

UNIVERSIDAD AMAZÓNICA DE PANDO

ÁREA DE CIENCIAS Y TECNOLOGÍA
CARRERA DE INGENIERÍA DE SISTEMAS



“Laboratorio 13”

Proyecto final de Asignatura

Prototipo de un sistema de seguridad para un DATA CENTER

Asignatura : Seguridad de Sistema

Docente : Ing. Omar Cesar Calizaya Quiñonez

Estudiantes : Univ. Jhowill Neytan Asturizaga Lanza

Semestre: 9no semestre

Cobija - Pando - Bolivia

2023

1. INTRODUCCIÓN:

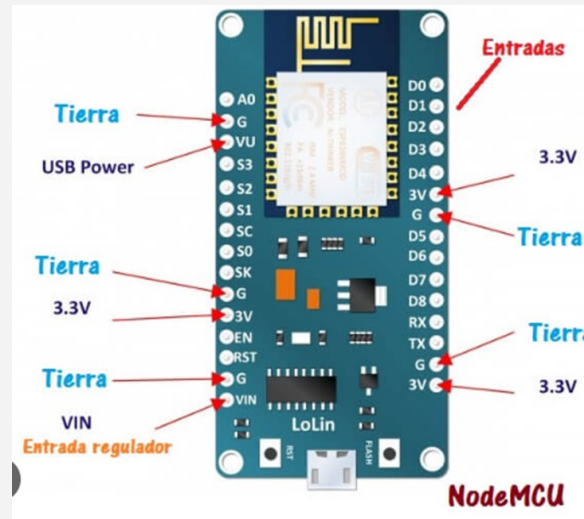
Objetivo

Desarrollar un prototipo de sistema de seguridad de dispositivos físicos para un DATA CENTER

2. Marco referencial

ESP8266

El ESP8266 es un SoC (system on chip) fabricado por la compañía china Espressif. Este SoC agrupa distintos componentes en un mismo integrado, siendo los principales un procesador de 32 bits y un chip WiFi con gestión de pila TCP/IP. (Llamas, 2018)



En resumen el ESP8266 es un chip que integra en un encapsulado un procesador de propósito general con conectividad WiFi completa.

SERVIDOR MQTT MOSQUITTO

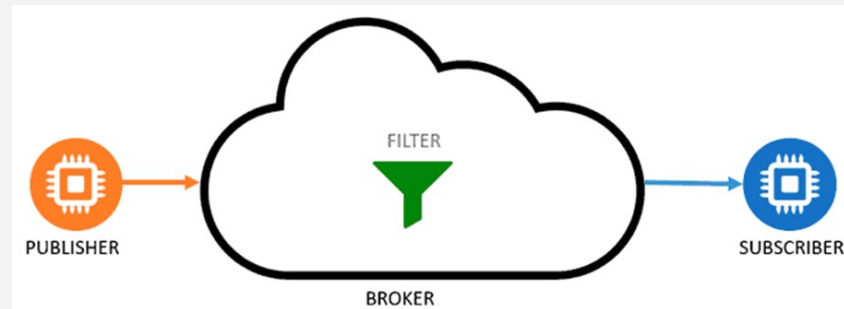
MQTT son las siglas MQ Telemetry Transport, aunque en primer lugar fue conocido como Message Queing Telemetry Transport. Es un protocolo de comunicación M2M (machine-to-machine) de tipo message queue. (Llamas, luisllamas, 2019)

Está basado en la pila TCP/IP como base para la comunicación. En el caso de MQTT cada conexión se mantiene abierta y se “reutiliza” en cada comunicación. Es una diferencia, por ejemplo, a una petición HTTP 1.0 donde cada transmisión se realiza a través de conexión.

MQTT fue creado por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom (ahora Eurotech) en 1999 como un mecanismo para conectar dispositivos empleados en la industria petrolera.

Aunque inicialmente era un formato propietario, en 2010 fue liberado y pasó a ser un estándar en 2014 según la OASIS (Organization for the Advancement of Structured Information Standards).

¿Cómo funciona el MQTT?



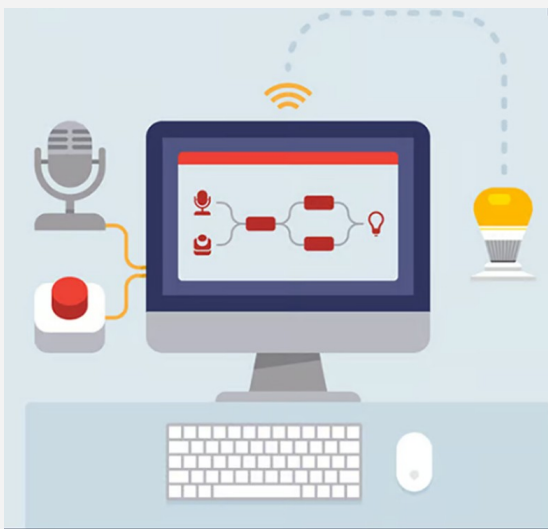
El funcionamiento del MQTT es un servicio de mensajería push con patrón publicador/suscriptor (pub-sub). Como vimos en la entrada anterior, en este tipo de infraestructuras los clientes se conectan con un servidor central denominado broker.

