

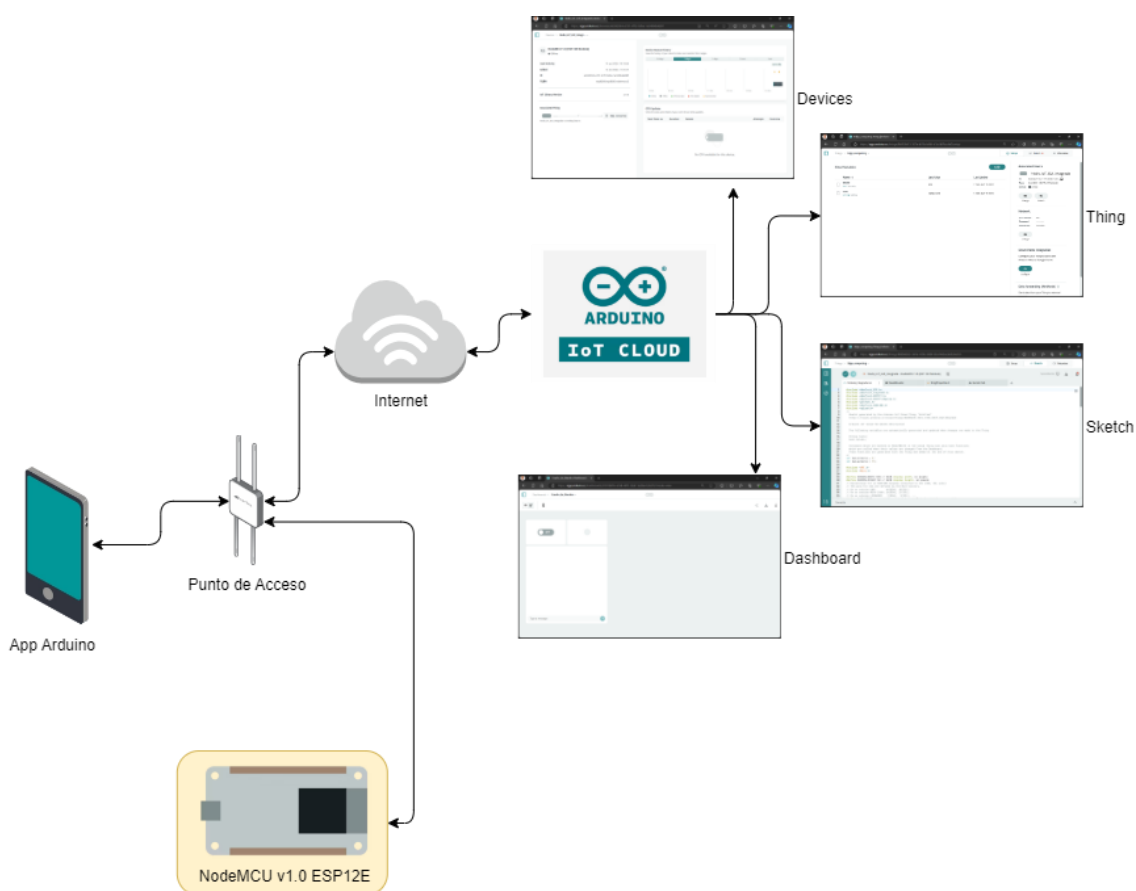
Práctica de laboratorio

IoT

1. Objetivos de la práctica

Este documento es un breve tutorial para la realización de un proyecto IoT basado en la plataforma NodeMCU v1.0 v3 (basado en el SoC ESP8266 y el convertidor USB a UART CH340) y una nube de procesamiento de la información (Arduino Cloud).

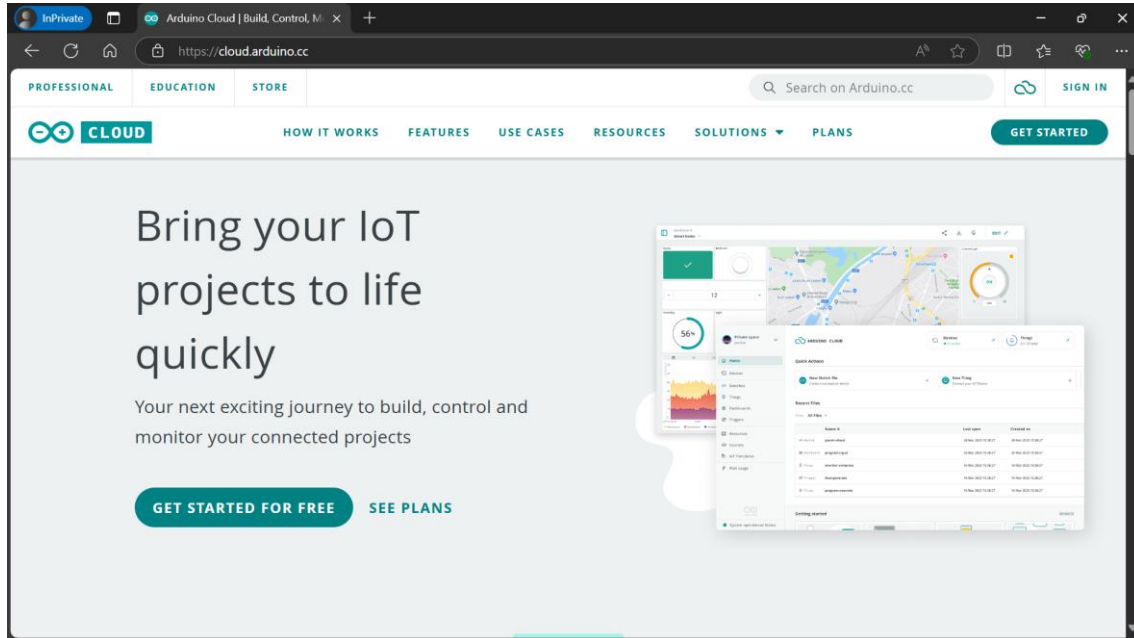
2. Arquitectura del sistema IoT



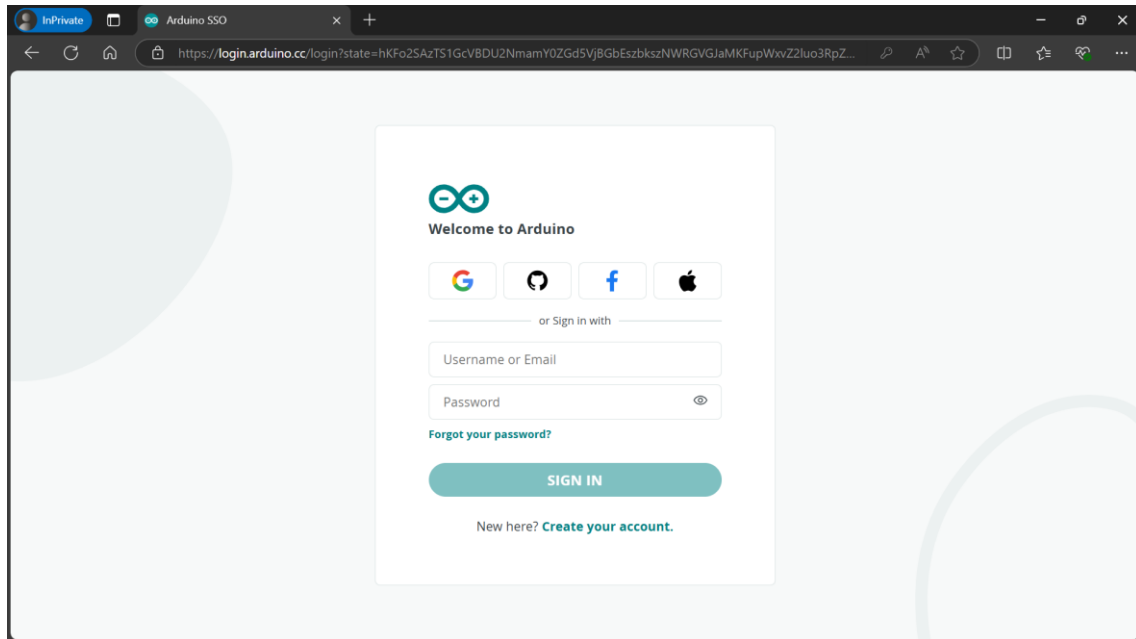
3. Desarrollo de la práctica

Se recogen a continuación una serie de actuaciones a realizar.

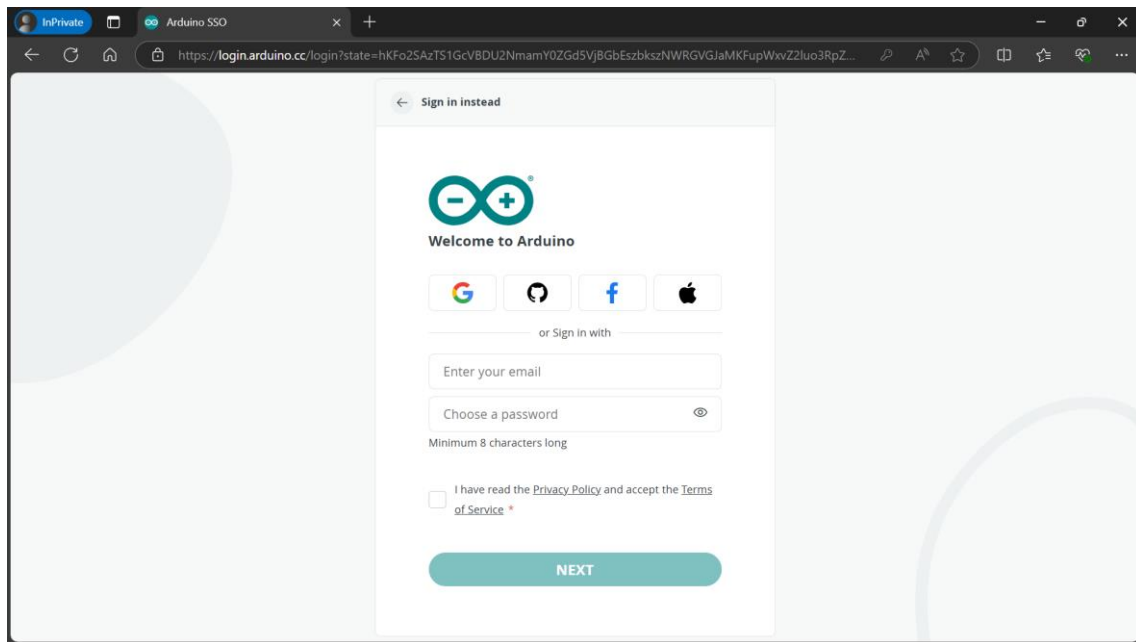
Paso 1. En un navegador web visitar el sitio: <https://cloud.arduino.cc/>



