada por Intel. Así mismo, el ordenador ha de ser compatible con el entorno de desarrollo de Arduino IDE, en concreto se ha programado en la versión 1.8.19.

Por otro lado, es necesario disponer de un teléfono inteligente (*smartphone*) con acceso a una tienda de software por la cual se pueda adquirir la aplicación Amazon Alexa. Concretamente se ha utilizado la versión 2.2.463321.0 para la configuración de Aurora.

2.1.3. Subsistema mecánico

En la tabla 2 se listan los materiales utilizados para el subsistema mecánico junto con sus cantidades.

Material	**Cantidad**
Plástico PLA para impresión 3D	230 g
Silicona A40	10 g
Perno cabeza hexagonal M10x120 DIN 933 de acero ISO 4017	1 unidad
Muelle 1 mm grosor	1 cm
Arandela M10 DIN 9021 de acero ISO 4017	1 unidad
Tornillo cabeza ranurada M3x10 DIN 933 de acero ISO 4017	7 unidades
Tornillo cabeza ranurada M3x20 DIN 933 de acero ISO 4017	1 unidad
Tornillo cabeza ranurada M3x30 DIN 933 de acero ISO 4017	2 unidades
Cola termofusible	4 g

Tabla 2: Lista de material mecánico

2.2. Condiciones de la ejecución

2.2.1. Conexiones eléctricas

En primer lugar, se unen las dos *protoboards* a través de los encajes macho/hembra de tal manera que las ranuras centrales queden paralelas quedando una matriz de 17x20 puntos.

A continuación, se inserta la placa de desarrollo poco a poco y ejerciendo presión homogénea en todos sus puntos hasta que esta quede bien encajada. Su orientación ha de mantener sus dos filas de pines paralelos a las ranuras centrales de las protoboards y el puerto USB orientado hacia la izquierda. La placa se deberá situar justo en medio (de derecha a izquierda) dejando una fila de puntos libre a cada lado y los pines superiores del módulo irán conectados en la cuarta fila por arriba.

Algunos componentes tienen unas conexiones particularmente condicionadas que se explican a continuación:

- El LDR se ha de soldar a dos cables macho a macho (M/M) y, antes de su conexión al divisor resistivo, han de pasarse por las dos ranuras del componente mecánico llamado conducto hasta que ambos sólidos se toquen.
- El sensor de temperatura se ha de situar en la última fila más abajo posible de la protoboard inferior.
- El convertidor DC-DC MT3608, una vez conectado a la batería, ha de ajustarse a un valor de tensión de salida de 7 V. Esto ha de hacerse girando el potenciómetro mediante un destornillador y sin carga conectada a la salida.
- Se ha de identificar la polaridad del motor DC, para ello, la tensión que genere un giro antihorario definirá el positivo y el negativo del componente.
- Antes de soldar los pines del interruptor, los cables en cuestión han de pasarse por el orificio inferior de la tapa.

Interconexiones		**Tipo de unión**
LED: pin negativo	R1: borne 2	*Protoboard*
NodeMCU: pin D2	LED: pin positivo	*Protoboard*
NodeMCU: pin GND	R1: borne 1	*Protoboard*
NodeMCU: pin A0	TMP36: pin analógico	Cable M/M (*Protoboard*)
NodeMCU: pin 3V	TMP36: pin positivo	Cable M/M (*Protoboard*)
NodeMCU: pin GND	TMP36: pin negativo	Cable M/M (*Protoboard*)
NodeMCU: pin GND	Divisor resistivo: pin GND	*Protoboard*
NodeMCU: pin D5	Divisor resistivo: pin V1	*Protoboard*
NodeMCU: pin D6	Divisor resistivo: pin V2	*Protoboard*
NodeMCU: pin D7	Divisor resistivo: pin V3	*Protoboard*

