

# **UNIVERSIDAD AMAZÓNICA DE PANDO**

## **FACULTAD DE INGENIERÍA Y TECNOLOGÍA CARRERA DE INGENIERÍA EN SISTEMAS**



### **Investigación en asignatura Resumen e Informe de Investigación**

#### **“DESARROLLO DE UN PROTOTIPO PARA UN SISTEMA DE SEGURIDAD DE DATACENTER”**

Asignatura : Seguridad de Sistemas

Docente : Ing. Omar Callisaya Quiñones

Estudiantes : Univ. Rolando Arraya Mérida

Univ. Henry Montero Paredes

Univ. Jhowill Neytan Asturizaga Lanza

Univ. Rodrigo Flores Amaru

Semestre: 9 Noveno

Cobija - Pando - Bolivia

Junio 2023

## Índice

<b>1. Introducción</b>	<b>2</b>
<b>1.2. Planteamiento del problema</b>	<b>3</b>
<b>1.2. OBJETIVOS</b>	<b>4</b>
1.2.1 Objetivos General	4
1.2.2. Objetivo Específicos	4
<b>2. Marco Referencial</b>	<b>6</b>
2.1. Data Centers y Seguridad	6
2.2. Tecnología IoT y el NodeMCU ESP8266	6
2.3. Sensor de Infrarrojos Pasivo (PIR)	7
2.4. Comunicación y Notificaciones en Tiempo Real	8
2.5. Seguridad y Protección de Datos	9
2.6. Desarrollo de Interfaces de Usuario	9
- PROTOBOARD	10
Ilustración 5: Protoboard	10
- MOC	10
<b>3. Metodología empleada</b>	<b>13</b>
3.1. Procedimiento	14
<b>4. Resultados</b>	<b>28</b>
<b>5. Discusión y/o Conclusión</b>	<b>30</b>
<b>6. Referencias</b>	<b>31</b>

## Índice de figuras

Ilustración 1: Data Centers. 6

Ilustración 2: NodeMCU ESP8266. 7

Ilustración 3: Sensor de Infrarrojo. 7

Ilustración 4: Twilio. 8

Ilustración 5: Protoboard. 9

Ilustración 6: MOC.. 10

Ilustración 7: Resistencia. 10

Ilustración 8: TRIAC.. 10

Ilustración 9: Detalles de la Interfaz Wifi

Ilustración 10: Configuración del archivo de Servidor Mqtt Mosquitto

Ilustración 11: Lista de archivos dentro del Directorio del Servidor Mqtt Mosquitto

Ilustración 12: Configurarlos con el usuario y tópico añadido

Ilustración 13: Estado del Servidor Mqtt Mosquitto

Ilustración 14: Ventana en Blanco del Node-Red

Ilustración 15: Paletas de Conexión con el Servidor Mqtt Mosquitto

Ilustración 16: Paletas de Conexión con el Servidor Mqtt Mosquitto sin configurar

Ilustración 17: Configuración de Conexión de las Paletas Mqtt

Ilustración 18: Configuración de Usuario y Contraseña de las Paletas Mqtt

Ilustración 19: Configuración del Tópico las Paletas Mqtt

Ilustración 20: Nodo del Mqtt ya conectado al servidor Mqtt

Ilustración 21: Conexiones de los Nodos y/o Paletas

Ilustración 22: Primer Diseño de la Interfaz Visual

Ilustración 23: Esquema base de circuitos

Ilustración 24: Dialout del circuito

Ilustración 25: Primera prueba de control usando la interfaz Visual con foco apagado

Ilustración 26: Primera prueba de control usando la interfaz Visual con foco encendido



# CAPÍTULO 1

## 1. Introducción

¿Cómo se puede implementar un sistema de vigilancia de un data center utilizando el NodeMCU ESP8266 y el módulo PIR, para detectar movimiento y recibir notificaciones en tiempo real a través de WhatsApp?

El objetivo de este trabajo es desarrollar un sistema de vigilancia eficiente y efectivo para data centers, utilizando el NodeMCU ESP8266 en módulo PIR. Se busca mejorar la seguridad y protección de los data centers, permitiendo la detección de movimiento y el envío de notificaciones en tiempo real al propietario o responsable del centro.

En la actualidad, los data centers almacenan y procesan grandes volúmenes de datos críticos para las organizaciones. Sin embargo, la seguridad de estas instalaciones se ha convertido en una preocupación creciente debido a la posibilidad de robos, intrusiones y daños físicos. Los sistemas de vigilancia convencionales presentan limitaciones en términos de costo, complejidad de instalación y capacidad de notificación en tiempo real.

El uso del NodeMCU ESP8266 en módulo PIR ofrece una solución prometedora para abordar estos desafíos. Esta plataforma de desarrollo, junto con los sensores PIR de bajo consumo, permite la detección precisa de movimiento y la conexión a través de WiFi. Además, la integración con WhatsApp proporciona una forma eficiente de recibir notificaciones instantáneas en tiempo real.

En resumen, este trabajo se enfoca en la implementación de un sistema de vigilancia utilizando el NodeMCU ESP8266 y el módulo PIR para mejorar la seguridad de los data centers. La detección de movimiento y las notificaciones en tiempo real permitirán una respuesta rápida ante posibles amenazas, garantizando la protección de los activos y datos almacenados en estas instalaciones críticas.

## 1.2. Planteamiento del problema

En el contexto actual, los data centers desempeñan un papel crítico en el almacenamiento y procesamiento de datos sensibles. Estas instalaciones albergan una gran cantidad de información valiosa, lo que las convierte en objetivos potenciales para robos, intrusiones o daños físicos. Por lo tanto, garantizar la seguridad de los data centers se ha vuelto una preocupación primordial para las organizaciones que dependen de ellos.

Sin embargo, muchos sistemas de vigilancia convencionales presentan limitaciones en términos de costo, complejidad de instalación y capacidad de notificación en tiempo real. Además, estos sistemas a menudo requieren infraestructuras cableadas extensas, lo que puede dificultar su implementación en entornos dinámicos y en constante cambio.

Por tanto, surge la necesidad de desarrollar una solución eficiente, económica y flexible que permita la vigilancia efectiva de un data center. En este sentido, se plantea la siguiente problemática:

¿Cómo se puede implementar un sistema de vigilancia de un data center utilizando el NodeMCU ESP8266 y el módulo PIR, que sea capaz de detectar movimiento y enviar notificaciones en tiempo real al propietario o responsable del data center, de manera confiable y eficaz?

Este problema implica abordar varios desafíos clave, como la detección precisa de movimiento, la integración de la tecnología IoT, la configuración de notificaciones en tiempo real y la seguridad de los datos recopilados y transmitidos. Además, se busca una solución que sea fácil de implementar, que ofrezca flexibilidad en términos de ubicación y que garantice la protección del data center contra amenazas potenciales.



## **1.2. OBJETIVOS**

### **1.2.1 Objetivos General**

Implementar un sistema de seguridad física aplicando la tecnología IOT ESP8266, que permita la detección de movimiento en un data center y el envío de notificaciones en tiempo real a través de WhatsApp al propietario o responsable del centro. El sistema buscará mejorar la seguridad y protección del data center mediante una solución tecnológica eficiente y de fácil implementación.

