

# manual dispensador para mascotas

El presente manual se ha desarrollado con el propósito de permitir al usuario implementar el Dispensador de comida para mascotas IOT a través del paso a paso expuesto en el mismo. Además, permite adquirir y ampliar los conocimientos referentes al mundo de Arduino.



## autores

- ANGELA GÓMEZ  
LIZARAZO
- DAIRON  
SHMELINGUER  
PORRAS CÁRDENAS

# GUIA DEL MANUAL



- ✓ **ASPECTOS Teóricos:** En este ítem, se encontrarán todos los conceptos y definiciones de los componentes del Dispensador desde un punto teórico para mejor entendimiento hacia el lector.
  
- ✓ **modelación 3d:** En este ítem, se encontrará la maquetación del Dispensador, junto a los archivos STL descargables con sus proporciones necesarias listas para su impresión
  
- ✓ **CIRCUITO Y programación ARDUINO:** En este ítem, se encontrarán todas las conexiones necesarias para el desarrollo del dispensador. Además, se especificarán todos los aspectos necesarios para la implementación del mismo, incluyendo las librerías y el código necesario para el funcionamiento. Cabe resaltar que en este ítem se desarrolla el programa en base los cuatro componentes de IOT (manual, programable, app y comando por voz)
  
- ✓ **BLINK – APP:** En este ítem, se encontrará el instructivo para registrarse en la plataforma virtual del BLYNK, la descarga de la APP, el código para implementar Blynk en el código fuente de Arduino, la sincronización de la APP y el dispensador.
  
- ✓ **SINCRONIZACION Google ASISTENTE:** En este ítem, se encontrará el instructivo para desarrollar el control del dispensador a través de google asistente.



¿Listo para iniciar tu propio proyecto?

Si es así, es hora de iniciar 

Aspectos teóricos...

## Arduino

Es una plataforma de creación electrónica con código abierto, lo cual está basada en hardware y software libre, se utiliza como un microcontrolador reprogramable con una serie de pines que permitirá establecer conexiones entre el controlador y los diferentes sensores, de la cual se podrán crear objetos electrónicos e interactivos como robots o sistemas



## Blynk

Es una plataforma IoT con IOS y ANDROID independiente de hardware con código abierto a través de wifi, nos permite conectar dispositivos a la nube y diseñar aplicaciones para controlarlas de forma remota, donde el usuario pueda crear proyectos IoT desde su celular con una conexión de red, permitiéndole modificar al instante agregando controles, botones.

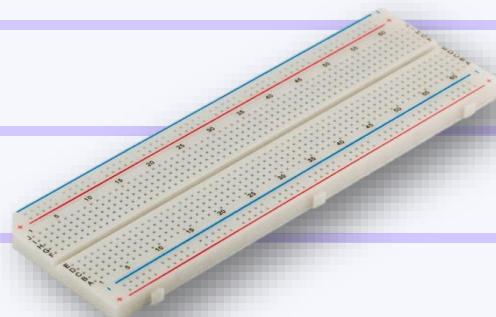
## ESP32

Es una serie de microcontroladores de bajo costo y de bajo consumo con sistema en chip con Wi-Fi y Bluetooth de modo dual integrados, y certificada que proporciona no solo la radio inalámbrica, sino también un procesador integrado con interfaces para conectarse con varios periféricos.



## Protoboard

La protoboard MB102 es una placa de pruebas especial para el montaje de proyectos electrónicos. Esta tiene orificios conectados entre sí por medio de unas láminas metálicas. En la protoboard se pueden montar resistencias, leds, condensadores integrados, entre otros.



## Servomotor MG995 metálico

es un actuador de rotación continua que permite el control preciso de la posición, velocidad y aceleración angular. Su rotación va de 0 grados a 180 grados. Este tipo de servomotor es utilizado en una amplia variedad de proyectos electrónicos. Este servo posee un conector de tres pines que son compatibles con la mayoría de controladores del mercado

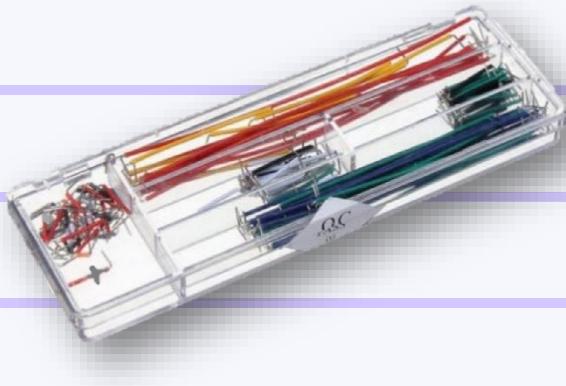


El módulo I2C es un bus de comunicación que usa dos líneas para enviar y recibir información y otras dos para alimentación. La pantalla tiene una retroiluminación de LED, arrojando imagen en sus dos filas con 16 caracteres por fila. En ella se puede apreciar los rectángulos para cada carácter y los pixeles que componen a cada uno.

## Pantalla LCD alfanumérica

2x16 con módulo I2C

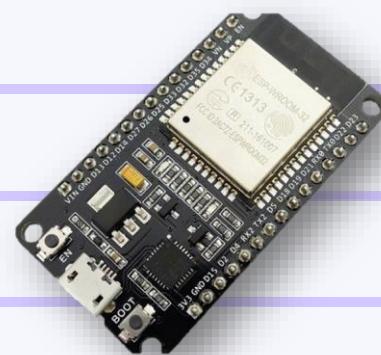
## Cable UTP



Este tipo de cable es especial para el desarrollo de proyectos electrónicos, pues me permite a través de la protoboard servir como puente de comunicación con otros componentes conectados a la misma.

## Tarjeta Esp32 wroom320

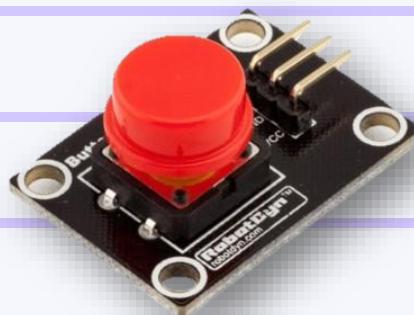
La ESP-32 está orientado para IoT, sin embargo, se puede usarse en diferentes áreas, ya que tiene unas características de rendimiento muy altas como un procesador que se puede usar en modo single o dual llegando a un máximo de 240Mhz, tiene 36 pines multi propósito donde tenemos señales de PWM canales ADC, DAC, interfaces como UART, I2C SPI, además tiene módulos Wireless con comunicación Wifi y bluetooth.



## Adaptador de voltaje 5V 2<sup>A</sup>

Se usa para dar fuente de alimentación USB donde quiera que se necesite. El adaptador no tiene un cable, en su lugar hay un puerto USB tipo A en la parte inferior, donde se puede conectar cualquier cable USB.

El adaptador funciona también para dispositivos IOS



## Pulsador en board con conectores

Componente eléctrico que impide o permite el paso de corriente eléctrica cuando se pulse. El pulsador solo se abre o se cierra cuando el usuario lo presiona y lo mantiene presionado.

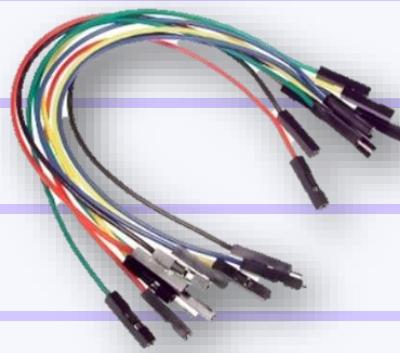
## Bornera hembra plug



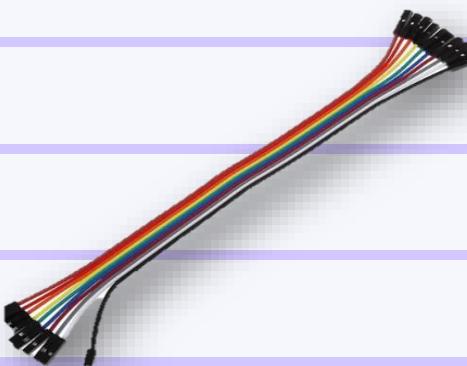
permite conectar un conector barrel jack a un par de cables. Un extremo tiene terminales de tornillo y el otro tiene un conector hembra Jack-DC de 5.5x2. 1mm de terminal central positiva. Ideal para conectar una fuente de alimentación con salida barrel jack a un protoboard/motor.

## Jumper macho - macho

es un elemento que permite cerrar el circuito eléctrico del que forma parte dos conexiones. La función del cable macho-macho es con frecuencia usado en el tablero protoboard haciendo posible la conexión de dos elementos ingresados en dicho tablero.



### Jumper hembra - hembra



Vienen con las 10 unidades separadas, de forma que puedes trabajar con ellos individualmente. Es muy útil para crear el cableado o puentes en tus prototipos de electrónica. Los puedes utilizar con una protoboard o cabezas para unir a la placa de desarrollo.

### Jumper macho hembra

Es con frecuencia usado en el tablero protoboard haciendo posible la conexión de dos elementos, uno ingresado en dicho tablero y el extremo opuesto al sensor (normalmente).





## Ahora vamos con la modelación de nuestro dispensador

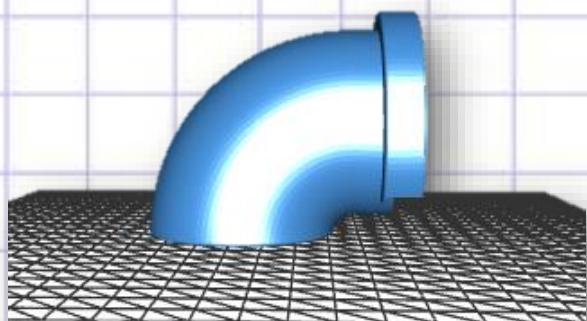
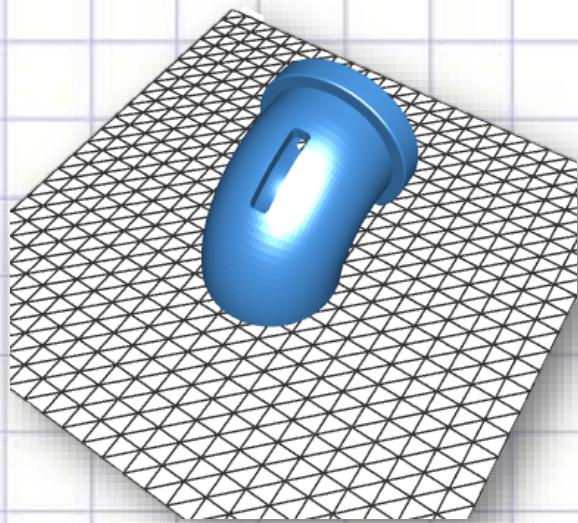
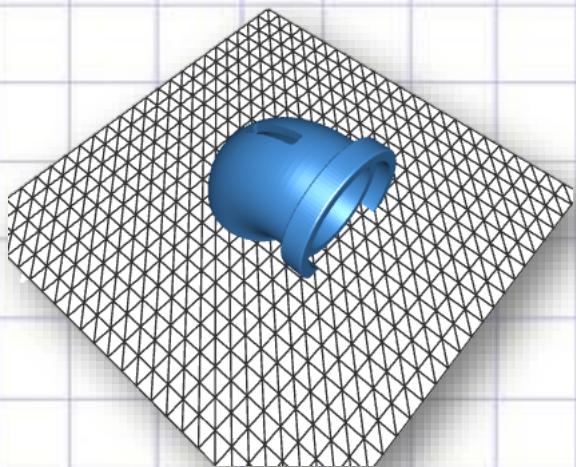
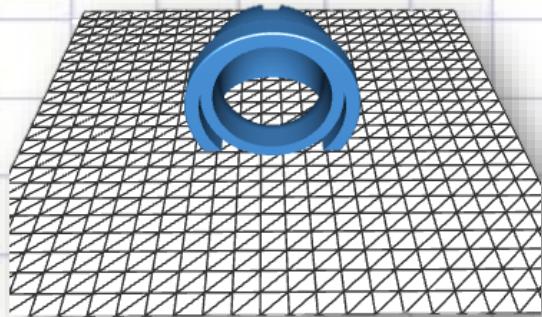
Te presentamos el diseño del Dispensador de comida para mascotas IOT, diseño el cual fue seleccionado después de una búsqueda minuciosa de dispensadores que se acoplaran al enfoque del proyecto. Cabe resaltar que el diseño ya estaba prediseñado, y la fuente de la que fue tomado permitía a través de su canal el uso del mismo.





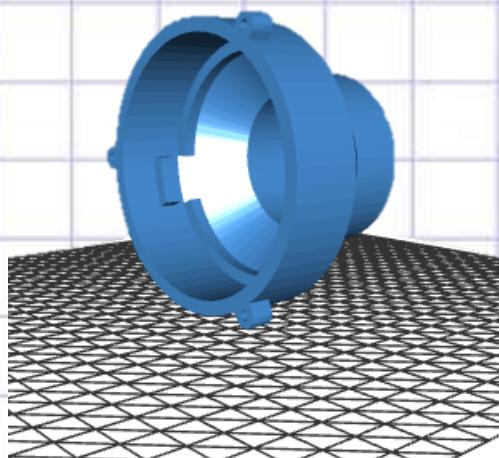
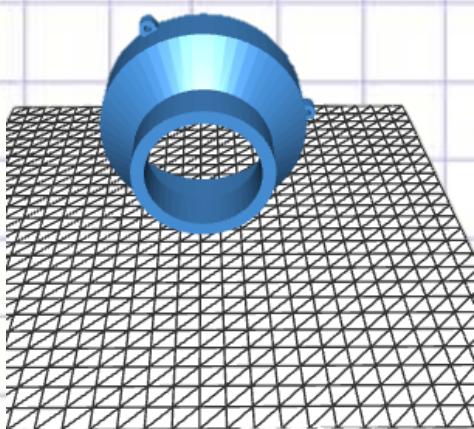
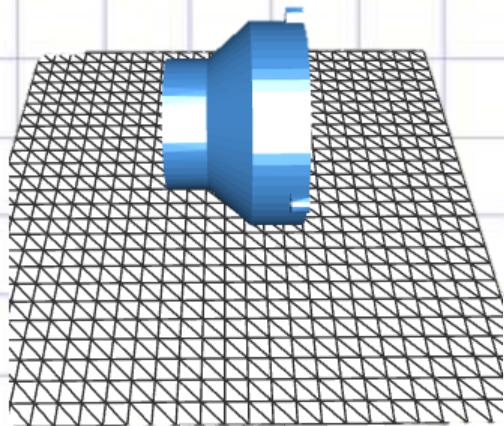
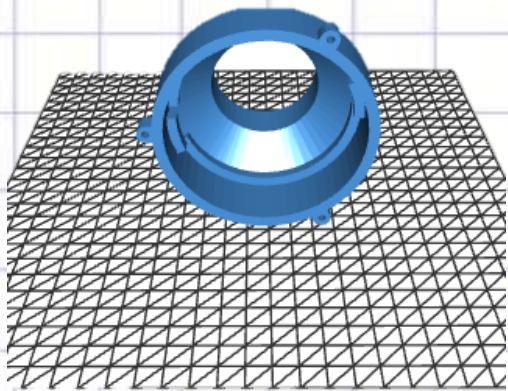
## PARTES PARA LA IMPRESIÓN 3D

### 2.1 CODO



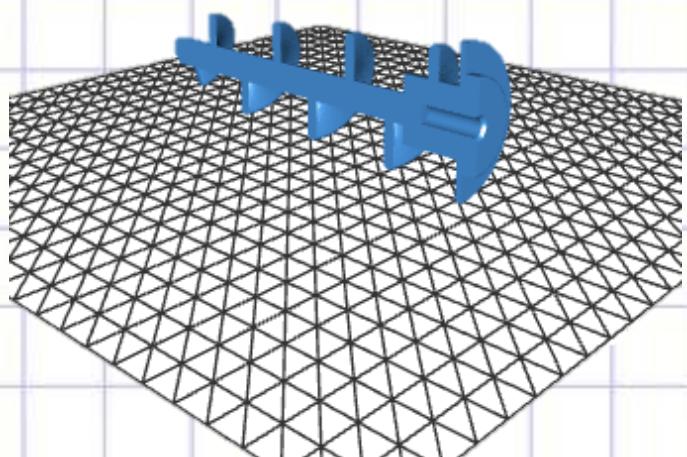
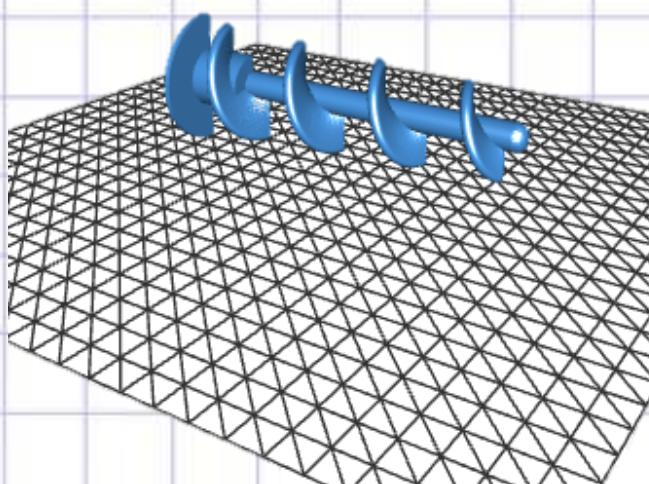
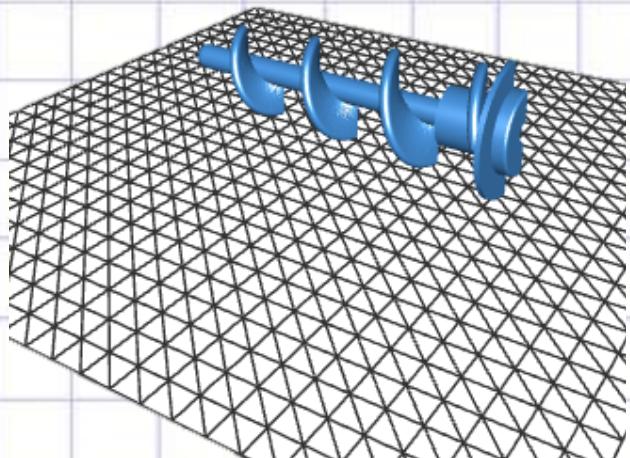
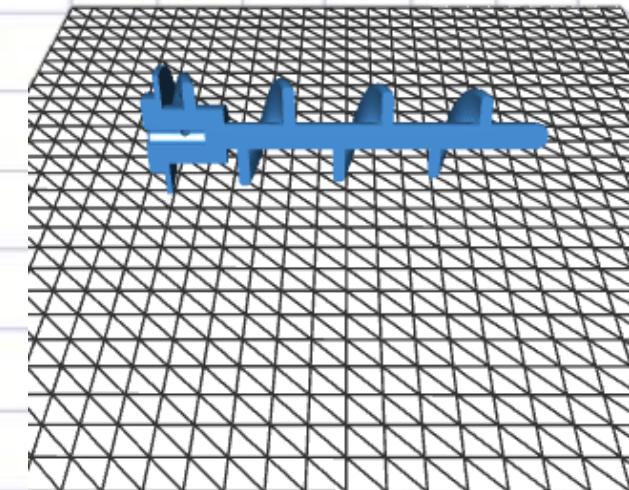


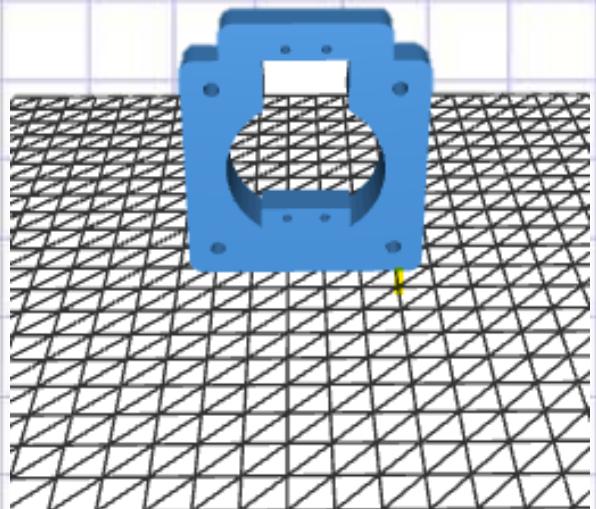
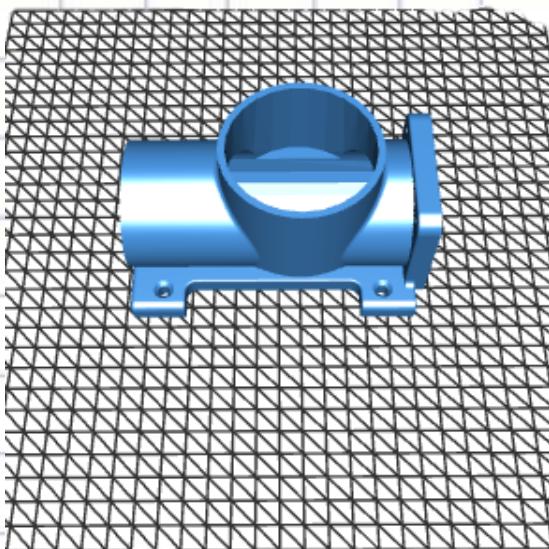
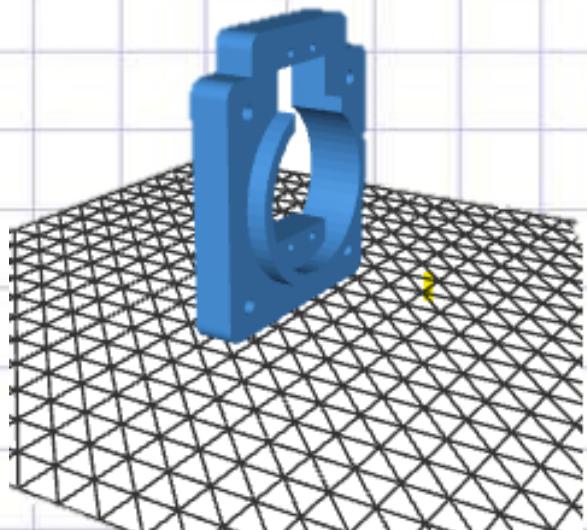
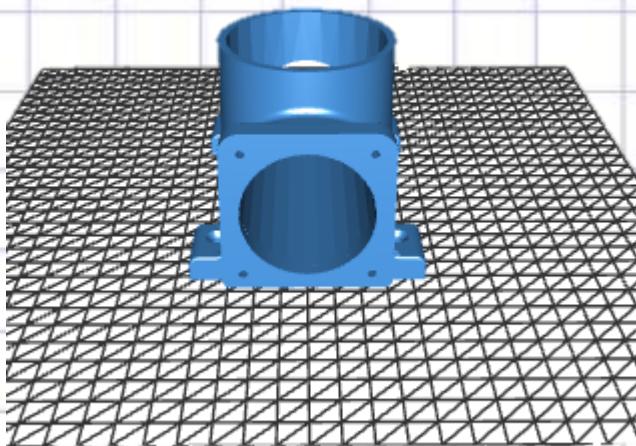
## 2.2. HOPPER

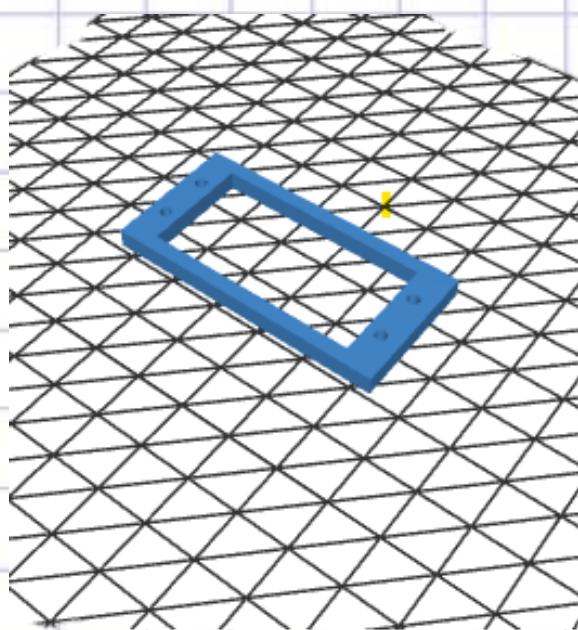
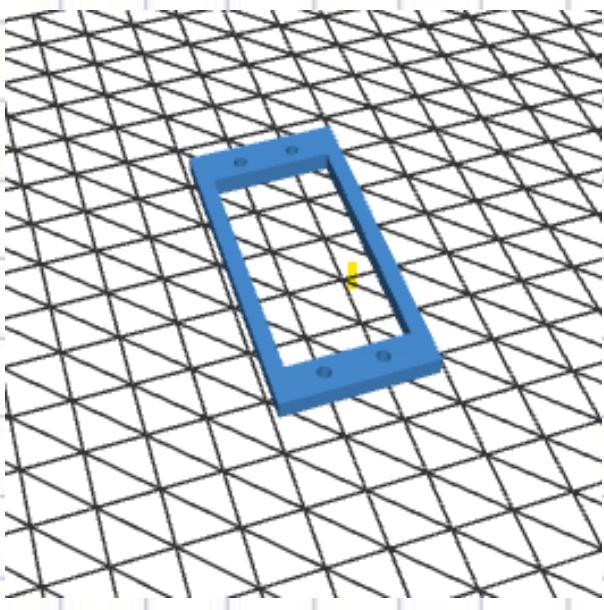
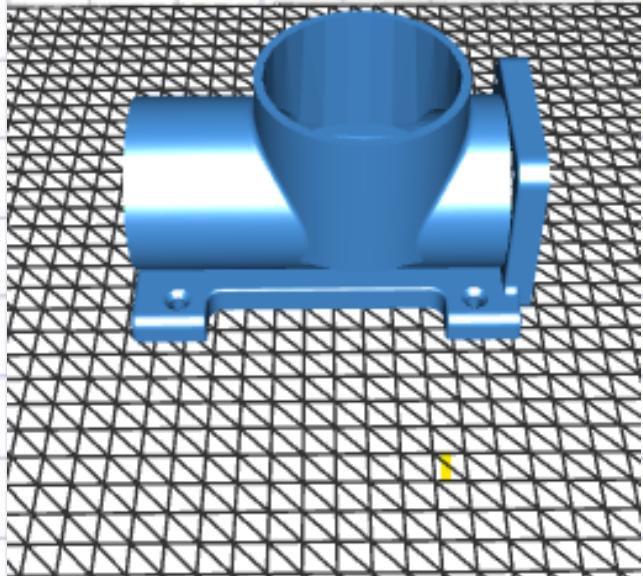
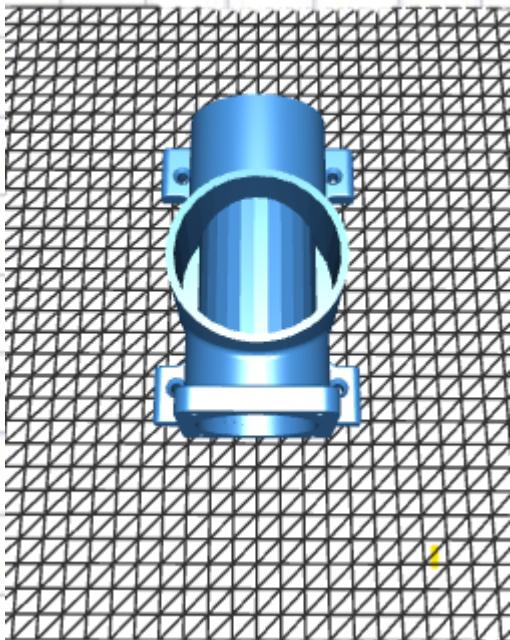




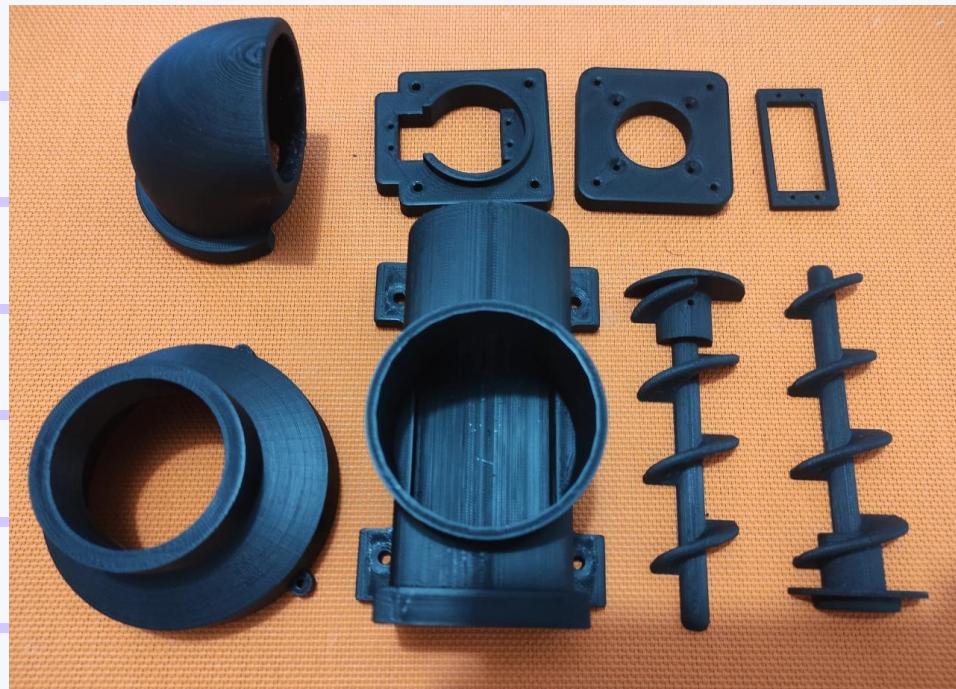
2.3







Una vez impresas las partes de nuestro dispensador expuestas anteriormente, se obtendrá como lo resultado nuestro prototipo de Dispensador.



