

---

# Tabla de contenido

0 Inicio	1.1
----------	-----

---

## 1 Introducción

1.1 Pensamiento computacional	2.1
1.2 Material	2.2
1.2.1 NodeMCU	2.2.1
1.2.2. Shield motor	2.2.2
1.2.3 Chasis	2.2.3
1.2.4 DHT22	2.2.4
1.2.5 Hall	2.2.5
1.2.6 LDR	2.2.6
1.3 Conexiones	2.3

---

## 2 ARDUINOBLOCKS

2.1 Introducción	3.1
2.2 Cuentas alumnos	3.2
2.3 Driver	3.3
2.4 Empezando	3.4
2.5 Intermitente	3.5
2.6 Sensores	3.6
2.7 Motores	3.7

---

## 3 BLYNK

3.1 Qué es	4.1
3.2 Crear cuenta	4.2
3.3 Endender LED	4.3
3.3.1 Token	4.3.1
3.3.2 Button	4.3.2
3.3.3 Arduinoblocks	4.3.3
3.4 Endender gradualmente LED	4.4
3.5 Medir luz	4.5
3.6 Medir T y H	4.6
3.7 Medir campo magnético	4.7
3.8 Motores	4.8

---

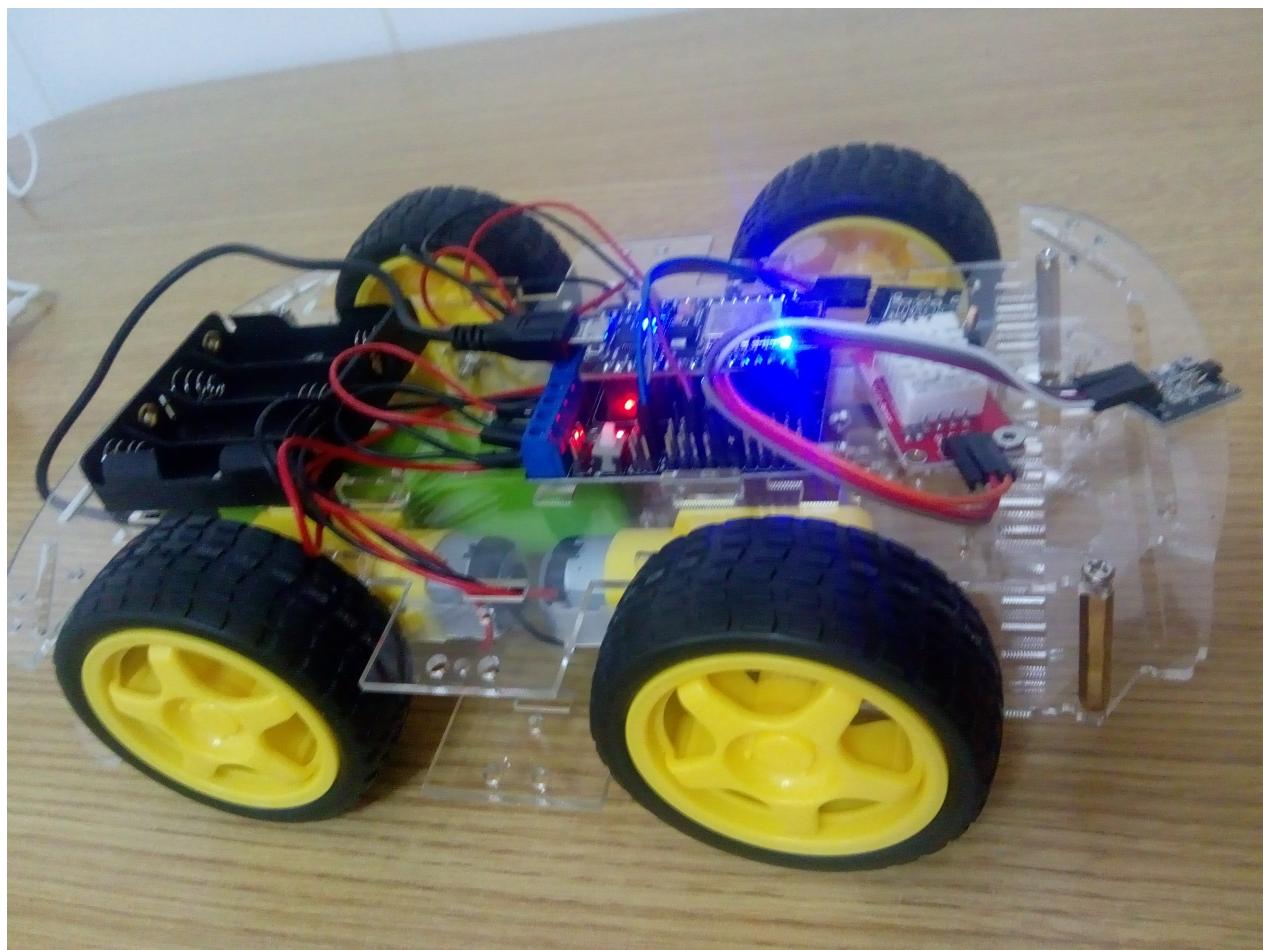
## 4 AVANZADO: SERVIDOR BLYNK LOCAL

---

4.1 Qué es	5.1
4.2 Primer paso: Raspberry	5.2
4.2 Segundo paso: Java	5.3
4.3 Tercer paso: Blynk	5.4
4.4 Cuarto paso: Panel control	5.5
4.5 Quinto paso: En la APP	5.6
Créditos	6.1

---

## ROVER MARCIANO CON NodeMCU ARDUINOBLOCKS y BLYNK

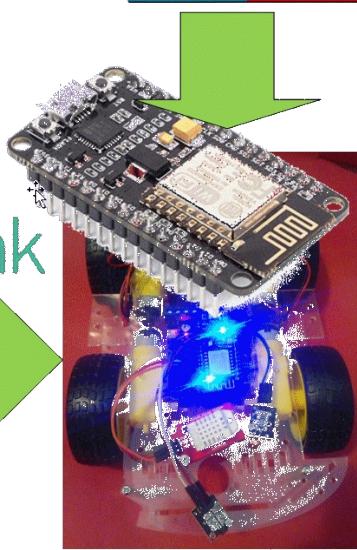


En este curso trataremos de manejar el rover propuesto utilizando las siguientes herramientas:

1. Conocer la electrónica del rover: NodeMCU y sensores
2. Programación por bloques [ArduinoBlocks](#) para controlar la placa **NodeMCU**.
3. Manejo de la [App Blink](#) para controlar el rover por wifi y con el móvil.
4. AVANZADO: Creación de un **servidor local BLINK con Raspberry**

En resumen: Manejamos nuestro rover y visualizaremos sus sensores con una App.

Arduino Blocks



## Pensamiento computacional

¿Dónde se encaja este robot? ¿se puede comparar este robot con otros robots de otros cursos que hacemos desde CATEDU?

Esta es la hoja de ruta que proponemos, no se tiene que tomar al pie de la letra, pero intenta ayudar al profesorado que tenga una visión global de tanta oferta:

Como se puede ver ROVER CON ARDUINO tiene la ventaja de tener un precio razonable, y dentro del rango de programación en bloques del Arduino en primaria.

## RoboTICa

Oferta de formación en Pensamiento computacional del Centro Aragonés de Tecnologías para la Educación.



## MATERIAL

El precio de este kit es aproximadamente 50€

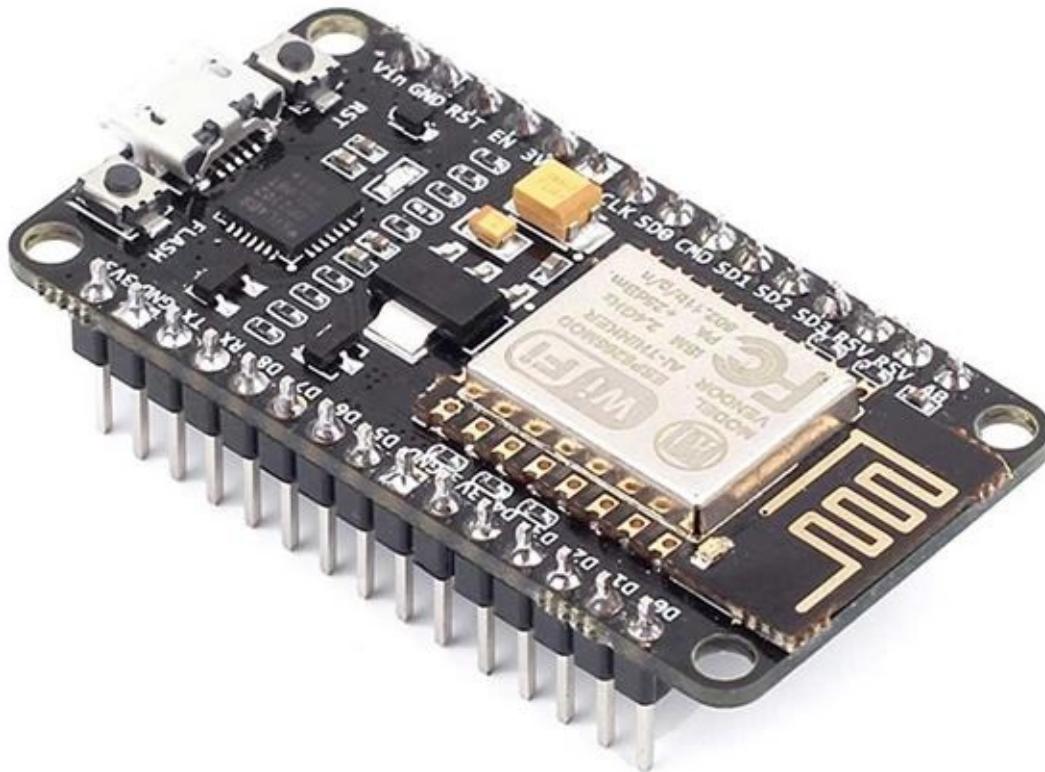
Lista de material : Hoja 1

Hoja 1

>

<

## NodeMCU



Es una placa de desarrollo basado en el SoC (System on a chip) ESP8266 e incorpora la comunicación WiFi que tanto falta en el Arduino.

Desde que salió el ESP01 como primer módulo del ESP8266, se ha evolucionado mucho, mejorando la popularidad, potencialidad y bajando los precios. Actualmente (2020) se fabrica el ESP32.

Nosotros utilizamos en el Rover el NodeMCU basado en el ESP12E conocida como **NodeMCU** que apareció en el 2014 que se programaba con [Lua](#).

Con el paso del tiempo esta placa se programa en varios lenguajes, siendo la más interesante desde el punto pedagógico que se pueda programar en el didáctico **entorno Arduino**.

Existen disparidad en los fabricantes, en este curso se va a utilizar la generación segunda de 8 pines

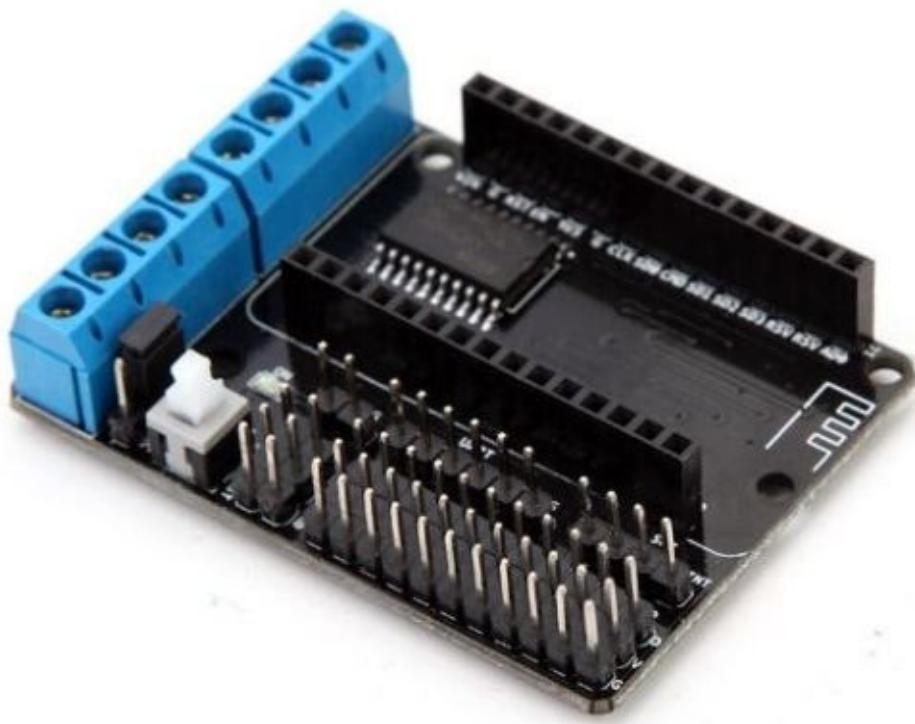
Generación	Versión		Módulo	Anchura	Comentario
Primera	0.9		ESP12	10 pines	La original, no está a la venta
Segunda	1.0	V2	ESP12E	8 pines	Versión "oficial"
Tercera	1.0	V3	ESP12E	10 pines	Versión de Wemos/Lolin. Conversor CH340G

De [Luis Llamas](#) CC-BY-NC-SA

Para más información recomendamos la página de Luis Llamas (<https://www.luisllamas.es/esp8266-nodemcu/>)



## Shield motor para NodeMCU



Esta shield permite conectar motores en los 4 primeros terminales azules con el NodeMCU que se coloca en el zócalo. Para saber cómo funcionan los motores con esta shield [ver aquí](#)

También tiene unos machos para las entradas y salidas digitales del NodeMCU y uno para la entrada analógica. Esto lo veremos en [conexiones](#).

## chasis



Se elije este chasis por su bajo coste y con tracción 4x4 (en marte no hay carreteras asfaltadas).

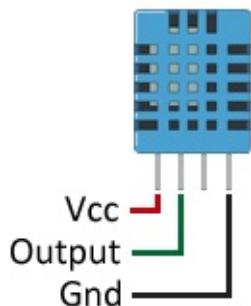
## DHT22



Créditos: CC-BY-NC-SA [Luis Llamas](#)

Es un sensor que mide de forma sencilla tanto la temperatura como la humedad. Este sensor aparece en el [curso de Aularagón Arduino con código](#)

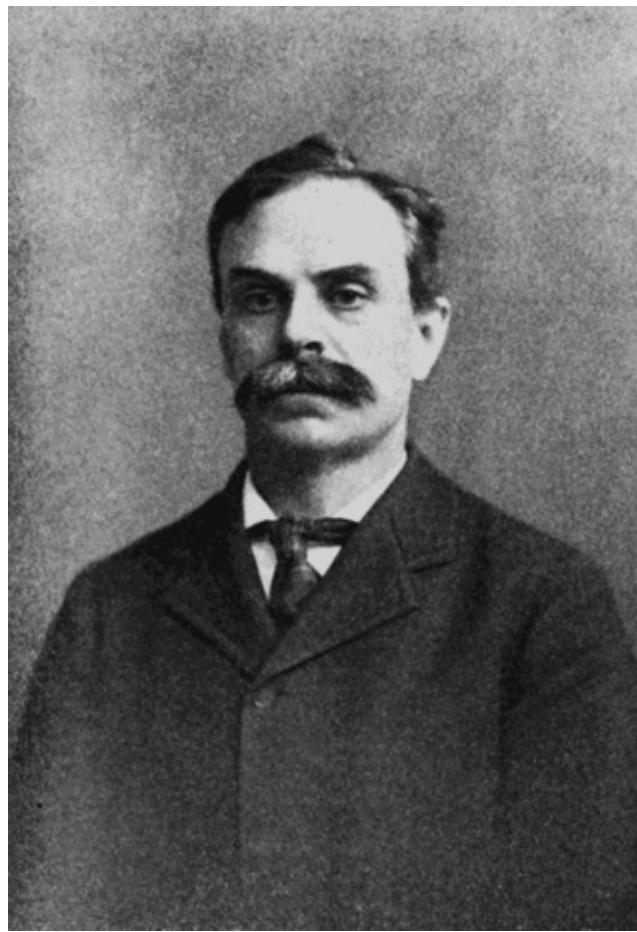
Su conexión es sencilla :



Créditos: CC-BY-NC-SA [Luis Llamas](#)

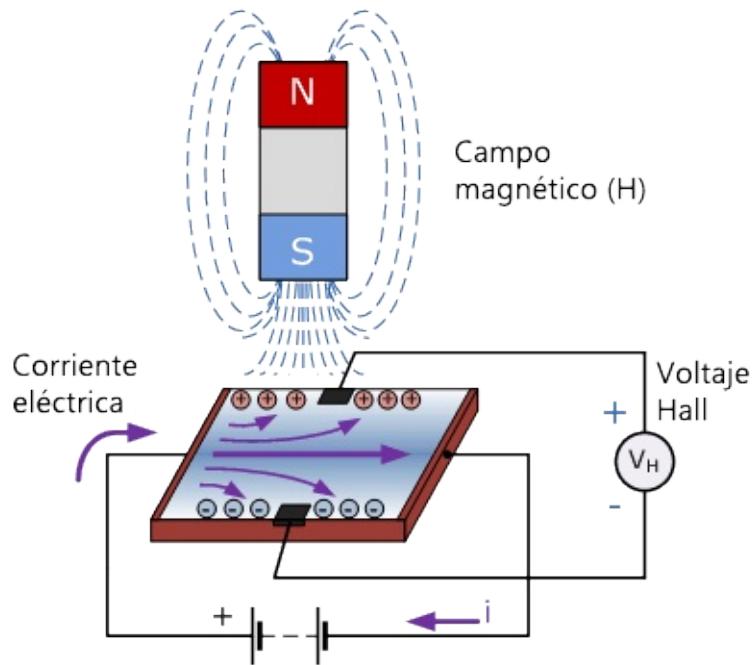
Para saber más recomendamos la página de [Luis Llamas](#)

## Sensor efecto Hall



De Desconocido - Popular Science Monthly Volume 64, Dominio público

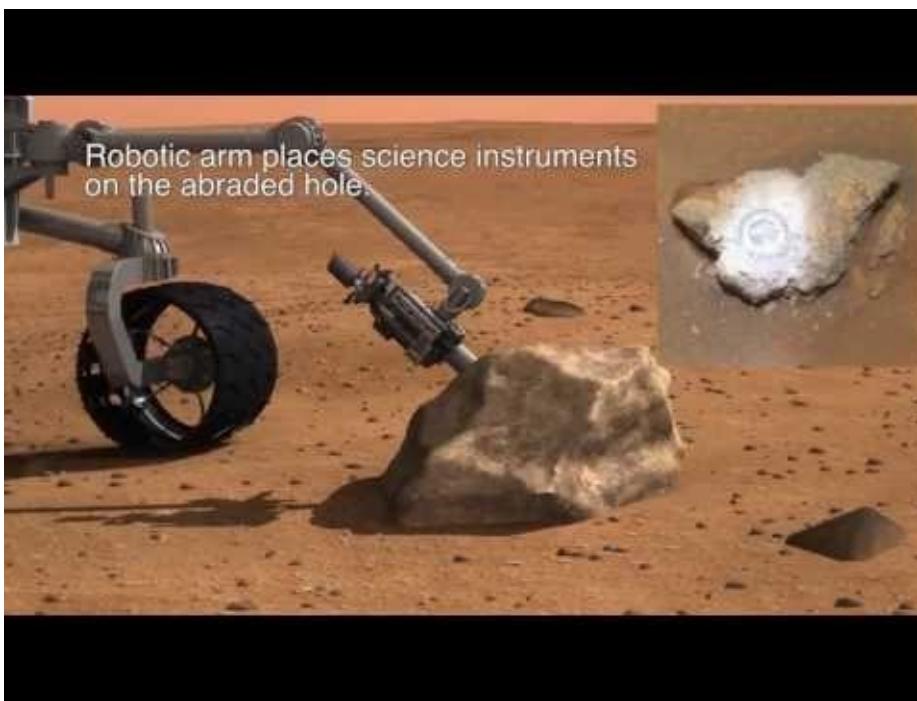
[Edwin Herbert Hall](#) descubrió en 1879 que en presencia de un campo magnético, un conductor que conduzca una corriente se le producía un campo eléctrico porque las cargas eléctricas se desviaban de su trayectoria principal, nuestro sensor simplemente mide ese campo eléctrico:



De [Luis Llamas CC-BY-NC](#)

El sensor tiene un led de color rojo que indica que hay una lectura de campo magnético.