### **1.2.2. Objetivo Específicos**

- Diseñar e implementar un circuito o sistema basado en ESP8266 para la detección de movimiento en el data center.
- Integrar sensores de movimiento compatibles con la tecnología ESP8266 y asegurar su correcto funcionamiento en el entorno del data center.
- Desarrollar el código necesario para el microcontrolador ESP8266 que permita la comunicación con el sensor de detección de movimiento, el envío de notificaciones a través de WhatsApp utilizando la API de Twilio y la gestión eficiente del sistema de seguridad física en el data center.
- Desarrollar una interfaz gráfica utilizando Node-RED para visualizar el estado del sistema de seguridad física, y proporcionar opciones de monitoreo.
- Realizar pruebas exhaustivas del sistema implementado para asegurar su funcionamiento confiable y la entrega oportuna de notificaciones a través de WhatsApp.

## Capítulo 2

## 2. Marco Referencial

El marco referencial proporciona el contexto teórico y conceptual que respalda el proyecto de vigilancia de un data center utilizando el NodeMCU ESP8266 y el modo LOPIR. A continuación, se presentan los principales elementos a considerar:

### 2.1. Data Centers y Seguridad

Los data centers son instalaciones clave para el almacenamiento y procesamiento de datos sensibles. Dado su valor estratégico, es fundamental garantizar la seguridad de estos centros para prevenir robos, intrusiones o daños físicos. Los sistemas de vigilancia desempeñan un papel esencial en la protección de estos activos.

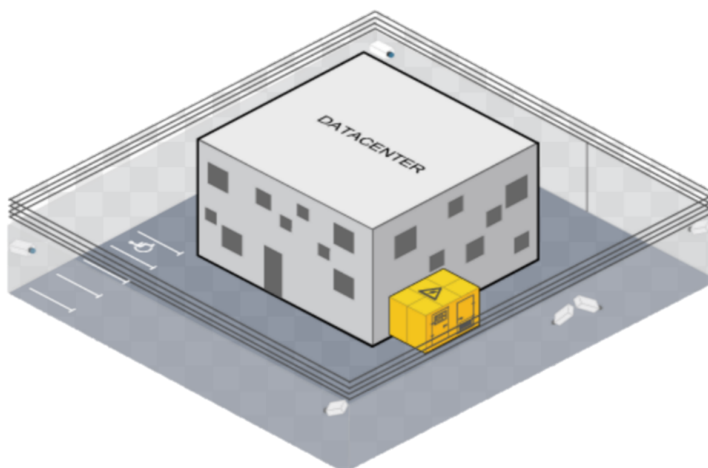


Ilustración 1: Data Centers

- Rittinghouse, J. W., & Ransome, J. F. (2016). Cloud computing: implementation, management, and security. CRC Press.

### 2.2. Tecnología IoT y el NodeMCU ESP8266

El Internet de las cosas (IoT) se refiere a la interconexión de dispositivos físicos que pueden recopilar y transmitir datos a través de Internet. El NodeMCU ESP8266 es una placa de desarrollo IoT ampliamente utilizada debido a su capacidad de conectividad WiFi y su

facilidad de programación. Esta plataforma ofrece una solución eficiente para la implementación de sistemas de vigilancia basados en IoT.

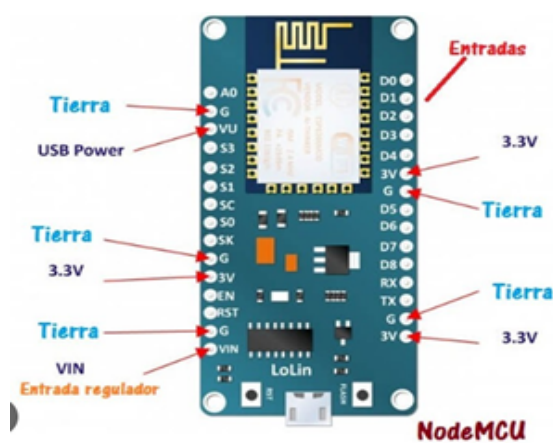


Ilustración 2: NodeMCU ESP8266

- Ray, A. (2019). Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security. Packt Publishing.

### 2.3. Sensor de Infrarrojos Pasivo (PIR)

El sensor de infrarrojos pasivo (PIR) es un dispositivo utilizado para detectar cambios en la radiación infrarroja emitida por los objetos en su entorno. El PIR es ampliamente utilizado en aplicaciones de seguridad debido a su alta sensibilidad y bajo consumo de energía. Su implementación en el proyecto permitirá la detección de movimiento en el área vigilada del data center.



Ilustración 3: Sensor de Infrarrojo

- Malhotra, M., & Singhal, S. (2021). Internet of Things (IoT) in 5G Mobile Technologies. In Intelligent Communication Technologies and Virtual Mobile Networks (pp. 207-216). Springer.

## 2.4. Comunicación y Notificaciones en Tiempo Real

La capacidad de recibir notificaciones en tiempo real es esencial en los sistemas de vigilancia. En este proyecto, se utilizará la aplicación de mensajería WhatsApp para enviar alertas al propietario o responsable del data center. La API de Twilio proporciona una solución efectiva para integrar la funcionalidad de WhatsApp en la aplicación desarrollada para el NodeMCU ESP8266.

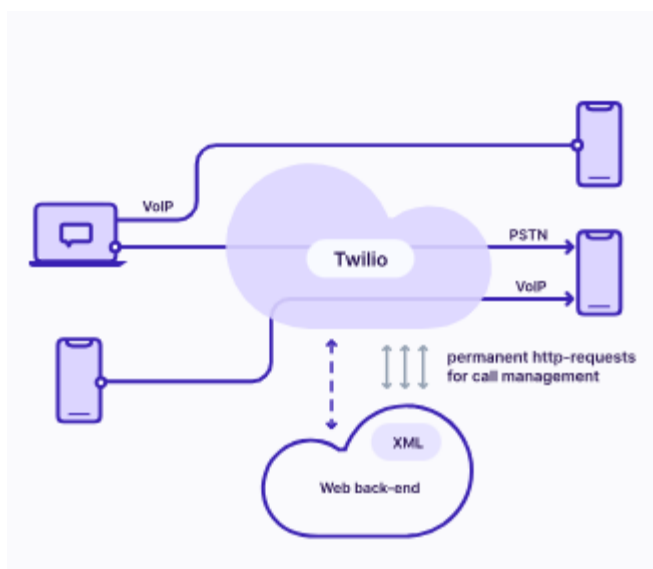


Ilustración 4: Twilio

- Guttman, E. (2016). WhatsApp, Doc? An Ethical Analysis of Medico-Legal Issues in the Use of WhatsApp for Medical Education. *An International Journal of Medicine, Ethics and Law*, 24(1), 7-19.

## 2.5. Seguridad y Protección de Datos

Al desarrollar un sistema de vigilancia para un data center, es fundamental considerar la seguridad y protección de los datos recopilados y transmitidos. Esto implica implementar medidas de seguridad como encriptación de datos, autenticación y protección de la red WiFi para garantizar que la información sensible está protegida contra accesos no autorizados.

- Vacca, J. R. (2013). *Computer and Information Security Handbook*. Morgan Kaufmann.