**NOTA:** los materiales que usamos para el proceso del armado son los siguientes:

- 4 tornillas m2.5 de 20 milímetros con tuerca
- 5 tornillos m3 de 15 milímetros (2 de ellos sin tuerca y el restante con tuerca arandela)
- 2 tornillos m3 de 20 milímetros con tuerca
- 4 tornillos de cabeza fresada m4 de 20 milímetros con arandela y tuerca
- Pega todo o pegante instantáneo

A través de este link podrás acceder a los STL para la impresión 3D, diseño que fue tomado de YouTube ya que está abierto al público, con finalidad de que sea implementada en diferentes desarrollo de dispensadores.  
(<https://github.com/hardrive9000/PetFeeder/tree/master/3D>)

## 1. CIRCUITO Y PROGRAMACIÓN

### 3.1. DISPOSITIVOS Y HERRAMIENTAS

- 1 protoboard
- pantalla lcd 16 x 2 i2c
- tarjeta esp32 – referencia esp-wroom-32
- servomotor de rotación continua- referencia mg995
- 3 pulsadores en board con conectores
- 1 bornera hembra pug
- 1 adaptador de voltaje 5v
- jumpers macho-macho, hembra-hembra, macho-hembra
- cinta aislante negra

### 3.2. CONEXIONES A LA PROTOBOARD

Puertos de la esp32 que se usaron, donde cada color representa un dispositivo en la protoboard

#### PARA LA LCD:

- GND
- VIN
- D21
- D22

#### SERVOMOTOR:

- GND
- VIN
- D32

#### PULSADOR 1

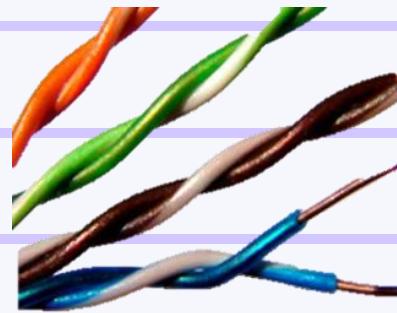
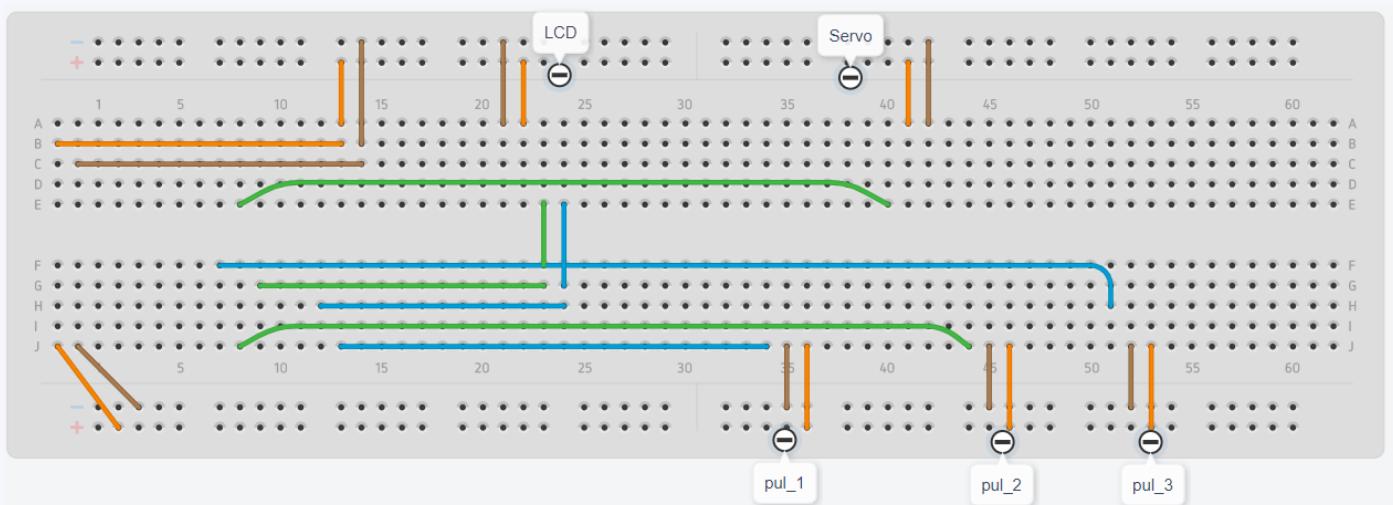
- GND
- 3V3
- D23

#### PULSADOR 2

- GND
- 3V3
- D19

#### PULSADOR 3

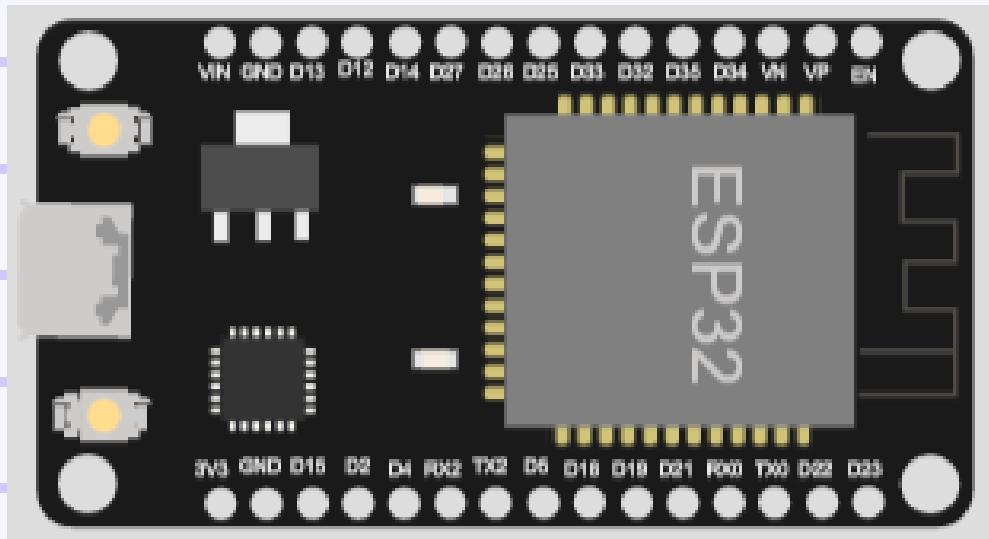
- GND
- 3V3
- D18



En la siguiente imagen encontrarás todas las conexiones necesarias para que los puertos de la esp32 funcionen, pues estas conexiones son indispensables para el funcionamiento de cada dispositivo conectado a la protoboard. El cable que se utiliza es UTP, ya previamente cortado. Recuerda quitar un poco del plástico de los extremos del cable como se muestra en la imagen para mayor comunicación.

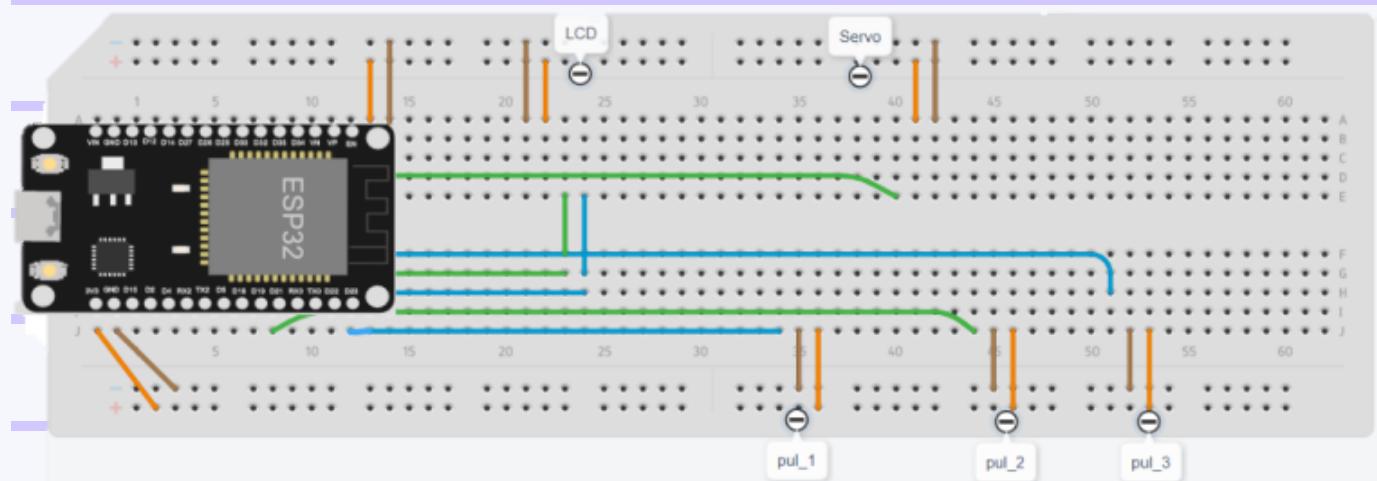
### 3.3. CONEXIÓN DE LA ESP32 A LA PROTOBOARD

En esta conexión debes asegurarte de que la ESP32 case bien en la protoboard. Su conexión de manera vertical va desde A hasta I, y de manera horizontal va desde el primer pin de A, contando 15 espacios, incluyendo a (A), pues la Esp32 tiene 15 pines. Debes posicionarla de esta manera

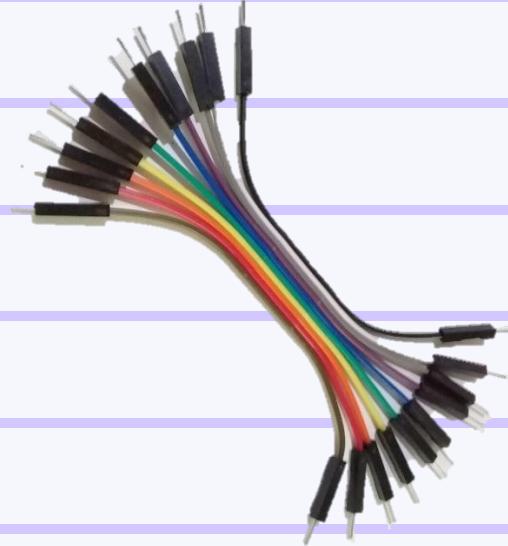


### 3.4. CONEXIÓN DEL SERVOMOTOR

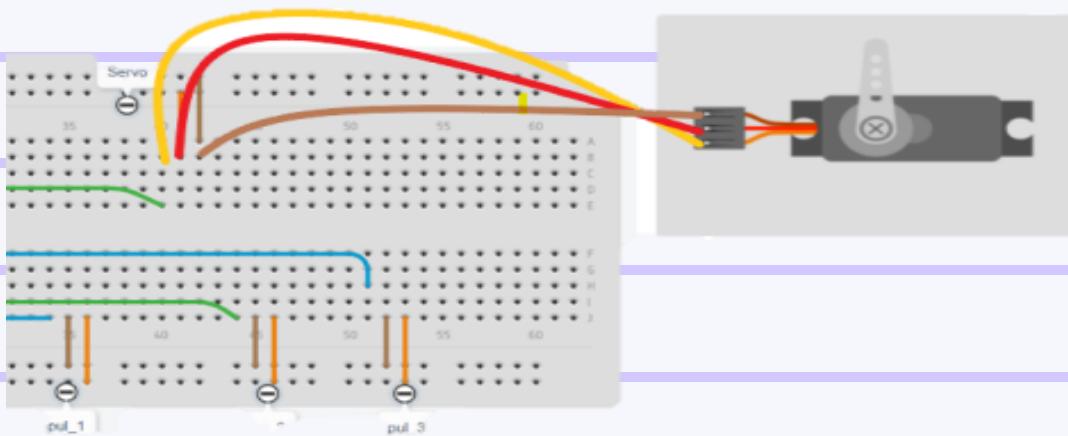
Para esta conexión debes usar servomotor de rotación continua MG995, el cual es indispensable para el desarrollo del dispensador. El servo cuenta con tres conectores: GND (polo a tierra-color marrón), VCC (voltaje-color rojo) y S (señal del servo- color amarillo). Debes usar jumpers macho – macho,



los cuales se encargarán de implementar la comunicación entre los dispositivos.

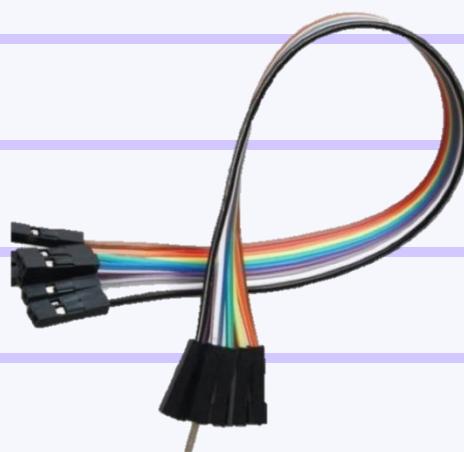


Insertas un extremo del jumper a la protoboard y el otro lo conectas al servo. Recuerda que debes conectarlos de manera coherente, es decir, el gnd de la protoboard con el gnd del servo, el vcc con los 5v de la ESP32 y S(señal) con el pin D32. Cabe resaltar que los cables UTP sacan los puertos de la ESP32, es decir, comunica al pin que necesitas.

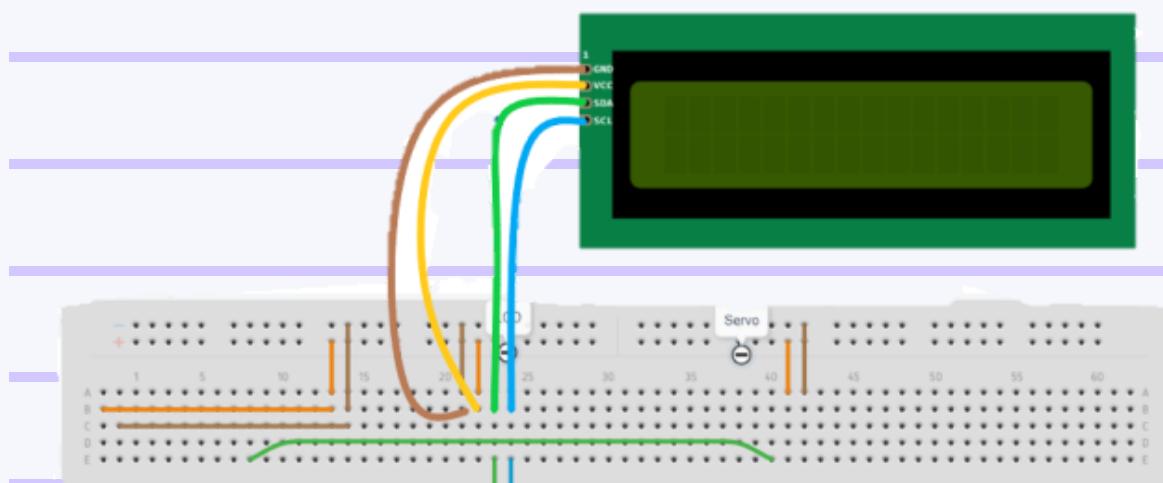


### 3.5. CONEXIÓN DE LA LCD

En esta conexión usaremos nuestra Pantalla LCD 16 X 2 I2C (2 filas por 16 columnas). La pantalla cuenta con 4 puertos las cuales se organizan de la siguiente manera GND (color marrón – tierra), VCC (color amarillo – voltaje), SDA y SCL (puertos de señal – verde y azul respectivamente).



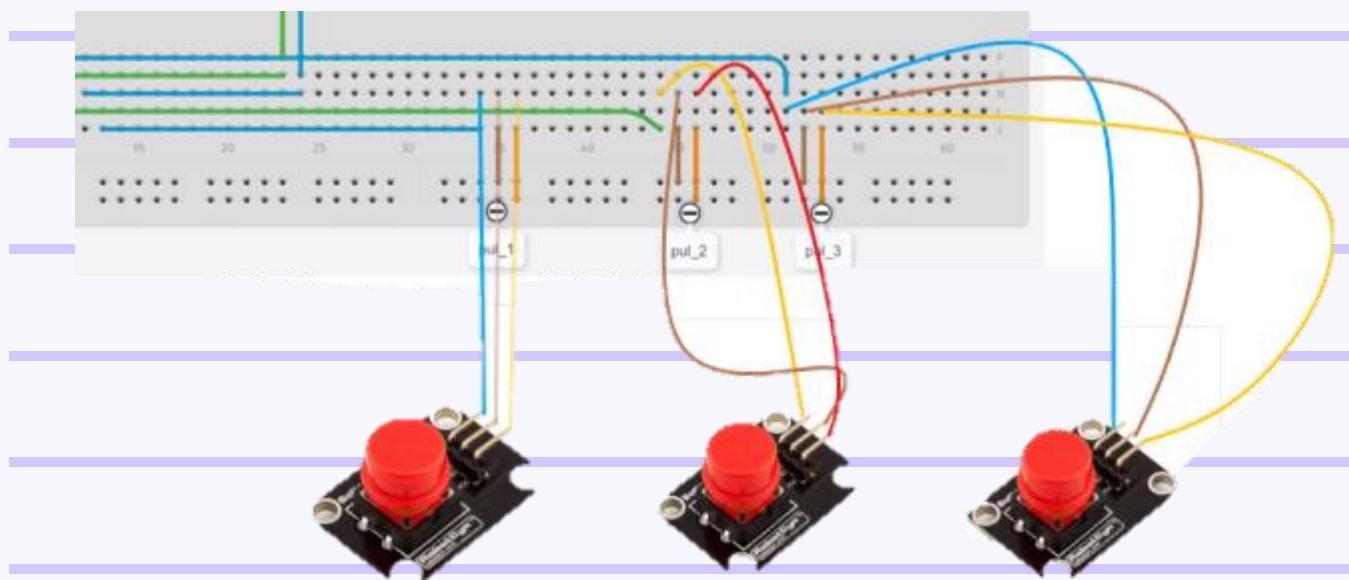
También se implementarán los cables jumpers hembra – macho de esta manera: en la protoboard se conecta el jumper macho y en la LCD el jumper hembra.



Las conexiones son las siguientes: en el puerto GND de la protoboard conectas GND de la LCD, en el puerto de los 5v de la protoboard conectas el VCC de la LCD, en el puerto D22(frente al cable de color verde UTP) conectas el SDA de la LCD, en el puerto D21 (frente al cable de color azul UTP) conectas el SCL de la LCD.

### 3.6. PULSADORES

En estas conexiones cada pulsador tiene su propio pin (D23 – D18 – D19) en ESP321 a través de los cables UTP. Cada pulsador tiene 3 puertos. El primer puerto es GND, el segundo VCC y el último S (puerto de señal).

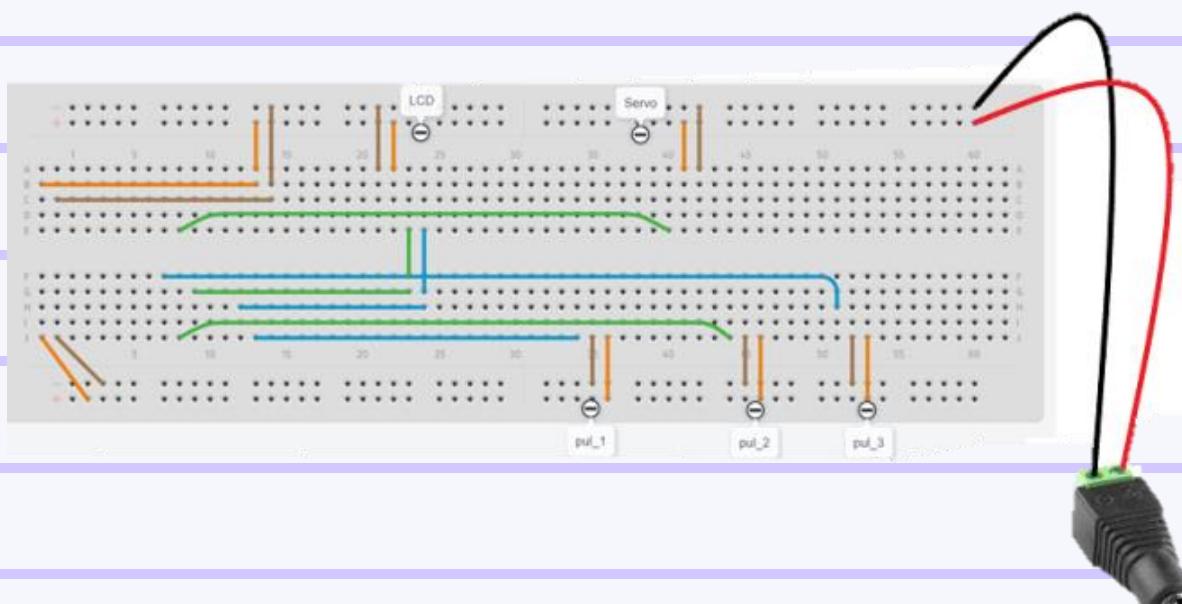


Los jumpers utilizados para cada puerto son los Macho – Hembra, donde el macho se conecta a la protoboard y el hembra al pulsador. Para ver los pines de cada pulsador en la esp32 vaya al punto **3.2 CONEXIONES A LA PROTOBOARD**

### 3.7. CONEXIÓN BORNERA A LA PROTOBOARD

En esta conexión utilizamos cables puente para conectar nuestra bornera a la protoboard. En este caso a través del convertidor o adaptador de voltaje se suministrará energía a toda la protoboard.

El cable negro representa la tierra o GND y el rojo el voltaje





## 2. PARTE PROGRAMACION

### 2.1. DESCARGA ARDUINO

Ingresas a la página oficial de Arduino para descargar el programa, el cual es indispensable en el funcionamiento del dispensador, pues a través de él, se creará el código. Das click en ARDUINO - HOME

Google search results for "arduino":

- [Arduino - Home](https://www.arduino.cc)  
Open-source electronic prototyping platform enabling users to create interactive electronic objects.
- [Software](#)  
Arduino IDE 1.8.19 ... The open-source Arduino Software (IDE) ...
- [Hardware](#)  
Portenta - Arduino Nano - Arduino Mega 2560 Rev3 - ...
- [Arduino Store](#)  
Explore the full range of official Arduino products including ...
- [Getting Started](#)

Al ingresar al ARDUINO – HOME nos lleva a esta página.

Arduino.cc homepage navigation bar:

- PROFESIONAL
- EDUCACIÓN
- TIENDA
- HARDWARE
- SOFTWARE
- NUBE
- DOCUMENTACIÓN ▾
- COMUNIDAD ▾
- BLOG
- SOBRE
- REGISTRARSE





Estando allí, en la parte superior encontramos un menú. De allí seleccionamos la opción de SOFTWARE el cual no dirige a la página de descarga de Arduino. En este ítem, encontrarás todos los archivos de instalación para ARDUINO IDE

The screenshot shows the Arduino website's navigation bar with options like HARDWARE, SOFTWARE, NUBE, DOCUMENTACIÓN, COMUNIDAD, BLOG, and SOBRE. The SOFTWARE option is highlighted. Below the bar, there's a search bar and a sidebar with links for CODIGO EN LÍNEA and EMPEZANDO. A central panel displays a snippet of code and a preview of the Arduino IDE interface.

## Descargas

This screenshot shows the download page for Arduino IDE 2.0.0. It features a large button for "Arduino IDE 2.0.0". Below it, there's a brief description of the new features and a link to the documentation. To the right, there's a section titled "OPCIONES DE DESCARGA" listing download links for Windows, Linux, and Mac OS.

Seleccionas en las opciones de descargas la versión que deseas descargar de acuerdo a tu Windows, en este caso descargaremos la primera opción WIN 10 Y POSTERIORES – 64 BITS. Una vez seleccionada, nos redirecciona a la siguiente página:

This screenshot shows the Arduino IDE donation page. It features a central box with the heading "Support the Arduino IDE" and a message about the number of downloads. Below this are buttons for different donation amounts (\$3, \$5, \$10, \$25, \$50, Other) and two main download buttons: "JUST DOWNLOAD" and "CONTRIBUTE & DOWNLOAD". There are also illustrations of a character and a robot.



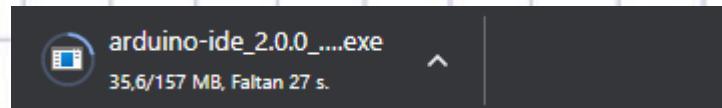


En esta página encontrarás información referente a donaciones para los colaboradores de Arduino. Esta donación es voluntaria. En esta ocasión sólo tomarás la opción JUST DOWNLOAD.

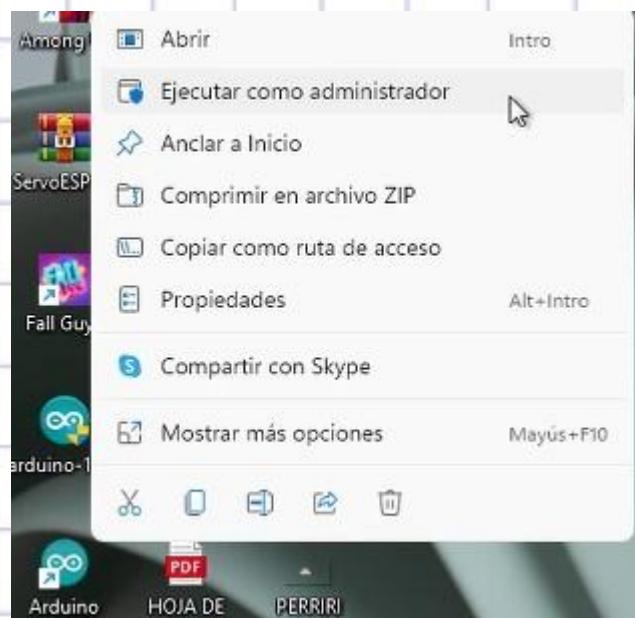
[JUST DOWNLOAD](#)

[CONTRIBUTE & DOWNLOAD](#)

Al dar click, el archivo de Arduino se descargará por completo.



Ya descargado el archivo, seleccionas una carpeta de pc para descomprimir el archivo. Ya establecido en la carpeta, seleccionamos el programa y lo ejecutamos como administrador.

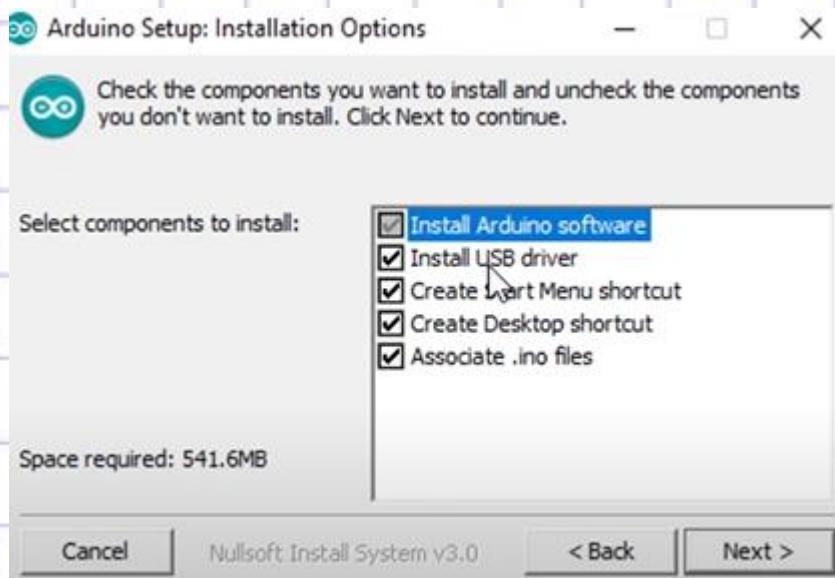




Al ejecutarlo como administrador el programa emitirá esta página

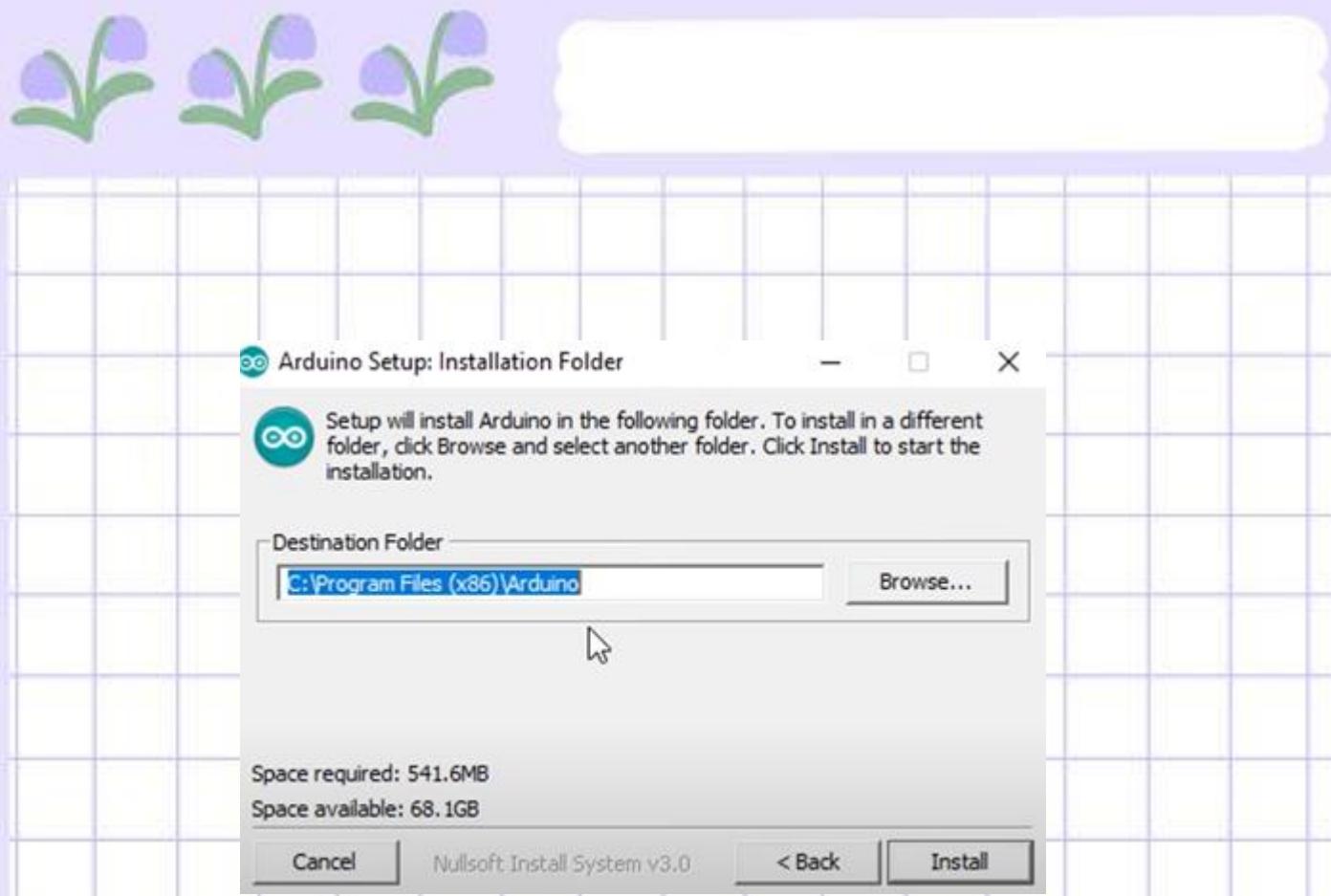


Esta es la licencia que emite el programa de Arduino. Damos click en I Agree y aceptas los términos de la instalación del programa.