Para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en topics organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el broker le hará llegar los mensajes suscritos.

Los clientes inician una conexión TCP/IP con el broker, el cual mantiene un registro de los clientes conectados. Esta conexión se mantiene abierta hasta que el cliente la finaliza. Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS.

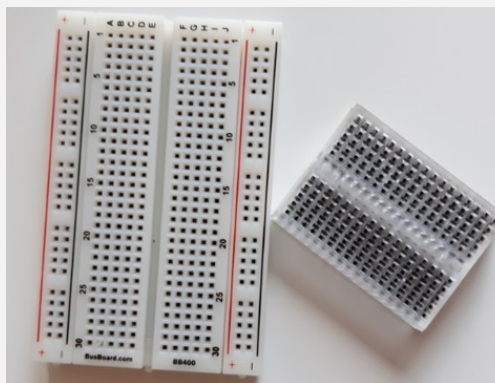
NODE-RED

Node-RED es una herramienta de programación visual. Muestra visualmente las relaciones y funciones, y permite al usuario programar sin tener que escribir una lengua. Node-RED es un editor de flujo basado en el navegador donde se puede añadir o eliminar nodos y conectarlos entre sí con el fin de hacer que se comuniquen entre ellos. (Dario, 2021)



En Node-RED, cada nodo es uno de los siguientes dos tipos: un nodo de inyección o un nodo de función. Los nodos de inyección producen un mensaje sin necesidad de entrada y lanzan el mensaje al siguiente nodo conectado a éste. Los nodos de función, por el contrario, tienen una entrada y realizan algún trabajo en él. Con una gran cantidad de estos nodos para elegir, Node-Red hace que el conectar los dispositivos de hardware, APIs y servicios en línea sea más fácil que nunca.

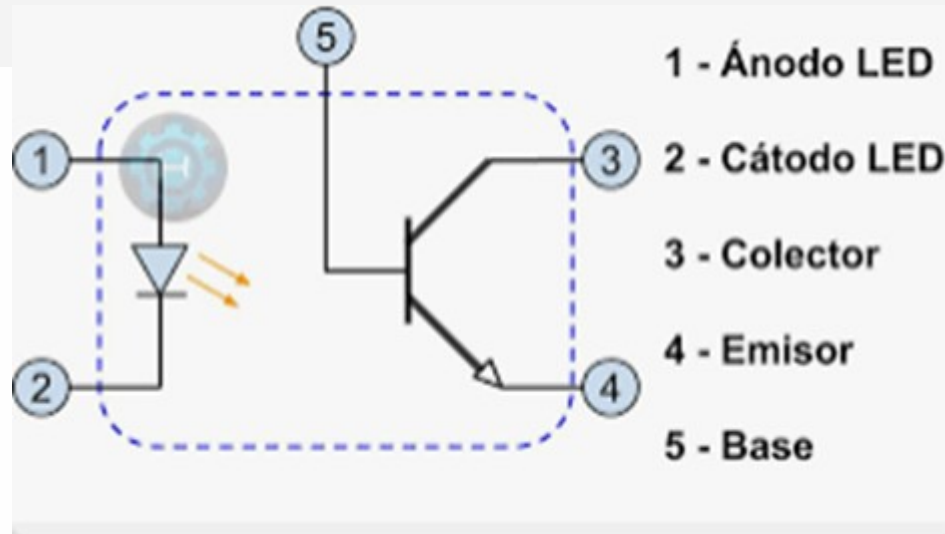
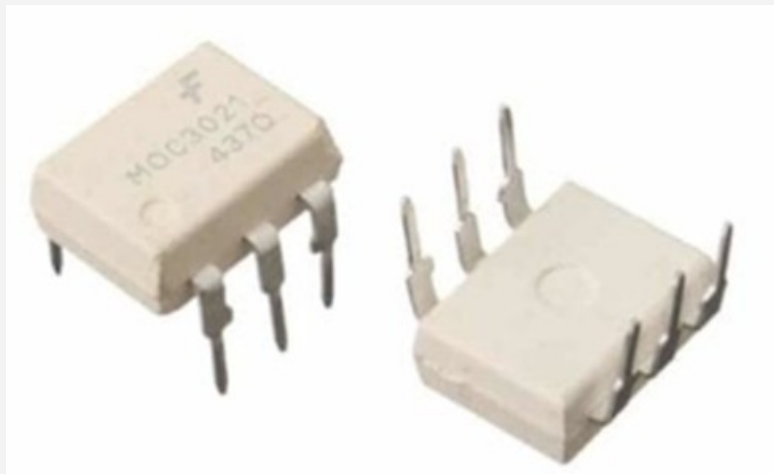
PROTOBOARD



