



# Proyecto Farola Inteligente

Sistemas Empotrados y Ubicuos

Autores: Evelin Flores  
Mohammed el Boutahiri  
Tarik Imlahi Rivas

## I) Descripción del Proyecto Básico

Para la parte de IoT(Internet of Things) se pretende fabricar una “Farola Inteligente” que pueda ser monitorizada y regulada a través de Internet. Dicha farola dispondrá como característica básica el incrementar la intensidad de luz en función del nivel de oscuridad presente en el ambiente, es decir, cuanta menos luz haya en la calle, más luz desprenderá el foco de la farola.

Los datos de intensidad se reflejarán de forma automática a través de un cuadro de mandos en forma de gráficas. A su vez, estos datos quedarán reflejados en un histórico.

## II) Descripción del Proyecto Avanzado

Nuestra Farola podrá hacer lo siguiente:

- Medir la luz ambiental
- Regular la intensidad del foco en función de la luz ambiental
- Enviar dichos datos a internet

Por otro lado, nuestra aplicación de monitorización podrá:

- Mostrar Gráficas de Niveles de Luminosidad e histórico.
- Mostrar Voltajes de Sensores.
- Enviar emails a los Técnicos de Mantenimiento en respuesta de eventos inesperados del sistema (sensores rotos, luminosidad extrema)
- Avisar de dichos eventos por medio de mensajes de voz automatizados
- Regular la luminosidad manualmente de una Segunda Farola para comprobar que la comunicación llegaría a ser bidireccional

## III) Requisitos Necesarios

Por un lado, se necesitan los siguientes materiales para el desarrollo:

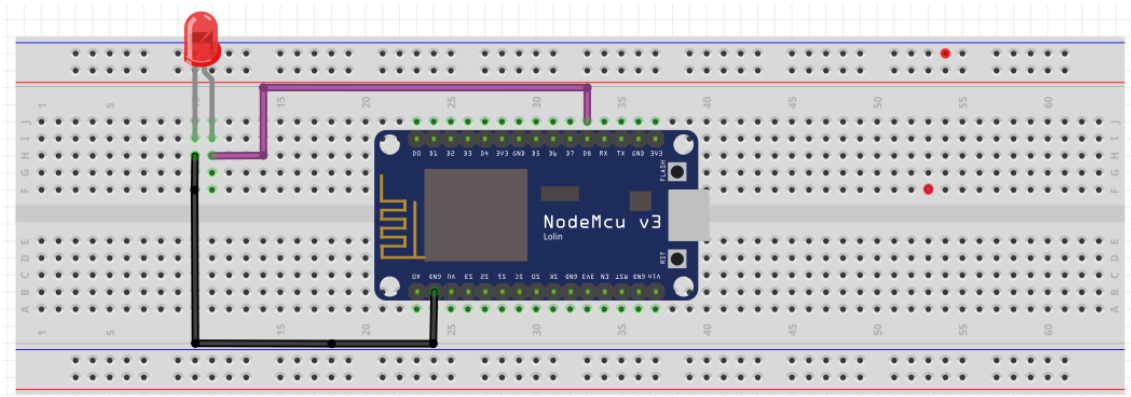
- 2 x NodeMCUv3 (Uno para una farola regulada automáticamente y otro para una regulada manualmente).
- 2 x LED.
- 1 x KOhm.
- 2 x Cables MiniUsb-USB.
- 1 x Sensor de Luminosidad.
- 1 x ProtoBoard y varios cables macho para conectar pines.

Y por otro, utilizaremos los siguientes Software:

- **Fritzing** para el diseño y presentación de nuestro circuito
- **ARDUINO Versión: v1.8.2** para programar nuestros NodeMCU
- **Node-RED Versión: v0.16.2** para diseñar e implementar nuestro Dashboard de
- **HiveMQTT** para la comunicación de nuestros dispositivos físicos y nuestra aplicación de monitorización