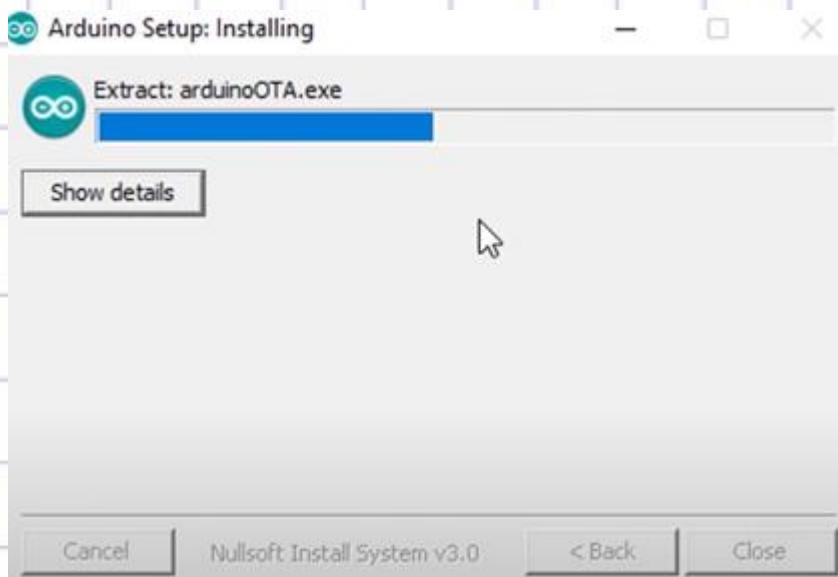


Dejamos las opciones que deja el programa tiene por defecto, y damos en la opción de NEXT.

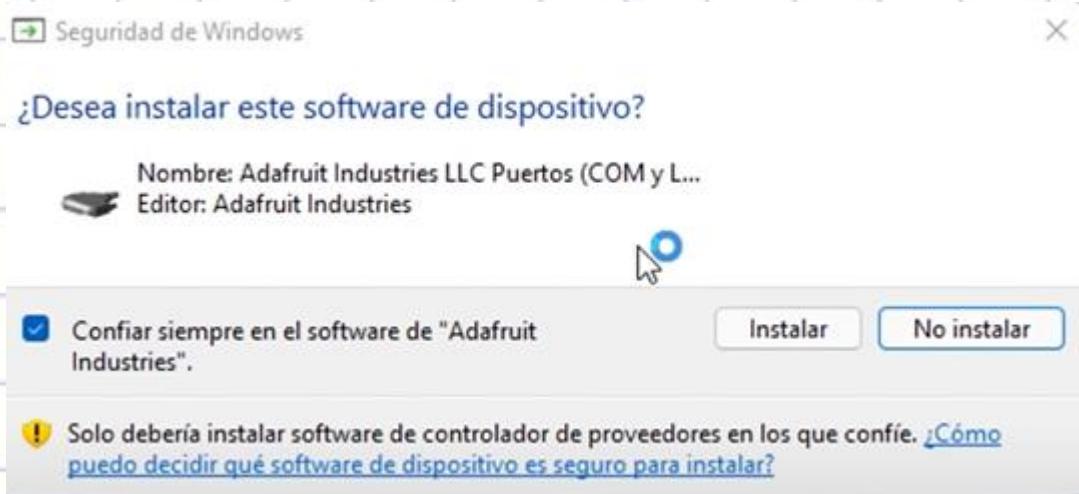




Seleccionamos la partición en la que deseamos la ejecución del programa y seleccionamos instalar.



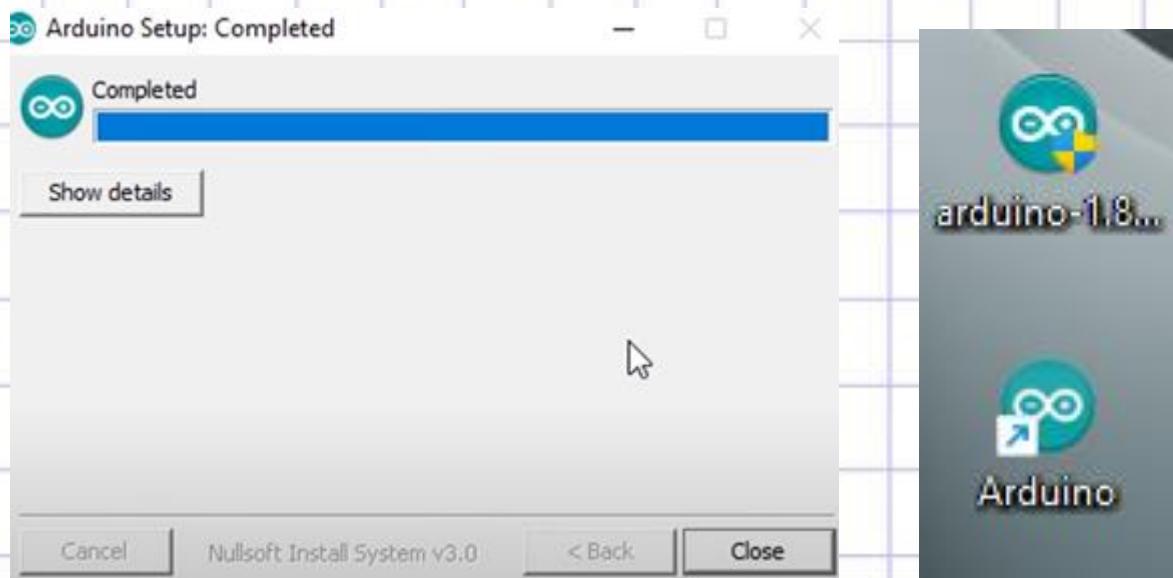
Una vez seleccionada la opción de instalar, el programa iniciará su instalación.



El programa emitirá este mensaje, y allí debemos deseleccionar el ítem como se muestra en la imagen



Una vez realizado el paso anterior, debes dar en la opción Instalar. Al final, la instalación se completa



Ingresamos al programa de Arduino para verificar su funcionamiento.





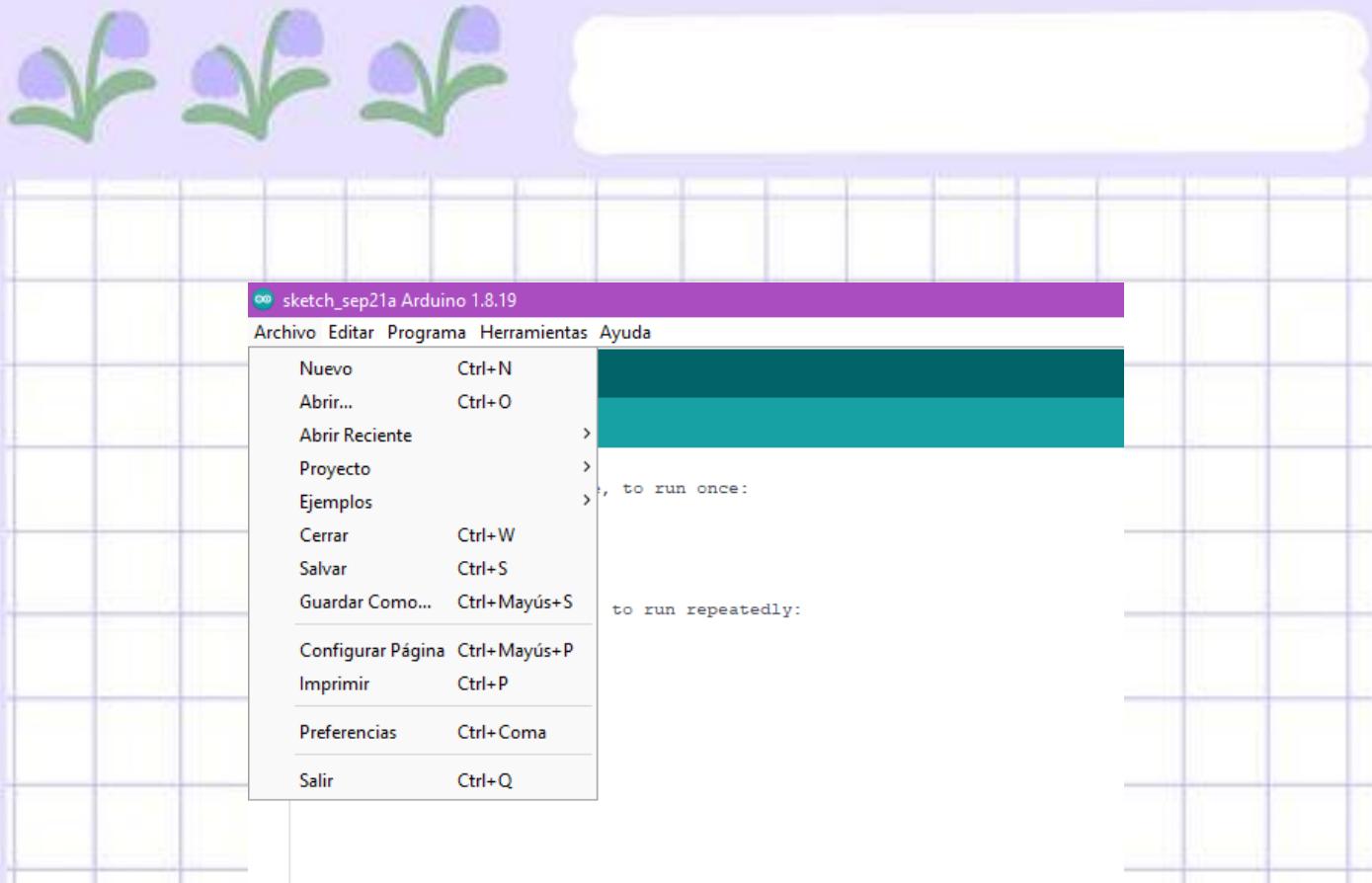
```
1 void setup() {  
2     // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7     // put your main code here, to run repeatedly:  
8  
9 }
```

Al ingresar al programa, aparece la primera ventana lista para iniciar a programar

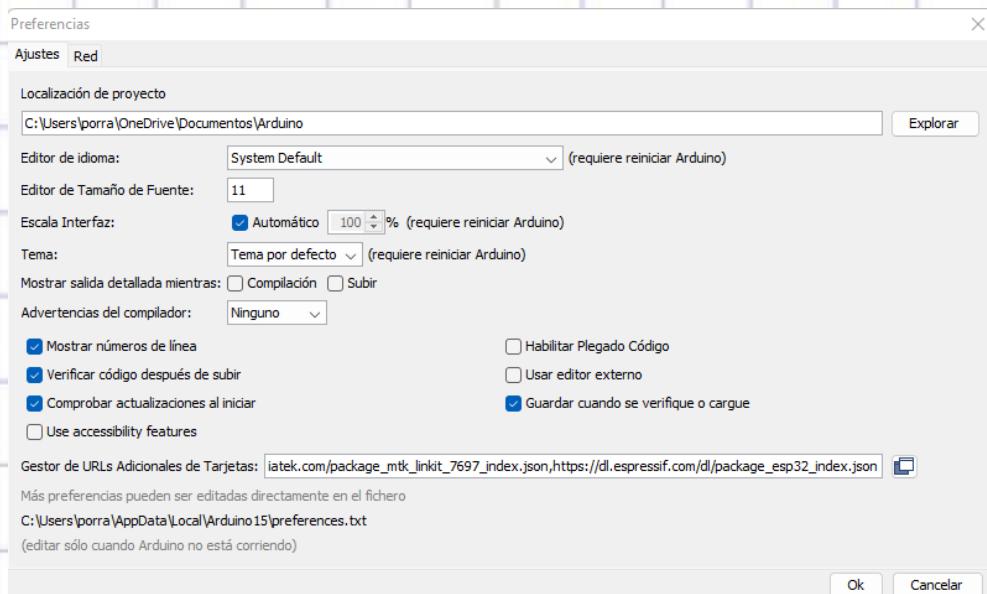
## 2.2 INTALACION DE LIBRERÍA PARA INCLUIR LA ESP32 EN EL PROGRAMA

Para la instalación de la librería de la tarjeta esp32 dentro de Arduino, debes tener en cuenta el siguiente link, el cual se encargará de incorporar la información y el código dentro del programa:  
[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json).

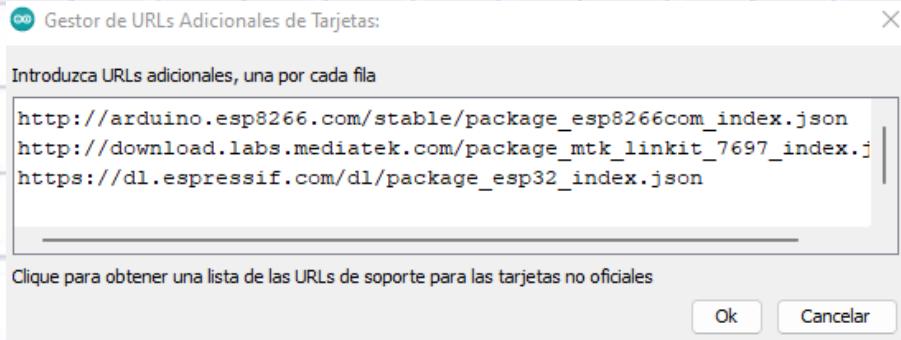
Ahora incorporarás el link de la siguiente manera. En la parte superior izquierda encontrarás un ítem nombrado ARCHIVO, debes seleccionarlo y dirigirte al ítem de preferencias.



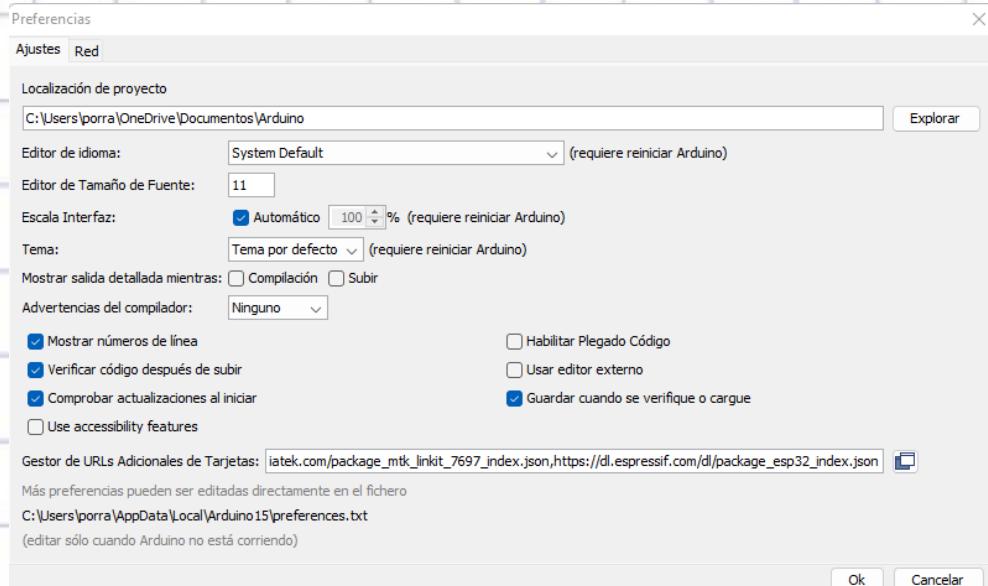
Al dar click en preferencias, el programa arroja como resultado una nueva ventana.



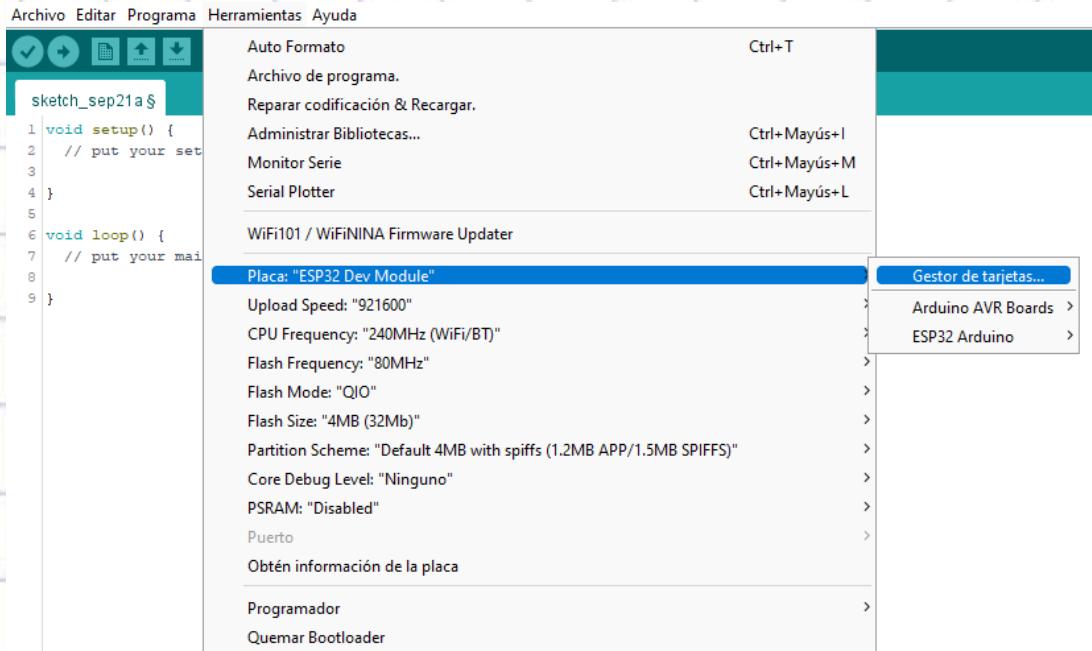
En gestor de actividades seleccionas el icono de las ventanitas ubicadas al costado derecho, al darle clic, se emitirá otra ventana, donde pegarás el link. Una vez pegado el link, damos en OK



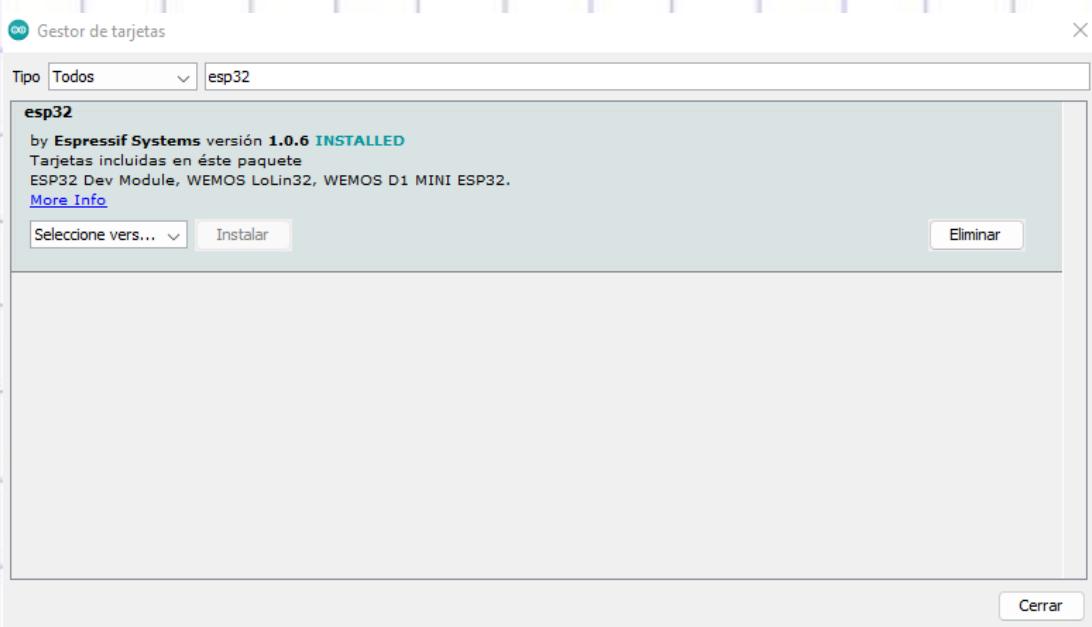
Vuelves a la página de preferencias y damos OK



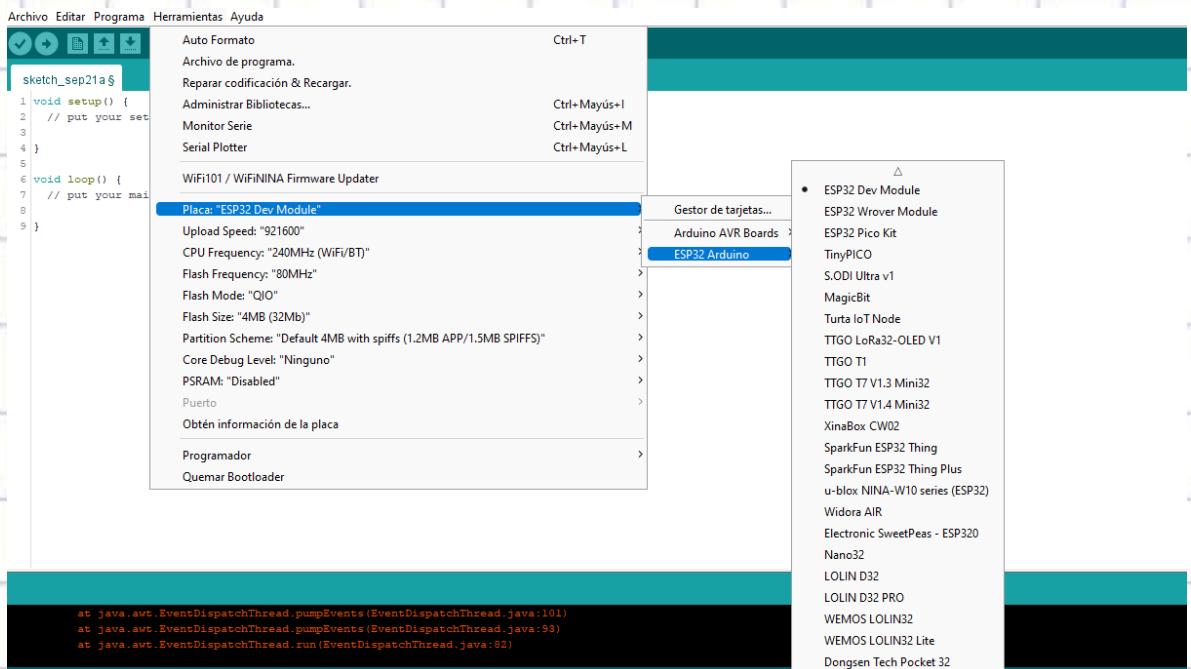
Ahora vamos al ítem de HERRAMIENTAS, luego PLACAS, y por último GESTOR DE TARJETAS. El ultimo ítem arrojará una nueva ventana



En esta nueva ventana buscarás la ESP32, si cumpliste con el primer paso del link, deberá aparecerte ya, la tarjeta. Al aparecer debes darle en la opción INSTALAR. Esperas un momento, mientras se descarga toda la información, y por último cierras la ventana.



Sigues la secuencia de la imagen, y si completaste los pasos anteriores, debe aparecerte la tarjeta ESP32 ARDUINO. Allí seleccionas el ítem de la ESP32 DEV MODULE.



Por último, conectas tu tarjeta ESP32 en el ítem herramientas, puedes observar otro ítem que nombrado PUERTO, una vez conectada tu tarjeta, en ese ítem(prueba) debe aparecerte el puerto que está utilizando tu tarjeta. Ahora solo queda cargar el programa que encontrarás más adelante junto con otras librerías necesarias para el correcto funcionamiento

## 1. INSTALACIÓN DE OTRAS LIBRERIAS PARA EL DISPENSADOR

En este ítem encontrarás el listado de librerías que debes instalar en el programa de Arduino.

Antes del listado es importante que aprendas a incluir librerías en ARDUINO. Vas a programas, seleccionas incluir librerías, administrador Bibliotecas y empiezas en la nueva ventana a incluir las librerías. Colocando cada nombre en la barra de búsqueda.



sketch\_sep21a Arduino 1.8.19

Archivo Editar Programa Herramientas Ayuda

Verificar/Compilar Ctrl+R  
Subir Ctrl+U  
Subir Usando Programador Ctrl+Mayús+U  
Exportar Binarios compilados Ctrl+Alt+S

Mostrar Carpeta de Programa Ctrl+K  
Incluir Librería  
Añadir fichero...

sketch\_sep2

```
1 void setup() {  
2     // put your setup code here, to run once:  
3 }  
4  
5 void loop() {  
6     // put your main code here, to run repeatedly:  
7 }  
8  
9 }
```

Administrador Bibliotecas... Ctrl+Mayús+I

Añadir biblioteca .ZIP...

Arduino bibliotecas

Bridge  
Esplora  
Ethernet  
Firmata  
GSM  
Keyboard

Gestor de Librerías

Tipo: Instalado Tema: Todos Filtre su búsqueda...

**Bridge**  
Built-In by Arduino Versión 1.7.0 INSTALLED  
Enables the communication between the Linux processor and the microcontroller. For Arduino/Genuino Yún, Yún Shield and TRE only. The Bridge library feature: access to the shared storage, run and manage linux processes, open a remote console, access to the linux file system, including the SD card, establish http clients or servers.  
[More info](#)

Seleccione versión ▾ Instalar

**Esplora**  
Built-In by Arduino Versión 1.0.4 INSTALLED  
Grants easy access to the various sensors and actuators of the Esplora. For Arduino Esplora only. The sensors available on the board are: 2-Axis analog joystick with center push-button, 4 push-buttons, microphone, light sensor, temperature sensor, 3-axis accelerometer, 2 TinkerKit input connectors. The actuators available on the board are: bright RGB LED, piezo buzzer, 2 TinkerKit output connectors.  
[More info](#)

**Ethernet**  
Built-In by Arduino Versión 2.0.0 INSTALLED  
Enables network connection (local and Internet) using the Arduino Ethernet Board or Shield. With this library you can use the Arduino Ethernet (shield or board) to connect to Internet. The library provides both client and server functionalities. The library permits you to connect to a local network also with DHCP and to resolve DNS.  
[More info](#)

**Firmata**  
Built-In by https://github.com/firmata/arduino Versión 2.5.8 INSTALLED  
Enables the communication with computer apps using a standard serial protocol. For all Arduino/Genuino boards. The Firmata library implements the Firmata protocol for communicating with software on the host computer. This allows you to write custom firmware without having to create your own protocol and objects for the programming environment that you are using.  
[More info](#)

**GSM**  
Built-In by Arduino Versión 1.0.6 INSTALLED  
Enables GSM/GPRS network connection using the Arduino GSM Shield. For all Arduino boards BUT Arduino DUE. Use this library to make/receive voice calls, to send and receive SMS with the Quectel M10\_GSM module. This library also allows you to connect...

Cerrar



Gestor de Librerías

Tipo: Instalado | Tema: Todos | Filtre su búsqueda...

**Keyboard**  
Built-In by Arduino Versión 1.0.2 **INSTALLED**  
**Allows an Arduino board with USB capabilities to act as a Keyboard.** This library plugs on the HID library. It can be used with or without other HID-based libraries (Mouse, Gamepad etc)  
[More info](#)

**LiquidCrystal**  
Built-In by Arduino Versión 1.0.7 **INSTALLED**  
**Allows communication with alphanumeric liquid crystal displays (LCDs).** This library allows an Arduino/Genuino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4 or 8 bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).  
[More info](#)

**Mouse**  
Built-In by Arduino Versión 1.0.1 **INSTALLED**  
**Allows an Arduino/Genuino board with USB capabilities to act as a Mouse.** This library plugs on the HID library. Can be used with or without other HID-based libraries (Keyboard, Gamepad etc)  
[More info](#)

**Robot Control**  
Built-In by Arduino Versión 1.0.4 **INSTALLED**  
**Enables easy access to the controls of the Arduino Robot Control board. For Arduino Robot only.** The Arduino robot is made by two independent boards. The Control Board is the top board of the Arduino Robot, with this library you can easily write sketches to control the robot.  
[More info](#)

**Robot IR Remote**  
Built-In by Arduino Versión 2.0.0 **INSTALLED**  
**Allows controlling the Arduino Robot via an IR remote control. For Arduino Robot only.**  
[More info](#)

[Cerrar](#)

Gestor de Librerías

Tipo: Instalado | Tema: Todos | Filtre su búsqueda...

