**Jonathan Betancur Espinosa**

**Esteban Vergara Giraldo**

[**https://github.com/QuitoTactico/Proyecto-CortinasInteligentes**](https://github.com/QuitoTactico/Proyecto-CortinasInteligentes/tree/main)

[**https://cortinas-inteligentes.anvil.app/**](https://cortinas-inteligentes.anvil.app/)

Cortinas Inteligentes v2.0 - Manual de Fabricación

[**Definición del proyecto 1**](#_kal245fu479b)

[**Paso 1: Descargar Thonny IDE 2**](#_w97cgntcbyfw)

[**Paso 2: Configurar el Raspberry 3**](#_1axxwdjhqcz0)

[**Paso 3: Crear un proyecto en Anvil works 4**](#_bboe3w47xllt)

[**Paso 4: Crear nuestra interfaz 6**](#_bc54yhwr279u)

[**Paso 5: Conectar los sensores y leds al Raspberry 9**](#_p7le7yv4qvi4)

[**Paso 6: Crear funciones en Raspberry 10**](#_8cvpvk8ie3nn)

[**Paso 7: Realizar llamados desde Anvil Works 13**](#_cmfulb2vywut)

[**Paso 8: Recibir datos y graficarlos 14**](#_48b1bolzc33y)

[**Producto final (v1.0) 17**](#_e86wl2sngg7j)

[**Segunda entrega (v2.0, contenido adicional) 18**](#_y5f2ixbogzi3)

[**Frenar toma de datos 18**](#_8e468zfql9e6)

[**Conversor 20**](#_r7ewkc5dvm6z)

[**Consola 21**](#_bwarv7ktbixr)

[**Producto final (v2.0) 22**](#_j8hkkrp211yk)

# Definición del proyecto

El proyecto sobre el cual se va a documentar se va a tratar sobre unas **cortinas inteligentes, que se abren o se cierran dependiendo cuanta iluminación haya en el exterior**, para que el usuario no tenga que pararse a cerrar la cortina si hace mucho sol, o abrirlas para que entre luz (ya que afuera está oscuro).

Teniendo en cuenta que **cuando es de noche, estas permanecerán cerradas** para más privacidad.

**El sistema podrá encenderse o apagarse**. Al estar apagadas, las cortinas funcionarán como cualquier cortina normal, quedándose estáticas.

Para mostrar y subir estos datos, se utilizará la plataforma Anvil Works, la cual es la encargada de proporcionar las visualizaciones en la nube correspondientes compatibles con el Raspberry pico, ya que este solo trabaja con el lenguaje python.

**Unas funcionalidades extra** que se agregaron en el contenido adicional, son: Frenar toma de datos, Conversor Luz → Apertura, y un visualizador para los mensajes de consola.

Por lo tanto, para simular esto, se utilizarán varias entradas y salidas digitales (binarias), y análogas (números y mediciones contínuas), de esta forma:

1- **Las 3 entradas digitales** desde Anvil Works serán 3 botones: el primero para simular el día o la noche, el segundo para apagar o encender el funcionamiento de la cortina automática, y el tercero es el botón para frenar la toma de datos (contenido adicional).

2- **La entrada análoga** será la luz del sol simulada con un potenciómetro que vaya variando los valores a criterio propio.

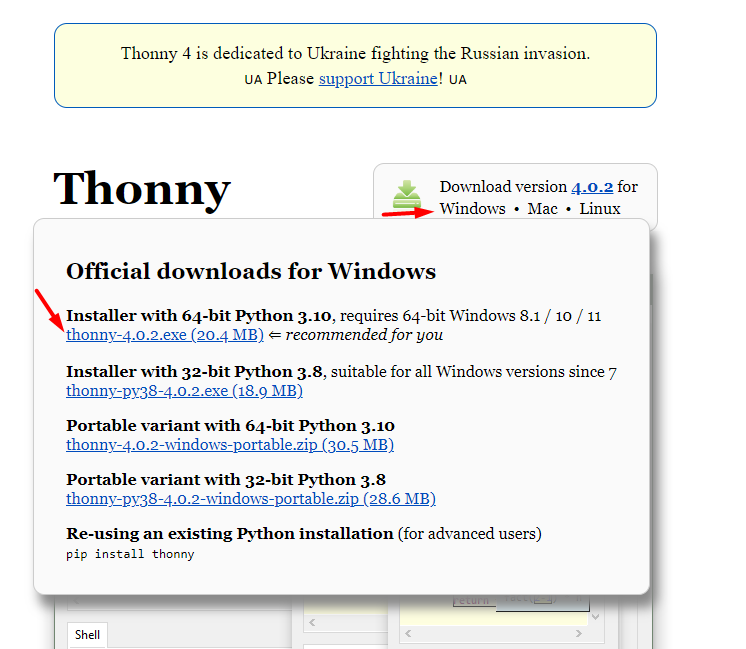
3- **Las 4 salidas digitales**, serán 3 leds, un led que se ilumine si está de día, y otro que se ilumine si está de noche. Esta información también se le mandará a anvil works, junto a si el sistema está encendido o apagado. El led integrado del raspberry estará iluminado siempre y cuando el sistema esté encendido. En el contenido adicional, **agregamos un cuarto led**, que estará encendido si se están tomando datos de luz, y se apaga cuando no.

4- **La salida análoga**, será acerca de qué tan abierta está la cortina, siendo 0% el valor que se da cuando esta está cerrada, y 100% cuando la cortina está abierta en su totalidad. Este valor depende de la cantidad de luz ambiente que esté siendo recibida por la entrada análoga (potenciómetro), y si es de día o de noche, o está encendido o apagado el sistema (lo cual es determinado por las entradas digitales, desde anvil). Si el sistema está apagado, las cortinas no cambiarán su estado de acuerdo al sol, en cambio, sólo será regulado por el botón de abrir/cerrar.

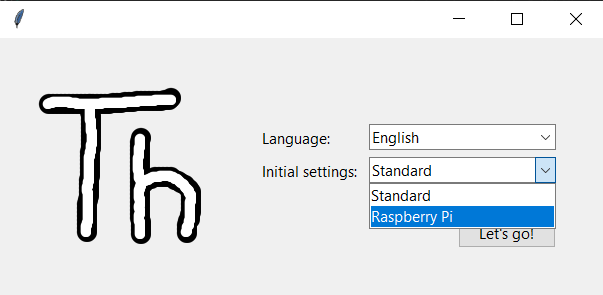
# Paso 1: Descargar Thonny IDE

Necesitamos esta aplicación para trabajar en el código de nuestro Raspberry, así que la descargaremos.

