Biolimpio

Desarrollo de prototipo esterilizador con leds UV

Pathways Tecnología Industrial S. de R.L. de C.V.
Colaboración con el Centro Inteligente de Capacitación e Innovación en Industria 4.0
Desarrollo de prototipo para esterilización mediante luz UV

Antecedentes del proyecto

Los avances en áreas como la medicina, microbiología y esterilización han tenido avances significativos durante las últimas décadas, los cuales han tenido un impacto positivo en la sociedad, logrando generar mayor conocimiento y conciencia en asuntos como enfermedades, limpieza, higiene, desinfección y microbios. Este conocimiento ampliamente adoptado por las sociedades actuales ha generado la necesidad del desarrollo de métodos de esterilización de objetos de uso común y de fácil acceso, sin la necesidad de acudir a un laboratorio especializado para conseguirlo, para que así el público en general tenga acceso a un entorno más saludable, evitando enfermedades comunes y problemas de salud.

Objetivos y alcances del proyecto

Se desarrollará el prototipado del **dispositivo esterilizador** para llevar a cabo prueba de concepto del producto que se vislumbra, así como identificar puntos críticos del dispositivo para que este pueda continuar su camino de desarrollo en etapas subsecuentes de diseño, ingeniería, producción y comercialización del producto.

Los objetivos principales del desarrollo del producto se establecen como:

// Desarrollo de prototipo esterilizador
// Desarrollo de aplicación de comunicación para visualización de esterilización

Requerimientos del proyecto

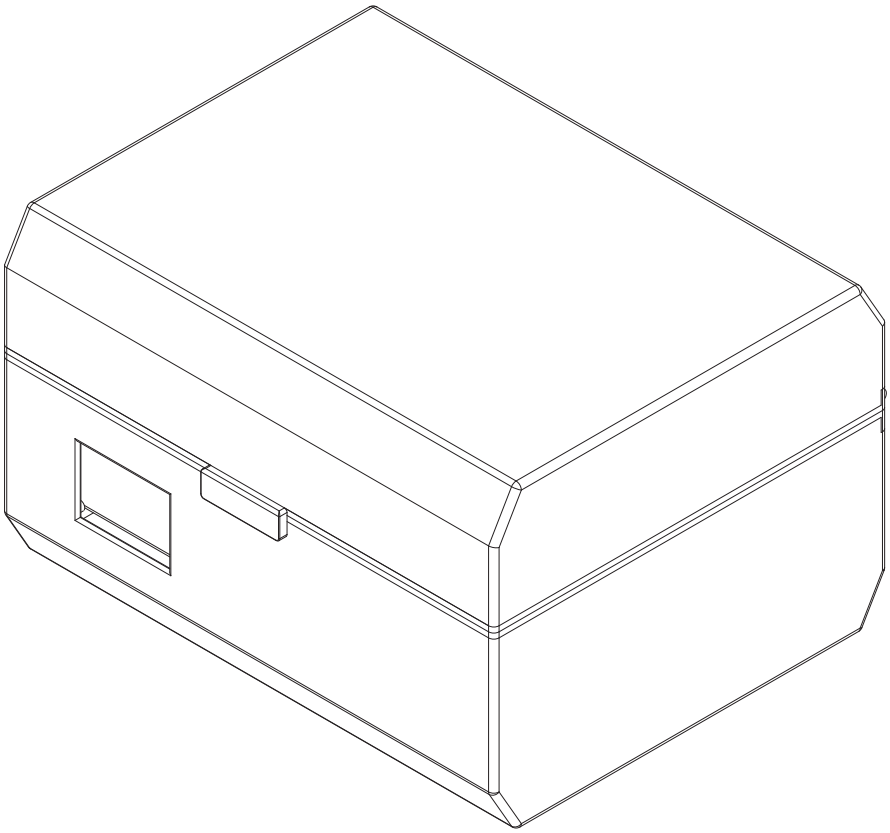
De acuerdo a los objetivos establecidos, se definen como **requerimientos** del proyecto:

- El dispositivo utilizará componentes comerciales de fácil acceso para su desarrollo.
- Se establece que el dispositivo contará con leds con frecuencia de emisión ultravioleta diseñados para esterilización de objetos.
- La integración de los componentes electrónicos se presentará en una carcasa sencilla y atractiva para un mercado general de consumidores
- Se considera la esterilización de objetos electrónicos, así como objetos utilitarios y de uso común; laptops, tabletas electrónicas, celulares, etc.
- La carcasa del dispositivo deberá ser capaz de contener dichos objetos a esterilizar.
- La carcasa contará con detección de apertura y cerrado de la tapa de la misma.
- Construcción de la carcasa será de aproximadamente 18” x 10” x 6”, con desviación del 20% en tamaño. El área de operación comprende un 30% de disminución aproximada en cada una de las dimensiones.
- El dispositivo utilizará toma de corriente 110 V AC.
- La construcción de la carcasa contempla materiales resistentes.
- La producción de la carcasa se realiza mediante impresión 3D permitiendo validación de dimensiones, modificaciones rápidas si estas son necesarias, construcción resistente, así como buena estética.

Memoria descriptiva y comunicación de resultados

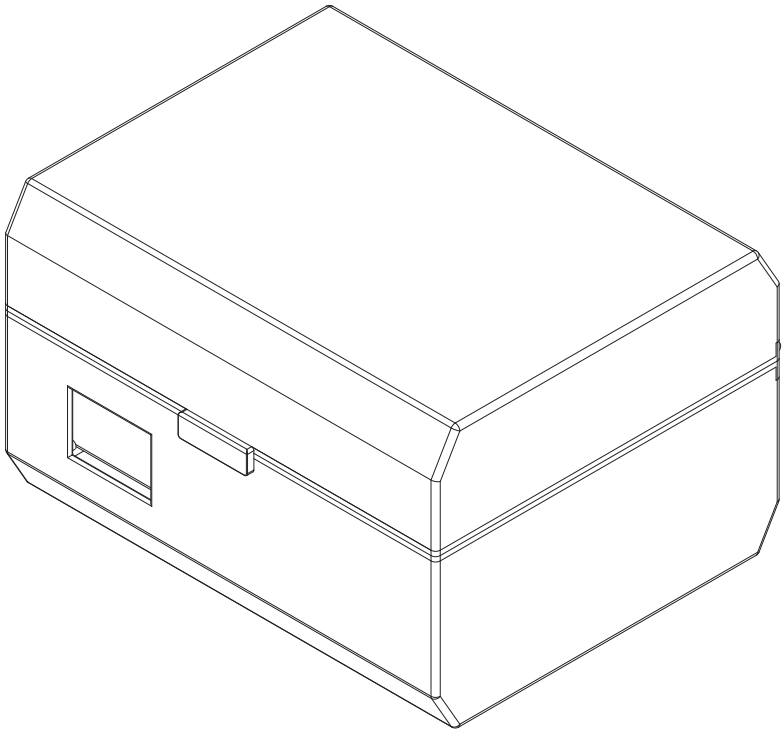
A continuación, se muestran los resultados finales del desarrollo del proyecto, abarcando esfuerzos de diseño, ingeniería y desarrollo de software e integración. Se habla acerca de factores funcionales, estéticos de producción e ingeniería general.

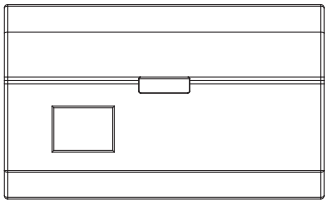
Diseño de carcasa

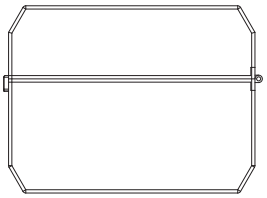


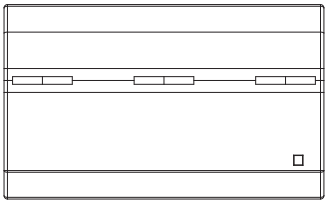
Ficha técnica del prototipo

Vistas generales e información general





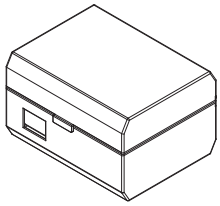


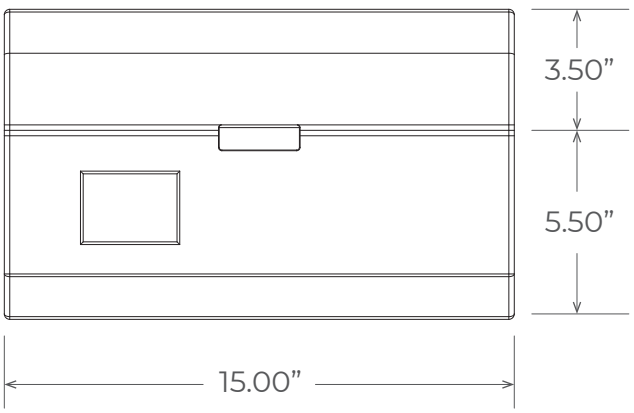


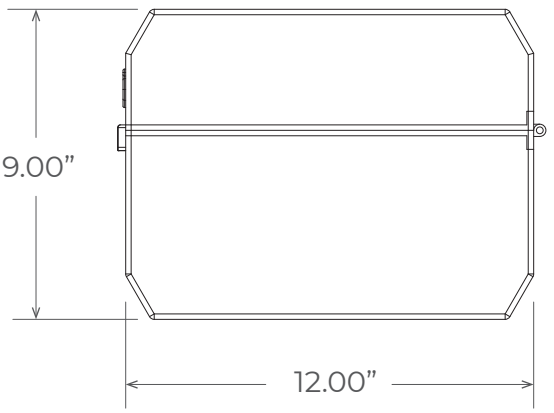
materiales: PLA
acabados: esmalte acrílico
peso: ~ 5 kg