NodeMCU: pin 3V	Divisor resistivo: pin 3,3 V	*Protoboard*
NodeMCU: pin D1	L298N: pin IN4	Cable M/H (*Protoboard*)
NodeMCU: pin D0	L298N: pin IN3	Cable M/H (*Protoboard*)
NodeMCU: pin VU	L298N: pin +5 V	Cable M/M (*protoboard*/atornillado)
NodeMCU: pin GND	L298N: pin GND	Cable M/M (*protoboard*/atornillado)
Motor DC: pin positivo	L298N: pin OUT3	Cable M/M (soldado/atornillado)
Motor DC: pin negativo	L298N: pin OUT4	Cable M/M (soldado/atornillado)
MT3608: pin VOUT -	L298N: pin GND	Cable M/M (soldado/atornillado)
MT3608: pin VOUT +	L298N: pin +12 V	Cable M/M (soldado/atornillado)
MT3608: pin VIN -	Interruptor: pin I	Cable M/M (soldado)
MT3608: pin VIN +	Batería: pin +	Cable M/M (soldado)
Interruptor: pin 0	Batería: pin -	Cable M/M (soldado)
TP4056: B+	Batería: pin +	Cable M/M (soldado)
TP4056: B-	Interruptor: pin II	Cable M/M (soldado)

Tabla 3: Conexiones y tipos de conexión de los componentes electrónicos

2.2.2. Impresión 3D de las piezas

La obtención de las piezas de plástico PLA será a través del uso de una impresora 3D de filamento o FDM. Se ha de configurar una expansión horizontal de los orificios a -0,1 mm y utilizar soportes de material soluble.

Las diferentes piezas obtenidas deben sumergirse en agua tibia entre 6 y 12 horas hasta la disolución del material auxiliar. La impresión puede generar impurezas que han de eliminarse con una lija del 180.

Para el caso concreto de los rodamientos, estos deben imprimirse bajo las mismas condiciones pero a dos pasos, es decir, primero la mitad superior y luego la inferior. Los dos pares de mitades se unen a través de cola de contacto para plásticos y se presionan durante 5 minutos y luego se deja un mínimo de 2 horas de reposo.

Se ha de utilizar una macho de roscar M3 para generar las roscas donde se encajan los tornillos en las respectivas piezas impresas: 3 roscas en pinza A y 7 roscas en armazón principal.

Así mismo, a través de un macho de roscar M10 se crea la espiral por donde circula el perno hexagonal. Este se sitúa en el orificio cilíndrico tras el armazón principal y, dada su gran profundidad, es necesario operar con cuidado e ir sacando el material cada 4 giros como máximo.

Por último, la pieza del neumático ha de fabricarse a través de un molde impreso en FDM y, dentro de este, introducir la silicona A40 líquida.

2.2.3. Fijación de los componentes electrónicos a la carcasa

Paso 1: se atornilla el motor en la parte superior del armazón principal con el eje apuntando hacia el exterior. El atornillado ha de ser a través de 1 tornillo M3x20 para el hueco superior del motor y 2 tornillos M3x30 para los dos huecos restantes.

Paso 2: el conjunto de las *protoboards* con la placa de desarrollo se ajustan a la derecha del interior del armazón pasando los dos huecos de la NodeMCU de la parte del puerto USB por los dos cilindros habilitados.

Paso 3: de manera análoga al anterior paso, el controlador de potencia se fija a los cilindros de la izquierda dejando el disipador de los transistores al fondo del armazón.

Paso 4: fijar con silicona caliente la batería en el interior de la parte inferior derecha de la tapa.

Paso 5: fijar con silicona caliente el cargador de baterías de forma que quede alineado con el orificio superior de la tapa.

Paso 6: adherir el convertidor DC-DC sobre la batería.

Paso 7: introducir el interruptor por su orificio habilitado.

2.2.4. Ensamblaje carcasa

Paso 1: la rueda se forma uniendo el neumático con el disco y esta se inserta en el eje del motor.

Paso 2: se coloca el conducto en la bisagra habilitada en la parte inferior del armazón principal.

Paso 3: manteniendo todos los cables en el interior y sin rozar la rueda, se cierra la carcasa uniendo la tapa con el armazón.

Paso 4: atornillar la carcasa a través de cuatro tornillos de cabeza ranurada M3x10.

2.2.5. Ensamblaje agarre superior

Paso 1: se unen las dos pinzas (A y B) manteniendo los rodamientos en su interior.

Paso 2: atornillar las pinzas a través de tres tornillos de cabeza ranurada M3x10.