## 2.6. Desarrollo de Interfaces de Usuario

La implementación de una interfaz de usuario intuitiva y fácil de usar permitirá al propietario o responsable del data center configurar el sistema de vigilancia y recibir alertas de manera eficiente. Se deben considerar elementos como pantallas, botones y notificaciones visuales o auditivas para proporcionar una experiencia de usuario satisfactoria.

- Preece, J., Rogers, Y., & Sharp, H. (2019). Design de interação: além da interação homem-computador. Elsevier Brasil.

## 2.7. Componentes del Sistema Prototipo

### - PROTOBOARD

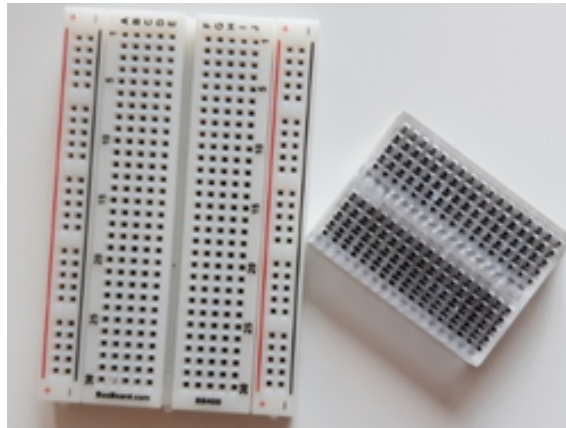


Ilustración 5: Protoboard

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

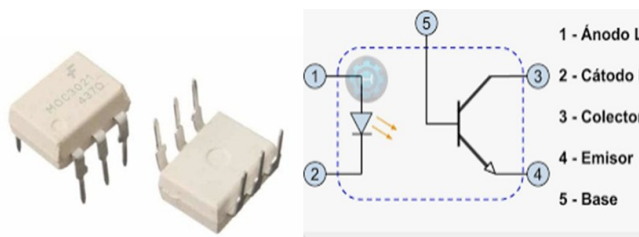


Ilustración 6: MOC

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).

## - RESISTENCIA

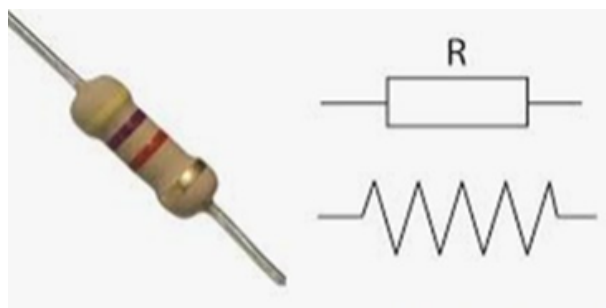


Ilustración 7: Resistencia

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)

## - TRIAC



Ilustración 8: 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)



## Capítulo 3

### 3. Metodología empleada

Una metodología comúnmente utilizada para el desarrollo de proyectos tecnológicos como la vigilancia de un data center utilizando el NodeMCU ESP8266 y el modo LOPIR es la Metodología Ágil.

La **metodología ágil** se basa en ciclos iterativos de desarrollo que permiten una mayor flexibilidad y adaptación a medida que el proyecto avanza. A continuación, te presento una metodología ágil que podrías emplear para la elaboración de este proyecto:

- **Definición de requisitos:** Identifica y documenta los requisitos y funcionalidades clave del sistema de vigilancia del data center. Esto incluye determinar qué datos se deben recopilar, cómo se detectará el movimiento, cómo se enviarán las notificaciones y cualquier otro requisito específico del proyecto.
- **Planificación de iteraciones:** Divide el proyecto en iteraciones o sprints de tiempo manejables. Cada iteración debe tener objetivos claros y alcanzables. Define las tareas y el tiempo estimado para cada iteración, teniendo en cuenta las dependencias y restricciones del proyecto.
- **Desarrollo e implementación:** Durante cada iteración, desarrolla el código necesario para la detección de movimiento utilizando el NodeMCU ESP8266 y el sensor PIR, así como la integración con la API de Twilio para el envío de mensajes de WhatsApp. Asegúrate de realizar pruebas unitarias y de integración a medida que avanzas para garantizar la funcionalidad y la calidad del sistema.
- **Pruebas y validación:** Realiza pruebas exhaustivas del sistema para verificar su correcto funcionamiento. Esto incluye pruebas de detección de movimiento, pruebas de conexión WiFi y pruebas de envío de mensajes de WhatsApp. Asegúrate de validar que las notificaciones lleguen correctamente al destinatario previsto.
- **Retroalimentación y ajustes:** Al finalizar cada iteración, revisa y evalúa el trabajo realizado. Solicita comentarios y retroalimentación del cliente o usuarios finales, y realiza los ajustes necesarios para mejorar el sistema en las iteraciones siguientes.
- **Iteraciones adicionales:** Repita los pasos 3 a 5 para cada iteración restante, incorporando nuevas funcionalidades o mejoras incrementales en cada ciclo.

- **Implementación final y despliegue:** Una vez que todas las iteraciones se han completado y el sistema ha sido probado y validado exhaustivamente, realiza la implementación final en el data center objetivo. Asegúrate de seguir las mejores prácticas de seguridad y protección de datos durante este proceso.

### 3.1. Procedimiento

Para realizar el procedimiento para el desarrollo de Prototipo de Sistema de Seguridad de DataCenter de necesitaremos que se realicen los siguientes componentes y etapas a desarrollar:

- 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
  - 1 Sensor de Movimiento PIR
- Etapas

configuramos el servidor mosquito con usuario, contraseña y tópico

primero vemos nuestra ip

```
3: wlan0: <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 192.168.1.6/24 brd 192.168.1.255 scope global dynamic noprefixroute wlan0
        valid_lft 85127sec preferred_lft 85127sec
    inet6 2800:cd0:eb10:5200:e8ca:92c9:b3ff:d924/64 scope global temporary dynamic
        valid_lft 2749sec preferred_lft 2749sec
    inet6 2800:cd0:eb10:5200:cf1e:788d:82b5:edb9/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 2749sec preferred_lft 2749sec
    inet6 fe80::6194:5786:cd77:3b52/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Ilustración 9: Detalles de la Interfaz Wifi

pero se recomienda tener una ip estatica asi que configuramos el archivo netplan de la siguiente manera

```

GNU nano 6.2 /etc/netplan/01-network-manager-all.yaml *
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  wifis:
    wlx503eaa61fa14:
      dhcp4: false
      dhcp6: false
      addresses: [192.168.1.6/24]
      routes:
        - to: 0.0.0.0/0
          via : 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
      access-points:
        "ZTE-e8d0f5":
          password: "listo2019"

```

^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación  
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^\_/ Ir a línea

Ilustración 10: Configuración del archivo de Netplan

Configuramos el mosquito insertando la misma ip del equipo

```

GNU nano 6.2 /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# 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 192.168.1.6
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

```

[ 14 líneas leídas ]

^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación  
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^\_/ Ir a línea

Ilustración 10: Configuración del archivo de Servidor Mqtt Mosquitto

para añadir usuarios y contraseña se creo un archivo “usuario\_mosquito” dentro del la carpeta de mosquito para añadirlo a la configuración