Nuestro rover analizará si las rocas que hay cerca de él tienen ferritas midiendo su componente magnético. Una aproximación tosca de los rovers actuales pero sirve para acercar el mundo de la medición remota al alumnado :



[Video link](#)

Nuestro robot detectará si hay un imán cerca o no .

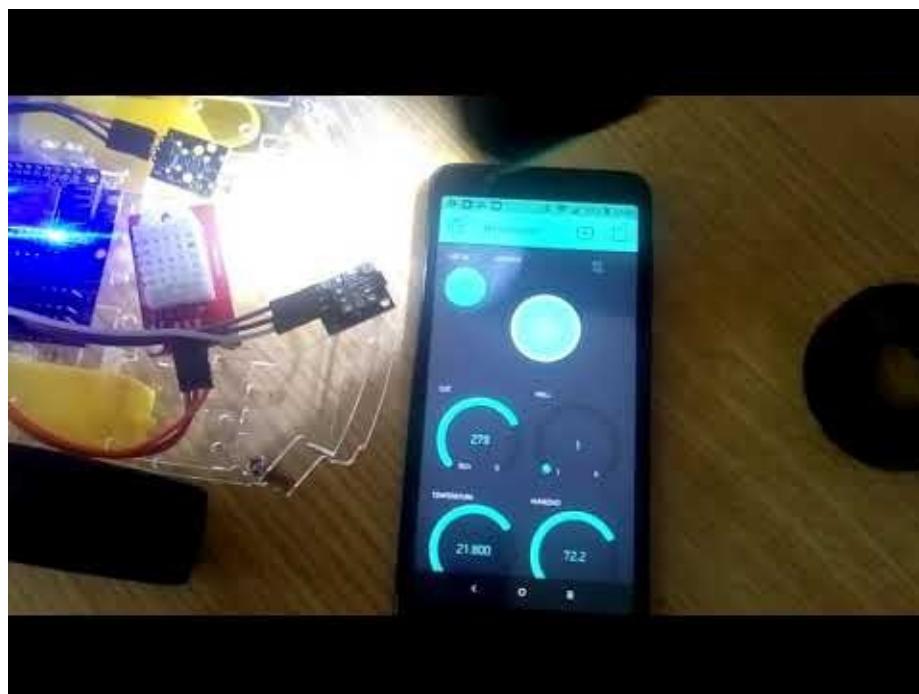


[Video link](#)

## Sensor de luz LDR.



El LDR es una resistencia que varía con la luz. La conexión interna de este sensor analógico hace que **cuanto más luz menos valor nos proporciona** por lo tanto va al revés. Esto será importante para la configuración de nuestros programas, por ejemplo el medidor "Gauge" de la APP Blink de este vídeo va de 1023 a 0 para medir correctamente la cantidad de luz:

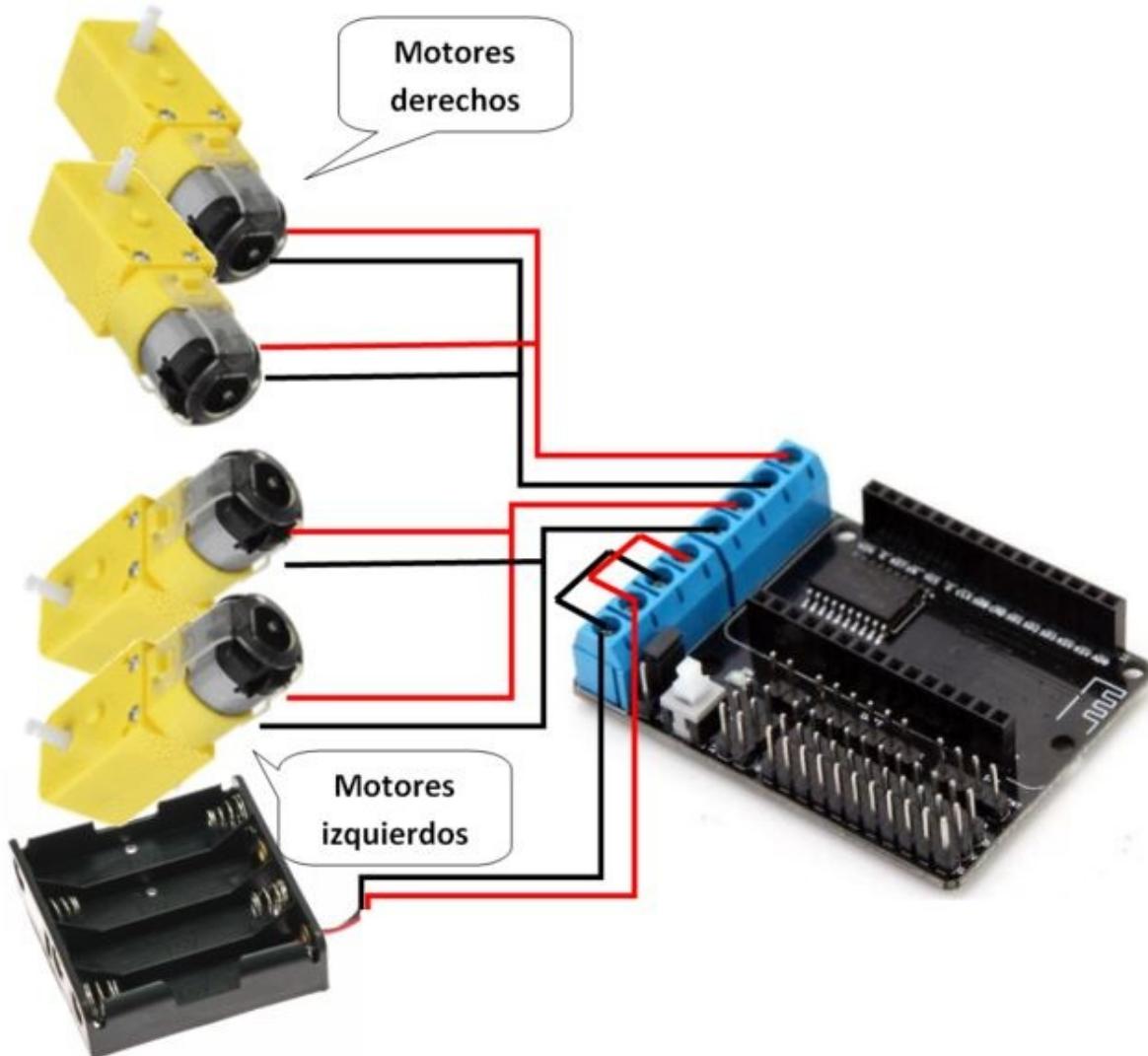


[Video link](#)

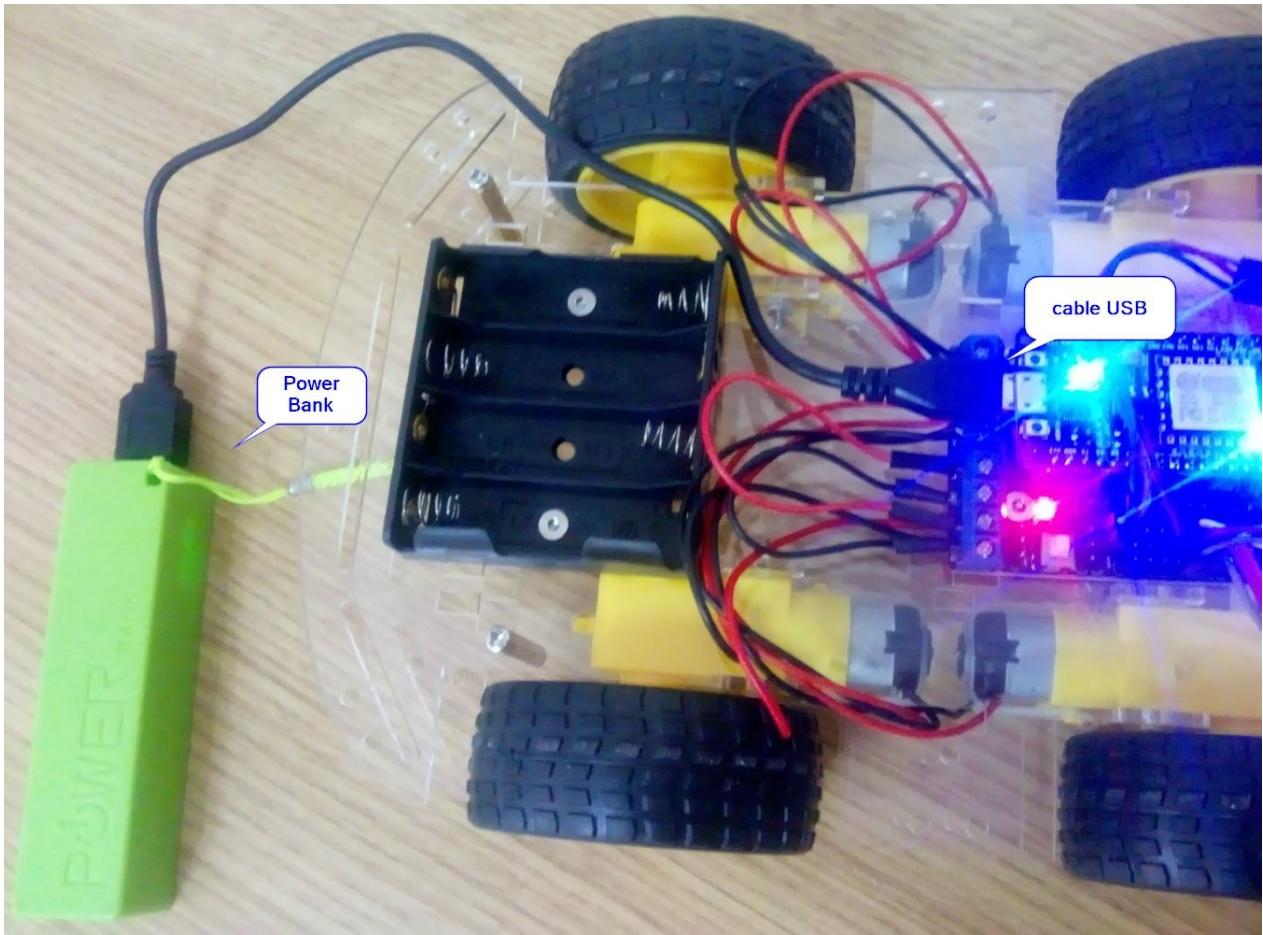
## Conexiones.

### Conexiones motores y batería

Este es el esquema de conexión de los motores y de la batería con el Shield :



Una alternativa a la conexión de la batería es utilizar un PowerBank y diréctamente al NodeMCU como podemos ver en la foto :



Otra posibilidad es utilizar baterías 18650 [ver](#) pero ojo con usar estas baterías [ver+](#)

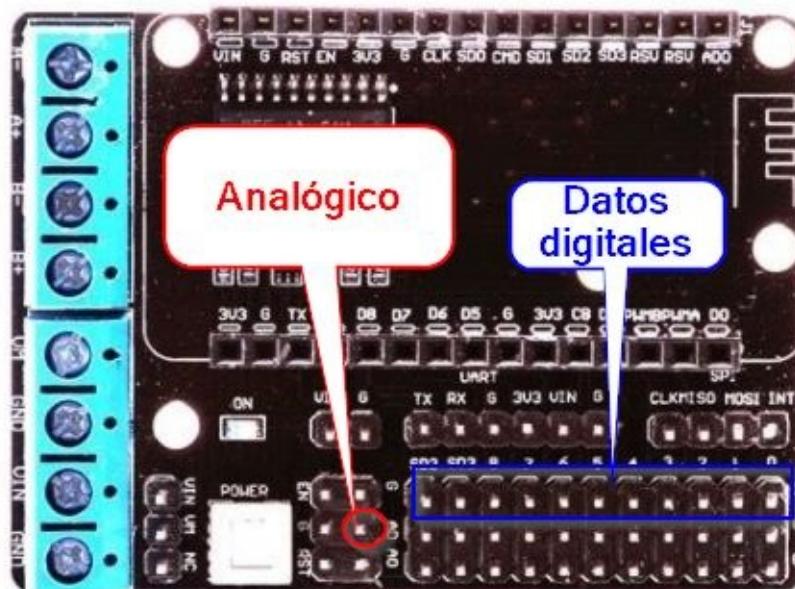
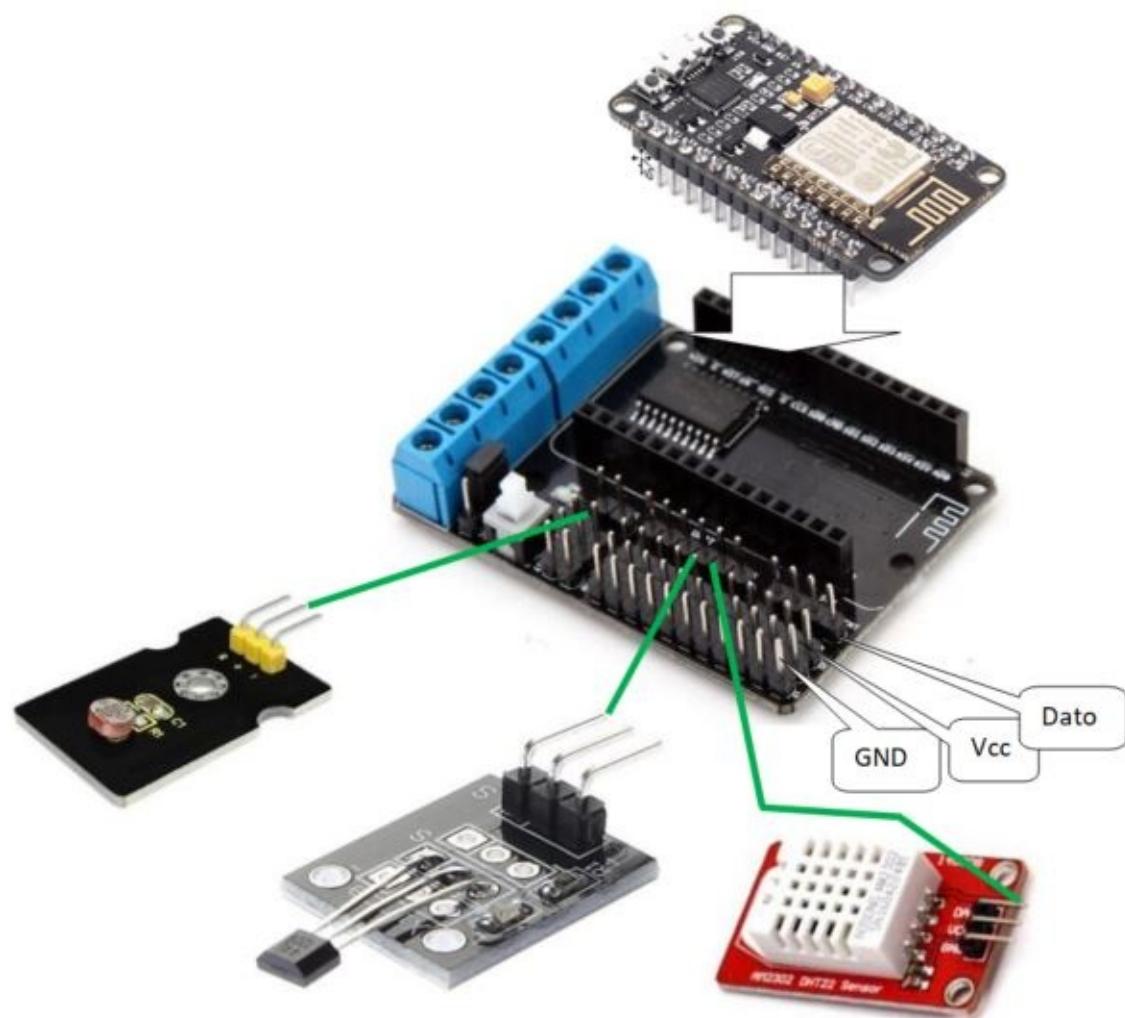
## Conexiones NodeMCU sensores con la Shield.