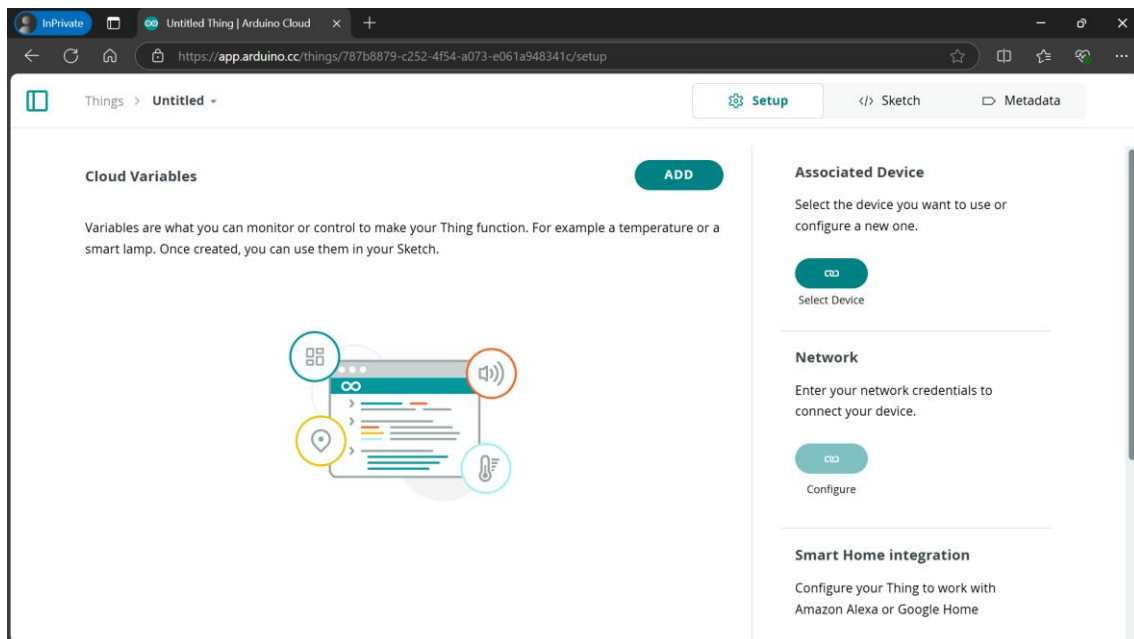
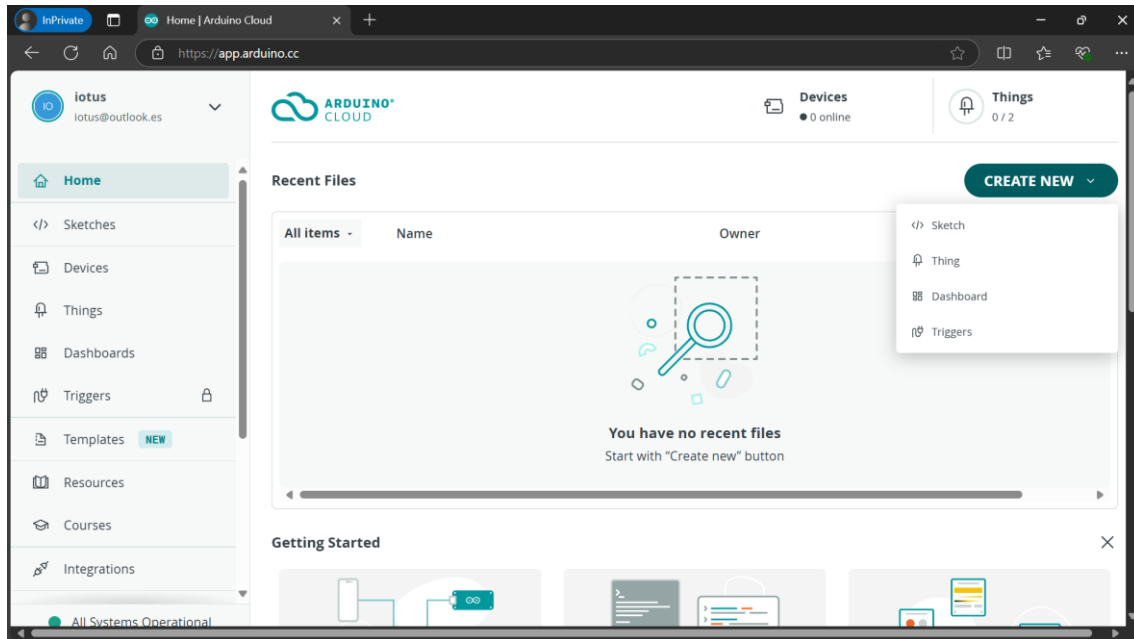
Paso 2. GET STARTED FOR FREE.



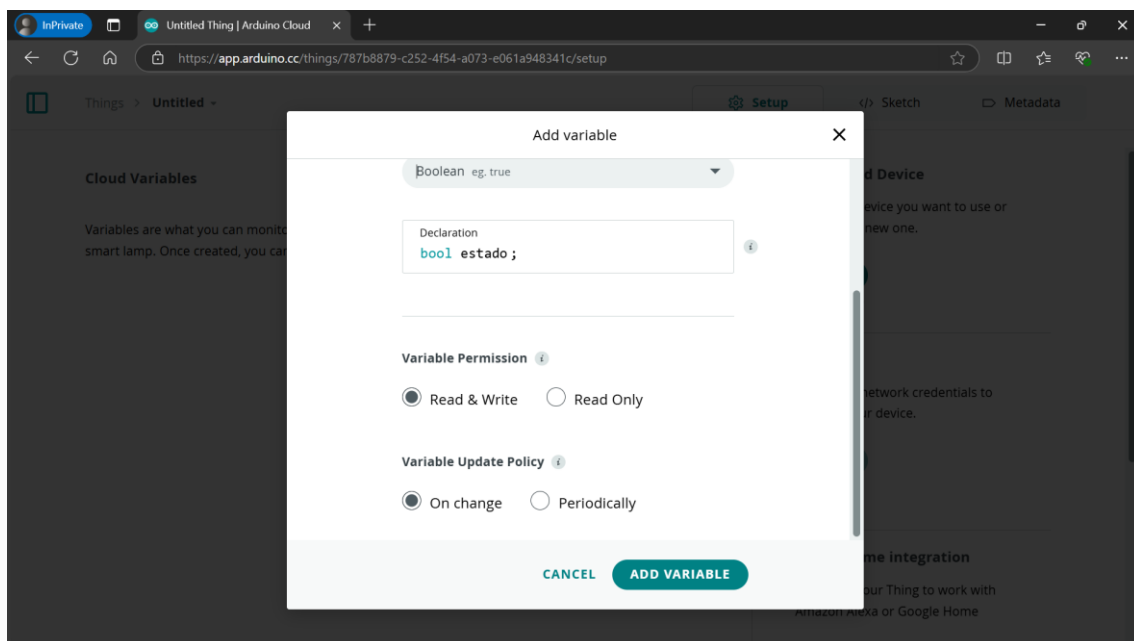
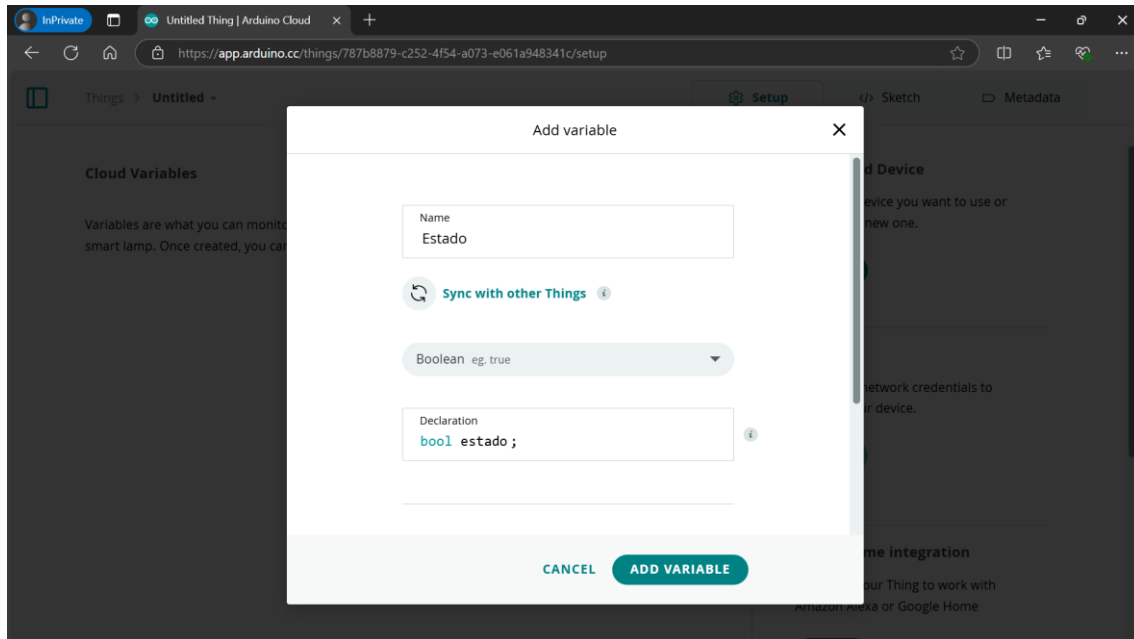
Paso 3. Crear una cuenta.

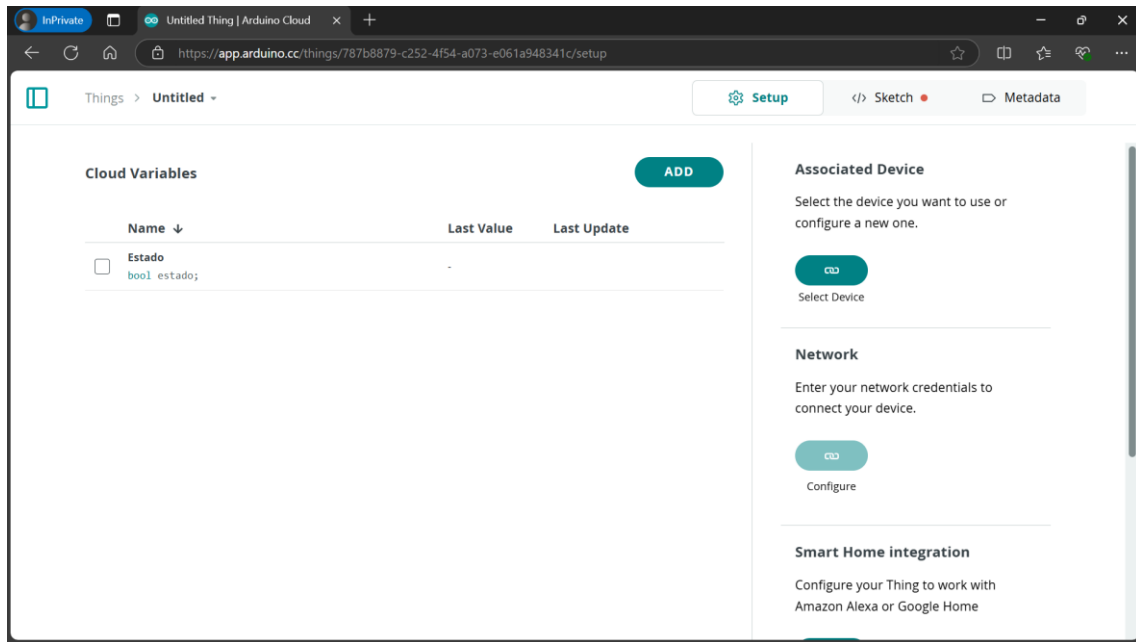


Paso 4. Crear nueva "Thing".

