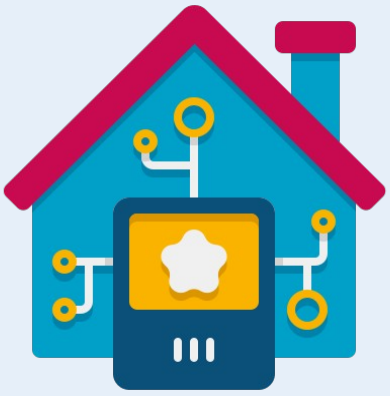
APETEGA

Domótica con MQTT e Node-Red

Prácticas Node-Red MQTT



Flaticon

1. Práctica 1. Conexión do ESP32 á rede WIFI e ao servidor MQTT

Nesta primeira práctica comprobaremos que o ESP32 se conecta correctamente á wifi e ao servidor MQTT.

Pasos a seguir:

1. Conectamos o ESP32 ao ordenador.
2. Abrimos o IDE de Arduino.
3. Configuramos o IDE para traballar coas ESP32:
    1. Archivo -> Preferencias -> URLs adicionales de gestor de placas: http://drazzy.com/package_drazzy.com_index.json,https://dl.espressif.com/dl/package_esp32_index.json
    2. Herramientas -> Placa -> Gestor de placas -> escribes ESP32 no buscador e instalas o que pon esp32 de Espressif Systems
    3. Seleccionamos a placa ESP32 Wrover kit (all versions)
4. Cargamos no IDE o programa Practica1.ino que tedes na aula virtual.
5. Cambiamos os datos necesarios para que funcione: SSID, contrasinal, IP do servidor MQTT
6. Conectamos o ESP32 co cable USB e comprobamos que o porto seleccionado é o correcto.
7. Cargamos o programa no ESP32.
8. Abrimos o monitor serie e comprobamos que o ESP se conecta correctamente

2. Práctica 2. Control dun LED

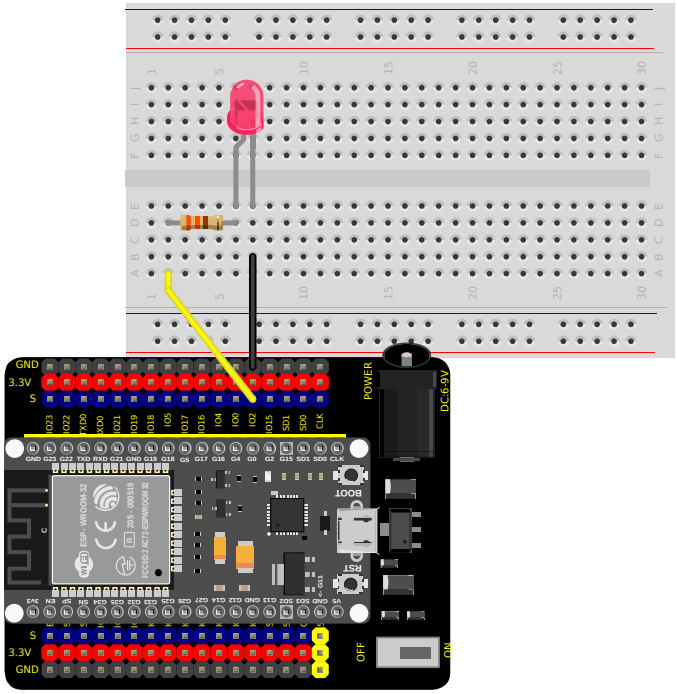
Como non podía ser doutro xeito a primeira práctica consistirá en acender e apagar un LED, pero neste caso o apagado e acendido será desde o panel de Node-Red.

Material necesario:

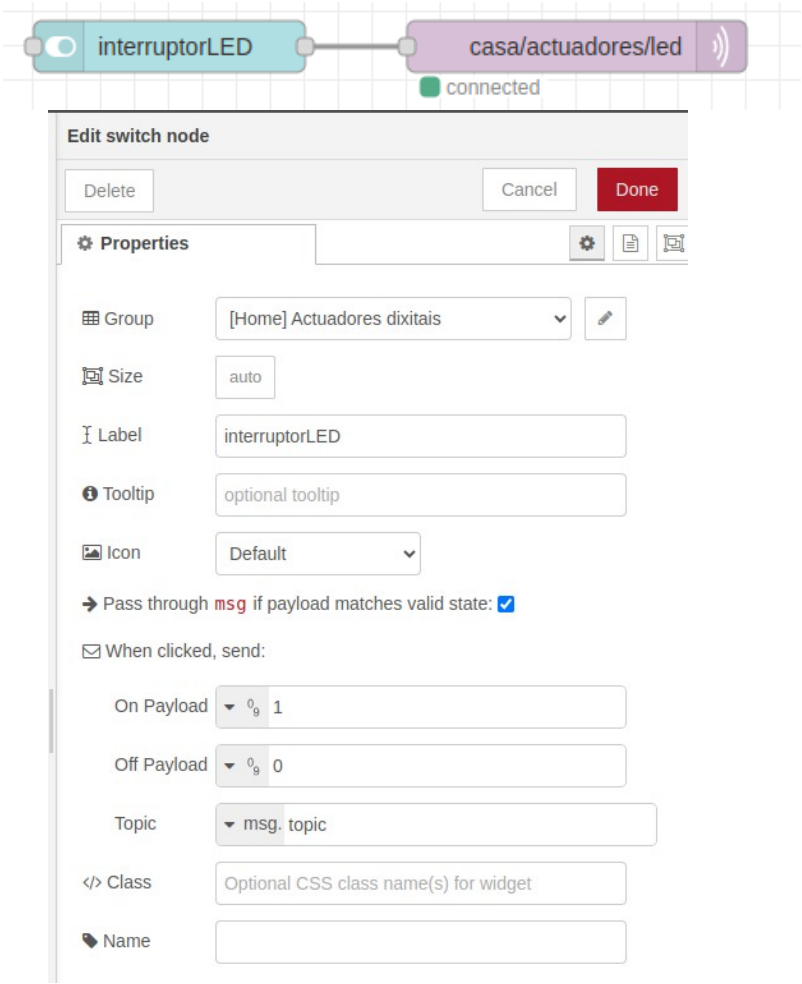
- ESP32 WROOM + shield
- LED 5mm
- Resistencia de 330 Ω
- Placa de conexións
- Cables M-F