**Robot Motor**  
Built-In by Arduino Versión 1.0.3 **INSTALLED**  
**Enables easy access to the motors of the Arduino Robot Motor board. For Arduino Robot only.**  
[More info](#)

**SD**  
Built-In by Arduino Versión 1.2.4 **INSTALLED**  
**Enables reading and writing on SD cards.** Once an SD memory card is connected to the SPI interface of the Arduino or Genuino board you can create files and read/write on them. You can also move through directories on the SD card.  
[More info](#)

**Servo**  
Built-In by Arduino Versión 1.1.8 **INSTALLED**  
**Allows Arduino/Genuino boards to control a variety of servo motors.** This library can control a great number of servos. It makes careful use of timers: the library can control 12 servos using only 1 timer. On the Arduino Due you can control up to 60 servos.  
[More info](#)

**SpacebrewYun**  
Built-In by Julio Terra Versión 1.0.2 **INSTALLED**  
**Enables the communication between interactive objects using WebSockets. For Arduino Yún only.** This library was developed to enable you to easily connect the Arduino Yún to Spacebrew. To learn more about Spacebrew visit Spacebrew.cc  
[More info](#)

**Stepper**  
Built-In by Arduino Versión 1.1.3 **INSTALLED**  
**Allows Arduino boards to control a variety of stepper motors. For all Arduino boards.** This library allows you to control unipolar or bipolar stepper motors. To use it you will need a stepper motor, and the appropriate hardware to control it.  
[More info](#)

[Cerrar](#)



**Blynk**  
by Volodymyr Shymansky Version 1.1.0 **INSTALLED**  
**Build a smartphone app for your project in minutes!** It supports WiFi, BLE, Bluetooth, Ethernet, GSM, USB, Serial. Works with many boards like ESP8266, ESP32, Arduino UNO, Nano, Due, Mega, Zero, MKR100, Yun, Raspberry Pi, Particle, Energia, ARM mbed, Intel Edison/Galileo/Joule, BBC micro:bit, DFRobot, RedBearLab, Microduino, LinkIt ONE ...  
[More info](#)

**ESP32Servo**  
by Kevin Harrington Versión 0.11.0 **INSTALLED**  
**Allows ESP32 boards to control servo, tone and analogWrite motors using Arduino semantics.** This library can control a many types of servos.  
It makes use of the ESP32 PWM timers: the library can control up to 16 servos on individual channels  
No attempt has been made to support multiple servos per channel.  
[More info](#)

**UniversalTelegramBot**  
by Brian Lough Versión 1.3.0 **INSTALLED**  
**Arduino Telegram Bot library for multiple different architectures.** A Universal Telegram library for arduino devices.  
[More info](#)

Gestor de Librerías Cerrar

Tipo **Instalado** Tema **Todos** Filtre su búsqueda...

**Temboo**  
Built-In by Temboo Versión 1.2.1 **INSTALLED**  
**This library enables calls to Temboo, a platform that connects Arduino/Genuino boards to 100+ APIs, databases, code utilities and more.** Use this library to connect your Arduino or Genuino board to Temboo, making it simple to interact with a vast array of web-based resources and services.  
[More info](#)

**TFT**  
Built-In by **Arduino** Versión 1.0.6 **INSTALLED**  
**Allows drawing text, images, and shapes on the Arduino TFT graphical display.** This library is compatible with most of the TFT display based on the ST7735 chipset.  
[More info](#) Seleccionar versión Instalar

**WiFi**  
Built-In by **Arduino** Versión 1.2.7 **INSTALLED**  
**Enables network connection (local and Internet) using the Arduino WiFi shield.** With this library you can instantiate Servers, Clients and send/receive UDP packets through WiFi. The shield can connect either to open or encrypted networks (WEP, WPA). The IP address can be assigned statically or through a DHCP. The library can also manage DNS.  
[More info](#)

**Adafruit Circuit Playground**  
Built-In by Adafruit Versión 1.11.3 **INSTALLED**  
**All in one library to control Adafruit's Circuit Playground board.** All in one library to control Adafruit's Circuit Playground board.  
[More info](#)

**ArduinoJson**  
by Benoit Blanchon Versión 6.19.4 **INSTALLED**  
**A simple and efficient JSON library for embedded C++.** ArduinoJson supports ✓ serialization, ✓ deserialization, ✓ MessagePack, ✓ fixed allocation, ✓ zero-copy, ✓ streams, ✓ filtering, and more. It is the most popular Arduino library on GitHub ❤️❤️❤️. Check out [arduinojson.org](#) for a comprehensive documentation.  
[More info](#)

Cerrar



#### Adafruit MQTT Library

by Adafruit Versión 2.4.3 **INSTALLED**

**MQTT library that supports the FONA, ESP8266, ESP32, Yun, and generic Arduino Client hardware.** Simple MQTT library that supports the bare minimum to publish and subscribe to topics.

[More info](#)

#### Adafruit SleepyDog Library

by Adafruit Versión 1.6.1 **INSTALLED**

**Arduino library to use the watchdog timer for system reset and low power sleep.** Arduino library to use the watchdog timer for system reset and low power sleep.

[More info](#)

NOTA: ES IMPORTANTE QUE VERIFIQUES SI LAS LIBRERÍAS YA EXISTEN EN TU ARDUINO. LAS LIBREIAS QUE SE INSTALARON SON PARA EL FUNCIONAMIENTO DE EL APLICATIVO WEB, PARA EL FUNCIONAMIENTO DEL CONTROL DE VOZ, Y FUNCIONAMIENTO DE ESP32 ENTRE OTROS PARAMETROS.

## EXPLICACIÓN DEL CÓDIGO EN ARDUINO PARA EL FUNCIONAMIENTO DEL DISPENSADOR IOT.

En este capítulo encontrarás todo el código de manera ordenada para el correcto funcionamiento del dispensador IOT. Cabe resaltar que se harán ciertas pautas donde deberás buscar los ítems mencionados para tener un mejor entendimiento de las líneas expuestas.

### 1. INVOCACIÓN DE LIBRERÍAS

A través del código `#include <NOMBRE DE LA LIBRERIA>` podrás incluir la librería necesaria para tu código. En la siguiente imagen, verás las librerías que incluimos.

```
1 #include <LiquidCrystal_I2C.h>
2 #include <ESP32Servo.h>
3
4 #define BLYNK_TEMPLATE_ID          "TMPLzkV54HY1"
5 #define BLYNK_DEVICE_NAME         "Quickstart Device"
6 #define BLYNK_AUTH_TOKEN          "cQD2TgFXvvEY-r0fVGfi0zKEsDIMmjuO"
7
8 #define BLYNK_PRINT Serial
9
10
11 #include <WiFi.h>
12 #include <WiFiClient.h>
13 #include <BlynkSimpleEsp32.h>
14
15 char auth[] = BLYNK_AUTH_TOKEN;
16 char ssid[] = "LOS_URIIBE"; //nombre red
17 char pass[] = "MunecaVal2022"; //contraseña
```

Como puedes observar, la imagen define otro código `#define BLYNK_TEMPLATE_ID` y otros definidos con `#define`. Para que puedas entender el contexto de esta parte del código, dirígete al capítulo de BLYNK.

En la línea de la 15 a la 17 encontramos ciertos comandos. El `char auth [] = BLYNK_AUTH_TOKEN` hace referencia al token de autenticación de Blynk y la tarjeta, el `char ssid [] = "NOMBRE RED WI-FI"` hace referencia al nombre de tu red Wi-fi, es decir, que debes poner tu red allí y el `char pass [] = "CONTRASEÑA WI-FI"` hace referencia a la contraseña, la cual deberás especificar.

## 2. CREACIÓN DE VARIABLES Y OBJETOS

```
19 LiquidCrystal_I2C lcd(0x27, 16, 2); //crear un objeto lcd (DIRECCIÓN pantalla, Tamaño x, Tamaño y)
20 Servo myservo;
21
22 int pos = 0;
23 int contador;
24 int tiempo_act; //tiempo de movimiento
25 int a = 0;
26 int b = 0;
27 int c = 0;
28 int servoPin = 32; //posición del servo - conexión
```

En la línea 19 encuentras la creación del objeto de la LCD, el cual se encarga de darle dirección a la pantalla, tamaño en el eje (X) y tamaño en el eje (Y). Luego en la línea 20 encuentras la invocación del servo, al cual se le asigna el nombre de myservo. Por último, de la línea 22 a la 28 encuentras las variables necesarias para el funcionamiento del código. Posición, contador, tiempo de movimiento, a-b-c para los ciclos y el servoPin al cual le asignamos el puerto que utilizará en la tarjeta ESP32.

## 3. PUERTOS DE BLYNK

```

30 BLYNK_WRITE(V0)
31 {
32     // Set incoming value from pin V0 to a variable
33     int value = param.asInt();
34
35     // Update state
36     Blynk.virtualWrite(V1, value);
37     a=value;
38     Serial.println(value);
39 }
40
41 BLYNK_WRITE(V2)
42 {
43     // Set incoming value from pin V0 to a variable
44     int value = param.asInt();
45
46     // Update state
47     Blynk.virtualWrite(V3, value);
48     b=value;
49     Serial.println(value);
50 }
51
52 BLYNK_WRITE(V4)
53 {
54     // Set incoming value from pin V0 to a variable
55     int value = param.asInt();
56
57     // Update state
58     Blynk.virtualWrite(V5, value);
59     c=value;

```

## 4. LIBRERÍAS Y SINCRONIZACIÓN DE ADAFRUIT

```

59     c=value;
60     Serial.println(value);
61 }
62
63 #include "Adafruit_MQTT.h"
64 #include "Adafruit_MQTT_Client.h"
65
66 #define WLAN_SSID      "LOS_URIBE"          // Your SSID
67 #define WLAN_PASS      "MunecaVal2022"        // Your password
68
69 #define AIO_SERVER     "io.adafruit.com" //Adafruit Server
70 #define AIO_SERVERPORT 1883
71 #define AIO_USERNAME    "DaironPorras"        // Username
72 #define AIO_KEY         "aio_LIVgl0uCOLj64fCCrhI8f3CkH8Pj" // Auth Key
73

```

Aquí puedes reconocer ya, la forma en que se incluyen las librerías. Ahora nos ubicamos específicamente en las líneas de la 66 a la 72, pues estas son las encargadas de la sincronización entre la tarjeta esp32 y la voz virtual. Para que entiendas mejor el funcionamiento de la voz virtual, puedes ver en el ítem final donde se especifica paso a paso la sincronización de la misma con el dispensador, Adafruit e IFTTT.

### LINEAS

66 – Debes poner el nombre de tu red Wi-fi

67 – Contraseña de tu red Wi-fi

71 – Nombre del usuario que asignaste al momento de crear una cuenta en Adafruit

72- Aquí debes poner la clave de autenticación que Adafruit te asigna al momento de crear tu cuenta, donde lo puedes ver en el ítem final de nuestro manual, donde explicaremos el desarrollo de la voz.

Te diriges al ícono de la llave amarilla, y ella te mostrará el clave de acceso de tu cuenta,

## TU LLAVE ADAFRUIT IO

Su Adafruit IO Key debe guardarse en un lugar seguro y tratarse con el mismo cuidado que su nombre de usuario y contraseña de Adafruit. Las personas que tienen acceso a su Adafruit IO Key pueden ver todos sus datos, crear nuevos feeds para su cuenta y manipular sus feeds activos.



Si necesita regenerar una nueva clave de E/S de Adafruit, todos sus programas y scripts existentes deberán cambiarse manualmente a la nueva clave.

Nombre de usuario

DaironPorras

Clave activa

aio\_LIVg10uC0Lj64fCCrhI8f3CkH8Pj

**REGENERAR CLAVE**

Ocultar muestras de código

arduino

```
#define IO_USERNAME "DaironPorras"  
#define IO_KEY "aio_LIVg10uC0Lj64fCCrhI8f3CkH8Pj"
```

## 5. WI-FI CLIENT

```
74 //WIFI CLIENT
75 WiFiClient client;
76
77 Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
78
79 Adafruit_MQTT_Subscribe Light1 = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME"/feeds/Med_1"); // Feeds name should be same everywhere
80 Adafruit_MQTT_Subscribe Light2 = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/Med_2");
81 Adafruit_MQTT_Subscribe Light3 = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/Med_3");
82
83
84 void MQTT_connect();
85
```

Enfocaremos nuestra atención en esta parte del código:

79 - Adafruit\_MQTT\_Subscribe Light1 = Adafruit\_MQTT\_Subscribe (&mqtt,  
AIO\_USERNAME "/feeds/Med\_1");

80 - Adafruit\_MQTT\_SubscribeLight2 = Adafruit\_MQTT\_Subscribe (&mqtt,  
AIO\_USERNAME "/feeds/Med\_2");

81 - Adafruit\_MQTT\_SubscribeLight3 = Adafruit\_MQTT\_Subscribe (&mqtt,  
AIO\_USERNAME "/feeds/Med\_3");

Tomaremos como ejemplo la línea 79, en ella encontrarás l forma de invocar la comunicación MQTT, seguido lo sombrado con azul representa el nombre de esa comunicación. Ahora en los paréntesis lo sombrado con rosa representa tu NOMBRE DE USARIO en Adafruit y lo sombreado representa la dirección que toma la comunicación, es decir, debe ir a Adafruit, entrar en el ítem feeds, y ubicarse en el Feed llamado Med\_1 – el Med\_1 es el nombre del feed que asignas en Adafruit, y toma el valor del primer pulsador del proyecto.

## 6. CÓDIGO PARA FUNCIONAMIENTO DEL SERVO Y LCD – BOTONES

```
87 void setup() {
88
89   ESP32PWM::allocateTimer(0);
90   ESP32PWM::allocateTimer(1);
91   ESP32PWM::allocateTimer(2);
92   ESP32PWM::allocateTimer(3);
93   myservo.setPeriodHertz(50);      // standard 50 hz servo
94   myservo.attach(servopin, 500, 2400); //señal para controlar el servo frecuencia 500 periodo 2400
95
96   lcd.begin();
97   lcd.backlight(); //Encender la luz de fondo
98
99   pinMode(23, INPUT); //botones
00   pinMode(19, INPUT); //botones
01   pinMode(18, INPUT); //botones
02
```

Aquí damos la orden a través del comando de la línea 96 – 97 iniciar y encender nuestra pantalla. Si realizaste bien tus conexiones debe encender tu LCD sin ningún inconveniente. La línea 99 a la 101 representan la activación de pines que usaremos de la esp32 en nuestra protoboard para los botones Med\_1 = pin 23, Med\_2= pin 19 y Med\_3=pin 18.

## 7. INICIAR BLINK – INICIAR CONEXIÓN WI-FI

```
103 Serial.begin(115200);
104 Blynk.begin(auth, ssid, pass);
105
106 Serial.println();
107 Serial.print("Connecting to ");
108 Serial.println(WLAN_SSID);
109
110 WiFi.begin(WLAN_SSID, WLAN_PASS);
111 while (WiFi.status() != WL_CONNECTED) {
112     delay(500);
113     Serial.print(".");
114 }
115 Serial.println();
116
117 Serial.println("WiFi connected");
118 Serial.println("IP address: ");
119 Serial.println(WiFi.localIP());
120
121 mqtt.subscribe(&Light1);
122 mqtt.subscribe(&Light3);
123 mqtt.subscribe(&Light2);
124
125 }
```

En esta parte del código iniciamos a través de diferentes comandos la conexión entre los dispositivos virtuales y electrónicos

## 8. FUNCIONAMIENTO MANUAL – VIRTUAL DE LOS BOTONES

```
130 void loop() {
131   Blynk.run();
132
133   if (digitalRead(23)) {
134     a=1;
135   }
136
137   if (digitalRead(19)) {
138     b=1;
139   }
140   if (digitalRead(18)) {
141     c=1;
142   }
143
144   MQTT_connect();
145
146   Serial.print("a: ");
147   Serial.println(a);
148   Serial.print("b: ");
149   Serial.println(b);
150   Serial.print("c: ");
151   Serial.println(c);
```

Aquí indicamos a Blynk que empiece a funcionar, luego damos valor a las variables a través del uso de los ciclos IF.

#### 9. DISEÑO DEL MENÚ E IMPRESIÓN EN PANTALLA LCD

Sabemos que la pantalla LCD está conformada por 2 filas y 16 columnas. En la primera fila hacemos la impresión SELECC MEDIDA – LINEA 155 y en la segunda fila imprimimos los tipos de medidas dentro del circuito línea 157 - 159

```
153 //Serial.println(digitalRead(23));
154 lcd.setCursor (0, 0);//poner el cursor en las coordenadas (x,y)
155 lcd.print(" Selec medida ");//muestra en la pantalla max 20 caracteres
156 lcd.setCursor (0, 1);//poner el cursor en las coordenadas (x,y)
157 lcd.print("Med1 ");
158 lcd.print("Med2 ");
159 lcd.print("Med3");
160
```

#### 10. COMUNICACIÓN ADAFRUIT - VOZ

```

161 Adafruit_MQTT_Subscribe *subscription;
162 while ((subscription = mqtt.readSubscription(1000))) {
163   if (subscription == &Light1) {
164     Serial.print(F("Got: "));
165     Serial.println((char *)Light1.lastread);
166     int Light1_State = atoi((char *)Light1.lastread);
167     Serial.print("variable1: ");
168     Serial.println(Light1_State);
169     a=1;
170     b=0;
171     c=0;
172   }
173   if (subscription == &Light2) {
174     Serial.print(F("Got: "));
175     Serial.println((char *)Light2.lastread);
176     int Light2_State = atoi((char *)Light2.lastread);
177     Serial.print("variable2: ");
178     Serial.println(Light2_State);
179     a=0;
180     b=1;
181     c=0;
182   }
183   if (subscription == &Light3) {
184     Serial.print(F("Got: "));
185     Serial.println((char *)Light3.lastread);
186     int Light3_State = atoi((char *)Light3.lastread);
187     Serial.print("variable3: ");
188     Serial.println(Light3_State);
189     a=0;
190     b=0;
191     c=1;
192   }
193 }
194
195
196   Serial.print("a: ");
197   Serial.println(a);
198   Serial.print("b: ");
199   Serial.println(b);
200   Serial.print("c: ");
201   Serial.println(c);

```

La forma de comunicación es muy sencilla, mientras en el primer if el valor de A sea igual a 1 , las otras variables deben hacerse cero, esto con el fin de que sólo se active A que en este caso representa a la Medida 1. Lo mismo pasará con los otros if que representan la medida 2 y medida 3 respectivamente.

- La medida 1 en google asistente se activará con el comando de voz :  
ACTIVAR PEQUEÑO
- La medida 2 en google asistente se activará con el comando de voz :  
ACTIVAR MEDIANO
- La medida 3 en google asistente se activará con el comando de voz :  
ACTIVAR GRANDE

## 11. FUNCIONAMIENTO PARTE MANUAL Y APP DE LOS BOTONES

```
203 if(digitalRead(23) || a==1){  
204     pos=60;  
205     tiempo_act=12000;  
206     lcd.setCursor (0, 0);  
207     lcd.print("Medida 1      ");  
208     conteo();  
209     Vis_Activado();  
210     a=0;  
211 }  
212  
213 if(digitalRead(19) || b==1){  
214     pos=70;  
215     tiempo_act=14000;  
216     lcd.setCursor (0, 0);  
217     lcd.print("Medida 2      ");  
218     conteo();  
219     Vis_Activado();  
220     b=0;  
221 }  
222  
223 if(digitalRead(18) || c==1){  
224     pos=89;  
225     tiempo_act=18000;  
226     lcd.setCursor (0, 0);  
227     lcd.print("Medida 3      ");  
228     conteo();  
229     Vis_Activado();  
230     c=0;  
231 }  
232 }
```

- En el primer if de esta sección, vemos que, si el botón es pulsado en el pin 23 o si en la app A toma el valor de 1, el programa debe ejecutar la siguiente función: su posición cambiará de cero a 60, el tiempo de activación será de 12 segundos, se imprimirá en pantalla la medida 1 a través de un método llamado conteo e imprimirá de nuevo que la medida 1 fue activada. Por último, regresa el valor de la variable a cero para que pueda ejecutarse otra actividad.
- En el segundo if de esta sección, vemos que, si el botón es pulsado en el pin 19 o si en la app B toma el valor de 1, el programa debe ejecutar la siguiente función: su posición cambiará de cero a 70, el tiempo de activación será de 14 segundos, se imprimirá en pantalla

la medida 2 a través de un método llamado conteo e imprimirá de nuevo que la medida 1 fue activada. Por último, regresa el valor de la variable a cero para que pueda ejecutarse otra actividad.

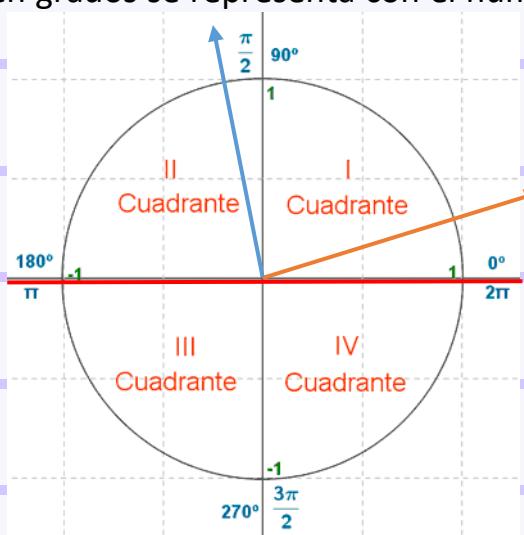
- En el tercer if de esta sección, vemos que, si el botón es pulsado en el pin 18 o si en la app C toma el valor de 1, el programa debe ejecutar la siguiente función: su posición cambiará de cero a 89, el tiempo de activación será de 18 segundos, se imprimirá en pantalla la medida 1 a través de un método llamado conteo e imprimirá de nuevo que la medida 1 fue activada. Por último, regresa el valor de la variable a cero para que pueda ejecutarse otra actividad.

```
234 void MQTT_connect() {  
235     int8_t ret;  
236  
237     if (mqtt.connected()) {  
238         return;  
239     }  
240  
241     Serial.print("Connecting to MQTT... ");  
242  
243     uint8_t retries = 3;  
244  
245     while ((ret = mqtt.connect()) != 0) {  
246         Serial.println(mqtt.connectErrorString(ret));  
247         Serial.println("Retrying MQTT connection in 5 seconds...");  
248         mqtt.disconnect();  
249         //delay(5000);  
250         delay(500);  
251         retries--;  
252         if (retries == 0) {  
253             while (1);  
254         }  
255     }  
256     Serial.println("MQTT Connected!");  
257  
258 }
```

## 12. MÉTODOS CONTEO, MOVER\_SERVO Y ACTIVACIÓN

```
260 void conteo(){
261     for (contador=5;contador>0;contador--){
262         lcd.setCursor (0, 1);
263         lcd.print("          ");
264         lcd.setCursor (0, 1);
265         lcd.print("tiempo ");
266         lcd.print(contador);
267         delay(1000);
268     }
269 }
270
271 void Mover_Servo(){
272     myservo.write(pos);
273     delay(tiempo_act);
274     myservo.write(90);
275 //delay(3000);
276 }
277
278 void Vis_Activado(){
279     lcd.setCursor (0, 0);
280     lcd.print("Activado      ");
281     lcd.setCursor (0, 1);
282     lcd.print("      ");
283     delay(3000);
284     Mover_Servo();
285     lcd.setCursor (0, 0);
286     lcd.print("      ");
287 }
```

- En el método conteo, generamos un contador de va de 5 a 0 segundos. Este, muestra en la pantalla LCD el tiempo que tardará en ejecutarse la orden del usuario.
- El método mover servo se activa a través de la invocación y el cambio de posición del mismo. Es decir, el servo está en una posición inicial de cero que en grados se representa con el número 90, si el usuario



genera una petición, el valor de la posición debe cambiar para que el

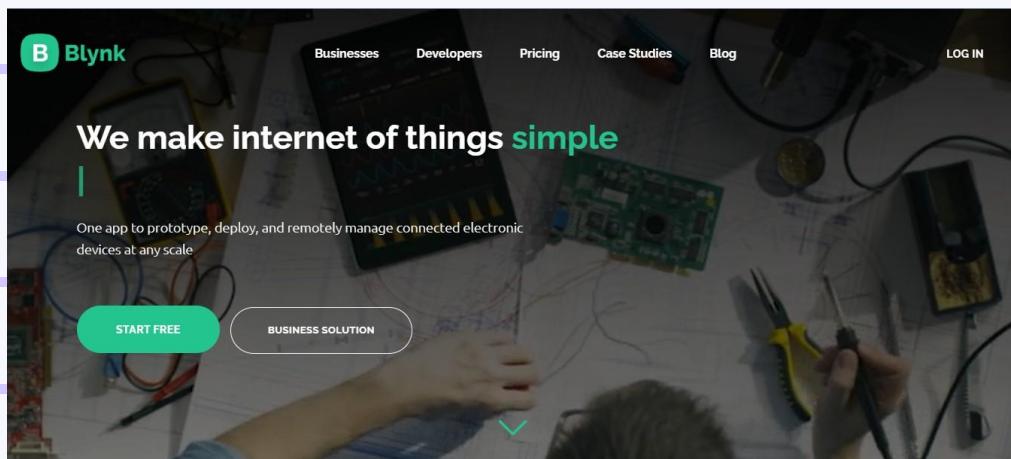
servo se empiece a mover. Cabe resaltar que el servo se mueve en una posición angular dentro de Arduino. Es decir, de 0 a 180 grados. Si en alguno de los dos cuadrantes está más cerca del 90 su movimiento será más lento, pero si está más lejos del 90 su movimiento será más rápido. Ahora bien, si utilizamos el cuadrante 1, es decir, de 90 a 0 el movimiento será de manera HORARIO, pero si implementamos el cuadrante 2 el movimiento será ANTI-HORARIO.

- El método de activación es simplemente ejecutar una impresión por cierto tiempo cuando el usuario ejecute una acción dentro del circuito, una vez realice la petición, el mensaje emitido será el de ACTIVADO.

## PASO A PASO DE LA INSTALACION DE BLINK IoT Y CONEXIÓN A LA PLATAFORMA ARDUINO Y TARJETA ESP32

En este ítem, explicaremos los pasos a realizar para la instalación de Blynk y su registro, también como enlazar Blynk con Arduino y su sincronización con la tarjeta ESP32. Recuerda que en estos pasos a realizar se debe tener conectada la tarjeta ESP32 a nuestro dispositivo para que la información sea cargada a la tarjeta y realizar la programación en Arduino, así todo sea exitoso.

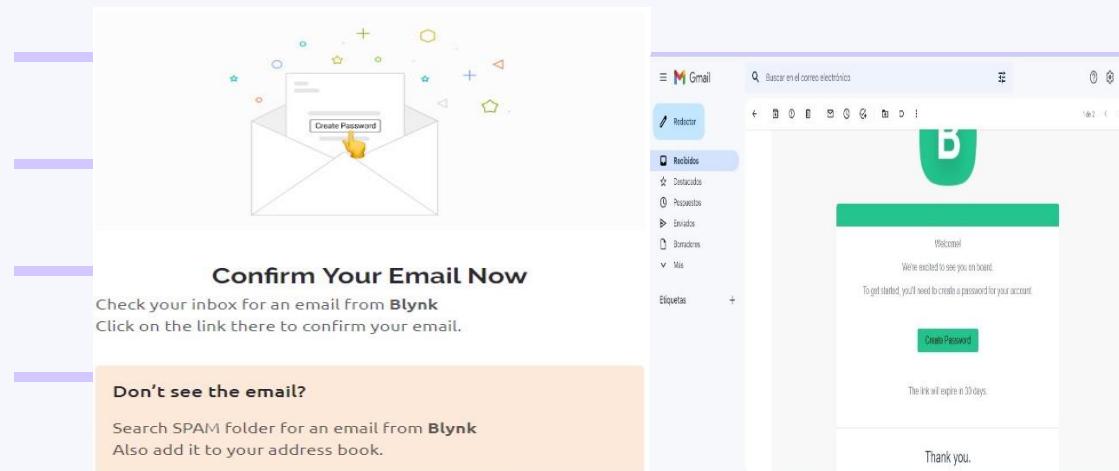
1. Ingresar a la página oficial de BLINK (<https://blynk.io/>)



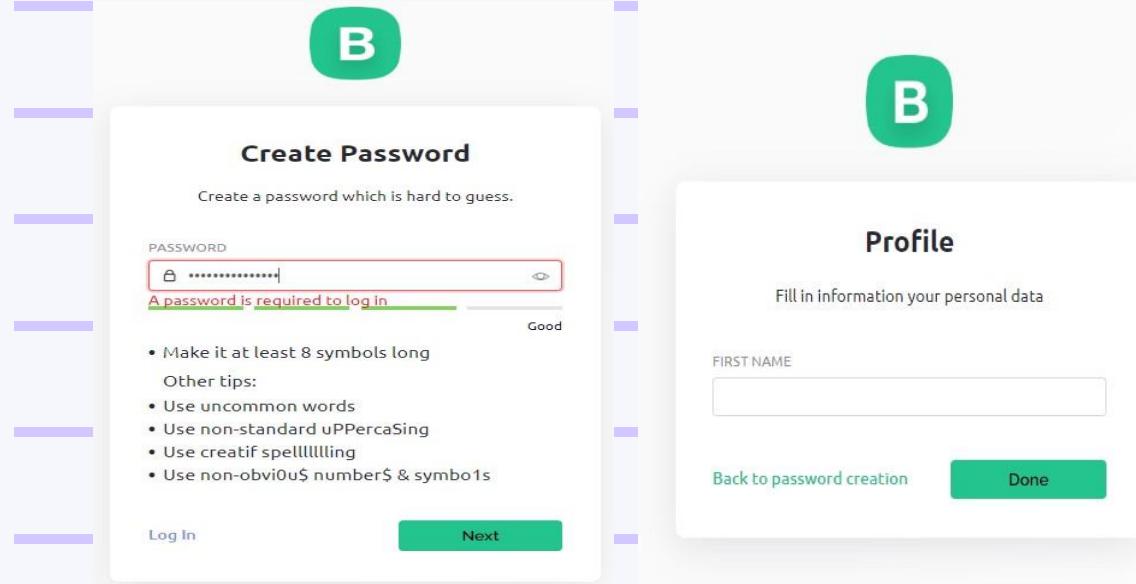
2. Proseguimos a registrarnos en la página de Blynk, creando una cuenta desde 0. Lo primero es ingresar un correo electrónico de su uso.

A screenshot of the Blynk Sign Up form. The form is titled "Sign Up" and includes a welcome message: "Welcome! Fill in your email address and we will send an account activation link." There is a large input field labeled "EMAIL" with an envelope icon, a checkbox for "I agree to Terms and Conditions and accept Privacy Policy", and a green "Sign Up" button. Below the button is a "Back to Login" link.