Nos remitimos a la página oficial del IDE: <https://thonny.org/>, allí encontraremos las opciones para los diferentes sistemas operativos, en este caso, windows



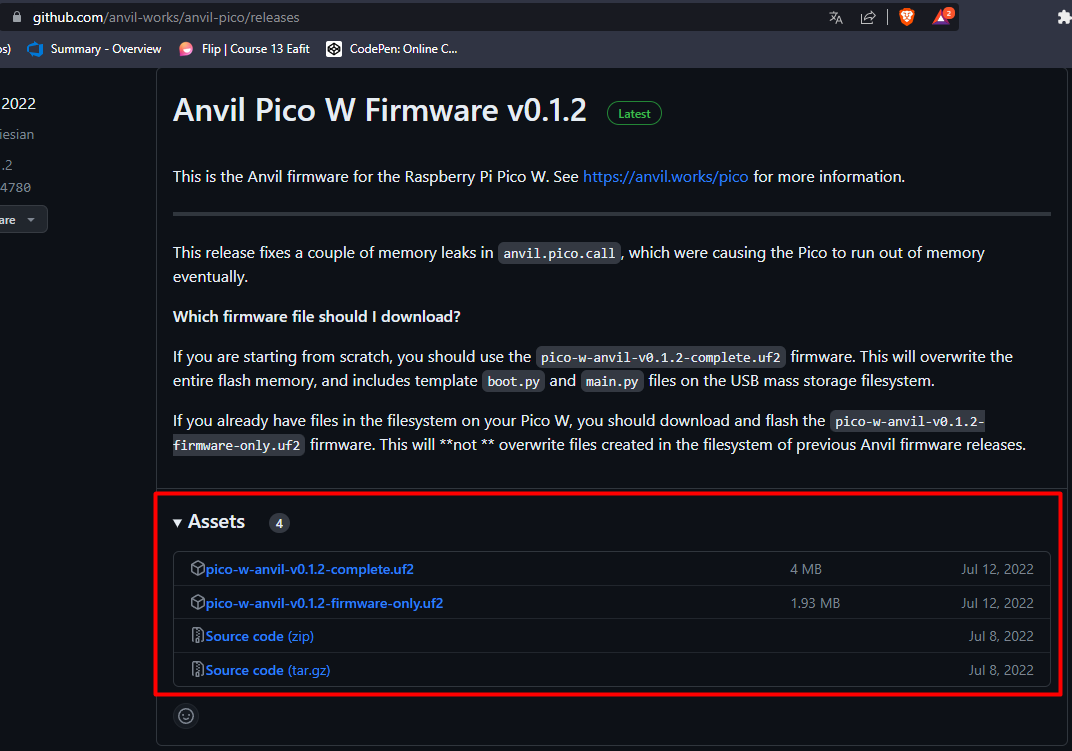
Cuando ejecutemos el archivo, se instalará el programa, y posteriormente este se podrá abrir, a lo que elegimos la opción de abrirlo como Raspberry Pi

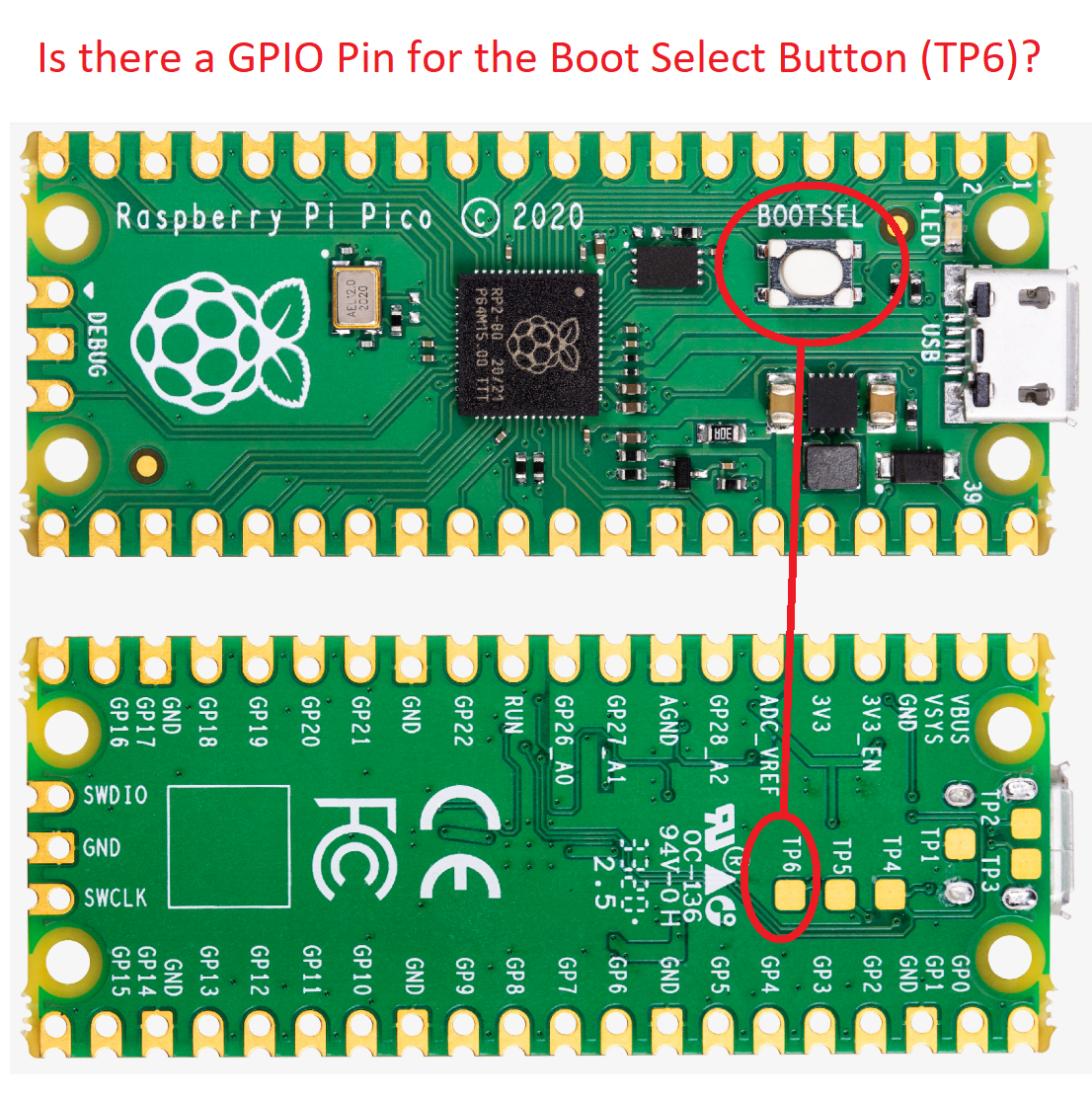


# Paso 2: Configurar el Raspberry

Para que nuestro Raspberry se pueda conectar a Anvil Works, necesitamos instalarle un firmware especial.

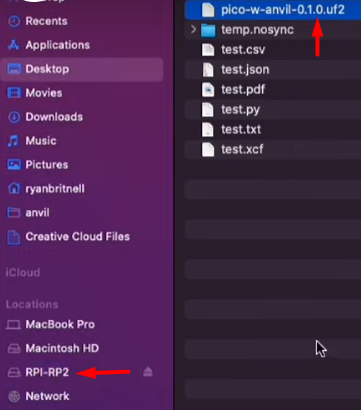
Para ello, debemos ir a <https://github.com/anvil-works/anvil-pico/releases/tag/v0.1.2> y descargar el primer archivo “pico-w-anvil-v0.1.2-complete.uf2”





Una vez descargado el archivo, presionamos el botón “boot” de nuestro Raspberry, al mismo tiempo que lo conectamos al ordenador. Esto hará que el PC lo reconozca como un pendrive y podremos introducir archivos en él.