Protoboard es una herramienta simple que se usa en proyectos de robótica que permite conectar fácilmente componentes electrónicos entre sí, sin necesidad de realizar una soldadura. Puede llamarse también breadboard o placa de pruebas. (Martínez, 2021)

MOC

Un moc es un optoacoplador. Dentro de su encapsulado tiene un led infrarrojo y un fototransistor, la finalidad de esto es aislar el circuito de control de el de carga. (Osorioen, 2009)

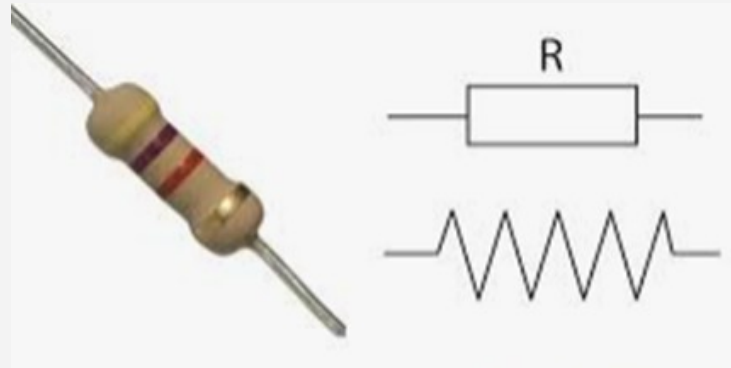


¿Qué son los optoacopladores y como funcionan?

Son conocidos como optoaisladores o dispositivos de acoplamiento óptico, basan su funcionamiento en el empleo de un haz de radiación luminosa para pasar señales de un circuito a otro sin conexión eléctrica. Estos son muy útiles cuando se utilizan por ejemplo, Microcontroladores PICs y/o PICAXE si queremos proteger nuestro microcontrolador este dispositivo es una buena opción. En general pueden sustituir los relés ya que tienen una velocidad de conmutación mayor, así como, la ausencia de rebotes. La gran ventaja de un optoacoplador reside en el aislamiento eléctrico que puede establecerse entre los circuitos de entrada y salida. Fundamentalmente este dispositivo está formado por una fuente emisora de luz, y un fotosensor de silicio, que se adapta a la sensibilidad espectral del emisor luminoso, todos estos elementos se encuentran dentro de un encapsulado que por lo general es del tipo DIP.

RESISTENCIA

La resistencia eléctrica es una escala de magnitud física que mide la tendencia de un cuerpo a oponerse al paso de una corriente eléctrica cuando se somete a una tensión eléctrica. Éste término también se utiliza para referirse un elemento de un circuito eléctrico que dificulta la circulación de las cargas eléctricas. (Planas, 2021)



¿Cómo funciona una resistencia eléctrica?

La resistencia al paso de la corriente se debe a que las cargas eléctricas (iones o electrones) que fluyen a través de un conductor eléctrico chocan contra átomos del propio conductor. Al chocar, parte de su energía cinética se convierte en calor. Es decir, uno de los efectos del paso de corriente en un conductor es su calentamiento. Este efecto se llama efecto Joule

TRIAC

El Triac de las siglas del inglés TRIodo para Alternating Current. Se define como un interruptor de CA(corriente alterna) de 3 terminales que es diferente de los otros rectificadores controlados por silicio en el sentido de que puede conducir en ambas direcciones(semiconductor bidireccional). Si la señal de puerta(Gate) aplicada es positiva(+) o negativa(-), permite el flujo de corriente. Por lo tanto, este dispositivo se puede utilizar para sistemas de CA como un interruptor. (Erick, 2021)



Este componente de 3 terminales y 4 capas que controla la energía de CA(corriente alterna). Se utilizan para conmutar y controlar la alimentación de CC(corriente continua). Hay triac en el mercado con una potencia máxima de 16 kw.

En la imagen superior se muestra el símbolo del triac, el MT1(Terminal Principal 1) y el MT2(Terminal Principal 2) conectados en paralelo inverso y un terminal puerta(Gate).

Usos del Triac

Circuitos de control.

Conmutación de lámparas de alta potencia.

Sistemas de control de energía CA.