Pasos a seguir:

1. Para poder controlar un LED desde o panel de control temos que crear o fluxo en Node-Red. Como temos que telo preparado para o resto de prácticas xa facemos o fluxo completo.
2. Todos os topics teremos que utilizalos nos programas, polo que é importante estruturalos correctamente e non equivocarse ao escribilos nos programas.
3. No documento Mapa_Topics podes ver a proposta de topics.
4. Neste caso creamos o topic casa/actuadores/led e utilizamos os nodos **switch** e **mqtt out**, que chamaremos **interruptorLED** e **casa/actuadores/led** para que no fluxo teñamos claro o que fan eses nodos.
5. Abrimos o programa Práctica2_LED.ino no IDE de Arduino.
6. Comprobamos que a configuración dos nodos do fluxo de Node Red se corresponde coa que aparece no código do programa. Que estamos a usar o mesmo topic e as mesmas mensaxes.
7. O esquema de conexións, a configuración do nodo **switch** e o fluxo de Node-Red podedes velos máis abaixo. LED conectado ao GPIO 2.
8. Cargamos o programa no ESP32.
9. Abrimos no navegador o panel (Dashboard): ip do servidor:1880/ui
10. Comprobamos que ao accionar o interruptor o LED acende e apaga.



Fritzing


interruptorLED
casa/actuadores/led
connected
Edit switch node
Delete
Cancel
Done
Properties
Group
[Home] Actuadores dixitais
Size
auto
Label
interruptorLED
Tooltip
optional tooltip
Icon
Default
Pass through msg if payload matches valid state:
When clicked, send:
On Payload
1
Off Payload
0
Topic
msg. topic
Class
Optional CSS class name(s) for widget
Name

3. Práctica 3. Relé

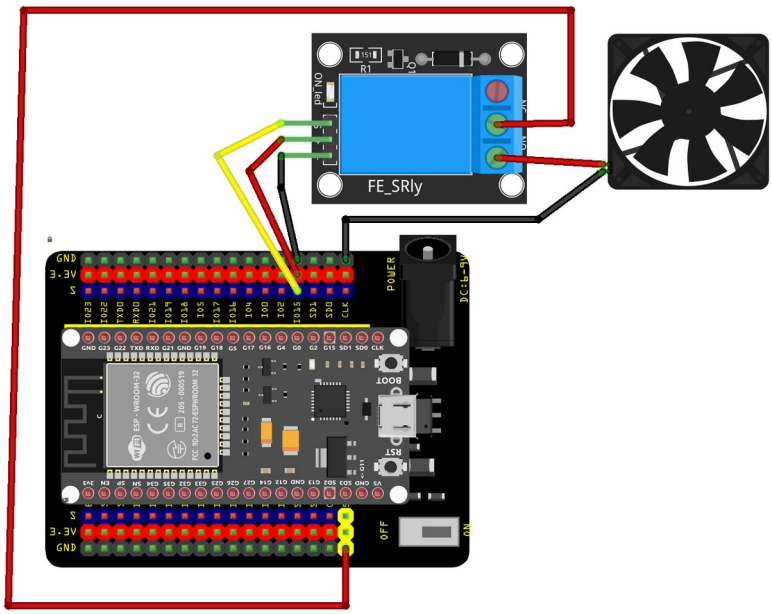
O relé funcionará, a efectos prácticos como o LED, será un interruptor que activaremos e desactivaremos para que este á súa vez acenda ou apague un actuador que necesita máis potencia que a que pode proporcionar o ESP32. No noso exemplo utilizaremos un ventilador que funciona con 5 V.

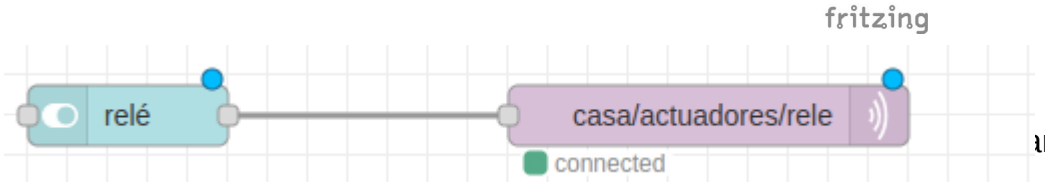
Material necesario:

- ESP32 WROOM + shield
- Módulo relé.
- Ventilador 5V.
- Cables M-M, M-F, F-F

Pasos a seguir:

1. Seguimos os mesmos pasos que na práctica do LED, pero co programa P3_Rele.ino. Ao nodo switch que creamos en Node-Red chamarémolo **relé** e ao módulo mqtt out chamámolo **casa/actuadores/rele**
2. O esquema de conexións está máis abaixo, o relé está conectado ao GPIO 15. O relé conectámolo a un dos pins do shield que proporcionan 5V.
3. Unha vez teñamos creados os nodos en Node-Red, conectados os compoñentes na placa e cargado o programa no ESP32 abrimos o panel de control e comprobamos que o relé funciona correctamente, facendo que o ventilador acenda e apague.





4. Práctica 4. Zoador.

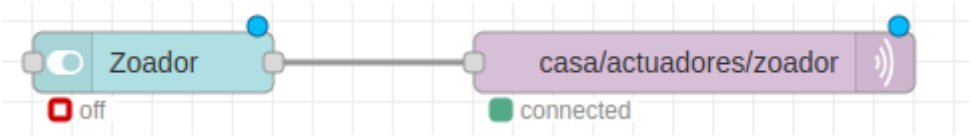
O zoador será activo e o activamos e desactivamos igual que ao LED e o relé. A patilla + do zoador conectámola ao GPIO12, a través dunha resistencia de 220Ω e a outra ao GND.