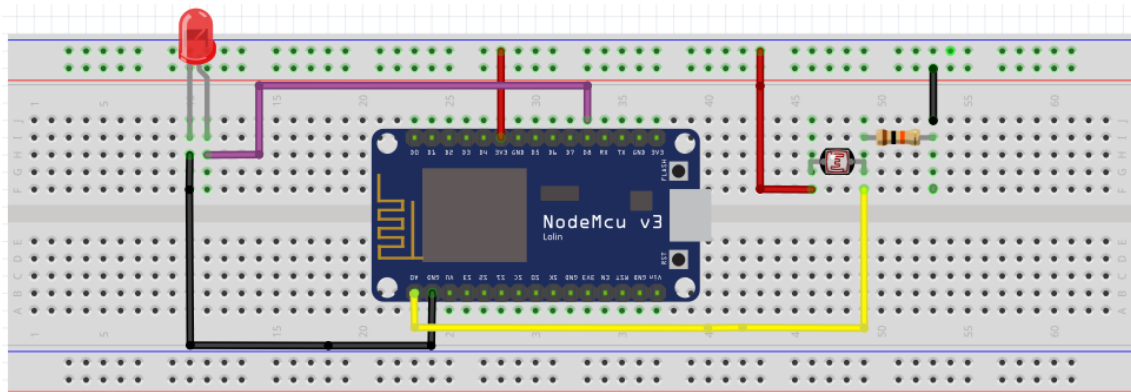
## IV) Diseño del Circuito en Fritzing.

### A) Farola Regulada Manualmente



En este primer circuito solo vamos a necesitar un LED que controlaremos manualmente desde nuestra aplicación a través de internet.

### B) Farola Regulada Automáticamente



En este segundo circuito añadiremos un sensor de luminosidad junto con una resistencia para que se regule la intensidad de la luz del LED automáticamente. Los valores de ese sensor conjunto algunos datos más serán monitorizados a través de nuestra aplicación remota.

## V) Servicio Mqtt

En este punto lo que haremos será explicar a grandes rasgos cual va a ser el servidor que nos va a brindar el intercambio de mensaje Mqtt para que nuestros dispositivos y programas puedan transmitirse información entre sí.

En este proyecto hemos usado el servidor público que se nos oferta en la página, sin embargo podemos en un momento dado crear un servidor propio gracias a su aplicación cliente.



**HIVEMQ**  
ENTERPRISE MQTT BROKER

[MQTT](#) [HIVEMQ](#) [RESOURCES](#) [EXTENSIONS](#) [SERVICES](#)

## First Steps

### New to MQTT and HiveMQ?

We have all the pieces available to prototype your MQTT use case in minutes

#### HiveMQ Public Broker

We have a public broker where you can do your first steps with MQTT. It is free of charge, already setup and ready to go.

##### Connect to Public Broker

Host: [broker.hivemq.com](https://broker.hivemq.com)

Port: **1883**

Websocket Port: **8000**

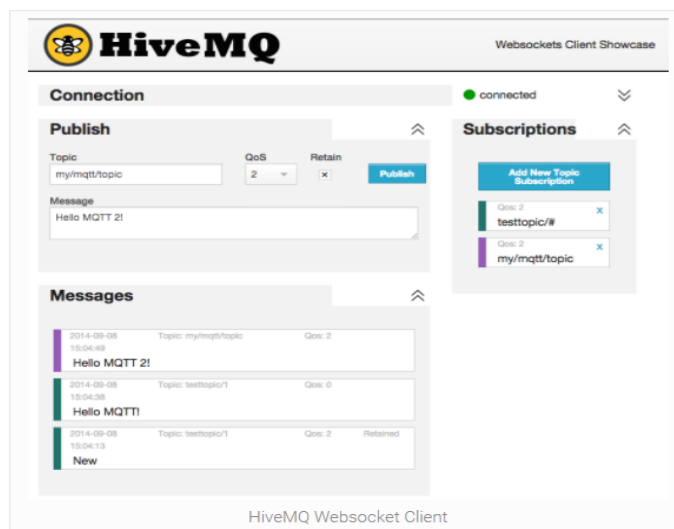
##### Setup local instance

you can always setup your private broker if you like.

[Download HiveMQ](#)

#### HiveMQ MQTT Websocket Client

In order to make it super simple to send your first MQTT messages, we have built a browser-based MQTT client that is easy to use and preconfigured with the HiveMQ public broker. So you can start right away to send and receive messages (See our [MQTT Essentials](#) on how MQTT works and how you can send messages).