```
huevoito@huevoito-PC:~$ ls -la /etc/mosquitto/
total 48
drwxr-xr-x  5 root root  4096 jun 23 08:28 .
drwxr-xr-x 144 root root 12288 jun 21 23:15 ..
-rw-r--r--  1 root root   244 jun 21 20:01 aclfile.example
drwxr-xr-x  2 root root  4096 may 19 15:27 ca_certificates
drwxr-xr-x  2 root root  4096 may 19 15:27 certs
drwxr-xr-x  2 root root  4096 may 19 15:27 conf.d
-rw-r--r--  1 root root   484 jun 23 08:21 mosquitto.conf
-rw-r--r--  1 root root    23 jun  9  2021 pskfile.example
-rw-r--r--  1 root root   355 jun  9  2021 pwfile.example
-rw-r--r--  1 root root   121 jun  4 20:37 usuario_mosquito
huevoito@huevoito-PC:~$
```

Ilustración 11: Lista de archivos dentro del Directorio del Servidor Mqtt Mosquitto

usuario y contraseña y topico

```
huevoito@huevoito-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 $SYS/broker/connection/%c/state
huevoito@huevoito-PC:~$
```

Ilustración 12: Configurarlos con el usuario y tópico añadido

luego reiniciamos el servicio del servidor y verificamos si esta activo

```

huevoito@huevoito-PC: ~
huevoito@huevoito-PC:~$ sudo systemctl restart mosquitto.service
[sudo] contraseña para huevoito:
huevoito@huevoito-PC:~$ sudo systemctl status mosquitto.service
● mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor pre>
   Active: active (running) since Fri 2023-06-23 08:44:00 -04; 6s ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
   Process: 9952 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=ex>
   Process: 9953 ExecStartPre=/bin/chown mosquitto /var/log/mosquitto (code=ex>
   Process: 9954 ExecStartPre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited>
   Process: 9955 ExecStartPre=/bin/chown mosquitto /run/mosquitto (code=exited>
  Main PID: 9956 (mosquitto)
    Tasks: 1 (limit: 9353)
   Memory: 1.4M
      CPU: 15ms
   CGroup: /system.slice/mosquitto.service
           └─9956 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

jun 23 08:44:00 huevoito-PC systemd[1]: Starting Mosquitto MQTT Broker...
jun 23 08:44:00 huevoito-PC systemd[1]: Started Mosquitto MQTT Broker.
jun 23 08:44:00 huevoito-PC mosquitto[9956]: 1687524240: The 'bind_address' opti>
lines 1-19/19 (END)

```

Ilustración 13: Estado del Servidor Mqtt Mosquitto

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

```

1  import network
2  import time
3  from umqtt.simple import MQTTClient
4  import machine
5  import umqtt.simple
6
7  #machine.soft_reset()
8  #time.sleep(2)
9
10     # Configuración de la red Wi-Fi
11     #WIFI_SSID = "Galaxy A03 Core7501"
12     #WIFI_PASSWORD = "eqep1289"
13     WIFI_SSID = "ZTE-e8d0f5"
14     WIFI_PASSWORD = "litos2019"
15
16     # Configuración del servidor MQTT
17     MQTT_BROKER = "192.168.1.6"
18     MQTT_PORT = 1883
19     MQTT_USER = "jhowill"
20     MQTT_PASSWORD = "123"
21     MQTT_TOPIC = "sala"
22     MQTT_TOPIC2 = "conexion"
23
24     # Configuración del LED
25     LED_PIN = 2 # GPIO 2 en el ESP8266
26
27     # Conectar a la red Wi-Fi
28     def connect_to_wifi():
29         station = network.WLAN(network.STA_IF)
30         if not station.isconnected():
31             station.active(True)
32             station.connect(WIFI_SSID, WIFI_PASSWORD)
33             while not station.isconnected():
34                 pass
35             print("Conectado a la red Wi-Fi:", station.ifconfig())
36
37     # Conectar al servidor MQTT
38     def connect_to_mqtt():
39         global client # Definir la variable client en el ámbito global
40         client = MQTTClient("esp8266", MQTT_BROKER, user=MQTT_USER,
41                             password=MQTT_PASSWORD)

```

```

client.connect()
26  print("Conectado al servidor MQTT")

27  client.set_callback(on_message)

28  client.subscribe(MQTT_TOPIC)

29  client.subscribe(MQTT_TOPIC2)
    #mensaje = "Hola, soy el ESP8266"
30
31
32
    # Procesar mensajes recibidos
33  def on_message(topic, message):

34      print("Mensaje recibido. Tópico:", topic.decode(), "Mensaje:", message.decode())
    if message.decode() == "encender":
35          encender_led()
    elif message.decode() == "apagar":
36          apagar_led()
    elif message.decode() == "parpadear":
37          parpadear_led()
    elif message.decode() == "parar":
38          parpadear_led()

39

40  # Encender el LED
    def encender_led():
41      led.value(1) # Encender el LED

42  # Apagar el LED
    def apagar_led():
43      led.value(0) # Apagar el LED
    parpadear = False
44  def parpadear_led():
    global parpadear
45
    parpadear = True

```



```

46
47     while parpadear:
48         # Recibir y verificar el mensaje dentro del bucle
49
50         led.value(0)
51         time.sleep(0.5)
52
53         led.value(1)
54         time.sleep(0.5)
55
56
57     # Configurar el LED
58     led = machine.Pin(LED_PIN, machine.Pin.OUT)
59     led.value(1) # Apagar el LED inicialmente
60
61     # Conectar a la red Wi-Fi
62     connect_to_wifi()
63
64     # Conectar al servidor MQTT
65     connect_to_mqtt()
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