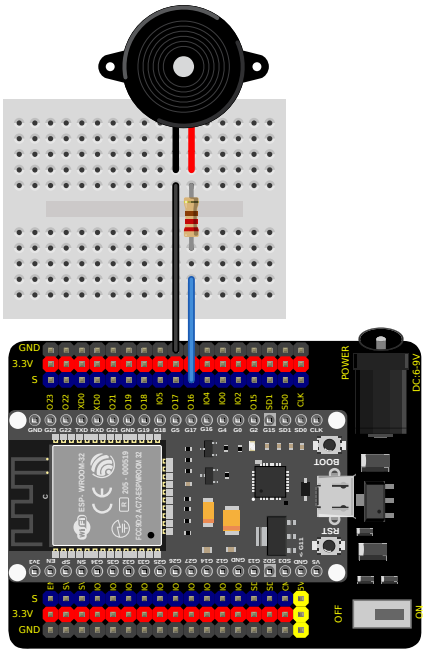
Material necesario:

- ESP32 WROOM + shield
- Zoador.
- Cables F-F

Pasos a seguir:

1. Seguimos os mesmos pasos que na práctica do LED, pero co programa P4_zoador.ino. Ao nodo switch que creamos en Node-Red chamarémolo **Zoador** e ao módulo mqtt out chamámolo **casa/actuadores/zoador**
2. O esquema de conexións e o fluxo de Node-Red están máis abaixo.
3. Unha vez teñamos creado os nodos en Node-Red, conectados os compoñentes na placa e cargado o programa no ESP32 abrimos o panel de control e comprobamos que o zoador funciona correctamente, facendo ruído ao activalo.





5. Práctica 5. Microservo

O control dun servo será moi útil para poder mover múltiples mecanismos. No exemplo da casa utilizamos unha porta accionada por un mecanismo de piñón – cremalleira.

Na placa que utilizamos podemos conectar a alimentación do servo a 3,3V ou a 5V, nos dous casos funcionará, pero xa que o utilizamos para mover unha porta conectarémolo ao pin de 5V, e ao GPIO5.

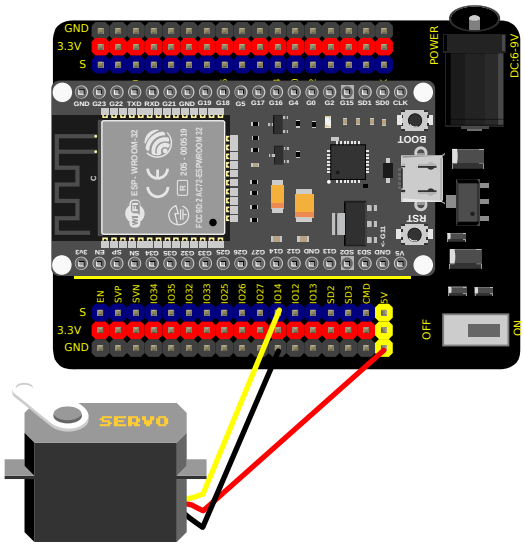
Para manexar o servo coa biblioteca ESP32Servo enviamos un valor ao servo de entre 0 e 100, que se traduce nun movemento do servo de entre 0 e 180º.

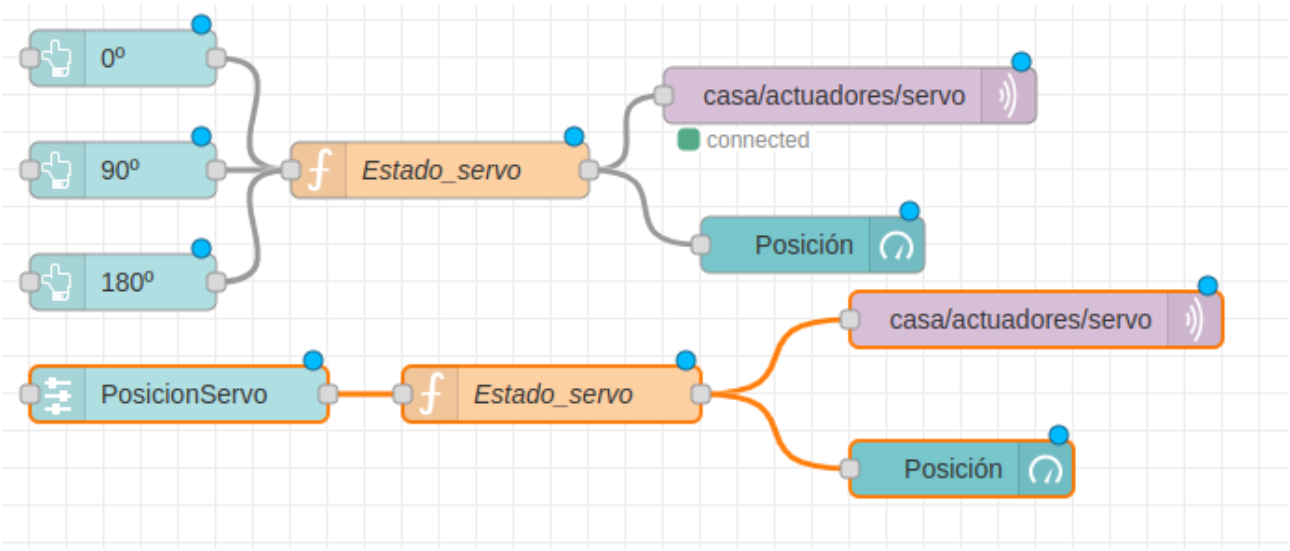
En Node-Red podemos crear botóns para posicións concretas ou utilizar un “slider” con valores 0-100% que o noso programa transformará en valores de entre 0 a 180º.

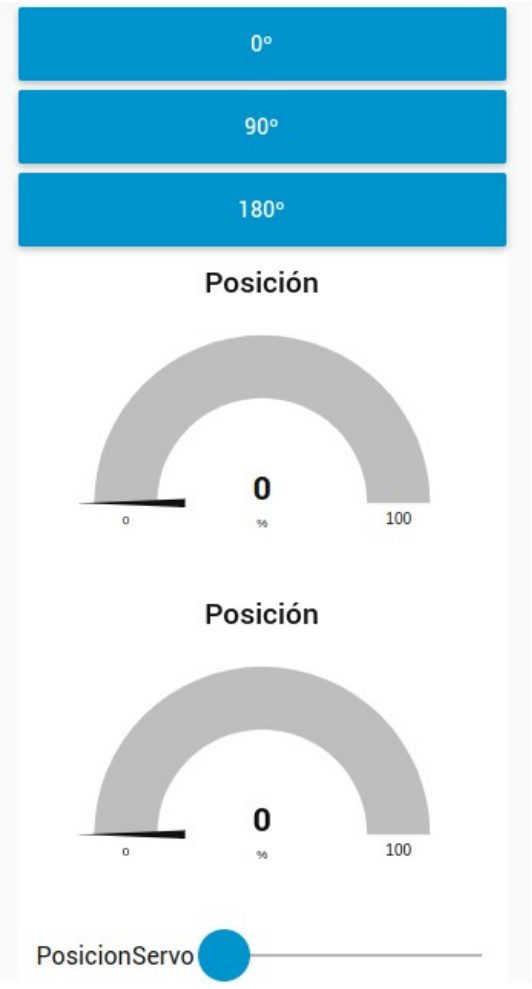
- ESP32 WROOM + shield Microservo 9g Cable M-F