## VI) Programación en Arduino

Vamos a mostrar ejemplos de los métodos y trozos de código más relevantes que han sido usados para programar los dos circuitos de las dos farolas respectivamente.

Para más detalle, en esta misma entrega del proyecto se adjuntan los códigos de Arduino utilizados.

### Inclusión de Librerías y Declaración de variables Globales:

```
#include "ESP8266WiFi.h"
#include "PubSubClient.h"

// WiFi parameters to be configured
const char* ssid = "Aquaris M5";
const char* password = "calimero79";

// Variables Globales
int lightPin = 0; // A0 pin analógico de entrada para el LDR
int led = 15; // D8 donde se conecta el led

int valorLDR = 0; // Valor que recibiremos de los sensores
int sum = 0; // Contador de prueba
```

### Conexión Wifi:

```
// Conexión con la red wifi
void setup_wifi()
{

  WiFi.begin(ssid, password);

  // while wifi not connected yet, print '.'
  // then after it connected, get out of the loop
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // print a new line, then print WiFi connected and the IP address
  Serial.println("");
  Serial.println("WiFi connected");
  // Print the IP address
  Serial.println(WiFi.localIP());
  delay(1000);
}
```

## ReConexión al Servicio Mqtt:

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266 Client")) {
      Serial.println("connected");
      // ... and subscribe to topic
      client.subscribe("intensidadtarik");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

```
void reconnect(){
|
while(!client.connected()){
  Serial.print("Esperando para conectar..");

  //Creamos un usuario aleatorio cada vez que nos reconectamos, por si el servicio nos echa fuera
  //por estar conectados muchas veces con el mismo usuario

  String clientId= "3b8bb6b902604c8e8cff2625ebea40f5";
  clientId +=String(random(0xffff), HEX)+ clientId;

  //Nos conectamos con el método connect. Si nos devuelve TRUE imprimimos conectar
  if(client.connect(clientId.c_str())){
    Serial.println("connected");
  }

  //Si el método connect nos devuelve FALSE, intentamos reconectar en unos 5 segundos
  else{
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds, usuario"+clientId);
    delay(5000);
  }

}
```

Método para la suscripción al servicio Mqtt:

```
void callback(char* topic, byte* payload, unsigned int length) {  
  Serial.print("Message arrived [");  
  Serial.print(topic);  
  Serial.print("] ");  
  for (int i=0;i<length;i++) {  
    char receivedChar = (char)payload[i];  
    //Serial.print(receivedChar);  
    someInt= receivedChar - '0';  
    Serial.print(someInt);  
  
  }  
  analogWrite(ledPin,someInt);  
  delay(2000);  
}
```

## VII) Instalación de Node-RED

La instalación de node-RED es muy sencilla y puede instalarse tanto para Windows como para Linux. Toda la información de los pormenores de la instalación vienen descritos en este enlace: <https://nodered.org/docs/getting-started/installation>

Para nuestro proyecto hemos instalado esta herramienta en un equipo Windows de la siguiente manera:

- a) Abrimos la consola de comandos y escribimos ejecutamos lo siguiente:

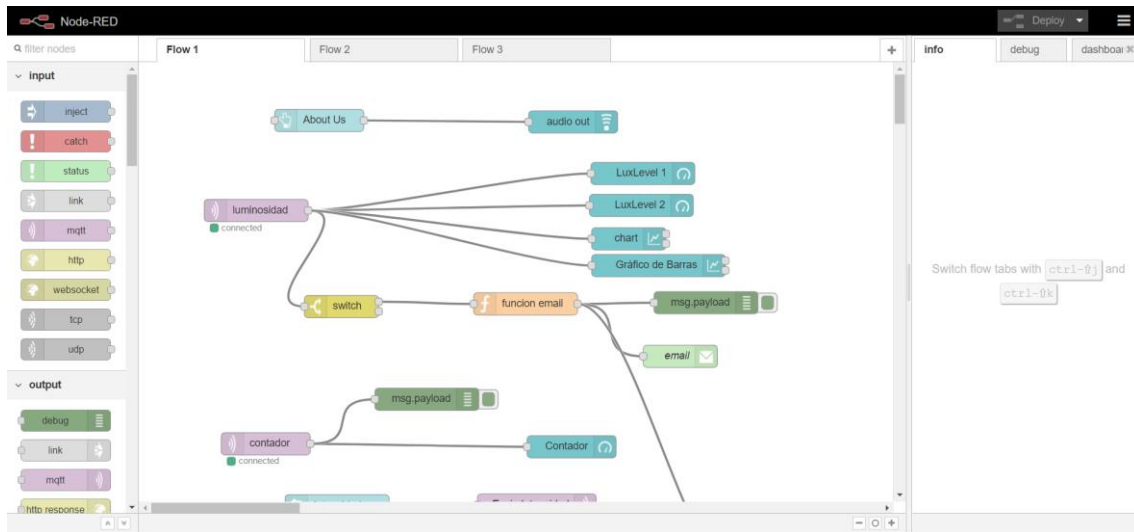
```
npm install -g -unsafe-perm node-red
```