El sensor de efecto Hall y el sensor DHT22 lo conectaremos a las entradas digitales, nosotros hemos elegido

- DHT22 a D5
- Efecto Hall al D6

Es recomendable no usar D4 pues lo utilizaremos como Led

El sensor de luz LDR al ser analógico, lo conectaremos en la única entrada analógica que tiene esta shield



## ARDUINOBLOCKS



Es un programa *online*\* creado por el profesor Juan José López Almendros:



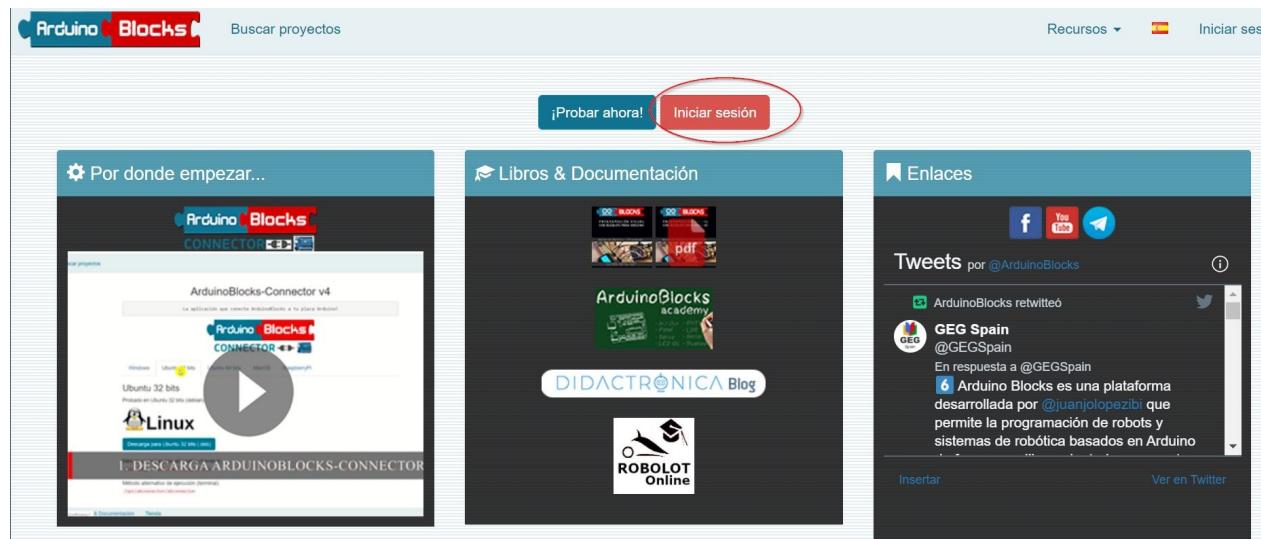
De [Juan José López Almendros](#) CC-BY-SA

Es una programación **gráfica** al estilo de Scratch y está pensado para utilizar a partir de niños y niñas de 8 años.

### Crear cuenta

Entramos en <http://www.arduinoblocks.com/>

e iniciamos sesión



Y rellenamos el formulario

**Nuevo usuario**

\*\*\* Recommended **GMail** accounts (Review SPAM folder) \*\*\* (Hotmail,Msn,... may not work due to spam filters)

<b>Correo electrónico</b>	<input type="text"/>
<b>Confirmación de correo electrónico</b>	<input type="text"/>
<b>Clave</b>	<input type="text"/>
<b>Confirmación de clave</b>	<input type="text"/>
<b>Nombre</b>	<input type="text"/>
<b>Apellidos</b>	<input type="text"/>
<b>País</b>	SPAIN <input type="button" value="▼"/>
<b>Ciudad</b>	<input type="text"/>

## Gestión de cuentas gestionadas para alumnos.

Tal y como dice el tutorial de Juanjo López :

*Permite a un usuario registrado con email, crear y administrar nuevas cuentas de usuario dentro de una organización, centro educativo o institución.*

Recomendamos seguir el tutorial de Juanjo López

[https://github.com/arduinoblocks/documentacion/blob/master/usuarios\\_gestionados.pdf](https://github.com/arduinoblocks/documentacion/blob/master/usuarios_gestionados.pdf)

## Arduinoblocks conector.

Para poder usar la herramienta Arduinoblocks tenemos que ejecutar antes **Arduinoblocks conector**.



Lo instalamos y lo ejecutamos y aparece esta ventana :

The screenshot shows the 'ArduinoBlocks-Connector' application window. The title bar says 'ArduinoBlocks-Connector'. The main area is a terminal window displaying the following text:

```

>> ArduinoBlocks-Connector v4
>> by Juanjo Lopez
>> www.arduinoblocks.com
>> Listening on port 9987
>> (Ctrl+C to finish)
>> Ready...
Checking libraries...
Libraries version: 32

```

**ATENCIÓN**

No podemos cerrar la ventana mientras utilizamos Arduinoblocks, la minimizamos simplemente.

En caso contrario, Arduinoblocks no se puede comunicar con nuestro Arduino, NodeMCU, ...

## Empezando un proyecto

Entramos en Proyectos y podemos ver nuestros proyectos creados como también empezar uno.

Cuando creas un proyecto, ten en cuenta que puedes hacer:

- Proyectos personales privados
- Proyectos personales públicos
- Proyectos para tu clase, donde lo compartes con tus alumnos y los puedes supervisar y corregir.

Lo primero que tenemos que elegir es para qué tipo de placa se hace el proyecto. En nuestro caso es para **NodeMCU**

Nuevo proyecto personal

Tipo de proyecto
Nombre
Descripción

Arduino Uno  
Arduino Nano / ATmega328  
Arduino Nano / ATmega328 (new bootloader)  
Arduino Mega / 2560  
Arduino Leonardo  
Arduino Pro Micro  
Imagina TDR STEAM  
3dBot / Imagina-Arduino  
Keyestudio EasyPlug  
Keyestudio KeyBot  
Otto DIY / Nano  
Otto DIY / Uno  
**NodeMCU / ESP8266**  
WeMos D1 / ESP8266  
ESP32 / ESP32-WROOM

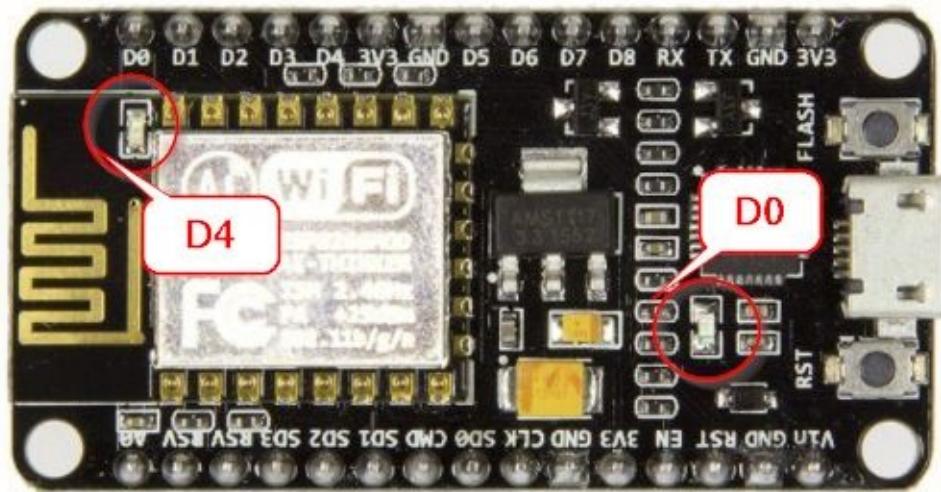
Luego el nombre y el resto de campos es optativo pero importante y buena costumbre rellenarlos, sobre todo si el proyecto lo compartimos:

- Descripción
- Componentes
- Comentarios

## Intermitente en D4

Nuestro primer programa que vamos a hacer es un intermitente en D4

¿Por qué con D4 y no con otro? Porque D4 tiene un led integrado en la placa, también D0.



### ATENCIÓN

Estos leds van al revés, (configuración pop-down) luego para encender el led, D4 tiene que estar a nivel bajo, y al revés para apagarlo, D4 en nivel alto.

El programa es muy sencillo y lo completamos que se pueda visualizar por la salida puerto serie, por eso al iniciar ponemos el puerto serie a 9600 baudios.



Le damos a **subir** (teniendo el programa Arduinoblocks conector minimizado, eso lo podemos ver enseguida pues detecta en que COM está conectado, en la figura sale COM5) y en **Cónsola** podemos ver el contenido del puerto serie.



## Sensores

La ventaja de ARDUINOBLOCKS es que tiene ya incorporado una cantidad de bloques con sus librerías y funciones integradas por lo que el alumno no tiene que pelear con códigos y situaciones especiales, simplemente arrastrar el bloque y funcionar:



## Probando el sensor LDR.

Vamos a probar este sensor que está conectado en una entrada analógica, para ello simplemente que escriba su valor en el puerto, y veremos cómo va cambiando.

Es recomendable para la lectura de los sensores, crear variables que almacenen los valores leidos, para luego utilizar estos valores:

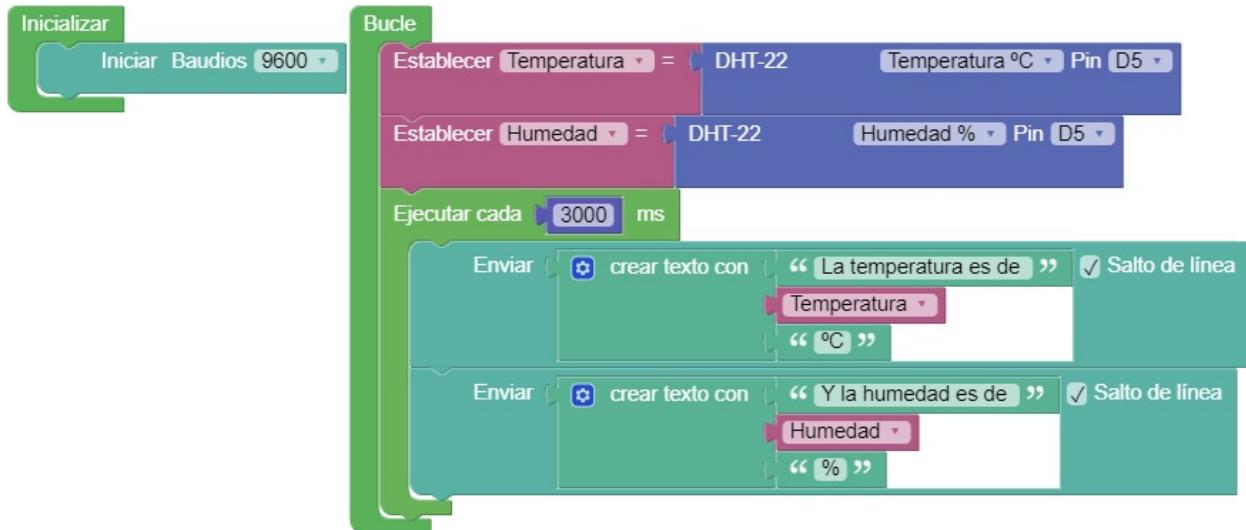


El siguiente código lee el sensor cada 2 seg. y lo vemos por el puerto serie. Podemos comprobar que **cuanto más luz, la lectura es menor**.



## Probando el sensor DHT22.

Vamos a probar este sensor, que mide a la vez temperatura y humedad. Con el bloque correspondiente, nos ahorramos bastante código.



## Probando el sensor Hall.

Este sensor tiene también su bloque :



Podemos construir un programa análogo al anterior y veremos que su salida es simplemente 0 o 1 depende si acercamos o no un imán.

# MOTORES

Los motores **derechos** se gobiernan con la siguiente instrucción :



- Activando el pin D1 encendemos los motores
- El pin D3 en ON van hacia delante, si queremos que vayan hacia atrás, ponemos D3 en OFF
- El valor PWM del pin D1 es la potencia que transferimos al motor, puede ser desde 0 hasta 1023

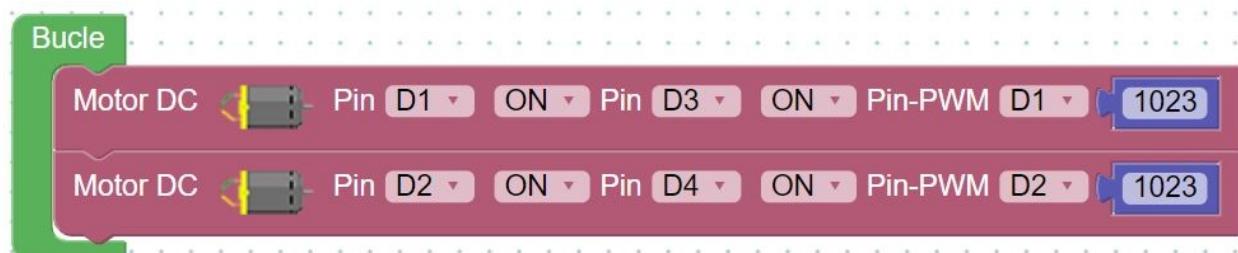
Recomendamos leer [esta página](#) sobre el significado de las salidas PWM.

Para los motores **izquierdos** se gobiernan con la siguiente instrucción :



- Activando el pin D2 encendemos los motores
- El pin D4 en ON van hacia delante, si queremos que vayan hacia atrás, ponemos D4 en OFF
- El valor PWM del pin D2 es la potencia que transferimos al motor, puede ser desde 0 hasta 1023.

Recomendamos ejecutar estas instrucciones y jugar con estos valores para ver si están bien conectados los motores :



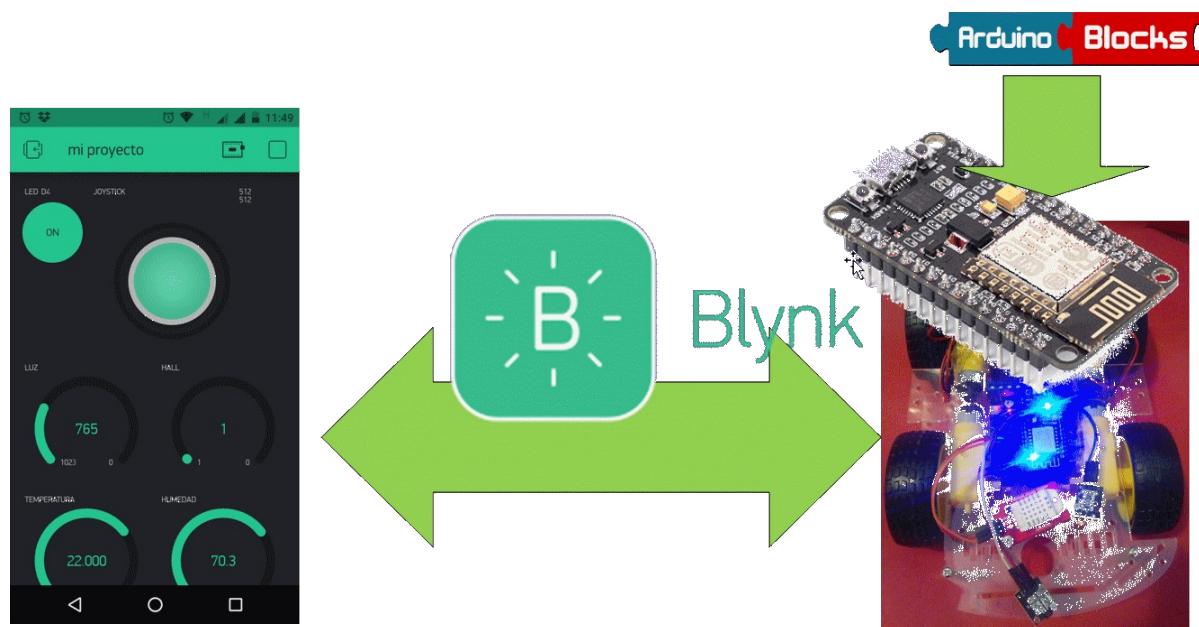
## BLYNK.

BLYNK pertenece al concepto de IoT Internet de las cosas



De Drawed by Wilgengenbroed on Flickr Translated by Prades97 CC BY-SA 3.0

Nos posibilita la conexión de un hardware Arduino, Raspberry, NodeMCU ... con la APP que puedes poner diferentes elementos de control.



Las especificaciones se muestran [aquí](#)

**A Smartphone**

- Android OS version 4.2+
- iOS version 9+

Blynk doesn't run on Windows Phones, Blackberries and other dead platforms. Sorry.

You can also run Blynk on emulators

**IoT Hardware**

Blynk can run on over 400 hardware modules. The most popular are:

- ESP8266
- ESP32
- NodeMCU
- Arduino (any model)
- Raspberry Pi (any model)
- Particle (any model)

[Full list of supported hardware →](#)

**Internet Connection**

To connect your hardware to the Internet, you can choose almost any module either built-in, or external shields

Supported connectivity

- WiFi
- Ethernet
- Cellular (GSM, 2g, 3g, 4g, LTE)
- Serial
- USB via your PC
- Bluetooth (BETA)

Y con la APP se hace un panel de control de nuestro rover, Arduino, etc.. en un tiempo record:

**Blynk para Arduino,ESP8266,RPi**

Blynk Inc. Herramientas

★★★★★ 8.371

PEGI 3

Ofrece compras en la aplicación

Esta aplicación está disponible para tu dispositivo

Instalada

## OJO

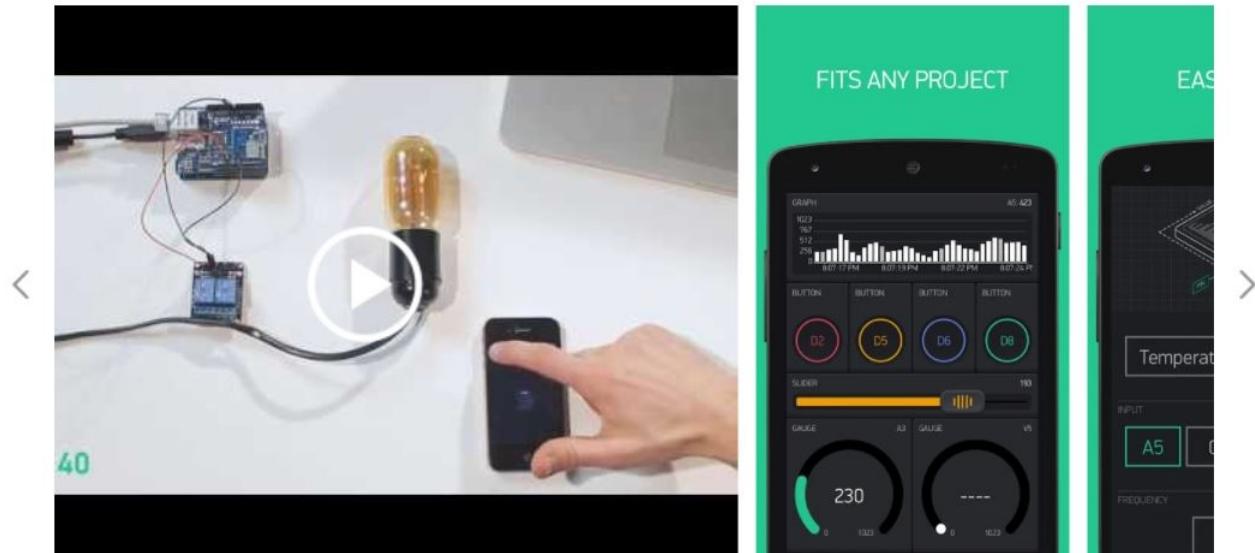
Tiene limitaciones: La APP Blynk no nos permite poner todos los elementos de control que queramos.