Pasos a seguir:

1. O esquema de conexións e os nodos en Node-Red son os das imaxes.
2. Creamos 3 botóns para as posicións 0, 90 e 180º. Outra opción é utilizar un nodo **slider** para seleccionar a posición de 0 a 100%, pero non é aconsellable utilizar os dous á vez xa que podemos enviar ordes contraditorias a un mesmo servo e bloquealo.
3. O módulo **function** serve para poder enviar a orde que enviamos ao módulo **mqtt out** e a un indicador de posición. Ao módulo **mqtt out** ten a etiqueta **casa/actuadores/servo**, que o mesmo que o topic.
4. Instalamos no ESP32 o programa Practica5_servo.ino , desde o IDE Arduino.
5. Unha vez teñamos creado os nodos en Node-Red, conectados os compoñentes na placa e cargado o programa no ESP32 abrimos o panel de control e comprobamos que o servo funciona correctamente, movéndose á posición que ordenamos.




0º
90º
180º
Estado_servo
casa/actuadores/servo
connected
Posición
PosicionServo
Estado_servo
casa/actuadores/servo
Posición


0º
90º
180º
Posición
0
%
0
100
Posición
0
%
0
100
PosicionServo

6. Práctica 6. LED RGB

Neste exemplo utilizaremos a biblioteca ArduinoJson.h, polo que temos que instalala desde o xestor de bibliotecas do IDE de Arduino, instalamos a de Benoit Blanchon.

Gestor de bibliotecas -> buscar ArduinoJson -> clic en instalar a que buscamos

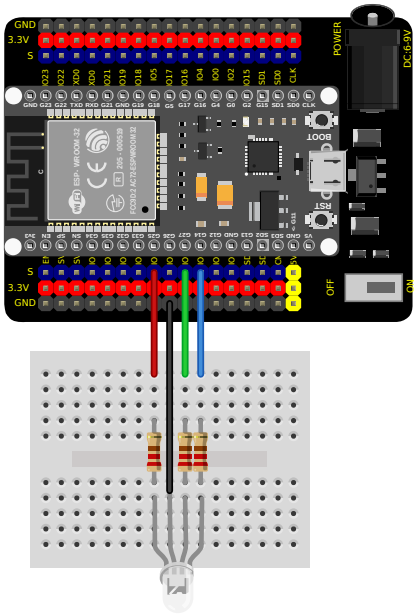
Utilizamos 3 GPIO: 25 para R, 26 para G e 27 para B.

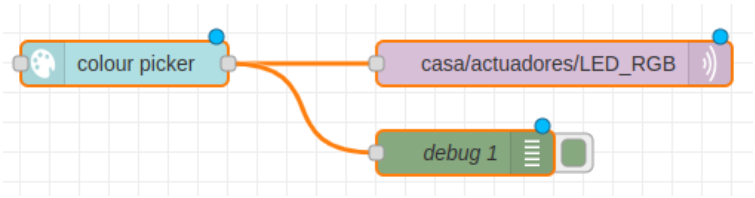
Material necesario:

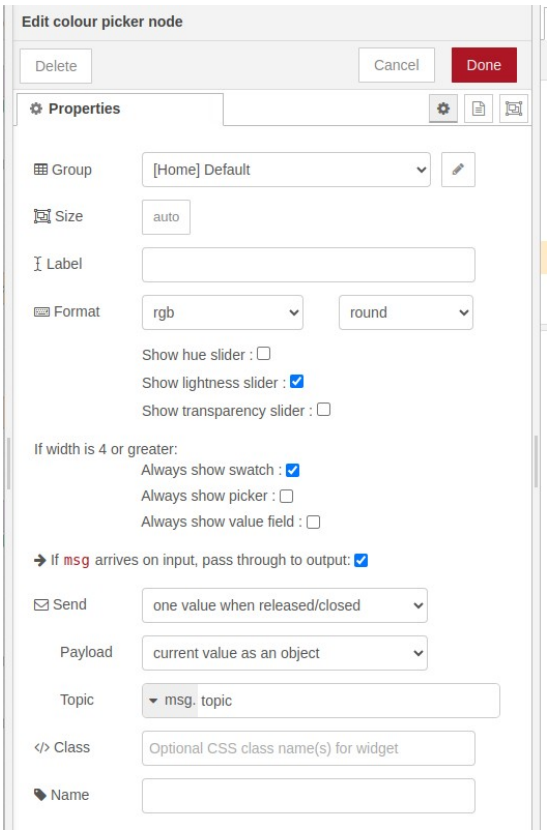
- ESP32 WROOM + shield
- 3 resistencias 220Ω
- LED RGB 5mm. Importante saber se é de ánodo ou de cátodo común.
- Placa protoboard
- Cables M-F

Pasos a seguir:

1. O esquema de conexións e os nodos en Node-Red son os das imaxes. Na configuración do nodo **Colour picker** selecciona: **Format** -> RGB; **Show lightness slider**; **Payload** -> one value as an object
2. Na configuración do nodo **mqtt out** escribir **casa/actuadores/LED_RGB** como **Topic**
3. Engadimos un nodo **debug** para poder comprobar a mensaxe enviada cando seleccionamos unha cor no panel.
4. Instalamos no ESP32 o programa Practica6_RGB.ino , desde o IDE Arduino.
5. Unha vez teñamos creado o fluxo en Node-Red, conectados os compoñentes na placa e cargado o programa no ESP32 abrimos o panel de control e comprobamos que podemos seleccionar a cor e a intensidade.