En algún punto de la instalación puede que se requiera o no la instalación de algún complemento, pero no hay ningún problema, la misma aplicación especificará el comando a utilizar.

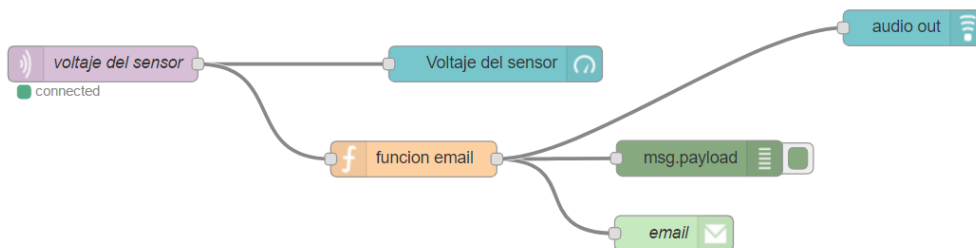
- b) Y para agregarle el complemento de DashBoard debemos ejecutar en el directorio raíz de la aplicación lo siguiente:

```
npm install node-red-dashboard
```

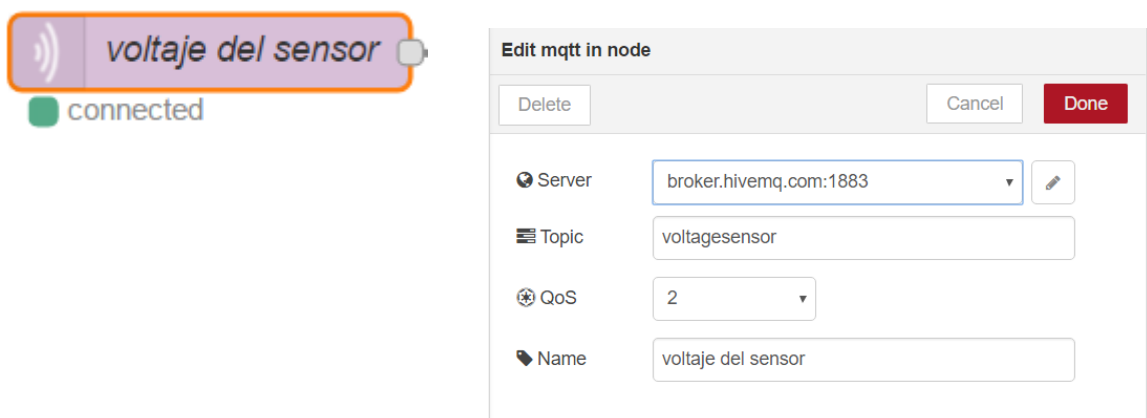
## VIII) Recepción, Emisión, Representación y Monitorización de los Datos.



Vamos a centrarnos en una sola parte de nuestro diagrama para ver de manera general cómo se trabaja en esta plataforma a nivel de recepción, envío y representación de la información.