Esta "pega" se soluciona en el apartado 4 AVANZADO pues se aprovecha que es de [código abierto](#)

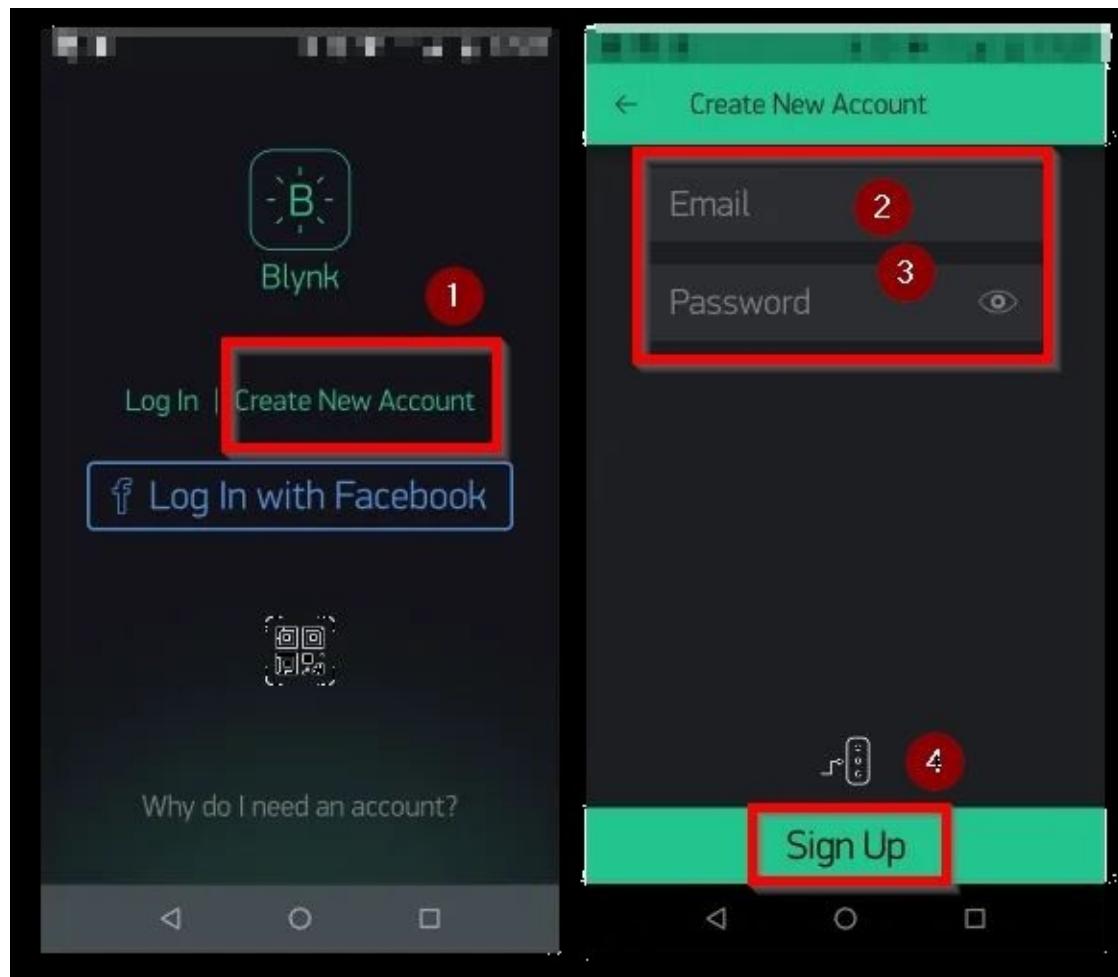
## CREAR CUENTA EN BLYNK.

Vamos a usar el propio servidor Blynk que nos dará limitaciones en los diferentes elementos a poner. Esto se solucionará en el apartado 4 AVANZADO

Descargamos en nuestro móvil o tableta la APP BLYNK

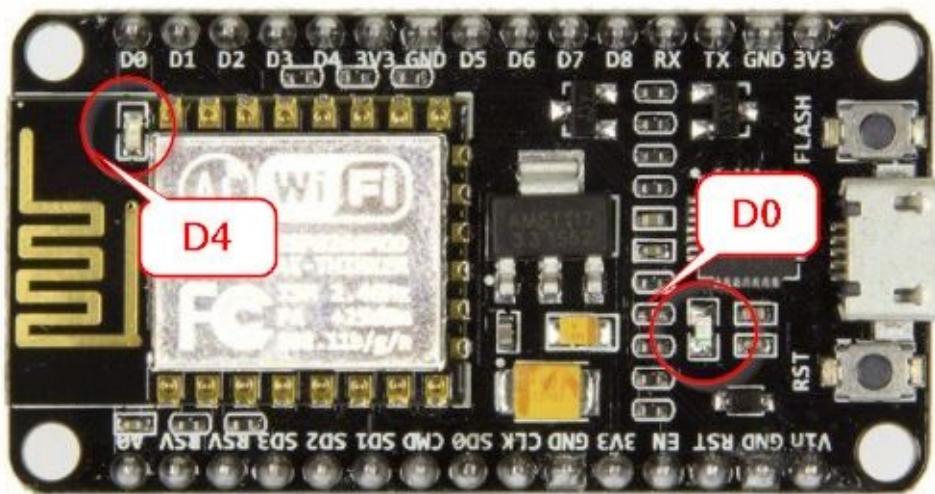


Para crear una cuenta, entramos en la primera pantalla, "Sign Up" y en la siguiente entramos un email verdadero y una contraseña

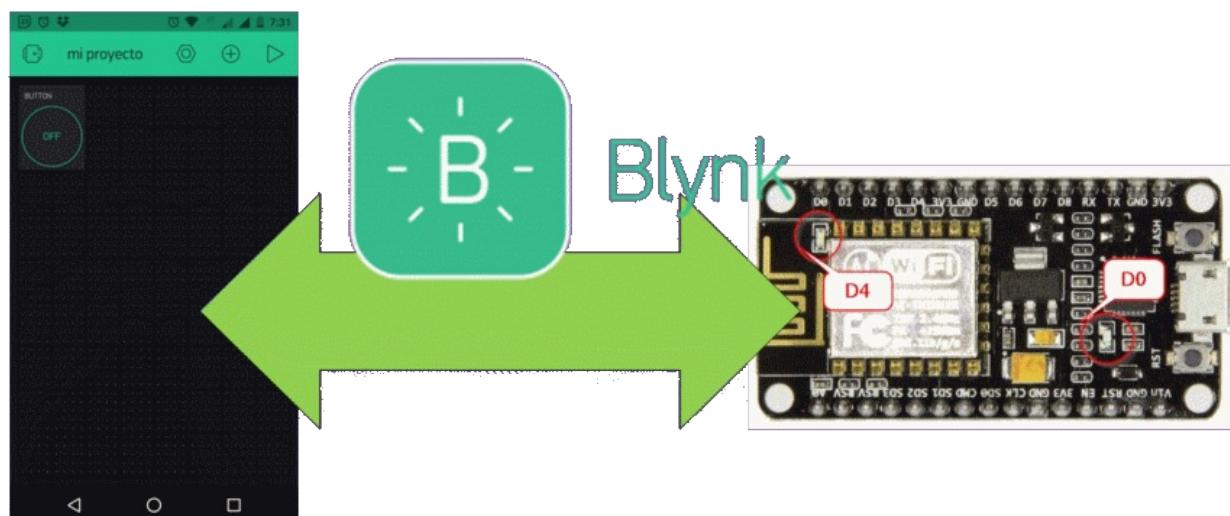


## Primer programa con Blynk: Encender D0

Si antes hicimos un intermitente por D4, ahora vamos a encender el LED en D0:

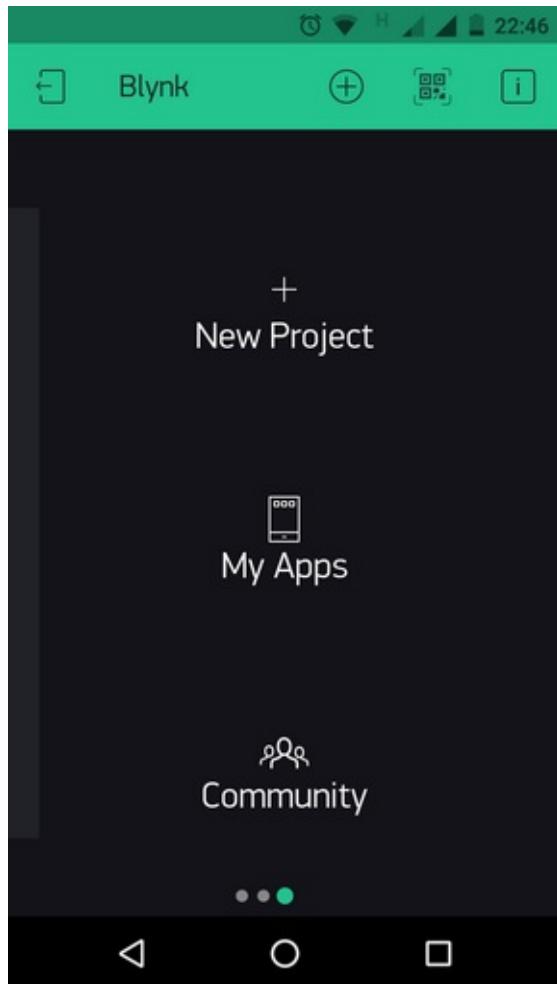


Y el servidor BLYNK hará de puente entre nuestra App y el NodeMCU :

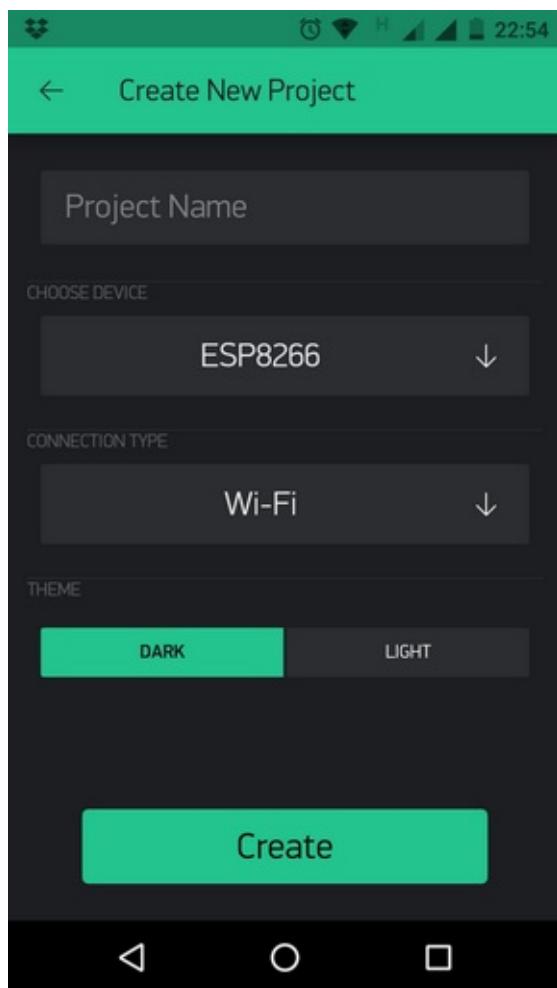


## Primer programa con Blynk: Encender D0 : 1 Conseguir el Token

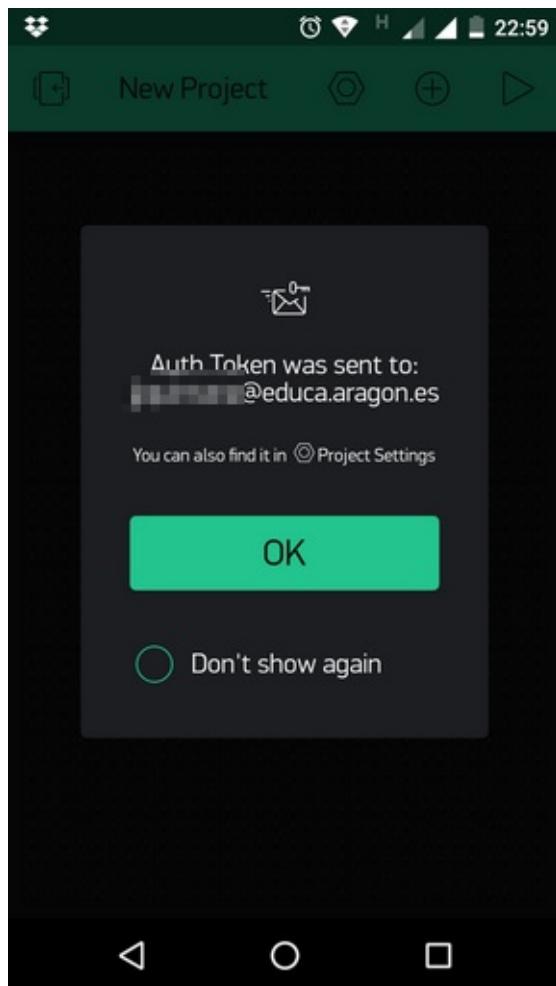
Empezamos en la APP Blynk creamos un nuevo proyecto:



En la siguiente pantalla pregunta por el dispositivo, le decimos ESP8266 y le ponemos un nombre:



Y nos envía un **Token** a nuestro email



En el email tenemos nuestro Token:

Auth Token : 61kVL1kkxnEUcG3p4O\_e-Y-pFwatqGcQ

Happy Blynking! - Getting Started Guide -> <https://www.blynk.cc/getting-started> Documentation -> <http://docs.blynk.cc/>  
Sketch generator -> <https://examples.blynk.cc/>

Latest Blynk library -> [https://github.com/blynkkk/blynk-library/releases/download/v0.6.1/Blynk\\_Release\\_v0.6.1.zip](https://github.com/blynkkk/blynk-library/releases/download/v0.6.1/Blynk_Release_v0.6.1.zip) Latest  
Blynk server -> <https://github.com/blynkkk/blynk-server/releases/download/v0.41.13/server-0.41.13.jar> - <https://www.blynk.cc>  
[twitter.com/blynk\\_app](http://twitter.com/blynk_app) [www.facebook.com/blynkap](http://www.facebook.com/blynkap)

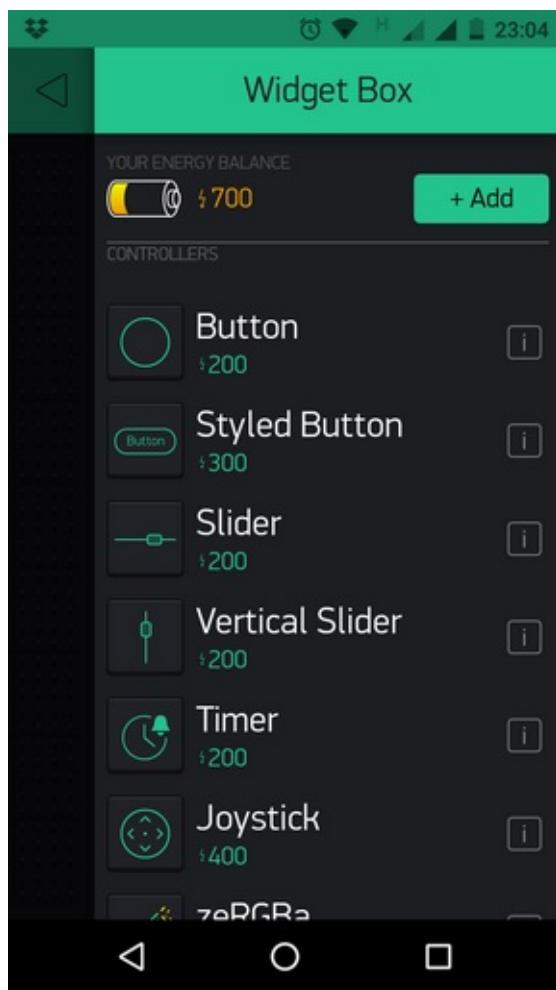
**Recuerda : Un proyecto en BLYNK = Un Token**

## Primer programa con Blynk: Encender D0 : 2 Poner los controles

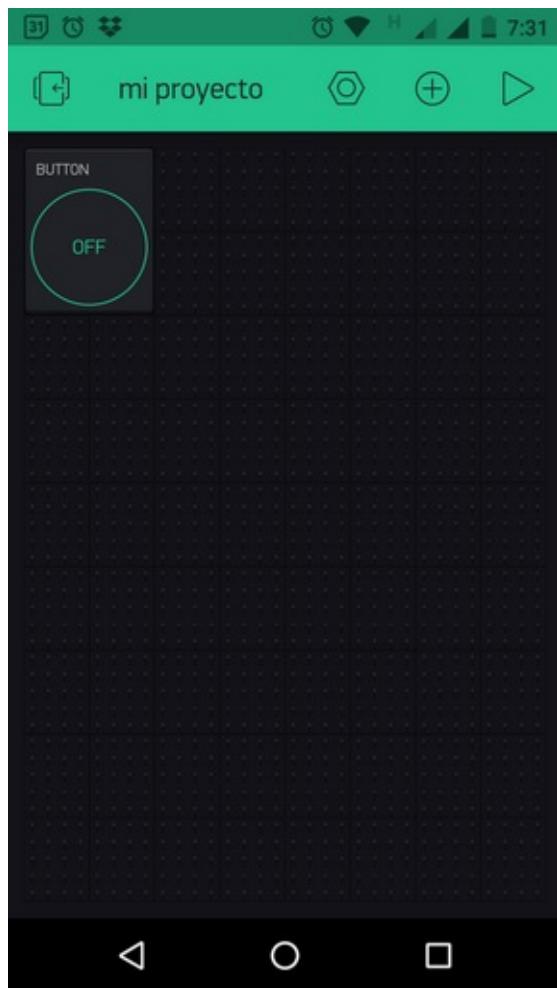
Una vez dentro del proyecto de la App aparece todo un panel si nada, añadimos controles apretando al + que hay en la parte superior