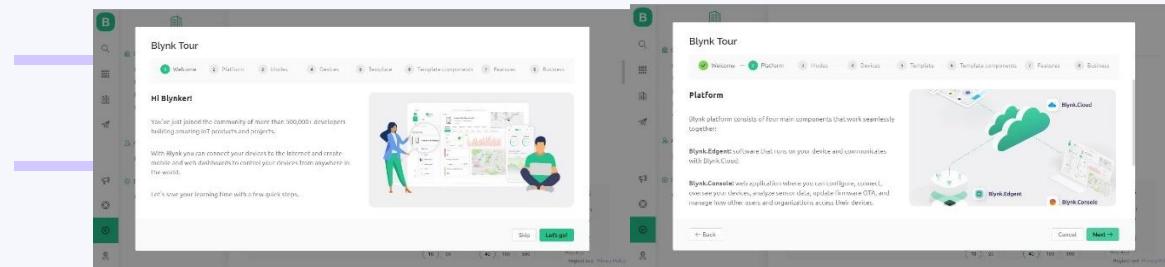
### 3. Se espera un correo de confirmación, vía correo electrónico para continuar con el proceso de registro

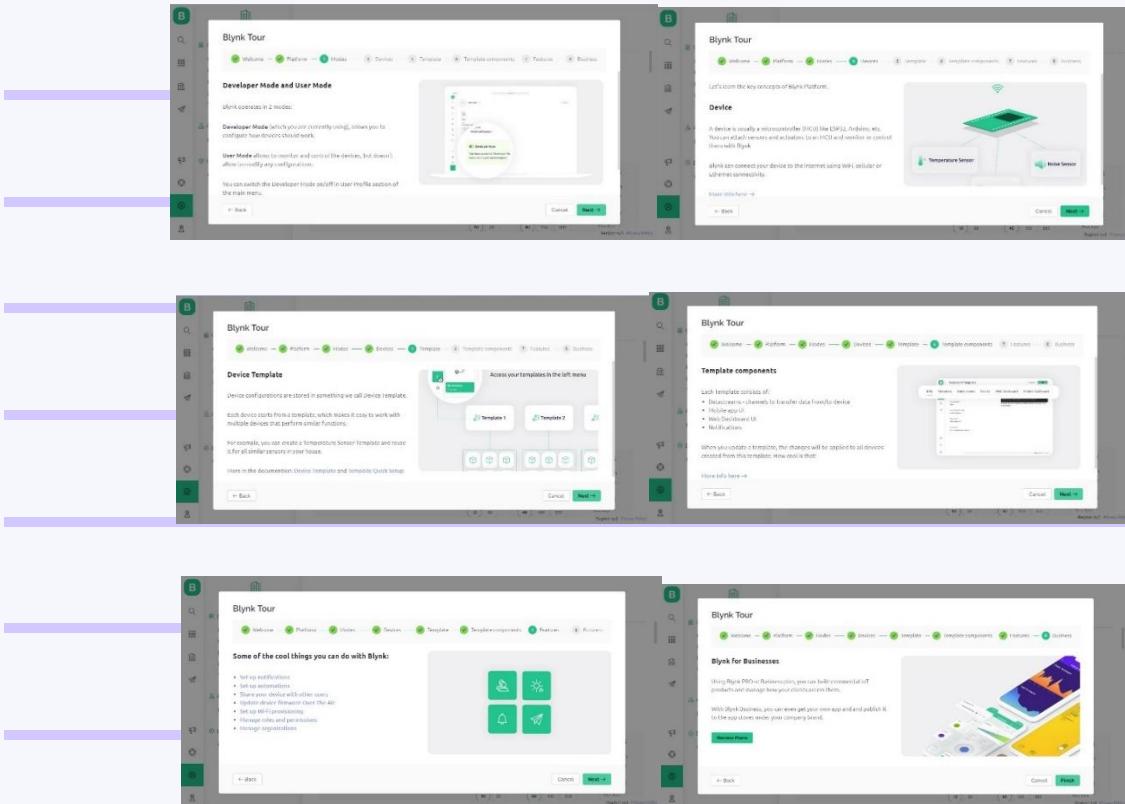


### 4. El siguiente paso es ingresar una contraseña y nombre de usuario para completar los pasos de registro en Blynk.

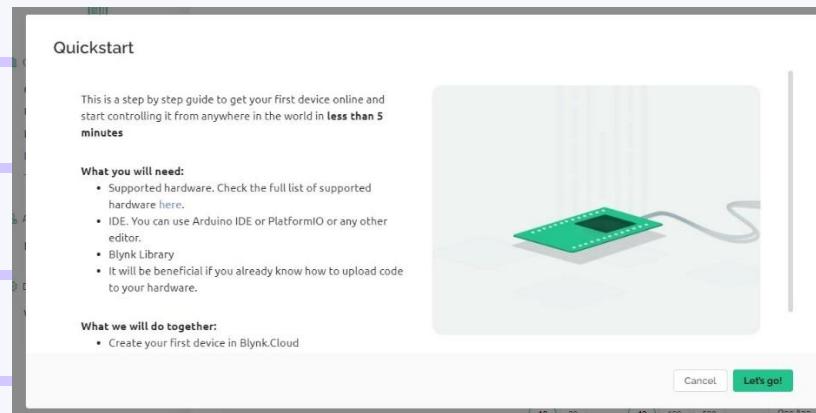


### 5. Ya registrados en la página de Blynk, nos mostrara una serie de indicaciones que leeremos atentamente y seguiremos avanzando.

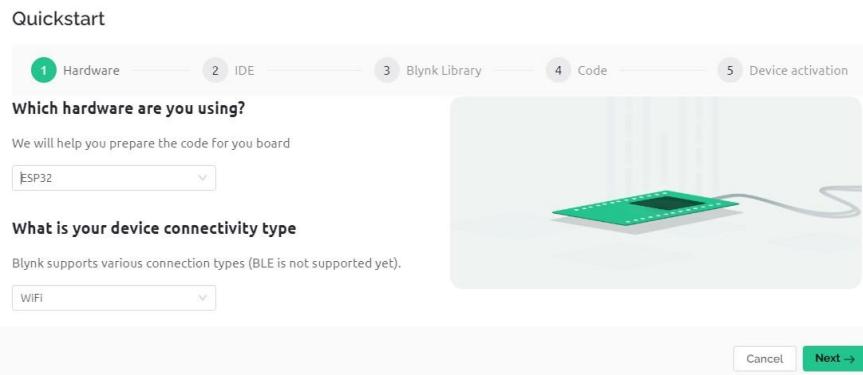




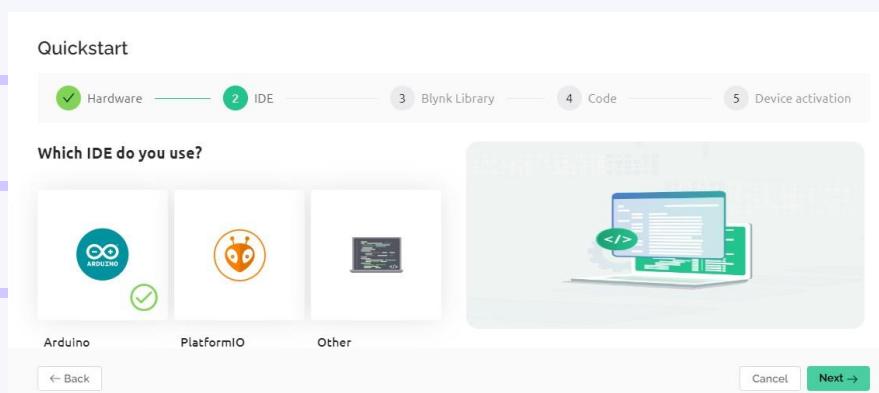
## 6. Terminado lo anterior proseguimos a realizar las conexiones con la tarjeta para obtener la sincronización con ella misma.



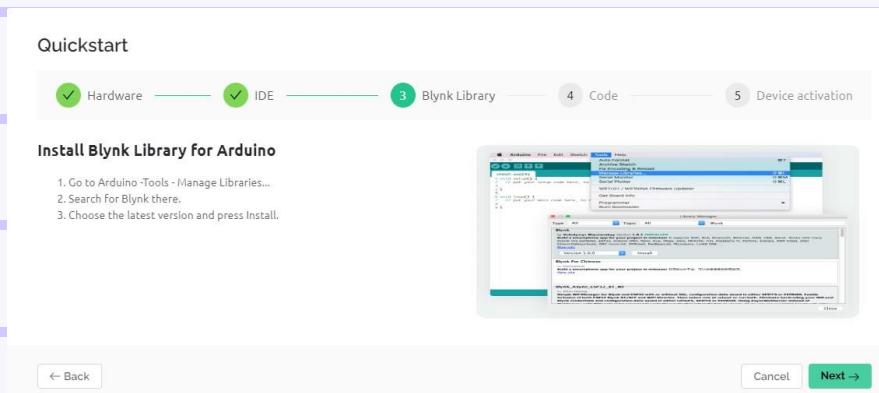
## 7. En la imagen nos muestra que debemos indicar el tipo de tarjeta que estamos trabajando, en este caso ESP32 y el tipo de conexión que será a través de Wifi



8. El siguiente paso se debe indicar el tipo de software con el cual se va a trabajar. En este caso indicamos ARDUINO.

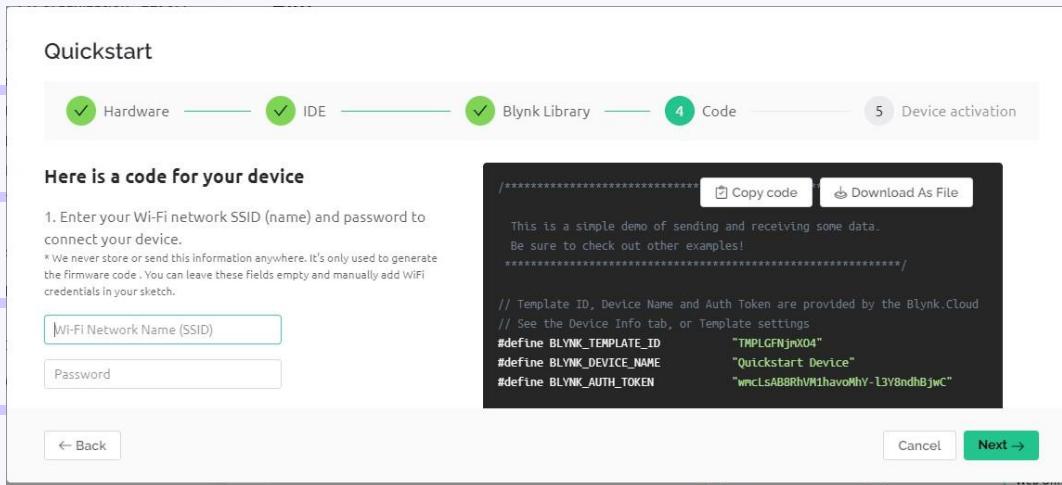


9. Lo siguiente es instalar las librerías de Blynk que se van a necesitar en el programa de ARDUINO, para realizar la conexión, paso que se realiza en la programación de ARDUINO.

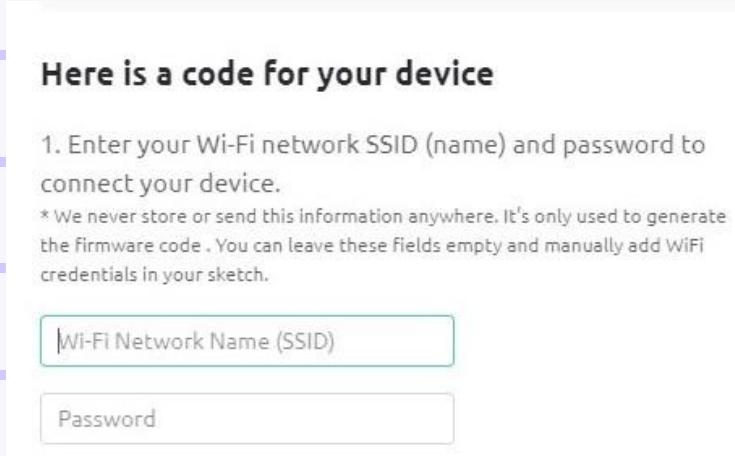


10. En la siguiente imagen nos indica a realizar dos pasos, el cual el primero es ingresar el nombre de la red y conectarla. Y descargar el

código que nos muestra en la pantalla negra, e ingresarla en la plataforma de Arduino para realizar la conexión.



- En la primera parte de este paso, es ingresar el nombre y contraseña de la red wifi a la que se va a conectar la tarjeta Arduino, por la cual recibirá conexión a internet.



- En la segunda parte de este paso, debemos copiar o descargar el código que se encuentra en el recuadro negro, el cual usaremos en un nuevo archivo que vamos a crear en la plataforma de Arduino.

```

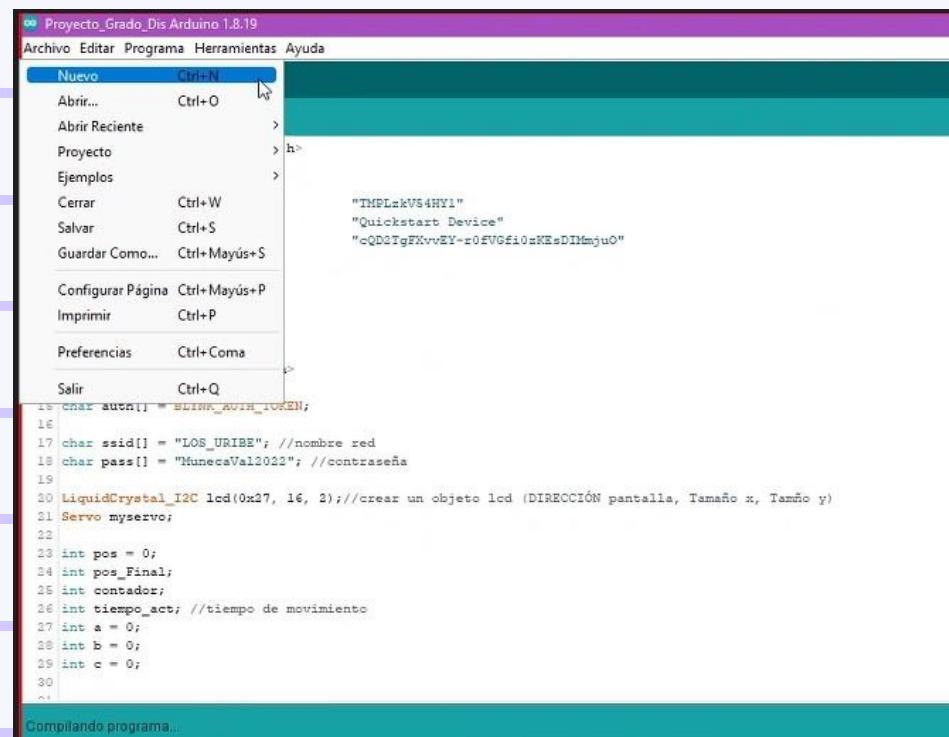
/*
This is a simple demo of sending and receiving some data.
Be sure to check out other examples!
*/
// Template ID, Device Name and Auth Token are provided by the Blynk.Cloud
// See the Device Info tab, or Template settings
#define BLYNK_TEMPLATE_ID      "TMPLGFNjmX04"
#define BLYNK_DEVICE_NAME      "Quickstart Device"
#define BLYNK_AUTH_TOKEN        "wmcLsAB8RhVM1havoMhY-13Y8ndhBjwC"

```

Cancel

Next →

- Nos dirigimos a nuestro proyecto en Arduino y buscamos en la parte superior la opción “Archivo” y luego “Nuevo” ahí crearemos un nuevo archivo dentro de nuestro proyecto que se llamará Blynk



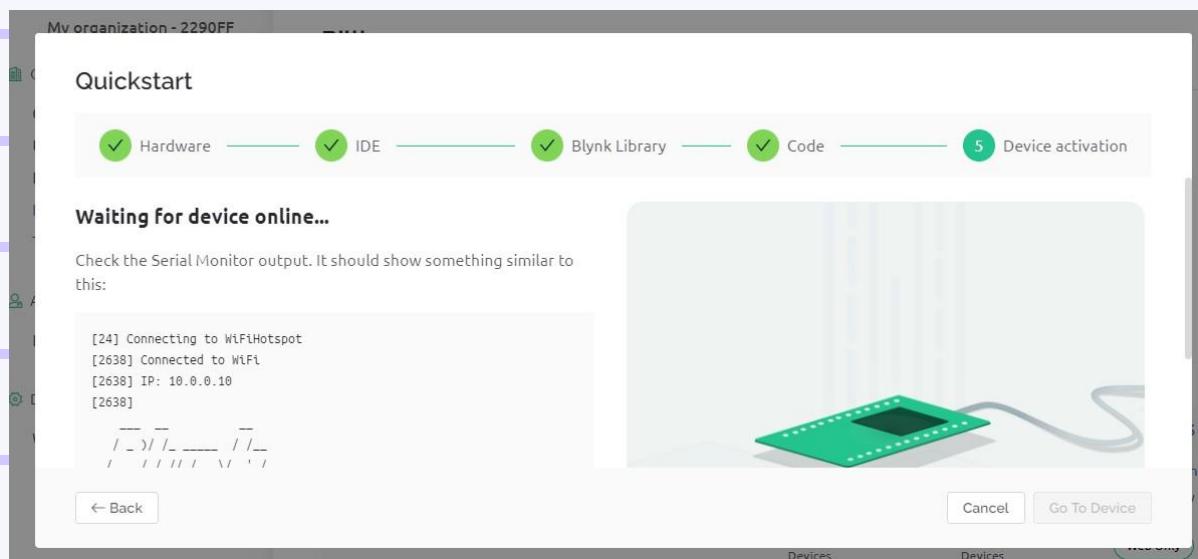
- Dentro de ese nuevo archivo vamos a copiar el código que descargamos anteriormente.
- Buscamos las siguientes líneas en nuestro código:

```
L7 char ssid[] = "LOS_URIBE"; //nombre red
L8 char pass[] = "MunecaVal2022"; //contraseña
```

Y ahí también debemos ingresar el nombre y contraseña de red a la cual se conectará nuestra tarjeta.

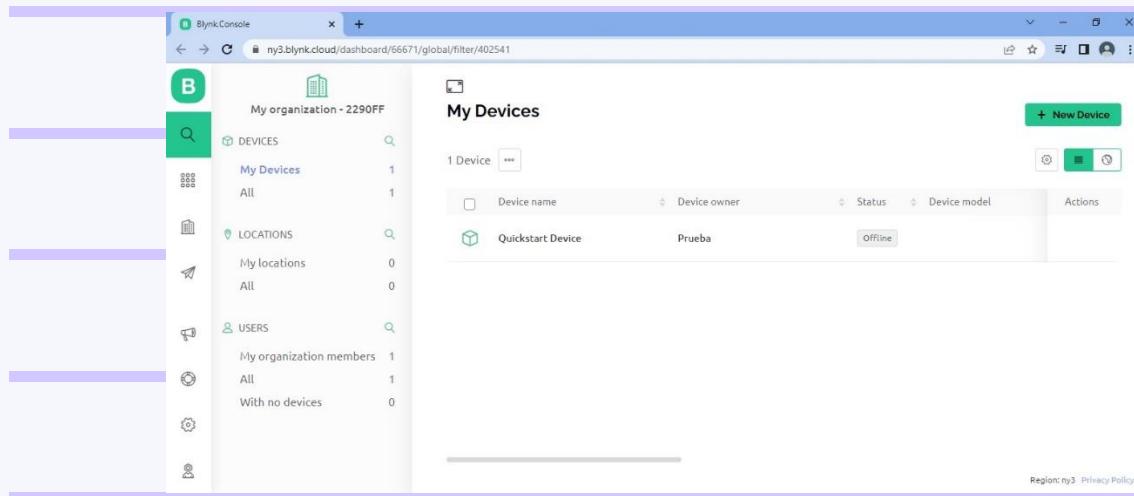
- Guardamos y cargamos todo a nuestra tarjeta para que se realice la programación

11. Luego de haber cumplido satisfactoriamente con los pasos anteriores lo último por realizar es esperar que la tarjeta se sincronice con Blynk y así ya empezar el desarrollo de nuestro aplicativo

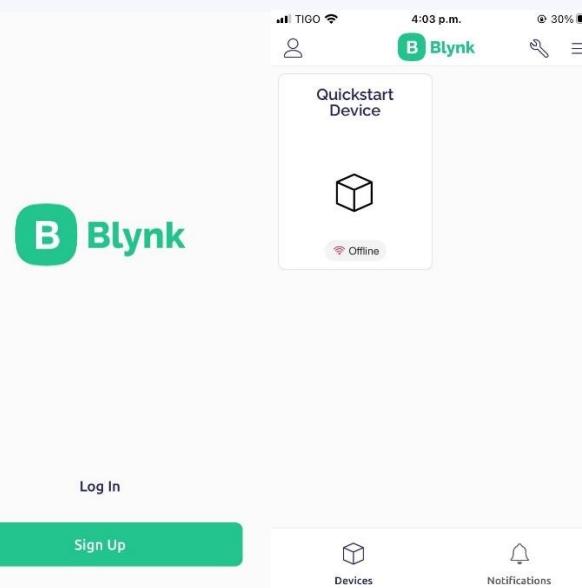


## CREAR APLICACIÓN ONLINE EN BLINK-PC O CELULAR

1.Ya luego de haber sincronizado nuestra tarjeta Arduino a Blynk proseguimos a crear nuestro aplicativo de manera web, lo primero es estar en el área de trabajo de Blynk, donde crearemos nuestro proyecto. Blynk nos ofrece un programa como ejemplo (prueba)



Un paso importante a tener en cuenta es descargar la aplicación de Blynk en nuestro teléfono e iniciar también sección con la cuenta creada. La encontramos en la play store, o app store. Nos registramos con el mismo correo que anteriormente creamos



Así ya podremos tener control desde el teléfono también.

2. En este paso debemos añadir nuevo código al código que ya tenemos en Arduino.

Ingresamos a nuestro software de Arduino y en la parte superior vamos a añadir las siguientes líneas de código. Buscamos el archivo que copiamos anteriormente, al que llamamos Blynk y copiamos las siguientes líneas de código.

```
#define BLYNK_TEMPLATE_ID      "TMPLGFNjmX04"
#define BLYNK_DEVICE_NAME       "Quickstart Device"
#define BLYNK_AUTH_TOKEN        "wmcLsAB8RhVM1havoMhY-13Y8ndhBjwC"

// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "";
char pass[] = "";
```

Las pegamos en la parte superior del código, de esta forma

```
Proyecto_Grado_Dis
1 #include <LiquidCrystal_I2C.h>
2 #include <ESP32Servo.h>
3
4 #define BLYNK_TEMPLATE_ID      "TMPLzkV54HY1"
5 #define BLYNK_DEVICE_NAME       "Quickstart Device"
6 #define BLYNK_AUTH_TOKEN        "cQD2TgFXvvEY-r0fVGfi0zKEsDIMmjuO"
7
8 #define BLYNK_PRINT Serial
9
10
11 #include <WiFi.h>
12 #include <WiFiClient.h>
13 #include <BlynkSimpleEsp32.h>
14
15 char auth[] = BLYNK_AUTH_TOKEN;
16
17 char ssid[] = "LOS_URIBE"; //nombre red
18 char pass[] = "MunecaVal2022"; //contraseña
19
20 LiquidCrystal_I2C lcd(0x27, 16, 2); //crear un objeto lcd (DIRECCIÓN pantalla
21 Servo myservo;
22
```

Recuerda que el código que descargaste es diferente al de este proyecto, ya que Blynk ofrece códigos diferentes para todos, pero con la misma

funcionalidad. También recuerda poner tu nombre de red y contraseña en la línea 17 y 18.

También creamos 3 nuevas variables en nuestro código de Arduino, a b y c, y las igualamos a 0. Estas variables serán el valor de nuestros pulsadores en la aplicación para que hagan la misma función que los pulsadores manuales.

```
17 //const char SSID[] = "www_yourname_is_myname";
18 char pass[] = "MunecaVal2022"; //contraseña
19
20 LiquidCrystal_I2C lcd(0x27, 16, 2); //crear un objeto LCD
21 Servo myservo;
22
23 int pos = 0;
24 int pos_Final;
25 int contador;
26 int tiempo_act; //tiempo de movimiento
27 int a = 0;
28 int b = 0;
29 int c = 0;
```

Continuamos añadiendo nuevo código, lo siguiente añadir irá todo antes de nuestro void setup (),

Este código nos indica los pines que se estarán utilizando Blynk para realizar las conexiones de los botones

```
32 int servoPin = 32; //posición del servo - conexión
33
34 BLYNK_WRITE(V0)
35 {
36     // Set incoming value from pin V0 to a variable
37     int value = param.toInt();
38
39     // Update state
40     Blynk.virtualWrite(V1, value);
41     //digitalWrite(19,value);
42     a=value;
43     Serial.println(value);
44 }
45
46 BLYNK_WRITE(V2)
47 {
48     // Set incoming value from pin V0 to a variable
49     int value = param.toInt();
50
51     // Update state
52     Blynk.virtualWrite(V3, value);
53     //digitalWrite(19,value);
54     b=value;
55     Serial.println(value);
56 }
```

```

57
58 BLYNK_WRITE(V4)
59 {
60     // Set incoming value from pin V0 to a variable
61     int value = param.asInt();
62
63     // Update state
64     Blynk.virtualWrite(V5, value);
65     //digitalWrite(19,value);
66     c=value;
67     Serial.println(value);
68 }
69
70
71 void setup() {
72
73     ESP32PWM::allocateTimer(0);
74     ESP32PWM::allocateTimer(1);
75     ESP32PWM::allocateTimer(2);
76     ESP32PWM::allocateTimer(3);
77     myservo.setPeriodHertz(50);    // standard 50 hz servo

```

El siguiente código lo agregamos ya dentro de nuestro void setup ()

```

83     pinMode(23,INPUT); //botones
84     pinMode(19,INPUT); //botones
85     pinMode(18,INPUT); //botones
86
87     Serial.begin(115200);
88     Blynk.begin(auth, ssid, pass);
89 }
90
91
92 void loop() {
93     Blynk.run();
94

```

Dentro nuestro void loop (), añadimos la siguiente línea de código, esta línea nos permitirá arrancar o ejecutar el Blynk.

```

void loop() {
    Blynk.run();

    //Serial.println(digitalRead(23));
    lcd.setCursor (0, 0);//poner el cursor en las co
    lcd.print(" Selec medida ");//muestra en la pa
    lcd.setCursor (0, 1);//poner el cursor en las co
    lcd.print("Med1 ");
    lcd.print("Med2 ");
    lcd.print("Med3");
}

```

Ahora buscamos nuestros condicionales, los cuales tendrán un breve cambio.

- El cual consta de ingresar dentro de nuestro condicional `|| a==1`, lo que quiere decir que, si al leer el pin 23 da 1, o si (a) es igual a 1 toma el valor 1, se ejecute la operación.
- Lo otro a añadir es antes de cerrar el condicional, que es colocar la variable (a) igualada a (0), `a=0`; Lo cual cumple con la función de cuando se oprima el botón en Blynk, la (a) se vuelve 1, así permitiendo que entre, para que haga la misma acción si se oprime el botón desde la protoboard. Y al terminar el proceso la (a) regresa a ser (0) para que se desactive y no se ejecute de nuevo el proceso.
- Lo mismo se realiza con cada una de las otras variables (b) y (c) como se muestra en las siguientes imágenes

```
if(digitalRead(23) || a==1){  
    //pos_Final = 60;  
    pos=70;  
    tiempo_act=12000;  
    lcd.setCursor (0, 0);  
    lcd.print("Medida 1      ");  
    conteo();  
    Vis_Activado();  
    a=0;  
}  
  
if(digitalRead(19) || b==1){  
    //pos_Final = 120;  
    pos=150;  
    tiempo_act=4000;  
    lcd.setCursor (0, 0);  
    lcd.print("Medida 2      ");  
    conteo();  
    Vis_Activado();  
    b=0;  
}
```

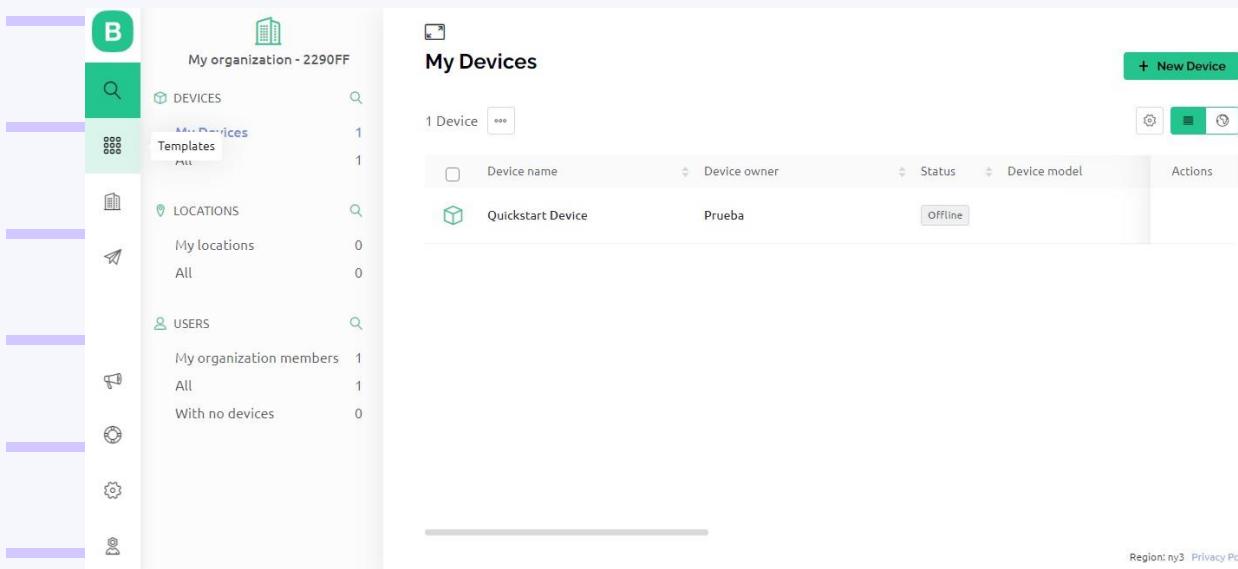
```

125 if(digitalRead(18) || c==1){
126     //pos_Final = 180;
127     pos=180;
128     tiempo_act=8000;
129     lcd.setCursor (0, 0);
130     lcd.print("Medida 3      ");
131     conteo();
132     Vis_Activado();
133     c=0;
134 }
135 }
136
137 void conteo(){
138     for (contador=5;contador>0;contador--)
139     lcd.setCursor (0, 1);
140     lcd.print("      ");

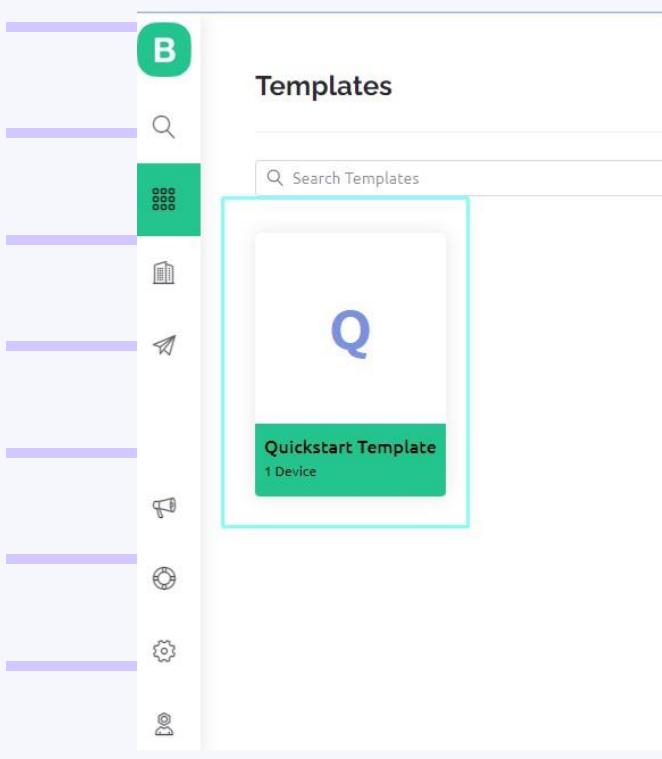
```

Se carga todos los cambios a la tarjeta, y nos dirigimos a la plataforma de Blynk.