Copiamos el archivo .uf2 que descargamos al raspberry, esto hará que comience el proceso de instalación



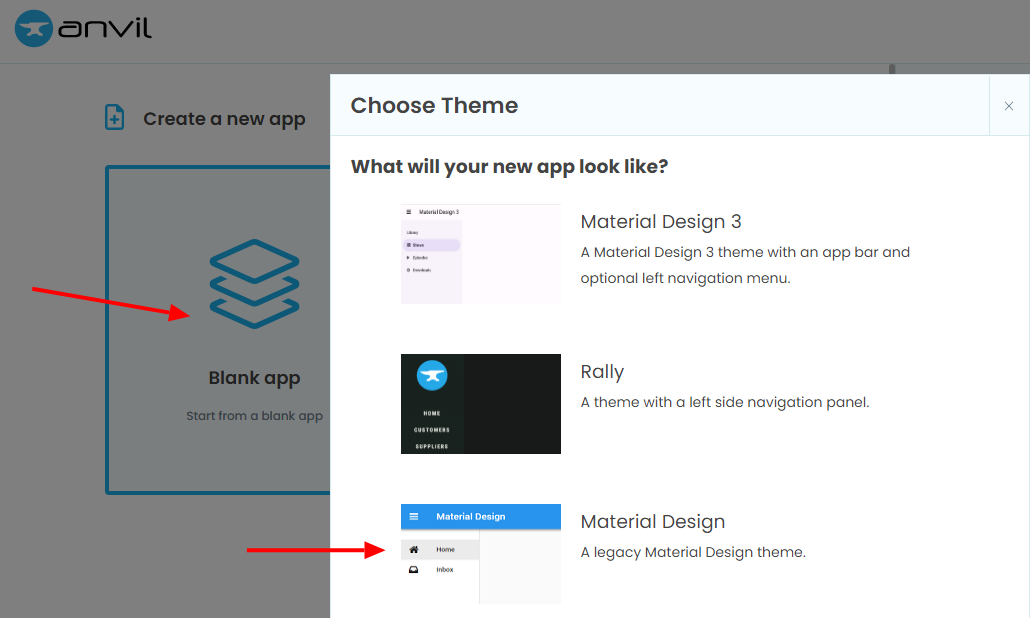
Se instalarán en el dispositivo dos archivos, **boot.py** y **main.py**, que nos ayudarán a lo largo de nuestro proyecto.

Finalmente, desconectamos el dispositivo y lo volvemos a conectar.

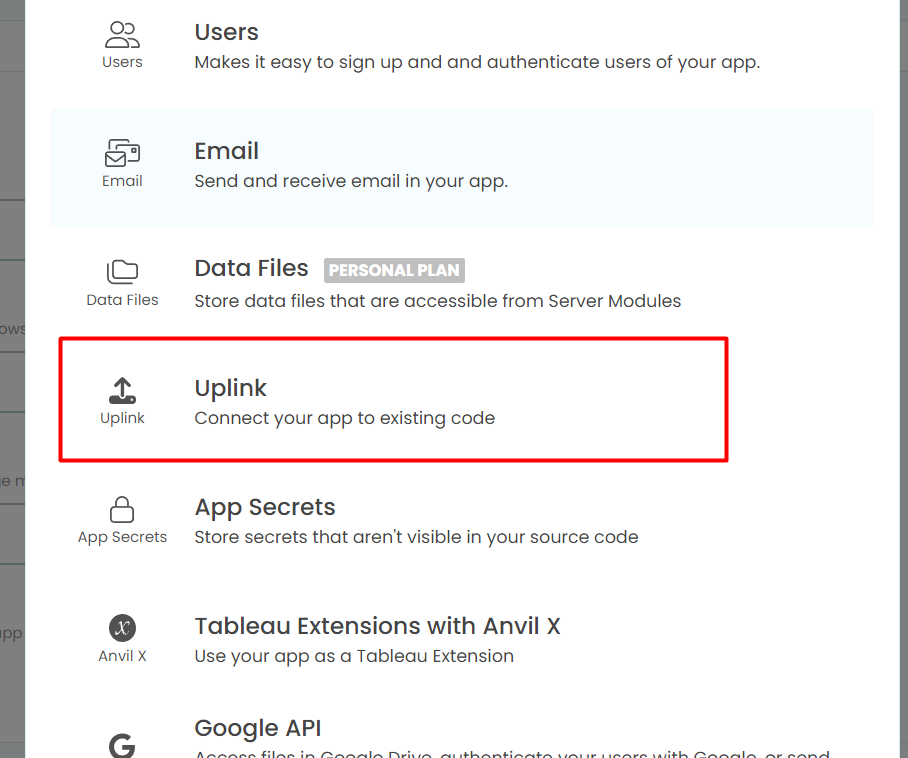
# Paso 3: Crear un proyecto en Anvil works

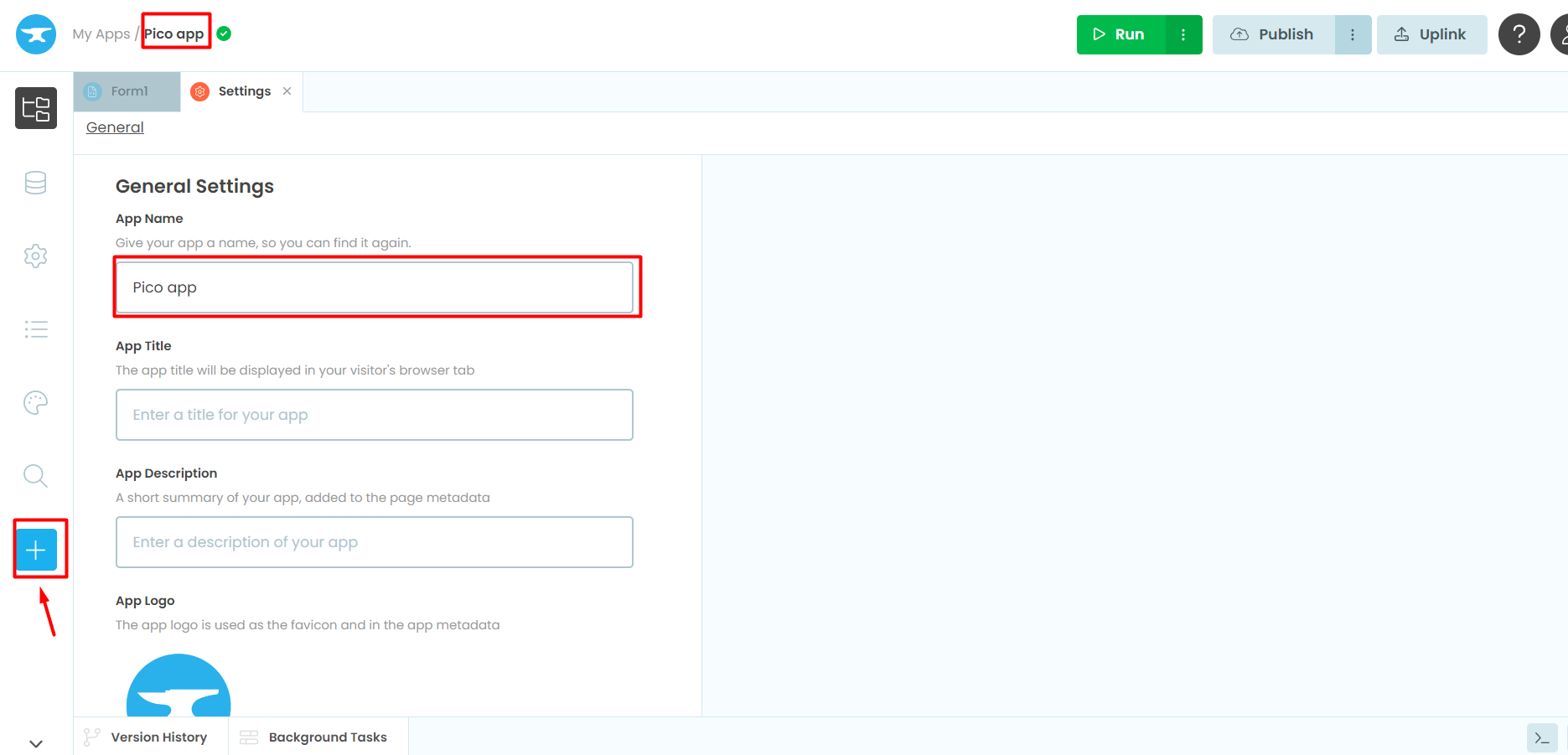
Primero necesitas crear una cuenta de Anvil Works para comenzar a crear proyectos, puedes crearla con tu cuenta de google si así lo deseas.