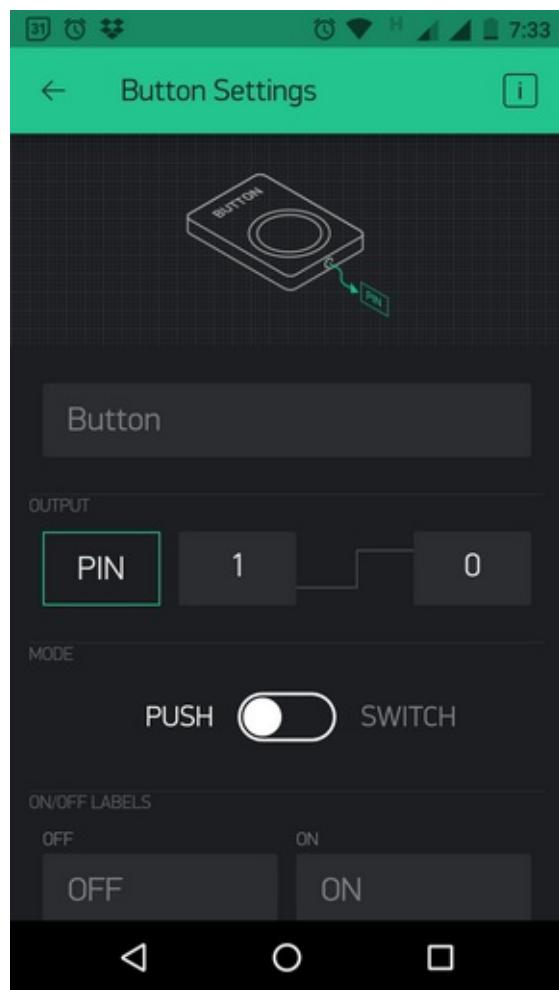
Seleccionamos el botón



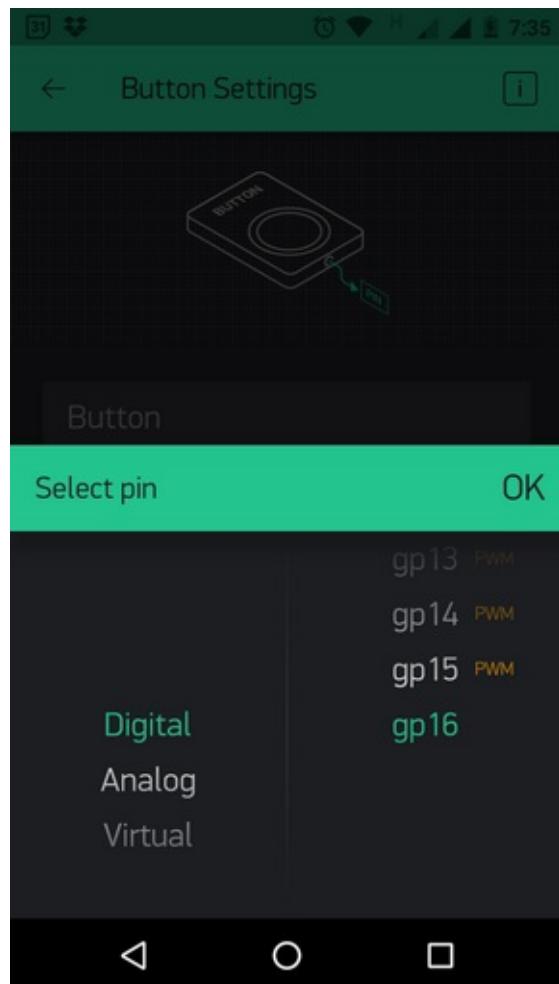
Nos aparece el botón



Pulsamos sobre él para entrar en sus propiedades y cambiamos que vaya de 1 a 0 pues recuerda que este pin va al revés



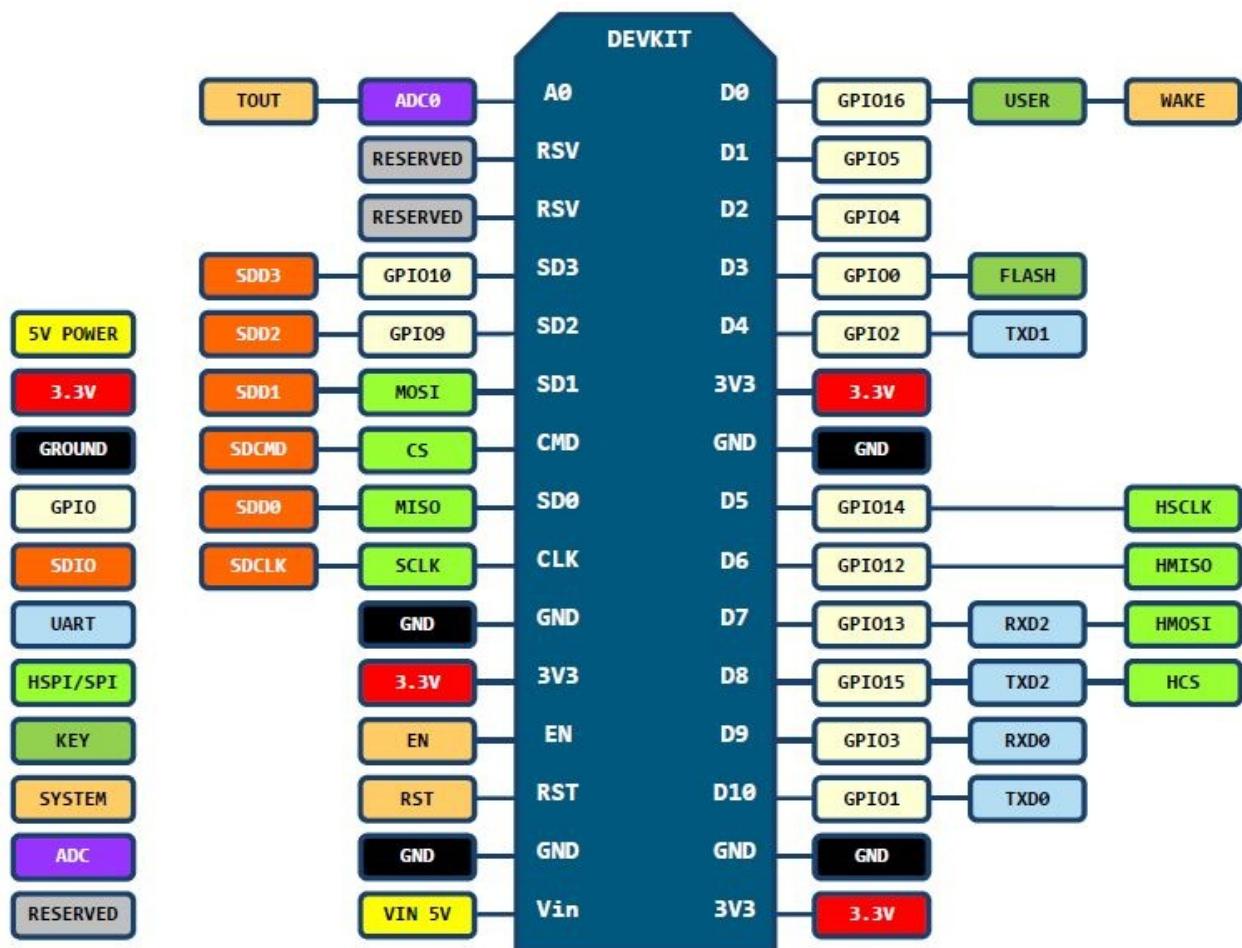
Pulsamos en PIN y le decimos que sea el D0



## ATENCIÓN

La nomenclatura de los pines cambia, el D0 es el GP16, viene de GPIO General Port Input Output.

El D4 es el gp2



## Primer programa con Blynk: Encender D0 : 3 ARDUINOBLOCKS

Ahora le tenemos que decir a nuestro NodeMCU la configuración de la Wifi, y el Token que hemos recibido, ¿cómo? muy fácil, con ARDUINOBLOCKS

Abrimos un proyecto y ponemos esta instrucción :



Donde :

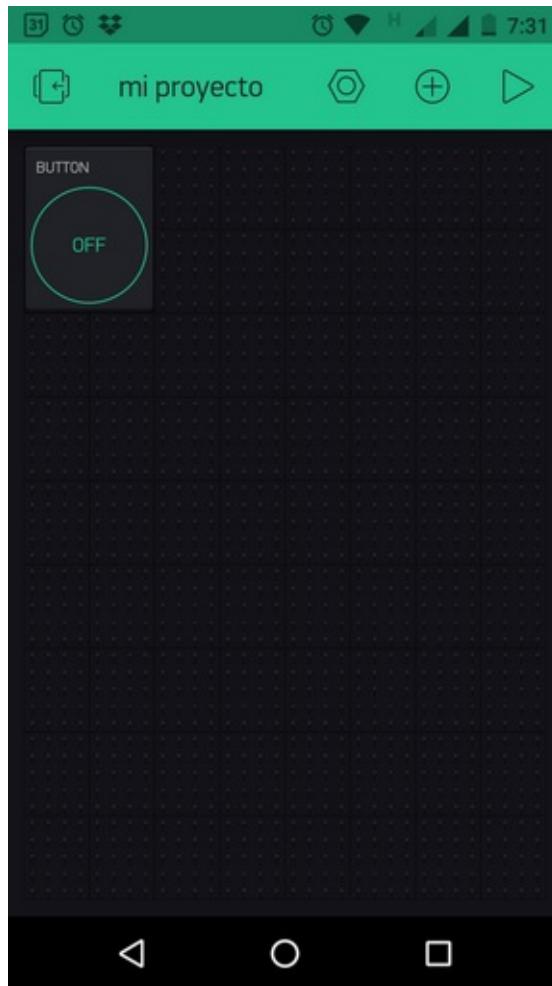
- **WIFI SSID** el nombre de la Wifi donde queremos que se conecte nuestro rover
- **WIFI CLAVE** la contraseña de la WIFI
- **IP Servidor** de momento ponemos 0.0.0.0 puerto 8080 pues vamos a utilizar el servidor Blynk oficial que hay en Internet, luego veremos en 4 AVANZADO que podemos cambiar este número por un servidor local.
- Código Auth es el código de autenticación para que haga caso a nuestro proyecto BLYNK que acabamos de hacer, es decir [el Token que hemos recibido](#).

### ¿Nada más?

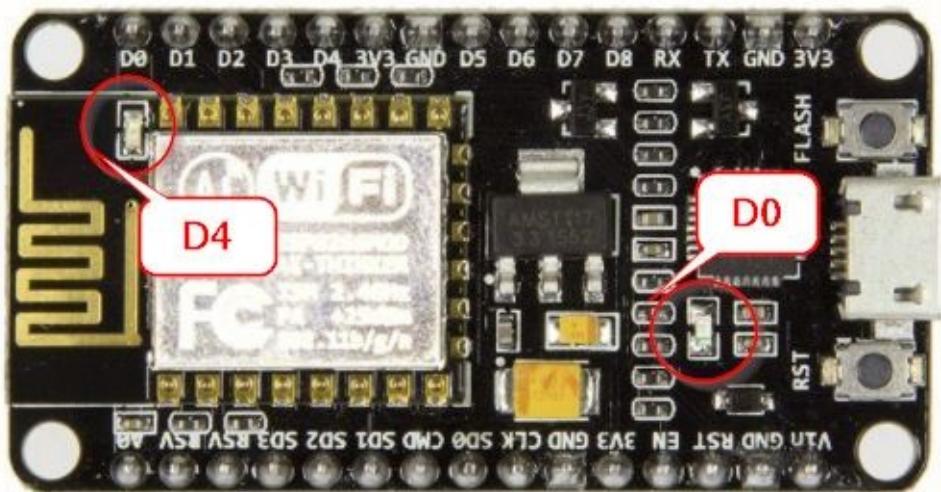
Nada más !! así de sencillo. Le damos a **subir** (teniendo el programa Arduinoblocks conector minimizado, eso lo podemos ver enseguida pues detecta en qué COM está conectado, en la figura sale COM5)



Vamos a la APP de BLYNK, Pulsamos al botón de play ► de arriba arriba. Luego al botón:



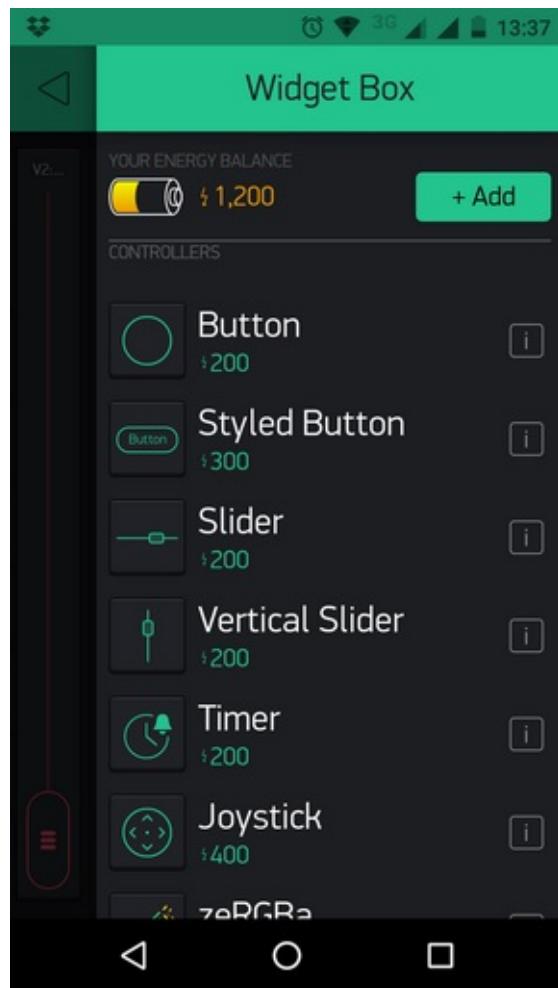
Y podemos ver cómo se enciende y apaga el led D0



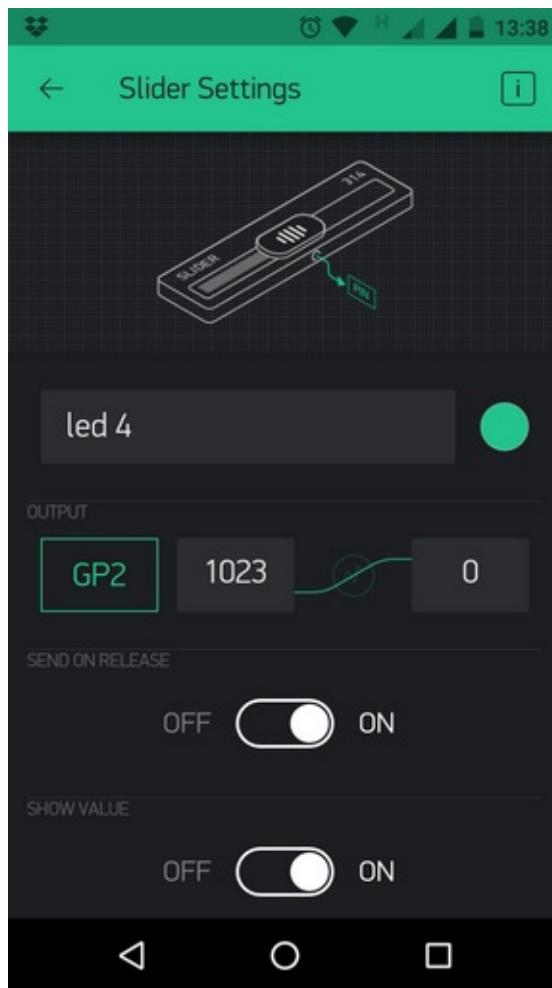
**Recuerda : Un proyecto en BLYNK = Un Token**

## Encender gradualmente el led D4

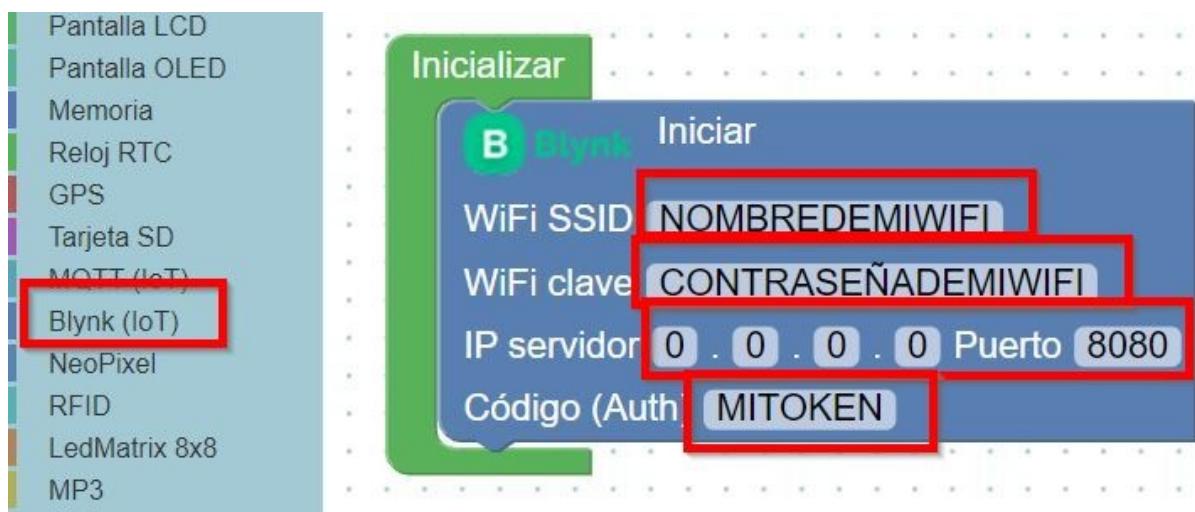
Para ello vamos a incorporar un Slider



Y de propiedades que vaya al GP2 y que vaya al revés de 1023 a 0

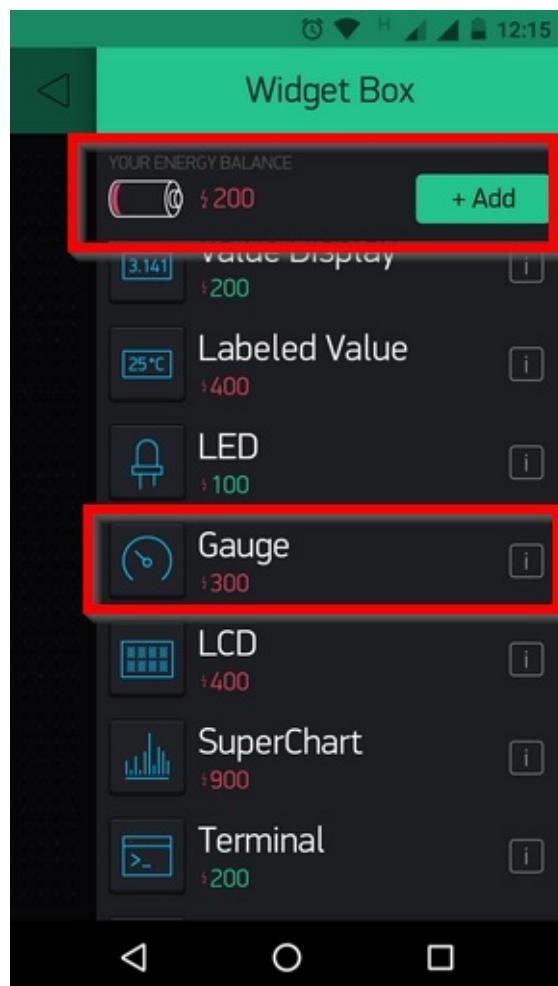


El programa en ARDUINOBLOCKS es el mismo !!! no hace falta tocar nada !!!