Paso 3: pasar la llave por el perno M10x120 y encajarlo a la cabeza hexagonal.

Paso 4: secuencialmente, pasar por el perno el muelle, la arandela y las pinzas atornilladas.

Paso 5: con ayuda de la llave unir el agarre superior con la carcasa enroscando el perno entre 4 y 7 cm a la vez que se introducen los dos pasadores del agarre por los orificios habilitados para ello.

2.2.6. Subida del programa

Se ha de conectar la placa de desarrollo a través del cable micro B 2.0 a USB al ordenador. Seguidamente, a través del Arduino IDE y con el programa abierto, se elige el puerto correspondiente.

A continuación, en la librería “arduino_secrets.h” se han de rellenar las variables particulares del nombre de la red y su contraseña. Por último, se sube el programa a la placa.

2.2.7. Control de ejecución

A través de los mensajes informativos del monitor serie de la herramienta de Arduino IDE y con la placa conectada se verifica que Aurora se conecta a la red y a la nube IoT de Arduino.

2.3. Pruebas y ajustes de servicio

Se instala el dispositivo Aurora en la barra de la cortina en cuestión, entre las dos últimas anillas y por el exterior del telón. Para ello, una vez colgado el prototipo, se ha de ajustar el perno girando la llave hasta que la rueda haga una ligera presión sobre la barra. La inscripción de “Luz Natural” de la tapa debe estar de cara hacia donde se quiere abrir la cortina.

Seguidamente, se orienta el conducto que contiene el sensor de luz hacia el exterior y se conecta Aurora poniendo el interruptor en la posición I.

A través de la aplicación móvil de Amazon Alexa se asocian las diferentes funcionalidades de Aurora. Mediante el comando de voz “Alexa, pon luz natural a X” se define la longitud de la barra siendo X el tamaño en decímetros.

Por último, se comprueba que Aurora reacciona a las órdenes de abrir y cerrar la cortina, así como a los diferentes niveles de luz activando el sensor de amanecer. También se verifica que Alexa puede informar de la temperatura que devuelve Aurora.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

AURORA: DISEÑO Y PROGRAMACIÓN DE UN AUTOMATISMO PARA CORTINAS COMUNICADO CON ALEXA

4. Presupuesto

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Rafael Luis Saneleuterio Temporal

TUTORIZADO POR

Patricia Balbastre Betoret

CURSO ACADÉMICO: 2021/2022

Índice presupuesto

Índice de tablas del presupuesto

1. Cuadro de precios elementales

Para obtener el listado de todos los elementos que intervienen en el presupuesto de Aurora, se ha optado por el método de precios descompuestos. Es por ello que, primero conviene inventariar todos los costes de material (tabla 1), mano de obra, equipos y *software* (tabla 2). En este apartado solo se tienen en cuenta los precios elementales, es decir, el coste unitario.

1. Cuadro de precios elementales I			
Ref	**Unidad**	**Descripción**	**Precio (€)**
Materiales			
m1	ud.	Placa de desarrollo NodeMCU Lolin V3	5,13
m2	ud.	Módulo controlador de potencia L298N	6,31
m3	ud.	Motor DC 3V-6V con reductor de un solo eje	2,55
m4	ud.	Diodo LED rojo	0,04
m5	ud.	Sensor de temperatura TMP36	2,05
m6	ud.	Sensor de luz LDR GL5516	0,26
m7	ud.	Resistencia 220 Ohmios E12	0,02
m8	ud.	Resistencia 1000 Ohmios E12	0,02
m9	ud.	Protoboard 170 conexiones	0,79
m10	ud.	Cable macho a macho	0,05
m11	ud.	Cable macho hembra	0,05
m12	ud.	Cable de programación USB 3.0 a micro B 2.0 de datos	3,12
m13	ud.	Batería de iones de litio 2600 mAh	16,58
m14	ud.	Convertidor DC-DC MT3608	1,10
m15	ud.	Cargador de batería TP4056	0,95
m16	ud.	Interruptor triestado	3,56
m17	g	Alambre de estaño 0,8 mm de grosor	0,14
m18	kg	Plástico PLA para impresión 3D	14,24
m19	kg	Silicona A40	27,64
m20	ud.	Perno de cabeza hexagonal M10x120	1,98
m21	ud.	Arandela M10	0,36
m22	cm	Muelle 10 mm diámetro interior 1 mm grosor	0,15
m23	ud.	Tornillo cabeza ranurada M3x10	0,06
m24	ud.	Tornillo cabeza ranurada M3x20	0,06
m25	ud.	Tornillo cabeza ranurada M3x30	0,05
m26	kg	Cola termofusible	1,25