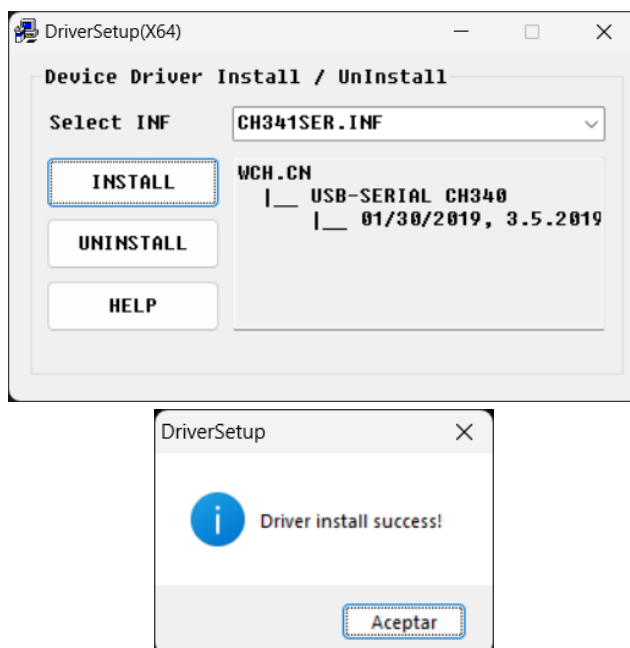


Paso 5. Crear nueva variable en la nube (Cloud Variable).

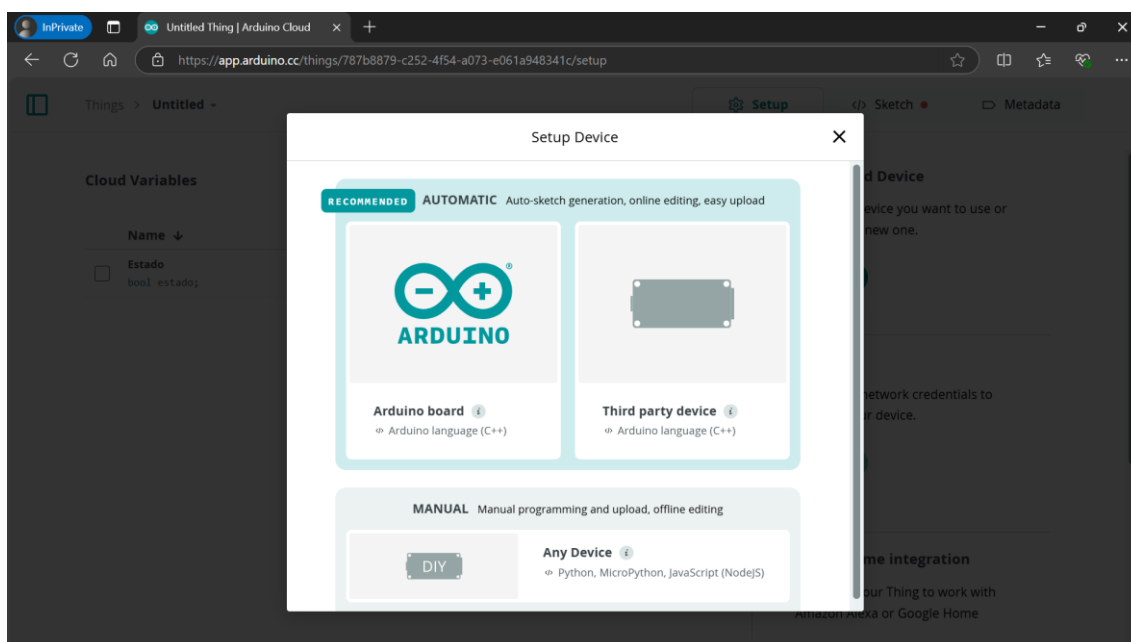




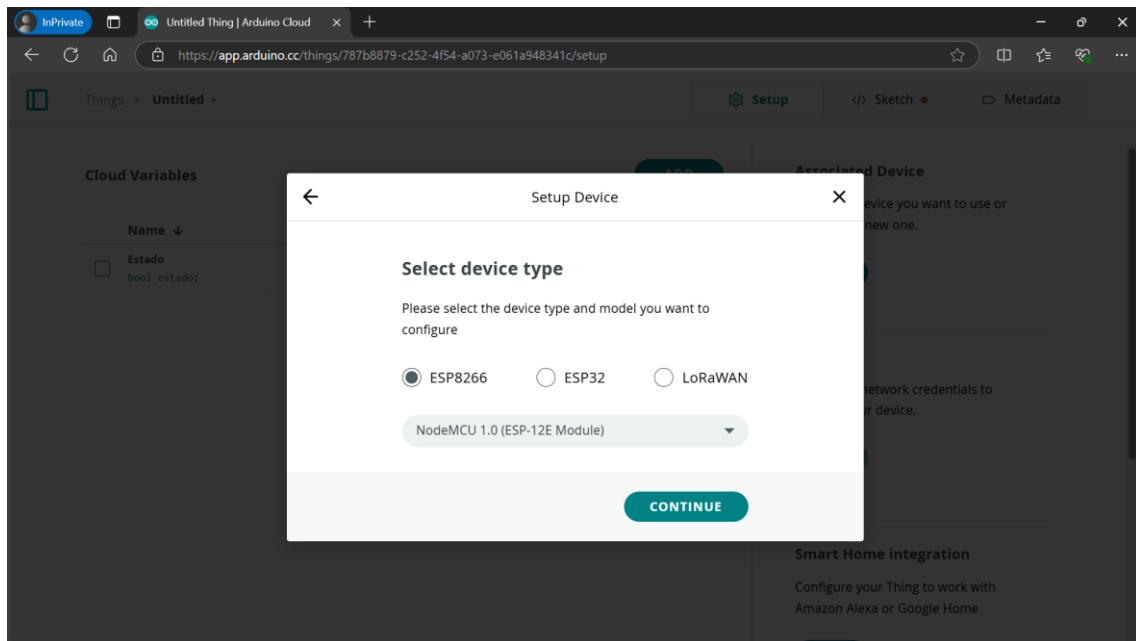
- Paso 6.** Conectar la plataforma NodeMCU v1.0 v3 al PC mediante cable USB.
- Paso 7.** Instalar los controladores del dispositivo CH340 mediante la ejecución del archivo CH341SER.EXE.



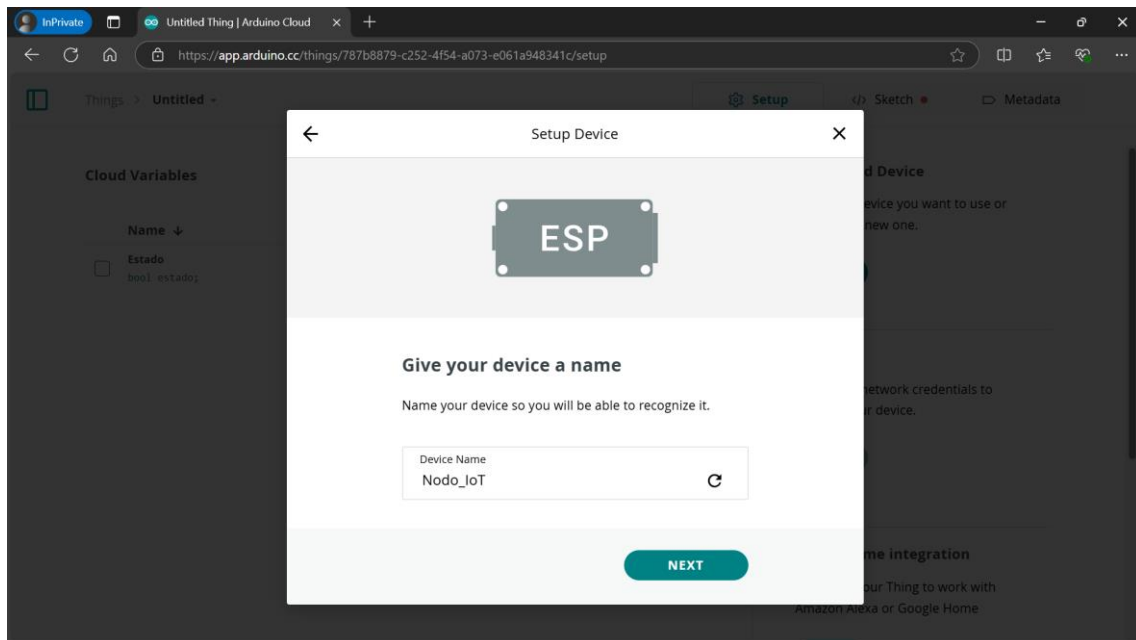
- Paso 8.** En la sección “Associated Device” hacer click sobre el botón “Select Device”.

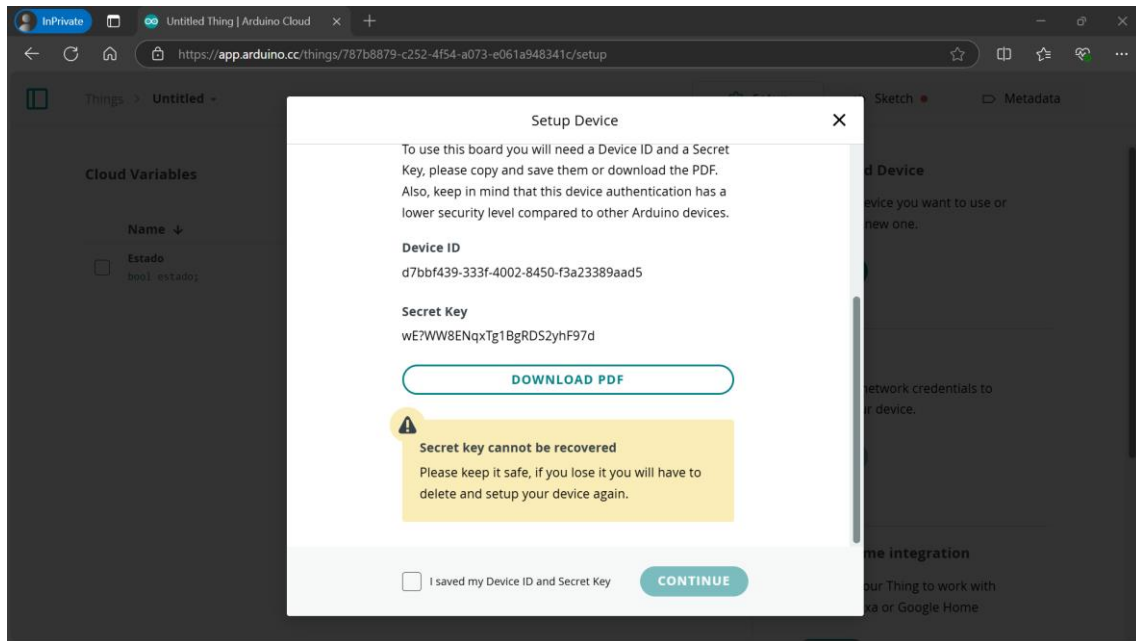
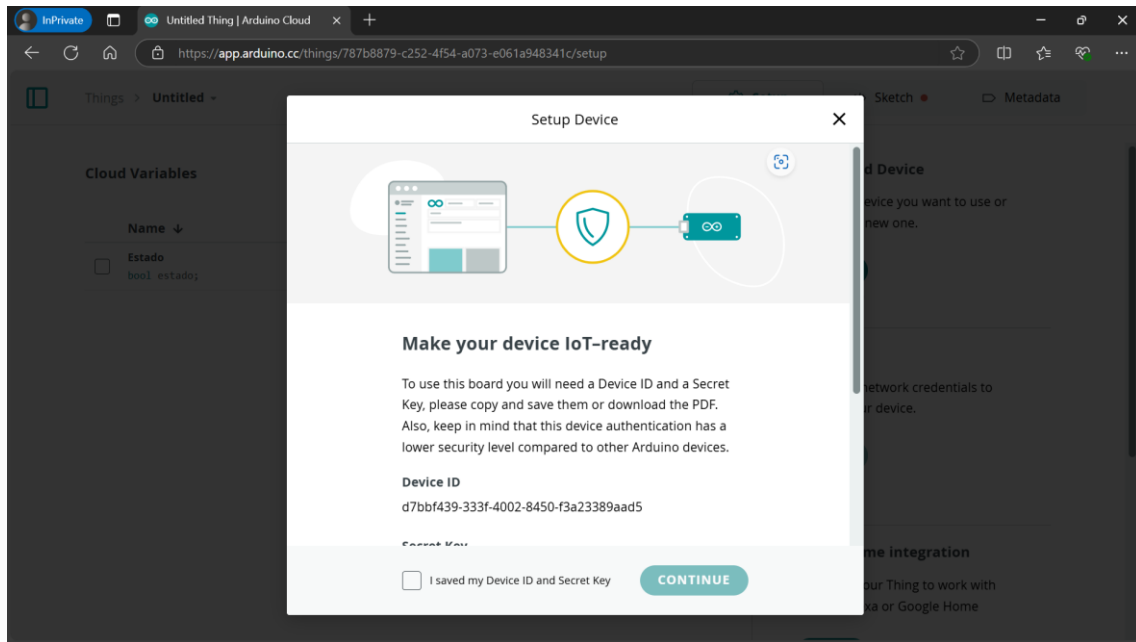


Paso 9. Elegir “Third party device” → “ESP8266” → “NodeMCU 1.0 (ESP-12E Module).”