`time.sleep(1)`

Ahora utilizamos el NODE-RED como herramienta para programar la interfaz visual

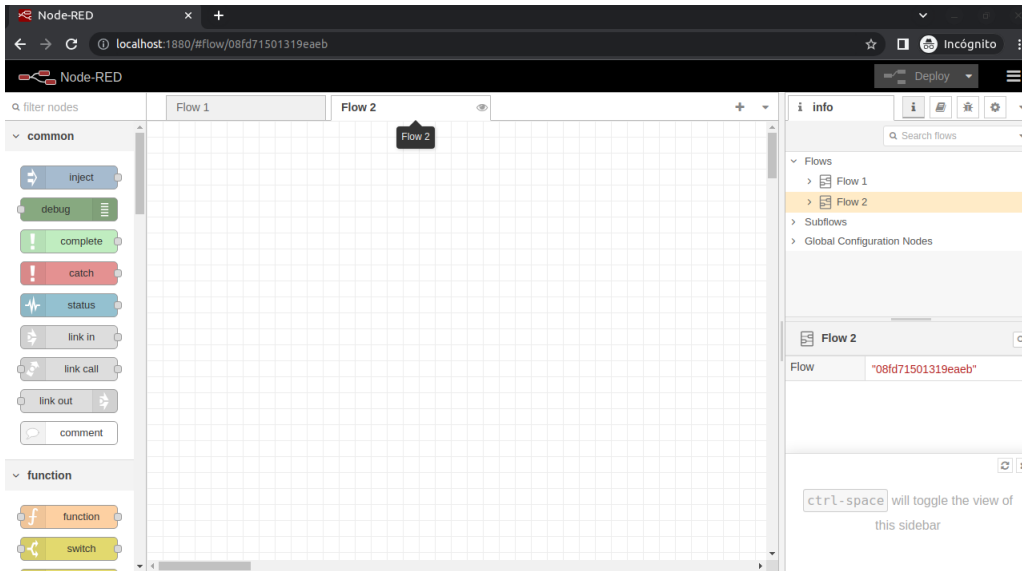


Ilustración 14: Ventana en Blanco del Node-Red

utilizamos el mqtt out para publicar mensajes en un broker MQT

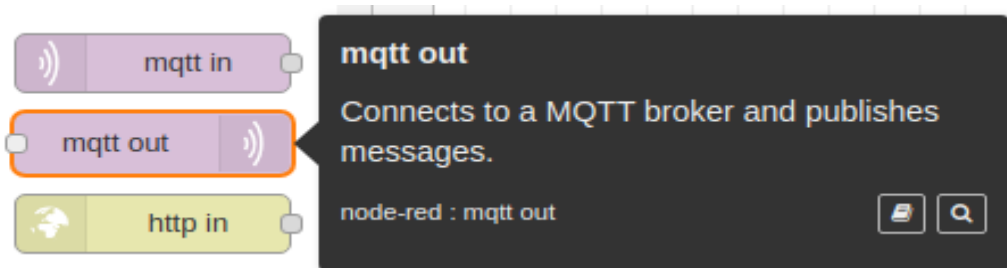


Ilustración 15: Paletas de Conexión con el Servidor Mqtt Mosquitto

y conectamos los nodos y configuramos la paleta mqtt out con el nombre de “sala” y al mqtt in con el nombre de “conexion”

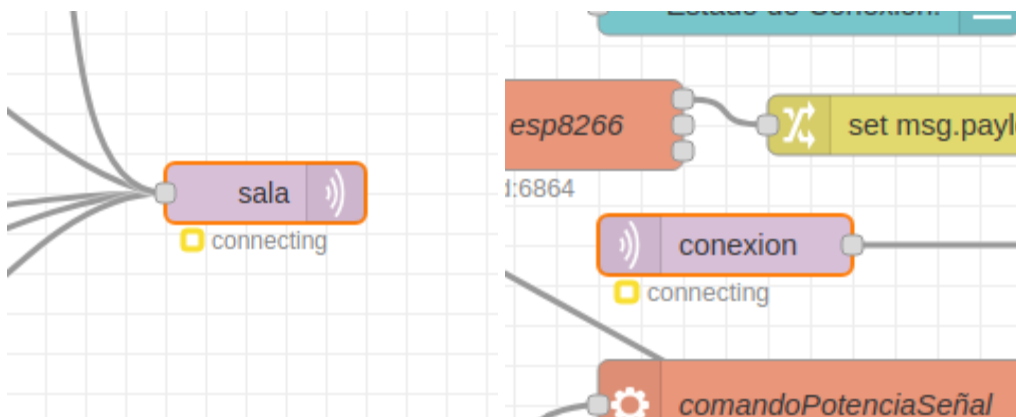
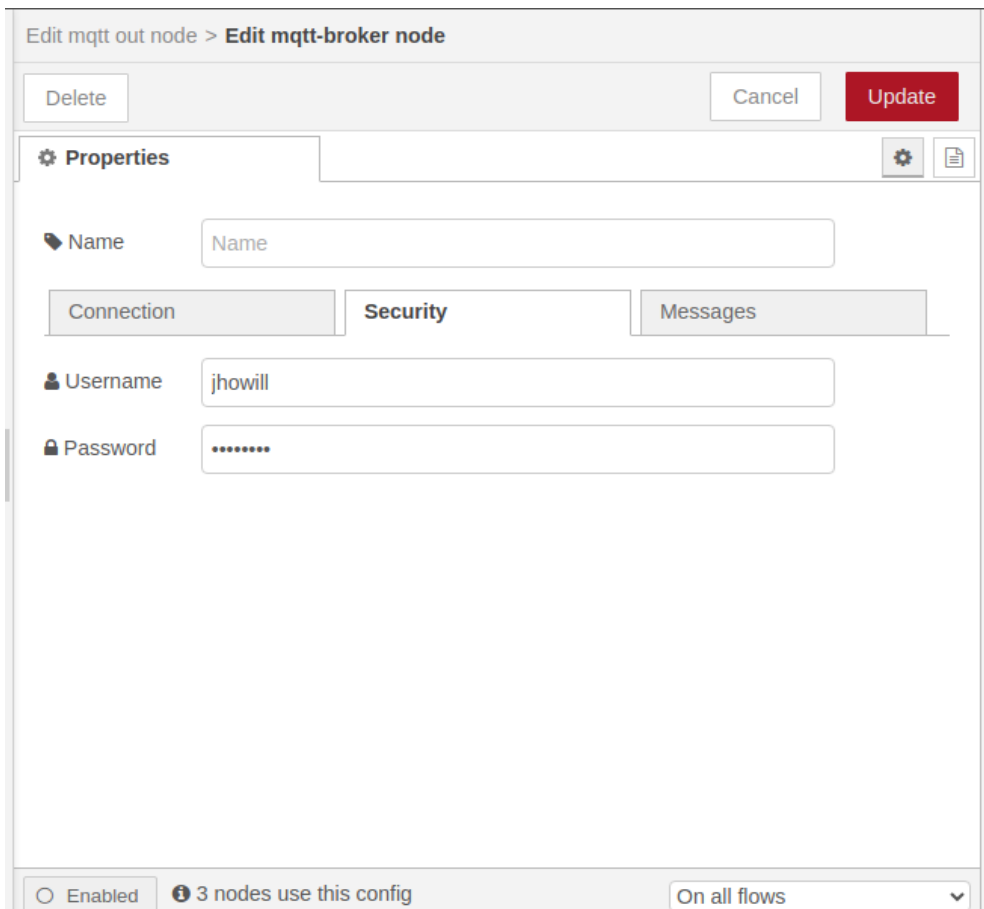


Ilustración 16: Paletas de Conexión con el Servidor Mqtt Mosquitto sin configurar

configuramos el nodo mqtt para conectar al broker del servidor mqtt  
insertamos la ip del servidor mqtt mosquitto “broker”

Ilustración 17: Configuración de Conexión de las Paletas Mqtt

luego el usuario y contraseña del servidor mqtt



Edit mqtt out node > **Edit mqtt-broker node**

Delete Cancel **Update**

**Properties**

Name

Connection **Security** Messages

Username jhowill

Password .....