2. Metodología empleada

primeramente necesitaremos los activo y configurado el servidor mosquito, el esp8266 con el programa en python para conectarse al broker del servidor mosquito y los siguientes componentes:

- 1 TRIAC BT139 600e
- 1 MOC3021
- 2 Resistencias de 330 ohm
- 1 Resistencia de 470 ohm
- varios cables
- 1 foco
- 1 enchufe para conectar a la corriente alterna
- 1 protoboard

configuramos el servidor mosquito con usuario, contraseña y tópico
primero vemos nuestra ip

```
3: wlx503eaa61fa14: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    link/ether 50:3e:aa:61:fa:14 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.2/24 brd 192.168.1.255 scope global dynamic noprefixroute wlx
503eaa61fa14
        valid_lft 86271sec preferred_lft 86271sec
    inet6 2800:cd0:eb09:ab00:a8a4:133f:9fa0:95ad/64 scope global temporary dynam
ic
        valid_lft 2566sec preferred_lft 2566sec
    inet6 2800:cd0:eb09:ab00:1d96:28c3:9ffb:6b66/64 scope global dynamic mngtppa
ddr noprefixroute
        valid_lft 2566sec preferred_lft 2566sec
    inet6 fe80::6194:5786:cdf7:3b52/64 scope link noprefixroute
```

configuracion del mosquito

```
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid
bind_address 192.168.1.2
acl_file /etc/mosquitto/aclfile.example
persistence true
persistence_location /var/lib/mosquitto/
password_file /etc/mosquitto/usuario_mosquito
log_dest file /var/log/mosquitto/mosquitto.log
allow_anonymous false
include_dir /etc/mosquitto/conf.d
```

usuario y contraseña y topico

```
huevito@huevito-PC:~$ cat /etc/mosquitto/aclfile.example
# This affects access control for clients with no username.
topic read $SYS/#

# This only affects clients with username "roger".
user jhowill
topic sala

# This affects all clients.
pattern write $SYS/broker/connection/%c/state
huevito@huevito-PC:~$
```