## Visualizar la luz LDR

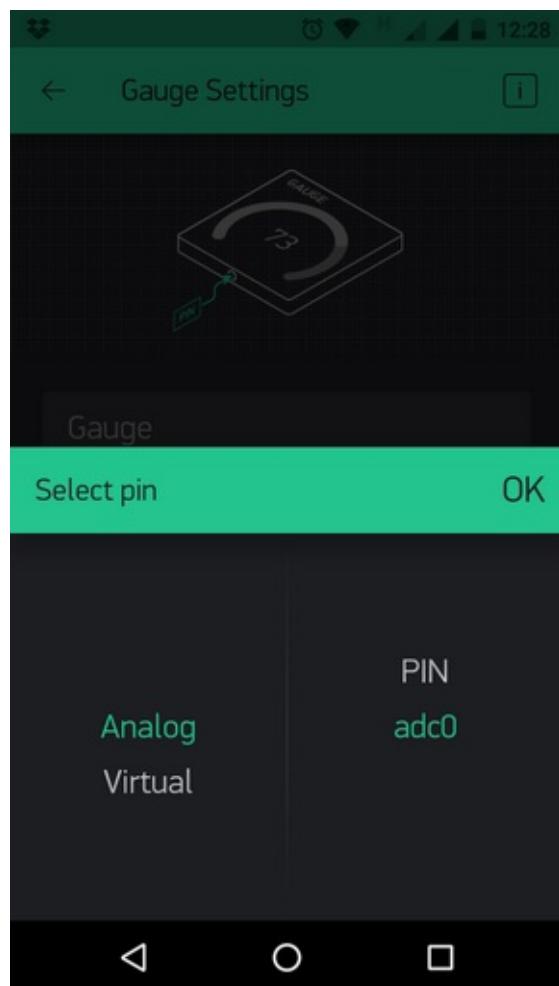
Ahora añadimos otro control Gauge



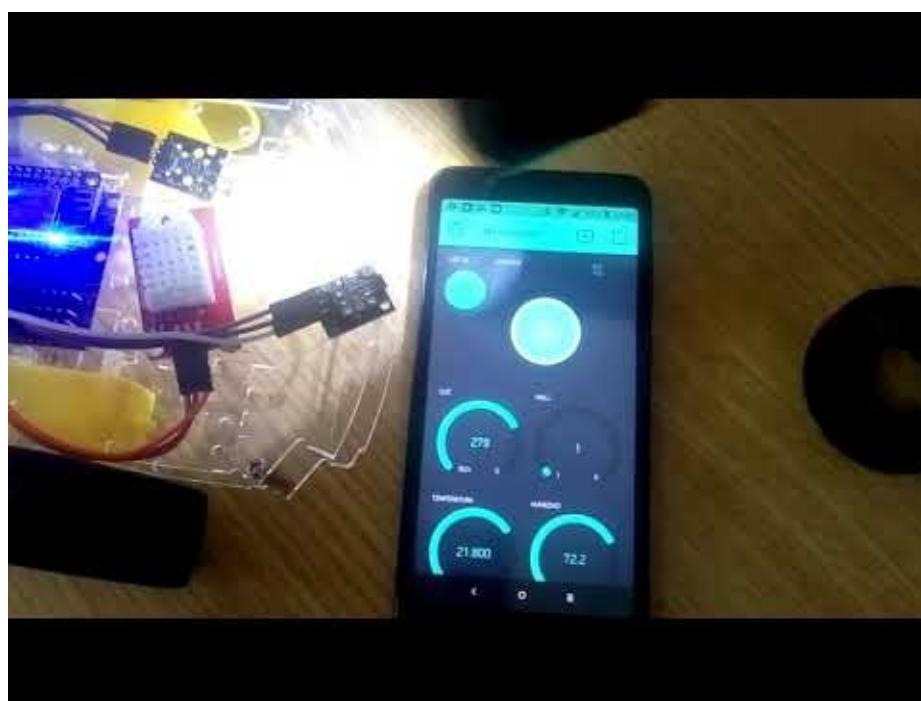
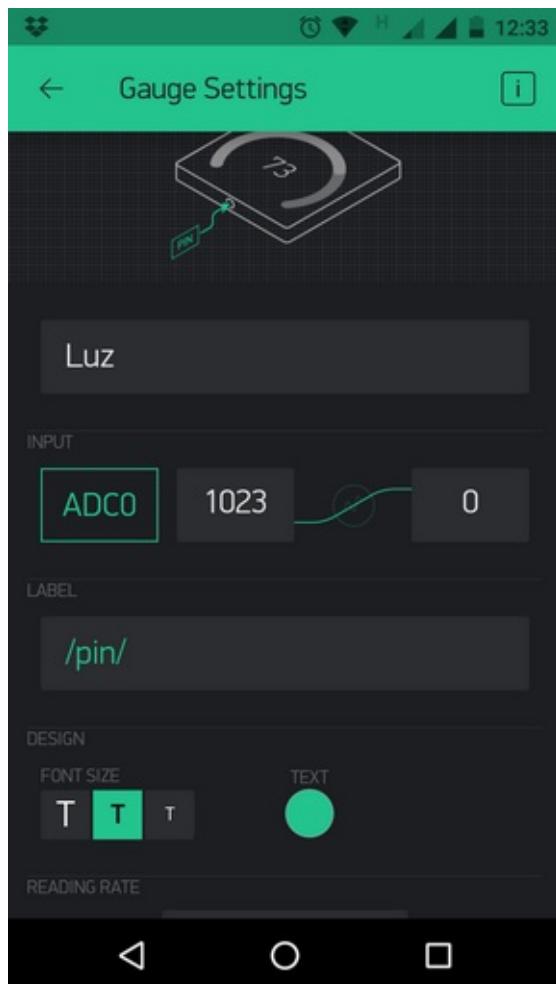
### #Nos estamos quedando sin energía

Como puedes ver tengo sólo 200 de energía, cada elemento de control tienen un coste de energía. Ya dijimos que esta APP versión gratuita tiene esta limitación que lo resolveremos en 4 AVANZADO

Elegimos de PIN el analógico (sólo hay uno). [Recuerda](#) que el LDR está conectado al pin analógico.



Y como va al revés, vamos a poner el medidor al revés: de 1023 a 0



[Video link](#)

## Ver la temperatura y la humedad

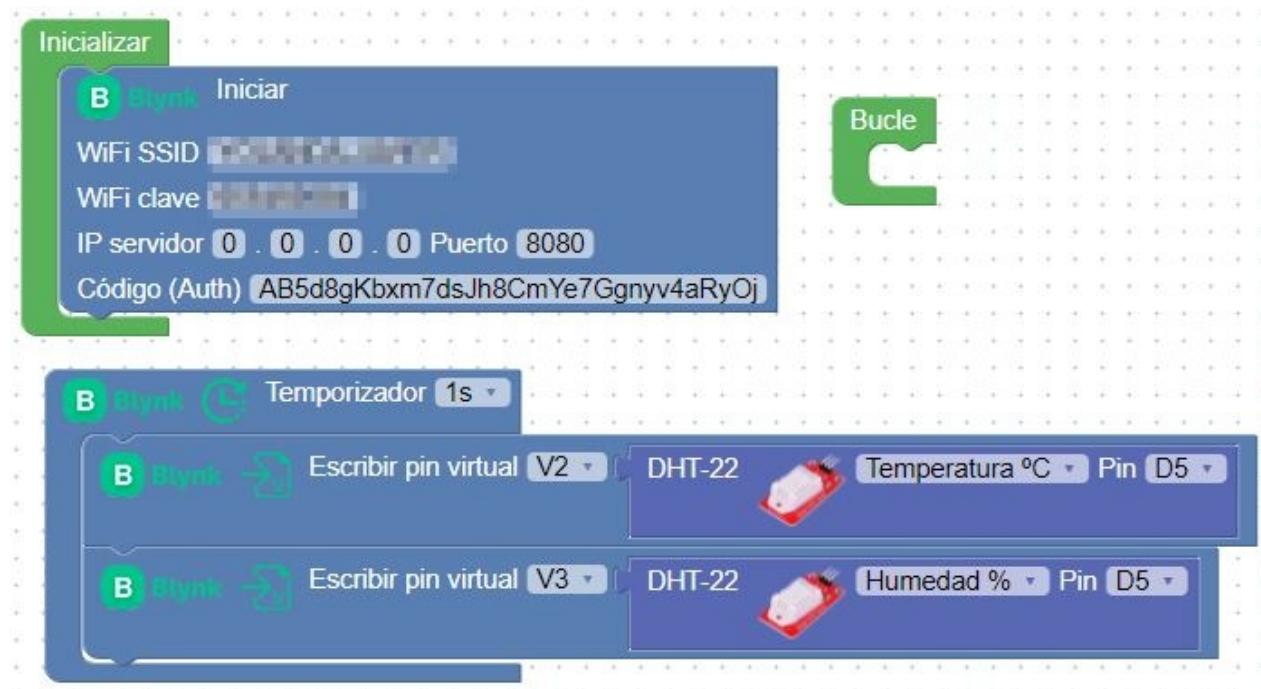
Este caso es distinto, pues el DHT22 está conectado al pin D5 digital, y mide dos variables **Temperatura Y Humedad**

¿Cómo hacemos para medir dos variables que están conectados a un sólo PIN?

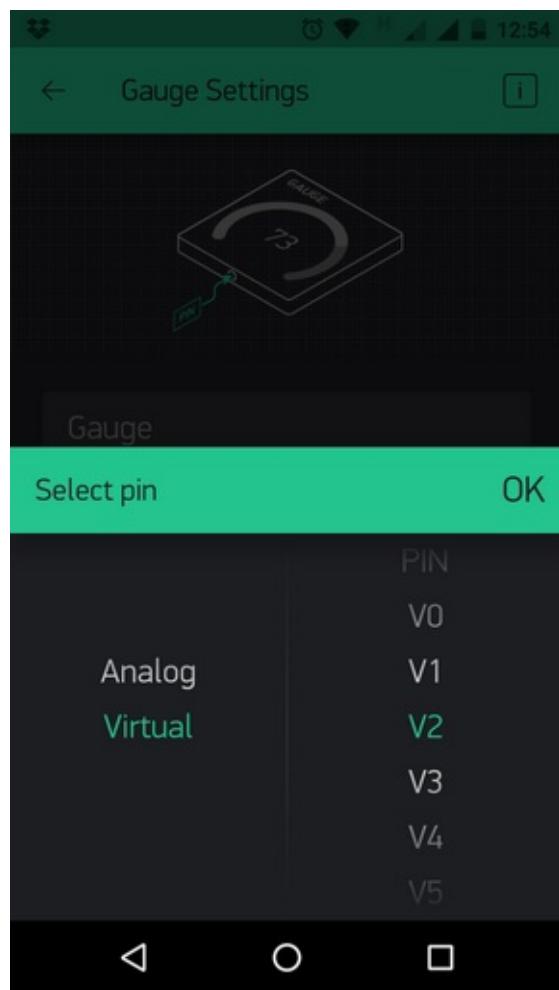
### Solución PINES VIRTUALES:

Utilizando **pines virtuales**.

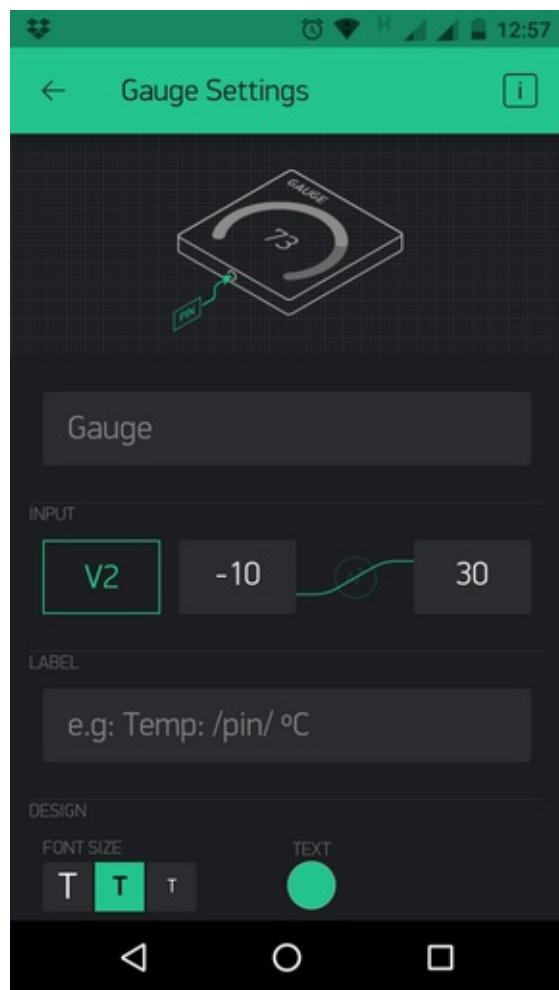
Vamos a ARDUINOBLOCKS y establecemos dos pines virtuales, uno para la temperatura y otro para la humedad, arbitrariamente he puesto V2 y V3 pero puede ser cualquiera.



Y en Blynk incorporamos un Gauge que sea al PIN VIRTUAL V2.



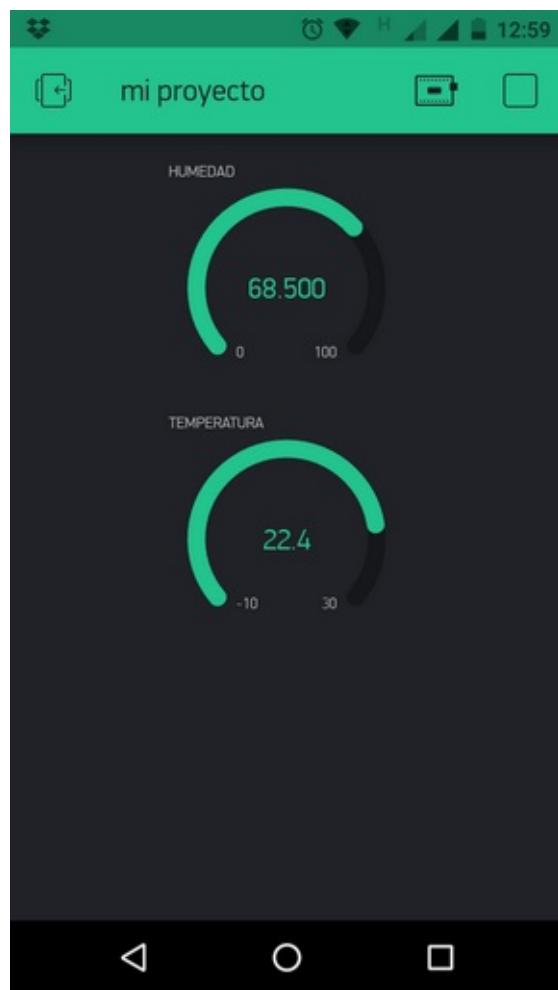
Modificamos también los límites, pues por defecto sale 0 a 1023 y se vería muy bajo la temperatura. Suponemos que en marte hace una temperataura desde -10°C a 30°C (realmente llega a -50°C)



Para la humedad hacemos lo mismo:

- Pin virtual V3
- Límites 0 a 100

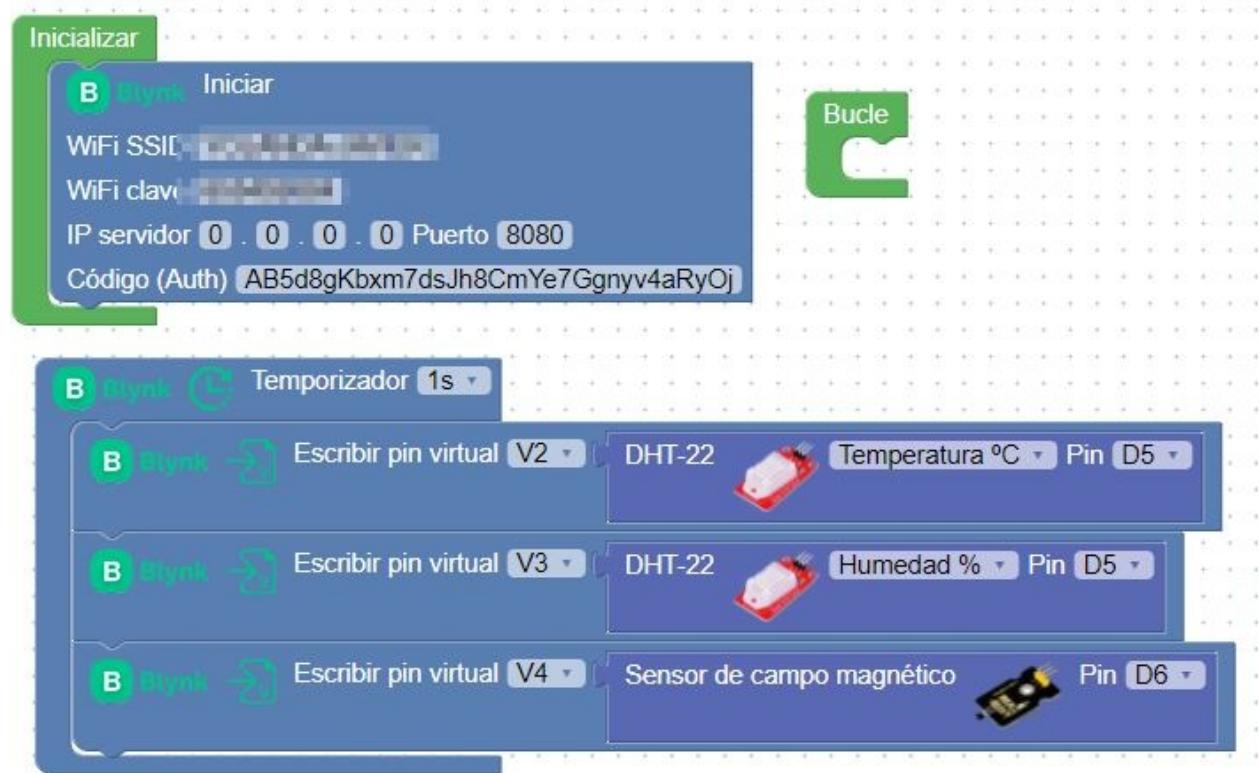
Resultado :



Por cierto para hacer esto, he tenido que borrar el Gauge anterior ¡¡me he quedado sin energía!!.