Ficha técnica del prototipo

Dimensiones generales carcasa ensamblada & cerrada

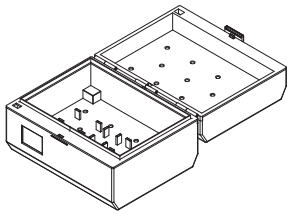


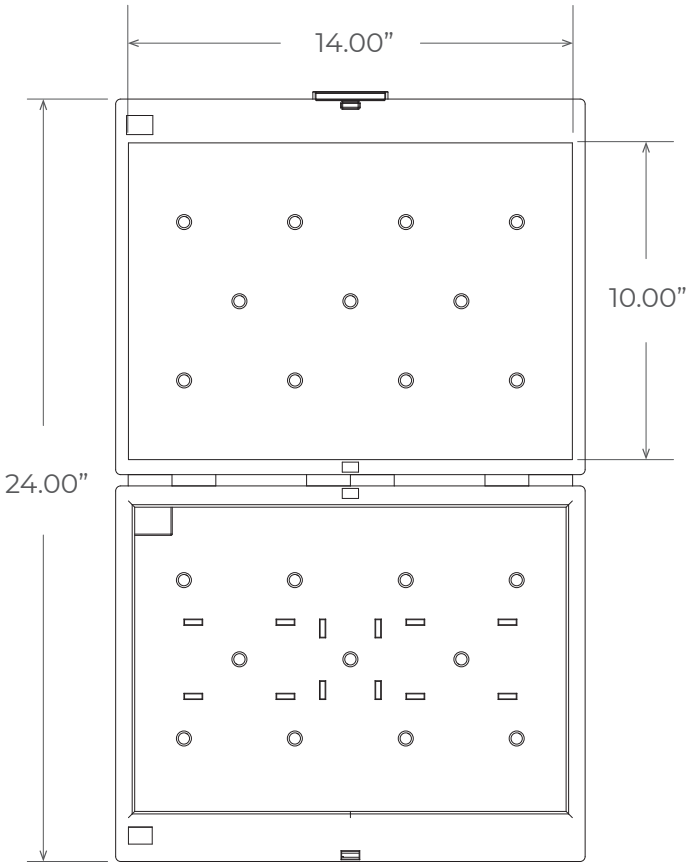


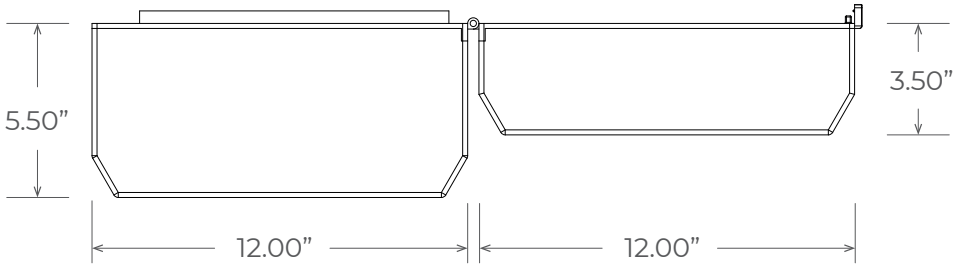


Ficha técnica del prototipo

Dimensiones generales carcasa ensamblada & abierta

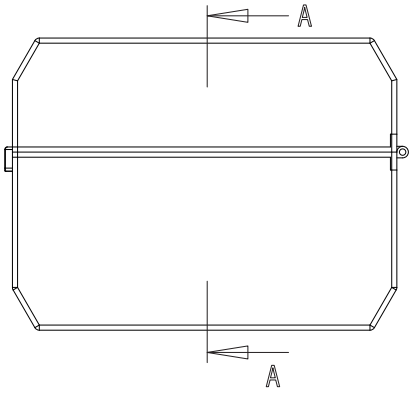


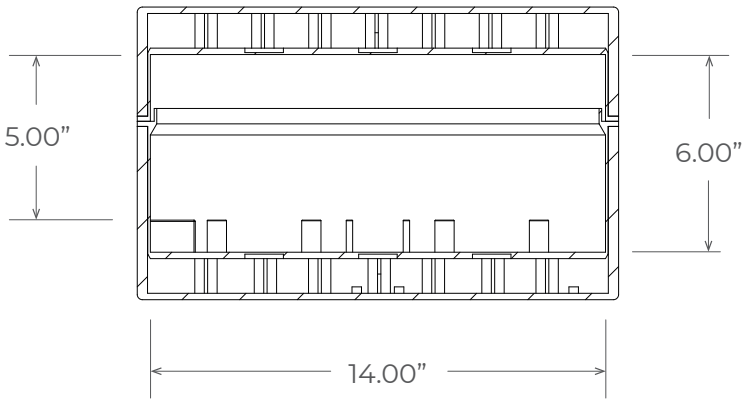


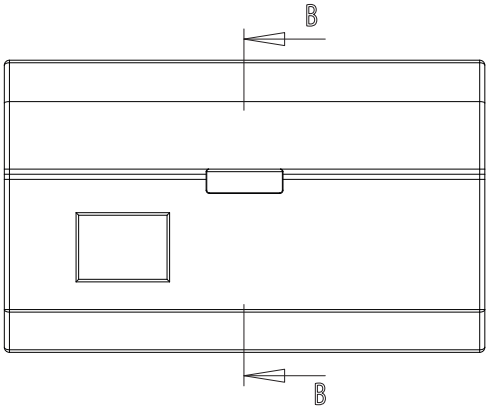


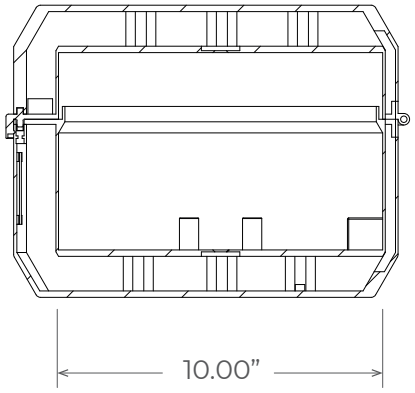
Ficha técnica del prototipo

Dimensiones generales del interior de la carcasa









Relación y ensamblaje de componentes de la carcasa

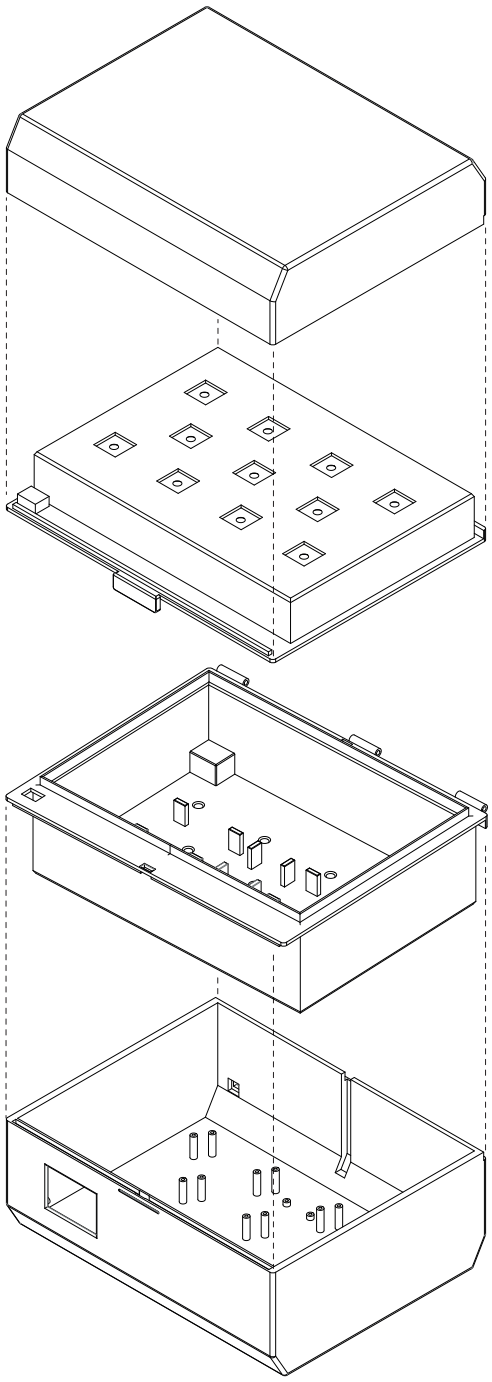
Vista explosiva #1 del ensamble

Carcasa externa superior (tapa) 1

Carcasa interna superior 2

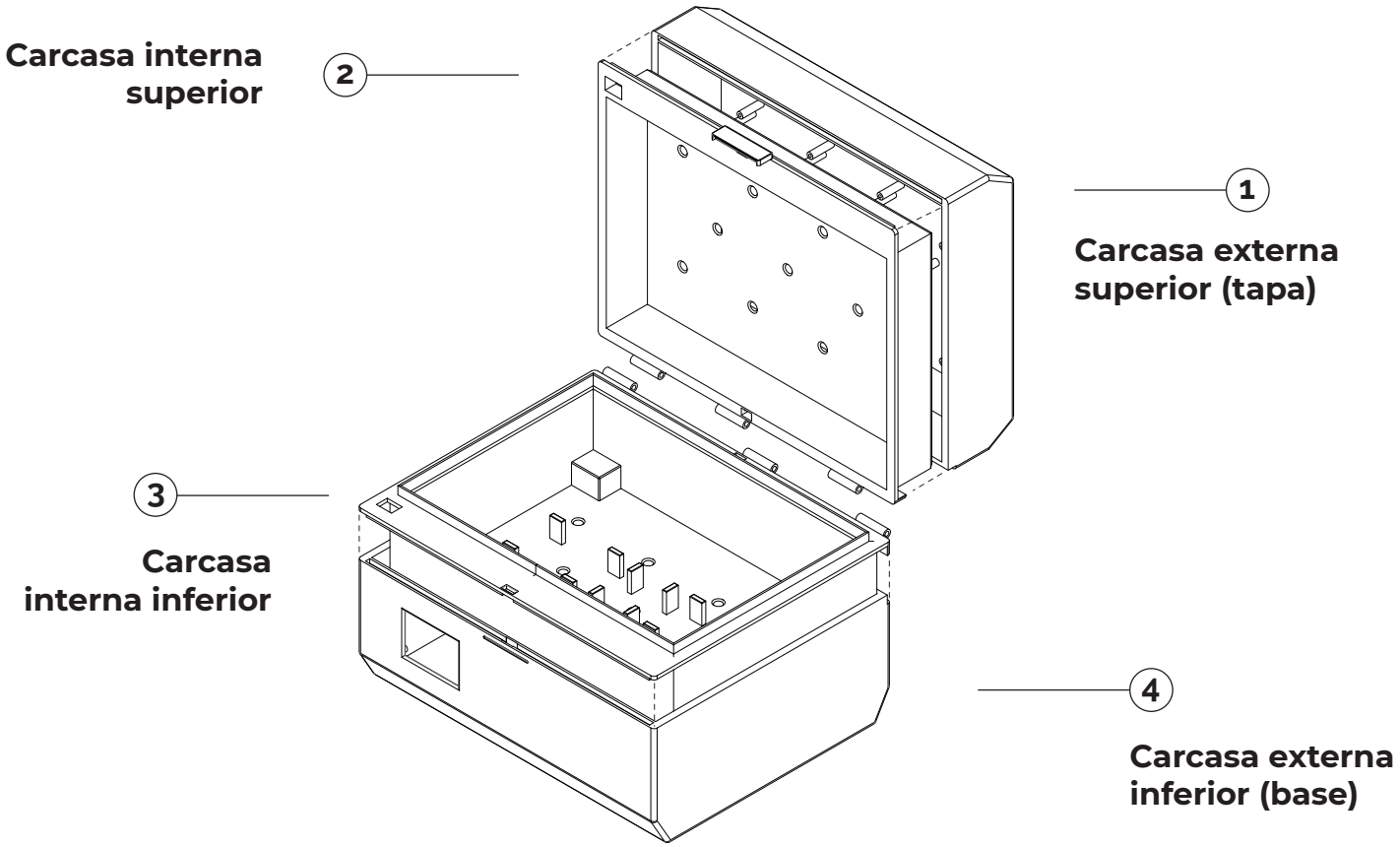
Carcasa interna inferior 3

Carcasa externa inferior (base) 4

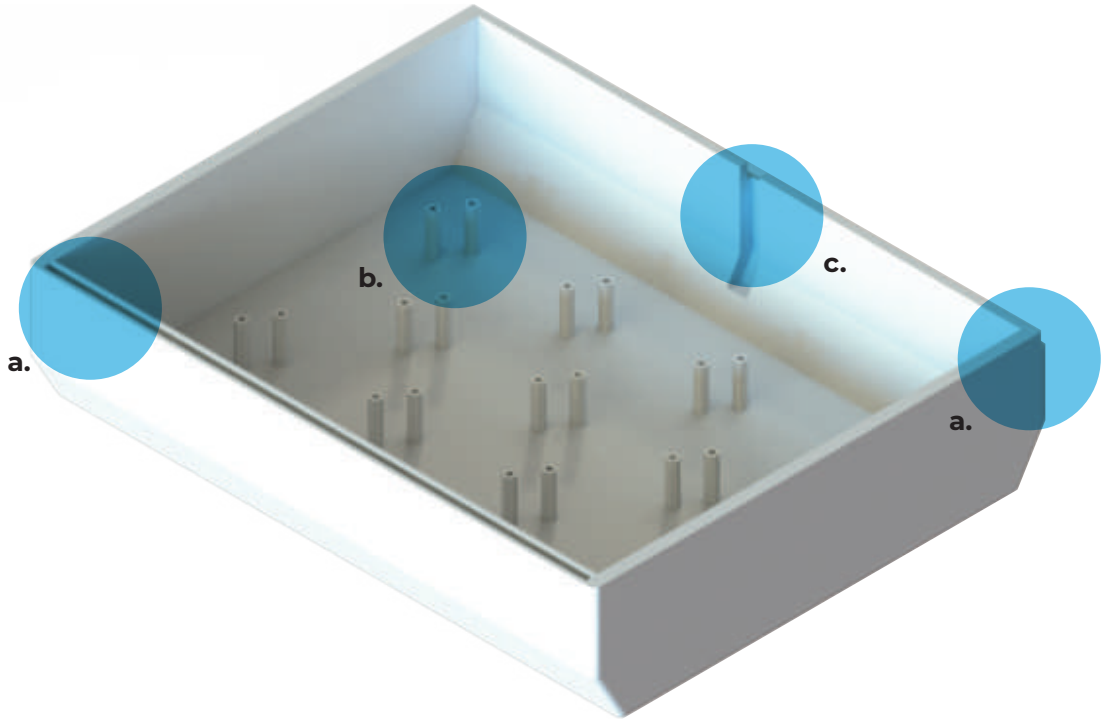


Relación y ensamblaje de componentes de la carcasa

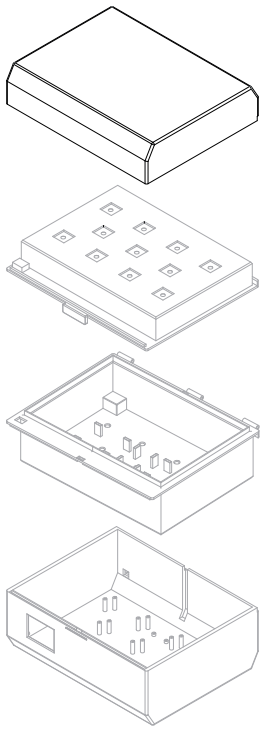
Vista explosiva #2 del ensamble



Características de los componentes



1 Carcasa externa superior (tapa)



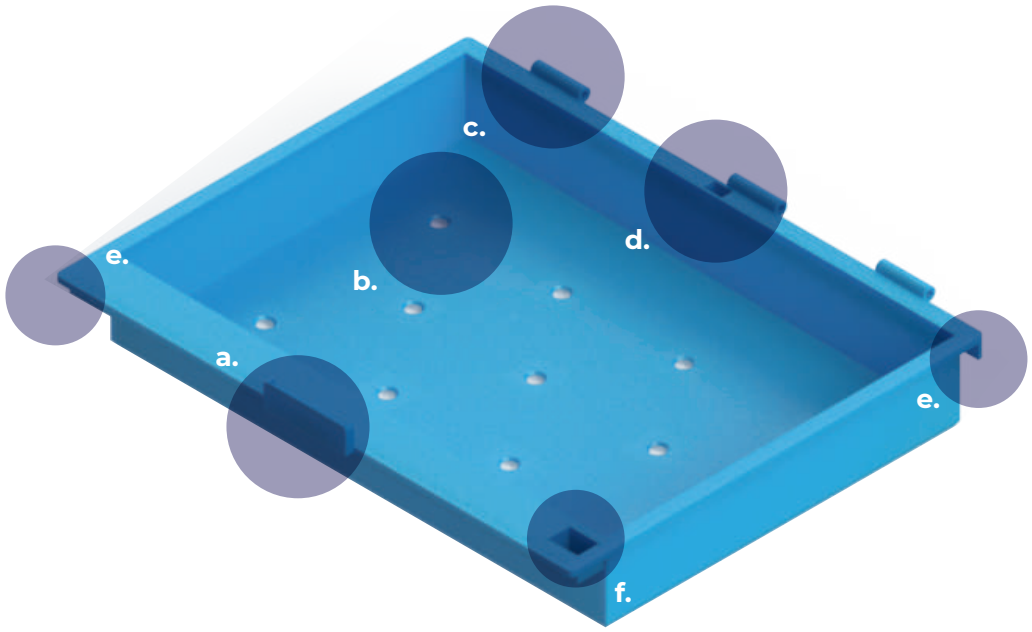
a. Elementos guía para unión y ensamblaje de componente 1 & 2.

b. Par de elementos para ubicación e instalación de LEDS UV, se cuenta con un total de 11 pares para instalación de 11 LEDS en el componente.

c. Ranura para conexión y unión de cableado y componentes electrónicos entre parte superior e inferior de la carcasa.