realizamos un programa en python para conectar el esp al broker de servidor mqtt


```
import network
import time
from umqtt.simple import MQTTClient
import machine
#machine.soft_reset()
#time.sleep(2)

# Configuración de la red Wi-Fi
WIFI_SSID = "Galaxy A03 Core7501"
WIFI_PASSWORD = "eqep1289"

# Configuración del servidor MQTT
MQTT_BROKER = "192.168.1.2"
MQTT_PORT = 1883
MQTT_USER = "jhowill"
MQTT_PASSWORD = "123"
MQTT_TOPIC = "sala"

# Configuración del LED
LED_PIN = 2 # GPIO 2 en el ESP8266

# Conectar a la red Wi-Fi
def connect_to_wifi():
    station = network.WLAN(network.STA_IF)
```

```
LED_PIN = 2 # GPIO 2 en el ESP8266

# Conectar a la red Wi-Fi
def connect_to_wifi():
    station = network.WLAN(network.STA_IF)
    if not station.isconnected():
        station.active(True)
        station.connect(WIFI_SSID, WIFI_PASSWORD)
        while not station.isconnected():
            pass
    print("Conectado a la red Wi-Fi:", station.ifconfig())

# Conectar al servidor MQTT
def connect_to_mqtt():
    global client # Definir la variable client en el ámbito global
    client = MQTTClient("esp8266", MQTT_BROKER, user=MQTT_USER, password=MQTT_P
    client.connect()
    print("Conectado al servidor MQTT")
    client.set_callback(on_message)
    client.subscribe(MQTT_TOPIC)
```

```
# Procesar mensajes recibidos
```

```
def on_message(topic, message):
```

```
    print("Mensaje recibido. Tópico:", topic.decode(), "Mensaje:", message.decode())
```

```
    if message.decode() == "encender":
```

```
        encender_led()
```

```
    elif message.decode() == "apagar":
```

```
        apagar_led()
```

```
    elif message.decode() == "parpadear":
```

```
        parpadear_led()
```

```
    elif message.decode() == "parar":
```

```
        parpadear_led()
```

```
# Encender el LED
```

```
def encender_led():
```

```
    led.value(1) # Encender el LED
```

```
# Apagar el LED
```

```
def apagar_led():
```

```
    led.value(0) # Apagar el LED
```

```
    led.value(0) # Apagar el LED
parpadear = False
def parpadear_led():
    global parpadear

    parpadear = True

    while parpadear:
        # Recibir y verificar el mensaje dentro del bucle

        led.value(0)
        time.sleep(0.5)

        led.value(1)
        time.sleep(0.5)

# Configurar el LED
led = machine.Pin(LED_PIN, machine.Pin.OUT)
led.value(1) # Apagar el LED inicialmente

# Conectar a la red Wi-Fi
connect_to_wifi()
```

```
        led.value(0)
        time.sleep(0.5)

        led.value(1)
        time.sleep(0.5)

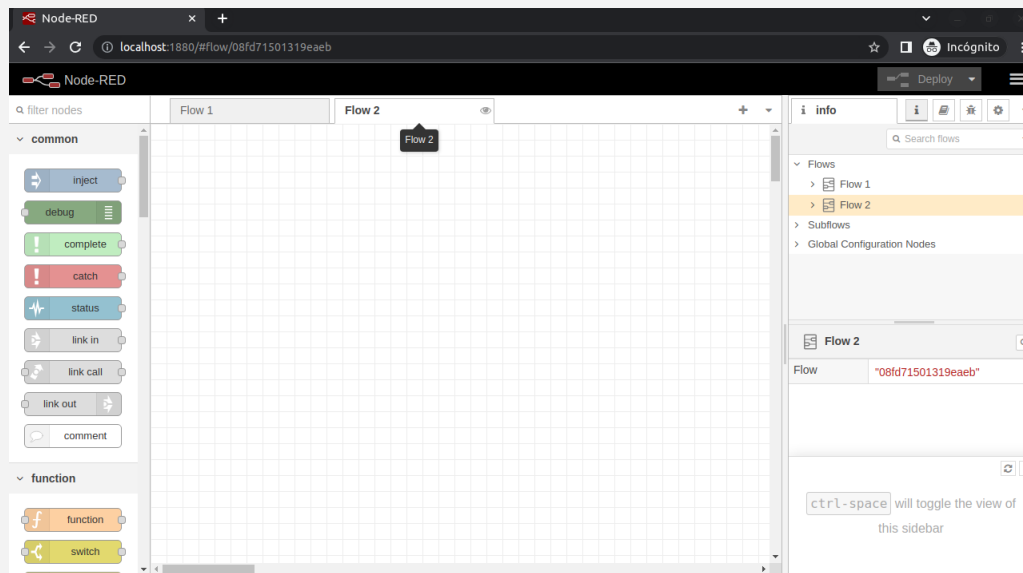
# Configurar el LED
led = machine.Pin(LED_PIN, machine.Pin.OUT)
led.value(1) # Apagar el LED inicialmente

# Conectar a la red Wi-Fi
connect_to_wifi()

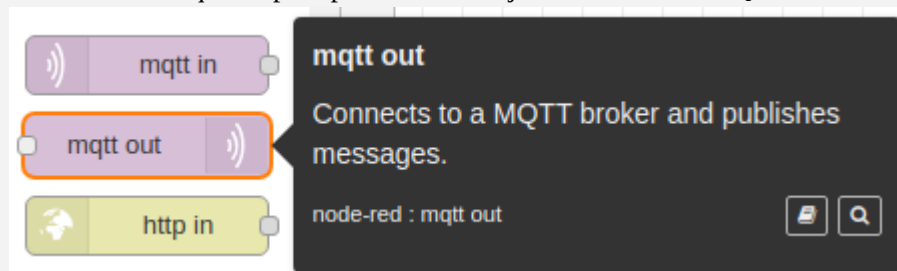
# Conectar al servidor MQTT
connect_to_mqtt()

# Bucle principal
while True:
    try:
        client.check_msg()
    except OSError:
        print("ocurrio un error...")
    time.sleep(1)
```

realizamos la interfaz en el NODE-RED



utilizamos el mqtt out para publicar mensajes en un broker MQT



y conectamos los nodos y configuramos la paleta mqtt out
configuramos el nodo mqtt out para conectar al broker del servidor mqtt