Crea una nueva app:

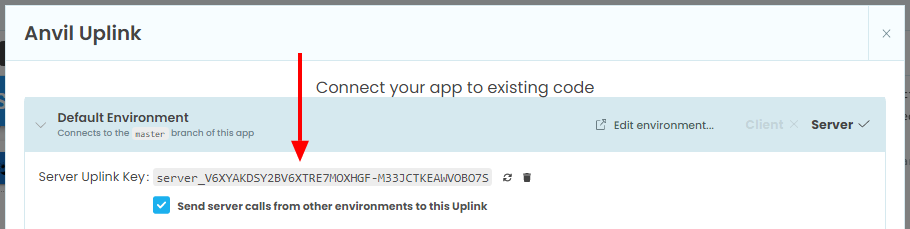


Al estar en la app, desde ajustes le cambiamos el nombre, y luego clickeamos este botón azul para añadir una conexión uplink con la que conectaremos nuestro dispositivo Raspberry.





Copiaremos el código que nos dan.



Y lo pegamos en esta sección del main.py de nuestro Raspberry

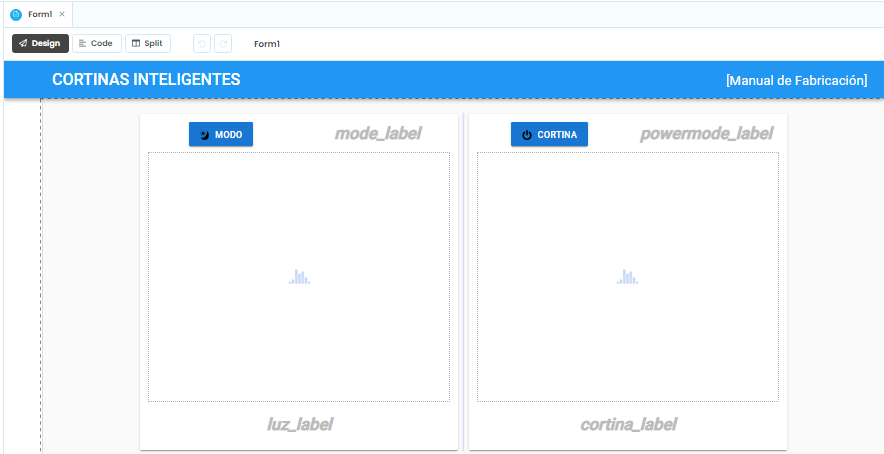


# Paso 4: Crear nuestra interfaz

Una vez configurada la app y los archivos junto con el Raspberry, procedemos a crear nuestra interfaz.

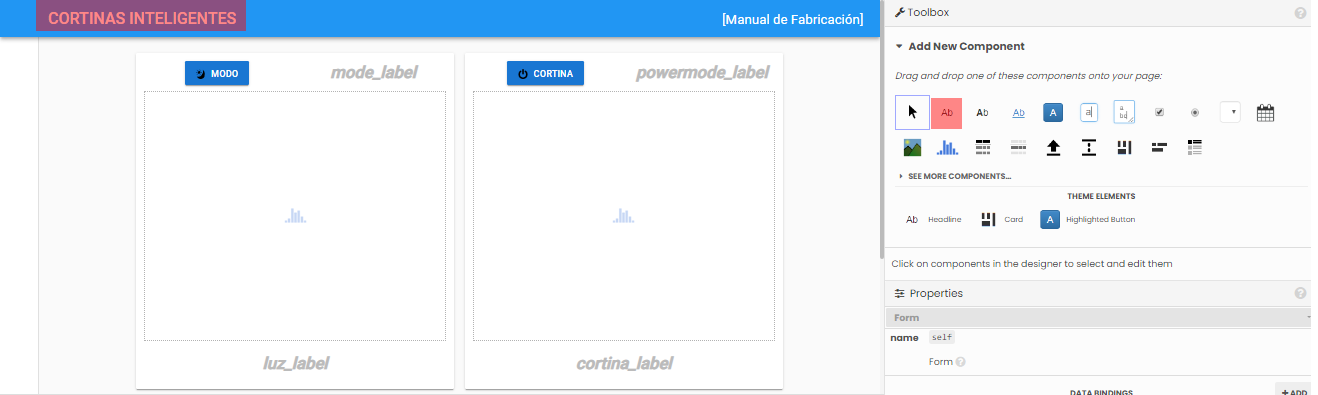
Los componentes pueden ser seleccionados desde la barra de herramientas, y se agregan al ser arrastrados hacia el diagrama.

Nuestra interfaz se verá de esta forma:

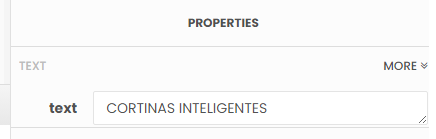


Contiene:

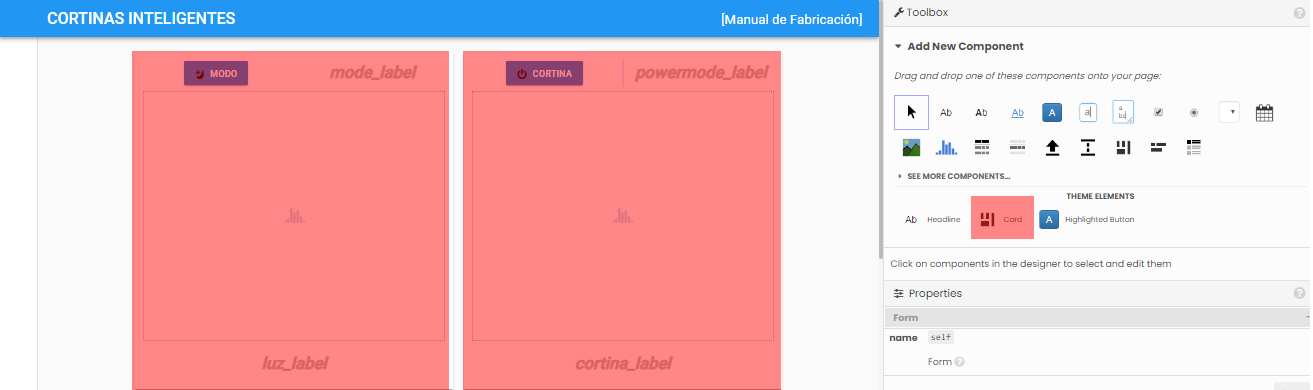
* Un título



Puedes modificar el texto desde sus propiedades



* Dos componentes “Card”

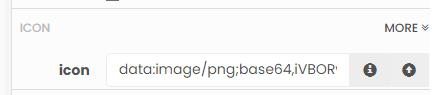


Solo sirven para almacenar a otros componentes, lo usaremos para separar la información en estos dos bloques.