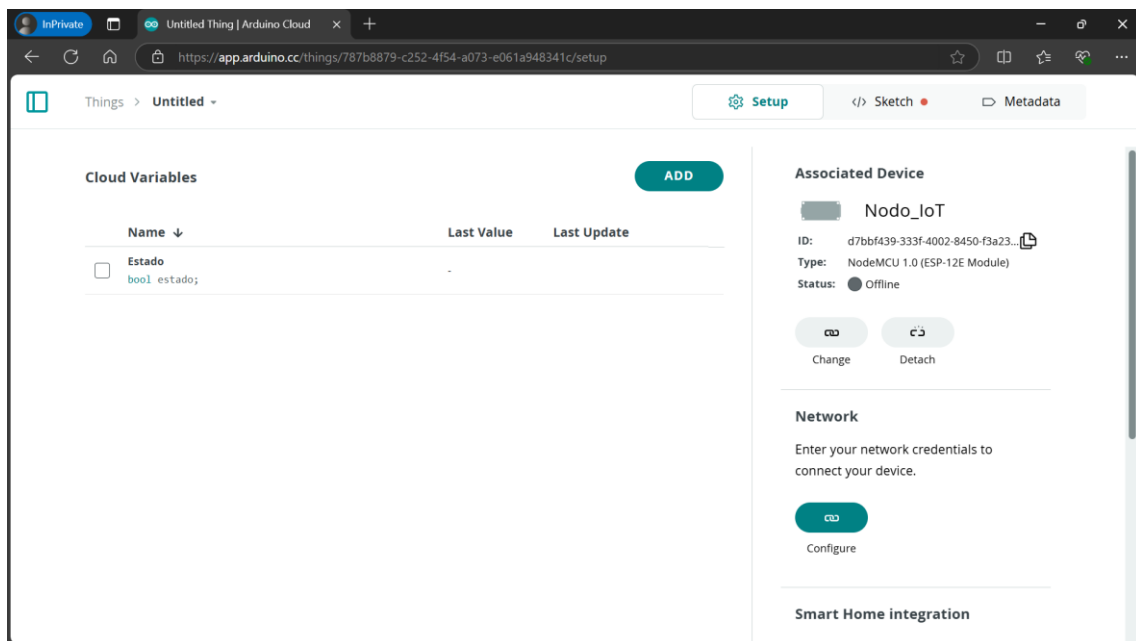
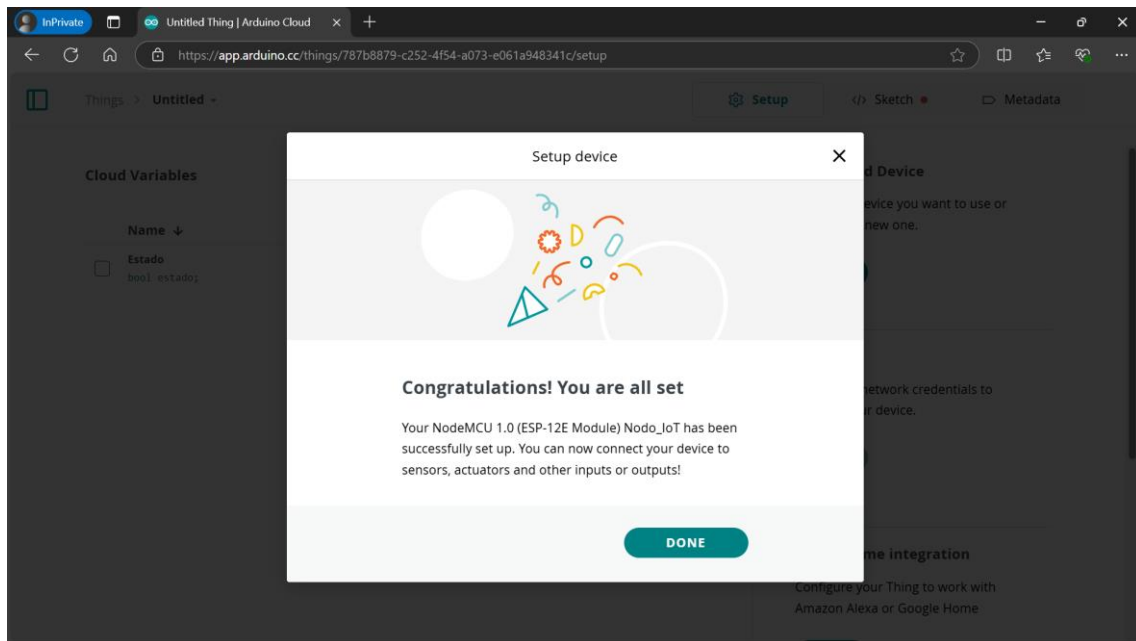


Paso 10. Indicar un nombre.

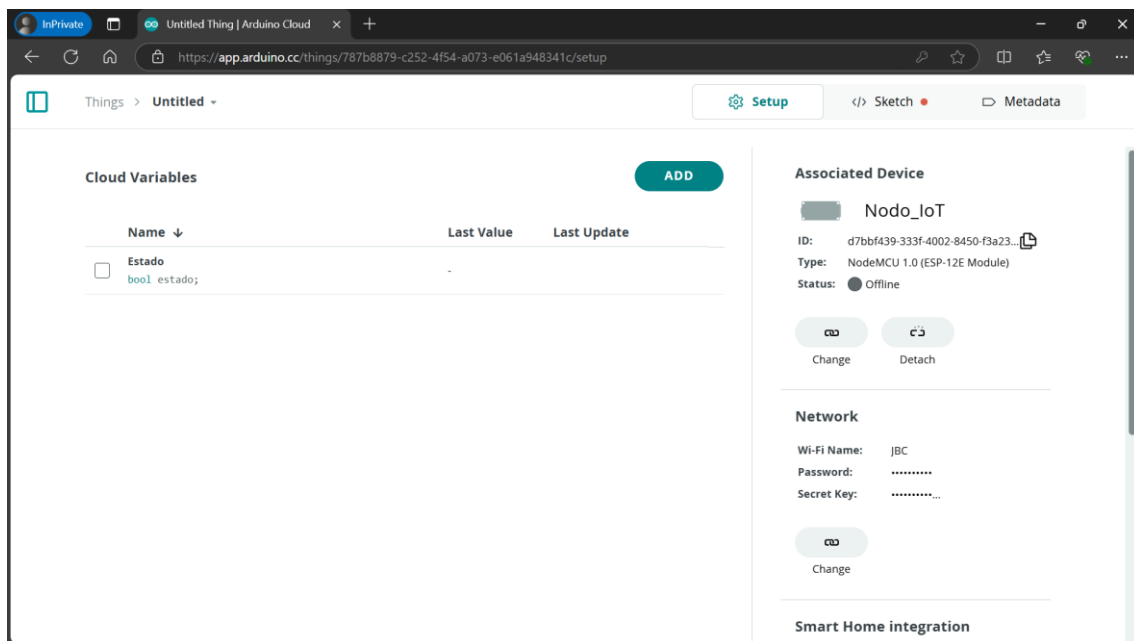
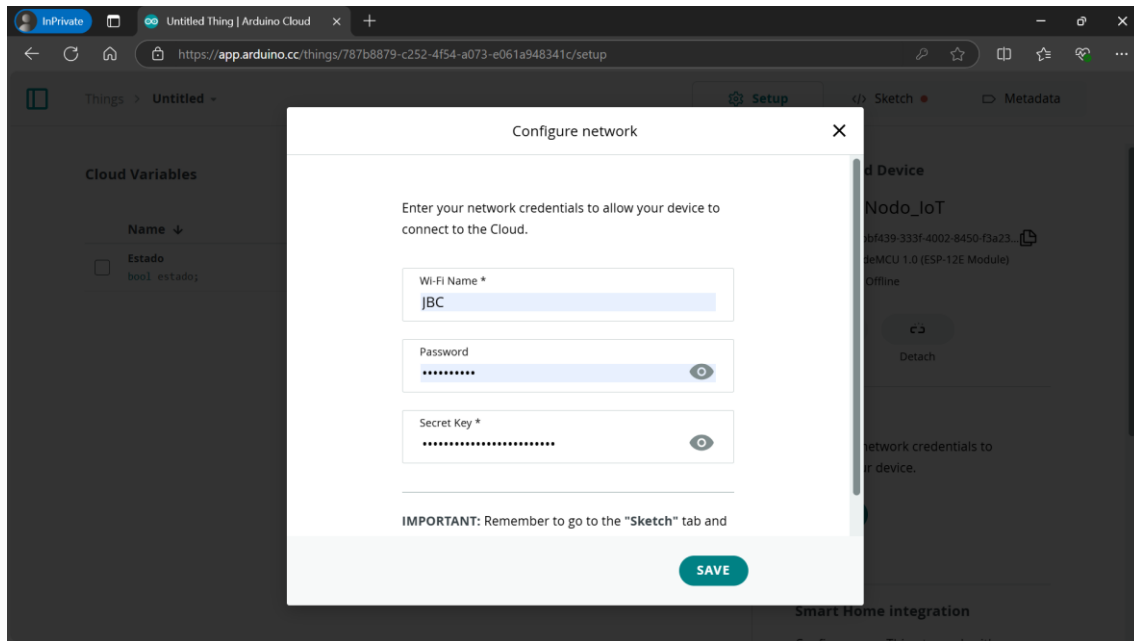