(consultar tabla 1 para mayor información acerca de componentes electrónicos)

Características de los componentes



2 Carcasa interna superior

a. Solapa auxiliar para apertura de la tapa de la carcasa y encendido/apagado de LEDS UV.

b. Perforaciones que permiten exposición de LED y LUZ UV al área de esterilización, el componente cuenta con un total de 11 perforaciones para 8 LEDS UV.

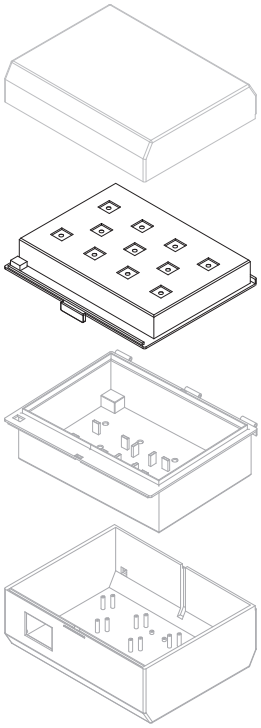
c. Bisagra, unión entre parte tapa y base (piezas superiores e inferiores). Permite apertura y cerrado de la carcasa.

d. Recorte para permitir paso de cableado desde parte inferior a superior y conexión electrónica.

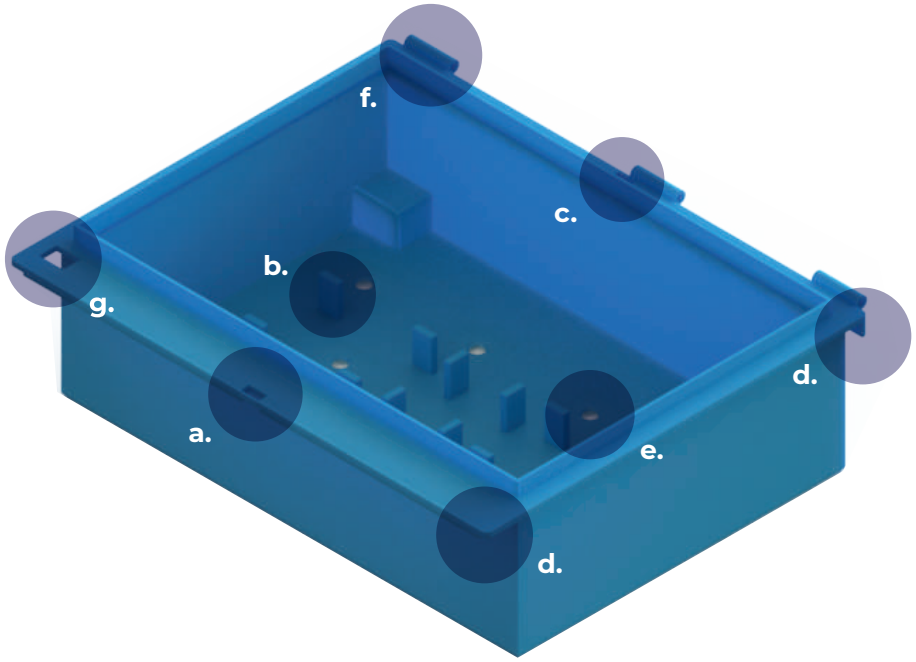
e. Elementos guía para ensamblaje de pieza 1 & 2.

f. Elemento para instalación rocker switch.

Notas: Las paredes interiores del componente estarán recubiertas de superficie reflectiva (espejo) para función óptima del prototipo.



Características de los componentes



③ Carcasa interna inferior

a. Recorte guía para ubicación e instalación de componente electrónico limit switch.

b. Relieves auxiliares para colocación de objetos a esterilizar.

c. Recorte para permitir paso de cableado desde parte inferior a superior y conexión electrónica.

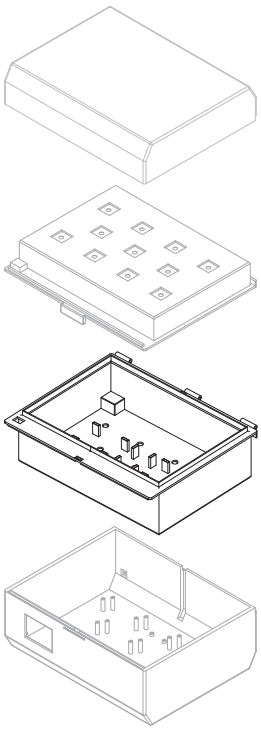
d. Elementos guía para ensamblaje de pieza 1 & 2.

e. Perforaciones que permiten exposición de LED y LUZ UV al área de esterilización, el componente cuenta con un total de 8 perforaciones para 11 LEDS UV.

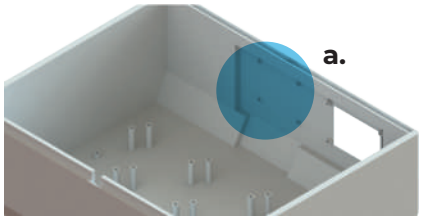
f. Bisagra, unión entre parte tapa y base (piezas superiores e inferiores). Permite apertura y cerrado de la carcasa.

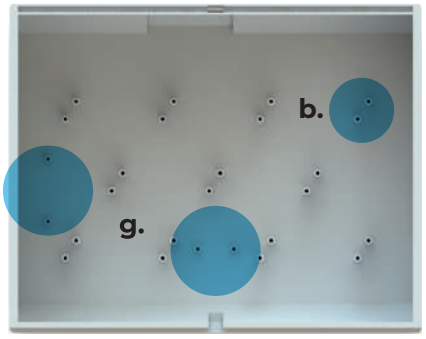
g. Elemento para instalación rocker switch.

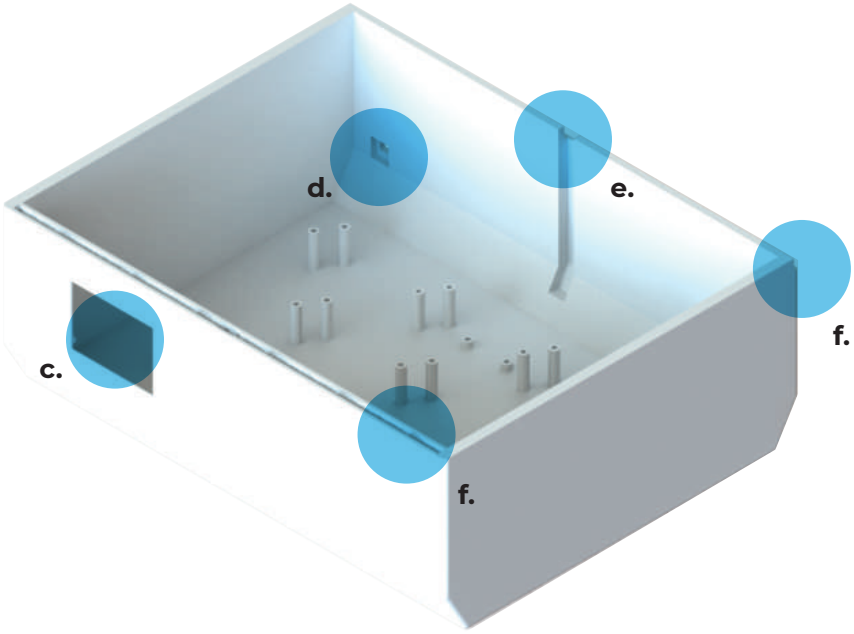
Notas: Las paredes interiores del componente estarán recubiertas de superficie reflectiva (espejo) para función óptima del prototipo.



Características de los componentes







4 Carcasa externa inferior (base)

a. Elementos para ubicación e instalación de componente Raspberry PI.

b. Elementos para ubicación de LEDs UV, elementos repetidos 11 veces para instalación de 11 LEDs.

c. Ubicación de pantalla touch screen.

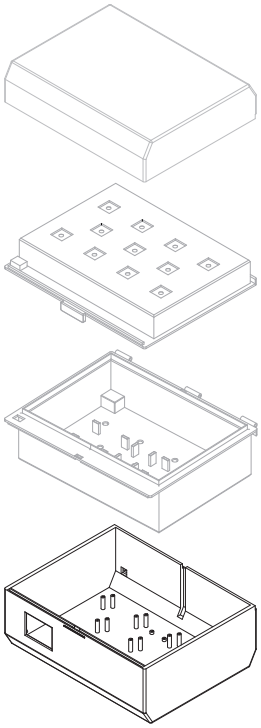
d. Ubicación salida fuente de energía.

e. Ranura para conexión y unión de cableado y componentes electrónicos entre parte superior e inferior de la carcasa.

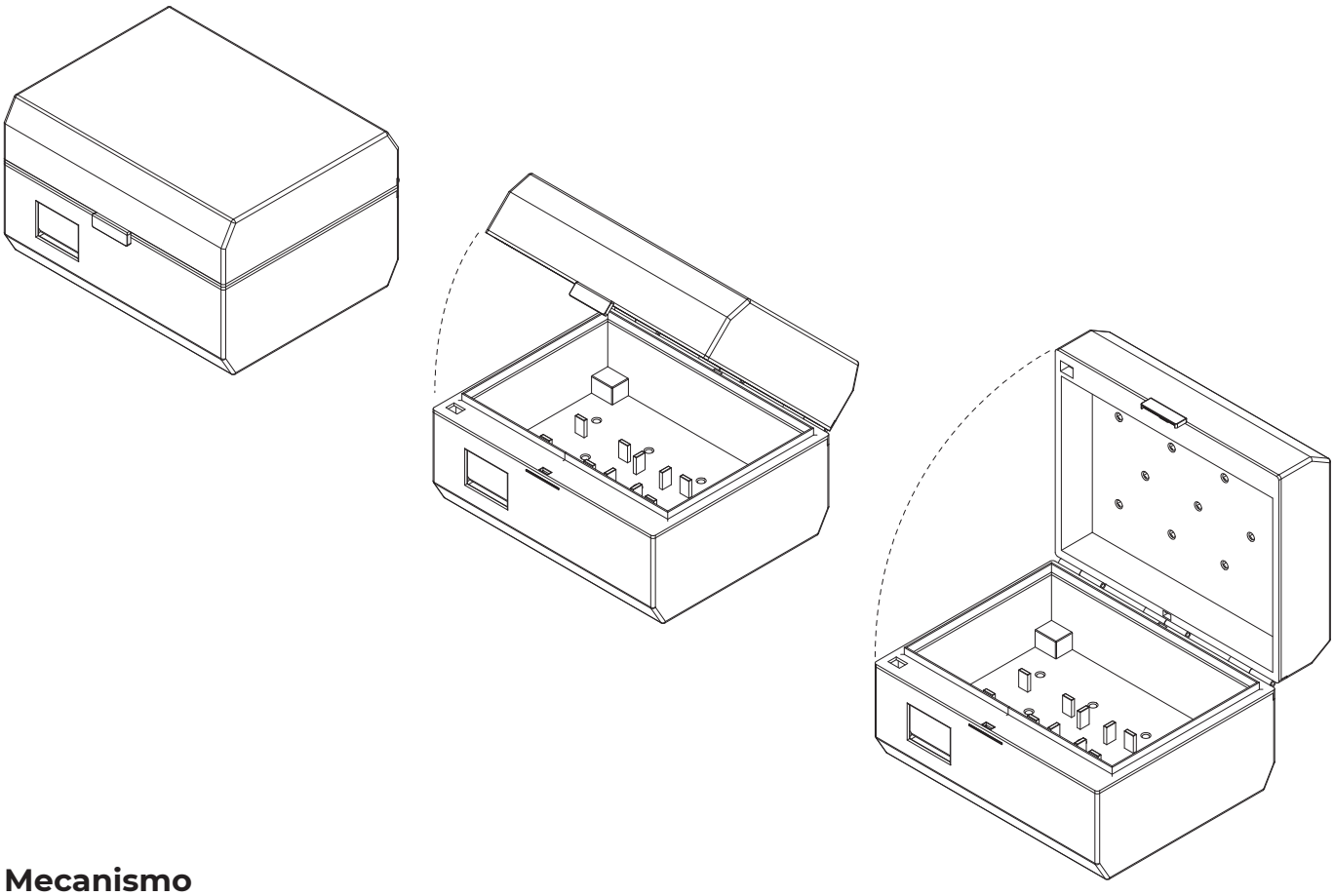
f. Elementos guía para unión y ensamblaje de componente 1 & 2.

g. Elementos auxiliares para instalación de componentes electrónicos.

(consultar tabla 1 para mayor informaicón acerca de componentes electrónicos)

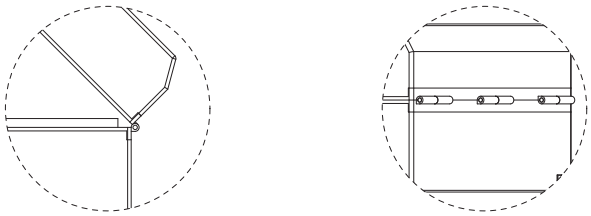


Mecanismos y funcionamiento

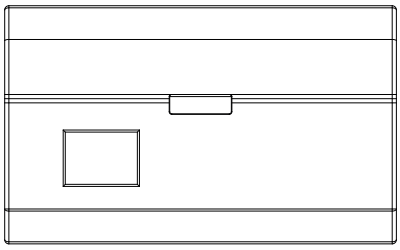


Mecanismo

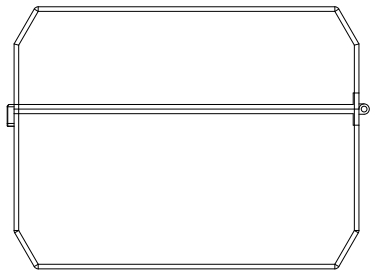
La parte inferior de la carcasa se une con la parte superior mediante una bisagra. La bisagra tiene 3 puntos de unión y permite el cierre y apertura de la “tapa” de la carcasa.