- Estando en la plataforma de Blynk nos dirigimos a la opción de templates



Y seleccionamos el recuadro que se marcó, lo cual vamos a trabajar sobre el ejemplo que nos brinda Blynk el cual editaremos a nuestro beneficio para cumplir con el desarrollo de nuestra app



Luego de abrirlo, nos dirigimos a la parte superior de la derecha y le damos la opción de edit.

The screenshot shows the 'Edit' view of the 'Quickstart Template'. At the top right are 'Duplicate' and 'Edit' buttons. The main area has tabs for 'Info', 'Metadata', 'Datastreams', 'Events', 'Automations', 'Web Dashboard', and 'Mobile Dashboard'. Under 'Info', details like 'HARDWARE: ESP32', 'CONNECTION TYPE: WiFi', and 'MANUFACTURER: Blynk' are shown. A 'FIRMWARE CONFIGURATION' section contains code: '#define BLYNK\_TEMPLATE\_ID "TMPLGFNjmXO4"' and '#define BLYNK\_DEVICE\_NAME "Quickstart Template"'. A note below says 'Template ID and Device Name should be included at the top of your main firmware'. Other sections include 'OFFLINE IGNORE PERIOD' (0 hrs 0 mins 0 secs), 'TEMPLATE IDs' (TMPLGFNjmXO4), and 'DESCRIPTION' (First time experience product).

Y le damos un nombre a nuestro proyecto, en este caso Dispensador IoT

**Quickstart Template**

**Info** Metadata Datastreams Events Automations Web Dashboard Mobile Dashboard

TEMPLATE NAME: Dispensador IoT

HARDWARE: ESP32 CONNECTION TYPE: WiFi

DESCRIPTION: First time experience product

TEMPLATE ID: TMPLGFNjmX04 MANUFACTURER: Blynk

OFFLINE IGNORE PERIOD: 00 hrs 00 mins 00 secs

FIRMWARE CONFIGURATION:

```
#define BLYNK_TEMPLATE_ID "TMPLGFNjmX04"  
#define BLYNK_DEVICE_NAME "Dispensador IoT"
```

Template ID and Device Name should be included at the top of your main firmware

Luego nos dirigimos a la opción de Web Dashboard

**Quickstart Template**

**Web Dashboard** Info Metadata Datastreams Events Automations Mobile Dashboard

**Widget Box** 2 of 30 widgets

CONTROL

- Switch
- Slider
- Number Input
- Image Button

Device name: Online  
Device Owner: Company Name

Dashboard

Last Hour 6 Hours 1 Day 1 Week 1 Month 3 Months Custom

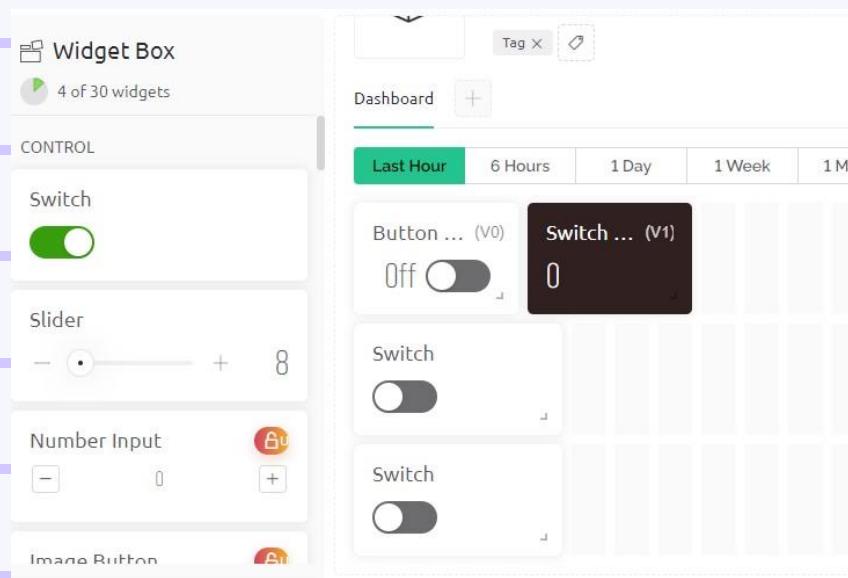
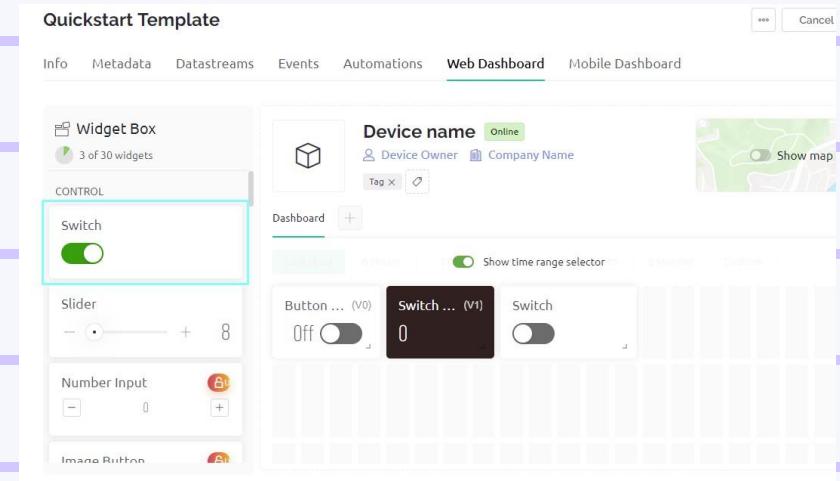
Button ... (V0) Switch ... (V1)  
Off 0

Show map UPGRADE

Region: ny3 Privacy Policy

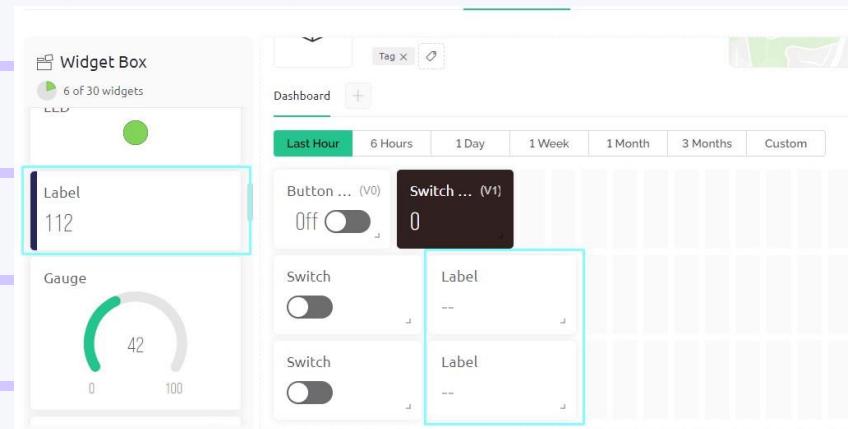
Estando ya ahí, añadiremos dos switches más ya que el ejemplo trae uno por defecto, de la siguiente forma se anexarán.

Buscamos la opción switch y lo arrastramos que quede de forma ordenada como se muestra en las siguientes imágenes. Estos serán nuestros botones en la app



Lo siguiente es añadir label, arrastrándolos al lado del switch, los labels serán el valor que tomara nuestro botón al ser oprimido. de la siguiente forma se anexan:

Se busca la opción de label y se arrastran de forma ordenada.



Ahora vamos a crear los Datastreams, los cuales captaran toda la información requerida.

The screenshot shows the 'Quickstart Template' interface with the 'Datastreams' tab selected. There are four datastreams listed:

ID	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Actions
1	Switch Control	Switch Control	Green	V0	Integer		false	0	
2	Switch Value	Switch Value	Green	V1	Integer		false	0	
3	Seconds	Seconds	Green	V2	Integer		false	0	
4	Button Image	Button Image	Green	V3	String		false		

Eliminamos las dos últimas opciones, ya que estos no son necesarios en nuestro desarrollo.

The screenshot shows the Datastreams interface with two datastreams listed:

ID	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Actions
1	Switch Control	Switch Control	Green	V0	Integer		false	0	
2	Switch Value	Switch Value	Green	V1	Integer		false	0	

Y creamos 4 nuevos datastreams

The screenshot shows the Datastreams interface with two datastreams listed. A modal window is open for creating a new datastream, showing options: Digital, Analog, Virtual Pin, Enumerable, and Location (with an 'UPGRADE' button).

ID	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Actions
1	Switch Control	Switch Control	Green	V0	Integer		false	0	
2	Switch Value	Switch Value	Green	V1	Integer		false	0	

### Virtual Pin Datastream

NAME: Med2  
PIN: V2  
DATA TYPE: Integer  
UNITS: None  
MIN: 0 MAX: 1 DEFAULT VALUE: 0  
ADVANCED SETTINGS  
Cancel Create

Crearemos otro con el nombre de Val2, el que será el valor de la media 2

### Virtual Pin Datastream

NAME: Val2  
PIN: V3  
DATA TYPE: Integer  
UNITS: None  
MIN: 0 MAX: 1 DEFAULT VALUE: 0  
ADVANCED SETTINGS  
Cancel Create

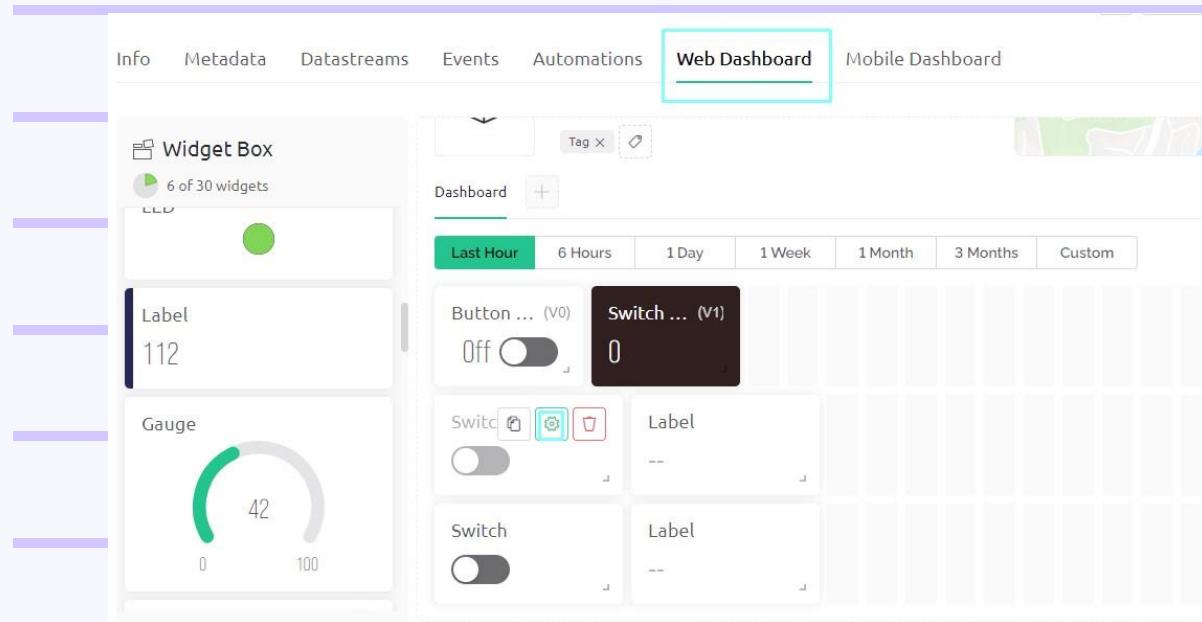
Y por último el med3 y su val3, para su valor. También le cambias el nombre a los dos primero a med1 y val1.

6 Datastreams							
	ID	Name	Alias	Color	Pin	Data Type	Units
1	Val1	Val1	V1	Green	V1	Integer	False
2	Med2	Med2	V2	Red	V2	Integer	False
3	Val2	Val2	V3	Yellow	V3	Integer	False
4	Med3	Med3	V4	Black	V4	Integer	False

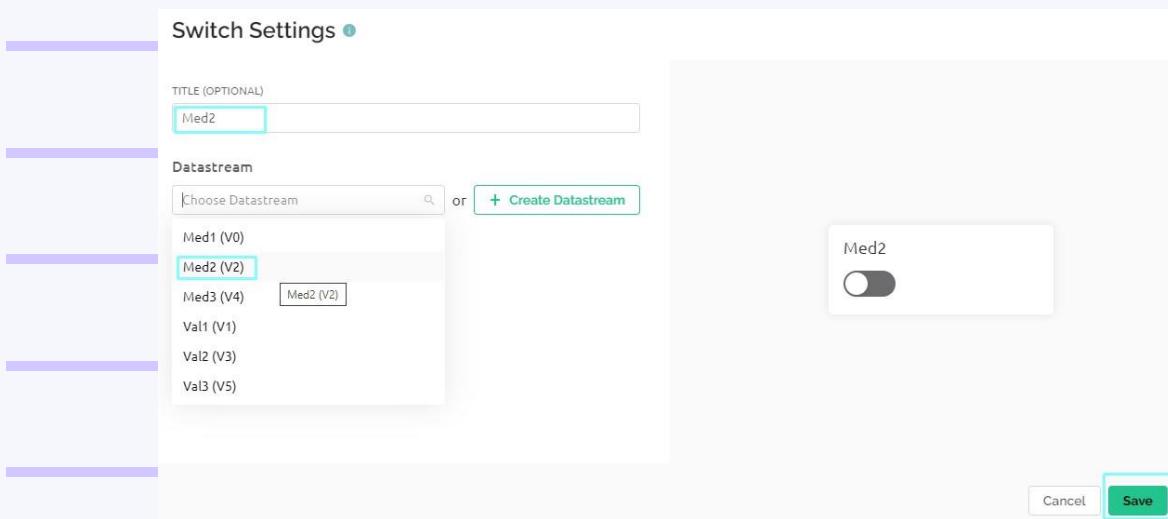
Region: ny3 Privacy Pol.

Luego de estar creado todo, ahí nos muestra los pines que les corresponde a cada uno de los Datastreams, y estos serán los que permitirán las conexiones entre Blynk y ESP32. Los cuales indicamos en nuestra programación.

Luego nos regresamos al Dashboard y seleccionamos un switch y le damos en configuración.



Se abrirá una ventana donde le das título a nuestro switch en este caso este será para la medida 2 de alimento entonces por ende será Med2 y su valor creado en el datastreams que también es med2, ya que así se captura la información como se muestra en la siguiente imagen.

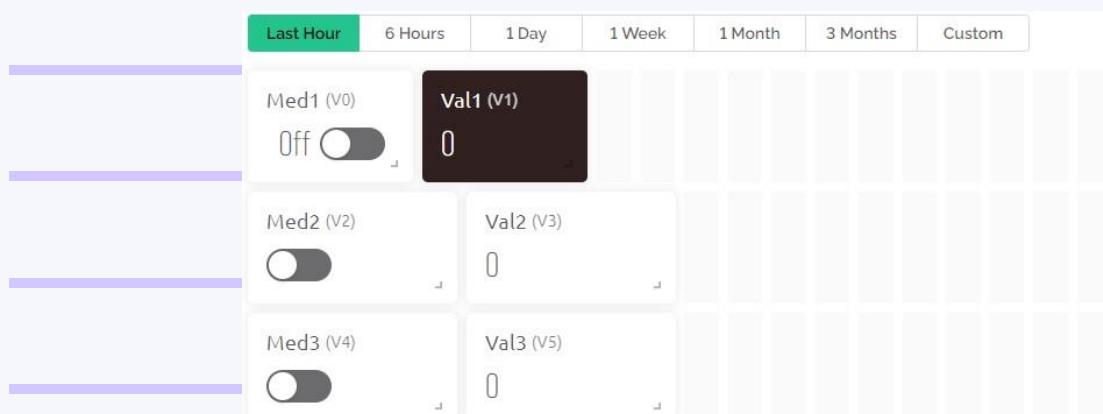


Ahora lo replicaremos en el label, donde también iremos a su configuración y le daremos un título, en este caso su título es val2, porque este será el valor de la medida 2 al momento de ejecutarse y su valor será val2 el cual creamos en nuestros datastreams, como lo mostramos en las siguientes imágenes.

Y luego repetimos el mismo proceso con cada uno del switch que serán nuestros botones en la app, y lo mismo con los labels que mostrarán el valor que tomara.

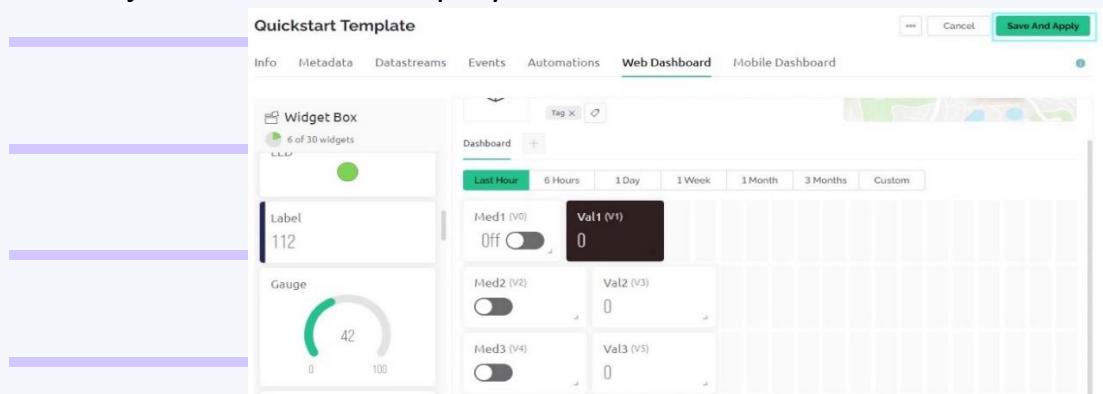
Para la medida 1, en el switch le daremos como título Med1 y le asignamos su valor donde se captarán los datos que es Med1, el mismo proceso con su label, que de título tendrá Val1, y su valor creado en el datastreams que es val1, donde se captara la información.

Luego repetimos el mismo procedimiento explicado con el ultimo que sería para la medida 3, el cual se le asigna también sus títulos y sus valores correspondientes. En la siguiente imagen se indicará el resultado final.

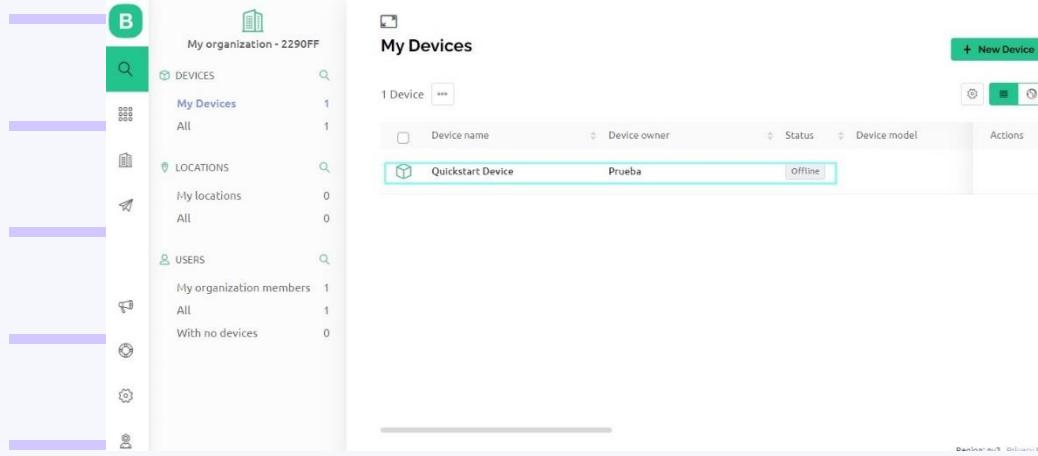


Se muestra como resultado final los botones con sus títulos y sus valores con los cuales se captarán la información y se pueda ejecutar y lo mismo en los labels se muestra con sus respectivos títulos y sus valores donde captaran información y mostraran al momento de ejecutar el programa.

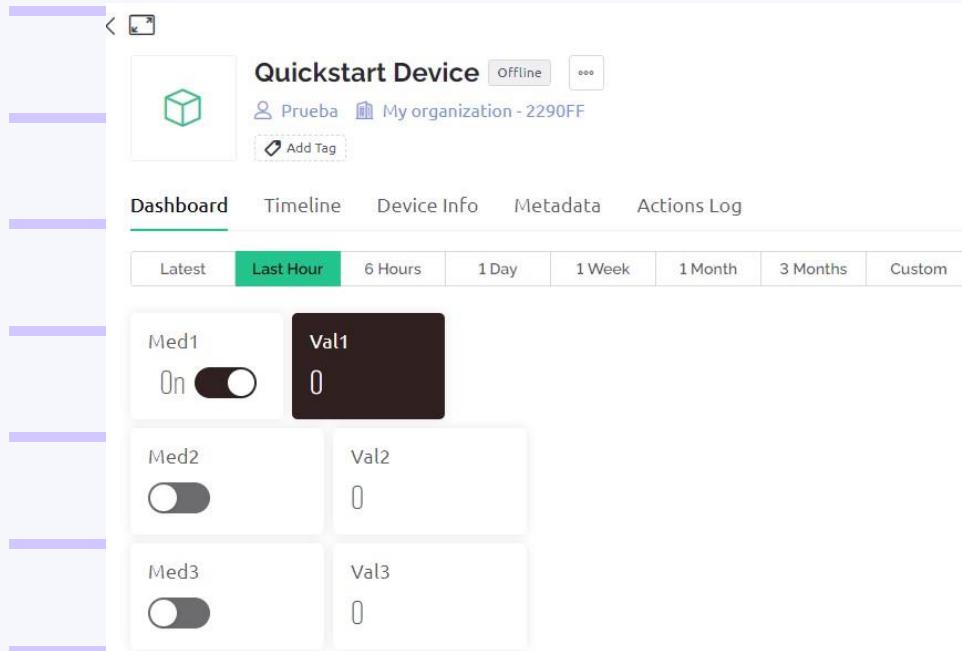
Ahora les indicaremos como salvar todo lo desarrollado para iniciar la ejecución de nuestro proyecto.



Regresamos al inicio y buscamos el proyecto y realizamos pruebas. Tenga en cuenta que debe tener conectada la tarjeta ESP32 para que funcione y todo sea cargado a la tarjeta.



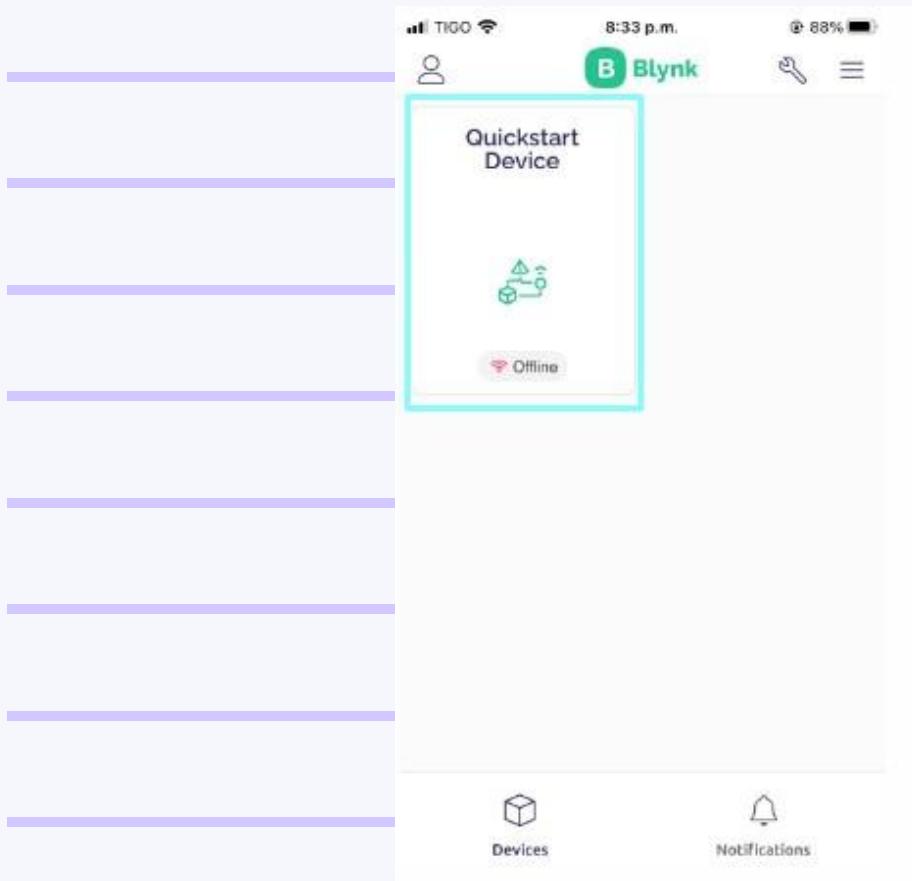
Oprimimos el botón y observamos algún cambio en nuestra pantalla LCD y dejamos que el val llegue al valor 1 para que nuestro servomotor empiece a girar y de nuevo lo oprimimos el botón para desactivarlo.



Y lo mismo realizamos con cada uno de los botones de las medidas, para realizar la prueba de que esté funcionando. Recuerda desactivarlo después de que el val tomó valor de 1 ya se pude apagar.

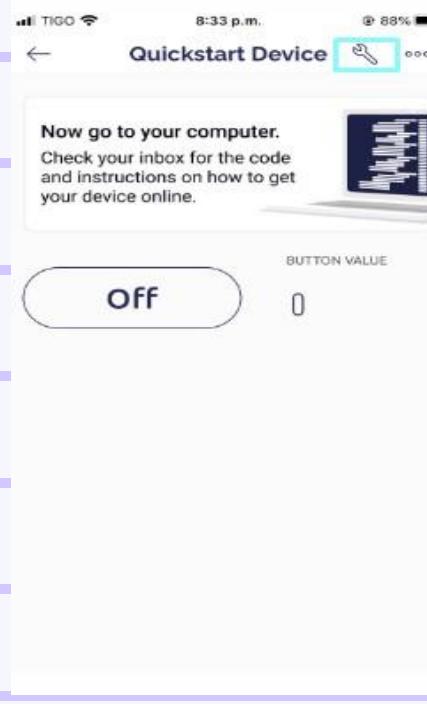
Ahora nos dirigimos a la aplicación en nuestro teléfono, recuerda que debes ingresar con el mismo correo con el que iniciaste sección por la web para que te salga el proyecto que creamos.

Estando ya dentro de nuestra app en nuestro teléfono, nos debe salir el proyecto ya creado, lo abrimos

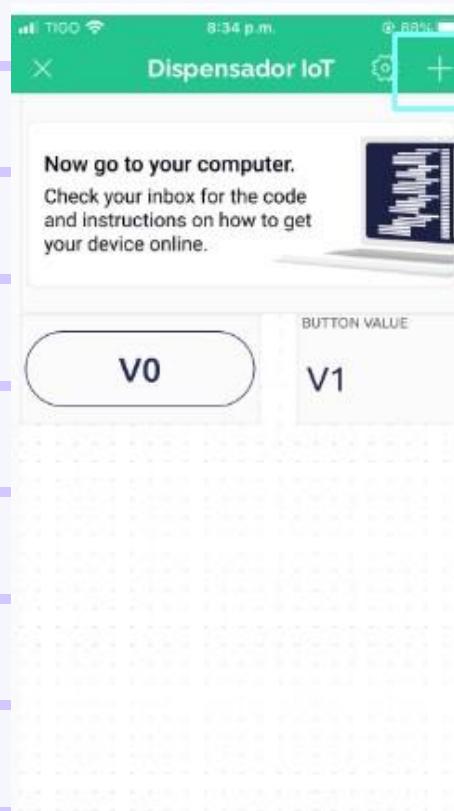


Y cuando lo abramos saldrá solo un botón con su respectivo label, es algo normal que no salga completo el proyecto desarrollado, pero eso no quiere decir que no exista, solo debemos arrastrar unos botones y labels como lo hicimos desde la web, solo que este caso ya no debemos crear datastreams porque ya existen, solo debemos asignárselos a nuestras opciones, ahora lo explicaremos.