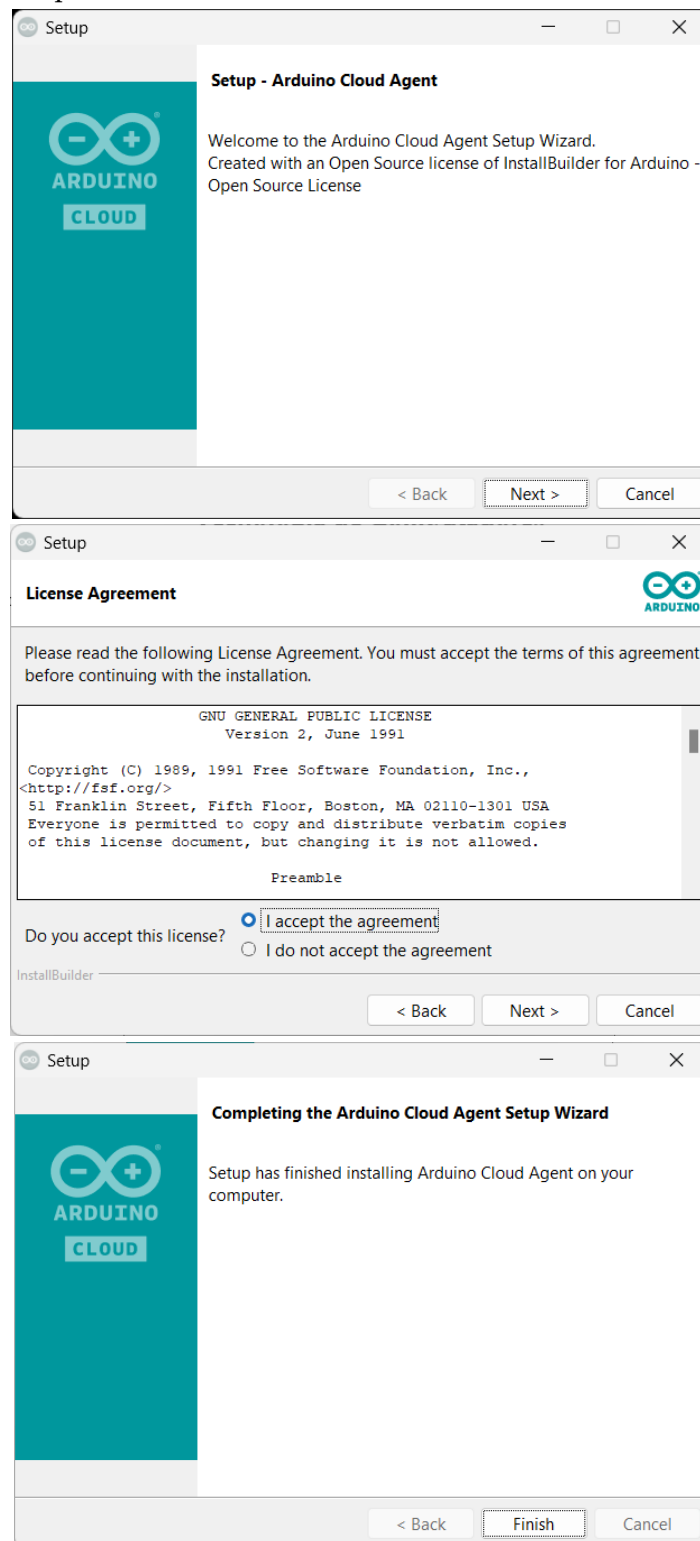
Paso 11. Descargar el PDF con los datos del dispositivo IoT asociado y marcar la casilla “I saved my Device ID and Secret Key”. Pulsar el botón “Continue”.



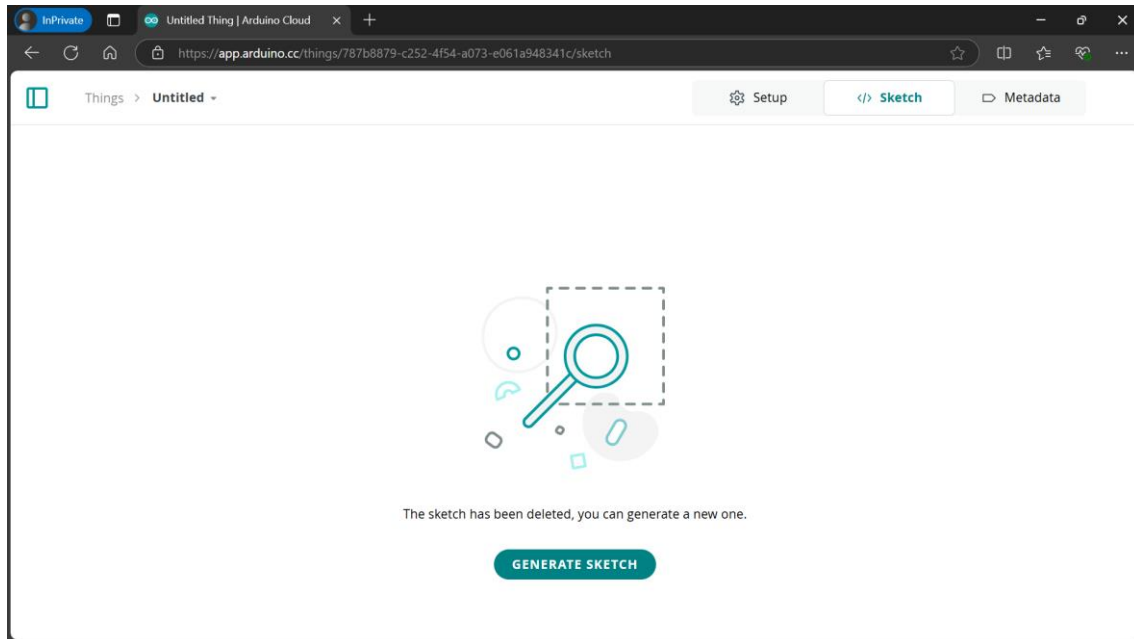
Paso 12. Configuración de la conexión del IoT a un punto de acceso: en la sección “Network”, hacer click en el botón “Configure”. En un terminal móvil, declarar un punto de acceso y completar la configuración con los datos del punto de acceso.



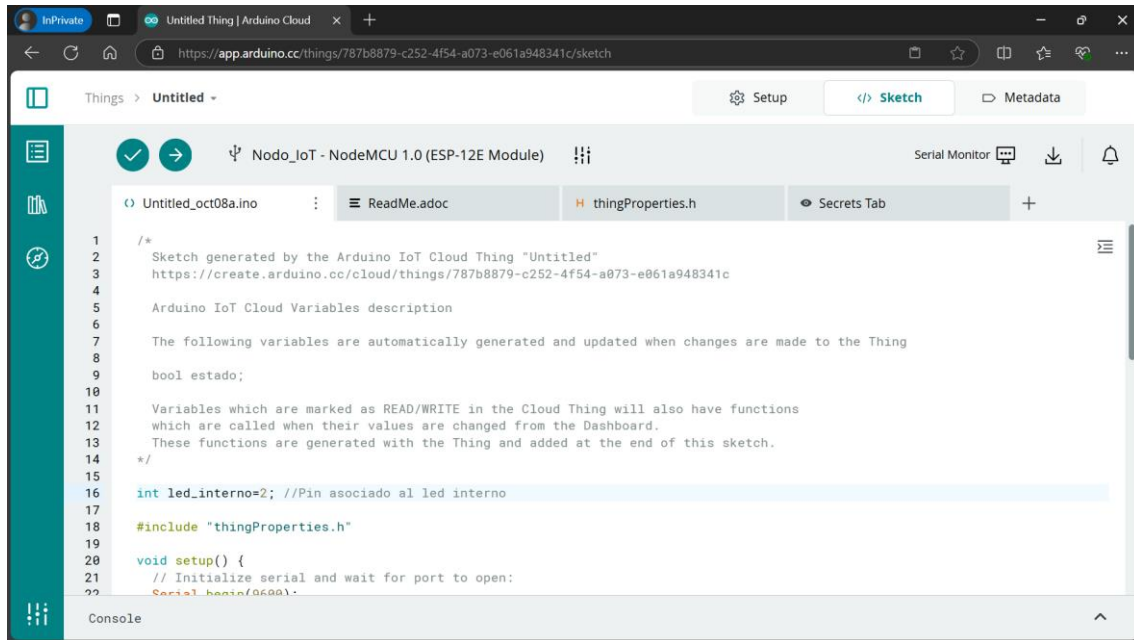
Paso 13. Instalar el software “Cloud Agent” (ArduinoCloudAgent-1.6.1-windows-386-installer.exe) encargado permitir la interacción entre la nube y el dispositivo IoT a través del PC. Este software permitirá volcar el programa en el dispositivo IoT.



Paso 14. En el menú superior, seleccionar “Sketches” → “GENERATE SKETCH”.



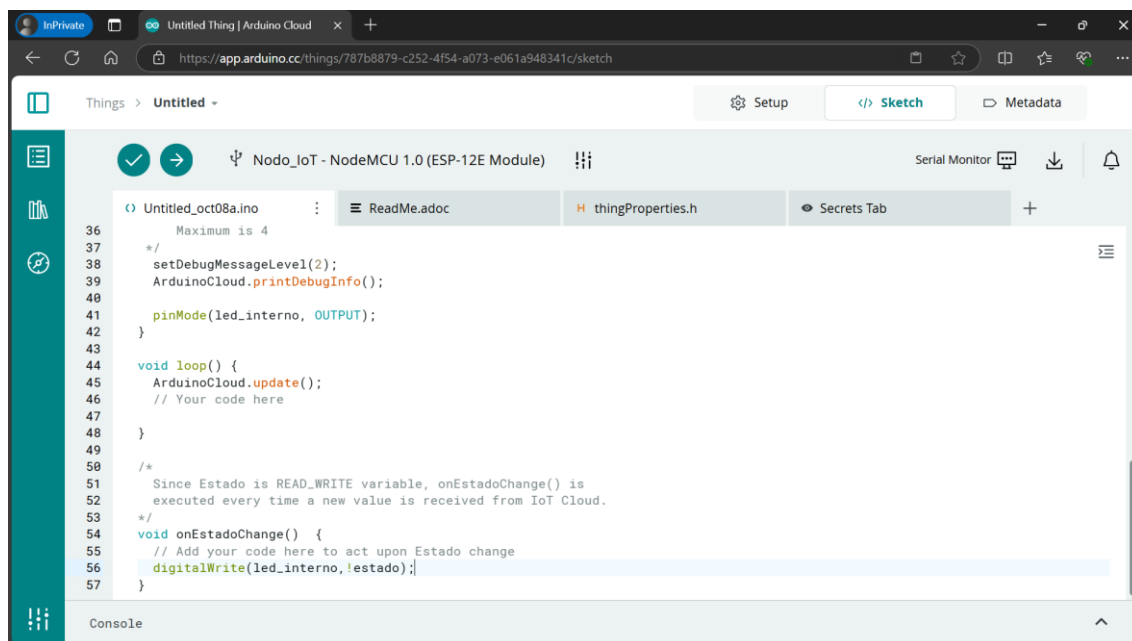
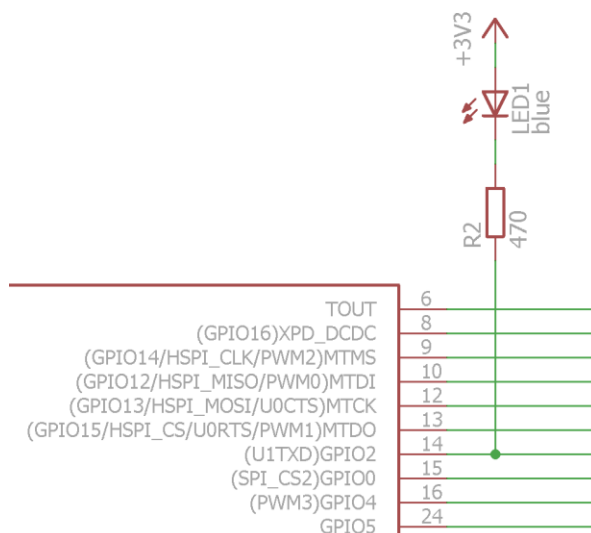
Paso 15. Declaramos como salida el puerto digital donde está conectado el LED interno (definido como LED_BUILIN). Este LED se encuentra conectado al GPIO 2.