GND
3.3V
POWER
DC-6-9V
BOOT
RST
OFF
ON


colour picker
casa/actuadores/LED_RGB
debug 1


Edit colour picker node
Delete
Cancel
Done
Properties
Group
[Home] Default
Size
auto
Label
Format
rgb
round
Show hue slider :
Show lightness slider :
Show transparency slider :
If width is 4 or greater:
Always show swatch :
Always show picker :
Always show value field :
If msg arrives on input, pass through to output:
Send
one value when released/closed
Payload
current value as an object
Topic
msg. topic
Class
Optional CSS class name(s) for widget
Name

7. Práctica 7. Sensor Temperatura e humidade DHT11.

Tomamos como exemplo o programa utilizado no curso de FP do CFR de Vigo de 2019:

https://www.edu.xunta.gal/centros/cfrvigo/aulavirtual/mod/resource/view.php?id=13930&forceview=1

O sensor DHT11 mide a temperatura e a humidade utilizando un único GPIO, non é moi exacto (+-2º), pero é moi económico.

Para usalo instalamos as bibliotecas DHT.h e DHT_U.h.

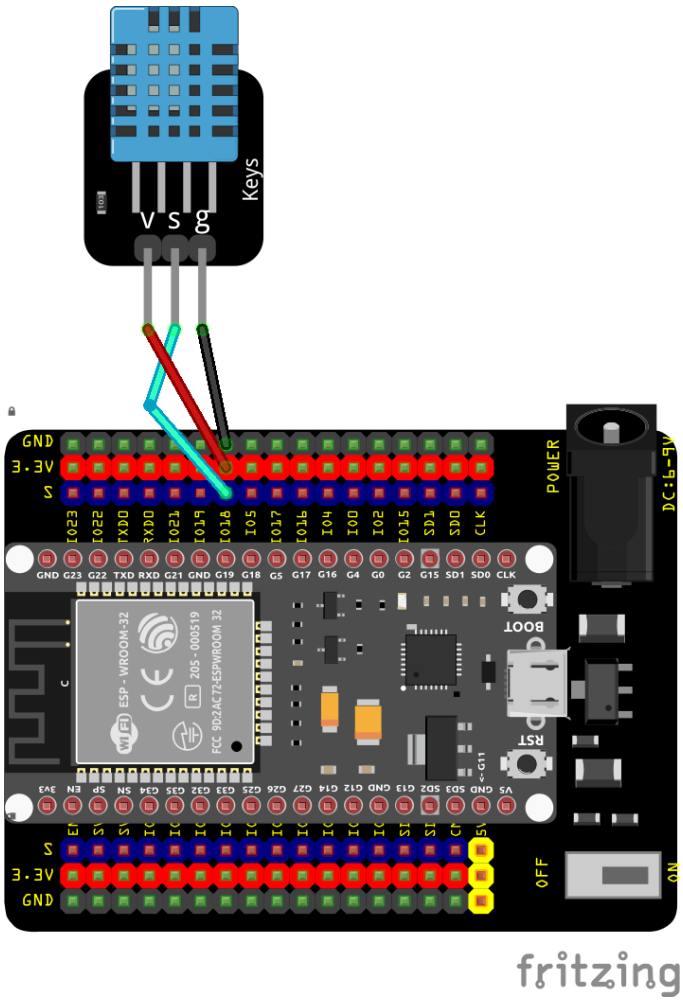
Conectamos o sensor ao GPIO 18.

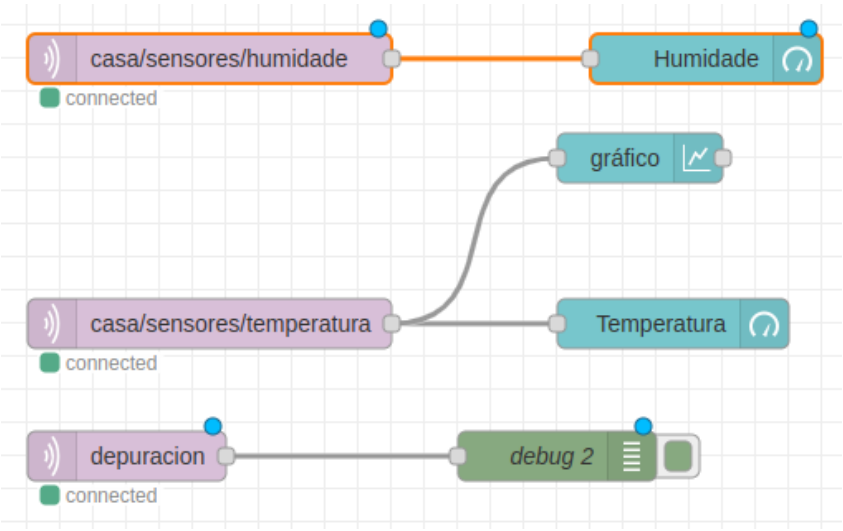
Material necesario:

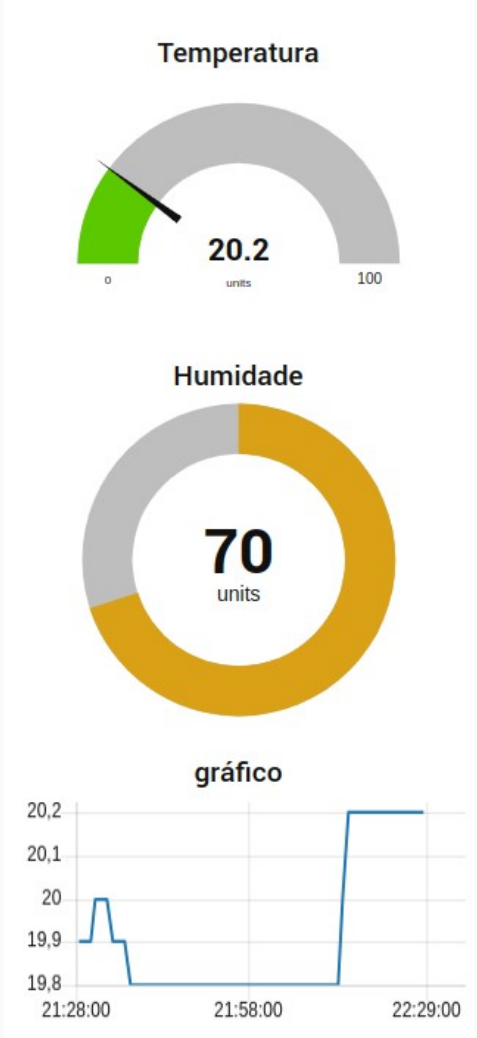
- ESP32 WROOM + shield
- Sensor DHT11
- Cables F-F

Pasos a seguir:

1. O esquema de conexións e os nodos en Node-Red son os das imaxes.
2. Na configuración do nodo **mqtt out,** que utilizamos para a humidade,escribir **casa/sensores/humidade** como **Topic**
3. Na configuración do nodo **mqtt out,** que utilizamos para a temperatura,escribir **casa/sensores/temperatura** como **Topic**
4. Engadimos un nodo **debug** para poder comprobar se o sensor está enviado mensaxes ao broker. Na configuración do nodo **mqtt in** que conectamos ao **debug** so temos que escribir correctamente o topic (depuracion) e seleccionar o noso broker mqtt.
5. Para visualizar os datos de temperatura e humidade utilizamos nodos **gauge**, que chamaremos temperatura e humidade. O de temperatura é tipo **gauge** e o de humidade o seleccionamos tipo **donut.** Tamén cambiamos o rango de unidades, de 0 a 100.
6. No caso da temperatura engadimos un nodo **chart**, que chamamos **gráfico**, como exemplo de visualización da evolución dos datos dun sensor.
7. Instalamos no ESP32 o programa Practica7_DHT11.ino, desde o IDE Arduino.
8. Unha vez teñamos creado o fluxo en Node-Red, conectados os compoñentes na placa e cargado o programa no ESP32 abrimos o panel de control e comprobamos que podemos ver a temperatura e humidade.


V S g
Keys
GND
3.3V
S
POWER
BOOT
RST
ESP - WROOM-32
fritzing


casa/sensores/humidade
connected
Humidade
gráfico
casa/sensores/temperatura
connected
Temperatura
depuracion
connected
debug 2


Temperatura
20.2
0
units
100
Humidade
70
units
gráfico
20,2
20,1
20
19,9
19,8
21:28:00
21:58:00
22:29:00

8. Práctica 8. Botón.

O botón e o exemplo dun sensor dixital, que pode adoptar dous estados pulsado - non pulsado; 0 – 1; 0 – 3,3V, ...

Para o noso exemplo conectaremos o botón en modo pull-down. (https://tecnoloxia.org/robotica/pull-up-e-pull-down/)

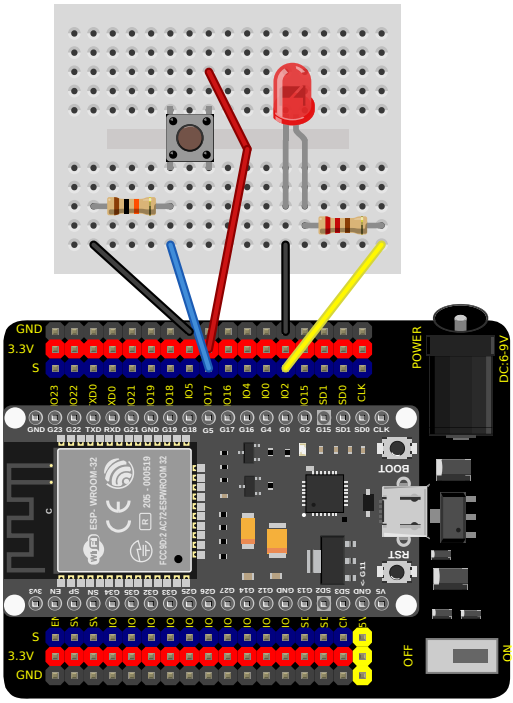
Para poder comprobar como funciona o botón utilizaremos un LED tal como o fixemos na práctica 1, no GPIO2, pero esta vez o LED acenderá e apagará cando pulsemos o botón conectado ao GPIO4.

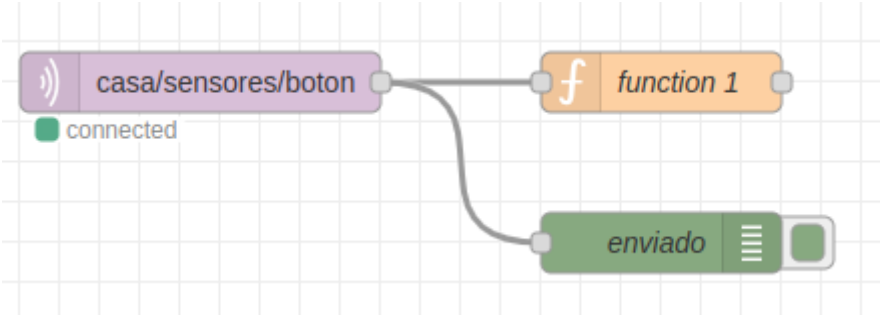
Material necesario:

- ESP32 WROOM + shield
- Pulsador
- Resistencia de 10kΩ
- LED
- Resistencia de 220Ω
- Protoboard
- Cables M-F

Pasos a seguir:

1. O esquema de conexións e o fluxo en Node-Red son os das imaxes.
2. Na configuración do nodo **mqtt in,** escribir **casa/sensores/boton** como **Topic**
3. No nodo **function** cargamos o programa en JavaScript **function_boton.js**, que fará que cando chegue a mensaxe de botón pulsado ao nodo mqtt in se acenda o LED.
4. O nodo **debug** que chamamos enviado serve para comprobar que mensaxe chega ao topic **casa/sensores/boton**
5. Instalamos no ESP32 o programa **Practica8_boton.ino**, desde o IDE Arduino.
6. Unha vez teñamos creado o fluxo en Node-Red, conectados os compoñentes na placa e cargado o programa no ESP32 abrimos o panel de control e comprobamos que ao pulsar o botón acende o LED e cando non o pulsamos o LED permanece apagado.


GND
3.3V
POWER
DC:6-9V
BOOT
RST
OFF
ON


casa/sensores/boton
connected
function 1
enviado

9. Práctica 9. Sensor PIR + zoador.

Un sensor infravermello pasivo (ou sensor PIR) é un sensor electrónico que mide a luz infravermella (IR) radiada dos obxectos situados no seu campo de visión. Utilízanse principalmente nos detectores de movemento baseados en PIR.

Nos utilizaremos un módulo PIR, pensado para utilizar coa placa Arduino, que é un sensor dixital que dará unha saída de 3,3V cando detecte presenza e de 0V cando non detecte presenza. O sensor ten dous potenciómetros que serven para axustar a sensibilidade e o tempo que pasa entre que detecta presenza e volve a estar activo.