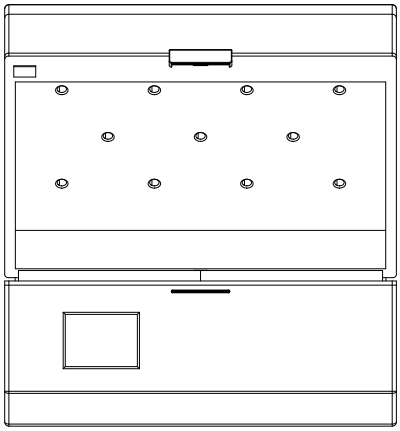


Mecanismos y funcionamiento

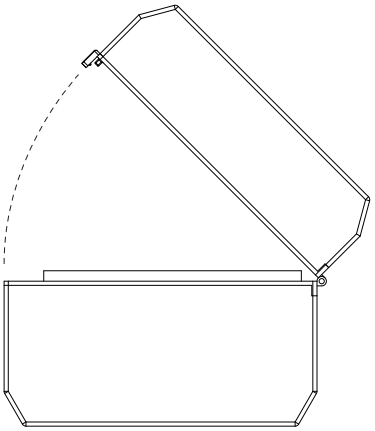


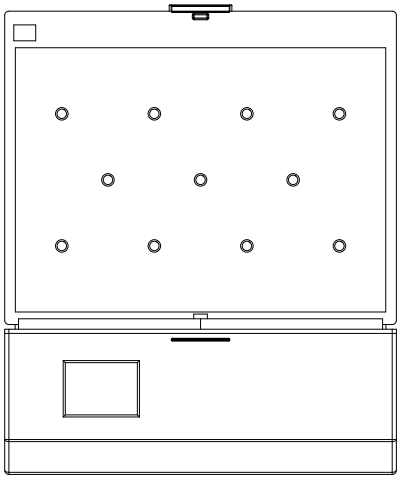
0°
Carcasa cerrada



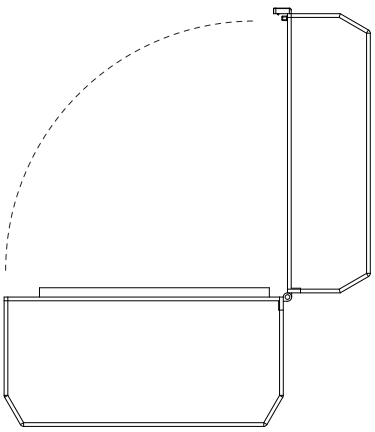


45°
Carcasa semi-abierta

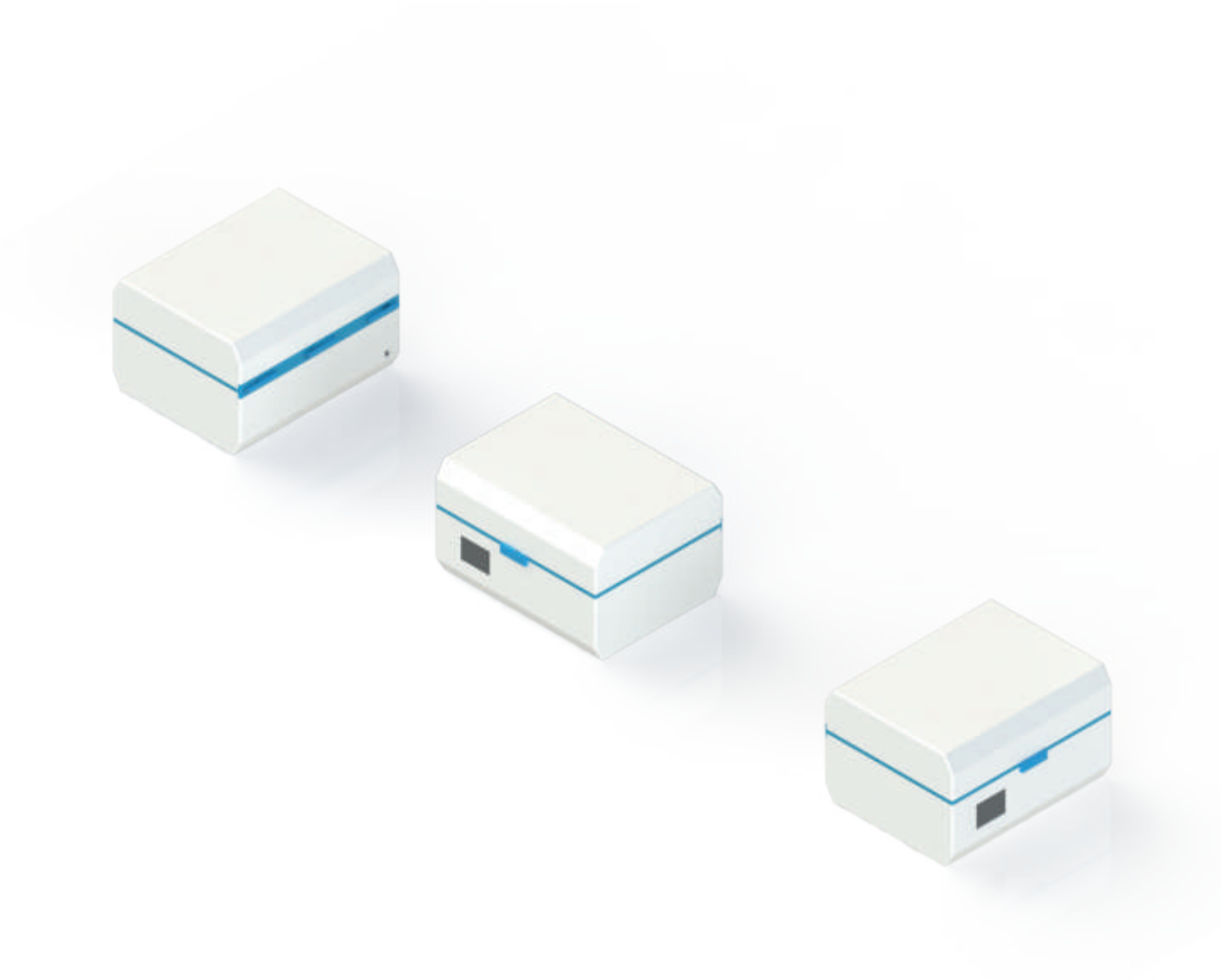




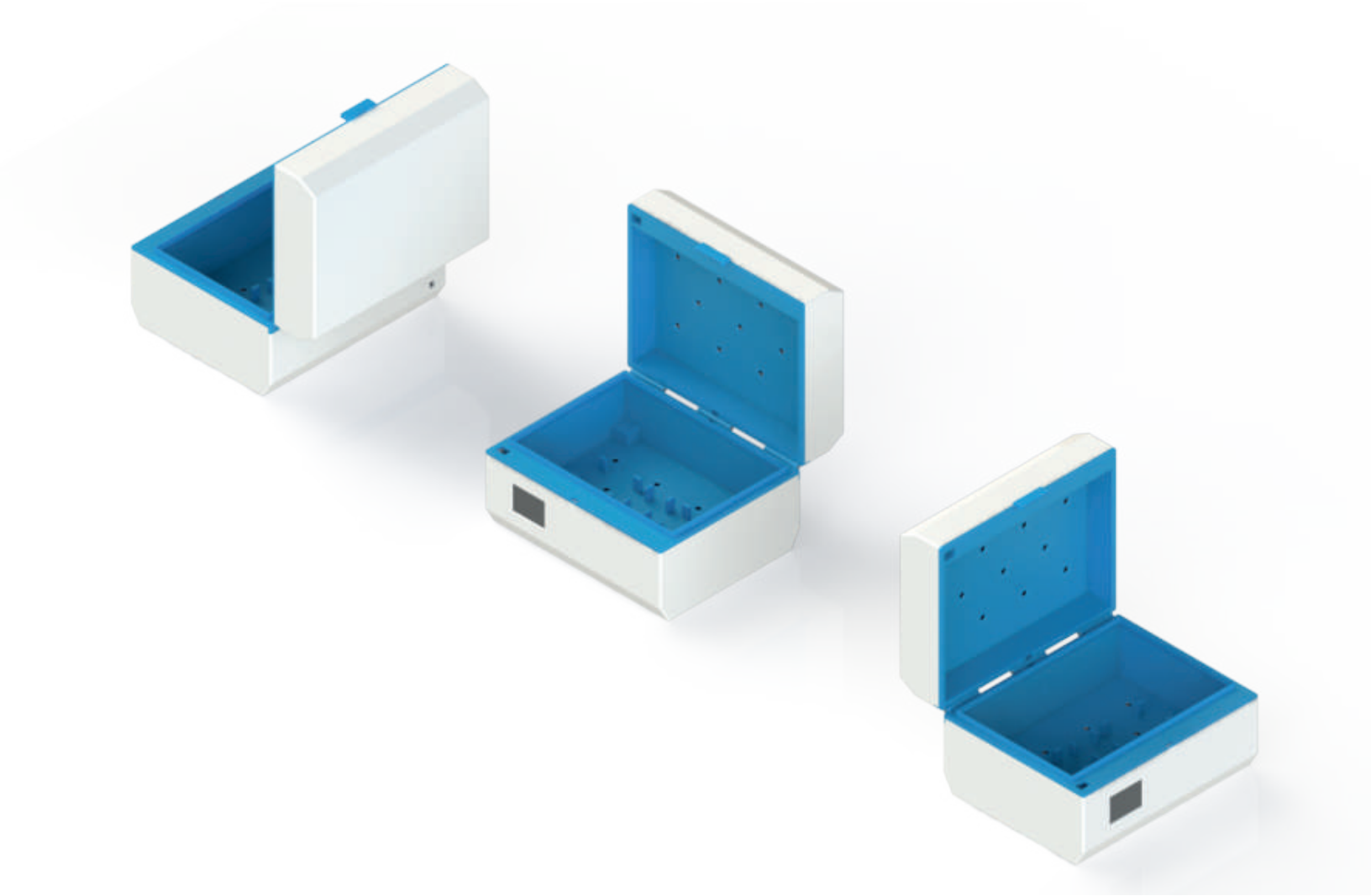
90°
Carcasa abierta



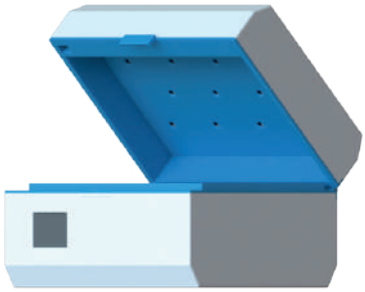
Vistas generales del prototipo

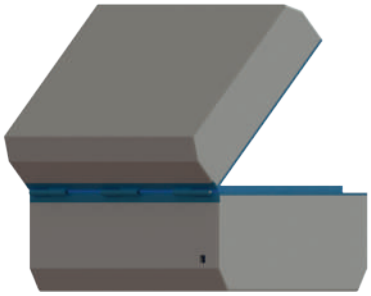


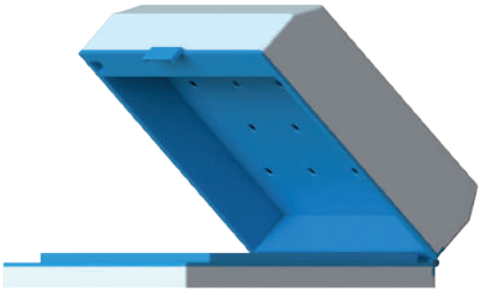
Vistas generales del prototipo

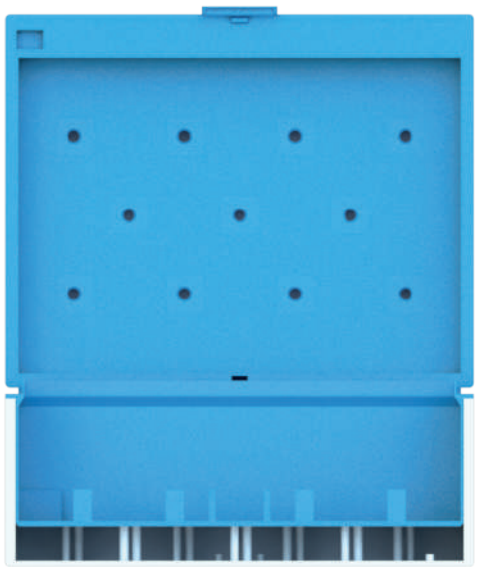


Vistas generales del prototipo

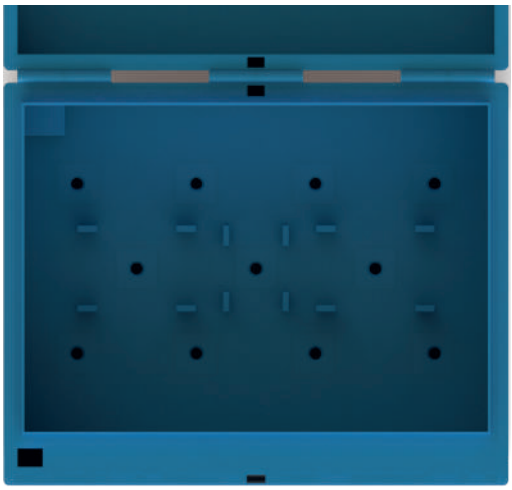








Vistas generales del prototipo



Componentes electrónicos y funcionamiento

Componentes electrónicos

Desglose de componentes

120 VAC – 12 VDC Convertidor

Convertidor de corriente alterna a corriente directa. Se encuentra en el cable de alimentación del equipo y se conecta a una toma de corriente alterna de 120V.