Tabla 1: Precios elementales de los materiales

1. Cuadro de precios elementales II			
Ref	**Unidad**	**Descripción**	**Precio (€)**
Mano de obra			
h1	h.	Ingeniero desarrollador *[1]	20,5
Equipos			
ei1	h.	Impresora 3D	0,04
ei2	h.	Estación soldadora	0,01
ei3	h.	Pistola de cola termofusible	0,01
ei4	h.	Ordenador	0,29
Software			
sw1	lic. mensual	Herramienta en línea Arduino IoT Cloud	0,00
sw2	lic. mensual	Arduino IDE	0,00
sw3	lic. mensual	Aplicación Amazon Alexa	0,00
sw4	lic. mensual	Programa Solidworks	7,89
Medios auxiliares			
	%	Medios auxiliares sobre costes directos *[2]	10%

Tabla 2: Precios elementales de mano de obra, equipo, *software* y medios auxiliares

*[1] El coste del ingeniero desarrollador se ha extrapolado del salario medio de los ingenieros industriales según las estadísticas de Jobted, [en línea]. Disponible en: <https://www.jobted.es/salario/ingeniero-industrial>

*[2] Los medios auxiliares hacen referencia a aquellos costes de difícil cuantificación incluyendo conceptos como las pequeñas herramientas, mermas de material, consumo eléctrico, etc. Estos gastos se integran en el presupuesto como un único porcentaje característico en proyectos de ingeniería (10 %) sobre el subtotal de materiales, mano de obra, equipos y *software.*

2. Cuadro de precios descompuestos

A continuación, se desglosa el proyecto en tres principales componentes que vendrían a ser el subsistema electrónico, el de programación y el mecánico. De manera que, se listan y cuantifican los materiales, equipos, mano de obra y *software* que intervienen en cada uno de los subsistemas, como se puede ver en las tablas 3, 4 y 5.

2. Cuadro de precios descompuestos I					
Ref	**Unidad**	**Descripción**	**Precio (€)**	**Cantidad**	**Parcial (€)**
d1	ud.	Subsistema electrónico formado por los componentes homónimos y las soldaduras incluyendo mano de obra			
Materiales					
m1	ud.	Placa de desarrollo NodeMCU Lolin V3	5,13	1	5,13
m2	ud.	Módulo controlador de potencia L298N	6,31	1	6,31
m3	ud.	Motor DC 3V-6V con reductor de un solo eje	2,55	1	2,55
m4	ud.	Diodo LED rojo	0,04	1	0,04
m5	ud.	Sensor de temperatura TMP36	2,05	1	2,05
m6	ud.	Sensor de luz LDR GL5516	0,26	1	0,26
m7	ud.	Resistencia 220 Ohmios E12	0,02	4	0,07
m8	ud.	Resistencia 1000 Ohmios E12	0,02	1	0,02
m9	ud.	Protoboard 170 conexiones	0,79	2	1,58
m10	ud.	Cable macho a macho	0,05	16	0,74
m11	ud.	Cable macho hembra	0,05	2	0,09
m12	ud.	Cable de programación USB 3.0 a micro B 2.0 de datos	3,12	1	3,12
m13	ud.	Batería de iones de litio 2600 mAh	16,58	1	16,58
m14	ud.	Convertidor DC-DC MT3608	1,10	1	1,10
m15	ud.	Cargador de batería TP4056	0,95	1	0,95
m16	ud.	Interruptor triestado	3,56	1	3,56
m17	g	Alambre de estaño 0,8 mm de grosor	0,14	5	0,69
Mano de obra					
h1	h.	Ingeniero desarrollador	20,5	100	2050,00
Equipos					
ei2	h.	Estación soldadora	0,01	1,5	0,02
Medios auxiliares					
	%	Medios auxiliares sobre costes directos	10%	2094,86	209,49
				Precio de desarrollo y ejecución material	2304,35