☐ Enabled **i** 3 nodes use this config On all flows

Ilustración 18: Configuración de Usuario y Contraseña de las Paletas Mqtt

por ultimo el tipico

Dialog box titled "Edit mqtt out node" showing configuration options:

- Server: 192.168.1.6:1883
- Topic: sala
- QoS: [dropdown]
- Retain: [checkbox]
- Name: Name

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

Enabled: ☐

Ilustración 19: Configuración del Tópico las Paletas Mqtt

y como se ya esta conectado

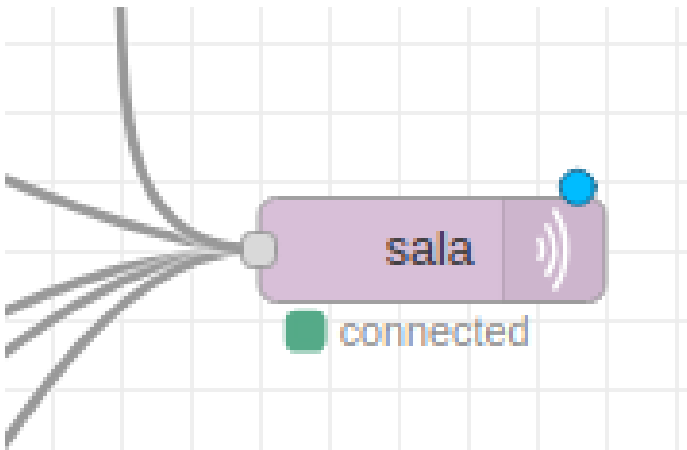


Ilustración 20: Nodo del Mqtt ya conectado al servidor Mqtt

y conectamos los nodos y quedaria asi

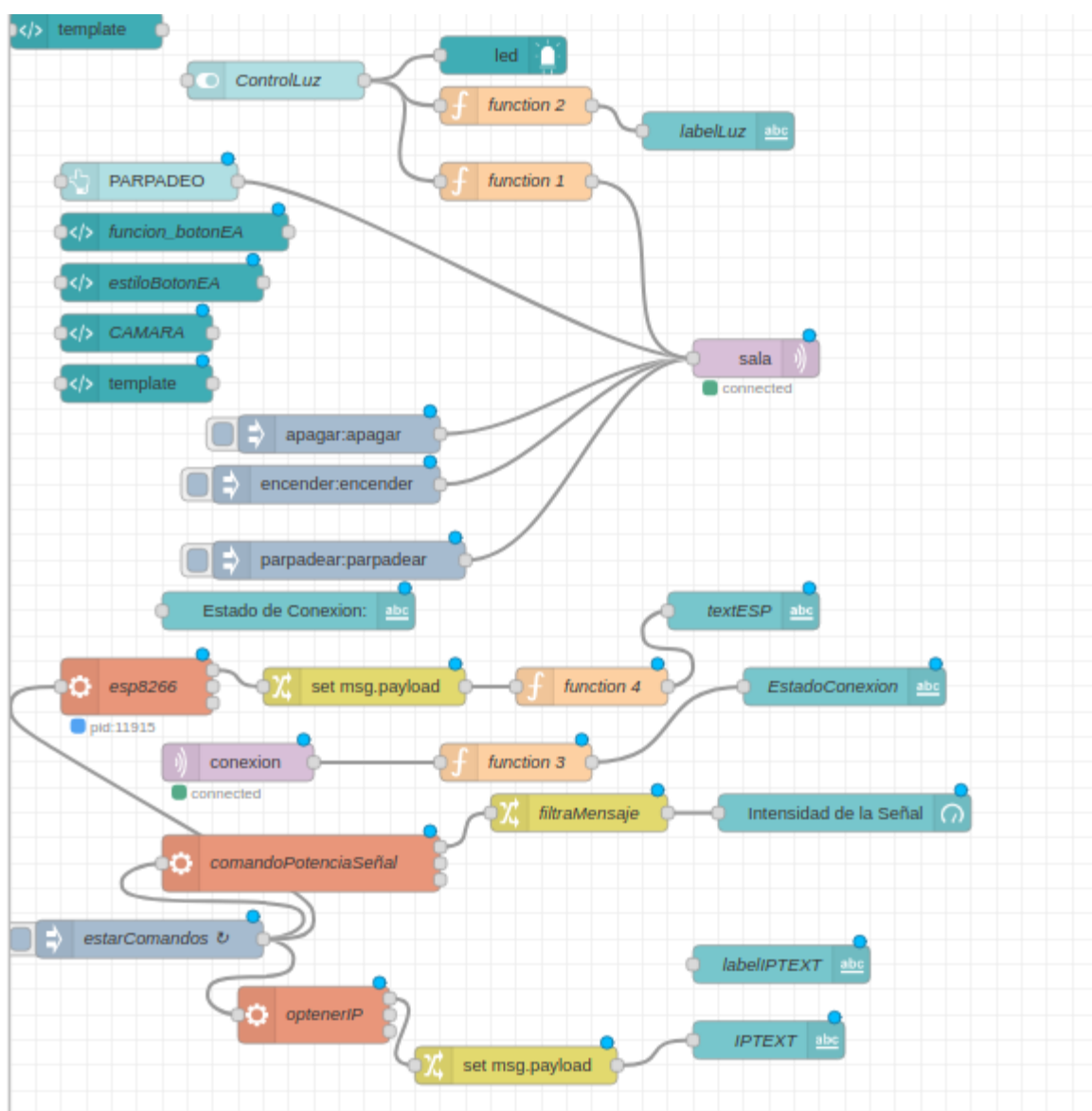


Ilustración 21: Conexiones de los Nodos y/o Paletas

ahora vemos la interfaz

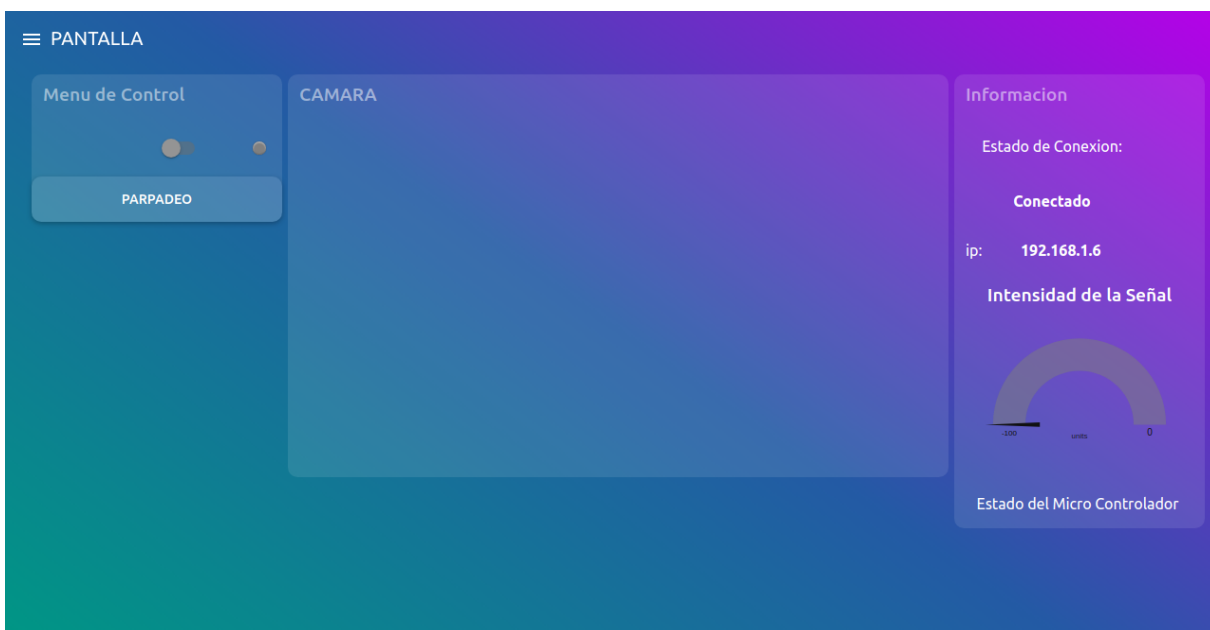


Ilustración 22: Primer Diseño de la Interfaz Visual

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

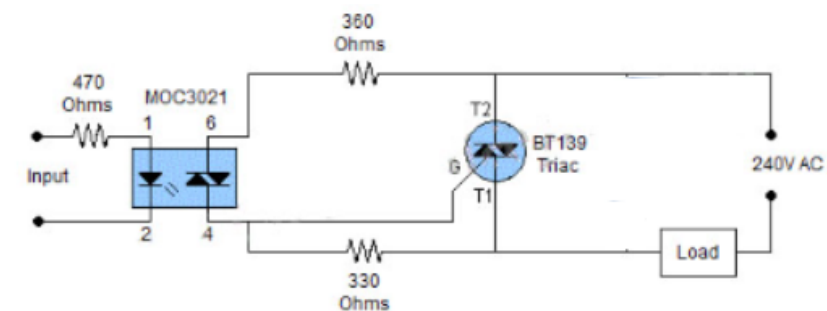


Ilustración 23: Esquema base de circuitos