* Cada uno con un botón, al que le agregamos un ícono



Le modificamos los títulos desde propiedades para ponerles “Modo” y “Cortina”, y le agregamos un ícono en .png desde aquí:



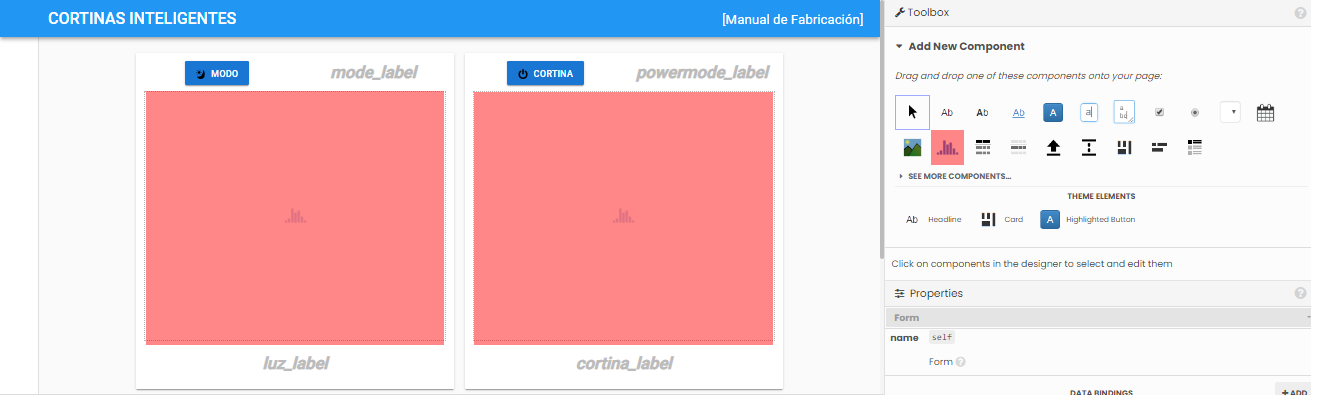
* Cada uno con un “label” o recuadro de texto superior



Para poder referirnos a ellos más tarde, los llamaremos mode\_label y powermode\_label desde aquí:

****

* Cada uno con un “plot”, o gráfica



Cuyos nombres serán “plot\_luz” y “plot\_cortina”





* Y un label inferior para cada uno.

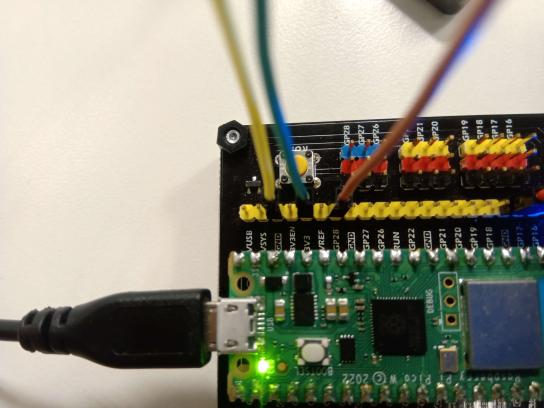


Cuyos nombres serán luz\_label y cortina\_label.

# Paso 5: Conectar los sensores y leds al Raspberry

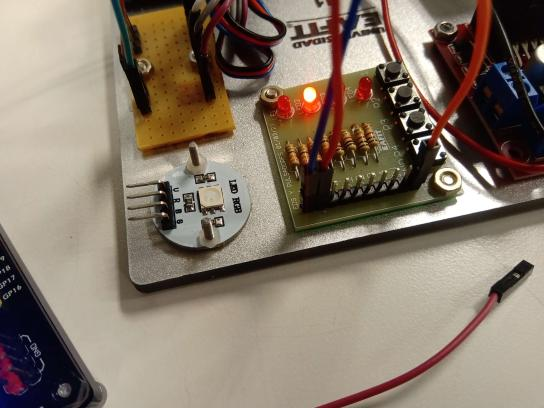
Conexión del potenciómetro:

* amarillo - GND a la izquerda
* verde - 3V3 a la derecha
* rojo - GP28 al medio



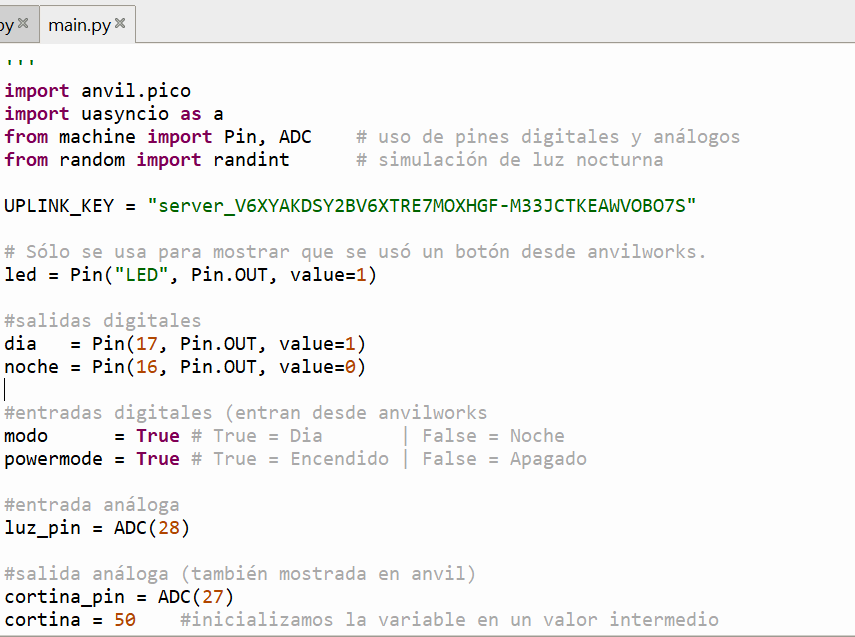
Conexión de leds:

* naranja - GND a GND
* azul - GP17 a L1
* rojo - GP16 a L2

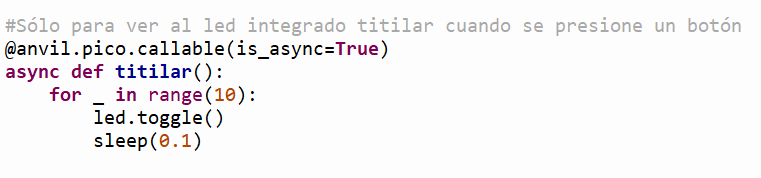


# Paso 6: Crear funciones en Raspberry

Con el programa Thonny abierto y configurado, se proceden a realizar los llamados de las librerías y los pines de entrada o salida necesarios

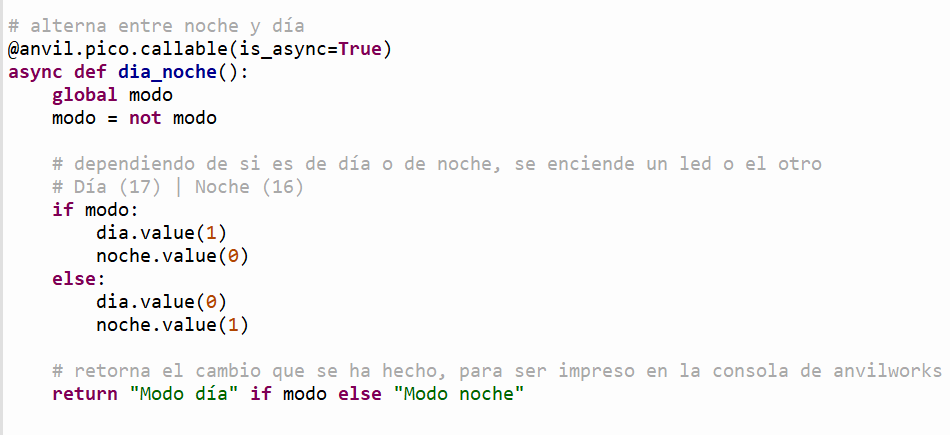


Después, procedemos a crear las funciones correspondientes que se van a utilizar, para este caso:

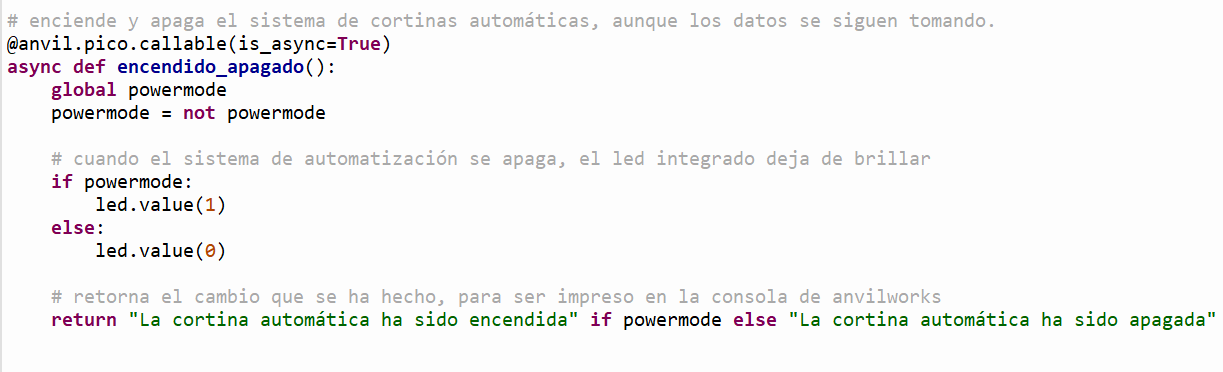


Esta función servirá para indicar que se hizo un cambio en el Raspberry y por tanto, el led parpadeará 5 veces indicando ese cambio

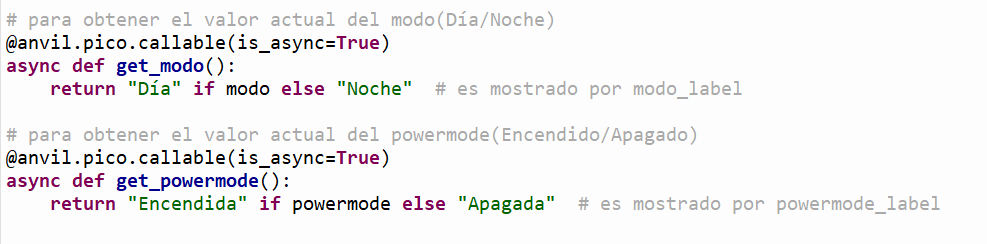
**@anvil.pico.callable(is\_async=True)**; Esta etiqueta está en todas las funciones, ya que indica que cada función se puede llamar en cualquier momento desde Anvil Works



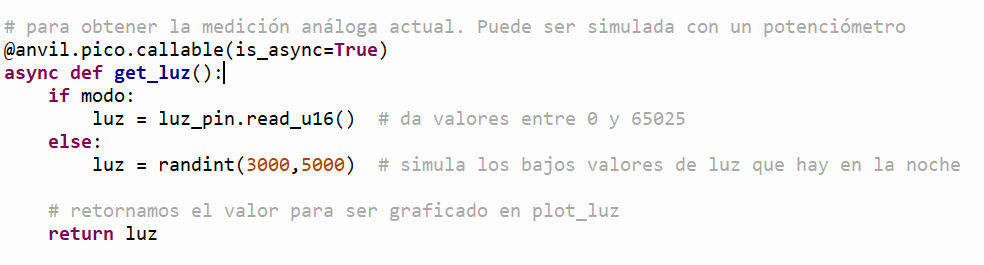
Esta función servirá para prender o apagar los 2 leds que están externos al Raspberry indicando si está de día o de noche, y por tanto, variar en su comportamiento respecto a las demás funcionalidades



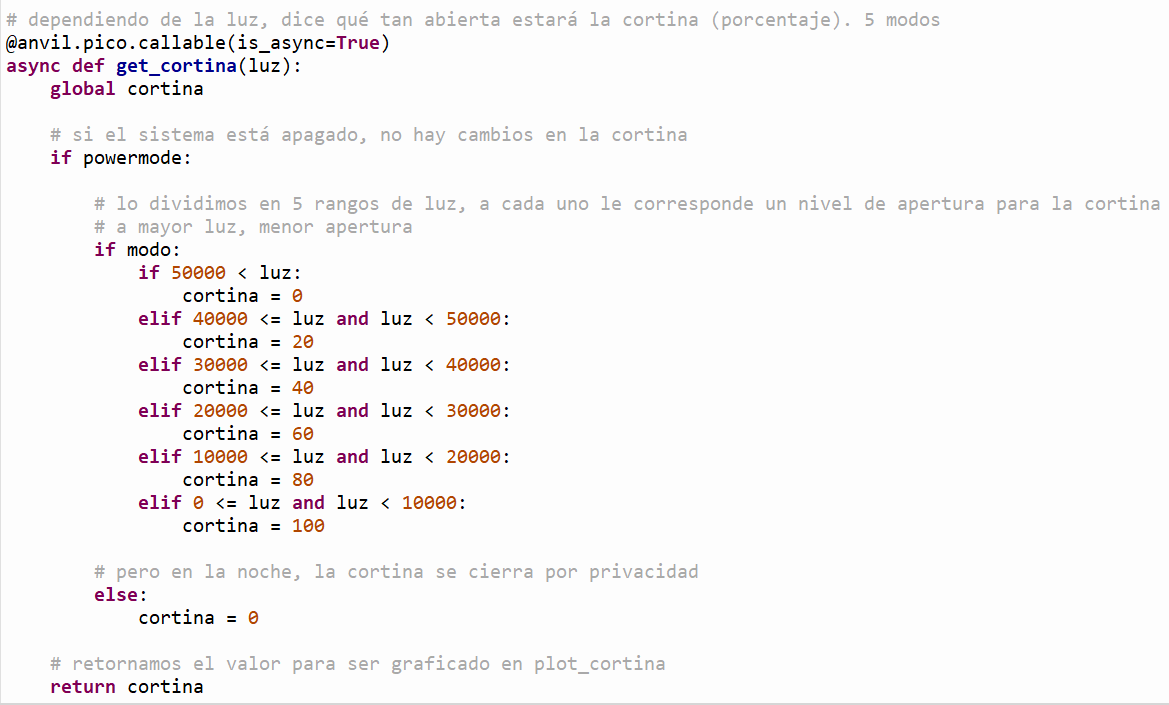
Esta función indicará si el sistema de cortinas automáticas está iniciado o no, manteniendo prendido o apagado el led que viene integrado al Raspberry



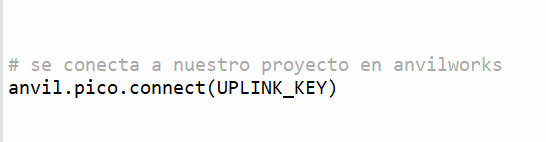
Estas 2 funciones servirán más que nada para indicaciones, el primero indicando si está de día o de noche cuando un led externo está prendido u otro, y la segunda función indicará si la cortina está prendida o apagada con el led interno.



La función get\_luz servirá para medir los niveles que entran de luz, simulados con el potenciómetro, y posteriormente estos se graficarán en AnvilWorks.