12 VDC -5VDC Convertidor

Convertidor de corriente directa a corriente directa. Se conecta al primer convertidor y es el responsable de alimentar al resto de los componentes electrónicos. Provee de una señal regulada de 5V de corriente continua.

Raspberry Pi 3 B v1.2

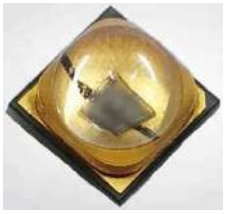
La raspberry Pi es la computadora que provee el control del equipo Biolimpio. En el dispositivo se almacena el software para controlar el encendido y apagado de los LEDs UV y la pantalla piTFT. Gran parte del pinout de la raspberry Pi se encuentra conectado a la pantalla touch. La raspberry Pi posee una interfaz para conexión inalámbrica, lo que permite conectar el equipo a una red Wi-Fi.

2.8” piTFT Touch Screen

La interfaz de usuario para controlar el equipo, su encendido, apagado, ajustes de tiempo y de red se hacen a través de una pantalla touch. Esta tiene una dimensión de 2.8 pulgadas y se conecta directamente a la Raspberry Pi a través de varios de los puertos del Pinout de la Raspberry Pi.

UV LED

Para la esterilización se utilizan LEDs UV con una longitud de onda entre 270 y 285 nm, recomendada para este tipo de procesos de inhibición bacteriana. Los LEDs se distribuyen en dos grupos principales, la tapa alta y la tapa baja, cada una con 11 LEDs, sumando 22 LEDs en total.



Limit Micro Switch

Llamado switch de límite, es utilizado para verificar la apertura y cierre de la tapa del dispositivo, para validar si los LEDs UV pueden encenderse o no, por cuestiones de seguridad.

Rocker Switch

Switch de encendido de dos estados, utilizado para apagar y encender el dispositivo.

Transistor 2N2222A

El transistor 2N222A es un transistor de uso general en circuitos electrónicos. En el caso de esta aplicación, es utilizado en modo “Switch ON-OFF”, donde utilizando una señal digital desde la Raspberry Pi es posible disminuir el consumo de corriente hacia los LEDs UV casi al mínimo, donde en términos prácticos es posible “apagarlos” cuando es necesario.

El LM317T	Regulador variable de voltaje, que soporta hasta 1.5 Amperes de corriente con su debida disipación de temperatura. Con un arreglo de resistencias y capacitores, es posible obtener valores de voltaje específicos entre 1.2 y 37 Voltios. En el caso de Biolimpio, es utilizado para proveer de voltaje a los LEDs UV, aproximadamente 8V.	

Tabla 1: desglose componentes electrónicos

Componentes electrónicos

LEDS UV

Se realizó una investigación sobre las opciones para esterilizar sin contacto superficies, y se llegó a la conclusión de que una de las opciones más accesibles y utilizadas es por medio de radiación ultravioleta. Según investigaciones médicas, el rango de longitud de onda más eficiente para la esterilización de virus y bacterias es de 260 a 280 nm.

Existen diferentes opciones para obtener esta radiación, pero las principales son LEDs (Diodo Emisor de Luz por sus siglas en inglés) y lámparas fluorescentes UV. Para el desarrollo del prototipo 0, se decidió optar por la opción de LEDs UV, por su mayor facilidad de distribuir la radiación en una superficie determinada con múltiples LEDs.

El modelo de los LEDs seleccionados es el siguiente:

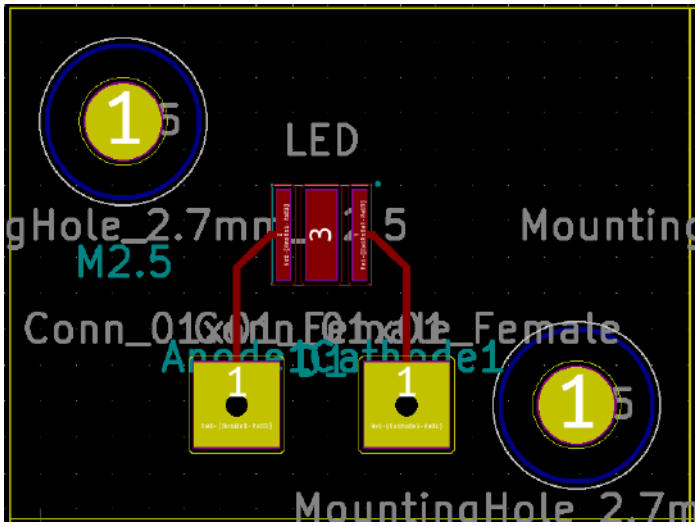
DUVTek UVC-LED Products

SPEC	Appearance	Wavelength	Optical Power	Driven Current	Operation Voltage	Package Size
L280 - QSC1F15A800		270~285 nm	1~3 mW	40 mA	5~8 V	3535
L280 - QSC1F20M001		270~285 nm	2~4 mW	40 mA	6~9 V	3535
L280 - SWC1S15A600		270~285 nm	2~4 mW	20 mA	5~8 V	3535
L280 - QSC1F30A700		270~285 nm	8~12 mW	120 mA	5~8 V	3535
L280 - SWC2S15A600		270~285 nm	8~12 mW	40 mA	10~16 V	6565

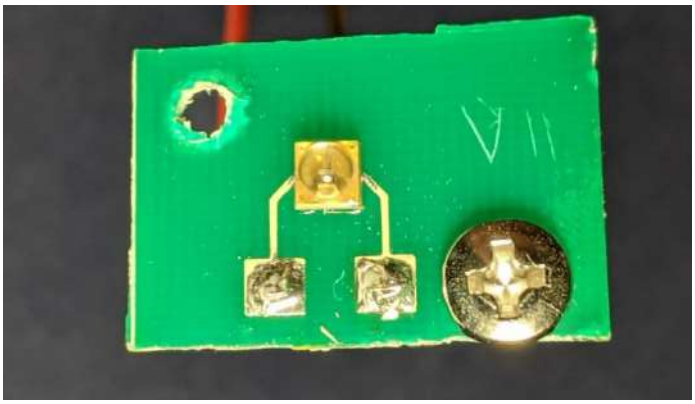
Especificaciones modelo LED UV

De ser necesario, existen opciones de mayor consumo de corriente y de mayor intensidad lumínica. **Se seleccionó la opción intermedia entre consumo e intensidad lumínica debido a que de esta forma se puede manejar una mayor cantidad de LEDs y distribuirse en una mayor área.**

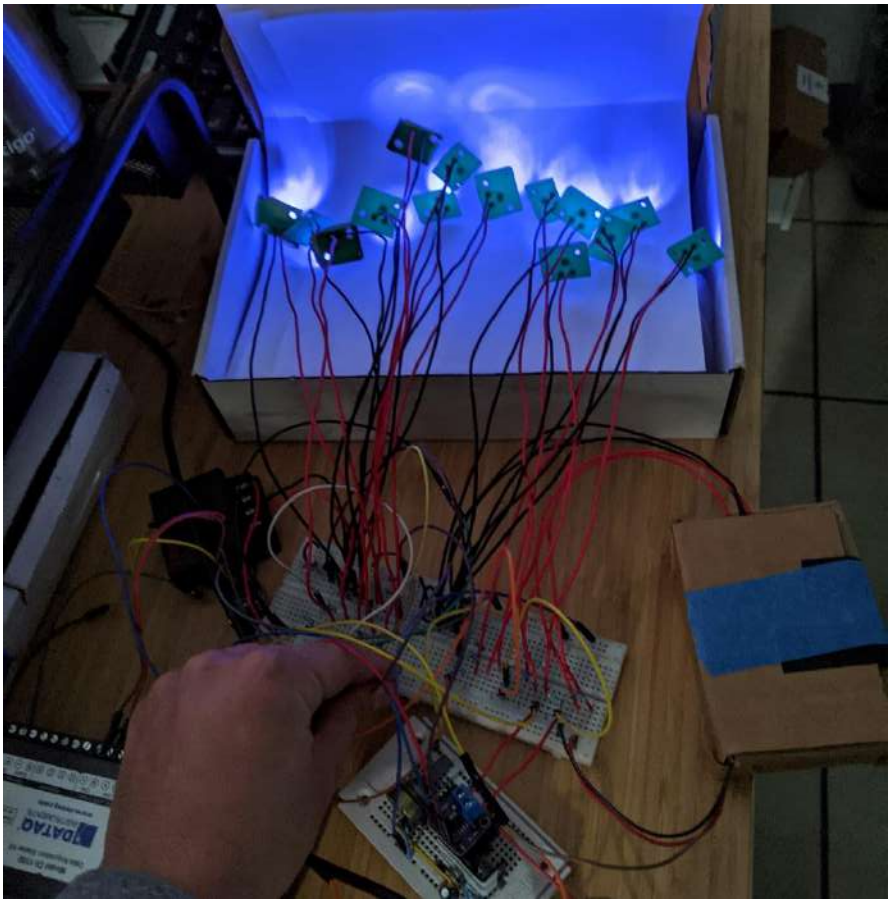
Para su instalación y uso, fue requerido fabricar placas de soldado a la medida, debido a que los LEDs adquiridos son fabricados para montaje superficial en circuitos impresos.



Diseño de PCB de soldado de LEDs UV en el software KiCAD



LED UV soldado en PCB



Prueba de encendido de LEDs UV

Componentes electrónicos

Análisis de consumo energético

Biolimpio está integrado de la siguiente manera:

El control de todo el dispositivo es realizado por medio de la microcomputadora Raspberry Pi, la cual controla la interfaz de usuario, conectividad, control de encendido y apagado de LEDs UV, y monitoreo de estado de señales de entrada.

Los transistores 2N2222A están configurados en su modo “ON/OFF”, donde utilizando una señal digital de bajo voltaje, es posible regular el paso de corriente hacia los LEDs UV, apagándolos y encendiéndolos instantáneamente. La señal de control para los transistores proviene de la Raspberry Pi.

La pantalla táctil es la encargada de recibir la interacción del usuario, y convertirla en señales que la Raspberry Pi pueda recibir para realizar la acción deseada.

Se realizaron pruebas de funcionamiento, obteniendo mediciones de corriente y voltaje. Se aproximan los parámetros de consumo de corriente para los principales parámetros en la siguiente tabla:

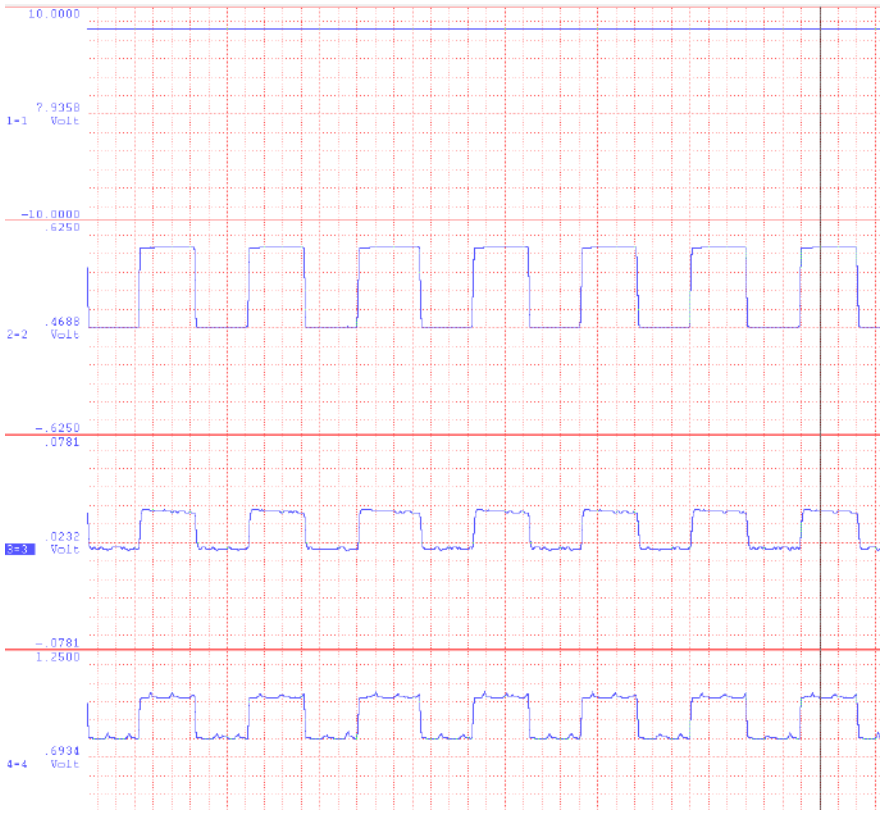
Medición	**Cantidad**
Consumo de 1 LED UV	45 a 55 mA aproximadamente.
Consumo de Raspberry Pi y Pantalla	380 a 480 mA aproximadamente.

Tabla 2: parámetros de consumo

Como se aprecia en el diagrama de conexiones (consultar en siguientes páginas), para mitigar los límites de corriente que pueden proveer los reguladores de voltaje, se dividen de la siguiente manera:

Un regulador de voltaje 5V, 3A provee de energía al Raspberry Pi y la pantalla táctil. El regulador de voltaje variable LM317T provee de energía a los LEDs UV. De esta forma, a pesar de la gran cantidad de consumo de corriente de 22 LEDs, no llegan a la corriente límite del LM317T de 1.5A. A pesar de esto, es necesario agregar un disipador de calor suficientemente grande para el circuito.