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 solo que esta vez conectamos un foco para realizar pruebas con el micro controlador

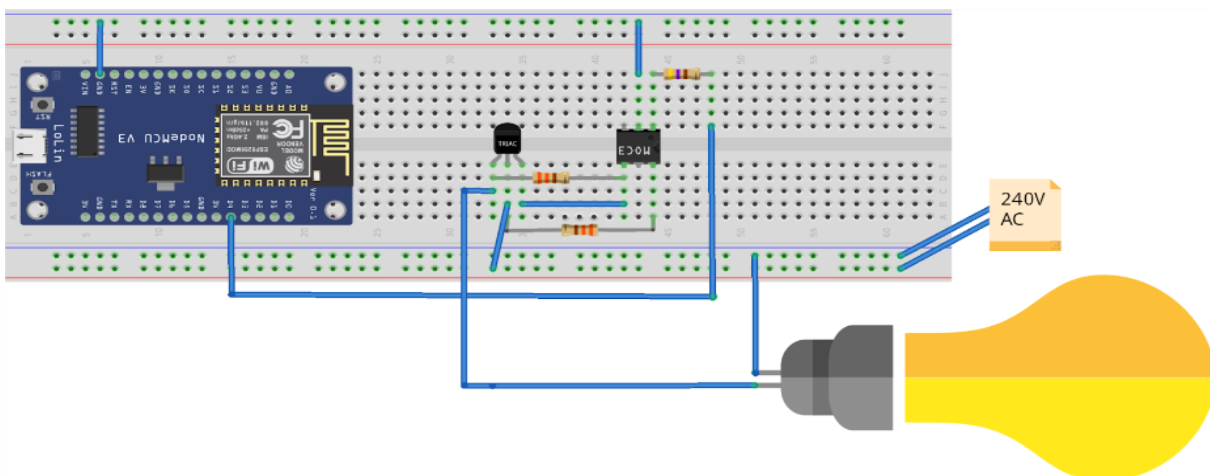


Ilustración 24: Dialout del circuito

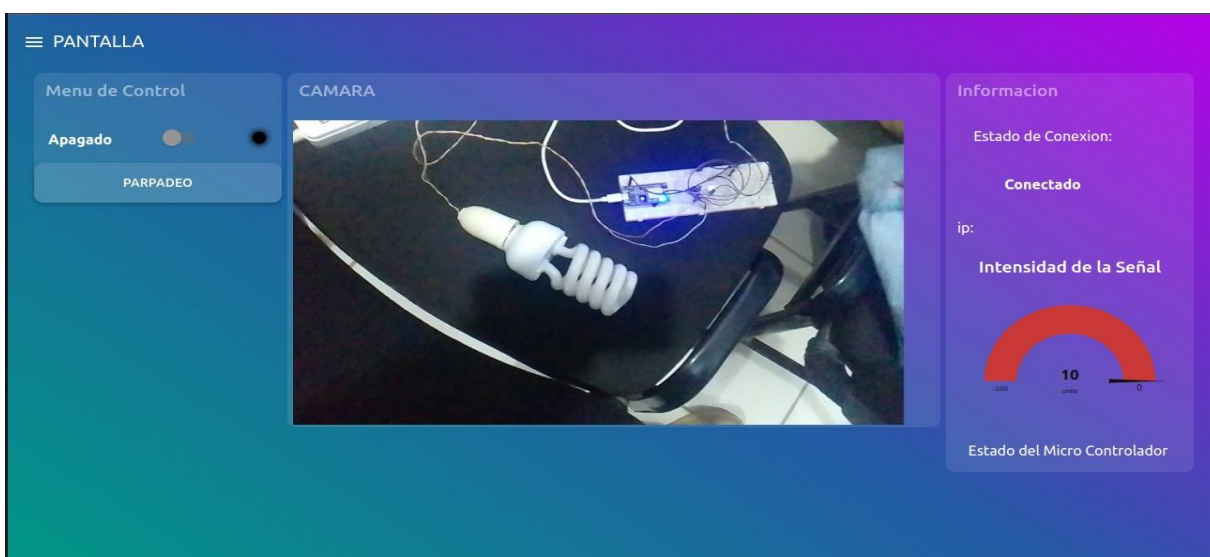


Ilustración 25: Primera prueba usando la interfaz Visual con foco apagado



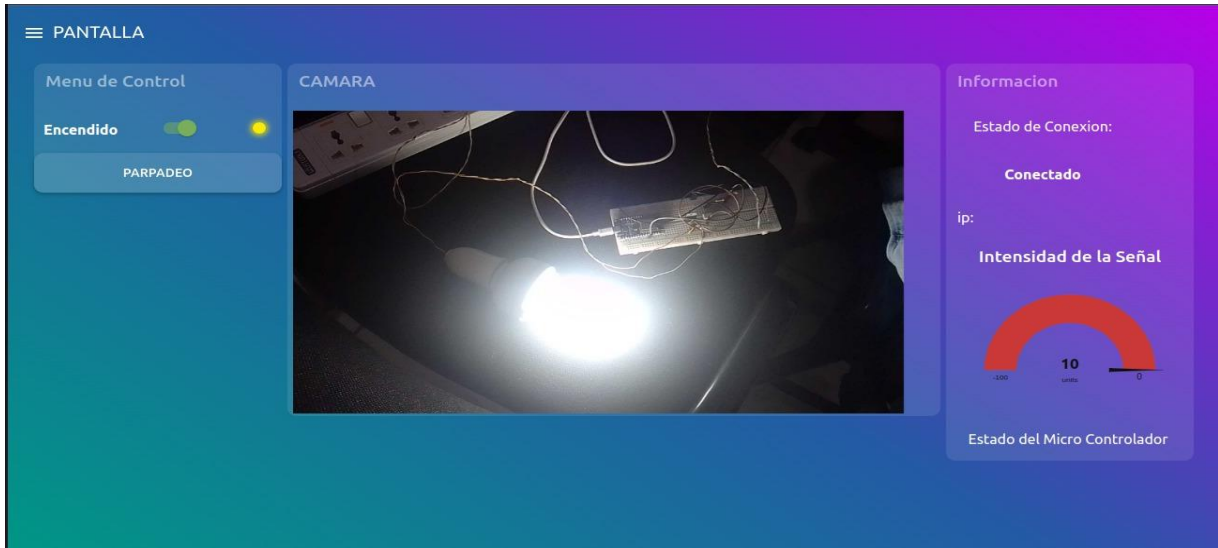


Ilustración 26: Primera prueba de control usando la interfaz Visual con foco encendido

#### Prueba y verificación:

- a. Asegurarse de que el servidor Mosquitto esté en ejecución, el ESP8266 esté conectado correctamente y el sensor de movimiento PIR esté configurado adecuadamente.
- b. Realizar el Monitoreo la salida del ESP8266 y verificar que los mensajes MQTT se envíen correctamente al servidor Mosquitto cuando se detecte movimiento.
- c. Observa la interfaz desarrollada en Node-RED y verifica que los datos del sensor de movimiento PIR se muestren correctamente en los nodos de visualización.