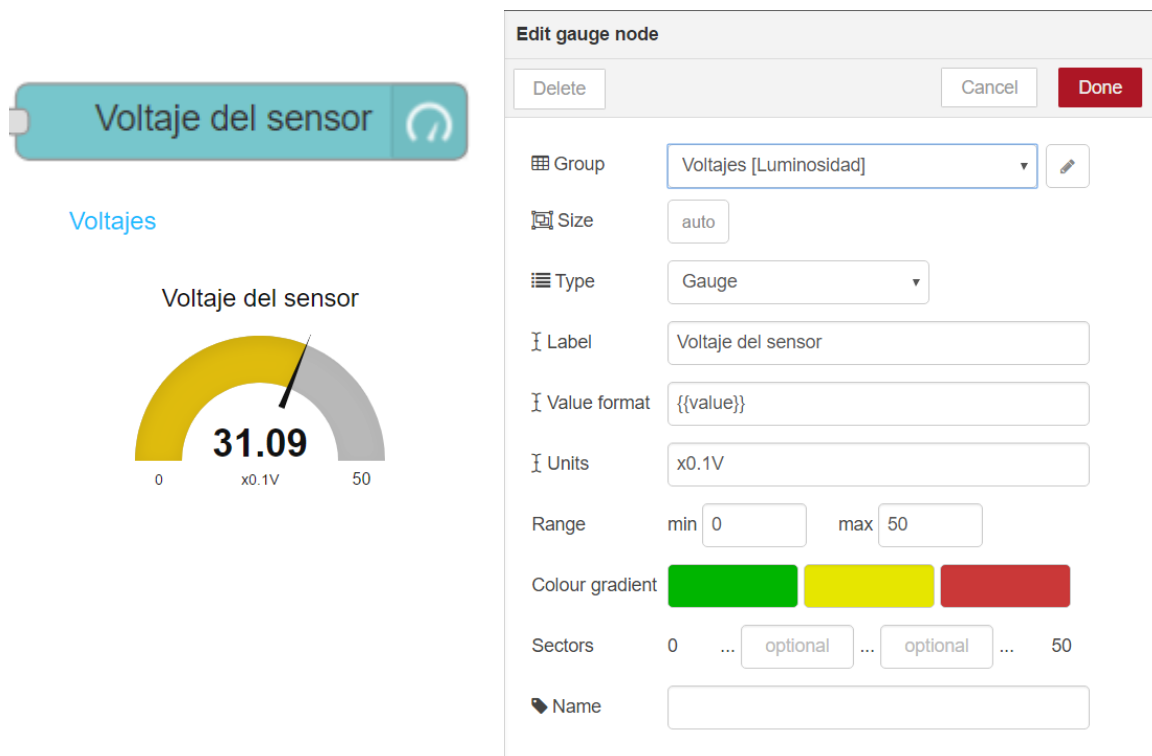


Aquí vemos los siguientes nodos conectados entre siguiendo un flujo de datos que empieza en el nodo de recepción “Voltaje del Sensor”. Este primer nodo recibe por Mqtt el mensaje enviado desde nuestro NODEMCU a través del Servidor bróker.hivemq.com y el puerto 1883 con el Topic de “voltagesensor”. Lo que hace con esa información es transmitirla a los siguientes nodos que están por delante.





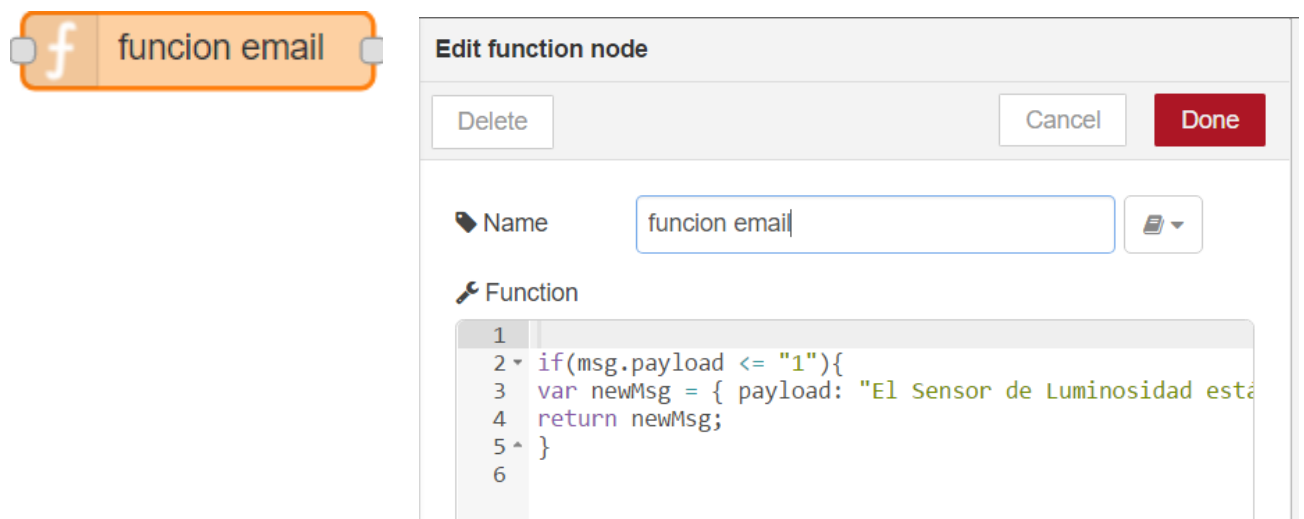
El otro nodo con la misma etiqueta “Voltaje del Sensor” es de tipo gráfico. Lo que hace es mostrar a través de un contador con aguja la cantidad que le llega.