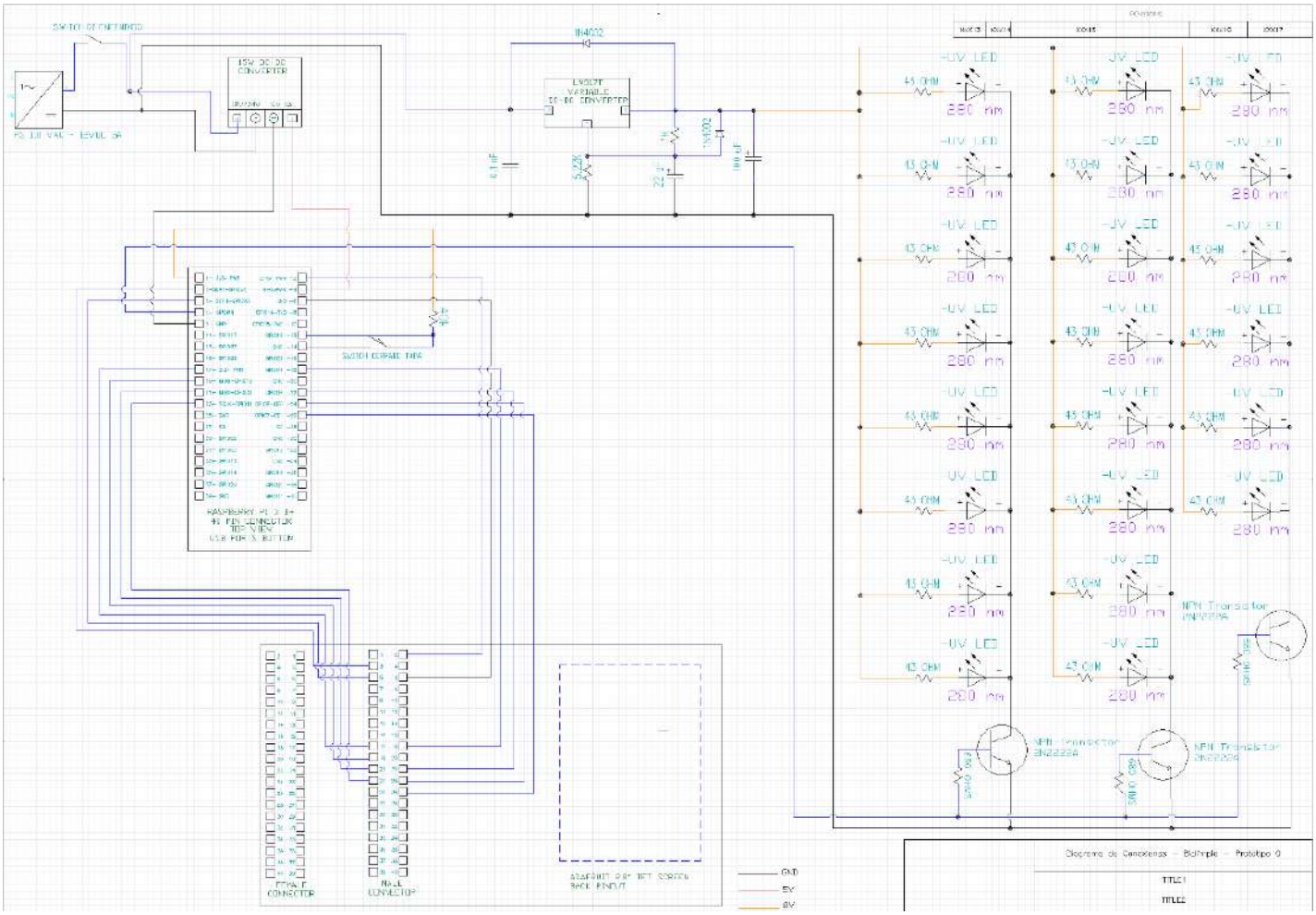
Para evitar que el calor llegue a temperaturas excesivas en uso continuo, se implementó la operación de encendido y apagado continuo de los LEDs UV, de forma que, por cada 5 segundos encendidos, los LEDs sean apagados 5 segundos. Así que, si se requiere un tiempo de exposición UV de 5 minutos (seleccionado en segundos en la pantalla de control), Biolimpio funcionará durante 10 minutos, la mitad del tiempo los LEDs están apagados, la otra mitad del tiempo encendidos.



Ejemplo de resultados de mediciones. La imagen muestra una prueba para 16 LEDs.

Componentes electrónicos

Diagrama de conexiones

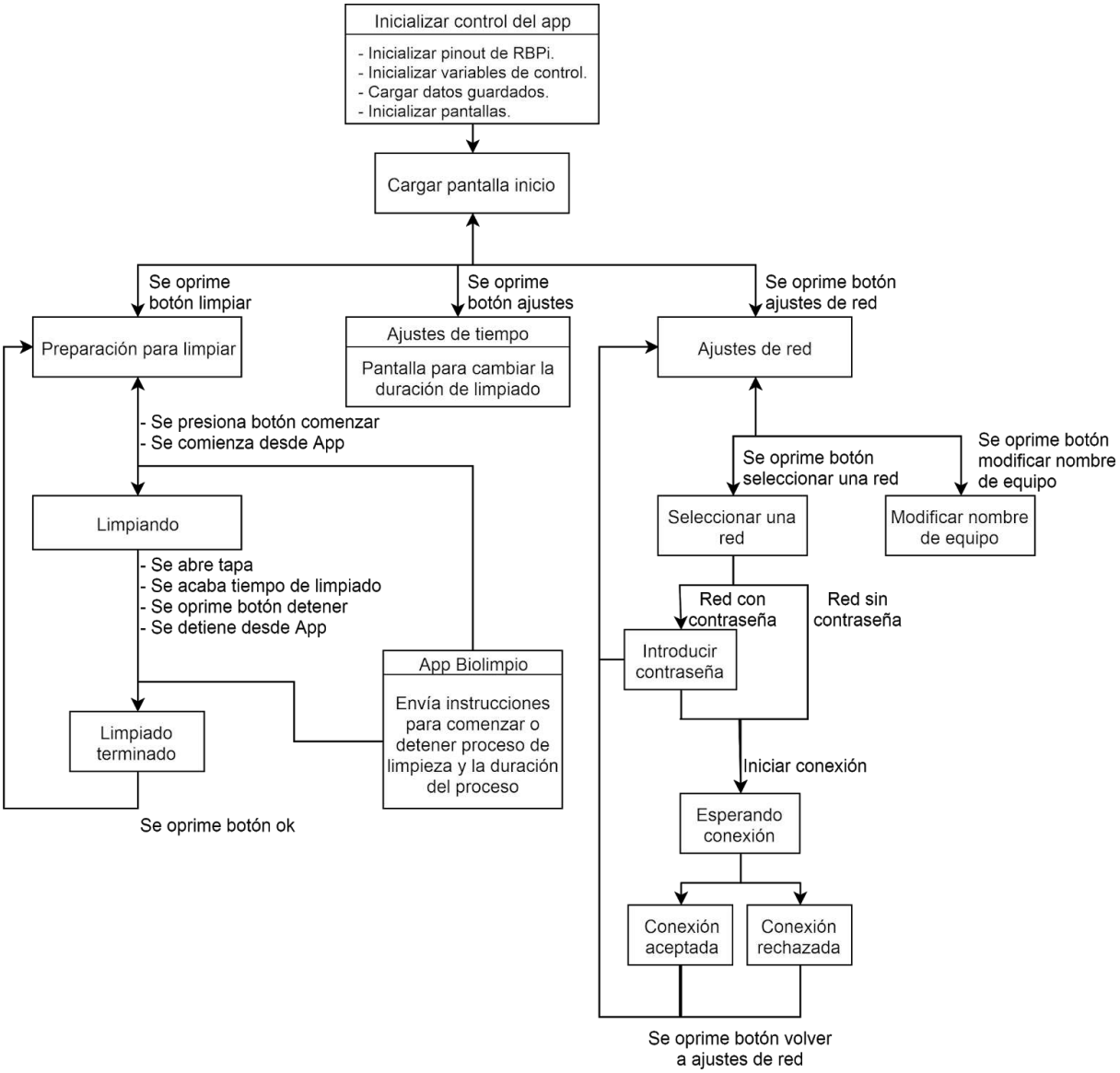


Desarrollo de software

Software

Back end

Funcionamiento general por diagrama de bloques



Descripción de archivos de programación

El software de control de Biolimpio consiste en 6 archivos: 4 programas escritos en Python y 2 archivos de texto donde se almacenan datos necesarios para el control. Los programas hechos en Python son: **app_control.py, server_pi.py, custom_tk.py y network_wpa.py.** Los archivos de texto se llaman: **data.txt y name.txt**.

app_control.py

Es el archivo principal del software de control. Aquí es donde se encuentran todos los métodos para inicializar y hacer funcionar la aplicación de control, así como la estructura de la interfaz gráfica.

Clases en app_control.py

app_control.py utiliza la herramienta Tkinter, la interacción entre el usuario y el control del equipo Biolimpio se realiza por medio de la clase BiolimpioApp, ahí se encuentran todos los métodos para el funcionamiento del dispositivo. El resto de las clases representan una pantalla diferente de la aplicación.

BiolimpioApp: la funcionalidad de la aplicación en sí y la interacción entre las pantallas.

Page_Start: pantalla de menú de inicio.

Page_Prepare: antes de comenzar a limpiar, indica los segundos de exposición a UV.

Page_Adjust: usada para cambiar el valor de la duración de la exposición a luz UV.

Page_Cleaning: se despliega en el momento en que encuentra limpiando el equipo. Muestra los segundos restantes para que termine el proceso de limpiado.

Page_Done: aparece cuando el proceso de limpiado a terminado.

Page_NetAdjust: se usa para configurar la conexión en red.

Page_WaitingConn: pantalla que temporalmente se despliega durante el intento de conexión en red.

Page_ConnAccepted: se despliega cuando la conexión en red se ha realizado exitosamente.

Page_ConnRejected: se despliega cuando hubo un error al intentar realizar la conexión a una red.

Page_NetSelect: muestra un menú con los nombres de todas las redes disponibles y permite elegir una.

Page_Password: muestra un teclado que permite introducir la contraseña para conectarse a una red.

Page_Name: muestra un teclado que permite introducir el nombre a asignar del equipo Biolimpio.

Variables de la clase BiolimpioApp

self.UVtime: guarda la duración en segundos de limpiado.

self.count: guarda el tiempo faltante para terminar una operación de limpieza.

self.count_rest: indica el tiempo que deben reposar los LEDs UV para no sobrecalentarse.

self.clean: indica si el equipo se encuentra limpiando.

self.clean_state: indica si los LEDs se encuentran encendidos o apagados, durante el proceso de limpieza

self.after_id: utilizada para marcar los segundos en una operación de limpieza.

self.wifiOn: indica si la conexión Wi-Fi es permitida.

self.waitingConn: indica si el equipo se encuentra esperando una conexión inalámbrica.

self.waitingConnCount: tiempo máximo para la espera de una respuesta en una red.

self.ssid: almacena el nombre de la red conectada al equipo.

self.ssid_connecting: almacena el nombre de la red a la cual se intentará conectar el equipo.

self.password_connecting: almacena la contraseña de la red a la cual se intentará conectar el equipo.

self.equipment_name: guarda el nombre asignado al equipo.

self.wireless_AP_list: guarda una lista con información de puntos de acceso disponibles.

self.ssid_list: guarda una lista con el nombre de red de los puntos de acceso disponibles.

self.device_IP: almacena la IP asignada al equipo conectado en red.

Métodos de la clase BiolimpioApp

def init (self, *args, **kwargs): utilizado para inicializar las variables y las pantallas de la aplicación.

def show_frame(self, page_name): sirve para cambiar las diferentes pantallas que se muestran al usuario.

def timeAdj(self, time): utilizado para cambiar el valor de self.UVtime y para actualizar todas las etiquetas donde aparece su valor.

def timeCorrect(self, time): por medio de este método se actualiza el valor de self.UVtime debido a la instrucción enviada desde la aplicación móvil.

def countdown(self, count): marca los segundos durante la operación de limpieza. También controla el encendido y apagado de los LEDs durante el proceso de limpieza para evitar sobrecalentamiento.

def checkData(self): este método revisa si hay instrucciones enviadas desde la aplicación móvil a través del archivo de texto data.txt. También revisa el estado actual de la red y controla el proceso mientras se espera una conexión en red inalámbrica. Este método se llama a sí mismo cada 2 segundos.

def checkData_15sec(self): similar al método checkData, se llama a sí mismo cada 15 segundos. Su función es únicamente actualizar las listas donde se almacena información de los puntos de acceso disponibles.

def network_selection(self): este método actualiza la pantalla para seleccionar una red.

def network_selected(self, x): se usa en cuanto el usuario ha seleccionado una red para iniciar un intento de conexión.

def startClean(self): inicia un proceso de limpieza.

def stopClean(self): finaliza un proceso de limpieza.

def button_callback(self, channel): se activa en cuanto la tapa de Biolimpio se abre en un proceso de limpieza para terminar con el proceso.

def connect_open(self): usado para conectar a redes que no tienen contraseña.

def connect_closed(self): usado para conectar a redes que tienen contraseña.

def initiate_connection_closed(self): hace aparecer la pantalla para introducir una contraseña y posteriormente intentar hacer una conexión a una red con la misma.

def initiate_connection(self): usado para identificar si la red a la que se quiere conectar tiene o no contraseña e iniciar un intento de conexión.

def wifi_checkBtn_pressed(self): permite activar o desactivar la conexión en red inalálmbrica del equipo.

def write_password(self, x): método utilizado cuando se teclea la contraseña para conectarse a una red.

def write_name(self, x): método utilizado cuando se teclea el nombre asignado al equipo por el usuario.

def write(self, x, keyboard): método utilizado para interactuar con las pantallas de introducir contraseña y asignar nombre a equipo.

def assignate_name(self): sirve para guardar el nombre asignado al archivo de texto name.txt.

def exitProgram(self): usado cuando el desarrollador desea salir de la aplicación. En el uso regular de la aplicación este método no debería ser llamado.

custom_tk.py

Este programa sirve como complemento para app_control.py. Son 2 clases almacenadas en otro archivo para tener mejor organización del software completo. La primera es KeyboardFrame, necesaria para ciertas pantallas que despliegan un teclado y la segunda es VerticalScrollFrame, necesaria para la pantalla donde se selecciona la red a la que se desea conectar el usuario.