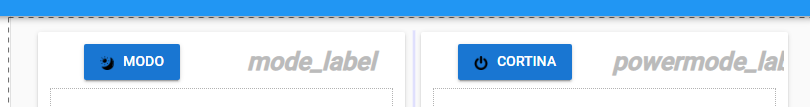
Finalmente, se tiene la función get\_cortina que dependerá de los niveles de luz (simulados con el potenciómetro) que entren, según los valores de luz recibidos, la cortina permanecerá cerrada (con el número 0) y si la luz es muy poca (es decir, está de noche) entonces la cortina se abrirá, además, si es de noche (osea, si el modo es False: noche) entonces la cortina aún así se cerrará por privacidad



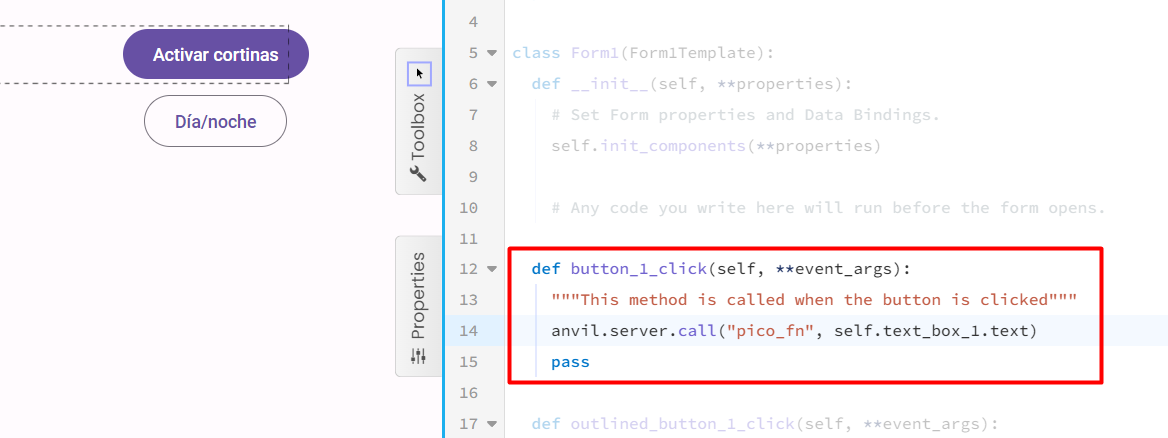
Al final se manda esta instrucción, para conectar el Raspberry con AnvilWorks porque se le manda la llave de conexión de la app

# Paso 7: Realizar llamados desde Anvil Works

Para nuestro proyecto, necesitamos poder cambiar manualmente entre el modo de día, y el de noche, además de poder encender y apagar el sistema. Esto lo haremos mediante dos botones que creamos anteriormente, “Modo” (que cambia el modo entre Día y Noche) y “Cortina” (que cambia el sistema entre encendido y apagado).

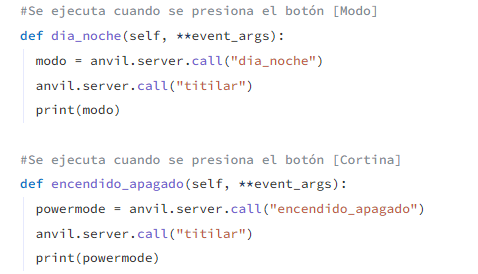


Les daremos doble click a cada botón. Esto nos creará dos funciones, que se ejecutarán cuando los botones sean clickeados.



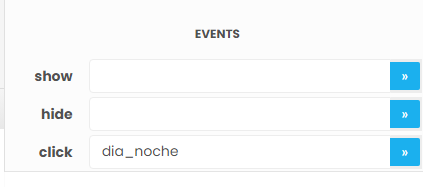
Renombramos las funciones como “dia\_noche” y “encendido\_apagado”, respectivamente, como las funciones que creamos en el Raspberry, ya que servirán para llamarlas.

Para llamar a las funciones que creamos anteriormente en el raspberry, nuestras funciones en Anvil se verán más o menos así:

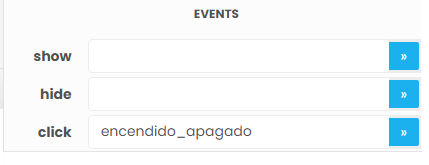


Para que los botones llamen a su función respectiva, debemos configurarlos desde sus propiedades > events, escribiendo el nombre de lo que quieres que ejecuten.

Así para el botón de modo:



Y así para el botón de cortina:



# Paso 8: Recibir datos y graficarlos

Finalmente podremos saber si el sistema está en modo de día o de noche, si la cortina está encendida o apagada, cuál es el valor actual de luz, qué tan abierta está la cortina, y ver una gráfica con el historial de estos dos últimos.

Para todo esto, necesitamos pedirle todos esos valores al Raspberry periódicamente, y mostrarlos en nuestra interfaz, la cual necesitará actualizarse cada vez que le lleguen datos nuevos.

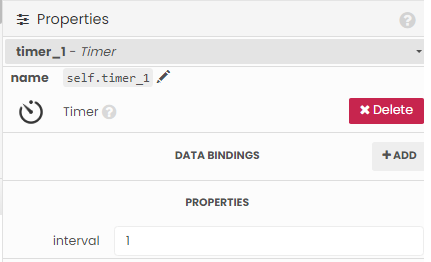
¿Cómo podemos ejecutar código periódicamente?, con el temporizador integrado de Anvil Works.

En nuestra interfaz, agregaremos este último componente dándole click a “más componentes”, y luego arrastrando este temporizador a la pantalla.

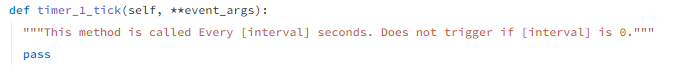


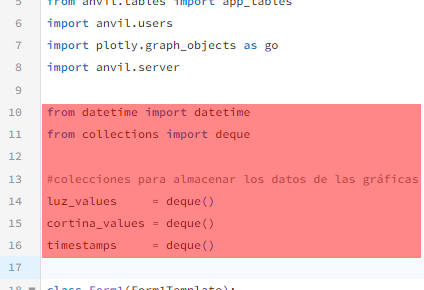
Anvil nos lo marcará como un componente invisible.

Le dejaremos su nombre por default, y le diremos que ejecute su función cada segundo.



Si le damos doble click, nos llevará a ella, en donde podremos comenzar a programar.



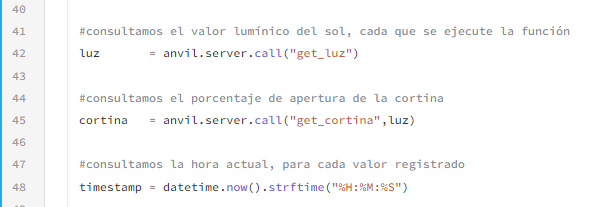


Antes de programar esta función, necesitamos tres estructuras para almacenar los datos, llamadas deque (double-ended queue), que son como listas, y poder tomar un registro de la hora actual, con la librería datetime.

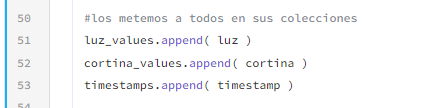
Llamamos a las deques que creamos, para poder modificarlas.



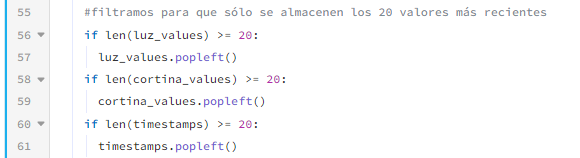
Pedimos las mediciones actuales



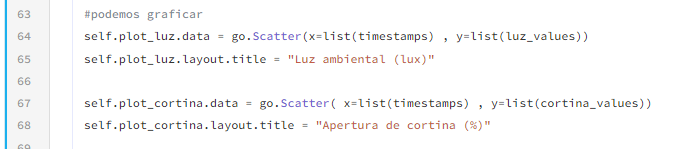
Los agregamos a la deque.



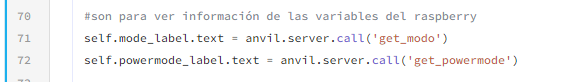
Sólo graficaremos las últimas 20 entradas, así que limpiamos las antiguas constantemente.



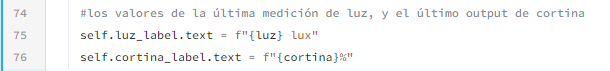
Con las dos listas que alimentamos, podemos comenzar a graficar, con actualización cada segundo.



Y al mismo tiempo, mostramos los valores actuales de modo (día o noche) y de la cortina (encendida o apagada).