The image shows a visual representation of a gauge node on the left and its configuration window on the right. The gauge node is labeled "Voltaje del sensor" and displays a value of 31.09. The configuration window, titled "Edit gauge node", includes the following settings:

- Group: Voltajes [Luminosidad]
- Size: auto
- Type: Gauge
- Label: Voltaje del sensor
- Value format: {{value}}
- Units: x0.1V
- Range: min 0, max 50
- Colour gradient: Green, Yellow, Red
- Sectors: 0, optional, optional, 50
- Name: (empty)

Los nodos tipo función tienen la característica de poder asignarle condiciones al parámetro de entrada que le entra y transmitir otro mensaje en la salida o decidir si el mensaje continúa su camino. Se programan en función de lo que nosotros estipulemos. En este caso hemos decidido que si el voltaje es inferior a 1 unidad, significa que el sensor esta inhoperativo o inexistente y en consecuencia transmitirá un mensaje a los siguientes nodos.

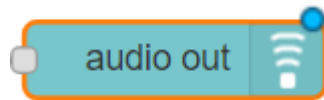


The image shows a visual representation of a function node on the left and its configuration window on the right. The function node is labeled "funcion email". The configuration window, titled "Edit function node", includes the following settings:

- Name: funcion email
- Function: 

```
1  
2 if(msg.payload <= "1"){  
3   var newMsg = { payload: "El Sensor de Luminosidad está  
4   return newMsg;  
5 }  
6
```

Y por último si logra salir el mensaje de emergencia a los siguientes nodos, tendremos por un lado un envío de un mail al técnico de mantenimiento y una alerta de voz en nuestra plataforma que nos redactará el mensaje automáticamente en el idioma que elijamos. Cabe destacar que tendremos que usar una cuenta de correo electrónico de apoyo y autorizar la conexión remota indirecta de nuestro servicio.



**Edit audio out node**

Delete

Cancel

Done

Group

Lampara 2 [Luminosidad] ▼

TTS Voice

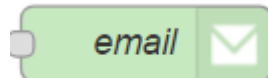
0 : Microsoft Helena Desktop - Spanish (Spain) (es-ES) -- ▼

☐

 Play audio when window not in focus.

Name

Name



**Edit e-mail node**

Delete

Cancel

Done

To

timlahi@yahoo.es

Server

smtp.gmail.com

Port

465

☒ Use secure connection.

Userid

tir403@inlumine.ual.es

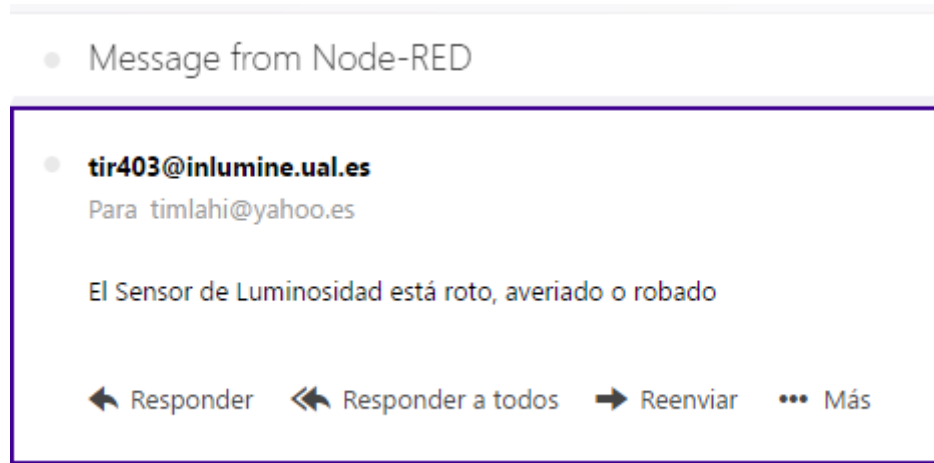
Password

.....