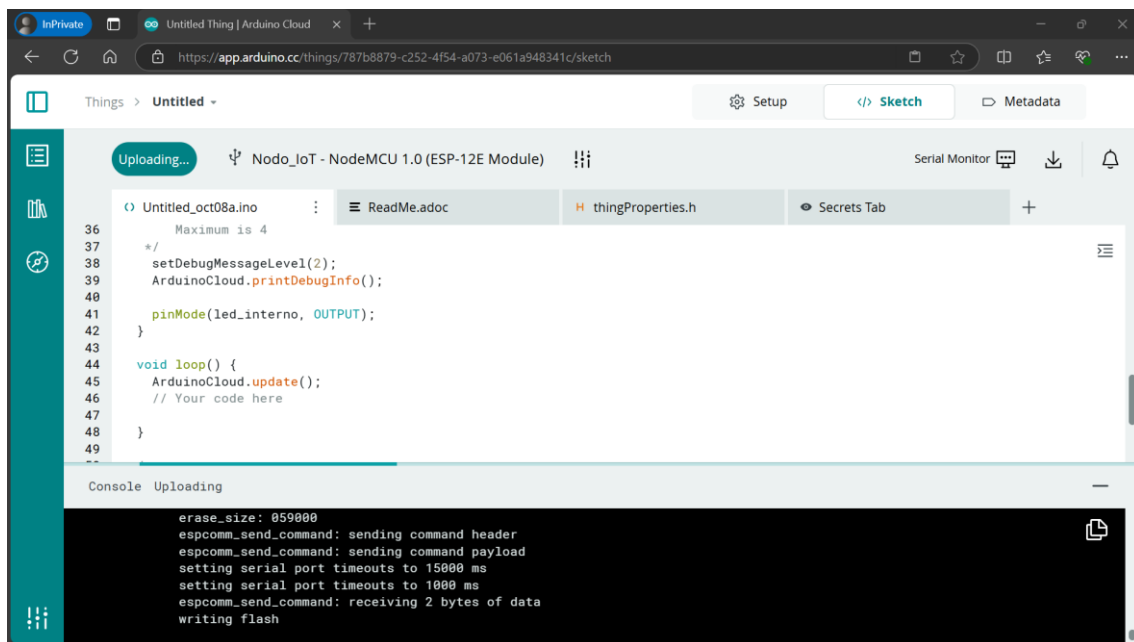
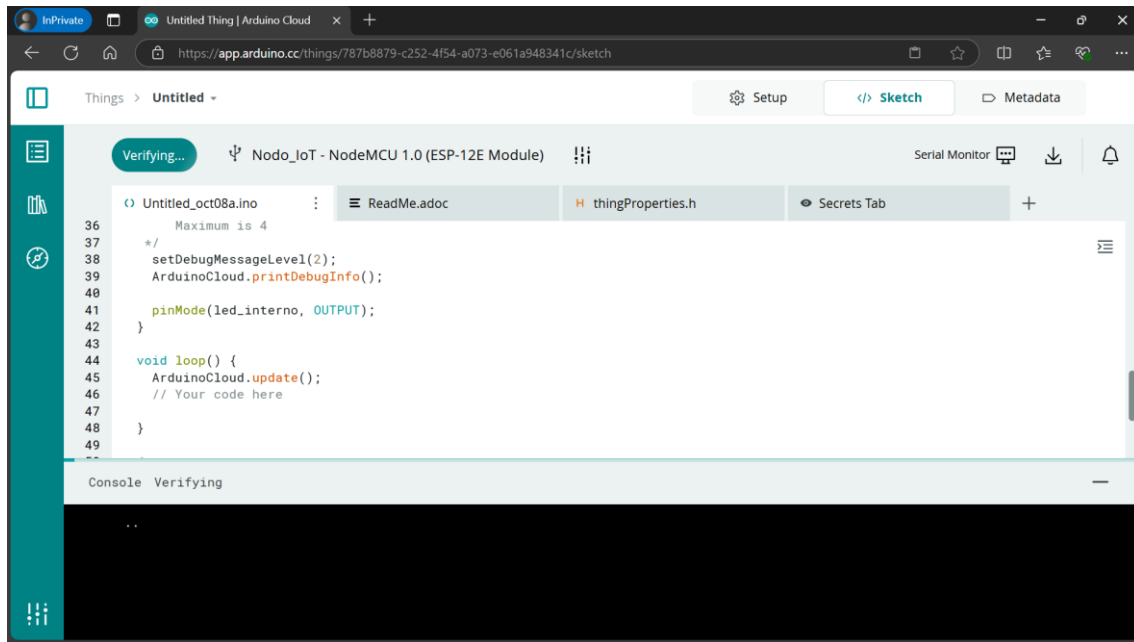
```

1  /*
2  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
3  https://create.arduino.cc/cloud/things/787b8879-c252-4f54-a073-e061a948341c
4
5  Arduino IoT Cloud Variables description
6
7  The following variables are automatically generated and updated when changes are made to the Thing
8
9  bool estado;
10
11 Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
12 which are called when their values are changed from the Dashboard.
13 These functions are generated with the Thing and added at the end of this sketch.
14 */
15
16 int led_interno=2; //Pin asociado al led interno
17
18 #include "thingProperties.h"
19
20 void setup() {
21   // Initialize serial and wait for port to open:
22   Serial.begin(9600);
  
```

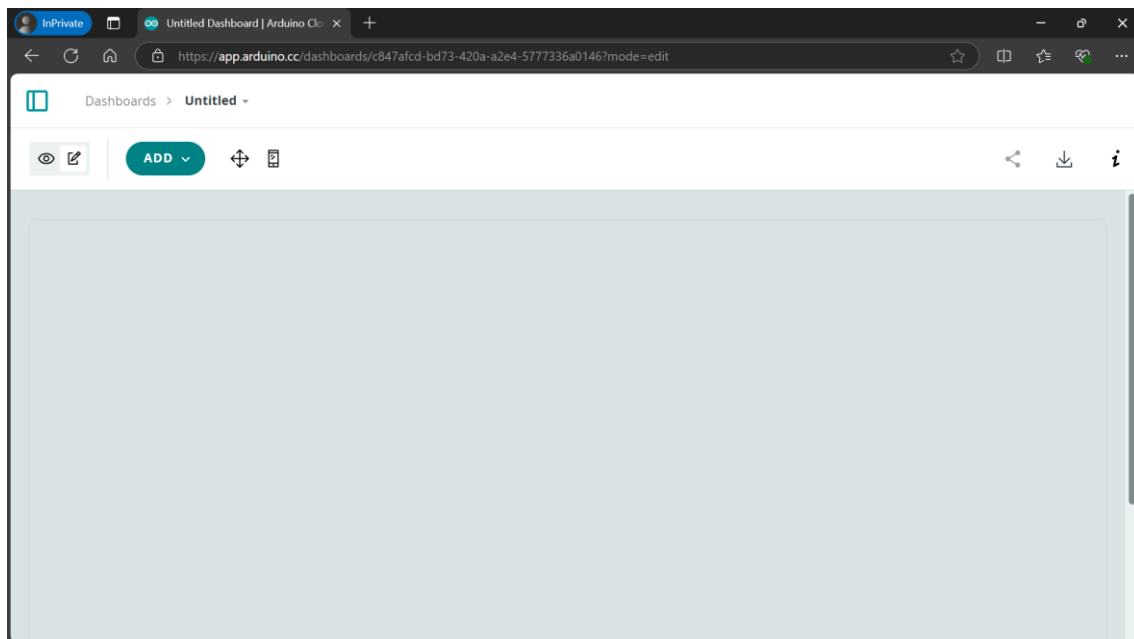
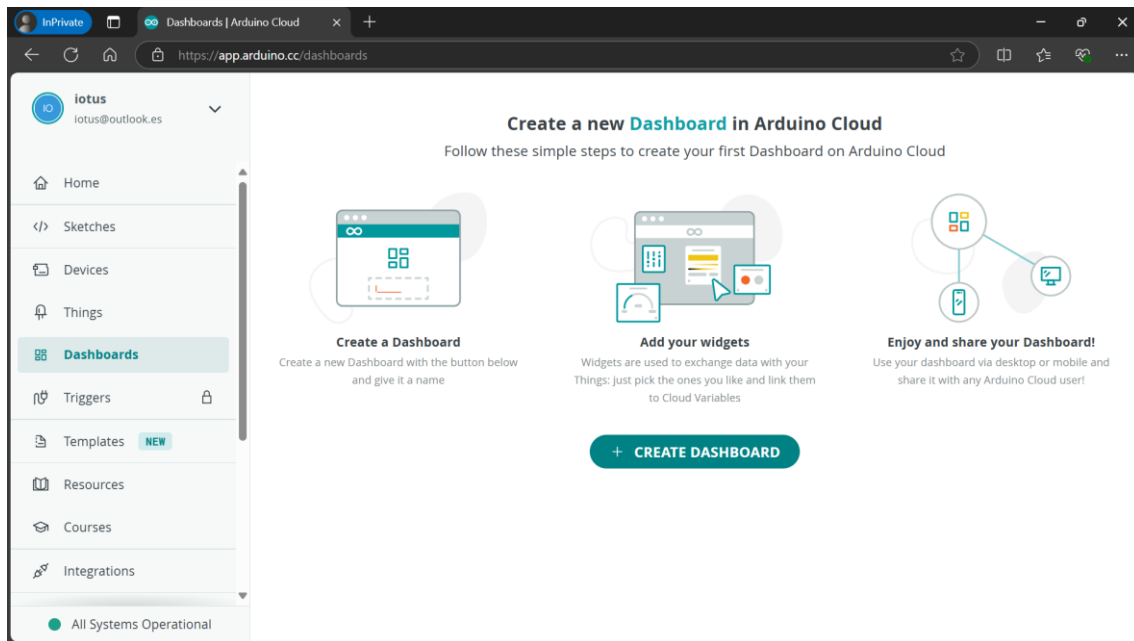
Paso 16. En la función que gestiona el evento de cambio de la variable en la nube “Estado” asociar el estado del LED interno con la variable. Debe tenerse en cuenta el tipo de conexionado existente en la placa entre el LED interno y el puerto asociado. Cuando en el puerto hay un cero lógico el LED está en ON y cuando en el puerto hay un uno lógico el LED está en OFF.