Edit mqtt out node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🖨

📶 Server

192.168.1.7:1883

✎

📄 Topic

sala

📶 QoS

🔄 Retain

📄 Name

Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

y conectamos los nodos y quedaria asi

filter nodes

Flow 1

Flow 2

Flow 3

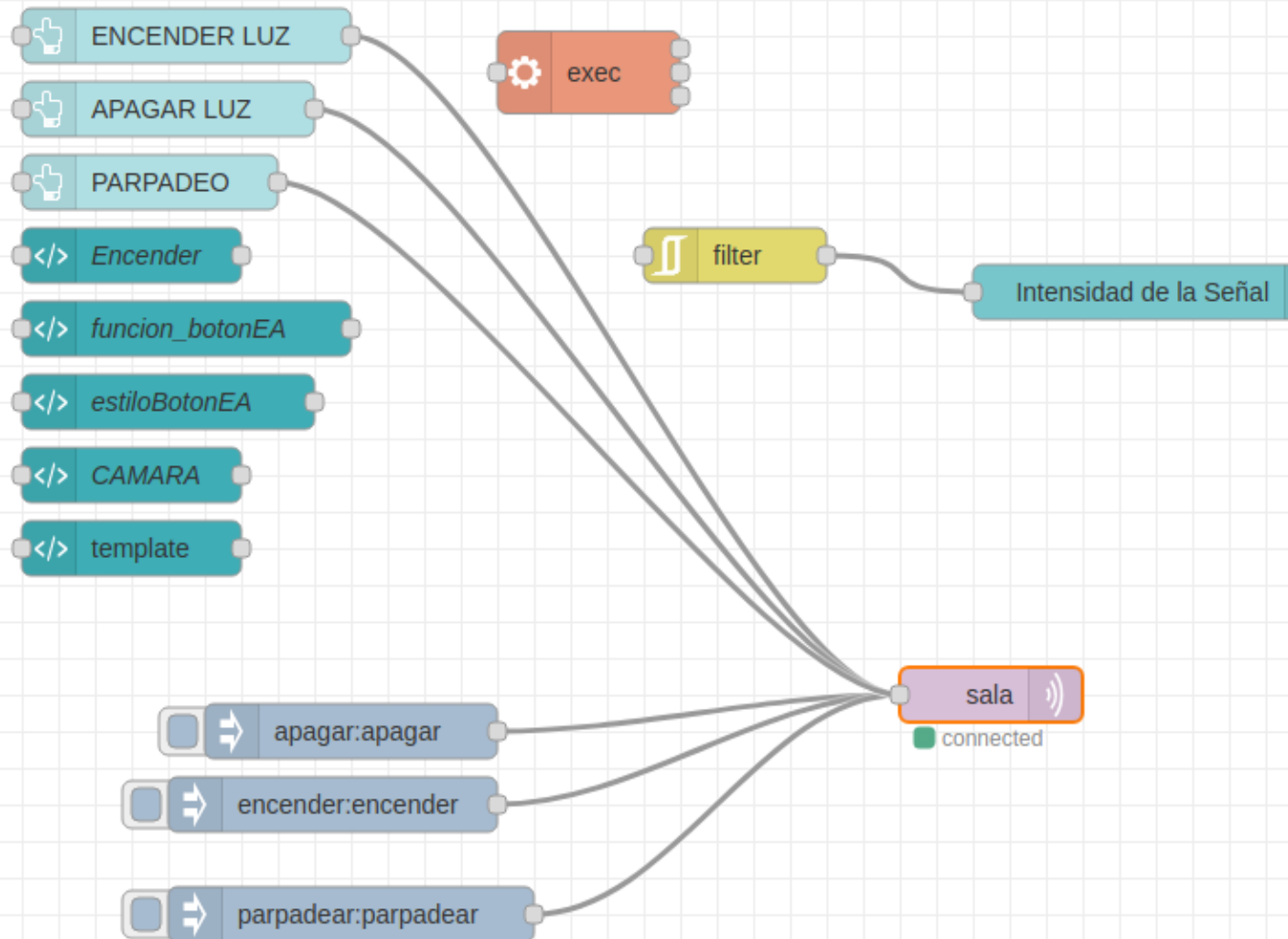


common

- inject
- debug
- complete
- catch
- status
- link in
- link call
- link out
- comment