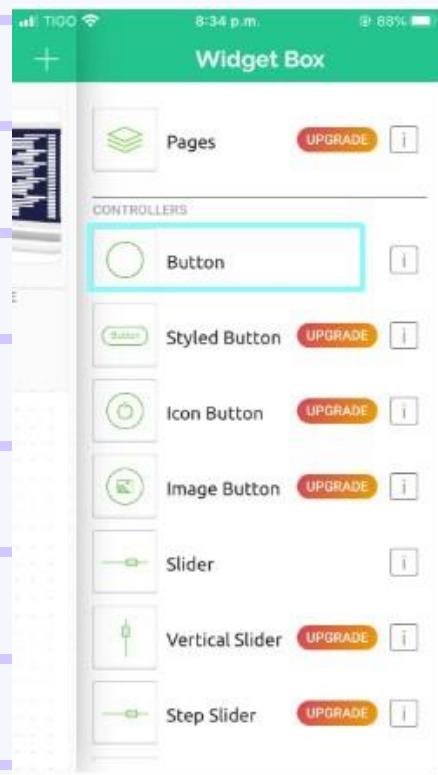
Ya estando dentro del programa, nos dirigimos a la opción de editar como lo mostraremos a continuación



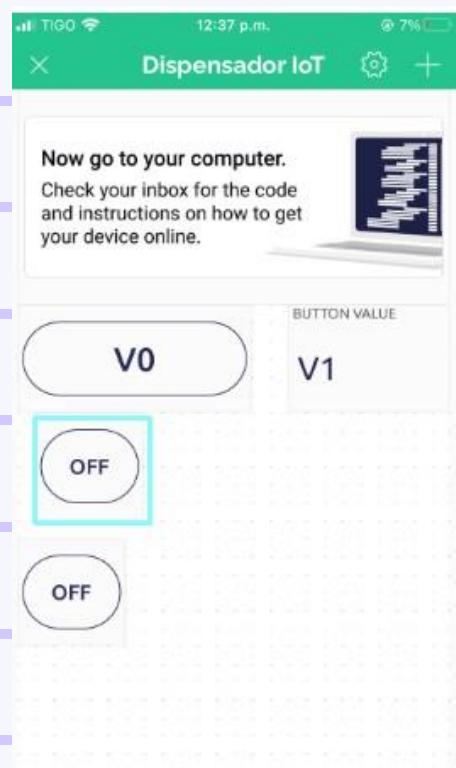
Ya realizando el paso anterior, estaremos ya dentro de nuestro Dashboard y empezaremos a incorporar los elementos necesarios, como se indicará en la siguiente imagen.



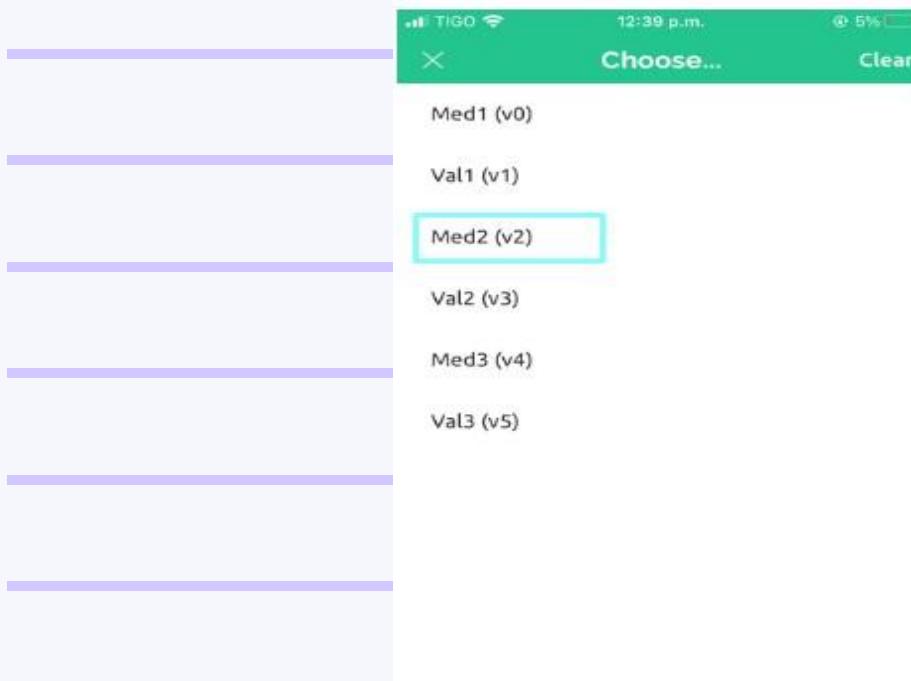
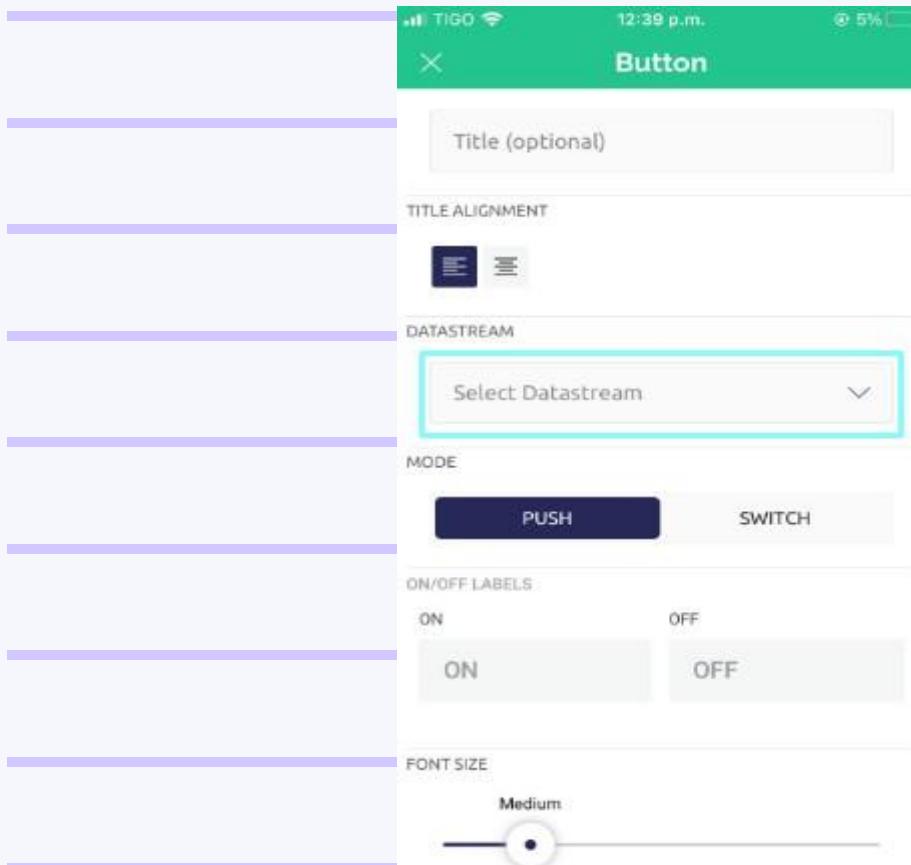
Seleccionando la opción marcada entramos a las herramientas donde buscaremos la opción button en los controllers.



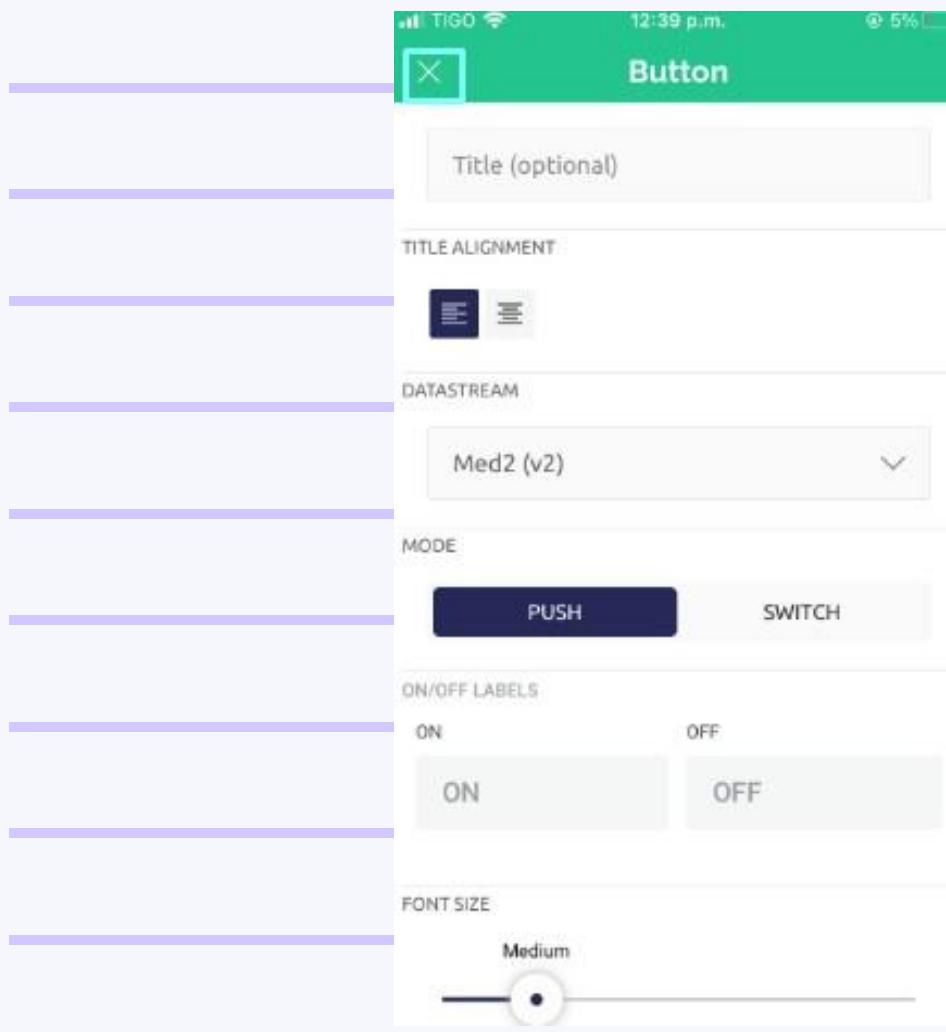
Lo agregamos dos veces, ya que por defecto ya tenemos solo uno creado



Luego lo seleccionamos para darle el valor correspondiente, en este caso el botón subrayado corresponde a la medida 2 así que le vamos a asignar el datastreams ya creado para captar los datos de la siguiente forma.

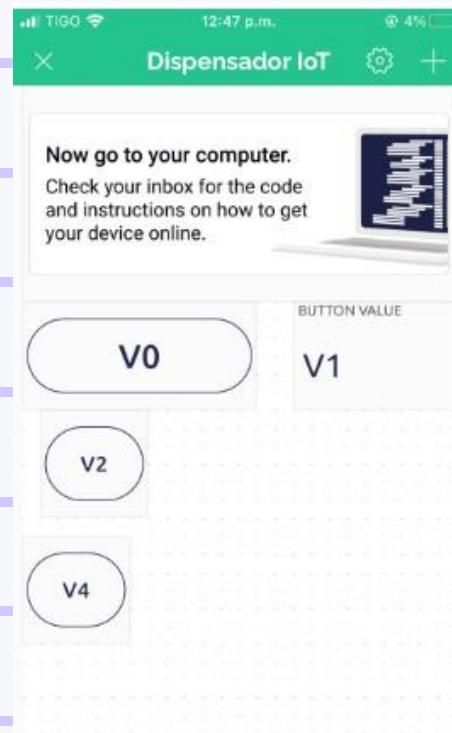


Seleccionamos el valor correspondido en este caso este será el botón para la medida 2, entonces corresponde a med2

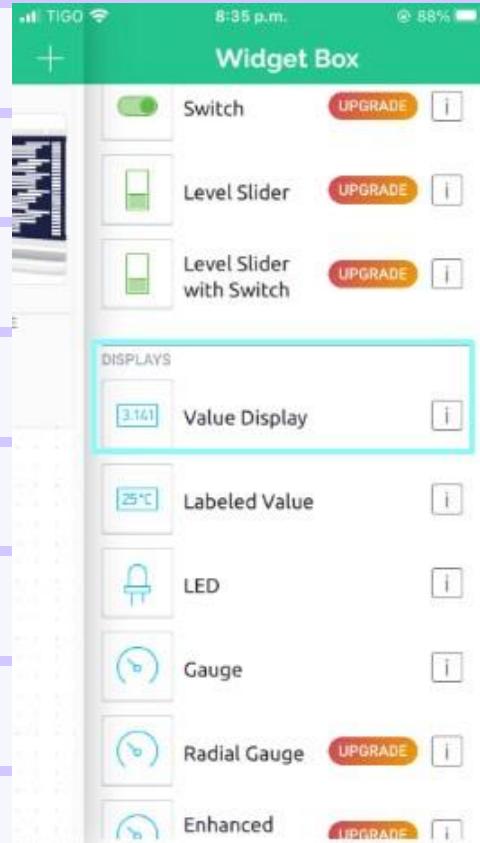


Y listo repetimos el mismo proceso con el ultimo botón que es para la medida 3, entonces su captador de datos es med3. Con el primer botón no es necesario ya que los tiene por defecto, pero puedes verificar en caso de que no, que tenga el valor de la medida 1.

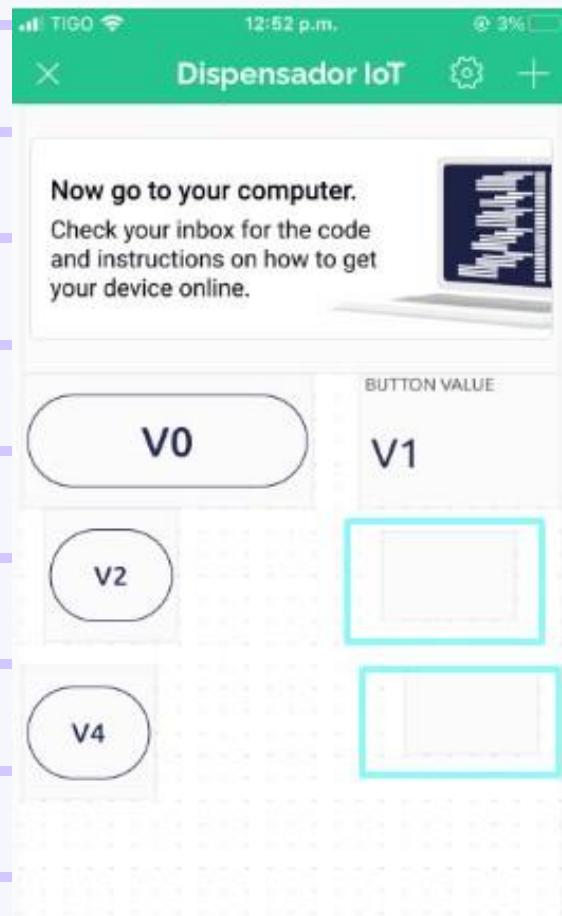
Para que queden de la siguiente forma:



Ahora vamos a añadir los labels que también creamos de forma web, nos dirigimos a las herramientas y buscamos en este caso la opción que diga Value Display.

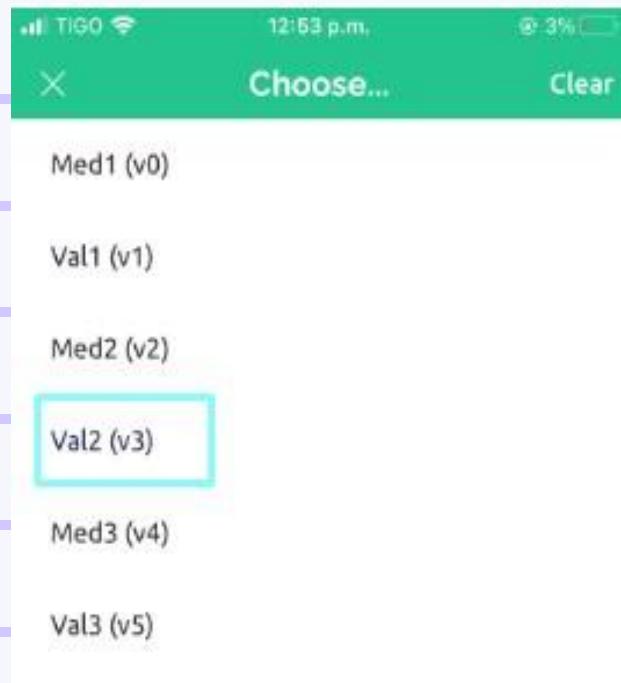
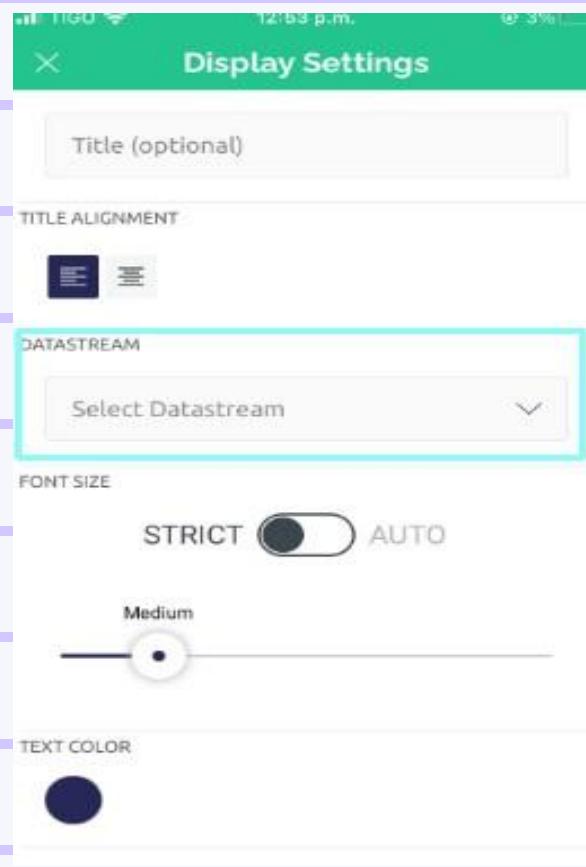


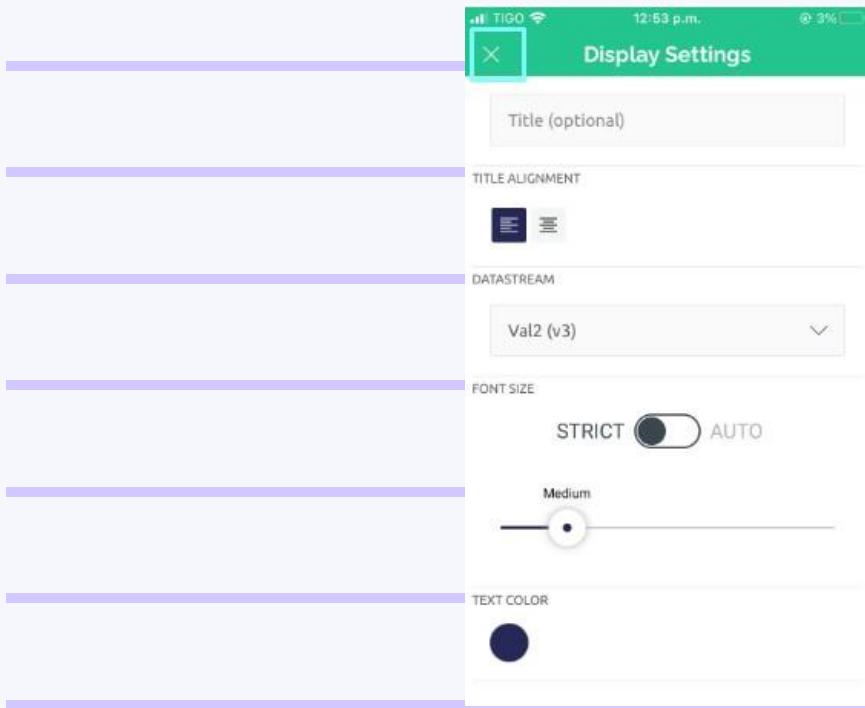
Añadimos dos Value Display para que quede de la siguiente forma.



Seleccionamos el segundo Value Display para asignarle su valor, en este caso este corresponde a la medida 2, así que por ende será el valor 2 de la medida 2.

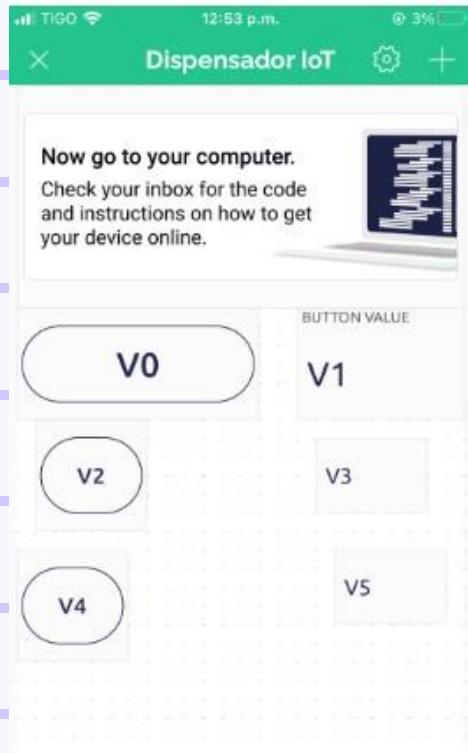
De la siguiente forma se realizará y los mismos pasos se realizan para el tercer Value Display que corresponde al tercer botón.



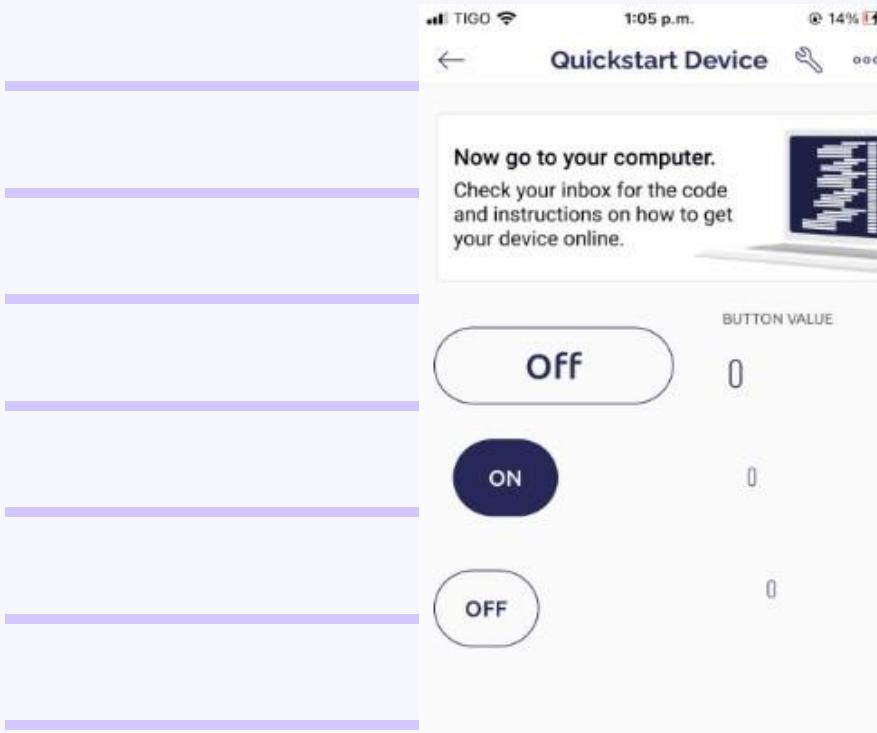


Repetimos el mismo paso con el ultimo Value Display para que todo quede de la siguiente forma. Recuerda que asignando sus valores correspondientes se captaran de forma ordenada sus datos para que se ejecute el programa de forma exitosa.

Por defecto el primer Value ya está configurado, pero puede revisarlo.

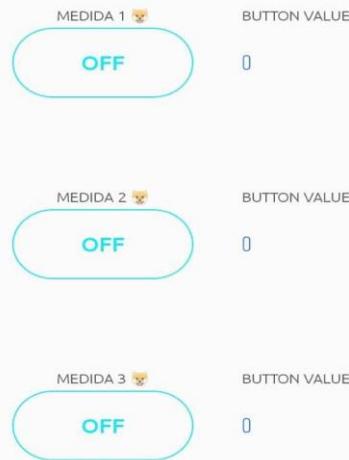


Ya teniendo todo como se muestra en la imagen anterior, nos regresamos al inicio de nuestro proyecto, para realizar las pruebas.

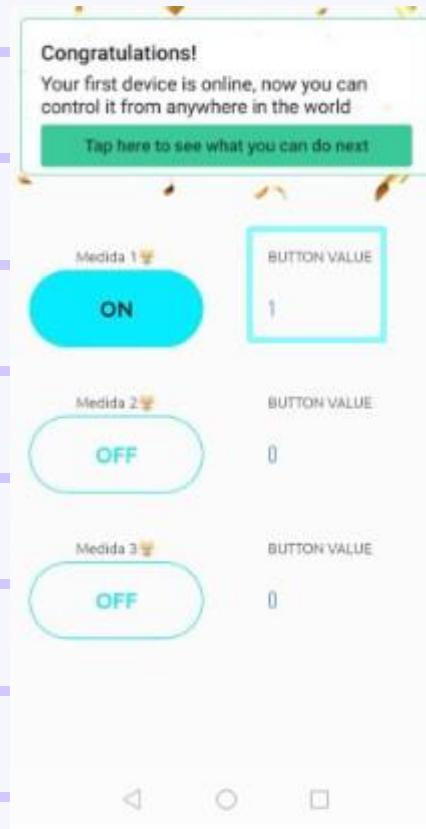


Se realiza pruebas con cada uno de los botones, debes tener en cuenta que en todo momento se debe tener conectada la tarjeta ESP32 para que se ejecute de manera exitosa la acción.

(Le puedes dar un diseño a tu gusto.)



También debes tener en cuenta que al momento de oprimir el botón en la App el label que añadimos debe tomar valor de 1, si es así ya está funcionando de manera exitosa nuestra App.



Recuerda oprimirlo para activarlo y oprimirlo para desactivarlo al momento. (video de ejemplo, <https://youtube.com/shorts/yqqhsivxII?feature=share>)



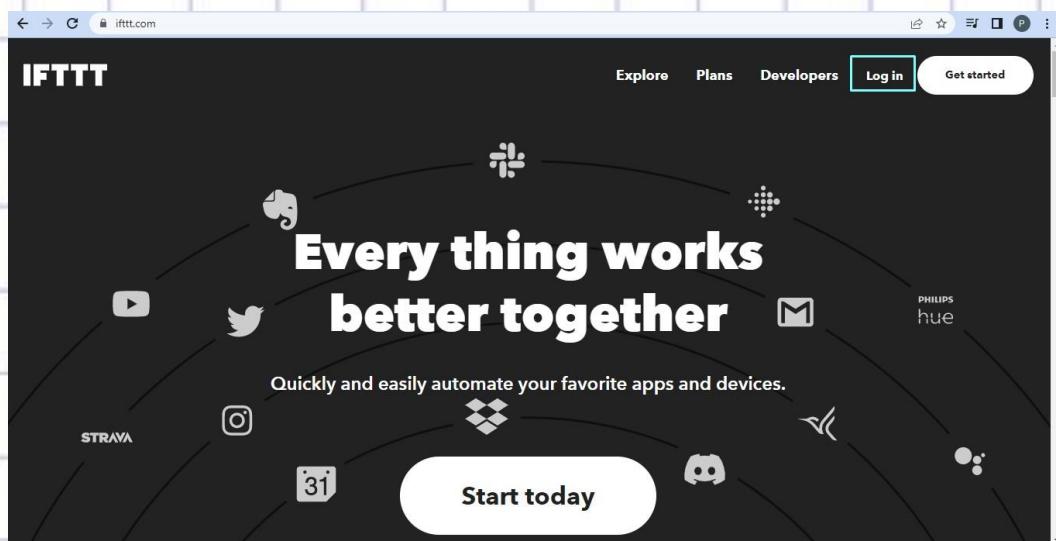
## PASO A PASO DE DESARROLLO DE GOOGLE ASISTENTE EN IoT

En este ítem daremos a conocer como es el desarrollo y sincronización de Google asistente en nuestro proyecto, para que este también tenga control por voz. Se realizará paso por paso para que el usuario tenga éxito en el desarrollo del mismo.

Tres cosas a tener en cuenta, es que utilizaremos las siguientes herramientas. (todas tienen logueo con el mismo correo usado anteriormente, si esto no se cumple, lo mismo no funcionara de forma satisfactoria)

- Adafruit
- IFTTT
- Google Home

### 1. Empezamos iniciando sesión en IFTTT





IFTTT

What is IFTTT?

Explore Plans Developers Log in

Get started

## Log in

Email

Password

Forgot your password?

Log in

Continue with Apple, Google, or Facebook

Seleccionan el tipo de cuenta que están usando

IFTTT

What is IFTTT?

Explore Plans Developers Log in

Get started

## Get started with IFTTT



Continue with Apple



Continue with Google



Continue with Facebook

Or use your email to [sign up](#) or [log in](#)

Seleccionan en sistema operativo están trabajando

## Let's start!

Where do you usually set up your automations?  
This is important so we can find the best Applets for you.



Android



iOS



Web

Y se loguean, para ya estar dentro de IFTTT

The screenshot shows the IFTTT Explore page. At the top, there are navigation links for "My Applets", "Explore", "Developers", "Upgrade", "Create", and a user profile icon. Below the header is a search bar with the placeholder "Search Applets or services". A prominent banner in the center says "New on IFTTT for September" over a background of colorful circuit board graphics.

## 2. Luego iniciaremos sesión en Adafruit

The screenshot shows the Adafruit homepage. The main visual features a green pot with a plant, an Arduino Uno microcontroller, and a battery pack on a wooden surface. To the right, a smartphone displays a data visualization with two circular gauges showing values of 7.50 and 5.36. The text "The internet of things for everyone" is overlaid on the image. On the right side of the screen, the text "scientists engineers students teachers makers" is displayed. The top navigation bar includes links for "Shop", "Learn", "Blog", "Forums", "LIVE!", "AdaBox", and "IO". The top right corner has "Get Started for Free", "Sign In", and a shopping cart icon with "0".

### SIGN IN

Your Adafruit account grants you access to all of Adafruit, including the shop, learning system, and forums.

EMAIL OR USERNAME

PASSWORD

[Forgot your password?](#)

[SIGN IN](#)

NEED AN ADAFRUIT ACCOUNT?

[SIGN UP](#)

### ORDER STATUS

Did you check out as a guest? Or do you just want to check your order status without signing in?