Polo tanto está práctica é igual que a anterior no referente a tipos de dispositivos e incluso o código é igual. So cambian os nomes dos sensores e actuadores e os GPIOs nos que os conectamos, o topic ao que se subscribe e as mensaxes que aparecen no monitor serie.

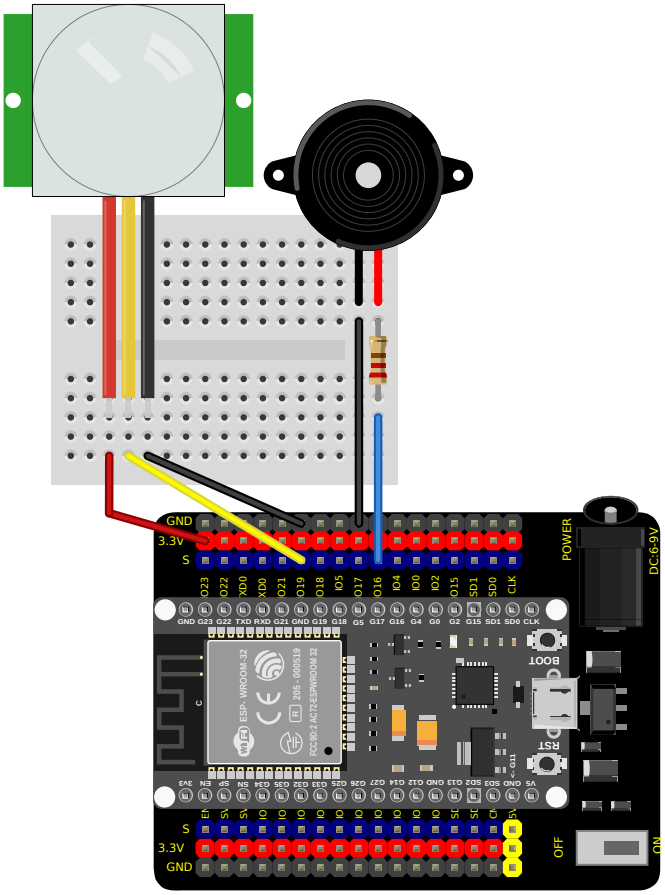
Sensor PIR conectado ao GPIO19, e zoador conectado ao GPIO12.

Material necesario:

- ESP32 WROOM + shield
- Sensor PIR
- Zoador
- Resistencia de 220Ω
- Protoboard
- Cables M-F

Pasos a seguir:

1. O esquema de conexións e os nodos en Node-Red son os das imaxes.
2. Na configuración do nodo **mqtt out,** escribir **casa/sensores/PIR** como **Topic**
3. No nodo **function** cargamos o programa en JavaScript **function_pir.js**, que fará que cando chegue a mensaxe de botón pulsado ao nodo mqtt in se acenda o LED.
4. O nodo **debug** que chamamos enviado serve para comprobar que mensaxe chega ao topic **casa/sensores/PIR**
5. Instalamos no ESP32 o programa **Practica9_PIR.ino**, desde o IDE Arduino.
6. Unha vez teñamos creado o fluxo en Node-Red, conectados os compoñentes na placa e cargado o programa no ESP32 abrimos o panel de control e comprobamos que ao pulsar o botón acende o LED e cando non o pulsamos o LED permanece apagado.


GND
3.3V
POWER
DC:6-9V
BOOT
RST
OFF
ON

10. Práctica 10. Sensor de luz.

Utilizaremos un LDR conectado a un divisor resistivo.

Todos os modelos de ESP32 teñen dispoñibles dous ADC (analog-to-dixital converter) SAR (Successive Approximation Register) de 12 bits, denominados ADC1 e ADC2.

Cunha resolución de 12bits, unha resolución de 3.3 voltios / 4096 unidades, é equivalente a 0.8 mV por paso. Ademais, podemos configurar mediante programación a resolución do ADC e o rango da canle.

A tensión máxima admisible é 3V3. Aínda que moitos GPIO do ESP32 son tolerantes a 5V, o ADC non o é. Así que non metades máis de 3.6V nun pin que usedes como entrada analóxica, ou danaredes o ADC. (Fonte: Luis Llamas)

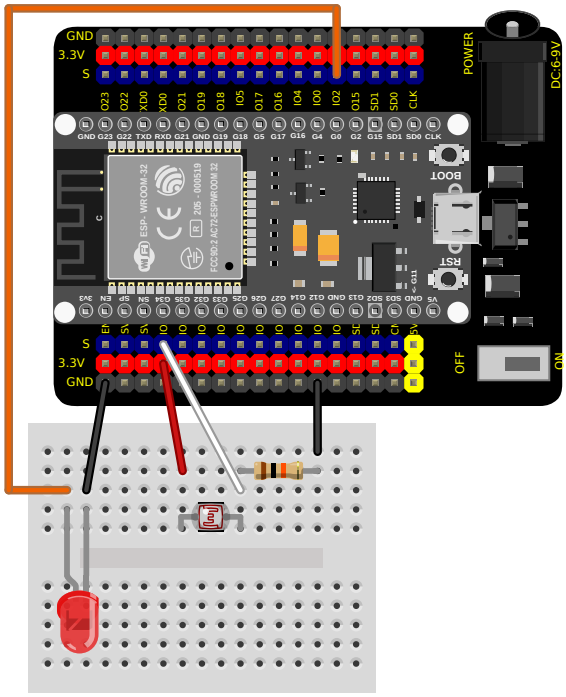
Conectamos o LDR ao GPIO 34.

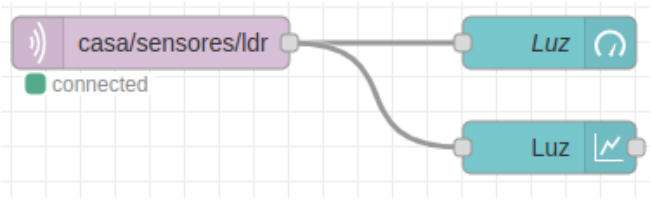
Material necesario:

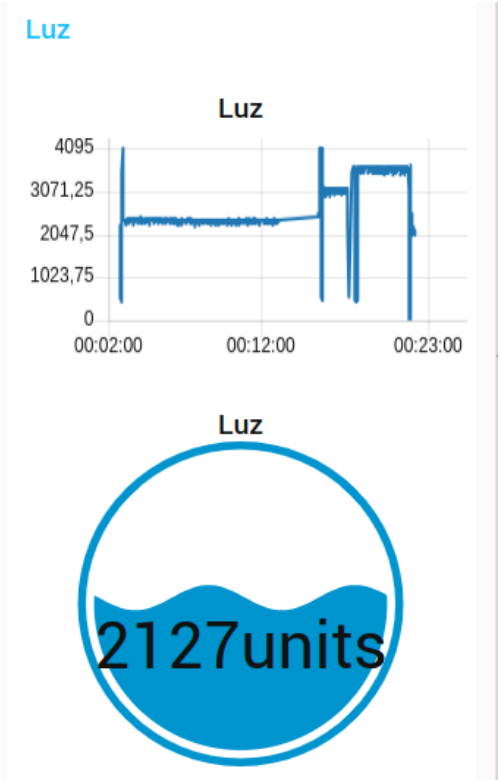
- ESP32 WROOM + shield
- LDR
- Resistencia 10kΩ
- Protoboard
- Cables M-F

Pasos a seguir:

1. O esquema de conexións e os nodos en Node-Red son os das imaxes.
2. Na configuración do nodo **mqtt out,** escribir **casa/sensores/ldr** como **Topic**
3. Configuramos un nodo **gauge** e outro **grafic**, igual que o sensor de humidade, pero neste caso indicará a luz, que será de entre 0 e 4095, xa que o conversor ADC e de 12 dixitos.
4. Instalamos no ESP32 o programa Practica10_ldr.ino, desde o IDE Arduino.
5. Unha vez teñamos creado o fluxo en Node-Red, conectados os compoñentes na placa e cargado o programa no ESP32 abrimos o panel de control e comprobamos que aparecen indicadas as variacións de luz nos indicadores gráficos.


GND
3.3V
POWER
DC6-9V
BOOT
RST
OFF
ON


casa/sensores/ldr
connected
Luz
Luz


Luz
Luz
4095
3071,25
2047,5
1023,75
0
00:02:00
00:12:00
00:23:00
Luz
2127units

11. Práctica 11. Pantalla LCD 16x02 e DHT11

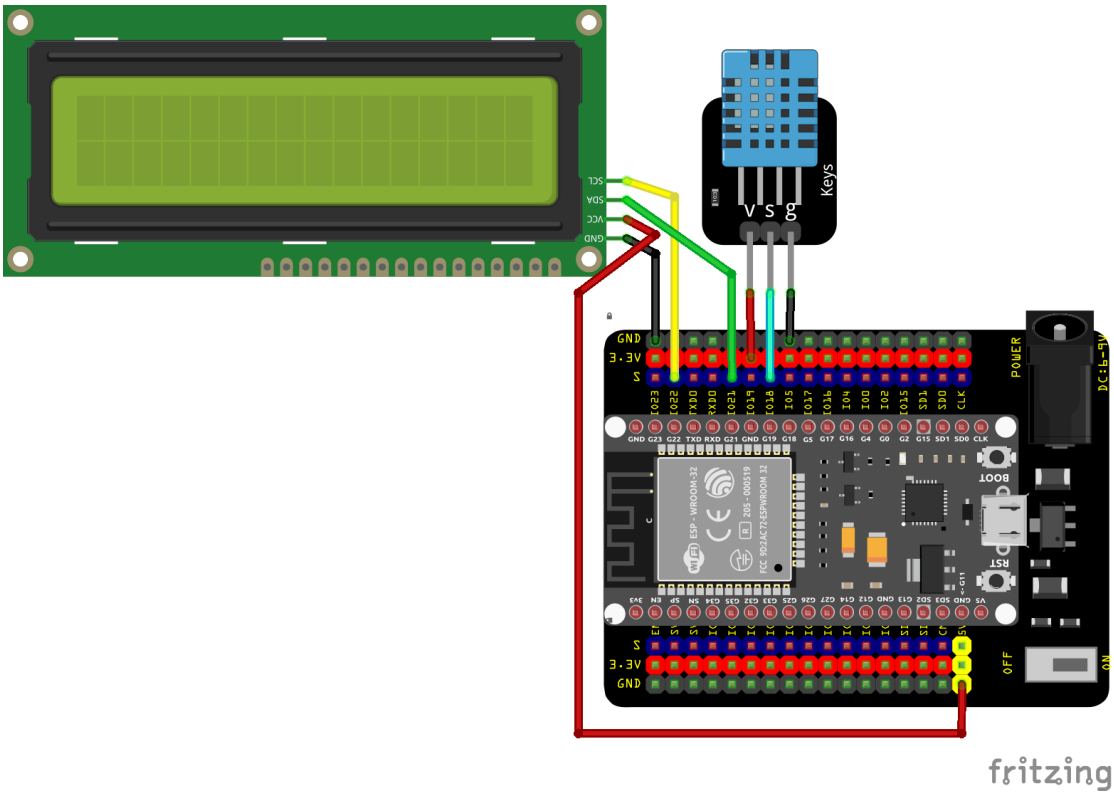
A pantalla LCD ten que ter un driver I2C para poder utilizala polo porto I2C do ESP.

Os GPIOs aos que temos que conectala son o GPIO21 (SDA) e GPIO22 (SCL), ademais de Gnd e neste caso utilizaremos un dos pines conectados a **5V** dos que dispón o shield que estamos a utilizar.

O DHT11 o conectamos igual que na práctica 7. Os nodos de Node-Red, tamén son os mesmos, non utilizamos nodos para o LCD xa que a función máis lóxica, que é a mostra dos datos do sensor xa é o que fai a LCD.

Os pasos a seguir son os mesmos que na práctica 7.

O resultado final será o da práctica 7, pero cos datos aparecendo ao mesmo tempo na pantalla LCD 16x2.



12. Práctica 11. Potenciómetro.

Esta práctica é igual que a do sensor de luz ou LDR, só teremos que cambiar o LDR por unha resistencia variable, xa que nos dous casos o que temos é un sensor analóxico que envía unha tensión entre 0 e 3,3 voltios a un GPIO ADC do ESP32. Como o conversor analóxico-dixital é de 12 díxitos, os valores que obteremos serán de entre 0 e 4095.

Non creamos outros nodos para non crear máis ruído, xa que unha vez conectado o potenciómetro e comezar a xirar para os lados o mando podes comprobar que o comportamento dos datos e igual que cando tapas e destapas o LDR.

Aquí tes o esquema de conexións dos compoñentes.