function

- function
- switch



PANTALLA

Menu de Control

ENCENDER LUZ

APAGAR LUZ

PARPADEO

Encender

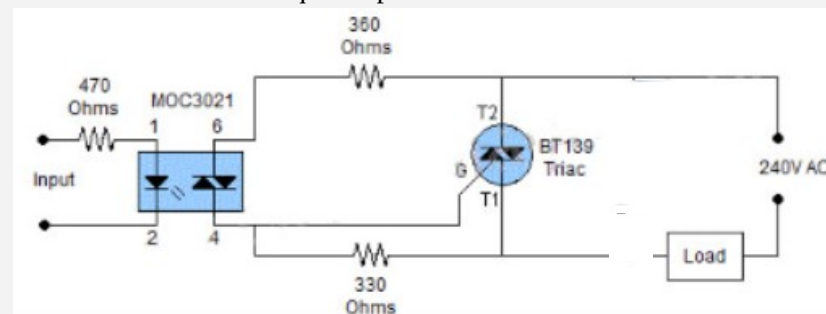
CAMARA

Informacion

Intensidad de la Señal

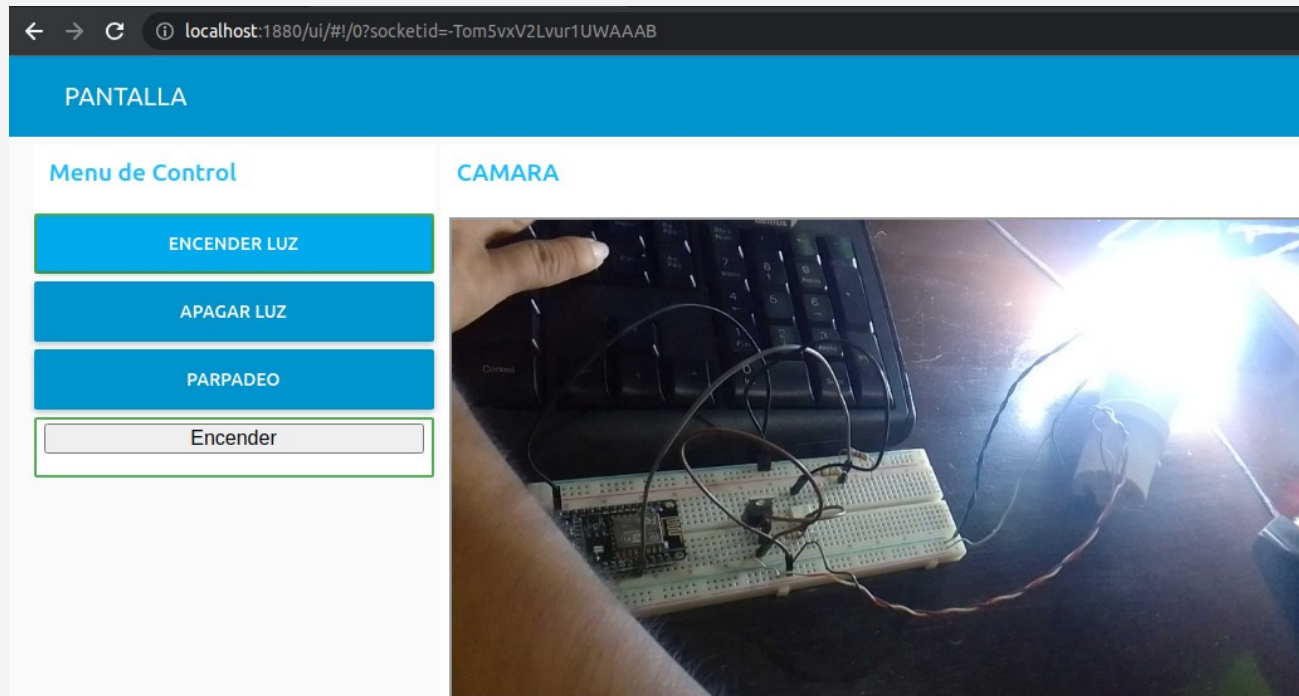
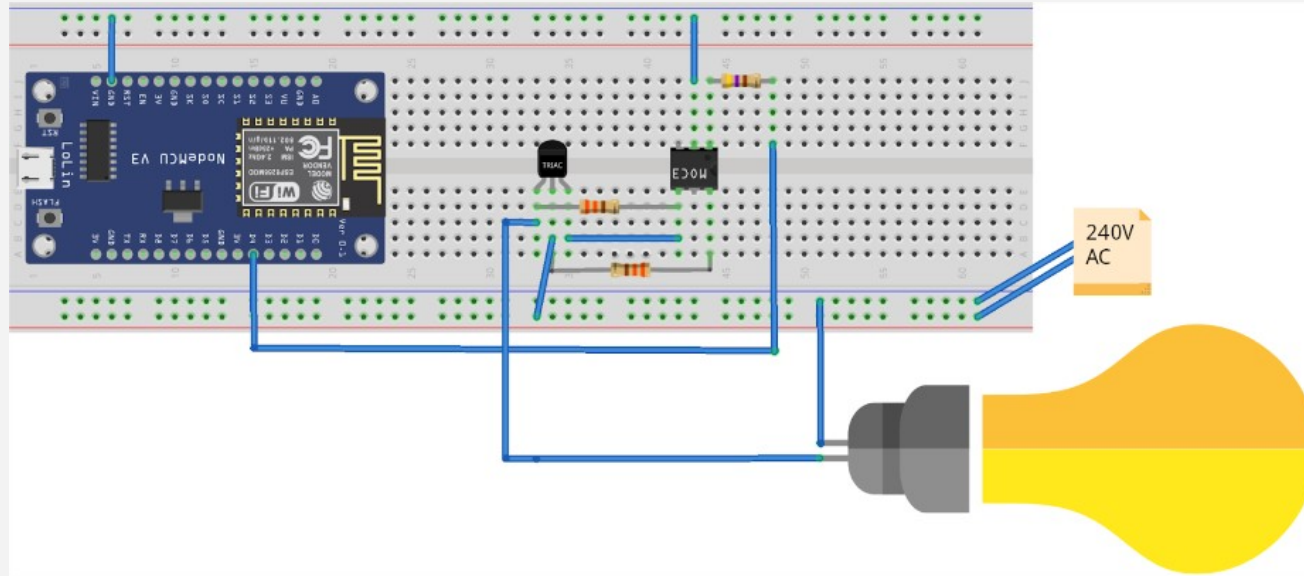


una vez configurado el servidor y la interfaz nos basamos en este esquema para la conexión de los materiales:



después hacemos nuestro propio dialout para tener un diseño más claro sobre como debemos realizar las conexiones de los componentes

aquí como se ve en este dialout realizado en Fritzing



← → ↻ ⓘ localhost:1880/ui/#!/0?socketid=-Tom5v

Captura de pantalla realizada
Puede pegar la imagen desde el portapapeles.