## Medir el campo magnético

El programa en ARDUINOBLOCKS es:

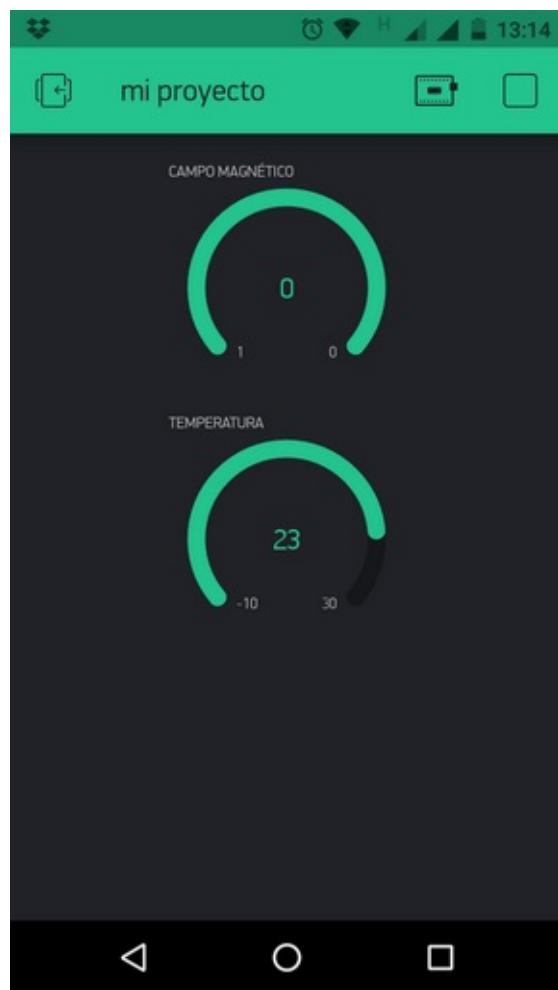


Y el programa en Blynk



te en cuenta que va al revés, cuando hay un imán, se pone a 0

El resultado es (acercando un imán):

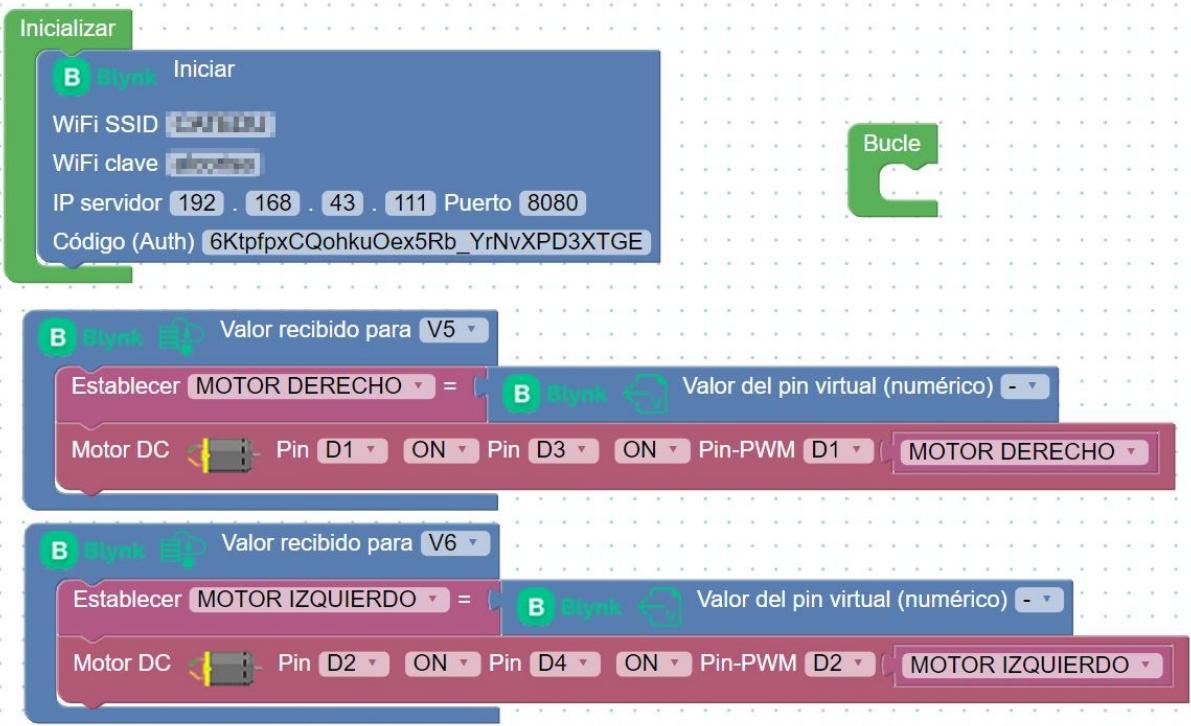


## Reto:

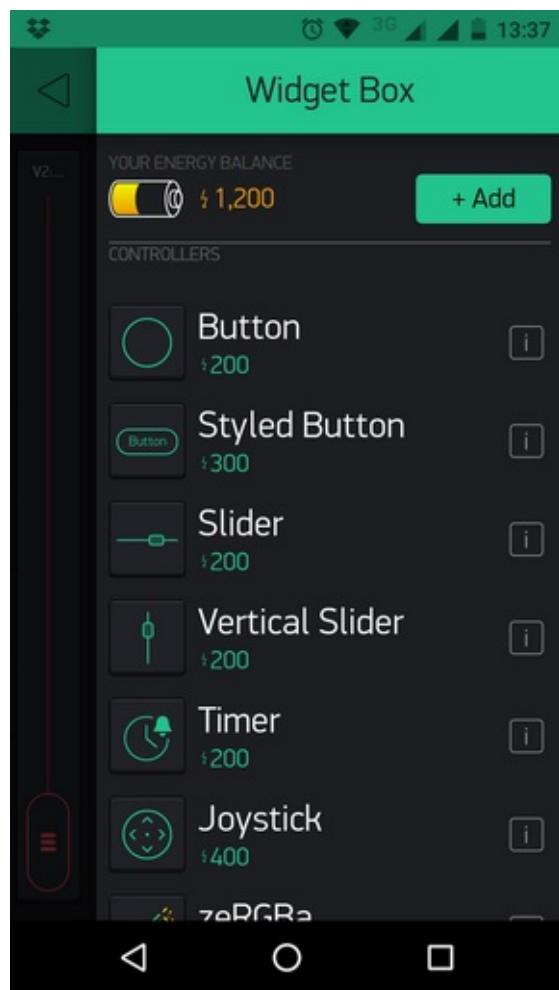
- Escribir un programa que encienda el pin D4 cuando se acerque un imán

## Motores

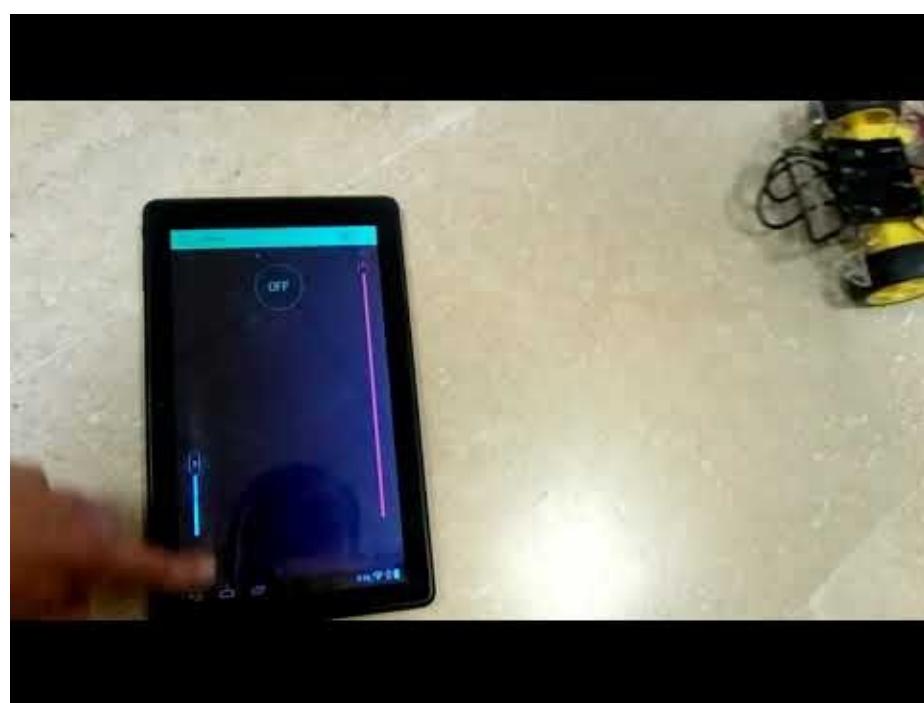
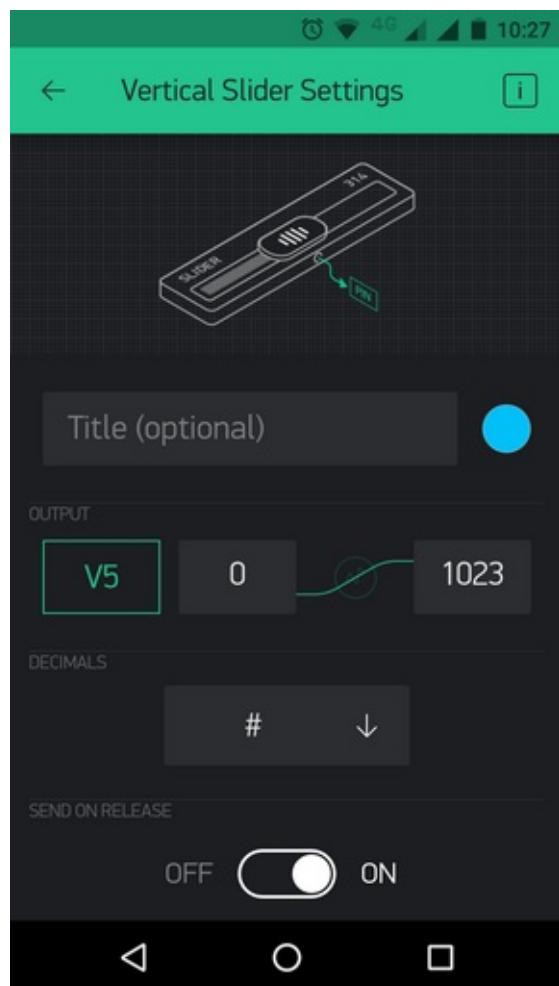
Cambiamos nuestro programa de ARDUINOBLOCKS utilizando unos pines virtuales para controlar los motores, arbitrariamente le hemos puesto V1 y V2 de tal manera que cuando se reciba un valor de estos pines virtuales, fijan la potencia de los motores:



Ponemos dos sliders verticales



Y configuramos uno para V5 y otro para V6:



[Video link](#)

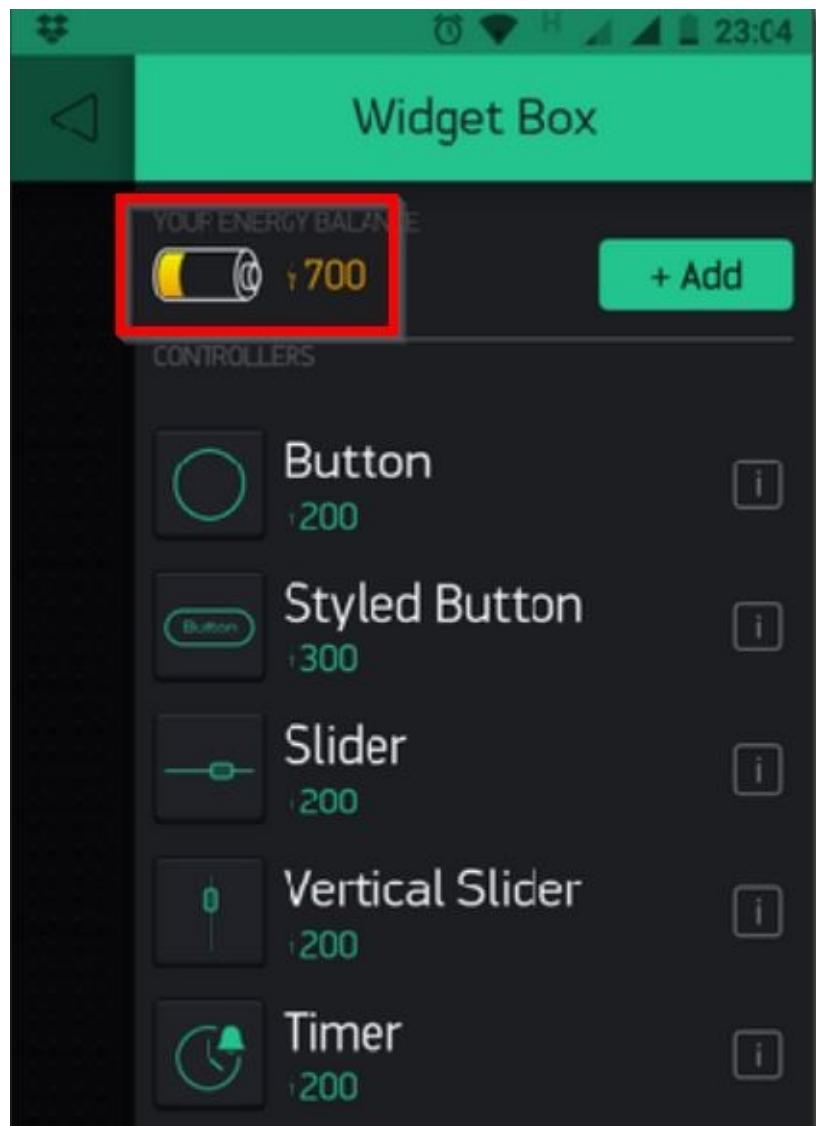
## ¿De qué se trata?

Nuestro objetivo es conseguir un **SERVIDOR BLYNK LOCAL** que haga de puente entre nuestro rover y nuestra App.

Se puede hacer en un ordenador normal, nosotros lo vamos a mostrar cómo se hace en una **Raspberry**.