Y el último valor registrado de luz y porcentaje de apertura de la cortina.



# Producto final (v1.0)

De día, mientras más luz haya, más se cierra la cortina, para comodidad del usuario.



Si la cortina está apagada, es una cortina común y corriente (no reacciona a la luz).



Cuando está en modo noche, la cortina se cierra por completo, por privacidad.

(Se simulan valores bajos aleatorios de luz, de acuerdo con la realidad).

# Segunda entrega (v2.0, contenido adicional)

Ahora hay un **botón nuevo** para poder alternar la toma de datos del sensor de luz, así podremos **dejar de tomar datos.**

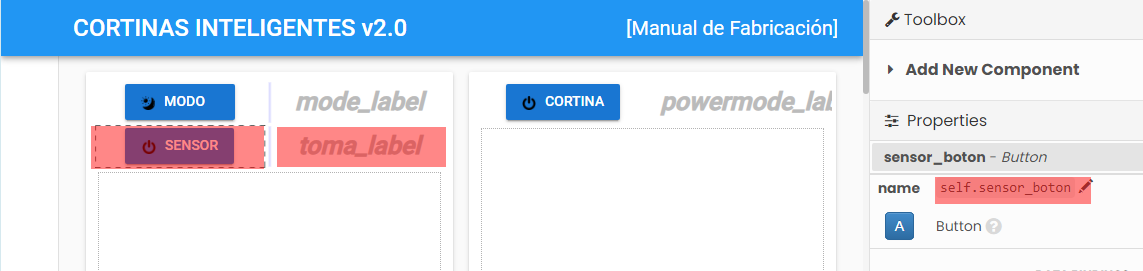
También hay un **conversor manual de Luz → Apertura**, en el que puedes introducir algún valor lumínico entre 0 y 50.000, y te dirá qué tan abierta estaría la cortina en ese caso.

Finalmente, **podremos ver los mensajes que se imprimen por consola**, que son enviados **cada que se cambia una variable** al usar un botón.

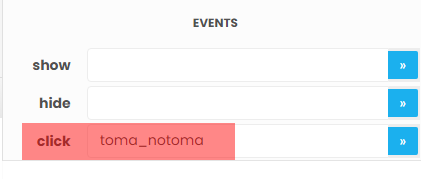
Ahora veremos cómo crearlos:

## Frenar toma de datos

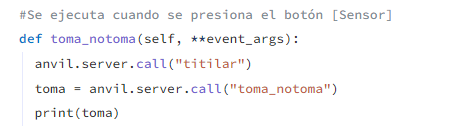
Añadiremos un botón y un label nuevos, justo como hicimos con los de arriba. El botón se llamará “sensor\_boton” y el label se llamará “toma\_label”

****

Configuramos el botón para que se ejecute la función “toma\_notoma” cada que se presione.

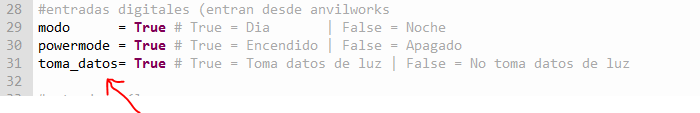
****

Creamos la función:



Lo que hará sólo será llamar a la función de igual nombre en el raspberry, así que la crearemos también allá.

Primero creamos la variable toma\_datos, que será la que vamos a alternar con la función.



Le asignamos un led, que solo estará encendido si se están tomando datos.



Y luego la función en sí.



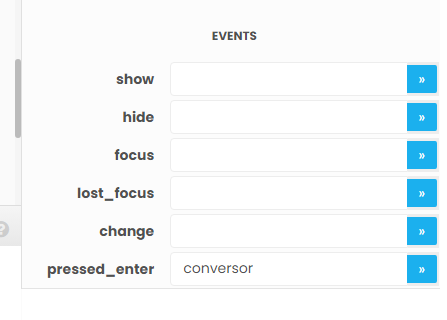
Alternará el valor toma\_datos entre True y False, y el toma\_led (18) entre encendido y apagado. También retornará un mensaje con la interpretación de lo que acaba de pasar, para imprimirlo en la consola.

## Conversor

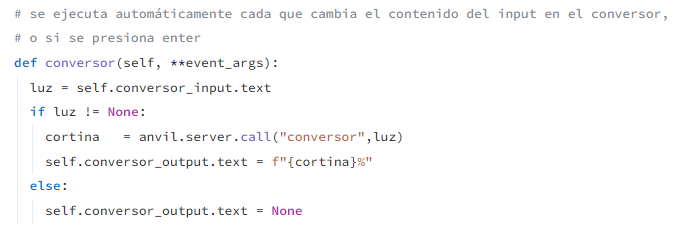
Creamos una nueva card en la interfaz, para ponerle el título a nuestra nueva funcionalidad. Un textbox que servirá de input llamado “conversor\_input”, y un label llamado “conversor\_output”.



Configuramos “conversor\_input” para que ejecute la función “conversor” cada que se presione enter.

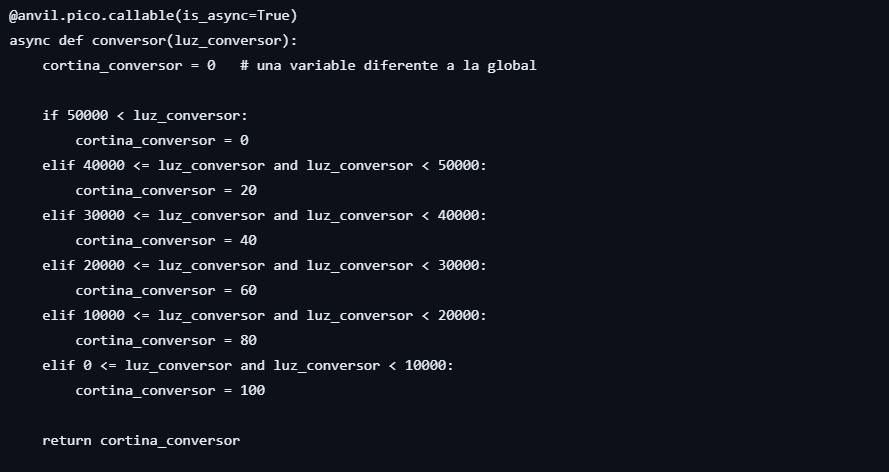


Lo que hace esta función es sólo recibir el valor que manda el input, y se lo manda a la función conversor del raspberry. La respuesta es recibida en otra variable, y luego la muestra el label de output.



También tiene una opción por si no se le manda nada.

Ahora vemos la función conversor en el raspberry:



El conversor tomará los datos que se pongan en el valor de la luz, y los almacenará en una variable local, y se retorna esta nueva variable

Es como la función get\_cortina, pero simplificada. Era necesario que fueran dos funciones diferentes con variables diferentes, o los resultados se confundirían.

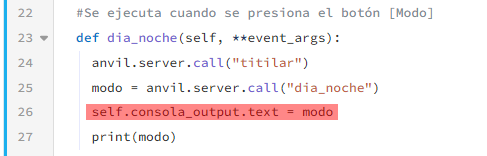
## Consola

Haremos una nueva card, con un título, y otra card adentro pero negra, en cuyo interior tendrá un label llamado consola\_output.

(Opcional): Cambiarle la fuente a Roboto, y el color a verde, para que se vea como una consola real.



Ahora, en todos los lugares en donde tengamos un print (en todas las funciones que se ejecutan al presionar algún botón), también le enviaremos esa información a consola\_output, de esta forma:



Haremos esto en dia\_noche, encendido\_apagado, y en toma\_notoma. Nos avisará cada que se actualice una variable.

# Producto final (v2.0)