Clase KeyboardFrame:

Esta clase es un teclado, con todos los caracteres necesarios para introducir nombres y contraseñas. El teclado incluye varios botones para cambiar entre el teclado numérico y el teclado alfabético, así como otros que contienen símbolos.

Métodos de KeyboardFrame

def init (self, parent, controller, keyboard): inicializa las variables para la operación del teclado.

def clear_frames(self): método utilizado para eliminar los botones que actualmente se despliegan, para después mostrar otros, dependiendo de si se selecciona un teclado numérico o alfabético.

def swap_case(self): método utilizado para cambiar de un teclado que despliega minúsculas a mayúsculas, o viceversa.

def swap_numeric(self): método usado para cambiar entre un teclado numérico y alfabético.

def swap_symbols(self): método usado para desplegar un teclado con símbolos.

def enter(self): cuando se presiona la tecla enter en el teclado, la pantalla cambia y el valor introducido es usado como contraseña o nombre del equipo, dependiendo del caso.

def letters_keyboard(self): se usa para llenar la pantalla con botones de un teclado alfabético.

def numeric_keyboard(self): se usa para llenar la pantalla con botones de un teclado numérico.

Clase VerticalScrolledFrame: es una clase que construye una lista con scroll compuesta de botones. Cada botón contiene el nombre de una red disponible para conexión. Esta lista se usa en la clase Page_NetSelect de app_control.py.

network_wpa.py

Este programa contiene un conjunto de métodos usados por app_control.py para efectuar conexiones Wi-Fi. Se separan del programa principal para mantener una mejor organización del software de la aplicación. La forma en que se configura la conexión inalámbrica es a través del archivo de configuración wpa_supplicant.conf.

Métodos de network_wpa.py

def clean_wpa_supplicant_conf(): elimina las líneas del archivo de configuración wpa_supplicant.conf para introducir nuevos datos de conexión como lo son el nombre de la red y la contraseña.

def append_wpa_supplicant_conf_psk(ssid, password): escribe en el archivo wpa_supplicant.conf los datos necesarios para establecer una conexión en red con contraseña, en el formato adecuado.

def append_wpa_supplicant_conf(ssid): escribe en el archivo wpa_supplicant.conf los datos necesarios para establecer una conexión en red sin contraseña, en el formato adecuado.

def restart_dhcpcd(): reinicia el servicio dhcpcd para que se carguen los nuevos datos de configuración de red y el equipo pueda realizar una nueva conexión.

def wlan0_network_request(ssid, password = None): utiliza todos los métodos descritos anteriormente para iniciar una conexión con una red Wi- Fi.

def disconnect_wlan0(): desactiva la conexión Wi-Fi.

def wpa_state_completed(): método utilizado para saber si se realizó exitosamente una conexión de red inalámbrica.

def connected_ssid(): método para conocer el ssid de la red a la que actualmente se encuentra conectado el equipo.

def IP_available(): devuelve un String conteniendo la dirección IP asignada. Devuelve “no conectado” si el equipo no se encuentra conectado a una red.

def IP_leased(): devuelve un valor booleano dependiendo de si una dirección IP fue asignada o no.

def device_IP(): devuelve un String conteniendo la dirección IP asignada. Devuelve “no conectado” si el equipo no se encuentra conectado a una red.

def scan_wireless_AP(): este método devuelve una lista de diccionarios. Cada diccionario representa una red inalámbrica disponible y contiene la siguiente información de la red: un valor numérico que representa la calidad de red, un booleano que representa si la red requiere de contraseña y el nombre de la red.

def connected_ssid(): este método regresa el nombre de la red a la que se encuentra conectado el equipo.

Señales de control

El programa de Python app_control.py interactúa con el driver de los LEDs UV por medio de dos señales de control. Una denominada EN, la cual habilita o deshabilita el paso de la corriente a través de los LEDs UV. La otra es DIM, la cual provee una señal PWM de 50 HZ, con la cual se reduce la intensidad de iluminación y la cantidad de calor generada por los LEDs UV.

APIs y Endpoints de conexión con App de control Android

server_pi.py

Este programa de python es un servidor el cual recibe las instrucciones de la aplicación móvil y las almacena en un archivo de texto: data.txt. Posteriormente la aplicación del programa app_control.py revisa periódicamente si han llegado nuevas instrucciones para ejecutar, desde ese archivo. Las instrucciones se almacenan en forma de json con dos valores: encendido y duración. El primero representa si el equipo debería encender o apagar los LEDs UV y el segundo representa la duración del encendido.

Métodos de server_pi.py

def init (self): inicializa el servidor, utiliza la ip asignada a la raspberry Pi y escucha en el puerto 8000.

def clientthread(self, conn): recibe la información del cliente, la procesa para que todos los datos puedan ser bien interpretados y la almacena en el archivo de texto data.txt en forma de json.

def accept_conn(self): espera para aceptar la conexión de un dispositivo móvil cliente. Invoca el método clientthread cuando se inicia una conexión.

def close_socket(self): sirve para cerrar el servidor. No es utilizado en el uso regular de la aplicación.

Software

Front end

La interfaz de usuario se construye a partir de un contenedor de frames (pantallas) las cuales se despliegan dependiendo de la interacción con el usuario y el estado del equipo. Estos frames se construyen por medio de la librería de Python Tkinter. Cada pantalla se construye a partir de una clase contenida en el programa app_control.py e instanciada desde la aplicación. El formato de cada frame es muy similar uno con el otro.

Variables utilizadas en cada frame (pantalla)

helv36: esta variable almacena el formato necesario para las etiquetas visibles.

self.controller: esta variable permite utilizar los métodos de la clase BiolimpioApp

self.frame: cada pantalla coloca todos sus elementos llamados Widgets dentro de este contenedor invisible.

Button1: por lo general cada pantalla cuenta con al menos uno de estos botones para el control de la aplicación.

Label1: toda información o datos son desplegados en etiquetas, las cuales pueden modificar su valor.

Métodos típicos utilizados en cada frame (pantalla)

frame.pack(): es un método para cargar cualquier widget a un frame en cierto orden.

button1.config(): permite modificar parámetros de los botones, como su formato, dimensión o la acción a realizar cuando es presionado.

label1.config(): permite modificar parámetros de las etiquetas tales como su contenido.

button1.grid(): es otra forma de cargar cualquier widget a un frame, pero de forma organizada en celdas. No es posible combinar pack y grid en un mismo frame.

Software

Manual de usuario

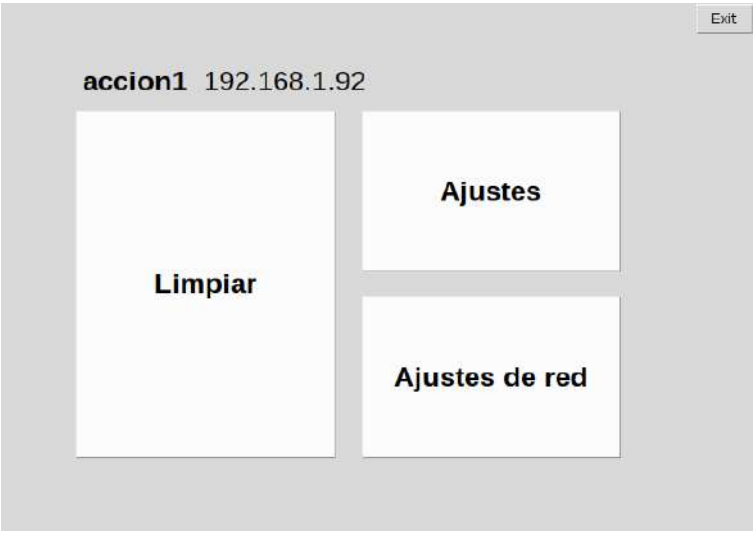
Encender el equipo y generalidades

En cuanto Biolimpio se conecta a la toma de corriente, **la pantalla del equipo comenzará a cargar la aplicación.** Después de unos segundos en la pantalla **se visualizarán los siguientes botones: Limpiar, Ajustes y Ajustes de red.**

El **botón Limpiar** dirige a otra pantalla para preparar un proceso de limpiado.

El **botón Ajustes** nos lleva a una pantalla donde podemos configurar la duración de exposición a la luz UV.

Y el tercer botón, **Ajustes de red** despliega otro menú que permite realizar conexiones a una red Wi-Fi.



Pantalla menú principal

En la parte superior de la pantalla, justo arriba de los botones, se visualizan dos etiquetas. La primera corresponde al nombre asignado al equipo, la segunda a la dirección IP asignada dentro de una red inalámbrica. Esta última es útil para saber si el equipo se encuentra conectado o no en red. En caso de no ser así, la etiqueta desplegará “no conectado”.

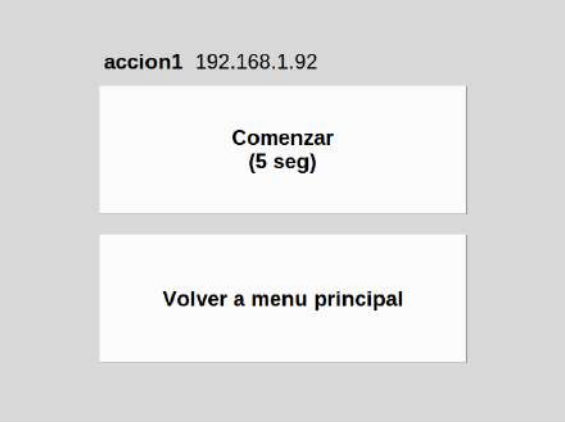
Proceso de limpieza

Al oprimir el botón limpiar pasamos a la pantalla preparar limpieza. Esta pantalla consiste en dos botones.

> **Con el primer botón comenzamos un proceso de limpieza**, sin embargo, este botón **no funciona a menos que la tapa del equipo se encuentre cerrada.**

De este modo evitamos que el usuario se exponga a la luz UV. Este botón también indica el tiempo en segundos de exposición a la luz UV.

El segundo botón sirve para regresar al menú principal.

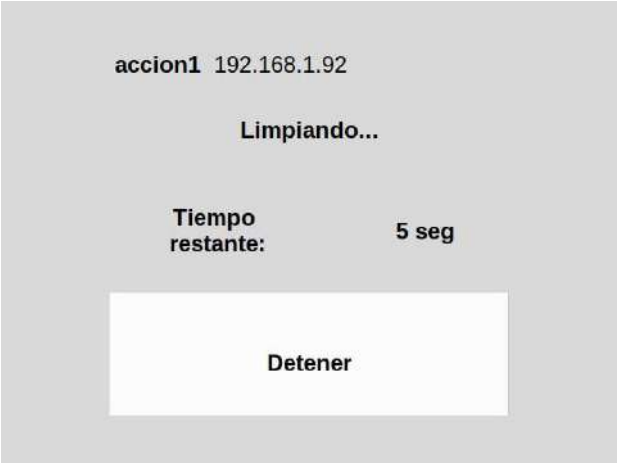


Pantalla preparar limpieza

En cuanto el botón Comenzar es presionado, la pantalla cambia a otra donde se puede **visualizar el tiempo faltante para que termine el proceso de limpieza**. Es de notar que este tiempo no es el mismo que se despliega en el botón comenzar. Esto es debido a que durante el proceso de limpieza los LEDs UV se encenderán y apagarán periódicamente para evitar un sobrecalentamiento del sistema.

Este cambio es visible en la etiqueta superior que dice “Limpiando...” cuando los LEDs se encuentran encendidos y “LEDs UV en reposo" cuando se encuentran apagados.

De este modo, **el tiempo desplegado en el botón Comenzar se refiere al tiempo en el cual los LEDs se mantienen encendidos y el tiempo desplegado en la pantalla durante el proceso de limpieza se refiere al tiempo total restante**, incluyendo los periodos en los cuales los LEDs se encuentran apagados.



Pantalla durante el proceso de limpieza

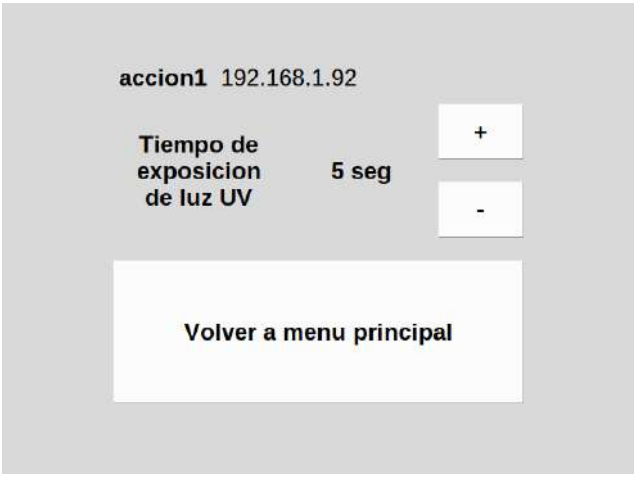
La pantalla durante el proceso de limpieza también posee un botón que sirve para **detener el proceso manualmente**. El proceso puede ser detenido de 4 maneras:

El tiempo del proceso se ha terminado
El usuario ha presionado el botón Detener
El usuario ha abierto la tapa del equipo
El usuario ha mandado la instrucción de Detener a través de la aplicación móvil

En cualquiera de los 4 casos, al momento se despliega la pantalla limpieza terminada, la cual solo sirve para corroborar que el proceso ha terminado. Esta pantalla contiene un botón para regresar a la pantalla preparación limpieza.