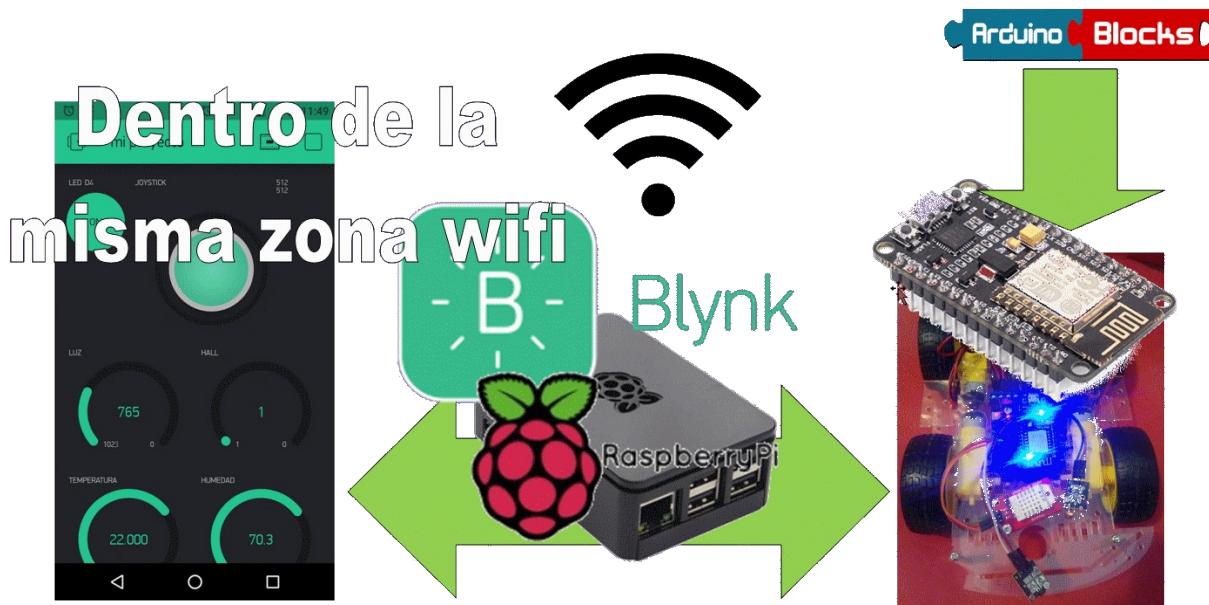
## Ventajas

Estamos liberados de la limitación del número de controles que podemos poner en nuestra App, lo que se llama *Energía* en Blynk



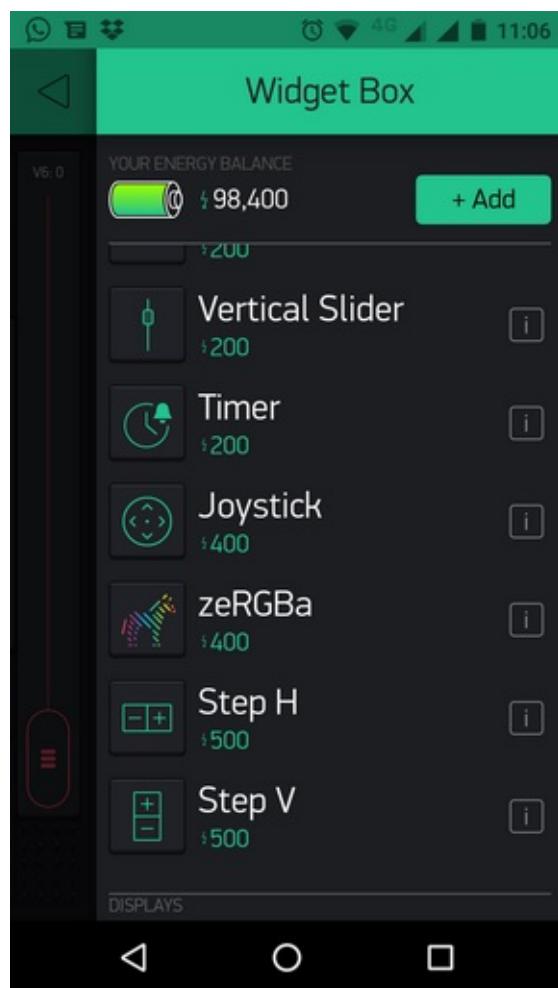
## Desventajas

Nuestro rover, la Raspberry y nuestro móvil **tiene que estar en la misma red local**



## Compensa

Es un esfuerzo configurar la Raspberry pero no es una cosa que tenemos que explicar a los alumnos, y ellos ya tienen TODA la energía para hacer sus proyectos



Además de tener control vía web de todos usuarios, pues tú eres el administrador:



## Primer paso: La Raspberry

Lo primero que tenemos que conseguir es:

- Tener una Raspberry con el sistema operativo Rasbian instalado
- Comunicarnos con la Raspberry por comandos SSH



Para instalar un **sistema operativo Raspbian** en una Raspberry aconsejamos ver estos apuntes:

<https://catedu.github.io/raspberry-muy-basico/3-raspbian.html>

Una vez instalado hay que **conectarlo a la red wifi** y **ponerle una IP FIJA**, para ello seguir estos apuntes:

<https://catedu.github.io/raspberry-muy-basico/4-primera-comunicacion.html>

Para **comunicarse vía comandos SSH**, aconsejamos hacerlo de forma remota, pues lo contrario implica tener siempre una pantalla, teclado y ratón conectado en la Raspberry, por lo tanto seguimos estos pasos :

<https://catedu.github.io/raspberry-muy-basico/5-ssh.html>

## Segundo paso: Instalar Java en la raspberry

Por el momento, para instalar Blynk, se necesita la **versión 8 de Java**

Primero actualicemos nuestro raspberry

Entramos en la ventana de comandos SSH y ejecutamos las siguientes órdenes, si en algún momento nos pide confirmación [Y/n] es porque faltan descargar paquetes o confirmación de instalación, por lo tanto aceptamos.

Con esta orden busca las actualizaciones

**sudo apt-get update**

y con esta las instalar

**sudo apt-get dist-upgrade**

Para instalar la versión 8 de java ejecutamos

**sudo apt install openjdk-8-jdk**

Por último una vez finalizado, comprobamos la versión que coje por defecto

**java -version**

Tiene que salir Openjdk version 1.8.0 etc...

## Si no sale esa versión, sale una más actualicemos

Suele pasar que tenga ya instalado la versión 11, tenemos que obligar a que sea la 8 por defecto, para ello ejecutamos

**sudo update-alternatives --config java**

```
pi@raspberrypi:~ $ sudo update-alternatives --config java
Existen 2 opciones para la alternativa java (que provee /usr/bin/java).
      Selección     Ruta                               Prioridad  Estado
-----+
 0  automático   /usr/lib/jvm/java-11-openjdk-armhf/bin/java    1111      modo a
 1  anual        /usr/lib/jvm/java-11-openjdk-armhf/bin/java    1111      modo m
* 2  anual        /usr/lib/jvm/java-8-openjdk-armhf/jre/bin/java  1081      modo m

Pulse <Intro> para mantener el valor por omisión [*] o pulse un número de selección: 2
```

Si seleccionamos el 2 nos lo pondrá como por defecto (sale con un (\*). Podemos comprobar que ya nos sale la versión con la orden :

**java -version**

Pero esto no nos sirve, pues al reiniciar la Raspberry volverá a la versión 11.

Para ponerlo por defecto, edita el fichero *environment*

**sudo nano /etc/environment**

Y si en la orden **sudo update-alternatives --config java** salía que la carpeta donde está la versión 8 es */usr/lib/jvm/java-8-openjdk-armhf/jre/bin/java* luego tenemos que poner en el fichero *environment* :

**JAVA\_HOME="/usr/lib/jvm/java-8-openjdk-armhf/jre/bin/java"**

Guardar, reiniciar

**sudo reboot**

Y comprueba

**java -version**

Y tiene que salir :

```
pi@raspberrypi:~/Blynk $ java -version
openjdk version "1.8.0_212"
OpenJDK Runtime Environment (build 1.8.0_212-8u212-b01-1+rpi1-b01)
OpenJDK Client VM (build 25.212-b01, mixed mode)
```

## Para más cosas

Por ejemplo, desistalarlo, volver a la 11, etc.. visitar esta página :

<https://phoenixnap.com/kb/install-java-raspberry-pi>

## Tercer paso: Instalar Blynk

### A.- Descarga Blynk

Crea una carpeta en tu directorio home, por ejemplo Blynk

**mkdir Blynk**

Y descarga Blynk :

```
wget "https://github.com/blynkkk/blynk-server/releases/download/v0.41.13/server-0.41.13-java8.jar"
```

### B.-Configurar server.properties

Necesitamos crear un fichero de configuración para las diferentes opciones que queremos en nuestro servidor Blynk.

Entra en la carpeta creada Blynk y crea el fichero *server.properties*.

**cd Blynk**

**sudo nano server.properties**

A continuación se muestra una muestra del posible contenido de *server.properties*.

```
#Definir los puertos para que no se solapen
#http and web sockets port
http.port=8080
#https and web sockets port,
https.port=9443
#application ssl port
app.ssl.port=8443
# para permitir administrar la página de administrador fuera, (si no lo haces tiene que ser en https://127.0.0.1:9443/admin) luego
allowed.administrator.ips=0.0.0.0/0
# es decir si la ip de la raspberry es 192.168.1.112 luego puedo administrarlo fuera en https://192.168.1.112:9443/admin
admin.rootPath=/admin
# Default admin name and password. Will be created on initial server start
admin.email=admin@blynk.cc
admin.pass=admin
```

No es la única configuración, se puede mejorar : [aquí por ejemplo tienes otra propuesta de \*server.properties\*](#)

### C.- Configurar mail.properties

Igualmente hay que crear este fichero para que el servidor envíe por email los tokens de los proyectos

Entra en la carpeta creada Blynk y crea el fichero *mail.properties*.

**cd Blynk**

**sudo nano mail.properties**

A continuación se muestra una muestra del posible contenido de *mail.properties* utilizando los datos de una cuenta de gmail tuya.

```
mail.smtp.auth=true
mail.smtp.starttls.enable=true
mail.smtp.host=smtp.gmail.com
mail.smtp.port=587
mail.smtp.username=Your EMAIL ID
```

```
mail.smtp.password=Password
```

## D.- Ejecutar el servidor Blynk local

En la Raspberry por comandos SSH, entramos en la carpeta donde hemos creado el servidor Blynk

**cd Blynk**

Y ejecutamos el servidor Blynk instalado, pero que cargue la configuración de server.properties:

```
java -jar server-0.41.13-java8.jar -dataFolder /home/pi/Blynk -serverConfig /home/pi/Blynk/server.properties
```

Para no repetir estos dos comandos cd Blynk y java -jar server-0.41.13-java8.jar -dataFolder /home/pi/Blynk -serverConfig /home/pi/Blynk/server.properties cada vez que reiniciamos la Raspberry puedes generar un script para que lo ejecute automáticamente, puedes ver buenos tutoriales en Internet, por ejemplo [aquí](#)

## E.-Para saber más :

- [Intalación de Blynk : How to Install a Blynk Local Server on Raspberry Pi](#)
- [Configuración de server.properties.](#)
- [Configuración mail](#)

## Cuarto paso: Panel de control del servidor BLYNK instalado localmente

Desde cualquier ordenador conectado en red, ejecutas en un navegador la siguiente dirección:

Si la IP de la Raspberry es 192.168.1.112 entonces entramos en:

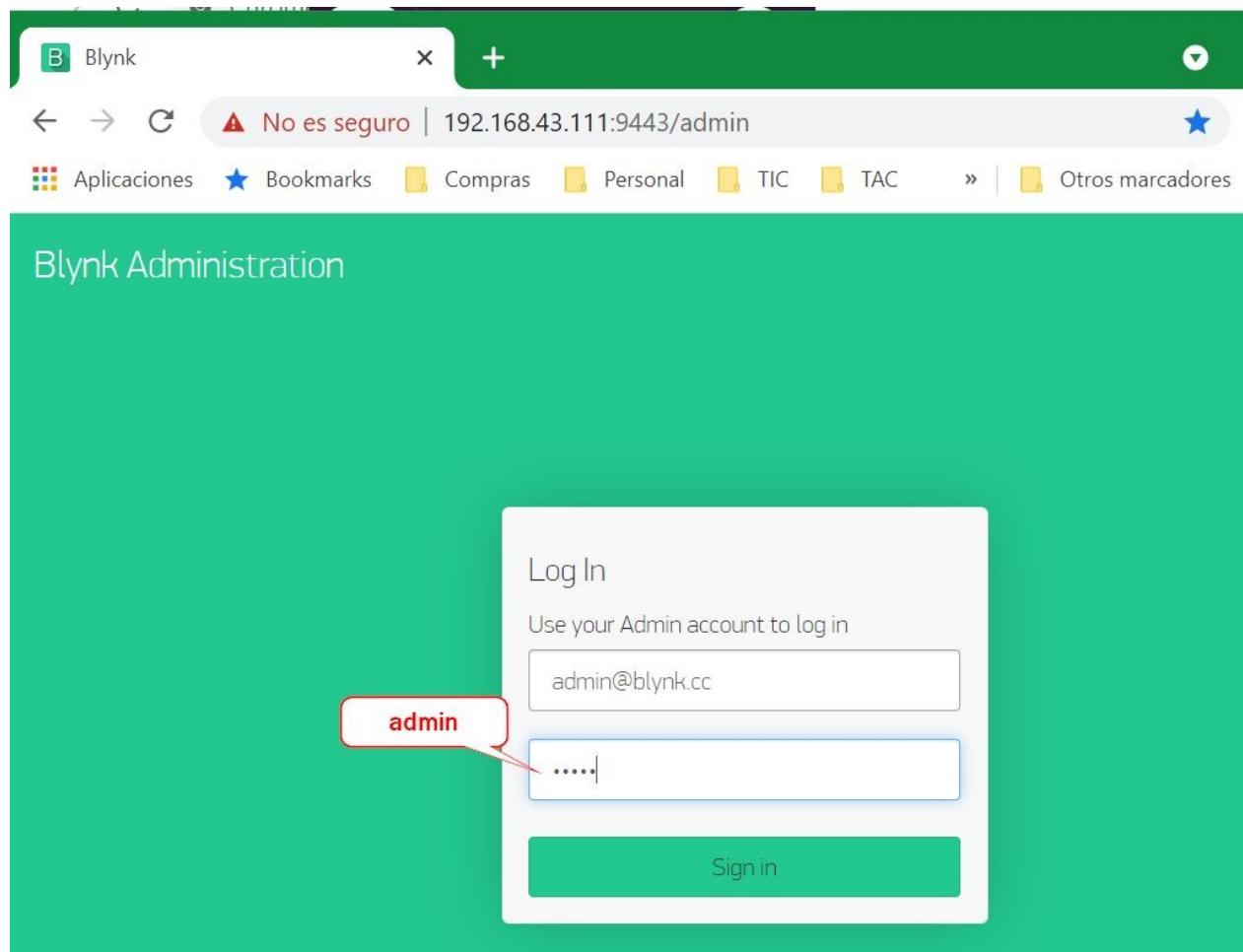
<https://192.168.1.112:9443/admin>

Con el usuario y contraseña fijada en server.properties :

```
# Default admin name and password. Will be created on initial server start
admin.email=admin@blynk.cc
admin.pass=admin
```

*Nota: Lo he intentado cambiar y no me ha funcionado ☹*

Luego entramos



Y podemos ver los usuarios creados (los usuarios se crean desde la APP, luego lo vemos)

Blynk Administration

- [Users](#)
- [Stats](#)
- [Hardware Info](#)
- [Config](#)

## Users list

Export

Email	AppName	# Of Projects	LastModifiedTs
jj.fernandez@elcorteingles.es	Blynk	1	2021-04-29 09:51:33
javierquintanapeiro@gmail.com	Blynk	1	2021-04-24 21:22:42
admin@blynk.cc	Blynk	1	2021-04-24 19:27:39

Incluso podemos ver los **tokens**, entrando en un usuario, vemos los proyectos y podemos ver el tokens

+ Add new widgets

Devices
**Id**
**Remove**

Name

BoardTy

X ▾

Token
JGnbIYNVLfbdgqNvHe3xK1LyJV0QDiE9

LastLogg0dIP
68.1.131

Connect

X ▾

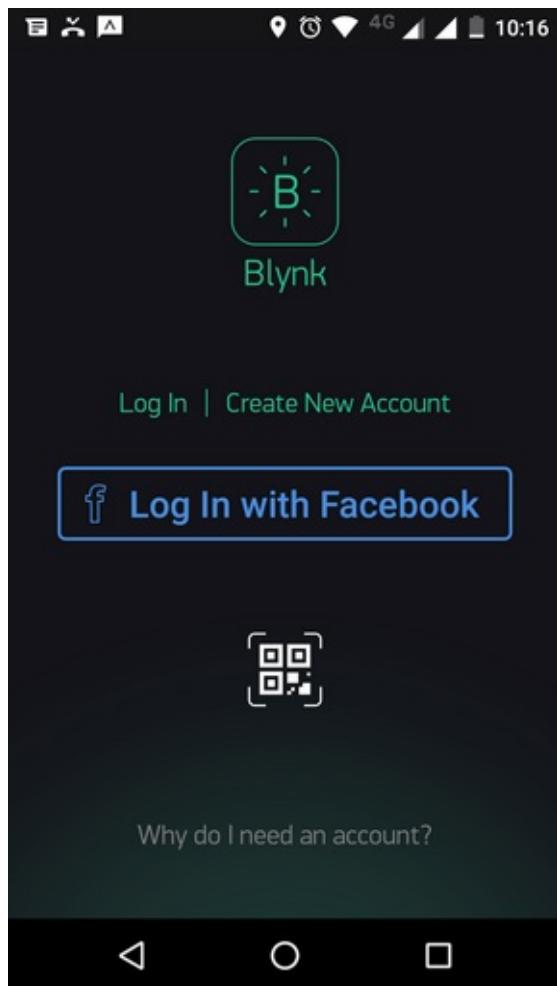
+ Add new devices

Útil si falla el envío del email de los tokens

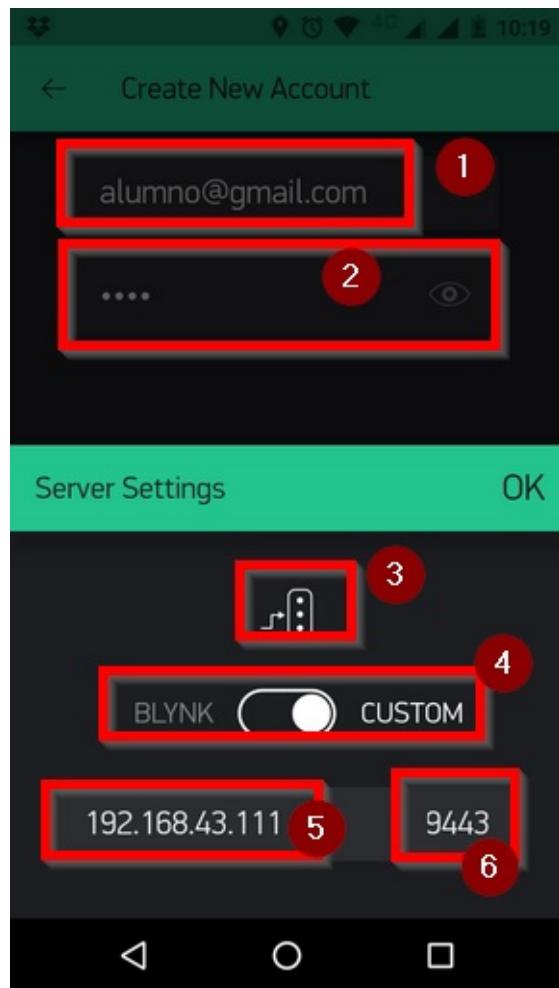
## En la APP de Blynk pero con el servidor local

### Conecrtar y crear nuevo usuario

Entramos en la APP de Blynk - Crear nuevo usuario



pero esta vez le damos al botón de abajo CUSTOM y ponemos LA IP DE LA RASPBERRY



Una de las ventaja de usar servidor local BLYNK es que el correo puede ser ficticio, BLYNK no podrá enviar el Token pero sí que podemos saber el Token en el panel de control, tal y como hemos visto en el punto 4.4. Esto es una solución para los alumnos menores de 16 años pues legalmente no pueden tener email.

## Autores

Agradecimientos a :

CONSEJERÍA DE EDUCACIÓN Y FORMACIÓN PROFESIONAL DEL GOBIERNO DE CANTABRIA



Coordinación y montaje:

Cualquier observación o detección de error por favor aquí [soporte.catedu.es](mailto:soporte.catedu.es)

Los contenidos se distribuye bajo licencia Creative Commons tipo BY-NC-SA.



# GOBIERNO DE ARAGÓN

Departamento de Educación,  
Cultura y Deporte

**CATEDU**   
CENTRO ARAGONÉS de TECNOLOGÍAS para la EDUCACIÓN