EMAIL ADDRESS

ORDER NUMBER

[Where do I find this?](#)

[CHECK ORDER STATUS](#)



Se llenan los siguientes campos con sus datos

SIGN UP

FIRST NAME  
Prueba

LAST NAME  
Proyecto

EMAIL  
pruebaproyectouts2022@gmail.com ✓

USERNAME  
pruebaproyecto2022 ✓

Username is viewable to the public on the forums, Adafruit IO, and elsewhere.

PASSWORD  
..... ✓

CREATE ACCOUNT

Y crea la cuenta, para ya estar logueados con Adafruit

Shop Learn Blog Forums LIVE! AdaBox **IO**

Hi, Prueba Proyecto | Account 0

**adafruit** Devices Feeds Dashboards Actions Power-Ups

ESP8266 based devices may be impacted due to SSL certificate updates, please see [this post](#) on our forums to ensure you can continue to connect to IO.

pruebaproyecto2022 / Devices

New to WipperSnapper?  
[Follow this guide to get connected!](#)

Get Help Quick Guides Learn IO Plus

3. Tenga en cuenta que debe estar en la opción IO



Shop Learn Blog Forums LIVE! AdaBox **IO**

**adafruit** Devices Feeds Dashboards Actions Power-Ups

ESP8266 based devices may be impacted due to SSL certificate updates, please see [this post on our forums](#) to ensure

pruebaproyecto2022 / Devices

+ New Device

New to WipperSnapper?  
Follow this guide to get connected!

Y luego nos dirigimos a Feeds

Shop Learn Blog Forums LIVE! AdaBox **IO**

Hi, Prueba Proyecto | Account **0**

**adafruit** Devices **Feeds** Dashboards Actions Power-Ups

ESP8266 based devices may be impacted due to SSL certificate updates, please see [this post on our forums](#) to ensure you can continue to connect to IO. **X**

pruebaproyecto2022 / Feeds

? Help

+ New Feed + New Group

Default

Feed Name	Key	Last value	Recorded
Welcome Feed	welcome-feed		9 minutes ago

Luego accedemos a New Feed

Shop Learn Blog Forums LIVE! AdaBox **IO**

Hi, Prueba Proyecto | Account **0**

**adafruit** Devices Feeds Dashboards Actions Power-Ups

ESP8266 based devices may be impacted due to SSL certificate updates, please see [this post on our forums](#) to ensure you can continue to connect to IO. **X**

pruebaproyecto2022 / Feeds

? Help

+ New Feed + New Group

Default

Feed Name	Key	Last value	Recorded
Welcome Feed	welcome-feed		18 minutes ago

Loaded in 0.28 seconds.





Y creamos un Feed, el cual será para medida 1 del alimento, en este caso su nombre será Med\_1 y le damos créate.

## Create a new Feed

X

### Name

Med\_1

Maximum length: 128 characters. Used: 5

### Description

Cancel

Create

Repetimos el mismo proceso para las demás medidas faltantes, para que quede así.

The screenshot shows the Adafruit IO Feeds page. At the top, there is a navigation bar with links for Devices, Feeds, Dashboards, Actions, and Power-Ups. There are also buttons for New Device and Help. Below the navigation bar, a message states: "ESP8266 based devices may be impacted due to SSL certificate updates, please see [this post](#) on our forums to ensure you can continue to connect to IO." A search bar is located at the top right. The main content area displays a table titled "Default" with four rows of feed data:

Feed Name	Key	Last value	Recorded	Action
Med_1	med-1		less than a minute ago	Lock
Med_2	med-2		less than a minute ago	Lock
Med_3	med-3		less than a minute ago	Lock

At the bottom left, it says "Loaded in 0.36 seconds."

Luego nos dirigimos a la opción de Dashboard





Screenshot of the Adafruit Dashboards interface. The top navigation bar includes 'Devices', 'Feeds', 'Dashboards' (which is selected), 'Actions', and 'Power-Ups'. A 'New Device' button is also present. A message at the top states: 'ESP8266 based devices may be impacted due to SSL certificate updates, please see [this post on our forums](#) to ensure you can continue to connect to IO.' A search bar and a 'Help' link are on the right.

The main content area shows a list of dashboards:

Name	Key	Created At
Welcome Dashboard	welcome-dashboard	September 22, 2022

Below the table, it says 'Loaded in 2.13 seconds.'

Luego seleccionamos la opción de New Dashboard

Screenshot of the Adafruit Dashboards interface, identical to the previous one but with the 'New Dashboard' button highlighted in blue.

The main content area shows a list of dashboards:

Name	Key	Created At
Welcome Dashboard	welcome-dashboard	September 22, 2022

Below the table, it says 'Loaded in 2.13 seconds.'

Y creamos un Dashboard llamado controles, ya que ahí dentro crearemos las opciones de las medidas.

Screenshot of a 'Create a new Dashboard' dialog box. The title is 'Create a new Dashboard' with a close button 'X'. It has two fields: 'Name' (containing 'Controles') and 'Description' (empty). At the bottom are 'Cancel' and 'Create' buttons, with 'Create' being highlighted with a blue outline.



## Ingresamos a controles

The screenshot shows the Adafruit IO web interface. At the top, there's a navigation bar with links for Devices, Feeds, Dashboards, Actions, and Power-Ups. To the right of the navigation is a 'New Device' button with a key icon. Below the navigation, a message states: "ESP8266 based devices may be impacted due to SSL certificate updates, please see [this post](#) on our forums to ensure you can continue to connect to IO." A blue 'X' icon is next to the message. The main content area is titled "pruebaproyecto2022 / Dashboards". It features a "New Dashboard" button with a plus sign and a magnifying glass search bar. On the left, there's a sidebar titled "Dashboards" with a table:

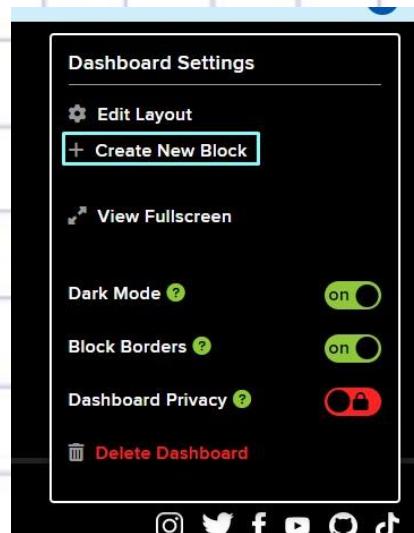
Name	Key	Created At
Controles	controles	September 22, 2022
Welcome Dashboard	welcome-dashboard	September 22, 2022

A small note at the bottom says "Loaded in 0.43 seconds."

Y dentro de controles nos saldrá un tablero, nos dirigimos a la parte de la derecha y seleccionamos la opción de configuración.

This screenshot shows the "Controles" dashboard within the Adafruit IO interface. The top navigation bar and message are identical to the previous screenshot. The main content area is titled "pruebaproyecto2022 / Dashboards / Controles". In the top right corner of this section, there is a gear icon with a dropdown arrow, which is highlighted with a red box. The rest of the dashboard is currently blacked out.

Luego se abrirá una pantallita, donde seleccionaremos la opción de Create New Block



Y seleccionamos la primera opción que nos aparece, ya que estos serán los labels que tendrán los valores de las medidas.

## Create a new block

X

Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.

A digital switch block with a green 'ON' button on the left and a white 'OFF' button on the right, set against a black background.

A large blue rectangular button with the word 'RESET' in white capital letters in the center.

A horizontal slider with a blue track and a circular blue knob in the middle. The word 'Slider' is written above it.

A circular gauge with a blue arc indicating progress, set against a black background.

A text block containing the text 'HELLO WORLD!' in white capital letters.

2023-09-11 10:22:45	= 42
2023-09-11 10:55:27	= 52
2023-09-11 10:55:39	= 38
2023-09-11 10:55:39	= 50
2023-09-11 10:55:39	= 28
2023-09-11 10:55:39	= 65
2023-09-11 10:55:39	= 37
2023-09-11 10:55:39	= 78
2023-09-11 10:55:40	= 44
2023-09-11 10:55:40	= 55
2023-09-11 10:55:40	= 39
2023-09-11 10:55:40	= 60
2023-09-11 10:55:47	= 55
2023-09-11 10:55:49	= 65
2023-09-11 10:55:58	= 61
2023-09-11 10:55:59	= 27

A gray camera icon with a white lens and a flash.

A line chart with a dark blue line showing a fluctuating trend over time.

A solid blue circle.

A map showing a street grid with labels like 'West Street', 'Main Street', and 'Highway 101'.

A circular control panel with several red and blue buttons and labels such as 'STOP', 'GO', 'UP', 'DOWN', 'LEFT', 'RIGHT', and 'SETUP'.

A white cloud icon on a black background.

Se abrirá otra pantalla, donde deberemos seleccionar a que feed ira asociado este label, ya que este será para la medida1, debe ir asociado al med\_1.

Connect a Feed			
A toggle button is useful if you have an ON or OFF type of state. You can configure what values are sent on press and release.			
Choose a single feed you would like to connect to this toggle. You can also create a new feed within a group.			
<div style="border: 1px solid #ccc; padding: 5px; display: flex; align-items: center;"><input type="text" value="Search for a feed"/> <span style="font-size: 2em;">🔍</span></div>			
Default	Feed Name	Last value	Recorded
<input checked="" type="checkbox"/>	Med_1	13 minutes	
<input type="checkbox"/>	Med_2	13 minutes	
<input type="checkbox"/>	Med_3	13 minutes	
<input type="checkbox"/>	Welcome Feed	32 minutes	
<input type="text" value="Enter new feed name"/>			



Luego se deben llenar los siguientes campos

### Block settings

X

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title (optional)

Med\_1

Block Preview

Med\_1

Button On Text:

ON

Limit of 6 characters for the toggle text. Use the block title to be more descriptive.

Button On Value (uses On Text if blank)

1

Toggle A toggle button is useful if you have an ON or OFF type of state. You can configure what values are sent on press and release.

Button Off Text:

OFF

Limit of 6 characters for the toggle text. Use the block title to be more descriptive.

Button Off Value (uses Off Text if blank)

0

Test Value

45

Published Value

0 bytes

< Previous step

Create block

Lo primero es darle un nombre al label el cual será igual que su feed, Med\_1.

Lo siguiente es asignar el valor de 1, ya que cuando este, esté encendido tome como valor 1.

Y luego asignamos el valor 0, para este mismo cuando se apague tome valor de 0. Estos datos son importantes para el momento de la programación en Arduino.



Se repite el mismo proceso para cada una de las medidas, para tener el siguiente resultado.

The screenshot shows the Adafruit IO dashboard for the project 'pruebaproyecto2022'. At the top, there are navigation links: Devices, Feeds, Dashboards, Actions, and Power-Ups. On the right, there are buttons for 'New Device' and a gear icon. A message at the top states: 'ESP8266 based devices may be impacted due to SSL certificate updates, please see [this post on our forums](#) to ensure you can continue to connect to IO.' Below the message, the dashboard title is 'pruebaproyecto2022 / Dashboards / Controles'. There are three control cards, each containing a toggle switch labeled 'OFF': 'Med\_1', 'Med\_2', and 'Med\_3'. To the right of the dashboard is a large white space.

Un paso más es buscar la llave que nos ofrece el código que se debe agregar en Arduino. (explicado en la parte del desarrollo del código). Otro paso es ingresar a la opción de devices

The screenshot shows the Adafruit IO devices page for the project 'pruebaproyecto2022'. At the top, there are navigation links: Shop, Learn, Blog, Forums, LIVE!, AdaBox, and IO. On the right, there are buttons for 'New Device' and a gear icon. A message at the top states: 'ESP8266 based devices may be impacted due to SSL certificate updates, please see [this post on our forums](#) to ensure you can continue to connect to IO.' Below the message, the page title is 'pruebaproyecto2022 / Devices'. There is one device listed: '+ New Device'. A note below the device says: 'New to WipperSnapper? Follow this guide to get connected!' To the right of the devices page is a large white space.

Y seleccionamos la opción de tarjeta que se está trabajando y listo.





pruebaproyecto2022 / Devices / New Device

? Help

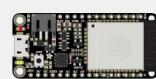
Set up new device

1 Choose your board

Choose your board



17 compatible boards



Feather HUZZAH  
ESP32  
by Adafruit

Choose Board



ESP32-S2 Feather  
by Adafruit

Choose Board



ESP32-S2 TFT  
Feather  
by Adafruit

Choose Board

#### 4. Descargamos en nuestro dispositivo la app de Google Home.

Mientras se descarga la app, nos dirigimos a la plataforma de IFTTT y damos la opción de Create.

IFTTT

Welcome to IFTTT

My Applets

Explore

Developers

Upgrade

Create



## Explore

All

Applets

Services

Stories

Search Applets or services



Seleccionamos la primera opción





## Create

Upgrade for more, faster, better Applets with advanced features. [Upgrade](#)

You're using 0 of 5 Applets

If This

Add

Then That

Luego buscamos Google Assistant V2

## Choose a service



Google Assistant V2

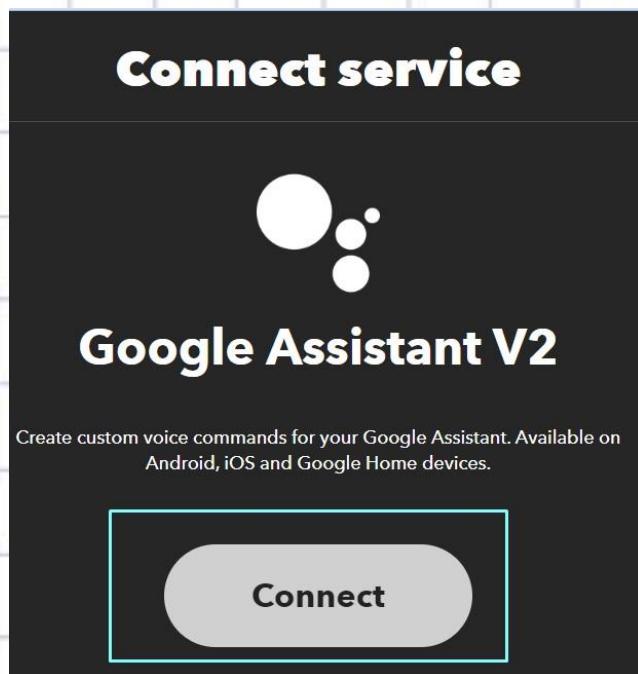


Google Assistant V2



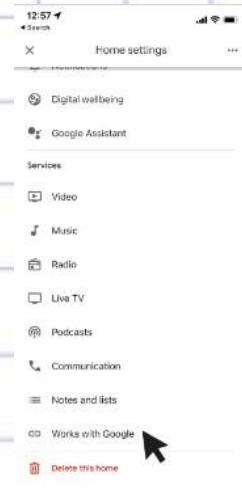


Luego deberemos conectar el servicio de Google Assistant V2 a IFTTT



Al oprimir Connect nos saldrá un recuadro dando las indicaciones que deberemos tomar para su sincronización.

- Se debe instalar Google Home en nuestro teléfono
- Iniciar sección con el mismo correo que se ingresó a IFTTT y Adafruit
- Ya luego dentro de la app, creamos nuestra casa la podemos llamar MyHome, nos vamos a la configuración y buscamos la opción que dice

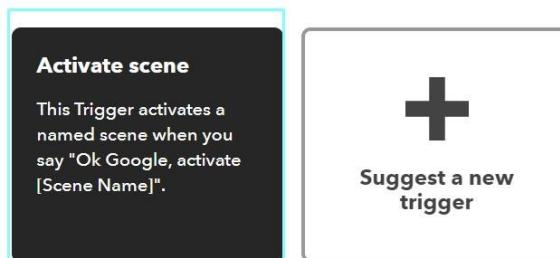
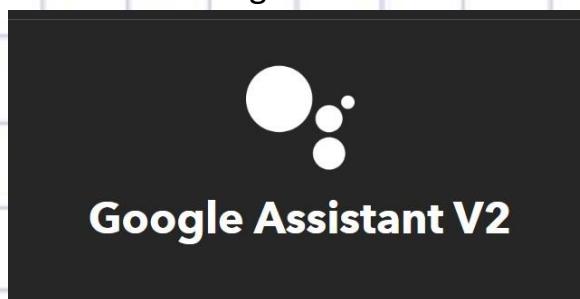




- Buscamos IFTTT y accedemos y damos todos los permisos para poder realizar la conexión
- Nos regresamos a la página de IFTTT y refreshcamos y ya debe estar conectada a Google

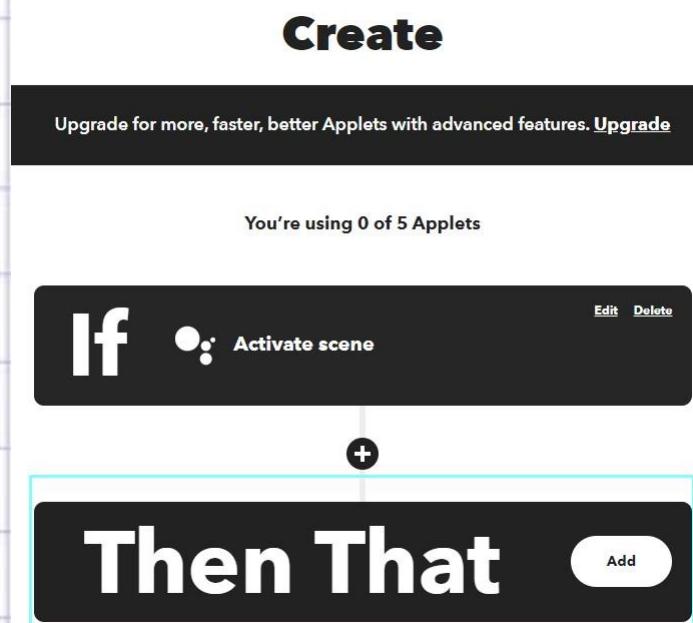
5. Luego de realizar los pasos anteriores ya podremos crear nuestra escena, la cual le deberemos dar un nombre.

- Para la medida1 será pequeño
- Para la medida2 será mediano
- Para la medida3 será grande

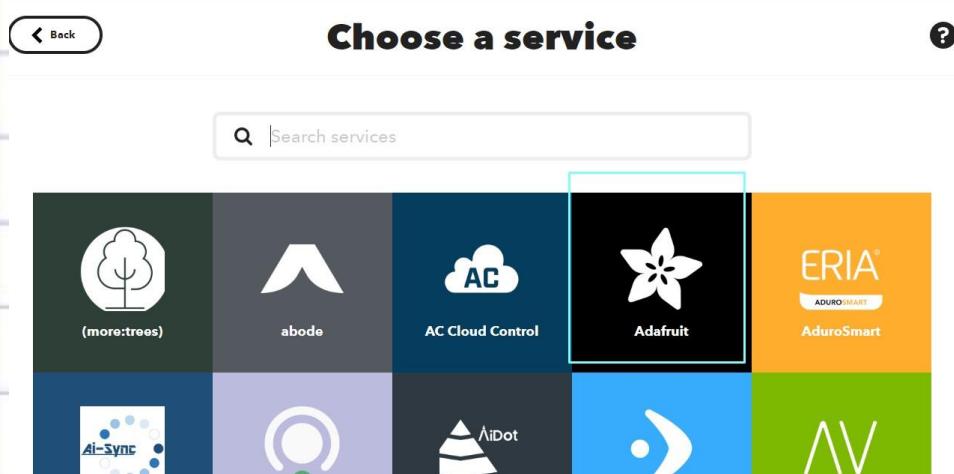




Le damos crear y luego nos vamos a la segunda opción, como se muestra en la siguiente imagen



Ingresamos y ya estando dentro buscamos la opción de Adafruit



Abrimos la opción marcada





## Adafruit

### Send data to Adafruit IO

This Action will send data to a feed in your Adafruit IO account.



Suggest a new action

Y conectamos Adafruit con IFTTT

## Connect service



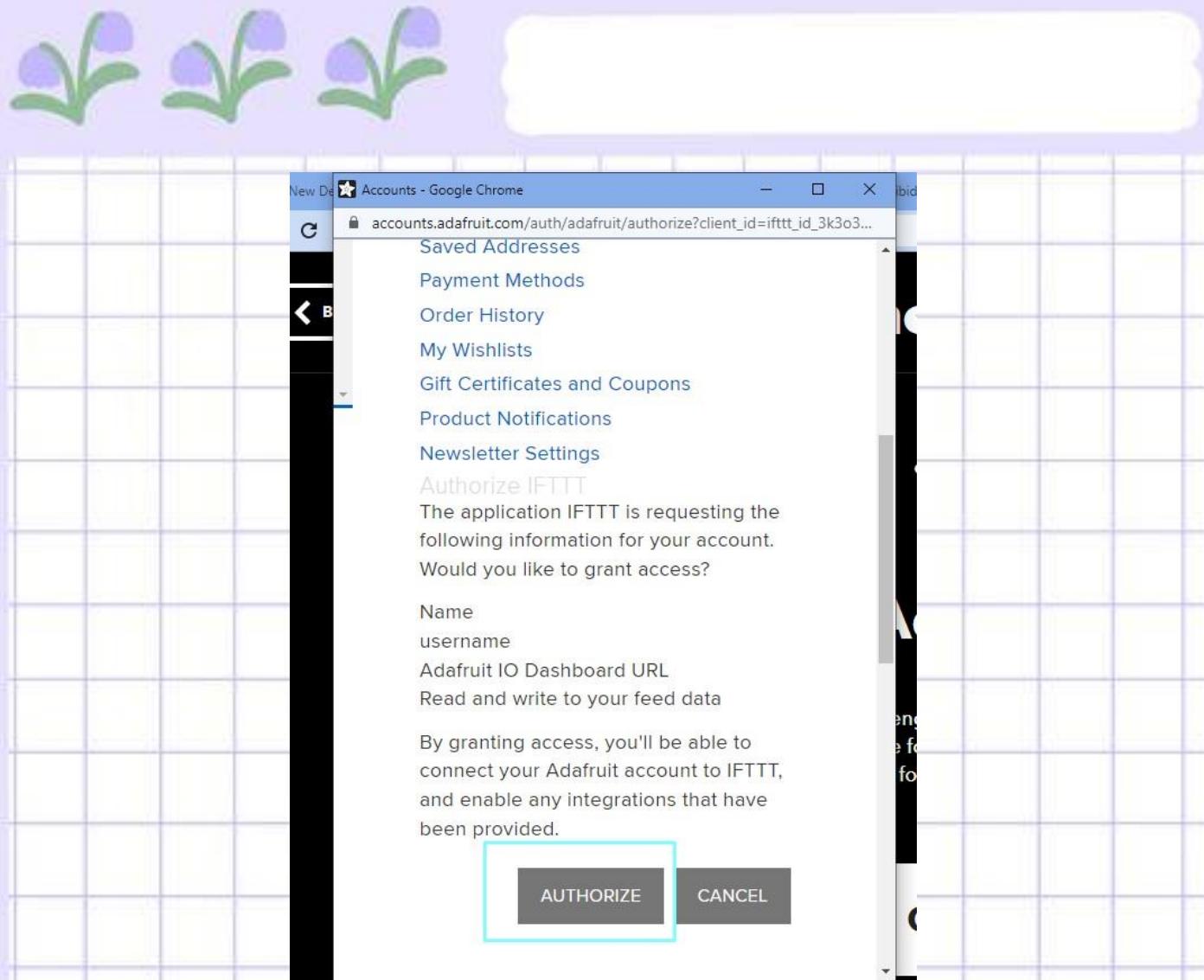
## Adafruit

Adafruit was founded by MIT engineer, Limor "Ladyada" Fried. Her goal was to create the best place online for learning electronics and making the best designed products for makers of all ages and skill levels.

Connect

Damos autorización

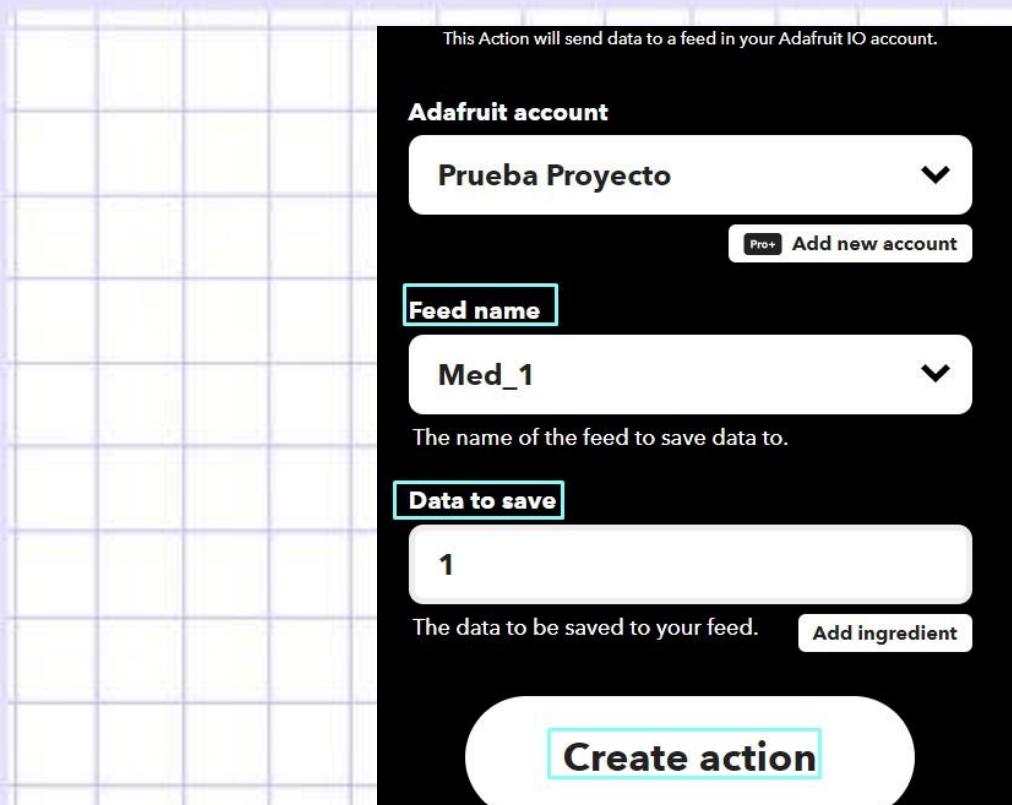




Ahora llenaremos los siguientes campos, en donde en feed name buscaremos el valor que le corresponderá a la escena “pequeño” que será med\_1.

Luego ingresaremos el valor 1 ya que ese es valor que se guardara y que tomara al momento de ejecutarse en la programación de Arduino

Y luego creamos la acción.



Damos en la opción de continuar



Y luego finalizamos toda la escena completa.



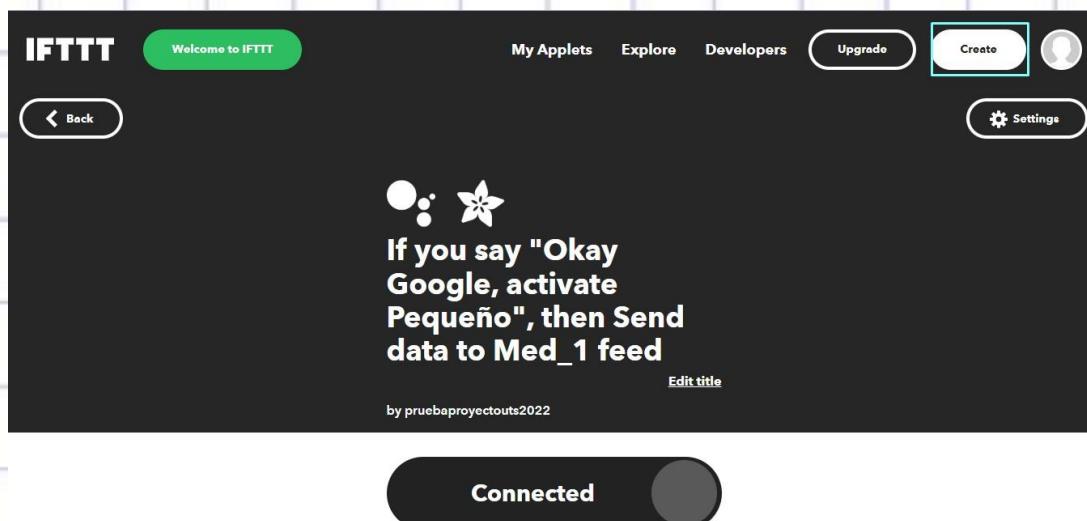


Receive notifications when this Applet runs



Finish

Repetimos el mismo proceso para las siguientes medidas, recuerda que med\_2 se llamará mediano, y med\_3 grande y que los datos a guardar para todas es 1, ya que ese será el valor que tomará al momento de ejecutarse con la tarjeta ESP32.



6. Luego nos dirigimos a Google home desde el teléfono y buscamos a la asistente de voz, a la cual se le indicará que ejecute la acción.



- Debes indicarle a la asistente de voz que “activar pequeño” o “activar mediano” o “activar grande” para que ella realice la acción
- Te diriges a la página de Adafruit para que veas si los botones se activan cuando se le indica a la asistente de opción debe activar.

The screenshot shows a dashboard titled "pruebaproyecto2022 / Dashboards / Controles". It contains three control elements labeled "Med\_1", "Med\_2", and "Med\_3", each with a toggle switch. "Med\_1" is set to "ON" (green), "Med\_2" is set to "OFF" (red), and "Med\_3" is set to "OFF" (red). A note at the top of the dashboard states: "ESP8266 based devices may be impacted due to SSL certificate updates, please see [this post on our forum](#)". The status was last updated on "September 22nd 2022, 3:49:55PM".

7. Los siguientes pasos a realizar son la programación que se debe suministrar al código existente en Arduino para que lo anterior se sincronice con todo el proyecto junto.

Este paso se encuentra en la primera parte de nuestro manual, donde se explica el desarrollo del código que se implementó en la plataforma de Arduino, para el desarrollo de la voz y el control de esta misma en el dispensador.





## Resultado final del dispensador.



Como base se utilizaron cajas de cartón para elevar el dispensador, y poder tener una altura adecuada para cuando este arroje el alimento. También se utilizó una botella de plástico donde se almacenará el alimento de las mascotas.

CODIGO: [https://github.com/daironporras/Proyecto\\_Grado\\_Dis\\_2](https://github.com/daironporras/Proyecto_Grado_Dis_2)

Se adjunta código con el que se realizó la programación para el funcionamiento del dispensador.