Pantalla limpiado terminado

Cuando se quiere **cambiar la duración de exposición de luz UV**. Se accede a la pantalla **ajustes** desde el menú principal. Esta pantalla despliega una etiqueta con el valor actual. Por default cada vez que se enciende Biolimpio, este valor es 5 seg. A lado de esta etiqueta hay unos **botones para aumentar o disminuir este valor**. Para salir de esta pantalla se oprime el botón Regresar a menú principal.

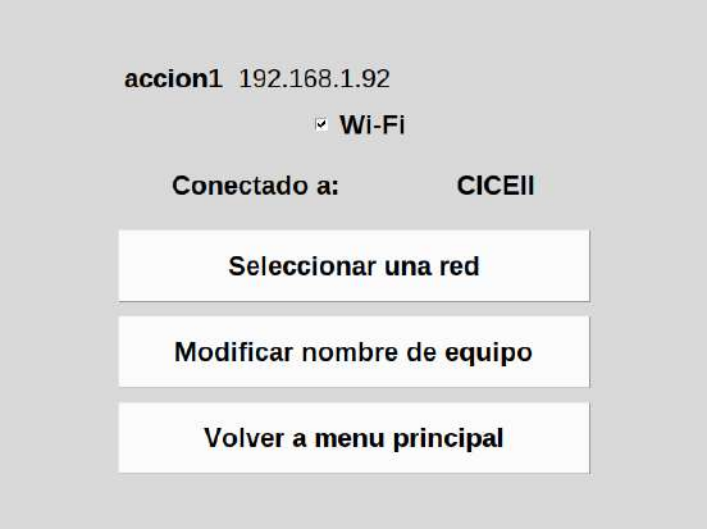


Pantalla ajustes

Conexión a la red

Para poder utilizar la aplicación Biolimpio, primero es **necesario conectar el equipo a una red inalámbrica**. Para hacer esto se oprime el botón **Ajustes de red**, desde el menú principal. En seguida aparecerá una pantalla con varias etiquetas y botones. De arriba hacia abajo, estas etiquetas y botones representan:

1. El nombre y la dirección IP del equipo
2. Un botón checklist que activa o desactiva la conexión inalámbrica
3. Una etiqueta que despliega el nombre de la red conectada al equipo.
4. Un botón para seleccionar una red inalámbrica disponible.
5. Un botón para modificar el nombre del equipo.
6. Un botón para regresar al menú principal.



Pantalla ajustes de red

El **botón checklist** sirve para inmediatamente activar o desactivar la conexión en red inalámbrica. Si se desactiva veremos el efecto en las demás etiquetas. Si deseamos conectar el equipo a una red, lo primero que hay que hacer es cerciorarse de que el botón checklist esté activado, después presionamos el **botón Seleccionar una red.** Enseguida aparecerá una pantalla con una lista de botones, cada uno con el nombre de una red inalámbrica disponible.

Es posible mover la lista hacia abajo o hacia arriba, deslizando el scroll que se encuentra a la derecha. Al presionar cualquier botón inmediatamente comenzará el proceso para conectarse en red. Si se desea volver al menú de ajustes en red solo hay que presionar el botón que se encuentra en la parte inferior.

Pantalla seleccionar red

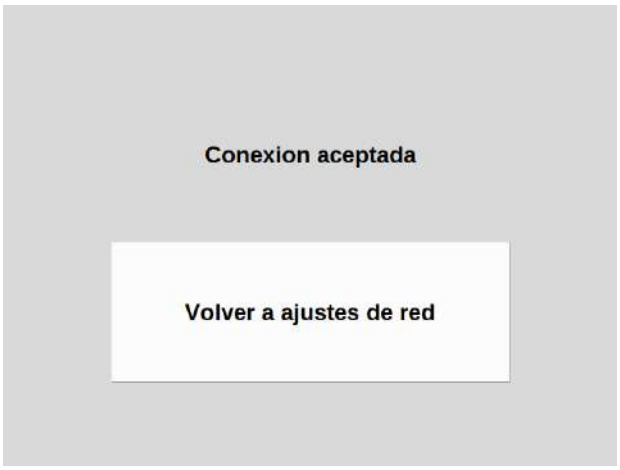
Al seleccionar una red, la aplicación despliega una pantalla dependiendo de si la red requiere una contraseña o no. Si fuera así, la pantalla siguiente muestra un teclado, con el cual podemos introducir la contraseña. Este teclado contiene todos los caracteres necesarios, si es necesario regresar al menú de ajustes de red, simplemente presionamos el botón en la parte inferior de la pantalla. Una vez que la contraseña ha sido introducida, se presiona el botón Enter del teclado y enseguida comenzará el proceso de conexión. Para lo cual la pantalla simplemente desplegará una etiqueta que dice “Esperando conexión de…”.

Pantalla introducir contraseña

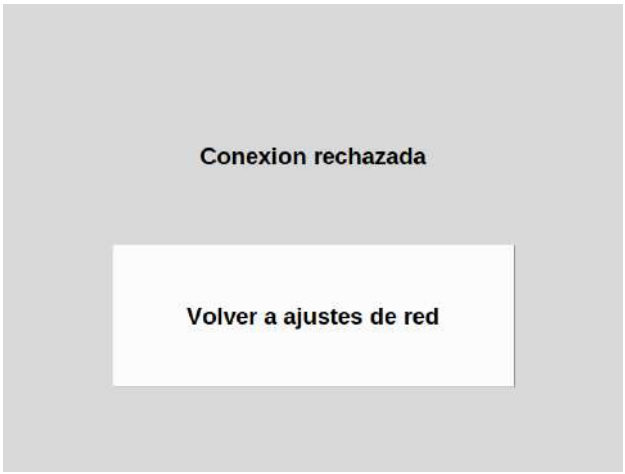
Pantalla esperando conexión

Si la conexión es exitosa, en seguida se despliega la pantalla “Conexión aceptada”. De

ahí podemos volver al menú de ajustes de red oprimiendo el único botón visible. Si la conexión no fue exitosa, aparece una pantalla con el mensaje “Conexión rechazada”. De igual forma podemos regresar al menú de ajustes de red oprimiendo el único botón desplegado.



Pantalla conexión aceptada



Pantalla conexión rechazada

Después de haber conectado el equipo exitosamente, el nombre de la red será

visible en el menú de ajustes de red. Así mismo, las etiquetas que aparecen en la parte superior de varias pantallas desplegarán una dirección IP, la cual es necesaria para poder empezar procesos de limpieza o detenerlos desde la aplicación móvil.

Otra de las herramientas encontradas en el menú de ajustes de red, es la opción para cambiar el nombre del equipo. Realmente esta herramienta no tiene ningún efecto en la funcionalidad del equipo actualmente. Pero bien podría ser utilizada para posteriormente controlar distintos equipos Biolimpio en una misma red o para encontrar automáticamente desde la aplicación móvil un equipo Biolimpio.

Pantalla asignar nombre

Software

App de control

La aplicación móvil de Biolimpio está disponible para equipos Android. El código de la aplicación está basado en Java y fue hecho en Android Studio.

El proyecto de Android se compone principalmente de 3 actividades: **MainActivity.java, SecondActivity.java y ThirdActivity.java correspondientes a las 3 pantallas de la aplicación.** Así mismo cada actividad está asociada a un archivo xml: activity_main.xml, activity_second.xml y activity_third.xml.

La primera pantalla es del menú principal, donde se muestra la dirección IP del equipo al que se va a controlar y dos botones, uno para comenzar un proceso de limpiado y otro para asignar otra dirección IP.

La segunda actividad sirve para enviar las instrucciones de encendido y apagado de los LEDs UV. En esta pantalla también se despliega la dirección IP del equipo al que se va a controlar, así como un espacio para introducir un valor numérico, el cual representa la duración en segundos de la exposición a la luz ultravioleta. En seguida se muestran dos botones, comenzar y detener que sirven para controlar el proceso de limpieza.

La tercera actividad sirve para que el usuario introduzca la dirección IP asignada al equipo Biolimpio al momento de conectar éste en red. Es posible visualizar esta dirección en la parte superior de la pantalla del equipo Biolimpio. Una vez introducida la dirección se oprime el botón asignar para que la aplicación guarde este valor y las instrucciones sean enviadas al dispositivo correcto.

Variables utilizadas en la aplicación

public String message: variable donde se guarda el mensaje, con las instrucciones en un String.

public JSONObject obj: variable donde se guarda el mensaje, con las instrucciones en un JSON.

public String IP_equipo: la dirección IP a la que se debe enviar el mensaje. Sockets: socket creado para establecer una conexión TCP con el servidor en el equipo Biolimpio.

Métodos utilizados en la aplicación

protected void onCreate(Bundle savedInstanceState): método utilizado para crear cada Actividad. Lo que equivale a crear cada pantalla.

clickable.setOnClickListener(new View.OnClickListener(): método utilizado para agregar una acción al oprimir un botón.

Protected Void doInBackground(Void...params): utilizado para realizar el proceso de envío de información al equipo Biolimpio.

Conexión con Backend RBPi

La app Biolimpio se comunica con el equipo por medio de una conexión TCP realizada a través de un socket. En ambas partes de la comunicación se utiliza un socket aunado a la dirección IP del respectivo equipo, y el servidor escucha a través del puerto 8000. Para realizar esta conexión, la aplicación cliente envía un paquete de datos a la dirección IP introducida por el usuario. Este paquete contiene un JSON con dos valores: encendido, el cual es un booleano que indica si el equipo debería comenzar o terminar un proceso de limpieza y un número el cual representa la duración del proceso de limpieza.

La comunicación entre la aplicación cliente y el servidor del equipo Biolimpio solo se puede realizar en una red a nivel local.

App

Manual de uso

La app de Biolimpio es muy sencilla de usar, al iniciar la aplicación la primera pantalla visible será la del **menú principal**. En ella se encontrarán dos botones, y una etiqueta con la funcionalidad de mostrar la dirección IP a la cual se enviarán las instrucciones.

El primer **botón "Limpiar"** despliega la pantalla para enviar instrucciones.
El **botón "Asignar equipo"** despliega otra pantalla para introducir otra dirección IP. Si ninguna dirección IP ha sido introducida la etiqueta de dirección no mostrará nada.

Pantalla de la app Biolimpio menú principal

Pantalla de la app Biolimpio introducir dirección IP

Al ingresar a la pantalla introducir dirección IP, nos aparecerá una etiqueta que permite introducir a través de un teclado la dirección IP del equipo. Esta dirección es visible en la pantalla del equipo Biolimpio en la parte superior a la derecha del nombre del equipo.

Es importante introducir la dirección tal cual se despliega en la pantalla, incluidos los cuatro puntos. Si en lugar de una dirección IP hay un mensaje que dice “no conectado”, esto quiere decir que nuestro equipo no se encuentra conectado a ninguna red.

Por lo tanto, hay que proceder a conectar el equipo Biolimpio primero, y posteriormente podremos introducir la dirección IP a la app Biolimpio. **Tanto el equipo Biolimpio como el dispositivo móvil tienen que estar conectados a la misma red local inalámbrica.** Después de haber introducido la dirección IP en la app Biolimpio, ésta será visible en el resto de las pantallas de la aplicación.

Ahora podemos ingresar a la pantalla **enviar instrucciones,** la cual contiene dos botones y otra entrada numérica.

En la **entrada numérica podemos ingresar cualquier valor que representa la duración en segundos** del proceso de limpieza.

El **botón Limpiar** comienza el proceso de limpieza en el equipo Biolimpio

El **botón Detener**, detiene el proceso de limpieza si es que se ha iniciado uno.

Pantalla de la app Biolimpio enviar instrucciones

Conclusiones del proyecto

El proyecto está concluido según los alcances. El prototipo 0 es una materialización de una idea, la primera aproximación de lo que se vislumbra que un producto podría llegar a ser, de esta forma, se llega a un primer diseño que puede tomarse como referencia para futuras implementaciones.

Este prototipo, si bien no ha sido verificado y tampoco puede utilizarse para fabricarse en masa, provee de información muy útil de factibilidad y posibles diseños que pudieran utilizarse para siguientes etapas. Los siguientes pasos para este prototipo es el de su verificación funcional, para obtener información de que tan alejado está el funcionamiento del dispositivo actual con el requerido en el futuro.

Siguiente de esto, necesita buscarse una compañía especializada que se encargue del nuevo diseño, planeación, certificación y manufactura de una primera versión de un producto fabricable, debido a que los siguientes pasos
requieren de especialistas en la materia.

En general, el proyecto ha sido un éxito según lo que se buscaba de él, el equipo de trabajo de CICEII y Pathways pusieron en práctica lo mejor de sus capacidades para desarrollar lo que hoy podemos llamar el primer prototipo de “Biolimpio”, un esterilizador automático de uso cotidiano. Se agradece todo el apoyo brindado y la confianza de emprender proyectos de innovación para mejora de la sociedad y la vida diaria.