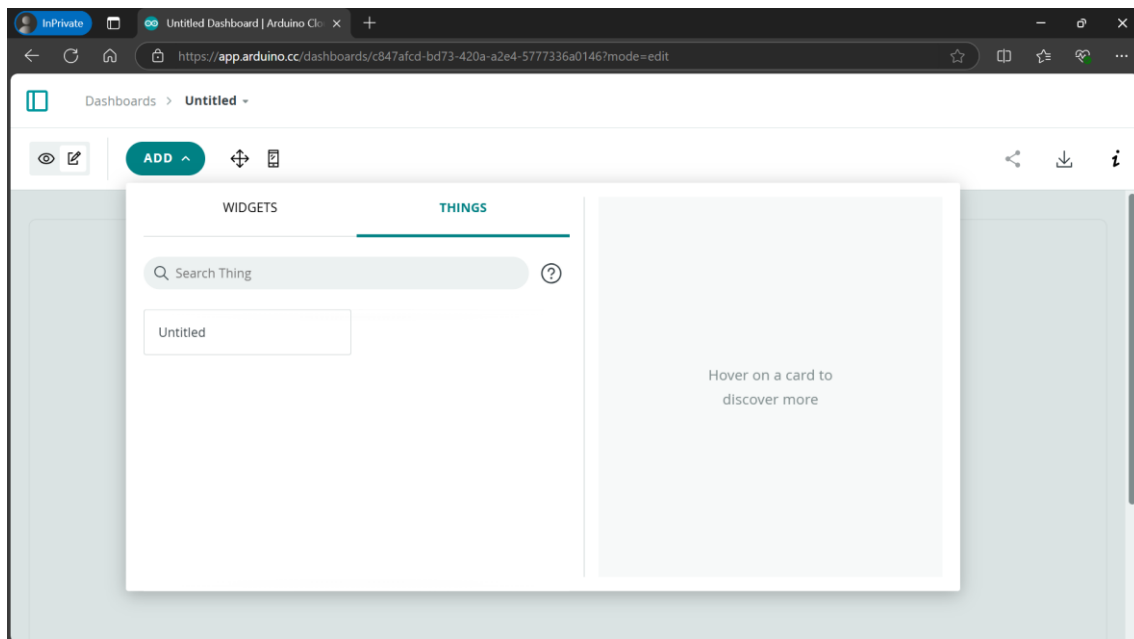
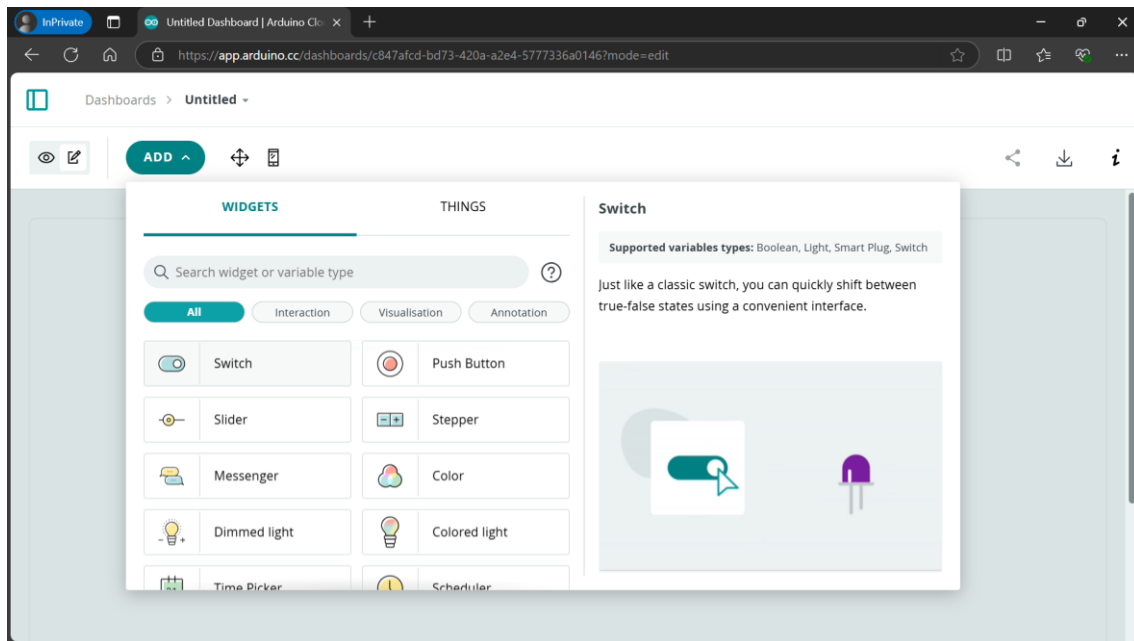
Paso 17. Verificar el código y cargarlo en el dispositivo IoT.

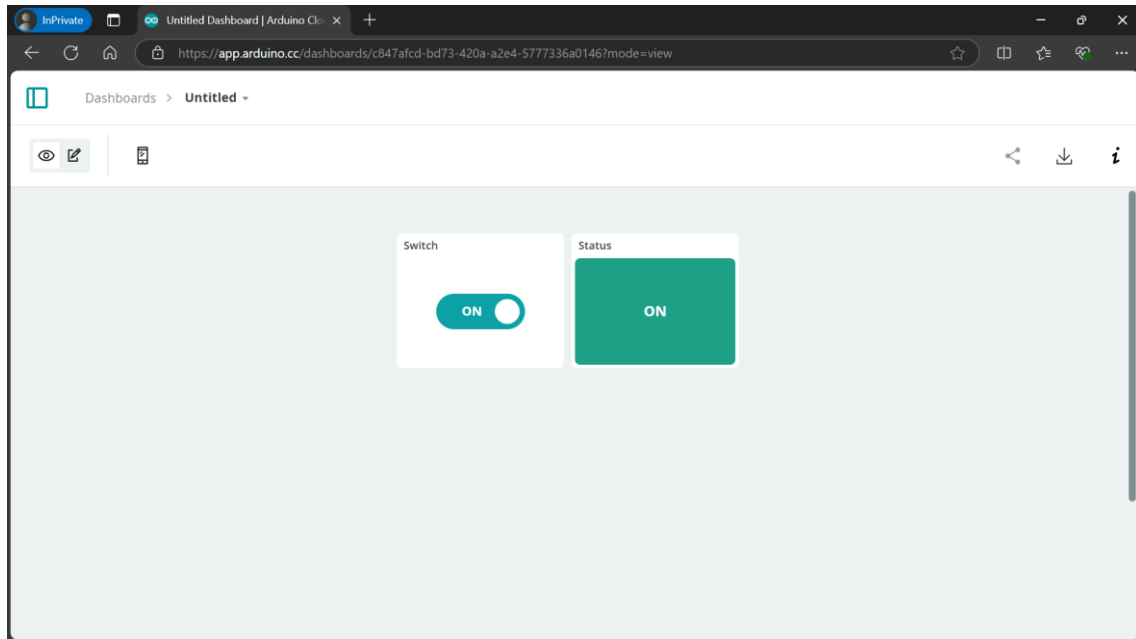
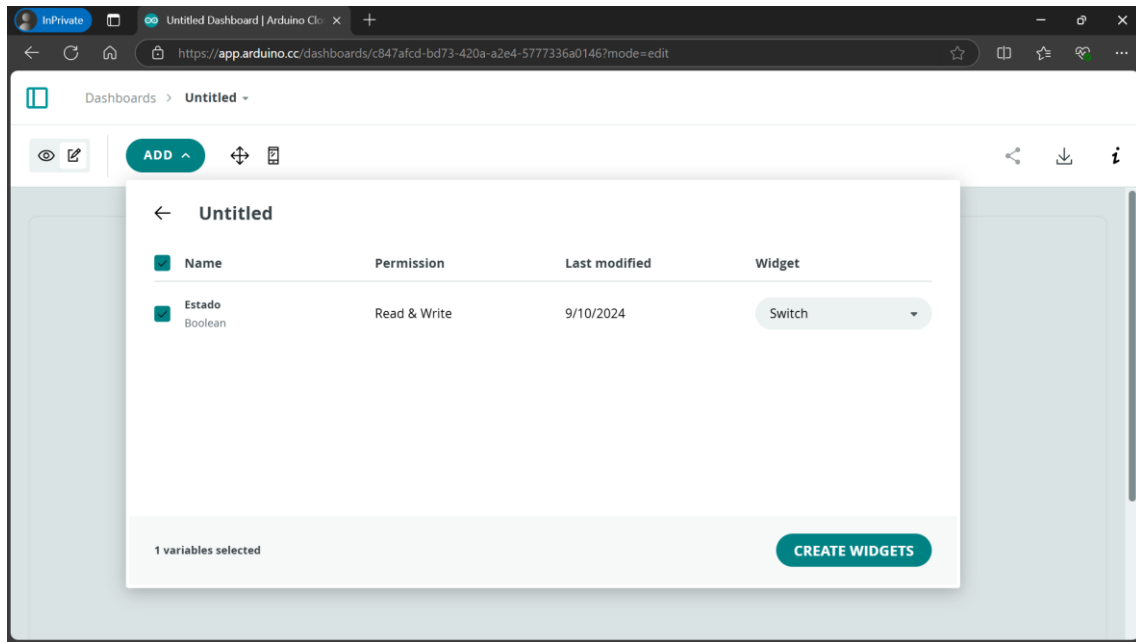


Paso 18. En el menú lateral seleccionar la sección de cuadros de mando (“Dashboards”). Crear un nuevo cuadro de mandos.



Paso 19. Añadir objetos para monitorización del estado de la variable y para cambiar su valor.



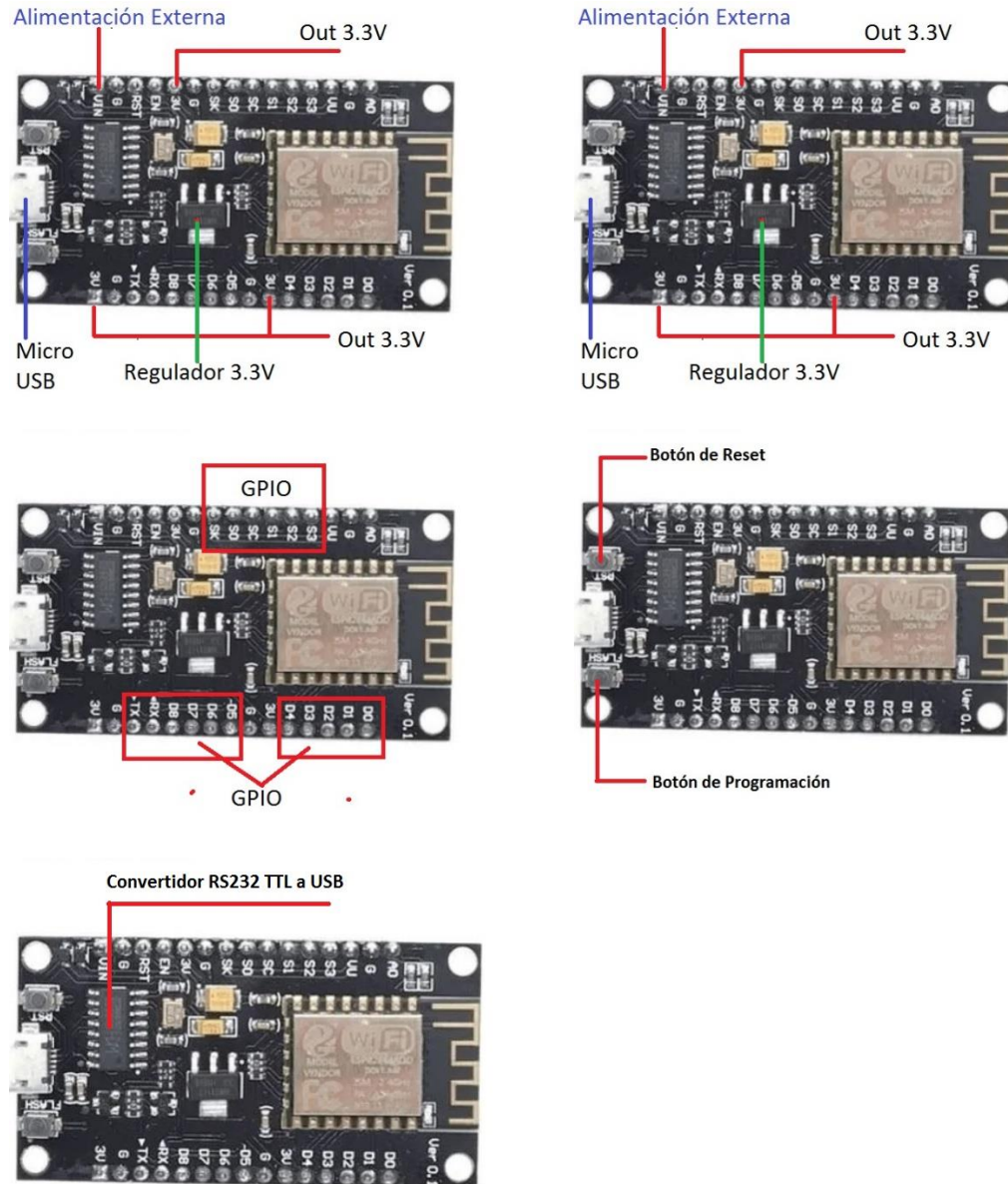


Paso 20. Realizar pruebas de funcionamiento desde el cuadro de mando (versión web y APP de móvil).