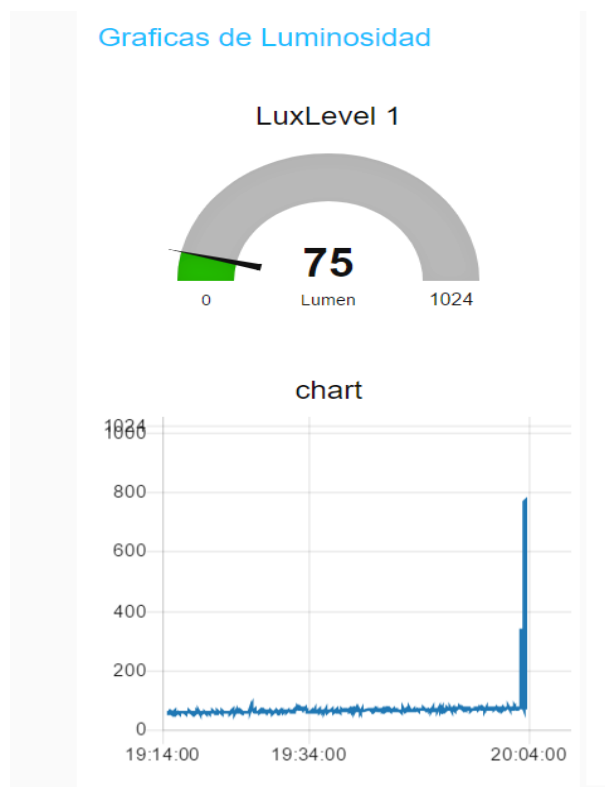
Name

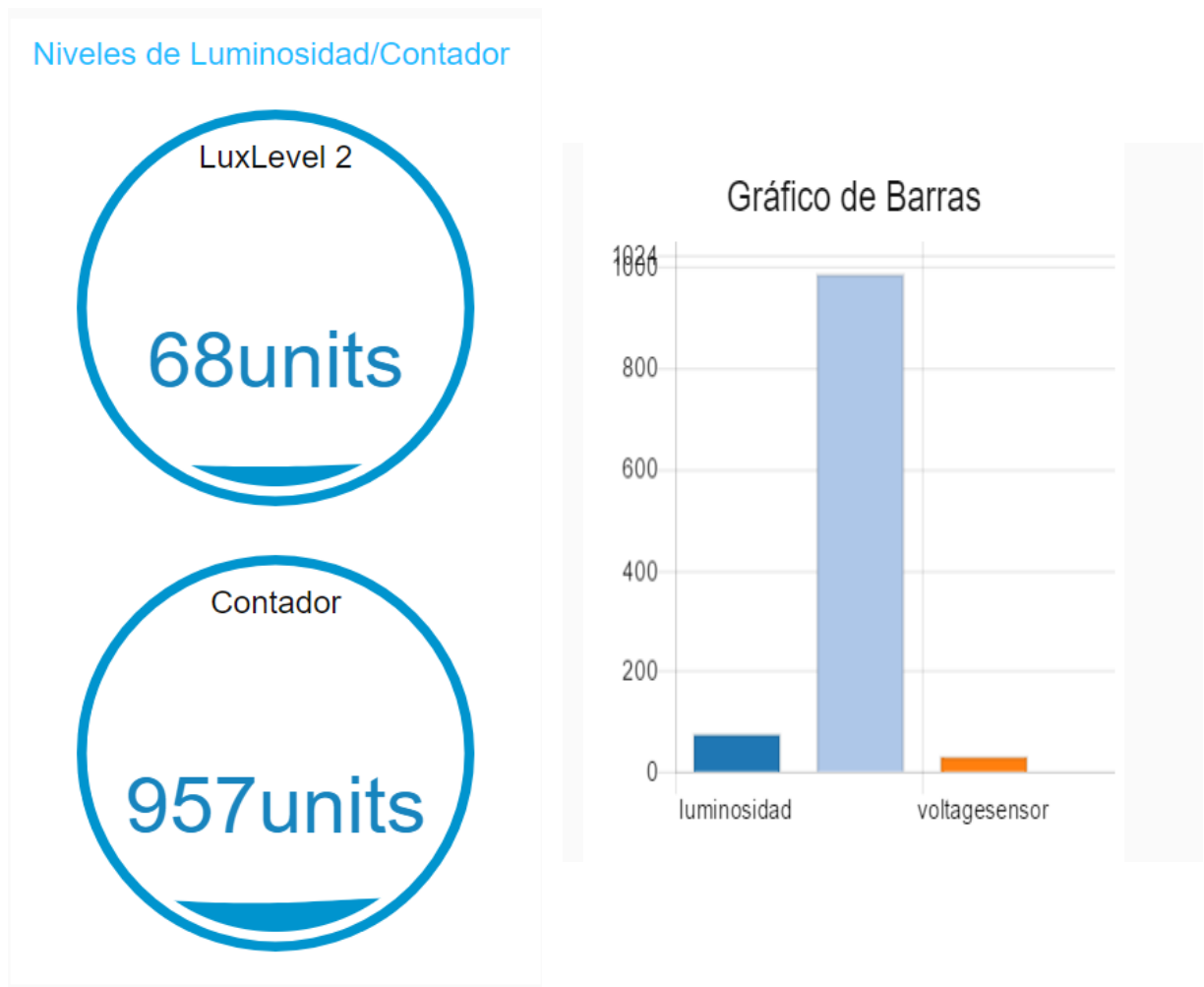
email

Finalmente podemos observar que si en un momento dado quitamos algún cable de los que están conectados a nuestro sensor, nos llegará un correo advirtiéndonos de esa incidencia.



Siguiendo prácticamente los mismos pasos hemos hecho pruebas con diferentes gráficas.





Para el envío de información desde nuestra plataforma a nuestro circuito hemos seguido los mismos pasos pero en vez de usar un nodo de tipo recepción hemos usado uno de tipo emisión. Hemos usado un nodo tipo Slider y un nodo de emisión para enviar los datos.



Cada vez que movemos el slider varía en proporción la intensidad del LED.

## IX) Problemas Encontrados en el Desarrollo del Proyecto

Realmente todo ha ido sobre ruedas exceptuando algunos puntos en los que nos hemos atrancado más bien debido al desconocimiento en profundidad a nivel físico del dispositivo que a la hora de programar dicho dispositivo. Por ejemplo:

- Un error cometido al principio fue conectar nuestro led al pin D0. Lo que no sabíamos era que dicho pin se comparte con la tarjeta de red para la comunicación, por lo que, a la hora de transmitir los datos pertinentes, se enviaban menos de diez mensajes al servidor y después el sistema se bloqueaba. La solución fue simplemente cambiar a otro pin y se acabó el problema
- Intentar medir la diferencia de potencial entre las conexiones del LED. No nos dimos cuenta de que eran pines de salida y no de entrada. La solución era desviar con una resistencia dichas conexiones a unos puertos de entrada y a partir de ahí se podría calcular el voltaje de esos dos puntos. No se ha implementado al carecer de la resistencia precisa para dicha acción.
- Problemas para conectar a desde la red de la universidad ya que hay puertos que no están operativos. Debemos conectarnos o a la wifi de nuestra casa o compartiendo wifi desde el móvil.

Si ha habido algún problema más no ha tenido la suficiente relevancia como para plasmarlo en esta memoria.

## X) Conclusiones Finales

Este pequeño proyecto nos ha dado pie a plantearnos poder crear muchas más aplicaciones en el campo de IoT, mucho más complejas y de manera realmente sencilla y dinámica. La asignatura brinda un abanico de posibilidades inmensa. No solo nos hemos percatado de esto en el desarrollo de nuestro proyecto, si no que hemos visto el avance en paralelo de los proyectos de los demás grupos de compañeros, que nos han parecido también muy interesantes.