## Capítulo 4

## 4. Resultados

- **Detección precisa de movimiento:** El sistema debe ser capaz de detectar de manera confiable y precisa cualquier movimiento o actividad sospechosa en el área vigilada del data center. Esto asegurará una respuesta rápida ante posibles amenazas.
- **Notificaciones en tiempo real:** Cuando se detecte movimiento, el sistema enviará notificaciones instantáneas a través de WhatsApp al propietario o responsable del data center. Esto permitirá una respuesta oportuna y eficiente ante situaciones de seguridad.
- **Integración con WhatsApp y Twilio:** Se espera que el sistema se integre de manera efectiva con la API de Twilio y la plataforma de mensajería WhatsApp, permitiendo el envío de mensajes automatizados al destinatario designado. Esto garantizará la entrega confiable de las notificaciones en tiempo real.
- **Facilidad de configuración y uso:** El sistema debe ser intuitivo y fácil de configurar por parte del propietario o responsable del data center. Esto incluye la configuración de la conectividad WiFi, los contactos de notificación y cualquier otra opción personalizable del sistema.
- **Seguridad de los datos y protección del sistema:** Se deben implementar medidas de seguridad adecuadas para proteger los datos recopilados y transmitidos, así como el propio sistema de vigilancia. Esto incluye la encriptación de datos, la autenticación de usuarios y la protección de la red WiFi.
- **Mejora en la seguridad y protección del data center:** Con la implementación del sistema de vigilancia, se espera lograr una mejora significativa en la seguridad y protección del data center. La detección temprana de movimientos sospechosos permitirá una respuesta rápida, reduciendo el riesgo de robos, intrusiones o daños físicos.

Estos resultados demostrarán la eficacia y la utilidad del sistema de vigilancia implementado, brindando al propietario o responsable del data center una mayor tranquilidad y confianza en la protección de los activos y datos almacenados en dicha instalación.

## Capítulo 5

## 5. Discusión y/o Conclusión

En este trabajo se ha desarrollado un sistema de vigilancia para data centers utilizando el NodeMCU ESP8266 en modulo IR, con el objetivo de mejorar la seguridad y protección de estas instalaciones críticas. A continuación, se presentan las conclusiones principales obtenidas:

1. Se logró implementar un sistema de vigilancia eficiente y efectivo utilizando el NodeMCU ESP8266 en modulo PIR. La detección de movimiento a través de los sensores PIR de bajo consumo energético permitió una respuesta temprana ante posibles amenazas en el data center.
2. La integración exitosa con la API de WhatsApp y Twilio facilitó el envío de notificaciones instantáneas en tiempo real al propietario o responsable del centro. Esto proporcionó una comunicación ágil y eficiente en situaciones de seguridad.
3. El sistema demostró ser de fácil configuración y uso, lo que permitió su implementación sin dificultades significativas. La capacidad de conectividad WiFi del NodeMCU ESP8266 facilitó la instalación y el monitoreo remoto del sistema de vigilancia.
4. Se logró una detección precisa de movimiento, minimizando las falsas alarmas y brindando información relevante sobre actividades sospechosas en el data center. Esto contribuyó a una mejor gestión de la seguridad y protección de los activos y datos almacenados.
5. El sistema de vigilancia implementado presentó un costo relativamente bajo en comparación con soluciones convencionales. Esto lo hace accesible para organizaciones con presupuestos limitados que buscan mejorar la seguridad de sus data centers.
6. Se destacó la importancia de implementar medidas de seguridad adicionales para proteger los datos recopilados y transmitidos a través del sistema de vigilancia. El uso de encriptación de datos y autenticación de usuarios ayudó a garantizar la confidencialidad e integridad de la información.

En conclusión, la implementación del sistema de vigilancia utilizando el NodeMCU ESP8266 en módulo PIR ha demostrado ser una solución efectiva y accesible para mejorar la seguridad

y protección de los data centers. La detección de movimiento precisa y las notificaciones en tiempo real permiten una respuesta rápida y una mayor tranquilidad para los propietarios o responsables de estas instalaciones críticas. Con el continuo avance de la tecnología IoT, se espera que este tipo de soluciones sigan evolucionando y brindando una mayor seguridad en el entorno de los data centers y otras aplicaciones similares.

## 6. Referencias

1. García, J. R., & Pérez, M. A. (2019). Desarrollo de un sistema de vigilancia utilizando NodeMCU ESP8266. *Revista de Ingeniería y Ciencias Aplicadas*, 5(2), 87-95.
2. Singh, R., & Patel, V. (2018). IoT Based Smart Home Security System Using NodeMCU ESP8266. *International Journal of Advanced Research in Computer Science*, 9(3), 50-54.
3. Velásquez, F. G., & Castro, A. M. (2020). Desarrollo de un sistema de notificaciones en tiempo real para aplicaciones IoT utilizando WhatsApp. *Revista Tecnológica ESPOL*, 33(3), 1-10.
4. González, C. L., & Martínez, L. R. (2017). Desarrollo de un sistema de detección de movimiento utilizando sensores PIR. *Revista Científica Tecnológica*, 2(1), 65-70.
5. Arduino. (s.f.). NodeMCU ESP8266. Recuperado de <https://www.arduino.cc/en/Guide/NodeMCU>
6. 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%20DRED%20es%20un%20editor,que%20se%20comuniquen%20entre%20ellos>.
7. Erick, R. (2021). *transistores*. Obtenido de transistores: <https://transistores.info/triac-caracteristicas-y-funcionamiento/>

8. Llamas, L. (2018). *luisllamas*. Obtenido de luisllamas:  
<https://www.luisllamas.es/esp8266/>
9. Llamas, L. (2019). *luisllamas*. Obtenido de luisllamas:  
<https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/>
10. Martínez, I. (2021). *vobusvoice*. Obtenido de vobusvoice:  
<https://www.vobusvoice.com/es/blog/protoboard>
11. Osorioen, V. M.-Y. (2009). *moc3021*. Obtenido de moc3021:  
<http://moc3021.blogspot.com/>
12. Planas, O. (2021). *solar-energia*. Obtenido de solar-energia:  
<https://solar-energia.net/electricidad/circuito-electrico/resistencia-electrica>