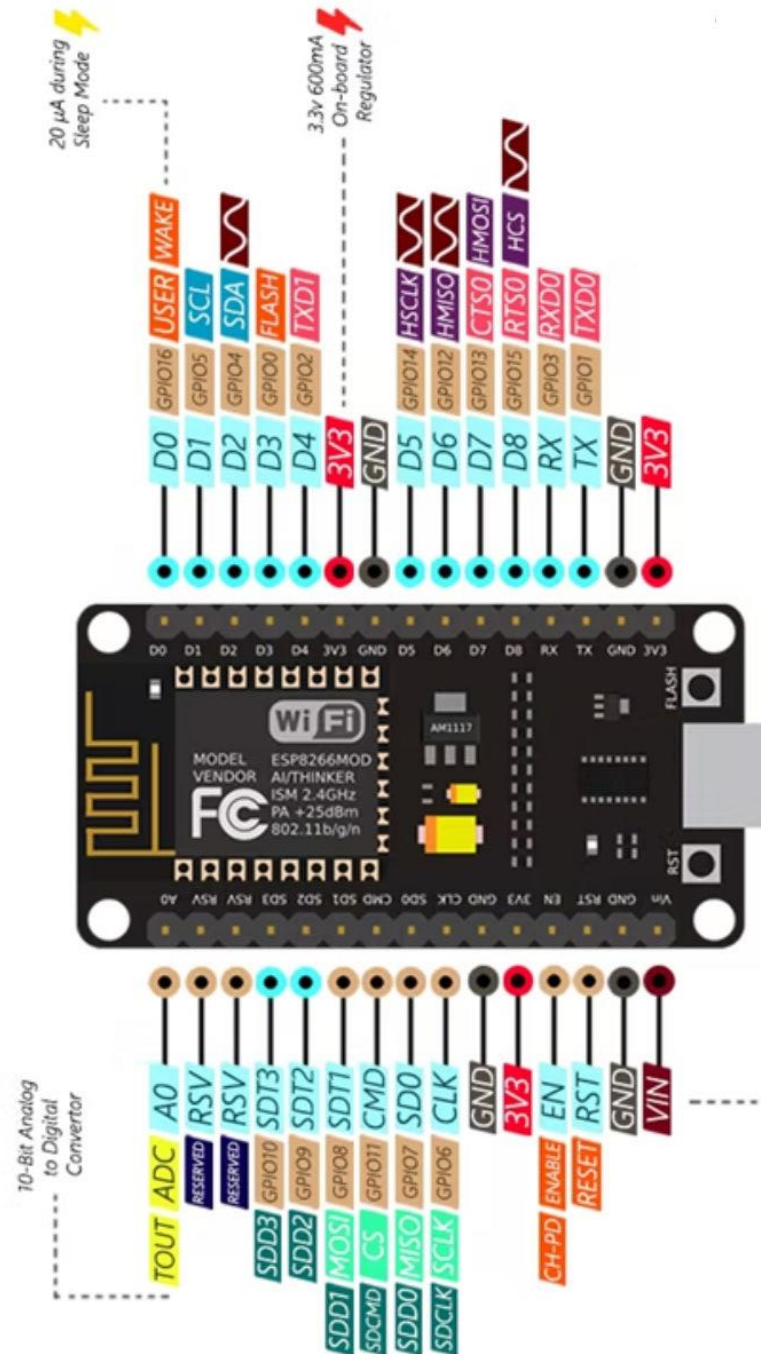
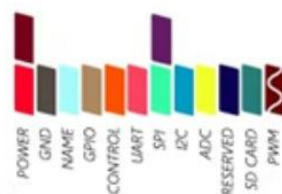
Paso 21. Ejercicios:

1. Conectar un LED con la correspondiente resistencia de protección en la salida digital GPIO15 de la plataforma. Debe tenerse en cuenta la notación del patillaje (pinout).
2. Programar el dispositivo para asociar el evento de cambio de la variable “estado” con la escritura en la salida asociada al LED conectado externamente.
3. Incluir en el cuadro de mando creado un objeto para visualización del LED externo y un elemento de tipo pulsador (Push Button) para cambio del valor del estado del LED externo.

4. Información técnica de la plataforma NodeMCU V1.0 / V3

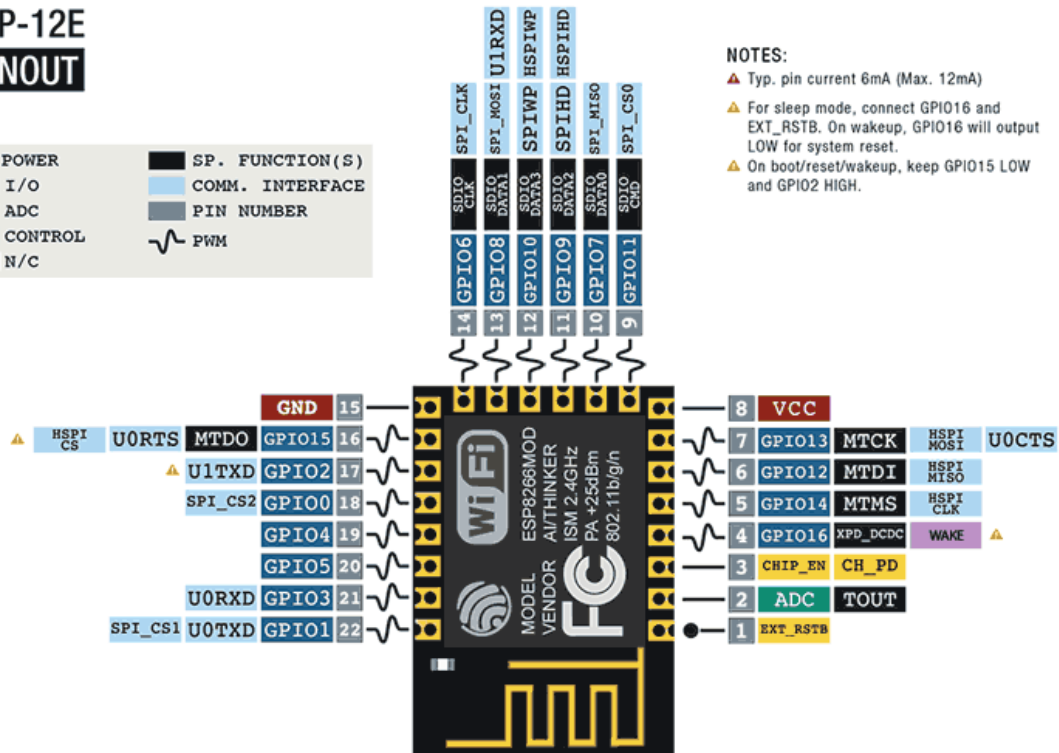


Pinouts:

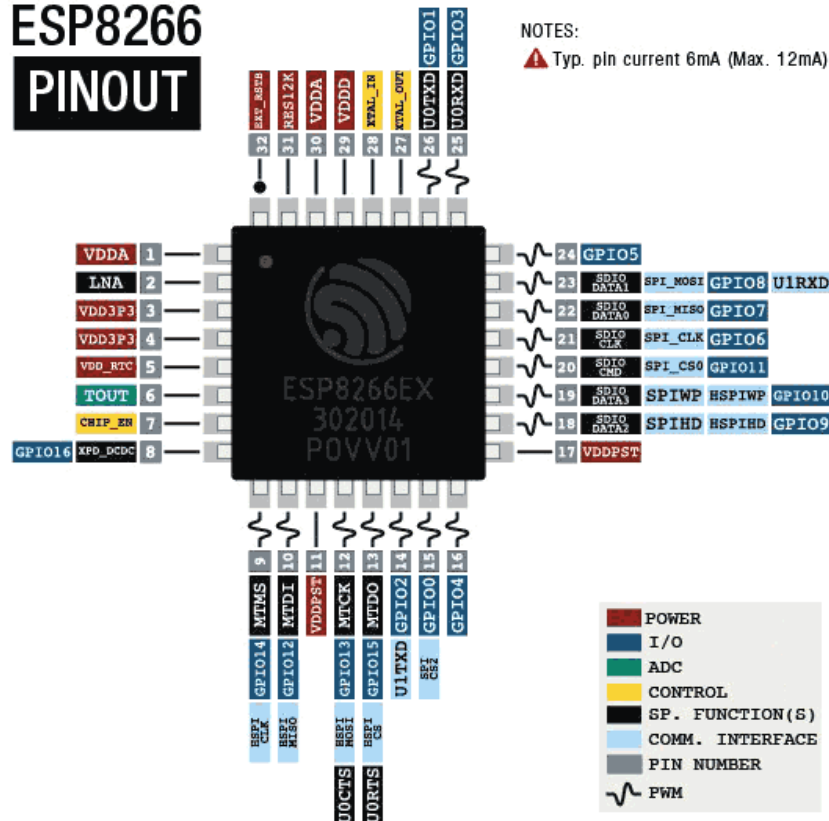


ESP-12E PINOUT

POWER	SP. FUNCTION(S)
I/O	COMM. INTERFACE
ADC	PIN NUMBER
CONTROL	PWM
N/C	



ESP8266 PINOUT



Pin	GPIO	Input	Output	Comentarios
D0	GPIO16	No interrupciones	No PWM No I2C	HIGH durante boot Resistencia Pull-Down Conectar a RST para Wake-Up
D1	GPIO5	OK	OK	SCL (I2C) (frecuentemente)
D2	GPIO4	OK	OK	SDA (I2c) (frecuentemente)
D3	GPIO0	Pulled Up	OK	Boot falla si pulled LOW Conectado a botón FLASH
D4	GPIO2	Pulled Up	OK	HIGH durante boot Boot falla si pulled LOW Built-in LED TX1
D5	GPIO14	OK	OK	SLCK (SPI)
D6	GPIO12	OK	OK	MISO (SPI)
D7	GPIO13	OK	OK	MOSI (SPI)
D8	GPIO15	Pulled GND	OK	CS (SPI) LOW durante boot Boot falla si pulled HIGH No tiene Pull-Up
RX	GPIO3	OK	RX	HIGH durante boot No usable si se usa UART
TX	GPIO1	TX	OK	HIGH durante boot Boot falla si pulled LOW Debug output en boot No usable si se usa UART
A0	ADC0	Analog Input	NO	