PANTALLA

Menu de Control

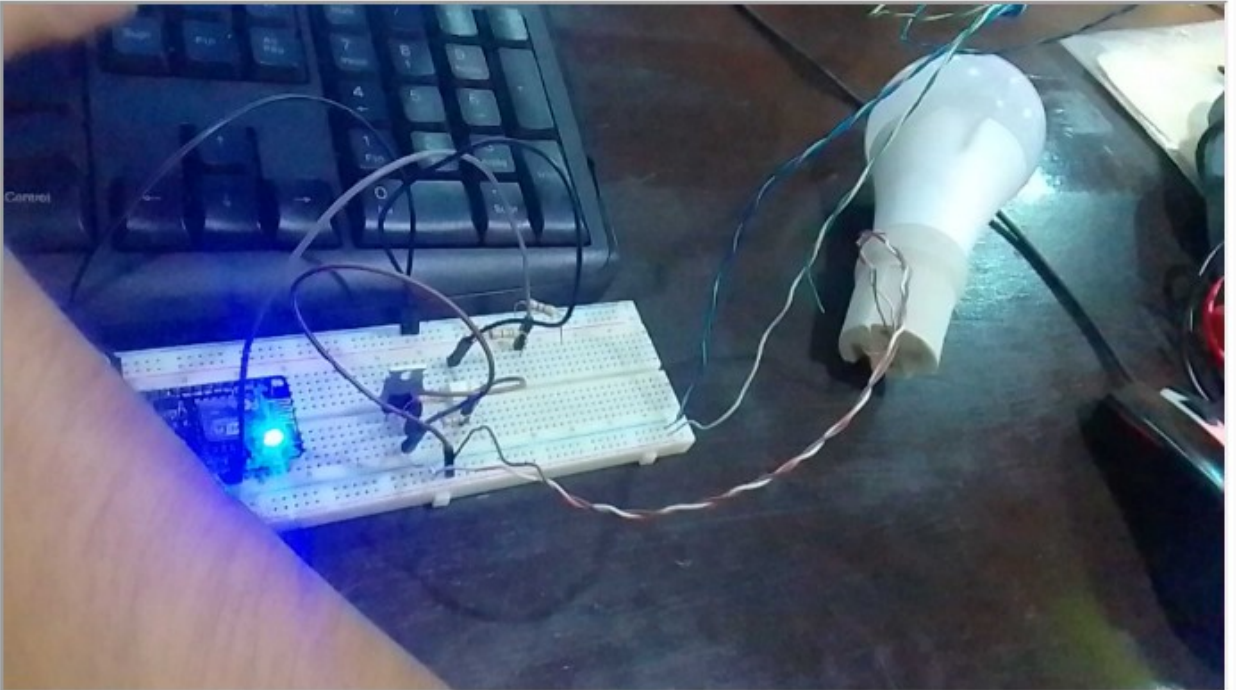
ENCENDER LUZ

APAGAR LUZ

PARPADEO

Encender

CAMARA



3. Resultados

4. Discusión y/o Conclusión

5. Referencias

- Dario, J. (2021). *toptal*. Obtenido de toptal: <https://www.toptal.com/nodejs/programacion-visual-con-node-red-conectando-el-internet-de-las-cosas-con-facilidad#:~:text=Node%2DRED%20es%20un%20editor,que%20se%20comuniquen%20entre%20ellos>.
- Erick, R. (2021). *transistores*. Obtenido de transistores: <https://transistores.info/triac-caracteristicas-y-funcionamiento/>
- Llamas, L. (2018). *luisllamas*. Obtenido de luisllamas: <https://www.luisllamas.es/esp8266/>
- Llamas, L. (2019). *luisllamas*. Obtenido de luisllamas: <https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/>
- Martínez, I. (2021). *vobusvoice*. Obtenido de vobusvoice: <https://www.vobusvoice.com/es/blog/protoboard>
- Osorioen, V. M.-Y. (2009). *moc3021*. Obtenido de moc3021: <http://moc3021.blogspot.com/>
- Planas, O. (2021). *solar-energia*. Obtenido de solar-energia: <https://solar-energia.net/electricidad/circuito-electrico/resistencia-electrica>