Tabla 3: Precios descompuestos del subsistema electrónico

2. Cuadro de precios descompuestos II					
Ref	**Unidad**	**Descripción**	**Precio (€)**	**Cantidad**	**Parcial (€)**
d2	ud.	Subsistema de programación formado por la configuración en Arduino IoT, Arduino IDE y Amazon Alexa incluyendo mano de obra			
Mano de obra					
h1	h.	Ingeniero desarrollador	20,5	100	2050,00
Equipos					
ei4	h.	Ordenador	0,29	100	28,97
Software					
sw1	lic. mensual	Herramienta en línea Arduino IoT Cloud	0,00	8	0,00
sw2	lic. mensual	Arduino IDE	0,00	8	0,00
sw3	lic. mensual	Aplicación Amazon Alexa	0,00	8	0,00
Medios auxiliares					
	%	Medios auxiliares sobre costes directos	10%	2078,97	207,90
				Precio de desarrollo y ejecución material	2286,86

Tabla 4: Precios descompuestos del subsistema de programación

2. Cuadro de precios descompuestos III					
Ref	**Unidad**	**Descripción**	**Precio (€)**	**Cantidad**	**Parcial (€)**
d3	ud.	Subsistema mecánico formado por el diseño asistido por ordenador (CAD), la fabricación de las piezas y el ensamblaje incluyendo mano de obra			
Materiales					
m18	kg	Plástico PLA para impresión 3D	14,24	0,23	3,28
m19	kg	Silicona A40	27,64	0,01	0,28
m20	ud.	Perno de cabeza hexagonal M10x120	1,98	1	1,98
m21	ud.	Arandela M10	0,36	1	0,36
m22	cm	Muelle 10 mm diámetro interior 1 mm grosor	0,15	1	0,15
m23	ud.	Tornillo cabeza ranurada M3x10	0,06	7	0,44
m24	ud.	Tornillo cabeza ranurada M3x20	0,06	1	0,06
m25	ud.	Tornillo cabeza ranurada M3x30	0,05	2	0,10
m26	kg	Cola termofusible	1,25	4	5,01
Mano de obra					
h1	h.	Ingeniero desarrollador	20,5	100	2050,00
Equipos					
ei1	h.	Impresora 3D	0,04	72	2,70
ei3	h.	Pistola de cola termofusible	0,01	1	0,01
ei4	h.	Ordenador	0,29	90	26,07
Software					
sw4	lic. mensual	Programa Solidworks	7,89	2	15,78
Medios auxiliares					
	%	Medios auxiliares sobre costes directos	10%	2106,19	210,62
				Precio de desarrollo y ejecución material	2316,81

Tabla 5: Precios descompuestos del subsistema mecánico

3. Valoración

El resumen del presupuesto lleva a la suma de todos los costes descompuestos (tabla 6). Finalmente, la valoración del proyecto Aurora: diseño y programación de un automatismo para cortinas comunicado con Alexa, asciende a 6 908,02 € sin IVA y 8 358,71 € con IVA.

3. Valoración					
Ref	**Unidad**	**Descripción**	**Precio (€)**	**Cantidad**	**Parcial (€)**
d1	ud.	Subsistema electrónico formado por los componentes homónimos y las soldaduras incluyendo mano de obra	2304,35	1	2304,35
d2	ud.	Subsistema de programación formado por la configuración en Arduino IoT, Arduino IDE y Amazon Alexa incluyendo mano de obra	2286,86	1	2286,86
d3	ud.	Subsistema mecánico formado por el diseño asistido por ordenador (CAD), la fabricación de las piezas y el ensamblaje incluyendo mano de obra	2316,81	1	2316,81
Total presupuesto de desarrollo y ejecución material antes de impuestos					6908,02
Impuestos					
	%	IVA	21%	6908,02	1450,68
Total presupuesto del proyecto Aurora					8358,71

Tabla 6: Valoración final del proyecto Aurora